OpenShift Container Platform 4.9

설치

OpenShift Container Platform 클러스터 설치 및 구성

Last Updated: 2023-07-17

OpenShift Container Platform 4.9 설치

OpenShift Container Platform 클러스터 설치 및 구성

법적 공지

초록

이 문서는 OpenShift Container Platform 설치 및 일부 구성 프로세스에 대해 자세히 설명합니다.

차례

1장. OPENSHIFT CONTAINER PLATFORM 설치 프로그램 개요

1.1. OPENSHIFT CONTAINER PLATFORM 설치 프로그램 개요

OpenShift Container Platform 설치 프로그램에서는 유연성을 제공합니다. 설치 프로그램을 사용하면 설치 프로그램이 프로비저닝하고 클러스터가 유지보수하는 인프라에 클러스터를 배포하거나 준비하고 유지보수하는 인프라에 클러스터를 배포할 수 있습니다.

이 두 가지 기본 유형의 OpenShift Container Platform 클러스터는 종종 설치 관리자 프로비저닝 인프라 클러스터 및 사용자 프로비저닝 인프라 클러스터라고 합니다.

두 유형의 클러스터에는 모두 다음과 같은 특징이 있습니다.

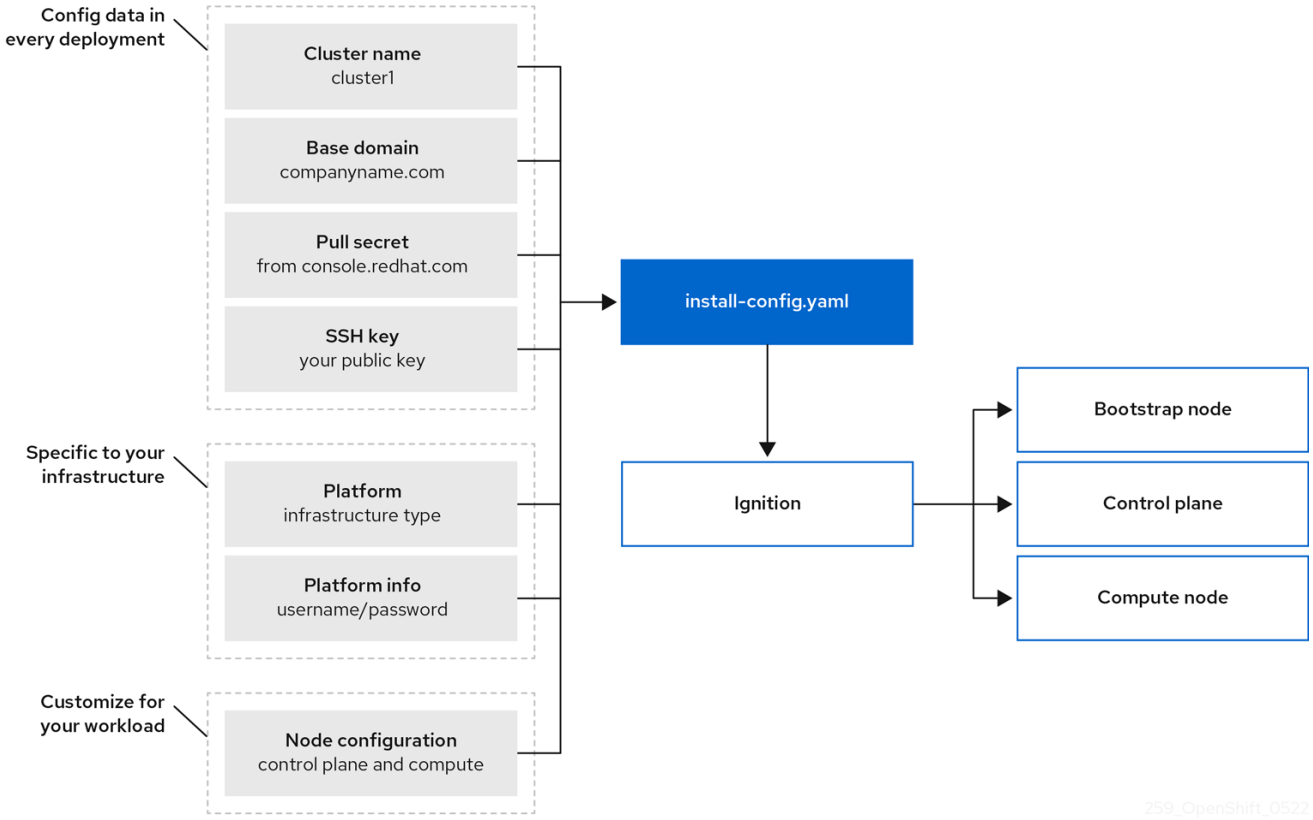
- 기본적으로 단일 장애 지점이 없는 고가용성 인프라를 사용할 수 있습니다.
- 관리자는 적용되는 업데이트 및 해당 시기에 관한 제어 권한을 유지합니다.

동일한 설치 프로그램을 사용하여 두 유형의 클러스터를 모두 배포합니다. 설치 프로그램에서 생성된 주요 자산은 부트스트랩, 마스터 및 작업자 머신의 Ignition 구성 파일입니다. 이 세 가지 구성과 올바르게 구성된 인프라를 사용하면 OpenShift Container Platform 클러스터를 시작할 수 있습니다.

OpenShift Container Platform 설치 프로그램은 일련의 대상 및 종속 항목을 사용하여 클러스터 설치를 관리합니다. 설치 프로그램에는 달성해야 할 대상 세트가 있으며 각 대상에는 종속 항목 세트가 있습니다. 각 대상은 고유한 종속 항목에만 관련되므로 여러 대상이 병렬로 수행되도록 설치 프로그램이 작동할 수 있습니다. 궁극적인 목표는 실행 중인 클러스터입니다. 설치 프로그램은 명령을 실행하지 않고 종속 항목을 충족함으로써 명령을 다시 작성하기 위해 명령을 실행하는 대신 기존 구성 요소를 인식하고 사용할 수 있습니다.

다음 다이어그램에서는 설치 대상 및 종속 항목의 서브 세트를 보여줍니다.

그림 1.1. OpenShift Container Platform 설치 대상 및 종속 항목



설치 후 각 클러스터 머신에서는 RHCOS(Red Hat Enterprise Linux CoreOS)를 운영 체제로 사용합니다. RHCOS는 RHEL(Red Hat Enterprise Linux)의 변경 불가능한 컨테이너 호스트 버전이며 기본적으로 SELinux가 활성화된 RHEL 커널을 제공합니다. 여기에는 Kubernetes 노드 에이전트인 **kubelet** 및 Kubernetes에 최적화된 CRI-O 컨테이너 런타임이 포함됩니다.

OpenShift Container Platform 4.9 클러스터의 모든 컨트롤 플레인 머신은 Ignition이라는 중요한 최초 부팅 프로비저닝 도구를 포함하는 RHCOS를 사용해야 합니다. 이 도구를 사용하면 클러스터가 머신을 구성할 수 있습니다. 운영 체제 업데이트는 Operator가 클러스터 전체에서 롤아웃한 컨테이너 이미지에 임베드된 Atomic OSTree 리포지토리로 제공됩니다. 실제 운영 체제 변경은 rpm-ostree를 사용하여 각 머신에서 원자 작업으로 수행됩니다. 이러한 기술을 사용하면 OpenShift Container Platform에서 전체 플랫폼을 최신 상태로 유지하는 내부 업그레이드를 통해 클러스터의 다른 애플리케이션을 관리하는 것처럼 운영 체제를 관리할 수 있습니다. 이러한 내부 업데이트는 운영 팀의 부담을 줄일 수 있습니다.

모든 클러스터 머신의 운영 체제로 RHCOS를 사용하는 경우 클러스터가 운영 체제를 포함한 구성 요소 및 머신의 모든 측면을 관리합니다. 그러면 설치 프로그램 및 Machine Config Operator만 머신을 변경할 수 있습니다. 설치 프로그램에서는 Ignition 구성 파일을 사용하여 각 머신의 정확한 상태를 설정하고 Machine Config Operator는 설치 후 새 인증서 또는 키 적용과 같은 머신에 대한 추가 변경을 완료합니다.

1.1.1. 설치 프로세스

OpenShift Container Platform 클러스터를 설치할 때 OpenShift Cluster Manager 사이트의 해당 인프라 공급자 페이지에서 설치 프로그램을 다운로드합니다. 이 사이트에서는 다음을 관리합니다.

- 계정용 REST API
- 필수 구성 요소를 얻는 데 사용하는 풀 시크릿인 레지스트리 토큰
- 사용 지표 수집이 용이하도록 클러스터 ID를 Red Hat 계정에 연결하는 클러스터 등록

OpenShift Container Platform 4.9에서 설치 프로그램은 자산 세트에서 일련의 파일 변환을 수행하는 Go 바이너리 파일입니다. 설치 프로그램과 상호 작용하는 방법은 설치 유형에 따라 다릅니다.

- 설치 관리자 프로비저닝 인프라가 있는 클러스터의 경우 인프라 부트스트랩 및 프로비저닝을 직접 수행하는 대신 설치 프로그램에 위임합니다. 설치 프로그램은 클러스터를 지원하는 데 필요한 모든 네트워킹, 머신 및 운영 체제를 생성합니다.
- 클러스터의 인프라를 프로비저닝하고 관리하는 경우 부트스트랩 머신, 네트워킹, 부하 분산, 스토리지 및 개별 클러스터 머신을 포함한 모든 클러스터 인프라 및 리소스를 제공해야 합니다.

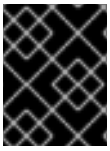
설치하는 동안 **install-config.yaml**이라는 설치 구성 파일, Kubernetes 매니페스트 및 머신 유형에 맞는 Ignition 구성 파일의 세 가지 파일 세트를 사용합니다.

중요

설치 중에 기본 RHCOS 운영 체제를 제어하는 Kubernetes 및 Ignition 구성 파일을 수정할 수 있습니다. 그러나 이러한 오브젝트를 수정한 내용이 적합한지 확인할 수 있는 유효성 검사는 없습니다. 이러한 오브젝트를 수정하면 클러스터가 작동하지 않을 수 있습니다. 이 위험 때문에 문서화된 절차를 따르거나 Red Hat 지원 부서에서 지시하지 않는 한 Kubernetes 및 Ignition 구성 파일 수정은 지원되지 않습니다.

설치 구성 파일은 Kubernetes 매니페스트로 변환된 다음 매니페스트가 Ignition 구성 파일로 래핑됩니다. 설치 프로그램은 이러한 Ignition 구성 파일을 사용하여 클러스터를 생성합니다.

설치 프로그램을 실행할 때 설치 구성 파일이 모두 제거되므로 다시 사용하려는 모든 구성 파일을 백업하십시오.



중요

설치 중에 설정한 매개변수는 수정할 수 없지만 설치 후에는 많은 클러스터 속성을 수정할 수 있습니다.

설치 관리자가 프로비저닝한 인프라를 사용하는 설치 프로세스
기본 설치 유형에서는 설치 관리자 프로비저닝 인프라를 사용합니다. 기본적으로 설치 프로그램은 설치 마법사 역할을 하여 자체적으로 결정할 수 없는 값을 입력하라는 메시지를 표시하고 나머지 매개변수에 대한 적절한 기본값을 제공합니다. 고급 인프라 시나리오를 지원하도록 설치 프로세스를 사용자 정의할 수도 있습니다. 설치 프로그램은 클러스터의 기본 인프라를 프로비저닝합니다.

표준 클러스터 또는 사용자 정의된 클러스터를 설치할 수 있습니다. 표준 클러스터에서는 클러스터를 설치하는 데 필요한 최소 세부 정보를 제공합니다. 사용자 지정 클러스터를 사용하면 컨트롤 플레인에서 사용하는 머신 수, 클러스터가 배포하는 가상 머신 유형 또는 Kubernetes 서비스 네트워크의 CIDR 범위와 같은 플랫폼에 대한 세부 정보를 지정할 수 있습니다.

가능하면 이 기능을 사용하여 클러스터 인프라를 프로비저닝 및 유지보수하지 않아도 됩니다. 다른 모든 환경에서는 설치 프로그램을 사용하여 클러스터 인프라를 프로비저닝하는 데 필요한 자산을 생성합니다.

OpenShift Container Platform은 설치 프로그램에서 프로비저닝한 인프라 클러스터를 통해 운영 체제 자체를 포함하여 클러스터의 모든 측면을 관리합니다. 각 머신은 결합하는 클러스터에서 호스팅되는 리소스를 참조하는 구성으로 부팅됩니다. 이 구성을 사용하면 업데이트가 적용될 때 클러스터가 자체적으로 관리될 수 있습니다.

사용자 프로비저닝 인프라를 사용하는 설치 프로세스
제공하는 인프라에 OpenShift Container Platform도 설치할 수 있습니다. 설치 프로그램을 사용하여 클러스터 인프라를 프로비저닝하고 클러스터 인프라를 생성한 다음 제공한 인프라에 클러스터를 배포하는 데 필요한 자산을 생성합니다.

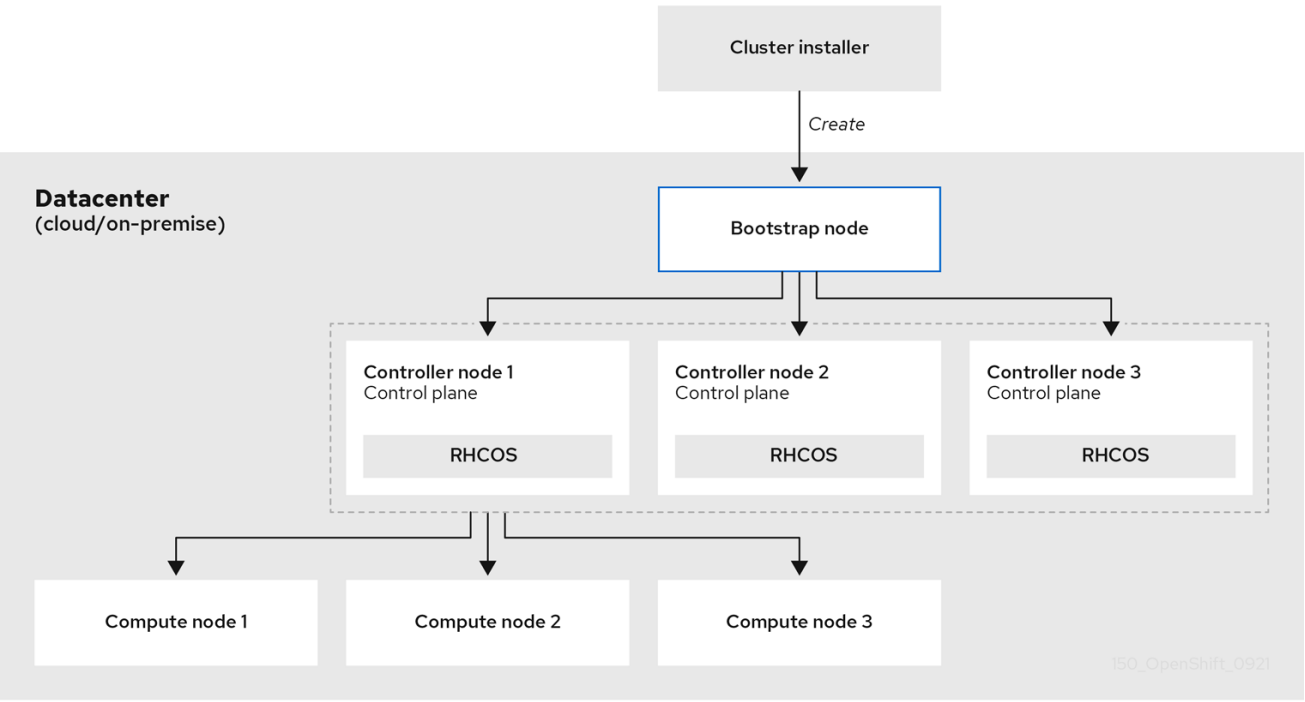
설치 프로그램이 프로비저닝한 인프라를 사용하지 않는 경우 다음을 포함하여 클러스터 자원을 직접 관리하고 유지보수해야 합니다.

- 클러스터를 구성하는 컨트롤 플레인 및 컴퓨팅 머신의 기본 인프라
- 로드 밸런서
- DNS 레코드 및 필수 서브넷을 포함한 클러스터 네트워킹
- 클러스터 인프라 및 애플리케이션용 스토리지

클러스터가 사용자 프로비저닝 인프라를 사용하는 경우 RHEL 컴퓨팅 머신을 클러스터에 추가할 수 있습니다.

설치 프로세스 세부사항
클러스터의 각 머신에는 프로비저닝 시 클러스터에 대한 정보가 필요하므로 OpenShift Container Platform은 초기 구성 중에 임시 *부트스트랩* 머신을 사용하여 필요한 정보를 영구 컨트롤 플레인에 제공합니다. 클러스터 생성 방법을 설명하는 Ignition 구성 파일을 사용하여 부팅됩니다. 부트스트랩 시스템은 컨트롤 플레인을 구성하는 컨트롤 플레인 시스템을 생성합니다. 그런 다음 컨트롤 플레인 시스템에서는 작업자 머신이라고도 하는 컴퓨팅 머신을 만듭니다. 다음 그림은 이 프로세스를 보여줍니다.

그림 1.2. 부트스트랩, 컨트롤 플레인 및 컴퓨팅 시스템 생성



클러스터 머신이 초기화되면 부트스트랩 머신이 손상됩니다. 모든 클러스터에서는 부트스트랩 프로세스를 사용하여 클러스터를 초기화하지만, 클러스터의 인프라를 프로비저닝하는 경우 많은 단계를 수동으로 완료해야 합니다.

중요

- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

클러스터 부트스트랩에는 다음 단계가 포함됩니다.

1. 부트스트랩 머신이 부팅되고 컨트롤 플레인 머신을 부팅하는 데 필요한 원격 리소스 호스팅이 시작됩니다. (인프라를 프로비저닝할 경우 수동 조작 필요)
2. 부트스트랩 머신은 단일 노드 etcd 클러스터와 임시 Kubernetes 컨트롤 플레인을 시작합니다.
3. 컨트롤 플레인 머신은 부트스트랩 머신에서 원격 리소스를 가져오고 부팅을 완료합니다. (인프라를 프로비저닝할 경우 수동 조작 필요)
4. 임시 컨트롤 플레인은 프로덕션 컨트롤러 플레인을 프로덕션 컨트롤 플레인 머신에 예약합니다.

5. CVO(Cluster Version Operator)가 온라인 상태가 되어 etcd Operator를 설치합니다. etcd Operator는 모든 컨트롤 플레인 노드에서 etcd를 확장합니다.
6. 임시 컨트롤 플레인이 종료되고 제어를 프로덕션 컨트롤 플레인에 전달합니다.
7. 부트스트랩 머신은 OpenShift Container Platform 구성 요소를 프로덕션 컨트롤 플레인에 주입합니다.
8. 설치 프로그램이 부트스트랩 머신을 종료합니다. (인프라를 프로비저닝할 경우 수동 조작 필요)
9. 컨트롤 플레인이 컴퓨팅 노드를 설정합니다.
10. 컨트롤 플레인은 일련의 Operator 형태로 추가 서비스를 설치합니다.

이 부트스트랩 프로세스의 결과는 OpenShift Container Platform 클러스터가 실행 중인 것입니다. 그런 다음 클러스터는 지원되는 환경에서 컴퓨팅 머신 생성을 포함하여 일상적인 작업에 필요한 나머지 구성 요소를 다운로드하고 구성합니다.

1.1.2. 설치 후 노드 상태 확인

OpenShift Container Platform 설치는 다음과 같은 설치 상태 점검에 성공하면 완료됩니다.

- 프로비저닝 호스트는 OpenShift Container Platform 웹 콘솔에 액세스할 수 있습니다.
- 모든 컨트롤 플레인 노드가 준비되었습니다.
- 모든 클러스터 Operator를 사용할 수 있습니다.

참고

설치가 완료되면 작업자 노드를 담당하는 특정 클러스터 Operator가 모든 작업자 노드를 지속적으로 프로비저닝하려고 합니다. 모든 작업자 노드가 **READY** 로 보고하는 데 다소 시간이 걸릴 수 있습니다. 베어 메탈에 설치하는 경우 작업자 노드의 문제를 해결하기 전에 최소 60분 정도 기다립니다. 다른 모든 플랫폼에 설치하는 경우 작업자 노드의 문제를 해결하기 전에 최소 40분 동안 기다립니다. 작업자 노드를 담당하는 클러스터 Operator의 **DEGRADED** 상태는 노드 상태가 아닌 Operator의 자체 리소스에 따라 다릅니다.

설치가 완료되면 다음 단계를 사용하여 클러스터의 노드 상태를 계속 모니터링할 수 있습니다.

사전 요구 사항

- 설치 프로그램이 터미널에서 성공적으로 확인됩니다.

프로세스

1. 모든 작업자 노드의 상태를 표시합니다.

```
$ oc get nodes
```

출력 예

```
NAME                         STATUS   ROLES    AGE   VERSION
example-compute1.example.com   Ready    worker   13m   v1.21.6+bb8d50a
example-compute2.example.com   Ready    worker   13m   v1.21.6+bb8d50a
example-compute4.example.com   Ready    worker   14m   v1.21.6+bb8d50a
```

```
example-control1.example.com   Ready   master   52m   v1.21.6+bb8d50a
example-control2.example.com   Ready   master   55m   v1.21.6+bb8d50a
example-control3.example.com   Ready   master   55m   v1.21.6+bb8d50a
```

2. 모든 작업자 머신 노드의 단계를 표시합니다.

```
$ oc get machines -A
```

출력 예

```
NAMESPACE               NAME                              PHASE     TYPE   REGION   ZONE   AGE
openshift-machine-api   example-zbbt6-master-0            Running                          95m
openshift-machine-api   example-zbbt6-master-1            Running                          95m
openshift-machine-api   example-zbbt6-master-2            Running                          95m
openshift-machine-api   example-zbbt6-worker-0-25bhp      Running                          49m
openshift-machine-api   example-zbbt6-worker-0-8b4c2      Running                          49m
openshift-machine-api   example-zbbt6-worker-0-jkbqt      Running                          49m
openshift-machine-api   example-zbbt6-worker-0-qrl5b      Running                          49m
```

추가 리소스

- 설치 후
- 설치 검증

설치 범위

OpenShift Container Platform 설치 프로그램의 범위는 의도적으로 한정됩니다. 단순성과 성공을 보장하도록 설계되었습니다. 설치가 완료된 후 많은 추가 구성 작업을 완료할 수 있습니다.

추가 리소스

- OpenShift Container Platform 구성 리소스에 대한 자세한 내용은 사용 가능한 클러스터 사용자 정의를 참조하십시오.

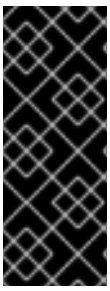
1.2. OPENSHIFT CONTAINER PLATFORM 클러스터에서 지원되는 플랫폼

OpenShift Container Platform 4.9에서는 다음 플랫폼에서 설치 관리자 프로비저닝 인프라를 사용하는 클러스터를 설치할 수 있습니다.

- AWS(Amazon Web Services)
- GCP(Google Cloud Platform)
- Microsoft Azure
- RHOSP(Red Hat OpenStack Platform) 버전 16.1 및 16.2
  - 최신 OpenShift Container Platform 릴리스는 최신 RHOSP 긴 수명 릴리스 및 중간 릴리스를 모두 지원합니다. 완전한 RHOSP 릴리스 호환성에 대해서는 RHOSP의 OpenShift Container Platform on RHOSP 지원 매트릭스를 참조하십시오.
- RHV(Red Hat Virtualization)
- VMware vSphere

- AWS의 VMware Cloud (VMC)
- 베어 메탈

이러한 클러스터의 경우, 설치 프로세스를 실행하는 컴퓨터를 포함한 모든 머신은 플랫폼 컨테이너의 이미지를 가져오고 원격 분석 데이터를 Red Hat에 제공하기 위해 인터넷에 직접 액세스할 수 있어야 합니다.



중요

설치 후 다음 변경 사항은 지원되지 않습니다.

- 클라우드 공급자 플랫폼 혼합
- 클러스터가 설치된 플랫폼과 다른 플랫폼의 영구 스토리지 프레임워크 사용과 같은 클라우드 공급자 구성 요소 혼합

OpenShift Container Platform 4.9에서는 다음 플랫폼에서 사용자 프로비저닝 인프라를 사용하는 클러스터를 설치할 수 있습니다.

- AWS
- Azure
- Azure Stack Hub
- GCP
- RHOSP 버전 16.1 및 16.2
- RHV
- VMware vSphere
- AWS의 VMware Cloud
- 베어 메탈
- IBM Z 또는 LinuxONE
- IBM Power

플랫폼의 지원 사례에 따라 사용자 프로비저닝 인프라에 을 설치하면 전체 인터넷 액세스가 가능한 머신을 실행하거나, 클러스터를 프록시 뒤에 배치하거나, *제한된 네트워크 설치*를 수행할 수 있습니다. 제한된 네트워크 설치에서는 클러스터를 설치하는 데 필요한 이미지를 다운로드하여 미러 레지스트리에 배치한 다음 해당 데이터를 사용하여 클러스터를 설치할 수 있습니다. vSphere 또는 베어 메탈 인프라에서 제한된 네트워크 설치로 플랫폼 컨테이너의 이미지를 가져오려면 인터넷에 액세스해야 하지만, 클러스터 컴퓨터는 인터넷에 직접 액세스할 필요가 없습니다.

다른 플랫폼의 통합 테스트에 대한 자세한 내용은 OpenShift Container Platform 4.x 통합 테스트 페이지를 참조하십시오.

추가 리소스

- 지원되는 각 플랫폼에서 사용할 수 있는 설치 유형에 대한 자세한 내용은 다른 플랫폼의 지원 설치 방법에서 참조하십시오.

- 설치 방법 선택 및 필수 리소스 준비에 대한 정보는 클러스터 설치 방법 선택 및 사용자를 위한 준비에서 참조하십시오.

2장. 클러스터 설치 방법 선택 및 사용자를 위한 준비

OpenShift Container Platform을 설치하기 전에 실행할 종류의 설치 프로세스를 결정하고 사용자를 위해 클러스터를 준비하는 데 필요한 모든 리소스가 있는지 확인합니다.

2.1. 클러스터 설치 유형 선택

OpenShift Container Platform 클러스터를 설치하기 전에 실행할 최상의 설치 지침을 선택해야 합니다. 최상의 옵션을 선택하기 위해 다음 질문에 대한 답변을 고려하십시오.

2.1.1. OpenShift Container Platform 클러스터를 직접 설치 및 관리하시겠습니까?

OpenShift Container Platform을 직접 설치하고 관리하려면 다음 플랫폼에 설치할 수 있습니다.

- AWS(Amazon Web Services)
- Microsoft Azure
- Microsoft Azure Stack Hub
- GCP(Google Cloud Platform)
- Red Hat OpenStack Platform (RHOSP)
- RHV(Red Hat Virtualization)
- IBM Z 및 LinuxONE
- IBM Z 및 LinuxONE for Red Hat Enterprise Linux (RHEL) KVM
- IBM Power
- VMware vSphere
- AWS의 VMware Cloud (VMC)
- 베어 메탈 또는 기타 플랫폼과 무관한 인프라

온프레미스 하드웨어 및 클라우드 호스팅 서비스에 모두 OpenShift Container Platform 4 클러스터를 배포할 수 있지만 클러스터의 모든 머신은 동일한 데이터 센터 또는 클라우드 호스팅 서비스에 있어야 합니다.

OpenShift Container Platform을 사용하지만 클러스터를 직접 관리하지 않기 위해 여러 관리 서비스 옵션이 있습니다. Red Hat에서 완전히 관리하는 클러스터를 원하는 경우 OpenShift Dedicated 또는 OpenShift Online를 사용할 수 있습니다. Azure, AWS, IBM Cloud 또는 Google Cloud에서 OpenShift를 관리 서비스로 사용할 수도 있습니다. 관리 서비스에 대한 자세한 내용은 OpenShift 제품 페이지를 참조하십시오. 클라우드 가상 머신을 가상 베어 메탈으로 사용하여 OpenShift Container Platform 클러스터를 설치하는 경우 해당 클라우드 기반 스토리지는 지원되지 않습니다.

2.1.2. OpenShift Container Platform 3을 사용한 적이 있으며 OpenShift Container Platform 4를 사용하고 싶으신가요?

OpenShift Container Platform 3을 사용했으며 OpenShift Container Platform 4를 시도하려면 OpenShift Container Platform 4가 어떻게 다른지 이해해야 합니다. OpenShift Container Platform 4는 Kubernetes 애플리케이션과 플랫폼이 실행되는 운영 체제인 RHCOS(Red Hat Enterprise Linux CoreOS)를 함께 원활

하게 패키지, 배포, 관리하는 Operator를 만듭니다. OpenShift Container Platform을 설치하기 위해 머신을 배포하고 운영 체제를 구성하는 대신 RHCOS 운영 체제는 OpenShift Container Platform 클러스터의 통합된 부분입니다. 클러스터 머신의 운영 체제를 OpenShift Container Platform의 설치 프로세스의 일부로 배포합니다. OpenShift Container Platform 3 및 OpenShift Container Platform 4 비교 를 참조하십시오.

OpenShift Container Platform 클러스터 설치 프로세스의 일부로 머신을 프로비저닝해야 하므로 OpenShift Container Platform 3 클러스터를 OpenShift Container Platform 4로 업그레이드할 수 없습니다. 대신 새로운 OpenShift Container Platform 4 클러스터를 생성하고 OpenShift Container Platform 3 워크로드를 마이그레이션해야 합니다. 마이그레이션에 대한 자세한 내용은 OpenShift 마이그레이션 모범 사례를 참조하십시오. OpenShift Container Platform 4로 마이그레이션해야 하므로 모든 유형의 프로덕션 클러스터 설치 프로세스를 사용하여 새 클러스터를 생성할 수 있습니다.

2.1.3. 클러스터에서 기존 구성 요소를 사용하시겠습니까?

운영 체제는 OpenShift Container Platform에 통합되므로 OpenShift Container Platform용 설치 프로그램이 모든 인프라를 가동하도록 하는 것이 더 쉬워집니다. 이는 *설치 관리자가 프로비저닝한 인프라* 설치라고 합니다. 이 유형의 설치에서는 클러스터에 일부 기존 인프라를 제공할 수 있지만, 설치 프로그램은 클러스터가 처음에 필요한 모든 머신을 배포합니다.

AWS, Azure, GCP 또는 AWS에서 VMC의 클러스터 또는 해당 기본 머신에 사용자 정의를 지정하지 않고 설치 관리자가 프로비저닝한 인프라 클러스터를 배포할 수 있습니다. 이러한 설치 방법은 프로덕션이 가능한 OpenShift Container Platform 클러스터를 배포하는 가장 빠른 방법입니다.

클러스터 머신의 인스턴스 유형과 같이 설치 관리자가 프로비저닝한 인프라 클러스터에 대한 기본 구성을 수행해야 하는 경우 AWS, Azure, GCP, AWS의 VMC에 대한 설치를 사용자 지정할 수 있습니다.

설치 관리자 프로비저닝 인프라 설치의 경우 기존 AWS의 VPC, Azure의 vNet 또는 GCP의 VPC를 사용할 수 있습니다. AWS, Azure, GCP, 또는 AWS의 VMC에서 클러스터가 사용자 환경의 기존 IP 주소 할당과 공존하고 기존 MTU 및 VXLAN 구성과 통합할 수 있으므로 네트워킹 인프라의 부분을 재사용할 수도 있습니다. 이러한 클라우드에 기존 계정과 인증 정보가 있는 경우 다시 사용할 수 있지만 계정을 수정하여 OpenShift Container Platform 클러스터를 설치하는 데 필요한 권한이 있어야 할 수 있습니다.

설치 관리자 프로비저닝 인프라 방법을 사용하여 하드웨어에 RHOSP, RHOSP with Kuryr, RHOSP on SR-IOV, RHV, vSphere, 베어 메탈 의 적절한 머신 인스턴스를 생성할 수 있습니다. 또한 vSphere 의 경우 AWS의 VMC를 설치하는 동안 추가 네트워크 매개변수를 사용자 지정할 수도 있습니다.

광범위한 클라우드 인프라를 재사용하려면 *사용자가 프로비저닝한 인프라* 설치를 완료할 수 있습니다. 이러한 설치를 통해 설치 프로세스 중에 클러스터에 필요한 머신을 수동으로 배포합니다. AWS,Azure, Azure Stack Hub,GCP 또는 AWS 에서 VMC에서 사용자가 프로비저닝한 인프라 설치를 수행하는 경우 제공된 템플릿을 사용하여 모든 필수 구성 요소를 유지할 수 있습니다. GCP에서 공유 VPC를 재사용할 수도 있습니다. 그렇지 않으면 공급자와 무관 한 설치 방법을 사용하여 다른 클라우드에 클러스터를 배포할 수 있습니다.

기존 하드웨어에서 사용자가 프로비저닝한 인프라 설치를 완료할 수도 있습니다. RHOSP, RHOSP on SR-IOV,RHV,IBM Z or LinuxONE ,IBM Z 또는 LinuxONE with RHEL KVM ,IBM Power 또는 vSphere 를 사용하는 경우 특정 설치 지침을 사용하여 클러스터를 배포합니다. 기타 지원되는 하드웨어를 사용하는 경우 베어 메탈 설치 절차를 따르십시오. RHOSP,vSphere,AWS의 VMC 및 베어 메탈 과 같은 일부 플랫폼의 경우 설치 중에 추가 네트워크 매개변수를 사용자 지정할 수도 있습니다.

2.1.4. 클러스터에 추가 보안이 필요하십니까?

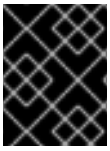
사용자가 프로비저닝한 설치 메서드를 사용하는 경우 클러스터에 대한 프록시를 구성할 수 있습니다. 지침은 각 설치 절차에 포함됩니다.

퍼블릭 클라우드의 클러스터가 외부에서 끝점을 노출하지 못하도록 하려면 AWS, Azure 또는 GCP에 설치 관리자가 프로비저닝한 인프라가 있는 프라이빗 클러스터를 배포할 수 있습니다.

연결이 끊긴 또는 제한된 네트워크 클러스터와 같은 인터넷에 대한 액세스가 제한된 클러스터를 설치해야 하는 경우 설치 패키지를 미러링하고 여기에서 클러스터를 설치할 수 있습니다. AWS, GCP, IBM Z 또는 LinuxONE, IBM Z 또는 LinuxONE with RHEL KVM , IBM Power, vSphere, AWS의 VMC, 또는 베어 메탈의 제한된 네트워크에 사용자가 프로비저닝한 인프라 설치는 상세한 지침을 따르십시오. AWS, GCP, AWS의 VMC, RHOSP, RHV, vSphere에 대한 자세한 지침에 따라 설치 관리자 프로비저닝 인프라를 사용하여 클러스터를 제한된 네트워크에 설치할 수도 있습니다.

AWS GovCloud 리전,AWS China region 또는 Azure government 리전에 클러스터를 배포해야 하는 경우 설치 관리자 프로비저닝 인프라 설치 중에 해당 사용자 지정 리전을 구성할 수 있습니다.

설치 중에 FIPS 검증 / 진행 중인 모듈 암호화 라이브러리 를 사용하도록 클러스터 머신을 구성할 수도 있습니다.



중요

FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.

2.2. 설치 후 사용자용 클러스터 준비

사용자가 클러스터에 액세스하기 전에 클러스터를 설치하는 데는 일부 구성이 필요하지 않습니다. 클러스터를 구성하는 Operator를 사용자 정의하여 클러스터 자체를 사용자 정의하고 ID 공급자와 같은 다른 필수 시스템과 클러스터를 통합할 수 있습니다.

프로덕션 클러스터의 경우 다음과 같은 통합을 구성해야 합니다.

- 영구 스토리지
- ID 공급자
- 핵심 OpenShift Container Platform 구성 요소 모니터링

2.3. 워크로드를 위한 클러스터 준비

워크로드 요구 사항에 따라 애플리케이션 배포를 시작하기 전에 추가 단계를 수행해야 할 수 있습니다. 예를 들어 애플리케이션 빌드 전략을 지원하기 위해 인프라를 준비한 후 대기 시간이 짧은 워크로드를 프로비저닝하거나 중요한 워크로드를 보호해야 할 수 있습니다. 애플리케이션 워크로드에 대한 모니터링을 구성할 수도 있습니다. Windows 워크로드를 실행하려는 경우 설치 프로세스 중에 OVN-Kubernetes를 사용하여 하이브리드 네트워킹을 활성화해야 합니다. 클러스터를 설치한 후에는 하이브리드 네트워킹을 활성화할 수 없습니다.

2.4. 다른 플랫폼에 지원되는 설치 방법

다른 플랫폼에서 다른 유형의 설치를 수행할 수 있습니다.

참고

다음 표에 표시된 대로 모든 플랫폼에서 모든 설치 옵션이 지원되는 것은 아닙니다. 확인 표시는 옵션이 지원되고 관련 섹션에 대한 링크를 나타냅니다.

표 2.1. 설치 프로그램에서 제공하는 인프라 옵션

	AWS	Azure	GCP	RHOSP	RHOSP on SR-IOV	RHV	베어 메탈	vSphere	VMC	IBM Z	IBM Power
기본	✓	✓	✓			✓	✓	✓	✓		
사용자 정의	✓	✓	✓	✓	✓	✓		✓	✓		
네트워크 사용자 지정	✓	✓	✓	✓				✓	✓		
제한된 네트워크	✓		✓	✓		✓	✓	✓	✓		
프라이빗 클러스터	✓	✓	✓								
기존 가상 사설망	✓	✓	✓								
정부 리전	✓	✓									
중국 지역	✓										

표 2.2. 사용자가 제공하는 인프라 옵션

	AWS	Azure	Azure Stack Hub	GCP	RHOSP	RHOSP on SR-IOV	RHV	베어메탈	vSphere	VMC	IBM Z	IBM Z with RHEL KVM	IBM Power	플랫폼에 구애받지 않음
사용자 지정	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
네트워크 사용자 지정					✓			✓	✓	✓				
제한된 네트워크	✓			✓			✓	✓	✓	✓	✓	✓	✓	

	AWS	Azure	Azure Stack Hub	GCP	RHOSP	RHOSP on SR-IOV	RHV	베어메탈	vSphere	VMC	IBM Z	IBM Z with RHEL KVM	IBM Power	플랫폼에 구애받지 않음
클러스터 프로젝트 외부에서 호스팅되는 공유 VPC				✓										

3장. 연결 해제된 설치를 위한 이미지 미러링

이 섹션의 절차를 사용하여 클러스터가 외부 콘텐츠에 대한 조직의 제어 조건을 충족하는 컨테이너 이미지만 사용하도록 할 수 있습니다. 제한된 네트워크에서 프로비저닝된 인프라에 클러스터를 설치하기 전에 필요한 컨테이너 이미지를 해당 환경에 미러링해야 합니다. 컨테이너 이미지를 미러링하려면 미러링을 위한 레지스트리가 있어야 합니다.

중요

필요한 컨테이너 이미지를 얻으려면 인터넷에 액세스해야 합니다. 이 절차에서는 네트워크와 인터넷에 모두 액세스할 수 있는 미러 호스트에 미러 레지스트리를 배치합니다. 미러 호스트에 액세스할 수 없는 경우 연결이 끊긴 클러스터에 사용할 Operator 카탈로그를 미러링 하여 네트워크 경계를 이동할 수 있는 장치에 이미지를 복사합니다.

3.1. 사전 요구 사항

- 다음 레지스트리 중 하나와 같이 OpenShift Container Platform 클러스터를 호스팅할 위치에 Docker v2-2 를 지원하는 컨테이너 이미지 레지스트리가 있어야 합니다.
  - Red Hat Quay
  - JFrog Artifactory
  - Sonatype Nexus Repository
  - Harbor

  Red Hat Quay에 대한 사용 권한이 있는 경우, 개념 증명 목적 으로 또는 Quay Operator를 사용하여 Red Hat Quay 배포에 대한 설명서를 참조하십시오. 레지스트리를 선택 및 설치하는데 추가 지원이 필요한 경우 영업 담당자 또는 Red Hat 지원팀에 문의하십시오.

- 컨테이너 이미지 레지스트리에 대한 기존 솔루션이 없는 경우 OpenShift Container Platform 구독자에게 Red Hat OpenShift의 미러 레지스트리 가 제공됩니다. *Red Hat OpenShift의 미러 레지스트리* 는 서브스크립션에 포함되어 있으며 연결이 끊긴 설치에서 OpenShift Container Platform의 필요한 컨테이너 이미지를 미러링하는 데 사용할 수 있는 소규모 컨테이너 레지스트리입니다.

3.2. 미러 레지스트리 정보

OpenShift Container Platform 설치 및 후속 제품 업데이트에 Red Hat Quay, JFrog Artifactory, Sonatype Nexus Repository 또는 Harbor와 같은 컨테이너 미러 레지스트리에 필요한 이미지를 미러링할 수 있습니다. 대규모 컨테이너 레지스트리에 액세스할 수 없는 경우 OpenShift Container Platform 서브스크립션에 포함된 소규모 컨테이너 *레지스트리인 Red Hat OpenShift에 미러* 레지스트리를 사용할 수 있습니다.

Docker v2-2, *Red Hat OpenShift의 미러 레지스트리* , Artifactory, Sonatype Nexus Repository 또는 Harbor와 같은 Docker v2-2를 지원하는 모든 컨테이너 레지스트리를 사용할 수 있습니다. 선택한 레지스트리에 관계없이 인터넷상의 Red Hat 호스팅 사이트의 콘텐츠를 격리된 이미지 레지스트리로 미러링하는 절차는 동일합니다. 콘텐츠를 미러링한 후 미러 레지스트리에서 이 콘텐츠를 검색하도록 각 클러스터를 설정합니다.

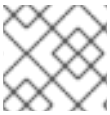
중요

OpenShift Container Platform 클러스터의 내부 레지스트리는 미러링 프로세스 중에 필요한 태그 없이 푸시를 지원하지 않으므로 대상 레지스트리로 사용할 수 없습니다.

Red Hat OpenShift의 미러 레지스트리가 아닌 컨테이너 레지스트리를 선택하는 경우 프로비저닝하는 클러스터의 모든 시스템에서 액세스할 수 있어야 합니다. 레지스트리에 연결할 수 없는 경우 설치, 업데이트 또는 워크로드 재배치와 같은 일반 작업이 실패할 수 있습니다. 따라서 고가용성 방식으로 미러 레지스트리를 실행해야하며 미러 레지스트리는 최소한 OpenShift Container Platform 클러스터의 프로덕션 환경의 가용성조건에 일치해야 합니다.

미러 레지스트리를 OpenShift Container Platform 이미지로 채우면 다음 두 가지 시나리오를 수행할 수 있습니다. 호스트가 인터넷과 미러 레지스트리에 모두 액세스할 수 있지만 클러스터 노드에 액세스 할 수 없는 경우 해당 머신의 콘텐츠를 직접 미러링할 수 있습니다. 이 프로세스를 *connected mirroring*(미러링 연결)이라고 합니다. 그러한 호스트가 없는 경우 이미지를 파일 시스템에 미러링한 다음 해당 호스트 또는 이동식 미디어를 제한된 환경에 배치해야 합니다. 이 프로세스를 *미러링 연결 해제*라고 합니다.

미러링된 레지스트리의 경우 가져온 이미지의 소스를 보려면 CRI-O 로그의 **Trying to access** 로그 항목을 검토해야 합니다. 노드에서 **crictl images** 명령을 사용하는 등의 이미지 가져오기 소스를 보는 다른 방법은 미러링되지 않은 이미지 이름을 표시합니다.



참고

Red Hat은 OpenShift Container Platform에서 타사 레지스트리를 테스트하지 않습니다.

추가 정보

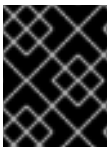
이미지 소스를 보기 위해 CRI-O 로그를 보는 방법에 대한 자세한 내용은 이미지 풀 소스 보기 를 참조하십시오.

3.3. 미러 호스트 준비

미러 단계를 수행하기 전에 호스트는 콘텐츠를 검색하고 원격 위치로 푸시할 준비가 되어 있어야 합니다.

3.3.1. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift CLI(**oc**)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 **oc**를 설치할 수 있습니다.



중요

이전 버전의 **oc**를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 **oc**를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 Linux에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Linux Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. **oc** 바이너리를 **PATH**에 있는 디렉터리에 배치합니다.
PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치
다음 절차에 따라 Windows에 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Windows Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. ZIP 프로그램으로 아카이브의 압축을 풉니다.
5. **oc** 바이너리를 **PATH**에 있는 디렉터리로 이동합니다.
PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치
다음 절차에 따라 macOS에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 MacOSX Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 해제하고 압축을 풉니다.
5. **oc** 바이너리 PATH의 디렉터리로 이동합니다.
PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

3.4. 이미지를 미러링할 수 있는 인증 정보 설정

Red Hat에서 미러로 이미지를 미러링할 수 있는 컨테이너 이미지 레지스트리 인증 정보 파일을 생성합니다.

주의

클러스터를 설치할 때 이 이미지 레지스트리 인증 정보 파일을 풀 시크릿(pull secret)으로 사용하지 마십시오. 클러스터를 설치할 때 이 파일을 지정하면 클러스터의 모든 시스템에 미러 레지스트리에 대한 쓰기 권한이 부여됩니다.

주의

이 프로세스에서는 미러 레지스트리의 컨테이너 이미지 레지스트리에 대한 쓰기 권한이 있어야 하며 인증 정보를 레지스트리 풀 시크릿에 추가해야 합니다.

사전 요구 사항

- 연결이 끊긴 환경에서 사용할 미러 레지스트리를 구성했습니다.
- 미러 레지스트리에서 이미지를 미러링할 이미지 저장소 위치를 확인했습니다.
- 이미지를 해당 이미지 저장소에 업로드할 수 있는 미러 레지스트리 계정을 제공하고 있습니다.

프로세스

설치 호스트에서 다음 단계를 수행합니다.

1. Red Hat OpenShift Cluster Manager에서 **registry.redhat.io** 풀 시크릿 을 다운로드하여 **.json** 파일에 저장합니다.
2. 미러 레지스트리에 대한 base64로 인코딩된 사용자 이름 및 암호 또는 토큰을 생성합니다.

```
$ echo -n '<user_name>:<password>' | base64 -w0 (1)
BGVtbYk3ZHAtqXs=
```

(1) **<user_name>** 및 **<password>**의 경우 레지스트리에 설정한 사용자 이름 및 암호를 지정합니다.

3. 풀 시크릿을 JSON 형식으로 복사합니다.

```
$ cat ./pull-secret.text | jq .  > <path>/<pull_secret_file_in_json> (1)
```

(1) 풀 시크릿을 저장할 폴더의 경로와 생성한 JSON 파일의 이름을 지정합니다.

4. 파일을 **~/.docker/config.json** 또는 **$XDG_RUNTIME_DIR/containers/auth.json** 으로 저장합니다.
   파일의 내용은 다음 예와 유사합니다.

```
{
  "auths": {
    "cloud.openshift.com": {
      "auth": "b3BlbnNo...",
      "email": "you@example.com"
    },
    "quay.io": {
      "auth": "b3BlbnNo...",
      "email": "you@example.com"
    },
    "registry.connect.redhat.com": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    },
    "registry.redhat.io": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    }
  }
}
```

5. 새 파일을 편집하고 레지스트리를 설명하는 섹션을 추가합니다.

```
  "auths": {
    "<mirror_registry>": { 1
      "auth": "<credentials>", 2
      "email": "you@example.com"
    }
  },
```

1 **<mirror_registry>**의 경우 미러 레지스트리가 콘텐츠를 제공하는데 사용하는 레지스트리 도메인 이름 및 포트 (선택 사항)를 지정합니다. 예: **registry.example.com** 또는 **registry.example.com:8443**

2 **<credentials>**의 경우 미러 레지스트리에 대해 base64로 인코딩된 사용자 이름 및 암호를 지정합니다.

   파일은 다음 예제와 유사합니다.

```
{
  "auths": {
    "registry.example.com": {
      "auth": "BGVtbYk3ZHAtqXs=",
      "email": "you@example.com"
    },
    "cloud.openshift.com": {
      "auth": "b3BlbnNo...",
      "email": "you@example.com"
    },
    "quay.io": {
```

```
      "auth": "b3BlbnNo...",
      "email": "you@example.com"
    },
    "registry.connect.redhat.com": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    },
    "registry.redhat.io": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    }
  }
}
```

3.5. RED HAT OPENSHIFT용 미러 레지스트리

Red Hat OpenShift의 미러 레지스트리 는 연결이 끊긴 설치에 OpenShift Container Platform의 필요한 컨테이너 이미지를 미러링하기 위한 대상으로 사용할 수 있는 작고 간소화된 컨테이너 레지스트리입니다.

Red Hat Quay와 같은 컨테이너 이미지 레지스트리가 이미 있는 경우 이러한 단계를 건너뛰고 OpenShift Container Platform 이미지 저장소 미러링 으로 바로 이동할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 서브스크립션.
- RHEL(Red Hat Enterprise Linux) 8 및 Podman 3.3 및 OpenSSL이 설치되어 있어야 합니다.
- DNS 서버를 통해 확인해야 하는 Red Hat Quay 서비스의 정규화된 도메인 이름입니다.
- 대상 호스트에서 암호가 없는 **sudo** 액세스 권한.
- 대상 호스트의 키 기반 SSH 연결입니다. 로컬 설치를 위해 SSH 키가 자동으로 생성됩니다. 원격 호스트의 경우 자체 SSH 키를 생성해야 합니다.
- 2개 이상의 vCPU
- 8GB RAM.
- OpenShift Container Platform 4.9 릴리스 이미지용 9.6GB 또는 OpenShift Container Platform 4.9 릴리스 이미지 및 OpenShift Container Platform 4.9 Red Hat Operator 이미지의 경우 약 444GB입니다. 스트림당 최대 1TB 또는 그 이상이 권장됩니다.

중요

이러한 요구 사항은 릴리스 이미지 및 Operator 이미지가 테스트된 로컬 테스트 결과를 기반으로 합니다. 스토리지 요구 사항은 조직의 요구에 따라 다를 수 있습니다. 일부 사용자에게는 예를 들어 여러 z-streams를 미러링할 때 더 많은 공간이 필요할 수 있습니다. 표준 Red Hat Quay 기능을 사용하여 불필요한 이미지를 제거하고 공간을 확보할 수 있습니다.

3.5.1. Red Hat OpenShift 소개용 미러 레지스트리

OpenShift Container Platform의 연결이 끊긴 배포의 경우 클러스터 설치를 수행하려면 컨테이너 레지스트리가 필요합니다. 이러한 클러스터에서 production-grade 레지스트리 서비스를 실행하려면 첫 번째 클

러스터를 설치하기 위해 별도의 레지스트리 배포를 생성해야 합니다. *Red Hat OpenShift 주소의 미러 레지스트리* 는 이러한 요구 사항이 있으며 모든 OpenShift 서브스크립션에 포함되어 있습니다. OpenShift 콘솔 **다운로드** 페이지에서 다운로드할 수 있습니다.

Red Hat OpenShift의 미러 레지스트리를 사용하면 ***mirror*** *-registry* CLI(명령줄 인터페이스) 툴을 사용하여 Red Hat Quay의 소규모 버전과 필요한 구성 요소를 설치할 수 있습니다. *Red Hat OpenShift의 미러 레지스트리* 는 사전 구성된 로컬 스토리지 및 로컬 데이터베이스와 함께 자동으로 배포됩니다. 또한 단일 입력 세트로 자동 생성된 사용자 자격 증명 및 액세스 권한이 포함되며, 시작할 추가 구성 선택 사항이 없습니다.

Red Hat OpenShift의 미러 레지스트리 는 사전 결정된 네트워크 구성 및 배포된 구성 요소 인증 정보 및 보고서 URL을 성공적으로 통해 URL에 액세스할 수 있습니다. FQDN(정규화된 도메인 이름) 서비스, 수퍼유저 이름 및 암호, 사용자 지정 TLS 인증서와 같은 제한된 선택적 구성 입력 세트도 제공됩니다. 이를 통해 제한된 네트워크 환경에서 OpenShift Container Platform을 실행할 때 모든 OpenShift Container Platform 릴리스 콘텐츠의 오프라인 미러를 쉽게 생성할 수 있도록 컨테이너 레지스트리가 제공됩니다.

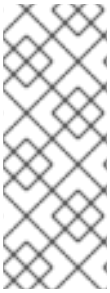
Red Hat OpenShift의 미러 레지스트리 는 릴리스 이미지 또는 Red Hat Operator 이미지와 같이 연결이 끊긴 OpenShift Container Platform 클러스터를 설치하는 데 필요한 이미지를 호스팅하도록 제한됩니다. RHEL(Red Hat Enterprise Linux) 머신의 로컬 스토리지를 사용하며 RHEL에서 지원하는 스토리지는 *Red Hat OpenShift의 미러 레지스트리에서* 지원합니다. 고객이 빌드한 콘텐츠는 *Red Hat OpenShift의 미러 레지스트리에서* 호스팅해서는 안 됩니다.

Red Hat Quay와 달리 *Red Hat OpenShift의 미러 레지스트리* 는 고가용성 레지스트리가 아니며 로컬 파일 시스템 스토리지만 지원됩니다. 클러스터 함대를 업데이트할 때 여러 클러스터가 단일 장애 지점을 생성할 수 있기 때문에 클러스터가 두 개 이상 클러스터에 *미러 레지스트리* 를 사용하는 것은 좋지 않습니다. *Red Hat OpenShift의 미러 레지스트리를 활용하여 OpenShift* Container Platform 콘텐츠를 다른 클러스터에 제공할 수 있는 Red Hat Quay와 같은 프로덕션 수준의 고가용성 레지스트리를 호스팅할 수 있는 클러스터를 설치하는 것이 좋습니다.

설치 환경에서 다른 컨테이너 *레지스트리를 이미 사용할 수 있는 경우 Red Hat OpenShift에 미러* 레지스트리를 사용하는 것이 선택 사항입니다.

3.5.2. Red Hat OpenShift의 미러 레지스트리를 사용하여 로컬 호스트에서 미러링

다음 절차에서는 ***mirror-registry* 설치 프로그램 도구를 사용하여 로컬 호스트에 Red Hat OpenShift의 미러 레지스트리** 를 설치하는 방법을 설명합니다. 이렇게 하면 OpenShift Container Platform 이미지의 미러를 저장하기 위해 포트 443에서 실행되는 로컬 호스트 레지스트리를 생성할 수 있습니다.



참고

***mirror-registry* CLI 툴을 사용하여 Red Hat OpenShift의 미러 레지스트리** 를 설치하면 머신을 몇 가지 변경할 수 있습니다. 설치 후 설치 파일, 로컬 스토리지 및 구성 번들이 있는 **/etc/quay-install** 디렉터리가 생성됩니다. 배포 대상이 로컬 호스트인 경우 신뢰할 수 있는 SSH 키가 생성되고, 컨테이너 런타임이 영구적인지 확인하기 위해 호스트 시스템의 systemd 파일이 설정됩니다. 또한 **init** 라는 초기 사용자는 자동으로 생성된 암호를 사용하여 생성됩니다. 모든 액세스 인증 정보는 설치 루틴이 끝나면 인쇄됩니다.

프로세스

1. OpenShift 콘솔 다운로드 페이지에 있는 *Red Hat OpenShift용 미러 레지스트리의 최신 버전의* ***mirror*** *-registry.tar.gz* 패키지를 **다운로드** 합니다.

2. ***mirror-registry* 도구를 사용하여 현재 사용자 계정으로 로컬 호스트에 Red Hat OpenShift의 미러 레지스트리** 를 설치합니다. 사용 가능한 플래그의 전체 목록은 "Red Hat OpenShift 플래그에 대한 레지스트리 미러링"을 참조하십시오.

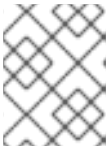
■

```
$ sudo ./mirror-registry install \
  --quayHostname <host_example_com> \
  --quayRoot <example_directory_name>
```

3. 설치 중에 생성된 사용자 이름과 암호를 사용하여 다음 명령을 실행하여 레지스트리에 로그인합니다.

```
$ podman login --authfile pull-secret.txt \
  -u init \
  -p <password> \
  <host_example_com>:8443> \
  --tls-verify=false 1
```

1 생성된 rootCA 인증서를 신뢰하도록 시스템을 구성하여 **--tls-verify=false** 실행을 방지할 수 있습니다. 자세한 내용은 "SSL을 사용하여 Red Hat Quay에 대한 연결 보호" 및 "시스템에서 인증 기관을 신뢰하도록 구성"을 참조하십시오.



참고

설치 후 **https://<host.example.com>:8443** 에서 UI에 액세스하여 로그인할 수도 있습니다.

4. 로그인 후 OpenShift Container Platform 이미지를 미러링할 수 있습니다. 요구 사항에 따라 이 문서의 "OpenShift Container Platform 이미지 리포지토리 미러링" 또는 "연결이 끊긴 클러스터에 사용할 Operator 카탈로그 미러링" 섹션을 참조하십시오.

참고

스토리지 계층 문제로 인해 *Red Hat OpenShift의 미러 레지스트리에 저장된 이미지에 문제가 있는 경우* OpenShift Container Platform 이미지를 다시 미러링하거나 더 안정적인 스토리지에 미러 레지스트리를 다시 설치할 수 있습니다.

3.5.3. Red Hat OpenShift의 미러 레지스트리를 사용하여 원격 호스트에서 미러링

다음 절차에서는 ***mirror-registry* 도구를 사용하여 원격 호스트에 Red Hat OpenShift의 미러 레지스트리** 를 설치하는 방법을 설명합니다. 이렇게 하면 사용자가 OpenShift Container Platform 이미지의 미러를 저장할 레지스트리를 생성할 수 있습니다.

참고

***mirror-registry* CLI 툴을 사용하여 Red Hat OpenShift의 미러 레지스트리** 를 설치하면 머신을 몇 가지 변경할 수 있습니다. 설치 후 설치 파일, 로컬 스토리지 및 구성 번들이 있는 **/etc/quay-install** 디렉터리가 생성됩니다. 배포 대상이 로컬 호스트인 경우 신뢰할 수 있는 SSH 키가 생성되고, 컨테이너 런타임이 영구적인지 확인하기 위해 호스트 시스템의 systemd 파일이 설정됩니다. 또한 **init** 라는 초기 사용자는 자동으로 생성된 암호를 사용하여 생성됩니다. 모든 액세스 인증 정보는 설치 루틴이 끝나면 인쇄됩니다.

프로세스

1. OpenShift 콘솔 다운로드 페이지에 있는 *Red Hat OpenShift용 미러 레지스트리의 최신 버전의* ***mirror****-registry.tar.gz* 패키지를 **다운로드** 합니다.

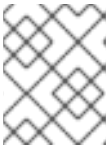
2. ***mirror-registry*** **도구를 사용하여 현재 사용자 계정으로 로컬 호스트에 Red Hat OpenShift의 미러 레지스트리** 를 설치합니다. 사용 가능한 플래그의 전체 목록은 "Red Hat OpenShift 플래그에 대한 레지스트리 미러링"을 참조하십시오.

```
$ sudo ./mirror-registry install -v \
  --targetHostname <host_example_com> \
  --targetUsername <example_user> \
  -k ~/.ssh/my_ssh_key \
  --quayHostname <host_example_com> \
  --quayRoot <example_directory_name>
```

3. 설치 중에 생성된 사용자 이름과 암호를 사용하여 다음 명령을 실행하여 미러 레지스트리에 로그인합니다.

```
$ podman login --authfile pull-secret.txt \
  -u init \
  -p <password> \
  <host_example_com>:8443> \
  --tls-verify=false 1
```

1 생성된 rootCA 인증서를 신뢰하도록 시스템을 구성하여 **--tls-verify=false** 실행을 방지할 수 있습니다. 자세한 내용은 "SSL을 사용하여 Red Hat Quay에 대한 연결 보호" 및 "시스템에서 인증 기관을 신뢰하도록 구성"을 참조하십시오.



참고

설치 후 **https://<host.example.com>:8443** 에서 UI에 액세스하여 로그인할 수도 있습니다.

4. 로그인 후 OpenShift Container Platform 이미지를 미러링할 수 있습니다. 요구 사항에 따라 이 문서의 "OpenShift Container Platform 이미지 리포지토리 미러링" 또는 "연결이 끊긴 클러스터에 사용할 Operator 카탈로그 미러링" 섹션을 참조하십시오.

참고

스토리지 계층 문제로 인해 *Red Hat OpenShift의 미러 레지스트리에 저장된 이미지에 문제가 있는 경우 OpenShift* Container Platform 이미지를 다시 미러링하거나 더 안정적인 스토리지에 미러 레지스트리를 다시 설치할 수 있습니다.

3.6. RED HAT OPENSHIFT의 미러 레지스트리 업그레이드

- 다음 명령을 실행하여 로컬 호스트에서 *Red Hat OpenShift의 미러 레지스트리* 를 업그레이드할 수 있습니다.

```
$ sudo ./mirror-registry upgrade
```

참고

- **./mirror-registry upgrade** 플래그를 사용하여 *Red Hat OpenShift의 미러 레지스트리* 를 업그레이드하는 사용자는 미러 레지스트리를 만들 때 사용되는 것과 동일한 인증 정보를 포함해야 합니다. 예를 들어 **--quayHostname <host_example_com> 및 --quay Root <example_directory_name>** 를 사용하여 *Red Hat OpenShift의 미러 레지스트리* 를 설치한 경우 미러 레지스트리를 올바르게 업그레이드하려면 해당 문자열을 포함해야 합니다.

3.6.1. Red Hat OpenShift의 미러 레지스트리 설치 제거

- 다음 명령을 실행하여 로컬 호스트에서 *Red Hat OpenShift의 미러 레지스트리* 를 설치 제거할 수 있습니다.

```
$ sudo ./mirror-registry uninstall -v \
  --quayRoot <example_directory_name>
```

참고

- *Red Hat OpenShift의 미러 레지스트리*를 삭제하면 삭제하기 전에 사용자에게 메시지를 표시합니다. **--autoApprove** 를 사용하여 이 프롬프트를 건너뛸 수 있습니다.
- --quayRoot 플래그를 사용하여 *Red Hat OpenShift의 미러 레지스트리* 를 설치하는 사용자는 제거할 때 **--quayRoot** 플래그를 포함해야 합니다. 예를 들어 **--quayRoot example_directory_name** 을 사용하여 *Red Hat OpenShift의 미러 레지스트리*를 설치한 경우 미러 레지스트리를 올바르게 제거하려면 해당 문자열을 포함해야 합니다.

3.6.2. Red Hat OpenShift 플래그의 레지스트리 미러링

다음 플래그는 *Red Hat OpenShift의 미러 레지스트리에 사용할* 수 있습니다.

플래그	설명
--autoApprove	대화형 프롬프트를 비활성화하는 부울 값입니다. **true** 로 설정하면 미러 레지스트리를 제거할 때 **quayRoot** 디렉터리가 자동으로 삭제됩니다. 지정되지 않은 경우 기본값은 **false** 입니다.
--initPassword	Quay 설치 중에 생성된 init 사용자의 암호입니다. 8자 이상이어야 하며 공백을 포함하지 않아야 합니다.
--initUser 문자열	초기 사용자의 사용자 이름을 표시합니다. 지정되지 않은 경우 기본값은 **init** 입니다.
--quayHostname	클라이언트가 레지스트리에 연결하는 데 사용할 미러 레지스트리의 정규화된 도메인 이름입니다. Quay **config.yaml** 의 **SERVER_HOSTNAME** 과 동일합니다. DNS로 확인해야 합니다. 지정되지 않은 경우 기본값은 **<targetHostname>:8443** 입니다. [1]

플래그	설명
--quayRoot, -r	**rootCA.key**,**rootCA.pem**, **rootCA.srl** 인증서를 포함하여 컨테이너 이미지 계층 및 구성 데이터가 저장되는 디렉터리입니다. OpenShift Container Platform 4.9 릴리스 이미지용 9.6GB 또는 OpenShift Container Platform 4.9 릴리스 이미지 및 OpenShift Container Platform 4.9 Red Hat Operator 이미지의 경우 약 444GB가 필요합니다. 지정되지 않은 경우 기본값은 **/etc/quay-install** 입니다.
--ssh-key, -k	SSH ID 키의 경로입니다. 지정되지 않은 경우 기본값은 **~/.ssh/quay_installer** 입니다.
--sslCert	SSL/TLS 공개 키 / 인증서의 경로입니다. 기본값은 **{quayRoot}/quay-config** 이며 지정되지 않은 경우 자동으로 생성됩니다.
--sslCheckSkip	**config.yaml** 파일의 **SERVER_HOSTNAME** 에 대해 인증서 호스트 이름의 검사를 건너뜁니다. [2]
--sslKey	HTTPS 통신에 사용되는 SSL/TLS 개인 키의 경로입니다. 기본값은 **{quayRoot}/quay-config** 이며 지정되지 않은 경우 자동으로 생성됩니다.
--targetHostname, -H	Quay를 설치할 대상의 호스트 이름입니다. 지정되지 않은 경우 기본값은 **$HOST** (예: 로컬 호스트)입니다.
--targetUsername, -u	SSH에 사용할 대상 호스트의 사용자입니다. 기본값은 **$USER** 입니다. 예를 들어 지정되지 않은 경우 현재 사용자입니다.
--verbose, -v	는 디버그 로그 및 Ansible 플레이북 출력을 표시합니다.
--version	*Red Hat OpenShift의 미러 레지스트리 버전을*보여줍니다.

1. 시스템의 공용 DNS 이름이 로컬 호스트 이름과 다른 경우 **--quayHostname** 을 수정해야 합니다.
2. **--sslCheckSkip** 은 미러 레지스트리가 프록시 뒤에 설정되고 노출된 호스트 이름이 내부 Quay 호스트 이름과 다른 경우 사용됩니다. 설치하는 동안 제공된 Quay 호스트 이름에 대해 인증서의 유효성을 검사하지 않으려면 사용자가 사용할 수도 있습니다.

추가 리소스

- SSL을 사용하여 Red Hat Quay 연결 보호
- 인증 기관을 신뢰하도록 시스템 구성
- OpenShift Container Platform 이미지 저장소 미러링
- 연결이 끊긴 클러스터와 함께 사용할 Operator 카탈로그 미러링

3.7. OPENSHIFT CONTAINER PLATFORM 이미지 저장소 미러링

클러스터 설치 또는 업그레이드 중에 사용할 OpenShift Container Platform 이미지 저장소를 레지스트리에 미러링합니다.

사전 요구 사항

- 미러 호스트가 인터넷에 액세스할 수 있습니다.
- 네트워크가 제한된 환경에서 사용할 미러 레지스트리를 설정하고 설정한 인증서 및 인증 정보에 액세스할 수 있습니다.
- Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 다운로드하여 미러 저장소에 대한 인증을 포함하도록 수정했습니다.
- Subject Alternative Name을 설정하지 않는 자체 서명된 인증서를 사용하는 경우 이 절차의 **oc** 명령 앞에 **GODEBUG=x509ignoreCN=0**을 지정해야 합니다. 이 변수를 설정하지 않으면 **oc** 명령이 다음 오류로 인해 실패합니다.

```
x509: certificate relies on legacy Common Name field, use SANs or temporarily enable Common Name matching with GODEBUG=x509ignoreCN=0
```

프로세스

미러 호스트에서 다음 단계를 완료합니다.

1. OpenShift Container Platform 다운로드 페이지 를 확인하여 설치할 OpenShift Container Platform 버전을 확인하고 Repository Tags 페이지에서 해당 태그를 지정합니다.
2. 필요한 환경 변수를 설정합니다.
   a. 릴리스 버전을 내보냅니다.

   ```
   $ OCP_RELEASE=<release_version>
   ```

   <release_version>에 대해 설치할 OpenShift Container Platform 버전에 해당하는 태그를 지정합니다 (예: **4.5.4**).

   b. 로컬 레지스트리 이름 및 호스트 포트를 내보냅니다.

   ```
   $ LOCAL_REGISTRY='<local_registry_host_name>:<local_registry_host_port>'
   ```

   <local_registry_host_name>의 경우 미러 저장소의 레지스트리 도메인 이름을 지정하고 **<local_registry_host_port>**의 경우 콘텐츠를 제공하는데 사용되는 포트를 지정합니다.

   c. 로컬 저장소 이름을 내보냅니다.

   ```
   $ LOCAL_REPOSITORY='<local_repository_name>'
   ```

   <local_repository_name>의 경우 레지스트리에 작성할 저장소 이름 (예: **ocp4/openshift4**)을 지정합니다.

   d. 미러링할 저장소 이름을 내보냅니다.

   ```
   $ PRODUCT_REPO='openshift-release-dev'
   ```

   프로덕션 환경의 릴리스의 경우 **openshift-release-dev**를 지정해야 합니다.

   e. 레지스트리 풀 시크릿의 경로를 내보냅니다.

   ```
   $ LOCAL_SECRET_JSON='<path_to_pull_secret>'
   ```

생성한 미러 레지스트리에 대한 풀 시크릿의 절대 경로 및 파일 이름을 **<path_to_pull_secret>**에 지정합니다.

f. 릴리스 미러를 내보냅니다.

```
$ RELEASE_NAME="ocp-release"
```

프로덕션 환경의 릴리스의 경우 **ocp-release**를 지정해야 합니다.

g. 서버의 아키텍처 유형 (예: **x86_64**)을 내보냅니다.

```
$ ARCHITECTURE=<server_architecture>
```

h. 미러링된 이미지를 호스트할 디렉터리의 경로를 내보냅니다.

```
$ REMOVABLE_MEDIA_PATH=<path> 1
```

1 초기 슬래시 (/) 문자를 포함하여 전체 경로를 지정합니다.

3. 미러 레지스트리에 버전 이미지를 미러링합니다.

- 미러 호스트가 인터넷에 액세스할 수 없는 경우 다음 작업을 수행합니다.

i. 이동식 미디어를 인터넷에 연결된 시스템에 연결합니다.

ii. 미러링할 이미지 및 설정 매니페스트를 확인합니다.

```
$ oc adm release mirror -a ${LOCAL_SECRET_JSON}  \
    --from=quay.io/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELEASE}-
${ARCHITECTURE} \
    --to=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY} \
    --to-release-
image=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-
${ARCHITECTURE} --dry-run
```

iii. 이전 명령의 출력에서 전체 **imageContentSources** 섹션을 기록합니다. 미러에 대한 정보는 미러링된 저장소에 고유하며 설치 중에 **imageContentSources** 섹션을 **install-config.yaml** 파일에 추가해야 합니다.

iv. 이동식 미디어의 디렉터리에 이미지를 미러링합니다.

```
$ oc adm release mirror -a ${LOCAL_SECRET_JSON} --to-
dir=${REMOVABLE_MEDIA_PATH}/mirror
quay.io/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELEASE}-
${ARCHITECTURE}
```

v. 미디어를 네트워크가 제한된 환경으로 가져와서 이미지를 로컬 컨테이너 레지스트리에 업로드합니다.

```
$ oc image mirror -a ${LOCAL_SECRET_JSON} --from-
dir=${REMOVABLE_MEDIA_PATH}/mirror
"file://openshift/release:${OCP_RELEASE}*"
${LOCAL_REGISTRY}/${LOCAL_REPOSITORY} 1
```

1 **REMOVABLE_MEDIA_PATH**의 경우 이미지를 미러링 할 때 지정한 것과 동일한 경로를 사용해야 합니다.

- 로컬 컨테이너 레지스트리가 미러 호스트에 연결된 경우 다음 작업을 수행합니다.

  i. 다음 명령을 사용하여 릴리스 이미지를 로컬 레지스트리에 직접 푸시합니다.

```
$ oc adm release mirror -a ${LOCAL_SECRET_JSON}  \
    --from=quay.io/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELEASE}-
${ARCHITECTURE} \
    --to=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY} \
    --to-release-
image=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-
${ARCHITECTURE}
```

  이 명령은 요약된 릴리스 정보를 가져오며, 명령 출력에는 클러스터를 설치할 때 필요한 **imageContentSources** 데이터가 포함됩니다.

  ii. 이전 명령의 출력에서 전체 **imageContentSources** 섹션을 기록합니다. 미러에 대한 정보는 미러링된 저장소에 고유하며 설치 중에 **imageContentSources** 섹션을 **install-config.yaml** 파일에 추가해야 합니다.

참고

미러링 프로세스 중에 이미지 이름이 Quay.io에 패치되고 podman 이미지는 부트스트랩 가상 머신의 레지스트리에 Quay.io를 표시합니다.

4. 미러링된 콘텐츠를 기반으로 설치 프로그램을 생성하려면 콘텐츠를 추출하여 릴리스 배포에 고정합니다.

- 미러 호스트가 인터넷에 액세스할 수 없는 경우 다음 명령을 실행합니다.

```
$ oc adm release extract -a ${LOCAL_SECRET_JSON} --command=openshift-install
"${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}"
```

- 로컬 컨테이너 레지스트리가 미러 호스트에 연결된 경우 다음 명령을 실행합니다.

```
$ oc adm release extract -a ${LOCAL_SECRET_JSON} --command=openshift-install
"${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-
${ARCHITECTURE}"
```

중요

선택한 OpenShift Container Platform 버전에 올바른 이미지를 사용하려면 미러링된 콘텐츠에서 설치 프로그램을 배포해야 합니다.

인터넷이 연결된 컴퓨터에서 이 단계를 수행해야 합니다.

연결이 끊긴 환경에 있는 경우 **--image** 플래그를 must-gather의 일부로 사용하여 페이로드 이미지를 가리킵니다.

5. 설치 프로그램에서 제공하는 인프라를 사용하는 클러스터의 경우 다음 명령을 실행합니다.

```
$ openshift-install
```

3.8. 연결이 끊긴 환경의 CLUSTER SAMPLES OPERATOR

연결이 끊긴 환경에서는 Cluster Samples Operator를 구성하기 위해 클러스터를 설치한 후 추가 단계를 수행해야 합니다. 준비 과정에서 다음 정보를 검토합니다.

3.8.1. 미러링을 위한 Cluster Samples Operator 지원

설치 프로세스 중에 OpenShift Container Platform은 **openshift-cluster-samples-operator** 네임스페이스에 **imagestreamtag-to-image**라는 구성 맵을 생성합니다. **imagestreamtag-to-image** 구성 맵에는 각 이미지 스트림 태그에 대한 이미지 채우기 항목이 포함되어 있습니다.

구성 맵의 데이터 필드에 있는 각 항목의 키 형식은 **<image_stream_name>_<image_stream_tag_name>**입니다.

OpenShift Container Platform의 연결이 끊긴 설치 프로세스 중에 Cluster Samples Operator의 상태가 **Removed**로 설정됩니다. **Managed**로 변경하려면 샘플이 설치됩니다.

참고

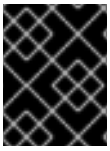
네트워크 제한 또는 중단된 환경에서 샘플을 사용하려면 네트워크 외부의 서비스에 액세스해야 할 수 있습니다. 일부 예제 서비스에는 GitHub, Maven Central, npm, RubyGems, PyPi 등이 있습니다. 클러스터 샘플 Operator의 오브젝트가 필요한 서비스에 도달할 수 있도록 하는 추가 단계가 있을 수 있습니다.

이 구성 맵을 사용하여 이미지 스트림을 가져오려면 이미지를 미러링해야 하는 이미지 참조로 사용할 수 있습니다.

- Cluster Samples Operator가 **Removed**로 설정된 경우 미러링된 레지스트리를 생성하거나 사용할 기존 미러링된 레지스트리를 확인할 수 있습니다.
- 새 구성 맵을 가이드로 사용하여 미러링된 레지스트리에 샘플을 미러링합니다.
- Cluster Samples Operator 구성 개체의 **skippedImagestreams** 필드에 미러링되지 않은 이미지 스트림을 추가합니다.
- Cluster Samples Operator 구성 개체의 **samplesRegistry** 를 미러링된 레지스트리로 설정합니다.
- 그런 다음 Cluster Samples Operator를 **Managed**로 설정하여 미러링된 이미지 스트림을 설치합니다.

3.9. 연결이 끊긴 클러스터와 함께 사용할 OPERATOR 카탈로그 미러링

oc adm catalog mirror 명령을 사용하여 Red Hat 제공 카탈로그 또는 사용자 정의 카탈로그의 Operator 콘텐츠를 컨테이너 이미지 레지스트리에 미러링 할 수 있습니다. 대상 레지스트리는 Docker v2-2를 지원해야 합니다. 제한된 네트워크에 있는 클러스터의 경우 이 레지스트리는 제한된 네트워크 클러스터 설치 중 생성된 미러 레지스트리와 같이 클러스터에 네트워크 액세스 권한이 있는 레지스트리일 수 있습니다.



중요

OpenShift Container Platform 클러스터의 내부 레지스트리는 미러링 프로세스 중에 필요한 태그 없이 푸시를 지원하지 않으므로 대상 레지스트리로 사용할 수 없습니다.

oc adm catalog mirror 명령은 또한 Red Hat 제공 인덱스 이미지이든 자체 사용자 정의 빌드 인덱스 이미지이든 미러링 프로세스 중에 지정하는 인덱스 이미지를 대상 레지스트리에 자동으로 미러링합니다. 그러면 미러링된 인덱스 이미지를 사용하여 OLM(Operator Lifecycle Manager)이 OpenShift Container Platform 클러스터에 미러링된 카탈로그를 로드할 수 있는 카탈로그 소스를 생성할 수 있습니다.

추가 리소스

- 제한된 네트워크에서 Operator Lifecycle Manager 사용

3.9.1. 사전 요구 사항

연결이 끊긴 클러스터에 사용할 Operator 카탈로그 미러링에는 다음과 같은 사전 요구 사항이 있습니다.

- 워크스테이션에서 무제한 네트워크 액세스가 가능합니다.
- **podman** 버전이 1.9.3 이상입니다.
- 기본 카탈로그를 필터링하거나 *정리하고* 선택적으로 Operator의 서브 세트만 미러링하려면 다음 섹션을 참조하십시오.
  - opm CLI 설치
  - SQLite 기반 인덱스 이미지 필터링
- Red Hat 제공 카탈로그를 미러링하려면 무제한 네트워크 액세스 권한이 있는 워크스테이션에서 다음 명령을 실행하여 **registry.redhat.io**로 인증합니다.

```
$ podman login registry.redhat.io
```

- Docker v2-2 를 지원하는 미러 레지스트리에 액세스합니다.
- 미러 레지스트리에서 미러링된 Operator 콘텐츠를 저장하는 데 사용할 네임스페이스를 결정합니다. 예를 들어 **olm-mirror** 네임스페이스를 생성할 수 있습니다.
- 미러 레지스트리가 인터넷에 액세스할 수 없는 경우 이동식 미디어를 무제한 네트워크 액세스 권한이 있는 워크스테이션에 연결합니다.
- **registry.redhat.io를 포함한 프라이빗 레지스트리로** 작업하는 경우 **REG_CREDS** 환경 변수를 이후 단계에서 사용할 레지스트리 자격 증명의 파일 경로로 설정합니다. 예를 들어 **podman** CLI의 경우:

```
$ REG_CREDS=${XDG_RUNTIME_DIR}/containers/auth.json
```

3.9.2. 카탈로그 콘텐츠 추출 및 미러링

oc adm catalog mirror 명령은 인덱스 이미지의 콘텐츠를 추출하여 미러링에 필요한 매니페스트를 생성합니다. 명령의 기본 동작은 매니페스트를 생성한 다음 인덱스 이미지 자체뿐만 아니라 인덱스 이미지의 모든 이미지 콘텐츠를 미러 레지스트리에 자동으로 미러링합니다.

또는 미러 레지스트리가 완전히 연결이 끊긴 호스트 또는 *에어갭(Airgap)* 호스트에 있는 경우 먼저 콘텐츠를 이동식 미디어로 미러링하고 미디어를 연결이 끊긴 환경으로 이동한 다음 미디어에서 레지스트리로 해당 콘텐츠를 미러링할 수 있습니다.

3.9.2.1. 동일한 네트워크의 레지스트리에 카탈로그 콘텐츠 미러링

미러 레지스트리가 무제한 네트워크 액세스 권한이 있는 워크스테이션과 동일한 네트워크에 있는 경우 워크스테이션에서 다음 작업을 수행합니다.

프로세스

1. 미러 레지스트리에 인증이 필요한 경우 다음 명령을 실행하여 레지스트리에 로그인합니다.

```
$ podman login <mirror_registry>
```

2. 다음 명령을 실행하여 콘텐츠를 미러 레지스트리에 추출하고 미러링합니다.

```
$ oc adm catalog mirror \
    <index_image> \ 1
    <mirror_registry>:<port>/<namespace> \ 2
    [-a ${REG_CREDS}] \ 3
    [--insecure] \ 4
    [--index-filter-by-os='<platform>/<arch>'] \ 5
    [--manifests-only] 6
```

1. 미러링할 카탈로그의 인덱스 이미지를 지정합니다. 예를 들어 이전에 생성한 정리된 인덱스 이미지 또는 **registry.redhat.io/redhat/redhat-operator-index:v4.9** 와 같은 기본 카탈로그의 소스 인덱스 이미지 중 하나일 수 있습니다.

2. Operator 콘텐츠를 미러링할 대상 레지스트리 및 네임스페이스의 정규화된 도메인 이름(FQDN)을 지정합니다. 여기서 **<namespace>** 는 레지스트리의 기존 네임스페이스입니다. 예를 들어 미러링된 콘텐츠를 모두 내보내기 위해 **olm-mirror** 네임스페이스를 생성할 수 있습니다.

3. 선택 사항: 필요한 경우 레지스트리 자격 증명 파일의 위치를 지정합니다. **registry.redhat.io** 에는 **{REG_CREDS}** 가 필요합니다.

4. 선택 사항: 대상 레지스트리에 대한 트러스트를 구성하지 않으려면 **--insecure** 플래그를 추가합니다.

5. 선택 사항: 여러 변형이 있을 때 선택할 수 있는 인덱스 이미지의 플랫폼 및 아키텍처를 지정합니다. 이미지는 **'<platform>/<arch>[/<variant>]'** 로 전달됩니다. 이는 인덱스에서 참조하는 이미지에는 적용되지 않습니다. 유효한 값은 **linux/amd64**, **linux/ppc64le**, **linux/s390x**, **.*** 입니다.

6. 선택 사항: 미러링에 필요한 매니페스트만 생성하고 실제로 이미지 콘텐츠를 레지스트리에 미러링하지 않습니다. 이 선택 사항은 미러링할 항목을 검토하는 데 유용할 수 있으며 패키지의 서브 세트만 필요한 경우 매핑 목록을 변경할 수 있습니다. 그런 다음 **oc image mirror** 명령과 함께 **mapping.txt** 파일을 사용하여 이후 단계에서 수정된 이미지 목록을 미러링할 수 있습니다. 이 플래그는 카탈로그에서 콘텐츠의 고급 선택적 미러링에만 사용됩니다. 이전에 인덱스 이미지를 정리하기 위해 사용한 경우 **opm index prune** 명령은 대부분의 카탈로그 관리 사용 사례에 적합합니다.

출력 예

```
src image has index label for database path: /database/index.db
using database path mapping: /database/index.db:/tmp/153048078
wrote database to /tmp/153048078 1
...
wrote mirroring manifests to manifests-redhat-operator-index-1614211642 2
```

1. 명령으로 생성된 임시 **index.db** 데이터베이스용 디렉터리입니다.
2. 생성된 매니페스트 디렉터리 이름을 기록합니다. 이 디렉터리는 후속 절차에서 참조됩니다.

참고

Red Hat Quay는 중첩 리포지토리를 지원하지 않습니다. 결과적으로 **oc adm catalog mirror** 명령을 실행하면 **401** 무단 오류와 함께 실패합니다. 이 문제를 해결하려면 **oc adm catalog mirror** 명령을 실행할 때 **--max-components=2** 옵션을 사용하여 중첩된 리포지토리 생성을 비활성화할 수 있습니다. 이 해결 방법에 대한 자세한 내용은 Quay 레지스트리 지식베이스 솔루션과 함께 catalog mirror 명령을 사용하는 동안 인증되지 않은 오류를 참조하십시오.

추가 리소스

- Operator에 대한 아키텍처 및 운영 체제 지원

3.9.2.2. 여유가 있는 레지스트리로 카탈로그 콘텐츠 미러링

미러 레지스트리가 완전히 연결이 끊긴 호스트 또는 항공 호스트에 있는 경우 다음 작업을 수행합니다.

프로세스

1. 무제한 네트워크 액세스 권한이 있는 워크스테이션에서 다음 명령을 실행하여 콘텐츠를 로컬 파일에 미러링합니다.

```
$ oc adm catalog mirror \
    <index_image> \ 1
    file:///local/index \ 2
    -a ${REG_CREDS} \ 3
    --insecure \ 4
    --index-filter-by-os='<platform>/<arch>' 5
```

1. 미러링할 카탈로그의 인덱스 이미지를 지정합니다. 예를 들어 이전에 생성한 정리된 인덱스 이미지 또는 **registry.redhat.io/redhat/redhat-operator-index:v4.9** 와 같은 기본 카탈로그의 소스 인덱스 이미지 중 하나일 수 있습니다.
2. 현재 디렉터리의 로컬 파일에 미러링할 콘텐츠를 지정합니다.
3. 선택 사항: 필요한 경우 레지스트리 자격 증명 파일의 위치를 지정합니다.
4. 선택 사항: 대상 레지스트리에 대한 트러스트를 구성하지 않으려면 **--insecure** 플래그를 추가합니다.
5. 선택 사항: 여러 변형이 있을 때 선택할 수 있는 인덱스 이미지의 플랫폼 및 아키텍처를 지정합니다. 이미지는 **'<platform>/<arch>[/<variant>]'** 로 지정됩니다. 이는 인덱스에서 참조하는 이미지에는 적용되지 않습니다. 유효한 값은 **linux/amd64**, **linux/ppc64le**, **linux/s390x**,

.*입니다.

출력 예

```
...
info: Mirroring completed in 5.93s (5.915MB/s)
wrote mirroring manifests to manifests-my-index-1614985528 (1)

To upload local images to a registry, run:

 oc adm catalog mirror file://local/index/myrepo/my-index:v1 REGISTRY/REPOSITORY (2)
```

(1) 생성된 매니페스트 디렉터리 이름을 기록합니다. 이 디렉터리는 후속 절차에서 참조됩니다.

(2) 제공된 인덱스 이미지를 기반으로 확장된 **file://** 경로를 기록합니다. 이 경로는 후속 단계에서 참조됩니다.

이 명령은 현재 디렉터리에 **v2/** 디렉터리를 생성합니다.

2. **v2/** 디렉터리를 이동식 미디어로 복사합니다.
3. 물리적으로 미디어를 제거하고 연결이 끊긴 환경의 호스트에 연결하여 미러 레지스트리에 액세스할 수 있습니다.
4. 미러 레지스트리에 인증이 필요한 경우 연결이 끊긴 환경의 호스트에서 다음 명령을 실행하여 레지스트리에 로그인합니다.

```
$ podman login <mirror_registry>
```

5. **v2/** 디렉터리가 포함된 상위 디렉터리에서 다음 명령을 실행하여 로컬 파일에서 미러 레지스트리로 이미지를 업로드합니다.

```
$ oc adm catalog mirror \
    file://local/index/<repo>/<index_image>:<tag> \ (1)
    <mirror_registry>:<port>/<namespace> \ (2)
    -a ${REG_CREDS} \ (3)
    --insecure \ (4)
    --index-filter-by-os='<platform>/<arch>' (5)
```

(1) 이전 명령 출력의 **file://** 경로를 지정합니다.

(2) Operator 콘텐츠를 미러링할 대상 레지스트리 및 네임스페이스의 정규화된 도메인 이름(FQDN)을 지정합니다. 여기서 **<namespace>** 는 레지스트리의 기존 네임스페이스입니다. 예를 들어 미러링된 콘텐츠를 모두 내보내기 위해 **olm-mirror** 네임스페이스를 생성할 수 있습니다.

(3) 선택 사항: 필요한 경우 레지스트리 자격 증명 파일의 위치를 지정합니다.

(4) 선택 사항: 대상 레지스트리에 대한 트러스트를 구성하지 않으려면 **--insecure** 플래그를 추가합니다.

(5)

선택 사항: 여러 변형이 있을 때 선택할 수 있는 인덱스 이미지의 플랫폼 및 아키텍처를 지정합니다. 이미지는 **'<platform>/<arch>[/<variant>]'** 로 지정됩니다. 이는 인덱스에서 참조하

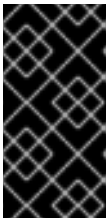
참고

Red Hat Quay는 중첩 리포지토리를 지원하지 않습니다. 결과적으로 **oc adm catalog mirror** 명령을 실행하면 **401** 무단 오류와 함께 실패합니다. 이 문제를 해결하려면 **oc adm catalog mirror** 명령을 실행할 때 **--max-components=2** 옵션을 사용하여 중첩된 리포지토리 생성을 비활성화할 수 있습니다. 이 해결 방법에 대한 자세한 내용은 Quay 레지스트리 지식베이스 솔루션과 함께 catalog mirror 명령을 사용하는 동안 인증되지 않은 오류를 참조하십시오.

6. **oc adm catalog mirror** 명령을 다시 실행합니다. 새로 미러링된 인덱스 이미지를 소스로 사용하고 이전 단계에서 사용한 것과 동일한 미러 레지스트리 네임스페이스를 대상으로 사용합니다.

```
$ oc adm catalog mirror \
    <mirror_registry>:<port>/<index_image> \
    <mirror_registry>:<port>/<namespace> \
    --manifests-only \ 1
    [-a ${REG_CREDS}] \
    [--insecure]
```

1 이 단계에서는 명령이 미러링된 모든 콘텐츠를 다시 복사하지 않도록 **--manifests-only** 플래그가 필요합니다.



중요

이전 단계에서 생성된 **imageContentSourcePolicy.yaml** 파일의 이미지 매핑은 로컬 경로에서 유효한 미러 위치로 업데이트해야 하므로 이 단계가 필요합니다. 이렇게 하지 않으면 이후 단계에서 **ImageContentSourcePolicy** 개체를 생성할 때 오류가 발생합니다.

카탈로그를 미러링한 후 나머지 클러스터 설치를 계속할 수 있습니다. 클러스터 설치가 성공적으로 완료되면 **ImageContentSourcePolicy** 및 **CatalogSource** 오브젝트를 생성하려면 이 프로세스에서 매니페스트 디렉터리를 지정해야 합니다. OperatorHub에서 Operator 설치를 활성화하려면 이러한 오브젝트가 필요합니다.

추가 리소스

- Operator에 대한 아키텍처 및 운영 체제 지원

3.9.3. 생성된 매니페스트

Operator 카탈로그 콘텐츠를 미러 레지스트리에 미러링한 후 현재 디렉터리에 매니페스트 디렉터리가 생성됩니다.

콘텐츠를 동일한 네트워크의 레지스트리에 미러링한 경우 디렉터리 이름은 다음 패턴을 사용합니다.

```
manifests-<index_image_name>-<random_number>
```

이전 섹션의 연결이 끊긴 호스트의 레지스트리에 콘텐츠를 미러링한 경우 디렉터리 이름은 다음 패턴을 사용합니다.

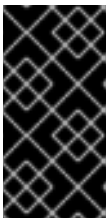
```
manifests-index/<namespace>/<index_image_name>-<random_number>
```

참고

매니페스트 디렉터리 이름은 후속 절차에서 참조됩니다.

매니페스트 디렉터리에는 다음 파일이 포함되어 있으며, 이 중 일부는 추가 수정이 필요할 수 있습니다.

- **catalogSource.yaml** 파일은 인덱스 이미지 태그 및 기타 관련 메타데이터로 미리 채워진 **CatalogSource** 오브젝트에 대한 기본 정의입니다. 이 파일은 있는 그대로 사용하거나 카탈로그 소스를 클러스터에 추가하도록 수정할 수 있습니다.



중요

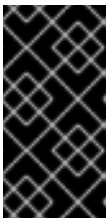
콘텐츠를 로컬 파일에 미러링한 경우 **metadata.name** 필드에서 .name 필드에서 백슬래시(/) 문자를 제거하려면 **catalogSource.yaml** 파일을 수정해야 합니다. 그러지 않으면 오브젝트 생성을 시도할 때 "잘못된 리소스 이름" 오류로 인해 실패합니다.

- **imageContentSourcePolicy.yaml** 파일은 Operator 매니페스트에 저장된 이미지 참조와 미러링된 레지스트리 간에 변환하도록 노드를 구성할 수 있는 **ImageContentSourcePolicy** 오브젝트를 정의합니다.

참고

클러스터에서 **ImageContentSourcePolicy** 오브젝트를 사용하여 저장소 미러링을 구성하는 경우 미러링된 레지스트리에 대한 글로벌 풀 시크릿만 사용할 수 있습니다. 프로젝트에 풀 시크릿을 추가할 수 없습니다.

- **mapping.txt** 파일에는 모든 소스 이미지와 대상 레지스트리에서 매핑할 위치가 포함되어 있습니다. 이 파일은 **oc image mirror** 명령과 호환되며 미러링 구성을 추가로 사용자 정의하는 데 사용할 수 있습니다.



중요

미러링 프로세스 중에 **--manifests-only** 플래그를 사용한 후 미러링할 패키지 서브 세트를 추가로 트리밍 하려면 **mapping.txt** 파일 수정 및 **oc image mirror** 명령으로 파일을 사용하는 데 대한 OpenShift Container Platform 4.7 설명서의 패키지 매니페스트 형식 카탈로그 이미지 미러링 단계를 참조하십시오.

3.9.4. 설치 후 요구 사항

카탈로그를 미러링한 후 나머지 클러스터 설치를 계속할 수 있습니다. 클러스터 설치가 성공적으로 완료되면 **ImageContentSourcePolicy** 및 **CatalogSource** 오브젝트를 생성하려면 이 프로세스에서 매니페스트 디렉터리를 지정해야 합니다. 이러한 오브젝트는 OperatorHub에서 Operator 설치를 채우고 활성화해야 합니다.

추가 리소스

- 미러링된 Operator 카탈로그에서 OperatorHub 채우기

3.10. 다음 단계

- VMware vSphere, 베어 메탈 또는 Amazon Web Services와 같이 네트워크가 제한된 환경에서 프로비저닝한 인프라에 클러스터를 설치하십시오.

3.11. 추가 리소스

- must-gather 사용에 대한 자세한 내용은 특정 기능에 대한 데이터 수집 을 참조하십시오.

4장. AWS에 설치

4.1. AWS에 설치할 준비

4.1.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비 에 대한 문서를 읽습니다.

4.1.2. AWS에 OpenShift Container Platform을 설치하기 위한 요구사항

AWS(Amazon Web Services)에 OpenShift Container Platform을 설치하려면 먼저 AWS 계정을 생성해야 합니다. 계정, 계정 제한, 계정 권한, IAM 사용자 설정 및 지원되는 AWS 리전 구성에 대한 자세한 내용은 AWS 계정 구성을 참조하십시오.

사용자의 환경에서 클라우드 ID 및 액세스 관리(IAM) API에 액세스할 수 없거나 **kube-system** 네임스페이스에 관리자 수준의 인증 정보 시크릿을 저장하지 않으려면 CCO(Cloud Credential Operator)가 Amazon Web Services Security를 사용하도록 구성하는 등 다른 옵션에 대해 AWS용 IAM 수동 생성을 참조하십시오.

4.1.3. AWS에 OpenShift Container Platform을 설치할 방법 선택

설치 관리자 프로비저닝 또는 사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치할 수 있습니다. 기본 설치 유형은 설치 프로그램이 클러스터의 기본 인프라를 프로비저닝하는 설치 관리자 프로비저닝 인프라를 사용합니다. 제공하는 인프라에 OpenShift Container Platform도 설치할 수 있습니다. 설치 프로그램에서 프로비저닝한 인프라를 사용하지 않는 경우 클러스터 리소스를 직접 관리하고 유지보수해야 합니다.

설치 관리자 프로비저닝 및 사용자 프로비저닝 설치 프로세스에 대한 자세한 내용은 설치 프로세스를 참조하십시오.

4.1.3.1. 설치 관리자 프로비저닝 인프라를 사용하여 클러스터 설치

다음 방법 중 하나를 사용하여 OpenShift Container Platform 설치 프로그램에서 프로비저닝한 AWS 인프라에 컨테이너를 설치할 수 있습니다.

- **AWS에서 클러스터 빠른 설치:** OpenShift Container Platform 설치 프로그램에서 프로비저닝한 AWS 인프라에 OpenShift Container Platform을 설치할 수 있습니다. 기본 구성 옵션을 사용하여 빠르게 클러스터를 설치할 수 있습니다.
- **AWS에 사용자 지정 클러스터 설치**: 설치 프로그램이 프로비저닝하는 AWS 인프라에 사용자 지정 클러스터를 설치할 수 있습니다. 설치 프로그램을 통해 설치 단계에서 일부 사용자 지정을 적용할 수 있습니다. 다른 많은 사용자 정의 옵션은 설치 후사용할 수 있습니다.
- **네트워크 사용자 지정으로 AWS에 클러스터 설치**: 설치 중에 OpenShift Container Platform 네트워크 구성을 사용자 지정할 수 있으므로 클러스터가 기존 IP 주소 할당과 공존하고 네트워크 요구사항을 준수할 수 있습니다.
- **제한된 네트워크의 AWS에 클러스터 설치**: 설치 릴리스 콘텐츠의 내부 미러를 사용하여 설치 프로그램 프로비저닝 인프라에 있는 AWS에 OpenShift Container Platform을 설치할 수 있습니다. 이 방법을 사용하여 소프트웨어 구성 요소를 받기 위해 활성 인터넷 연결이 필요하지 않은 클러스터를 설치할 수 있습니다.

- **기존 VPC에 클러스터 설치**: 기존 AWS VPC(Virtual Private Cloud)에 OpenShift Container Platform을 설치할 수 있습니다. 새 계정 또는 인프라를 생성할 때 제한과 같이 회사의 지침에 따라 설정되는 제약 조건이 있는 경우 이 설치 방법을 사용할 수 있습니다.

- **기존 VPC에 프라이빗 클러스터 설치**: 개인 AWS VPC에 프라이빗 클러스터를 설치할 수 있습니다. 이 방법을 사용하여 인터넷에 표시되지 않는 내부 네트워크에 OpenShift Container Platform을 배포할 수 있습니다.

- **AWS에 있는 클러스터를 정부 또는 시크릿 리전에 설치**: OpenShift Container Platform은 연방, 주, 지역 수준의 정부 기관은 물론 클라우드에서 중요한 워크로드를 실행해야 하는 미국 정부 기관, 계약자, 교육 기관 및 기타 미국 고객을 위해 설계된 AWS 리전에 배포할 수 있습니다.

4.1.3.2. 사용자 프로비저닝 인프라를 사용하여 클러스터 설치

다음 방법 중 하나를 사용하여 프로비저닝하는 AWS 인프라에 클러스터를 설치할 수 있습니다.

- **사용자가 제공하는 AWS 인프라에 클러스터 설치**: 사용자가 제공하는 AWS 인프라에 OpenShift Container Platform을 설치할 수 있습니다. 제공된 CloudFormation 템플릿을 사용하여 OpenShift Container Platform 설치에 필요한 각 구성 요소를 나타내는 AWS 리소스 스택을 생성할 수 있습니다.

- **사용자 프로비저닝 인프라가 있는 제한된 네트워크의 AWS에 클러스터 설치**: 설치 릴리스 콘텐츠의 내부 미러를 사용하여 사용자가 제공하는 AWS에 OpenShift Container Platform을 설치할 수 있습니다. 이 방법을 사용하여 소프트웨어 구성 요소를 받기 위해 활성 인터넷 연결이 필요하지 않은 클러스터를 설치할 수 있습니다. 이 설치 방법을 사용하여 클러스터가 외부 콘텐츠에 대해 조직의 제어 조건을 충족하는 컨테이너 이미지만 사용하도록 할 수 있습니다. 미러링된 콘텐츠를 사용하여 OpenShift Container Platform 클러스터를 설치할 수는 있지만 AWS API를 사용하려면 여전히 클러스터에 인터넷 액세스가 필요합니다.

4.1.4. 다음 단계

- AWS 계정 구성

4.2. AWS 계정 구성

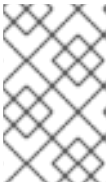
OpenShift Container Platform을 설치하려면 먼저 AWS(Amazon Web Services) 계정을 구성해야 합니다.

4.2.1. 경로 53 구성

OpenShift Container Platform을 설치하려면 사용하는 AWS(Amazon Web Services) 계정에 Route 53 서비스의 전용 공개 호스팅 영역이 있어야 합니다. 도메인에 대한 권한도 이 영역에 있어야 합니다. Route 53 서비스는 클러스터와의 외부 연결에 필요한 클러스터 DNS 확인 및 이름 조회 기능을 제공합니다.

프로세스

1. 도메인 또는 하위 도메인과 등록 기관을 식별합니다. 기존 도메인 및 등록 기관을 이전하거나 AWS 또는 다른 소스를 통해 새 도메인을 구입할 수 있습니다.



참고

AWS를 통해 새 도메인을 구입하는 경우 관련 DNS 변경사항이 전파되는 데 시간이 걸립니다. AWS를 통한 도메인 구입에 대한 자세한 내용은 AWS 문서에서 Amazon Route 53을 사용하여 도메인 이름 등록 을 참조하십시오.

2. 기존 도메인과 등록 기관을 사용하는 경우에는 해당 DNS를 AWS로 마이그레이션합니다. AWS 문서의 Amazon Route 53을 기존 도메인의 DNS 서비스로 지정 을 참조하십시오.

3. 도메인 또는 하위 도메인의 공개 호스팅 영역을 생성합니다. AWS 문서의 공개 호스팅 영역 생성을 참조하십시오.
적절한 루트 도메인(예: **openshiftcorp.com**) 또는 하위 도메인(예: **clusters.openshiftcorp.com)**을 사용합니다.

4. 호스팅 영역 레코드에서 권한이 있는 새 이름 서버를 추출합니다. AWS 문서의 공개 호스팅 영역의 이름 서버 가져오기를 참조하십시오.

5. 도메인에서 사용하는 AWS Route 53 이름 서버의 등록 기관 레코드를 업데이트합니다. 예를 들어 도메인을 다른 계정의 Route 53 서비스에 등록한 경우 AWS 문서의 이름 서버 또는 글루 레코드 추가 또는 변경 항목을 참조하십시오.

6. 하위 도메인을 사용하는 경우 해당 위임 레코드를 상위 도메인에 추가합니다. 이렇게 하면 Amazon Route 53에 하위 도메인에 대한 책임이 부여됩니다. 상위 도메인의 DNS 공급자에 의해 요약된 위임 프로세스를 따릅니다. 상위 수준 프로세스의 예는 AWS 문서에서 상위 도메인을 마이그레이션하지 않고 Amazon Route 53을 DNS 서비스로 사용하여 하위 도메인 생성을 참조하십시오.

4.2.1.1. AWS Route 53에 대한 Ingress Operator 엔드 포인트 구성

Amazon Web Services (AWS) GovCloud (US) US-West 또는 US-East 리전에 설치하는 경우 Ingress Operator는 Route53 및 태그 지정 API 클라이언트에 **us-gov-west-1** 리전을 사용합니다.

Ingress Operator는 'us-gov-east-1' 문자열을 포함하는 태그 지정 사용자 지정 엔드 포인트가 구성된 경우 **https://tagging.us-gov-west-1.amazonaws.com** 을 태그 지정 API 엔드 포인트로 사용합니다.

AWS GovCloud (US) 엔드 포인트에 대한 자세한 내용은 GovCloud (US)에 대한 AWS 설명서에서 Service Endpoints를 참조하십시오.

중요

us-gov-east-1 리전에 설치할 때 AWS GovCloud에서 연결 해제된 개인 설치는 지원되지 않습니다.

Route 53 구성의 예

```
platform:
  aws:
    region: us-gov-west-1
    serviceEndpoints:
    - name: ec2
      url: https://ec2.us-gov-west-1.amazonaws.com
    - name: elasticloadbalancing
      url: https://elasticloadbalancing.us-gov-west-1.amazonaws.com
    - name: route53
      url: https://route53.us-gov.amazonaws.com 1
    - name: tagging
      url: https://tagging.us-gov-west-1.amazonaws.com 2
```

1 Route 53의 기본값은 두 AWS GovCloud (US) 리전 모두에서 **https://route53.us-gov.amazonaws.com**입니다.

2 미국 서부 (US-West) 지역에만 태그 지정을 위한 엔드 포인트가 있습니다. 클러스터가 다른 지역에 있는 경우 이 매개 변수를 생략하십시오.

4.2.2. AWS 계정 제한

OpenShift Container Platform 클러스터는 여러 AWS(Amazon Web Services) 구성 요소를 사용하며 기본 서비스 제한이 OpenShift Container Platform 클러스터 설치하는 데 영향을 미칩니다. 특정 클러스터 구성을 사용하거나 특정 AWS 리전에 클러스터를 배포하거나 사용자 계정에서 여러 클러스터를 실행하는 경우 AWS 계정의 추가 리소스를 요청해야 할 수 있습니다.

다음 표에는 해당 제한이 OpenShift Container Platform 클러스터를 설치하고 실행하는 데 영향을 미칠 수 있는 AWS 구성 요소가 요약되어 있습니다.

구성 요소	기본적으로 사용 가능한 클러스터 수	기본 AWS 제한	설명
인스턴스 제한	변동 가능	변동 가능	기본적으로 각 클러스터는 다음 인스턴스를 생성합니다. • 설치 후 제거되는 하나의 부트스트랩 시스템 • 컨트롤 플레인 노드 세 개 • 작업자 노드 세 개 이러한 인스턴스 유형 수는 새 계정의 기본 제한 내에 있습니다. 더 많은 작업자 노드를 배포하거나 자동 크기 조정을 활성화하거나 대규모 워크로드를 배포하거나 다른 인스턴스 유형을 사용하려면 계정 제한을 검토하여 클러스터가 필요한 시스템을 배포할 수 있는지 확인합니다. 대부분의 리전에서 부트스트랩 및 작업자 시스템은 **m4.large** 시스템을 사용하고 컨트롤 플레인 시스템은 **m4.xlarge** 인스턴스를 사용합니다. 이러한 인스턴스 유형을 지원하지 않는 모든 리전을 포함한 일부 리전에서는 **m5.large** 및 **m5.xlarge** 인스턴스를 대신 사용합니다.

구성 요소	기본적으로 사용 가능한 클러스터 수	기본 AWS 제한	설명
탄력적 IP(EIP)	0 ~1	계정 당 EIP 5개	설치 프로그램은 클러스터를 고가용성 구성으로 프로비저닝하기 위해 각각의 리전 내 가용성 영역의 퍼블릭 및 프라이빗 서브넷을 만듭니다. 프라이빗 서브넷마다 NAT 게이트웨이가 필요하며 NAT 게이트웨이마다 개별 탄력적 IP가 필요합니다. 각 리전의 가용성 영역 수를 판별하려면 AWS 영역 지도를 검토합니다. 기본 고가용성을 활용하려면 세 개 이상의 가용성 영역이 있는 리전에 클러스터를 설치합니다. 여섯 개 이상의 가용성 영역이 있는 리전에 클러스터를 설치하려면 EIP 제한을 늘려야 합니다.  **중요** **us-east-1** 리전을 사용하려면 계정의 EIP 제한을 늘려야 합니다.
가상 사설 클라우드(VPC)	5	리전당 VPC 5개	각 클러스터마다 자체 VPC를 생성합니다.
탄력적 로드 밸런싱(ELB/NLB)	3	리전당 20개	기본적으로 각 클러스터는 마스터 API 서버용 내부 및 외부 네트워크 로드 밸런서를 생성하고 라우터용 단일 클래식 탄력적 로드 밸런서를 생성합니다. **LoadBalancer** 유형으로 Kubernetes **Service** 개체를 더 배포하면 추가 로드 밸런서가 생성됩니다.
NAT 게이트웨이	5	가용성 영역당 5개	클러스터는 각 가용성 영역에 하나의 NAT 게이트웨이를 배포합니다.
탄력적 네트워크 인터페이스(ENI)	12개 이상	리전당 350개	기본 설치는 21개의 ENI와 함께 리전 내 각 가용성 영역마다 하나의 ENI를 생성합니다. 예를 들어 **us-east-1** 리전에는 여섯 개의 가용성 영역이 있으므로 해당 영역에 배포되는 클러스터는 27개의 ENI를 사용합니다. 각 리전의 가용성 영역 수를 판별하려면 AWS 영역 지도를 검토합니다. 클러스터 사용 및 배포된 워크로드에 의해 생성되는 추가 시스템 및 탄력적 로드 밸런서마다 추가 ENI가 생성됩니다.
VPC 게이트웨이	20	계정당 20개	각 클러스터는 S3 액세스를 위한 단일 VPC 게이트웨이를 생성합니다.
S3 버킷	99	계정당 버킷 100개	설치 프로세스에서 임시 버킷을 생성하고 각 클러스터의 레지스트리 구성 요소가 버킷을 생성하므로 AWS 계정당 OpenShift Container Platform 클러스터를 99개만 생성할 수 있습니다.

구성 요소	기본적으로 사용 가능한 클러스터 수	기본 AWS 제한	설명
보안 그룹	250	계정당 2,500개	클러스터마다 10개의 개별 보안 그룹을 생성합니다.

4.2.3. IAM 사용자에게 필요한 AWS 권한

참고

기본 클러스터 리소스를 삭제하려면 IAM 사용자에게 **us-east-1** 리전에 권한 **태그:GetResources** 가 있어야 합니다. AWS API 요구 사항의 일부로 OpenShift Container Platform 설치 프로그램은 이 리전에서 다양한 작업을 수행합니다.

AWS(Amazon Web Services)에서 생성되는 IAM 사용자에게 **AdministratorAccess** 정책을 연결하면 해당 사용자에게 필요한 모든 권한이 부여됩니다. OpenShift Container Platform 클러스터의 모든 구성 요소를 배포하려면 IAM 사용자에게 다음과 같은 권한이 필요합니다.

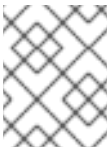
예 4.1. 설치에 필요한 EC2 권한

- **ec2:AuthorizeSecurityGroupEgress**
- **ec2:AuthorizeSecurityGroupIngress**
- **ec2:CopyImage**
- **ec2:CreateNetworkInterface**
- **ec2:AttachNetworkInterface**
- **ec2:CreateSecurityGroup**
- **ec2:CreateTags**
- **ec2:CreateVolume**
- **ec2:DeleteSecurityGroup**
- **ec2:DeleteSnapshot**
- **ec2:DeleteTags**
- **ec2:DeregisterImage**
- **ec2:DescribeAccountAttributes**
- **ec2:DescribeAddresses**
- **ec2:DescribeAvailabilityZones**
- **ec2:DescribeDhcpOptions**
- **ec2:DescribeImages**

- **ec2:DescribeInstanceAttribute**
- **ec2:DescribeInstanceCreditSpecifications**
- **ec2:DescribeInstances**
- **ec2:DescribeInstanceTypes**
- **ec2:DescribeInternetGateways**
- **ec2:DescribeKeyPairs**
- **ec2:DescribeNatGateways**
- **ec2:DescribeNetworkAcls**
- **ec2:DescribeNetworkInterfaces**
- **ec2:DescribePrefixLists**
- **ec2:DescribeRegions**
- **ec2:DescribeRouteTables**
- **ec2:DescribeSecurityGroups**
- **ec2:DescribeSubnets**
- **ec2:DescribeTags**
- **ec2:DescribeVolumes**
- **ec2:DescribeVpcAttribute**
- **ec2:DescribeVpcClassicLink**
- **ec2:DescribeVpcClassicLinkDnsSupport**
- **ec2:DescribeVpcEndpoints**
- **ec2:DescribeVpcs**
- **ec2:GetEbsDefaultKmsKeyId**
- **ec2:ModifyInstanceAttribute**
- **ec2:ModifyNetworkInterfaceAttribute**
- **ec2:RevokeSecurityGroupEgress**
- **ec2:RevokeSecurityGroupIngress**
- **ec2:RunInstances**
- **ec2:TerminateInstances**

예 4.2. 설치 과정에서 네트워크 리소스를 생성하는 데 필요한 권한

- **ec2:AllocateAddress**
- **ec2:AssociateAddress**
- **ec2:AssociateDhcpOptions**
- **ec2:AssociateRouteTable**
- **ec2:AttachInternetGateway**
- **ec2:CreateDhcpOptions**
- **ec2:CreateInternetGateway**
- **ec2:CreateNatGateway**
- **ec2:CreateRoute**
- **ec2:CreateRouteTable**
- **ec2:CreateSubnet**
- **ec2:CreateVpc**
- **ec2:CreateVpcEndpoint**
- **ec2:ModifySubnetAttribute**
- **ec2:ModifyVpcAttribute**



참고

기존 VPC를 사용하는 경우 네트워크 리소스를 생성하기 위해 계정에 이러한 권한이 필요하지 않습니다.

예 4.3. 설치에 필요한 Elastic Load Balancing 권한(ELB)

- **elasticloadbalancing:AddTags**
- **elasticloadbalancing:ApplySecurityGroupsToLoadBalancer**
- **elasticloadbalancing:AttachLoadBalancerToSubnets**
- **elasticloadbalancing:ConfigureHealthCheck**
- **elasticloadbalancing:CreateLoadBalancer**
- **elasticloadbalancing:CreateLoadBalancerListeners**
- **elasticloadbalancing:DeleteLoadBalancer**
- **elasticloadbalancing:DeregisterInstancesFromLoadBalancer**

- **elasticloadbalancing:DescribeInstanceHealth**
- **elasticloadbalancing:DescribeLoadBalancerAttributes**
- **elasticloadbalancing:DescribeLoadBalancers**
- **elasticloadbalancing:DescribeTags**
- **elasticloadbalancing:ModifyLoadBalancerAttributes**
- **elasticloadbalancing:RegisterInstancesWithLoadBalancer**
- **elasticloadbalancing:SetLoadBalancerPoliciesOfListener**

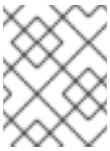
예 4.4. 설치에 필요한 Elastic Load Balancing 권한(ELBv2)

- **elasticloadbalancing:AddTags**
- **elasticloadbalancing:CreateListener**
- **elasticloadbalancing:CreateLoadBalancer**
- **elasticloadbalancing:CreateTargetGroup**
- **elasticloadbalancing:DeleteLoadBalancer**
- **elasticloadbalancing:DeregisterTargets**
- **elasticloadbalancing:DescribeListeners**
- **elasticloadbalancing:DescribeLoadBalancerAttributes**
- **elasticloadbalancing:DescribeLoadBalancers**
- **elasticloadbalancing:DescribeTargetGroupAttributes**
- **elasticloadbalancing:DescribeTargetHealth**
- **elasticloadbalancing:ModifyLoadBalancerAttributes**
- **elasticloadbalancing:ModifyTargetGroup**
- **elasticloadbalancing:ModifyTargetGroupAttributes**
- **elasticloadbalancing:RegisterTargets**

예 4.5. 설치에 필요한 IAM 권한

- **iam:AddRoleToInstanceProfile**
- **iam:CreateInstanceProfile**
- **iam:CreateRole**
- **iam:DeleteInstanceProfile**

- **iam:DeleteRole**
- **iam:DeleteRolePolicy**
- **iam:GetInstanceProfile**
- **iam:GetRole**
- **iam:GetRolePolicy**
- **iam:GetUser**
- **iam:ListInstanceProfilesForRole**
- **iam:ListRoles**
- **iam:ListUsers**
- **iam:PassRole**
- **iam:PutRolePolicy**
- **iam:RemoveRoleFromInstanceProfile**
- **iam:SimulatePrincipalPolicy**
- **iam:TagRole**



참고

AWS 계정에서 탄력적 로드 밸런서 (ELB)를 생성하지 않은 경우 IAM 사용자에게 **iam:CreateServiceLinkedRole** 권한이 필요합니다.

예 4.6. 설치에 필요한 Route 53 권한

- **route53:ChangeResourceRecordSets**
- **route53:ChangeTagsForResource**
- **route53:CreateHostedZone**
- **route53:DeleteHostedZone**
- **route53:GetChange**
- **route53:GetHostedZone**
- **route53:ListHostedZones**
- **route53:ListHostedZonesByName**
- **route53:ListResourceRecordSets**
- **route53:ListTagsForResource**
- **route53:UpdateHostedZoneComment**

예 4.7. 설치에 필요한 S3 권한

- **s3:CreateBucket**
- **s3:DeleteBucket**
- **s3:GetAccelerateConfiguration**
- **s3:GetBucketAcl**
- **s3:GetBucketCors**
- **s3:GetBucketLocation**
- **s3:GetBucketLogging**
- **s3:GetBucketObjectLockConfiguration**
- **s3:GetBucketReplication**
- **s3:GetBucketRequestPayment**
- **s3:GetBucketTagging**
- **s3:GetBucketVersioning**
- **s3:GetBucketWebsite**
- **s3:GetEncryptionConfiguration**
- **s3:GetLifecycleConfiguration**
- **s3:GetReplicationConfiguration**
- **s3:ListBucket**
- **s3:PutBucketAcl**
- **s3:PutBucketTagging**
- **s3:PutEncryptionConfiguration**

예 4.8. 클러스터 Operator에 필요한 S3 권한

- **s3:DeleteObject**
- **s3:GetObject**
- **s3:GetObjectAcl**
- **s3:GetObjectTagging**
- **s3:GetObjectVersion**

- **s3:PutObject**
- **s3:PutObjectAcl**
- **s3:PutObjectTagging**

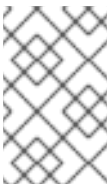
예 4.9. 기본 클러스터 리소스를 삭제하는 데 필요한 권한

- **autoscaling:DescribeAutoScalingGroups**
- **ec2:DeleteNetworkInterface**
- **ec2:DeleteVolume**
- **elasticloadbalancing:DeleteTargetGroup**
- **elasticloadbalancing:DescribeTargetGroups**
- **iam:DeleteAccessKey**
- **iam:DeleteUser**
- **iam:ListAttachedRolePolicies**
- **iam:ListInstanceProfiles**
- **iam:ListRolePolicies**
- **iam:ListUserPolicies**
- **s3:DeleteObject**
- **s3:ListBucketVersions**
- **tag:GetResources**

예 4.10. 네트워크 리소스를 삭제하는 데 필요한 권한

- **ec2:DeleteDhcpOptions**
- **ec2:DeleteInternetGateway**
- **ec2:DeleteNatGateway**
- **ec2:DeleteRoute**
- **ec2:DeleteRouteTable**
- **ec2:DeleteSubnet**
- **ec2:DeleteVpc**
- **ec2:DeleteVpcEndpoints**
- **ec2:DetachInternetGateway**

- **ec2:DisassociateRouteTable**
- **ec2:ReleaseAddress**
- **ec2:ReplaceRouteTableAssociation**



참고

기존 VPC를 사용하는 경우 네트워크 리소스를 삭제하기 위해 계정에 이러한 권한이 필요하지 않습니다. 대신 사용자 계정에는 네트워크 리소스를 삭제하기 위한 **tag:UntagResources** 권한만 필요합니다.

예 4.11. 공유 인스턴스 역할이 있는 클러스터를 삭제하는 데 필요한 권한

- **iam:UntagRole**

예 4.12. 매니페스트를 생성하는 데 필요한 추가 IAM 및 S3 권한

- **iam:DeleteAccessKey**
- **iam:DeleteUser**
- **iam:DeleteUserPolicy**
- **iam:GetUserPolicy**
- **iam:ListAccessKeys**
- **iam:PutUserPolicy**
- **iam:TagUser**
- **s3:PutBucketPublicAccessBlock**
- **s3:GetBucketPublicAccessBlock**
- **s3:PutLifecycleConfiguration**
- **s3:HeadBucket**
- **s3:ListBucketMultipartUploads**
- **s3:AbortMultipartUpload**

참고

mint 모드를 사용하여 클라우드 공급자 인증 정보를 관리하는 경우 IAM 사용자에게 **iam:CreateAccessKey** 및 **iam:CreateUser** 권한이 필요합니다.

예 4.13. 인스턴스에 대한 선택적 권한 및 설치에 대한 할당량 검사

- **ec2:DescribeInstanceTypeOfferings**
- **servicequotas:ListAWSDefaultServiceQuotas**

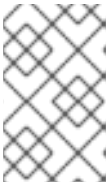
4.2.4. IAM 사용자 생성

각 AWS(Amazon Web Services) 계정에는 계정을 생성하는 데 사용한 이메일 주소를 기반으로 하는 루트 사용자 계정이 포함되어 있습니다. 이 계정은 권한이 높은 계정이므로 초기 계정 및 결제 구성, 초기 사용자 집합 생성 및 계정 보안 용도로만 사용하는 것이 좋습니다.

OpenShift Container Platform을 설치하기 전에 보조 IAM 관리자를 생성합니다. AWS 문서의 AWS 계정에서 IAM 사용자 생성 프로시저를 완료하면 다음 옵션을 설정합니다.

프로세스

1. IAM 사용자 이름을 지정하고 **Programmatic access**를 선택합니다.
2. **AdministratorAccess** 정책을 연결하여 클러스터를 생성할 수 있는 충분한 권한이 계정에 있는지 확인합니다. 이 정책은 각 OpenShift Container Platform 구성 요소에 자격 증명을 부여하는 기능을 클러스터에 제공합니다. 클러스터는 필요한 자격 증명만 구성 요소에 부여합니다.



참고

필요한 모든 AWS 권한을 부여하는 정책을 생성하여 사용자에게 연결할 수는 있지만 바람직한 옵션은 아닙니다. 클러스터는 개별 구성 요소에 추가 자격 증명을 부여할 수 없으므로 모든 구성 요소가 동일한 자격 증명을 사용합니다.

3. 선택사항: 태그를 첨부하여 사용자에게 메타데이터를 추가합니다.
4. 지정한 사용자 이름에 **AdministratorAccess** 정책이 부여되었는지 확인합니다.
5. 액세스 키 ID 및 시크릿 액세스 키 값을 기록합니다. 설치 프로그램을 실행하도록 로컬 시스템을 구성할 때 이 값을 사용해야 합니다.

중요

다단계 인증 장치를 사용하여 AWS에 인증하는 동안 생성한 임시 세션 토큰은 클러스터를 배포할 때 사용할 수 없습니다. 클러스터는 클러스터의 전체 수명 동안 현재 AWS 자격 증명을 계속 사용하여 AWS 리소스를 생성하므로 키 기반의 장기 자격 증명을 사용해야 합니다.

추가 리소스

- 설치하기 전에 수동 모드로 CCO(Cloud Credential Operator)를 설정하는 단계는 AWS의 IAM 수동 생성을 참조하십시오. 클라우드 ID 및 액세스 관리(IAM) API에 연결할 수 없는 환경에서 이 모드를 사용하거나 관리자 수준 인증 정보 시크릿을 클러스터 **kube-system** 프로젝트에 저장하지 않도록 합니다.

4.2.5. IAM 정책 및 AWS 인증

기본적으로 설치 프로그램은 클러스터가 작동하는 데 필요한 권한이 있는 부트스트랩, 컨트롤 플레인 및 컴퓨팅 인스턴스에 대한 인스턴스 프로필을 생성합니다.

그러나 고유한 IAM 역할을 생성하여 설치 프로세스의 일부로 지정할 수 있습니다. 클러스터를 배포하거나 설치 후 클러스터를 관리하려면 자체 역할을 지정해야 할 수 있습니다. 예를 들면 다음과 같습니다.

- 조직의 보안 정책에는 클러스터를 설치하기 위해 보다 제한적인 권한 세트를 사용해야 합니다.
- 설치 후 클러스터는 추가 서비스에 액세스해야 하는 Operator로 구성됩니다.

고유한 IAM 역할을 지정하도록 선택하는 경우 다음 단계를 수행할 수 있습니다.

- 기본 정책으로 시작하고 필요에 따라 조정합니다. 자세한 내용은 " IAM 인스턴스 프로필에 대한 기본 권한"을 참조하십시오.
- AWS IAM(Identity and Access Management Access Analyzer)을 사용하여 클러스터의 활동을 기반으로 하는 정책 템플릿을 생성합니다. 자세한 내용은 "AWS IAM Analyzer를 사용하여 정책 템플릿을 생성"을 참조하십시오.

4.2.5.1. IAM 인스턴스 프로파일의 기본 권한

기본적으로 설치 프로그램은 클러스터가 작동하는 데 필요한 권한으로 부트스트랩, 컨트롤 플레인 및 작업자 인스턴스에 대한 IAM 인스턴스 프로필을 생성합니다.

다음 목록은 컨트롤 플레인 및 컴퓨팅 머신에 대한 기본 권한을 지정합니다.

예 4.14. 컨트롤 플레인 인스턴스 프로파일에 대한 기본 IAM 역할 권한

- **ec2:AttachVolume**
- **ec2:AuthorizeSecurityGroupIngress**
- **ec2:CreateSecurityGroup**
- **ec2:CreateTags**
- **ec2:CreateVolume**
- **ec2:DeleteSecurityGroup**
- **ec2:DeleteVolume**
- **ec2:Describe***
- **ec2:DetachVolume**
- **ec2:ModifyInstanceAttribute**
- **ec2:ModifyVolume**
- **ec2:RevokeSecurityGroupIngress**
- **elasticloadbalancing:AddTags**
- **elasticloadbalancing:AttachLoadBalancerToSubnets**
- **elasticloadbalancing:ApplySecurityGroupsToLoadBalancer**
- **elasticloadbalancing:CreateListener**

- **elasticloadbalancing:CreateLoadBalancer**
- **elasticloadbalancing:CreateLoadBalancerPolicy**
- **elasticloadbalancing:CreateLoadBalancerListeners**
- **elasticloadbalancing:CreateTargetGroup**
- **elasticloadbalancing:ConfigureHealthCheck**
- **elasticloadbalancing:DeleteListener**
- **elasticloadbalancing:DeleteLoadBalancer**
- **elasticloadbalancing:DeleteLoadBalancerListeners**
- **elasticloadbalancing:DeleteTargetGroup**
- **elasticloadbalancing:DeregisterInstancesFromLoadBalancer**
- **elasticloadbalancing:DeregisterTargets**
- **elasticloadbalancing:Describe***
- **elasticloadbalancing:DetachLoadBalancerFromSubnets**
- **elasticloadbalancing:ModifyListener**
- **elasticloadbalancing:ModifyLoadBalancerAttributes**
- **elasticloadbalancing:ModifyTargetGroup**
- **elasticloadbalancing:ModifyTargetGroupAttributes**
- **elasticloadbalancing:RegisterInstancesWithLoadBalancer**
- **elasticloadbalancing:RegisterTargets**
- **elasticloadbalancing:SetLoadBalancerPoliciesForBackendServer**
- **elasticloadbalancing:SetLoadBalancerPoliciesOfListener**
- **kms:DescribeKey**

예 4.15. 컴퓨팅 인스턴스 프로파일에 대한 기본 IAM 역할 권한

- **ec2:DescribeInstances**
- **ec2:DescribeRegions**

4.2.5.2. 기존 IAM 역할 지정

설치 프로그램에서 기본 권한으로 IAM 인스턴스 프로필을 생성하도록 허용하는 대신 **install-config.yaml** 파일을 사용하여 컨트롤 플레인 및 컴퓨팅 인스턴스에 기존 IAM 역할을 지정할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.

프로세스

1. 컨트롤 플레인 시스템의 기존 역할로 **compute.platform.aws.iamRole** 을 업데이트합니다.

   컴퓨팅 인스턴스의 IAM 역할이 있는 샘플 install-config.yaml 파일

   ```
   compute:
   - hyperthreading: Enabled
     name: worker
     platform:
       aws:
         iamRole: ExampleRole
   ```

2. 컴퓨팅 시스템의 기존 역할로 **controlPlane.platform.aws.iamRole** 을 업데이트합니다.

   컨트롤 플레인 인스턴스의 IAM 역할이 있는 샘플 install-config.yaml 파일

   ```
   controlPlane:
     hyperthreading: Enabled
     name: master
     platform:
       aws:
         iamRole: ExampleRole
   ```

3. 파일을 저장하고 OpenShift Container Platform 클러스터를 설치할 때 참조합니다.

추가 리소스

- 클러스터 배포를 참조하십시오.

4.2.5.3. AWS IAM Analyzer를 사용하여 정책 템플릿 생성

컨트롤 플레인 및 컴퓨팅 인스턴스 프로파일에 필요한 최소 권한 세트는 클러스터를 일상 작업에 맞게 구성하는 방법에 따라 다릅니다.

클러스터 인스턴스에 필요한 권한을 결정하는 한 가지 방법은 AWS IAM(Identity and Access Access Analyzer)을 사용하여 정책 템플릿을 생성하는 것입니다.

- 정책 템플릿에는 지정된 기간 동안 클러스터가 사용한 권한이 포함되어 있습니다.
- 그런 다음 템플릿을 사용하여 세분화된 권한으로 정책을 생성할 수 있습니다.

프로세스

전체 프로세스는 다음과 같습니다.

1. CloudTrail이 활성화되어 있는지 확인합니다. CloudTrail은 정책 템플릿을 생성하는 데 필요한 API 호출을 포함하여 AWS 계정의 모든 작업과 이벤트를 기록합니다. 자세한 내용은 CloudTrail 사용에 대한 AWS 설명서를 참조하십시오.

2. 컨트롤 플레인 인스턴스의 인스턴스 프로필과 컴퓨팅 인스턴스의 인스턴스 프로필을 생성합니다. 각 역할에 PowerUserAccess와 같은 허용 정책을 할당해야 합니다. 자세한 내용은 인스턴스 프로필 역할 생성 을 위한 AWS 설명서를 참조하십시오.

3. 개발 환경에 클러스터를 설치하고 필요에 따라 구성합니다. 클러스터가 프로덕션 환경에서 호스팅할 모든 애플리케이션을 배포해야 합니다.

4. 클러스터를 철저하게 테스트합니다. 클러스터를 테스트하면 필요한 모든 API 호출이 기록됩니다.

5. IAM Access Analyzer를 사용하여 각 인스턴스 프로필에 대한 정책 템플릿을 생성합니다. 자세한 내용은 CloudTrail 로그를 기반으로 정책을 생성하는 AWS 설명서를 참조하십시오.

6. 각 인스턴스 프로필에 세분화된 정책을 생성하고 추가합니다.

7. 각 인스턴스 프로필에서 허용 정책을 제거합니다.

8. 새 정책과 함께 기존 인스턴스 프로필을 사용하여 프로덕션 클러스터를 배포합니다.

참고

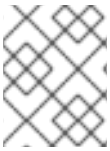
IAM 조건을 정책에 추가하여 보다 제한적이고 조직 보안 요구 사항을 준수할 수 있습니다.

4.2.6. 지원되는 AWS Marketplace 리전

북미에서 제품을 구매한 고객은 AWS Marketplace 이미지를 사용하여 OpenShift Container Platform 클러스터를 설치할 수 있습니다.

이 프로모션은 북미 지역에서만 구매할 수 있지만 다음과 같은 지원되는 파티션에 클러스터를 배포할 수 있습니다.

- public
- GovCloud

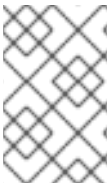


참고

AWS Marketplace 이미지를 사용하여 OpenShift Container Platform 클러스터를 배포하려면 AWS 시크릿 리전 또는 중국 리전에서 지원되지 않습니다.

4.2.7. 지원되는 AWS 리전

OpenShift Container Platform 클러스터를 배포할 수 있는 리전은 다음과 같습니다.



참고

기본 클러스터 리소스를 삭제하려면 IAM 사용자에게 **us-east-1** 리전에 권한 **태그:GetResources** 가 있어야 합니다. AWS API 요구 사항의 일부로 OpenShift Container Platform 설치 프로그램은 이 리전에서 다양한 작업을 수행합니다.

4.2.7.1. AWS 공용 리전

지원되는 AWS 공용 리전은 다음과 같습니다.

- **af-south-1** (케이프타운)

- **ap-east-1** (홍콩)
- **ap-northeast-1** (도쿄)
- **ap-northeast-2** (서울)
- **ap-northeast-3** (오사카)
- **ap-south-1** (뭄바이)
- **ap-southeast-1** (싱가포르)
- **ap-southeast-2** (시드니)
- **ca-central-1** (센트럴)
- **eu-central-1** (프랑크푸르트)
- **eu-north-1** (스톡홀름)
- **eu-south-1** (밀라노)
- **eu-west-1** (아일랜드)
- **eu-west-2** (런던)
- **eu-west-3** (파리)
- **me-south-1** (바레인)
- **sa-east-1** (상파울루)
- **us-east-1** (버지니아 북부)
- **us-east-2** (오하이오)
- **us-west-1** (캘리포니아 북부)
- **us-west-2** (오레곤)

4.2.7.2. AWS GovCloud 리전

다음 AWS GovCloud 리전이 지원됩니다.

- **us-gov-west-1**
- **us-gov-east-1**

4.2.7.3. AWS C2S Secret 리전

us-iso-east-1 리전이 지원됩니다.

4.2.7.4. AWS 중국 리전

다음 AWS 중국 리전이 지원됩니다.

- **cn-north-1(** 베이징)

- **cn-northwest-1** (닝샤)

4.2.8. 다음 단계

- OpenShift Container Platform 클러스터 설치:
  - 설치 관리자가 프로비저닝한 인프라에 기본 옵션으로 클러스터의 빠른 설치
  - 설치 관리자가 프로비저닝한 인프라에 클라우드 사용자 지정으로 클러스터 설치
  - 설치 관리자가 프로비저닝한 인프라에 네트워크 사용자 지정으로 클러스터 설치
  - CloudFormation 템플릿을 사용하여 사용자 프로비저닝 인프라에 클러스터 설치

4.3. AWS의 IAM 수동 생성

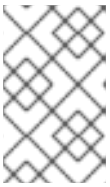
클라우드 ID 및 액세스 관리(IAM) API에 연결할 수 없는 환경에서 또는 관리자가 클러스터 **kube-system** 네임 스페이스에 관리자 수준의 인증 정보 시크릿을 저장하지 않는 것을 선호하는 경우 클러스터를 설치하기 전에 CCO(Cloud Credential Operator)를 수동 모드로 설정할 수 있습니다.

4.3.1. kube-system 프로젝트에 관리자 수준 시크릿을 저장하는 대안

CCO(Cloud Credential Operator)는 클라우드 공급자 인증 정보를 Kubernetes CRD(사용자 지정 리소스 정의)로 관리합니다. **install-config.yaml** 파일에서 **credentialsMode** 매개변수의 다른 값을 설정하여 조직의 보안 요구 사항에 맞게 CCO를 구성할 수 있습니다.

클러스터 **kube-system** 프로젝트에 관리자 수준 인증 정보 시크릿을 저장하지 않으려면 OpenShift Container Platform을 설치할 때 다음 옵션 중 하나를 선택할 수 있습니다.

- **Amazon Web Services 보안 토큰 서비스 사용**:
  CCO 유틸리티 (**ccoctl**)를 사용하여 Amazon Web Services Security Token Service (AWS STS)를 사용하도록 클러스터를 구성할 수 있습니다. CCO 유틸리티를 사용하여 STS의 클러스터를 구성할 때 단기적이고 제한된 권한 보안 인증 정보를 제공하는 IAM 역할을 구성 요소에 할당합니다.

  참고

  이 인증 정보 전략은 새 OpenShift Container Platform 클러스터에 대해서만 지원되며 설치 중에 구성해야 합니다. 이 기능을 사용하기 위해 다른 인증 정보 전략을 사용하는 기존 클러스터를 재구성할 수 없습니다.

- **클라우드 인증 정보를 수동으로 관리**:
  CCO의 **credentialsMode** 매개변수를 수동으로 클라우드 인증 정보를 관리하도록 **Manual**로 설정할 수 있습니다. 수동 모드를 사용하면 각 클러스터 구성 요소에는 클러스터에 관리자 수준 인증 정보를 저장하지 않고 필요한 권한만 보유할 수 있습니다. 환경이 클라우드 공급자 공용 IAM 끝점에 연결되지 않은 경우 이 모드를 사용할 수도 있습니다. 그러나 업그레이드할 때마다 새 릴리스 이미지로 권한을 수동으로 조정해야 합니다. 또한 요청하는 모든 구성 요소에 대한 인증 정보를 수동으로 제공해야 합니다.

- **mint 모드를 사용하여 OpenShift Container Platform을 설치한 후 관리자 수준 인증 정보 시크릿 제거**:
  credentialsMode 매개변수가 **Mint**로 설정된 CCO를 사용하는 경우 OpenShift Container Platform을 설치한 후 관리자 수준 인증 정보를 제거하거나 순환할 수 있습니다. Mint 모드는 CCO의 기본 구성입니다. 이 옵션을 사용하려면 설치 중에 관리자 수준 인증 정보가 있어야 합니

다. 관리자 수준 인증 정보는 설치 중에 일부 권한이 부여된 다른 인증 정보를 Mint하는 데 사용됩니다. 원래 인증 정보 시크릿은 클러스터에 영구적으로 저장되지 않습니다.

참고

z-stream 외 업그레이드 이전에는 관리자 수준 인증 정보를 사용하여 인증 정보 시크릿을 복원해야 합니다. 인증 정보가 없으면 업그레이드가 차단될 수 있습니다.

추가 리소스

- CCO 유틸리티 (**ccoctl**)를 사용하여 AWS STS를 사용하도록 CCO를 구성하는 방법에 대한 자세한 내용은 STS를 사용하여 수동 모드 사용을 참조하십시오.
- OpenShift Container Platform을 설치한 후 관리자 수준 인증 정보 시크릿을 순환하거나 제거하는 방법을 알아보려면 클라우드 공급자 인증 정보 교체 또는 제거 를 참조하십시오.
- 사용 가능한 모든 CCO 인증 정보 모드 및 지원되는 플랫폼에 대한 자세한 내용은 Cloud Credential Operator 정보를 참조하십시오.

4.3.2. 수동으로 IAM 생성

Cloud Credential Operator (CCO)는 클라우드 아이덴티티 및 액세스 관리 (IAM) API에 연결할 수 없는 환경에서 설치하기 전에 수동 모드로 전환할 수 있습니다. 또는 관리자가 클러스터 **kube-system** 네임 스페이스에 관리자 수준의 인증 정보 시크릿을 저장하지 않도록 합니다.

프로세스

1. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행하여 **install-config.yaml** 파일을 생성합니다.

```
$ openshift-install create install-config --dir <installation_directory>
```

   여기서 **<installation_directory>** 는 설치 프로그램이 파일을 생성하는 디렉터리입니다.

2. **install-config.yaml** 구성 파일을 편집하여 **credentialsMode** 매개 변수가 **Manual**로 설정되도록 합니다.

   install-config.yaml 설정 파일 예

```
apiVersion: v1
baseDomain: cluster1.example.com
credentialsMode: Manual (1)
compute:
- architecture: amd64
  hyperthreading: Enabled
...
```

   (1) 이 행은 **credentialsMode** 매개변수를 **Manual**로 설정하기 위해 추가됩니다.

3. 매니페스트를 생성하려면 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행합니다.

```
$ openshift-install create manifests --dir <installation_directory>
```

여기서 **<installation_directory>** 는 설치 프로그램이 파일을 생성하는 디렉터리입니다.

4. 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행하여 **openshift-install** 바이너리가 사용하도록 빌드된 OpenShift Container Platform 릴리스 이미지에 대한 세부 정보를 가져옵니다.

```
$ openshift-install version
```

출력 예

```
release image quay.io/openshift-release-dev/ocp-release:4.y.z-x86_64
```

5. 다음 명령을 실행하여 배포 중인 클라우드를 대상으로 하는 이 릴리스 이미지에서 모든 **CredentialsRequest** 오브젝트를 찾습니다.

```
$ oc adm release extract quay.io/openshift-release-dev/ocp-release:4.y.z-x86_64 \
  --credentials-requests \
  --cloud=aws
```

이 명령을 수행하면 각 **CredentialsRequest** 오브젝트에 대해 YAML 파일이 생성됩니다.

샘플 CredentialsRequest 개체

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component-credentials-request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
    - effect: Allow
      action:
      - iam:GetUser
      - iam:GetUserPolicy
      - iam:ListAccessKeys
      resource: "*"
    ...
```

6. 이전에 생성한 **openshift-install** 매니페스트 디렉터리에 시크릿 YAML 파일을 만듭니다. 시크릿은 각 **CredentialsRequest** 오브젝트의 **spec.secretRef**에 정의된 네임 스페이스 및 시크릿 이름을 사용하여 저장해야 합니다.

보안이 포함된 샘플 CredentialsRequest 오브젝트

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component-credentials-request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
```

```
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
    - effect: Allow
      action:
      - s3:CreateBucket
      - s3:DeleteBucket
      resource: "*"
      ...
  secretRef:
    name: <component-secret>
    namespace: <component-namespace>
  ...
```

샘플 Secret 오브젝트

```
apiVersion: v1
kind: Secret
metadata:
  name: <component-secret>
  namespace: <component-namespace>
data:
  aws_access_key_id: <base64_encoded_aws_access_key_id>
  aws_secret_access_key: <base64_encoded_aws_secret_access_key>
```

중요

수동으로 유지 관리되는 인증 정보를 사용하는 클러스터를 업그레이드하기 전에 CCO가 업그레이드 가능한 상태인지 확인해야 합니다. 자세한 내용은 클라우드 공급자에 대한 설치 콘텐츠의 "수동으로 유지 관리되는 인증 정보를 사용하여 클러스터 업그레이드" 섹션을 참조하십시오.

4.3.3. 수동으로 유지 관리되는 인증 정보로 클러스터 업그레이드

CCO(Cloud Credential Operator) 수동으로 유지 관리되는 인증 정보가 있는 클러스터의 **Upgradable** 상태는 기본적으로 **False** 입니다.

- 마이너 릴리스(예: 4.8에서 4.9로)의 경우 이 상태는 업데이트된 권한을 처리하고 **CloudCredential** 리소스에 주석을 달아 권한이 다음 버전에 필요에 따라 업데이트되었음을 나타낼 때까지 업그레이드되지 않도록 합니다. 이 주석은 **Upgradable** 상태를 **True**로 변경합니다.
- 예를 들어 4.9.0에서 4.9.1으로 z-stream 릴리스의 경우 권한이 추가되거나 변경되지 않으므로 업그레이드가 차단되지 않습니다.

수동으로 유지 관리되는 인증 정보로 클러스터를 업그레이드하기 전에 업그레이드할 릴리스 이미지에 대한 새 인증 정보를 생성해야 합니다. 또한 기존 인증 정보에 필요한 권한을 검토하고 해당 구성 요소에 대한 새 릴리스에 새 권한 요구 사항을 수용해야 합니다.

절차

1. 새 릴리스에 대한 **CredentialsRequest** 사용자 지정 리소스를 추출하고 검사합니다.
   클라우드 공급자용 설치 콘텐츠의 "수동으로 IAM 생성" 섹션에서는 클라우드에 필요한 인증 정보를 획득하고 사용하는 방법을 설명합니다.

2. 클러스터에서 수동으로 유지 관리되는 인증 정보를 업데이트합니다.

   - 새 릴리스 이미지에서 추가한 **CredentialsRequest** 사용자 정의 리소스에 대한 새 시크릿을 생성합니다.
   - 시크릿에 저장된 기존 인증 정보에 대한 **CredentialsRequest** 사용자 정의 리소스가 권한 요구 사항이 변경된 경우 필요에 따라 권한을 업데이트합니다.

3. 모든 보안이 새 릴리스에 대해 올바른 경우 클러스터를 업그레이드할 준비가 되었음을 나타냅니다.

   a. **cluster-admin** 역할의 사용자로 OpenShift Container Platform CLI에 로그인합니다.

   b. **CloudCredential** 리소스를 편집하여 **metadata** 필드 내에 **upgradeable-to** 주석을 추가합니다.

   ```
   $ oc edit cloudcredential cluster
   ```

   추가할 텍스트

   ```
   ...
     metadata:
       annotations:
         cloudcredential.openshift.io/upgradeable-to: <version_number>
   ...
   ```

   여기서 **<version_number>**는 **x.y.z** 형식으로 업그레이드할 버전입니다. 예를 들어 OpenShift Container Platform 4.8.2 의 경우 **4.8.2**입니다.

   주석을 추가한 후 업그레이드 가능 상태가 변경되는 데 몇 분이 소요될 수 있습니다.

4. CCO를 업그레이드할 수 있는지 확인합니다.

   a. 웹 콘솔의 **관리자** 화면에서 **관리자 → 클러스터 설정**으로 이동합니다.

   b. CCO 상태 세부 정보를 보려면 **Cluster Operators** 목록에서 **cloud-credential**을 클릭합니다.

   c. **Conditions** 섹션의 **Upgradeable** 상태가 **False**인 경우 **upgradeable-to 주석**에 오타 오류가 없는지 확인합니다.

Conditions 섹션의 **Upgradeable** 상태가 **True** 이면 OpenShift Container Platform 업그레이드를 시작할 수 있습니다.

```
:_content-type: CONCEPT
```

4.3.4. Mint 모드

Mint 모드는 OpenShift Container Platform의 기본 CCO(Cloud Credential Operator) 인증 정보 모드입니다. 이 모드에서 CCO는 제공된 관리자 수준 클라우드 인증 정보를 사용하여 클러스터를 실행합니다. Mint 모드는 AWS 및 GCP에서 지원됩니다.

mint 모드에서 **admin** 인증 정보가 **kube-system** 네임 스페이스에 저장된 다음 CCO에서 클러스터의 **CredentialsRequest** 오브젝트를 처리하고 각각에 대해 특정 권한이 있는 사용자를 만드는 데 사용됩니다.

mint 모드의 장점은 다음과 같습니다.

- 각 클러스터 구성 요소에는 필요한 권한만 있습니다.
- 추가 인증 정보 또는 권한을 포함한 클라우드 인증 정보에 대해 지속적인 자동 조정이 이루어집니다. 이는 업그레이드에 필요할 수 있습니다.

한 가지 단점은 mint 모드에 클러스터 **kube-system** 시크릿에 대한 **admin** 인증 정보 저장소가 필요하다는 것입니다.

4.3.5. 관리자 수준 인증 정보를 제거하거나 교체하는 Mint 모드

현재 이 모드는 AWS 및 GCP에서만 지원됩니다.

이 모드에서 사용자는 일반 Mint 모드와 마찬가지로 관리자 수준 인증 정보를 사용하여 OpenShift Container Platform을 설치합니다. 그러나 이 프로세스에서는 설치 후 클러스터에서 관리자 수준 인증 정보 시크릿을 제거합니다.

관리자는 모든 **CredentialsRequest** 오브젝트에 필요한 사용 권한이 있고 따라서 내용을 변경해야 하는 경우가 아니면 관리자 수준의 자격 증명이 필요 없음을 확인하기 위해 Cloud Credential Operator가 읽기 전용 자격 증명을 스스로 요청하도록 할 수 있습니다. 연결된 인증 정보를 제거한 후 원하는 경우 기본 클라우드에서 삭제하거나 비활성화할 수 있습니다.

참고

z-stream 외 업그레이드 이전에는 관리자 수준 인증 정보를 사용하여 인증 정보 시크릿을 복원해야 합니다. 인증 정보가 없으면 업그레이드가 차단될 수 있습니다.

관리자 수준 인증 정보는 클러스터에 영구적으로 저장되지 않습니다.

이러한 단계를 따르려면 짧은 기간 동안 클러스터의 관리자 수준 인증 정보가 필요합니다. 또한 각 업그레이드에 대해 관리자 수준 인증 정보로 시크릿을 수동으로 복원해야 합니다.

4.3.6. 다음 단계

- OpenShift Container Platform 클러스터 설치:
  - 설치 관리자가 프로비저닝한 인프라에 기본 옵션으로 AWS에 빠르게 클러스터 설치
  - 설치 관리자가 프로비저닝한 인프라에 클라우드 사용자 지정으로 클러스터 설치
  - 설치 관리자가 프로비저닝한 인프라에 네트워크 사용자 지정으로 클러스터 설치
  - CloudFormation 템플릿을 사용하여 사용자 프로비저닝 인프라에 클러스터 설치

4.4. AWS에 빠르게 클러스터 설치

OpenShift Container Platform 4.9 버전에서는 기본 구성 옵션을 사용하는 클러스터를 AWS(Amazon Web Services)에 설치할 수 있습니다.

4.4.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비 에 대한 문서를 읽습니다.

- 클러스터를 호스팅할 AWS 계정을 구성 했습니다.

중요

컴퓨터에 AWS 프로필이 저장되어 있는 경우 다단계 인증 장치를 사용하는 동안 생성한 임시 세션 토큰을 해당 프로필이 사용해서는 안 됩니다. 클러스터는 클러스터의 전체 수명 동안 현재 AWS 자격 증명을 계속 사용하여 AWS 리소스를 생성하므로 키 기반의 장기 자격 증명을 사용해야 합니다. 적절한 키를 생성하려면 AWS 문서의 IAM 사용자의 액세스 키 관리 를 참조하십시오. 설치 프로그램을 실행할 때 키를 제공할 수 있습니다.

- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성 해야 합니다.

- 사용자의 환경에서 클라우드 ID 및 액세스 관리(IAM) API에 액세스할 수 없거나 **kube-system** 네임스페이스에 관리자 수준의 인증 정보 시크릿을 저장하지 않으려면 IAM 인증 정보 수동 생성 및 유지 관리를 참조하십시오. 수동 모드는 클라우드 IAM API에 연결할 수 없는 환경에서도 사용할 수 있습니다.

4.4.2. OpenShift Container Platform 용 인터넷 액세스

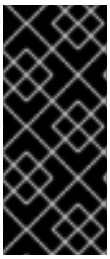
OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.

- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.

- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.



중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

4.4.3. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 **core** 사용자의 **~/.ssh/authorized_keys** 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 **core**로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. **./openshift-install gather** 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

참고

AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: **~/.ssh/id_ed25519**)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 **'~/.ssh** 디렉터리에 있는지 확인하십시오.

참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 **ed25519** 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 **rsa** 또는 **ecdsa** 알고리즘을 사용하는 키를 생성합니다.

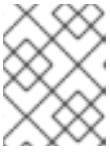
2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 **~/.ssh/id_ed25519.pub** 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 **./openshift-install gather** 명령을 사용하려는 경우 필요합니다.



참고

일부 배포에서는 **~/.ssh/id_rsa** 및 **~/.ssh/id_dsa**와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. **ssh-agent** 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. **ssh-agent**에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: **~/.ssh/id_ed25519**).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

4.4.4. 설치 프로그램 받기

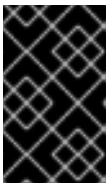
OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

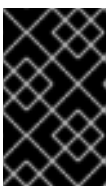
프로세스

1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.
2. 인프라 공급자를 선택합니다.
3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.



중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.



중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 OpenShift Container Platform 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

4.4.5. 클러스터 배포

호환되는 클라우드 플랫폼에 OpenShift Container Platform을 설치할 수 있습니다.

중요

최초 설치 과정에서 설치 프로그램의 **create cluster** 명령을 한 번만 실행할 수 있습니다.

사전 요구 사항

- 클러스터를 호스팅하는 클라우드 플랫폼으로 계정을 구성합니다.
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

프로세스

1. 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터 배포를 초기화합니다.

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>**는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.

2 다른 설치 세부 사항을 보려면 **info** 대신 **warn**, **debug** 또는 **error**를 지정합니다.

중요

비어 있는 디렉터리를 지정합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

화면에 나타나는 지시에 따라 필요한 값을 입력합니다.

a. 선택사항: 클러스터 시스템에 액세스하는 데 사용할 SSH 키를 선택합니다.

참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

b. 대상 플랫폼으로 **aws**를 선택합니다.

c. 컴퓨터에 AWS(Amazon Web Services) 프로필이 저장되어 있지 않은 경우 설치 프로그램을 실행하도록 구성한 사용자의 AWS 액세스 키 ID와 시크릿 액세스 키를 입력합니다.

참고

AWS 액세스 키 ID와 시크릿 액세스 키는 설치 호스트에 있는 현재 사용자의 홈 디렉터리에서 **~/.aws/credentials**에 저장됩니다. 내보낸 프로필의 인증 정보가 파일에 없으면 설치 프로그램에서 인증 정보에 대한 메시지를 표시합니다. 설치 프로그램에 사용자가 제공하는 인증 정보는 파일에 저장됩니다.

d. 클러스터를 배포할 AWS 리전을 선택합니다.

e. 클러스터에 대해 구성한 Route53 서비스의 기본 도메인을 선택합니다.

f. 클러스터를 설명할 수 있는 이름을 입력합니다.

g. Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 붙여넣습니다.

참고

호스트에 구성된 클라우드 공급자 계정에 클러스터를 배포하기에 충분한 권한이 없는 경우, 설치 프로세스가 중단되고 누락된 권한을 알리는 메시지가 표시됩니다.

클러스터 배포가 완료되면 웹 콘솔로 연결되는 링크와 **kubeadmin** 사용자의 인증 정보가 포함된 클러스터 액세스 지침이 사용자 터미널에 표시됩니다.

출력 예

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```

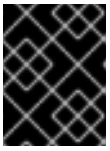
참고

설치에 성공하면 클러스터 액세스 및 인증 정보도 **<installation_directory>/.openshift_install.log**로 출력됩니다.

중요

- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.

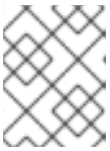
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.



중요

설치 프로그램에서 생성되는 파일이나 설치 프로그램을 삭제해서는 안 됩니다. 클러스터를 삭제하려면 두 가지가 모두 필요합니다.

2. 선택사항: 클러스터를 설치하는 데 사용한 IAM 계정에서 **AdministratorAccess** 정책을 제거하거나 비활성화합니다.



참고

AdministratorAccess 정책에서 제공하는 승격된 권한은 설치 중에만 필요합니다.

추가 리소스

- AWS 프로필 및 인증 정보 구성에 대한 자세한 내용은 AWS 문서의 구성 및 인증 정보 파일 설정을 참조하십시오.

4.4.6. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift CLI(**oc**)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 **oc**를 설치할 수 있습니다.

중요

이전 버전의 **oc**를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 **oc**를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 Linux에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Linux Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.

4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. **oc** 바이너리를 **PATH**에 있는 디렉터리에 배치합니다.
PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치
다음 절차에 따라 Windows에 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Windows Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. ZIP 프로그램으로 아카이브의 압축을 풉니다.
5. **oc** 바이너리를 **PATH**에 있는 디렉터리로 이동합니다.
PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치
다음 절차에 따라 macOS에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 MacOSX Client** 항목 옆에 있는 **지금 다운로드를** 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 해제하고 압축을 풉니다.
5. **oc** 바이너리 PATH의 디렉터리로 이동합니다.
PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

4.4.7. CLI를 사용하여 클러스터에 로그인

클러스터 **kubeconfig** 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. **kubeconfig** 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- **oc** CLI를 설치했습니다.

프로세스

1. **kubeadmin** 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 **oc** 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

4.4.8. 웹 콘솔을 사용하여 클러스터에 로그인

kubeadmin 사용자는 OpenShift Container Platform 설치 후 기본적으로 존재합니다. OpenShift Container Platform 웹 콘솔을 사용하여 **kubeadmin** 사용자로 클러스터에 로그인할 수 있습니다.

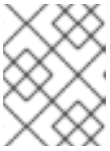
사전 요구 사항

- 설치 호스트에 대한 액세스 권한이 있어야 합니다.
- 클러스터 설치를 완료했으며 모든 클러스터 Operator를 사용할 수 있습니다.

프로세스

1. 설치 호스트의 **kubeadmin-password** 파일에서 **kubeadmin** 사용자의 암호를 가져옵니다.

```
$ cat <installation_directory>/auth/kubeadmin-password
```

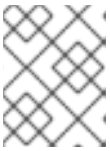


참고

대안으로 설치 호스트의 **<installation_directory>/.openshift_install.log** 로그 파일에서 **kubeadmin** 암호를 가져올 수 있습니다.

2. OpenShift Container Platform 웹 콘솔 경로를 나열합니다.

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```



참고

대안으로 설치 호스트의 **<installation_directory>/.openshift_install.log** 로그 파일에서 OpenShift Container Platform 경로를 가져올 수 있습니다.

출력 예

```
console     console-openshift-console.apps.<cluster_name>.<base_domain>           console
https   reencrypt/Redirect   None
```

3. 웹 브라우저의 이전 명령 출력에 자세히 설명된 경로로 이동하고 **kubeadmin** 사용자로 로그인합니다.

추가 리소스

- OpenShift Container Platform 웹 콘솔 액세스 및 이해에 대한 자세한 내용은 웹 콘솔에 액세스를 참조하십시오.

4.4.9. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

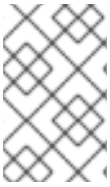
- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

4.4.10. 다음 단계

- 설치를 확인합니다.
- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.
- 필요한 경우 클라우드 공급자 인증 정보를 제거 할 수 있습니다.

4.5. 사용자 지정으로 AWS에 클러스터 설치

OpenShift Container Platform 4.9 버전에서는 설치 프로그램이 AWS(Amazon Web Services)에 프로비저닝하는 인프라에 사용자 지정 클러스터를 설치할 수 있습니다. 설치를 사용자 지정하려면 클러스터를 설치하기 전에 **install-config.yaml** 파일에서 매개변수를 수정합니다.

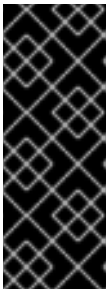


참고

OpenShift Container Platform 설치 구성의 범위는 의도적으로 한정됩니다. 단순성과 성공을 보장하도록 설계되었습니다. 설치가 완료된 후 많은 추가 OpenShift Container Platform 구성 작업을 완료할 수 있습니다.

4.5.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비 에 대한 문서를 읽습니다.
- 클러스터를 호스팅할 AWS 계정을 구성했습니다.



중요

컴퓨터에 AWS 프로필이 저장되어 있는 경우 다단계 인증 장치를 사용하는 동안 생성한 임시 세션 토큰을 해당 프로필이 사용해서는 안 됩니다. 클러스터는 클러스터의 전체 수명 동안 현재 AWS 자격 증명을 계속 사용하여 AWS 리소스를 생성하므로 장기 자격 증명을 사용해야 합니다. 적절한 키를 생성하려면 AWS 문서의 IAM 사용자의 액세스 키 관리를 참조하십시오. 설치 프로그램을 실행할 때 키를 제공할 수 있습니다.

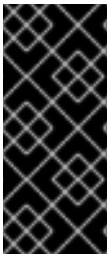
- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성해야 합니다.
- 사용자의 환경에서 클라우드 ID 및 액세스 관리(IAM) API에 액세스할 수 없거나 **kube-system** 네임스페이스에 관리자 수준의 인증 정보 시크릿을 저장하지 않으려면 IAM 인증 정보 수동 생성 및 유지 관리를 참조하십시오.

4.5.2. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.



중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

4.5.3. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 **core** 사용자의 **~/.ssh/authorized_keys** 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 **core**로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. **./openshift-install gather** 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

참고

AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: **~/.ssh/id_ed25519**)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 **'~/.ssh** 디렉터리에 있는지 확인하십시오.

참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 **ed25519** 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 **rsa** 또는 **ecdsa** 알고리즘을 사용하는 키를 생성합니다.

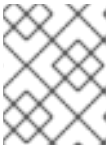
2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 **~/.ssh/id_ed25519.pub** 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 **./openshift-install gather** 명령을 사용하려는 경우 필요합니다.



참고

일부 배포에서는 **~/.ssh/id_rsa** 및 **~/.ssh/id_dsa**와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. **ssh-agent** 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. **ssh-agent**에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: **~/.ssh/id_ed25519**).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

4.5.4. AWS Marketplace 이미지 가져오기

AWS Marketplace 이미지를 사용하여 OpenShift Container Platform 클러스터를 배포하는 경우 먼저 AWS를 통해 구독해야 합니다. 이 제공을 구독하면 설치 프로그램이 작업자 노드를 배포하는 데 사용하는 AMI ID를 제공합니다.

참고

AWS Marketplace 이미지를 사용하여 OpenShift Container Platform 클러스터를 배포하는 것은 시크릿 리전 또는 중국 지역에서 지원되지 않습니다.

사전 요구 사항

- 이 제안을 구매할 수 있는 AWS 계정이 있어야 합니다. 이 계정은 클러스터를 설치하는 데 사용되는 계정과 같을 필요는 없습니다.

프로세스

1. AWS Marketplace 에서 OpenShift Container Platform 서브스크립션을 완료합니다.
2. 특정 지역에 대한 AMI ID를 기록합니다. 설치 프로세스의 일부로 클러스터를 배포하기 전에 **install-config.yaml** 파일을 이 값으로 업데이트해야 합니다.

AWS Marketplace 작업자 노드가 있는 샘플 install-config.yaml 파일

```
apiVersion: v1
baseDomain: example.com
compute:
- hyperthreading: Enabled
  name: worker
  platform:
    aws:
      amiID: ami-06c4d345f7c207239 (1)
      type: m5.4xlarge
  replicas: 3
metadata:
  name: test-cluster
platform:
  aws:
    region: us-east-2 (2)
sshKey: ssh-ed25519 AAAA...
pullSecret: '{"auths": ...}'
```

(1) AWS Marketplace 서브스크립션의 AMI ID입니다.

(2) AMI ID는 특정 AWS 리전과 연결되어 있습니다. 설치 구성 파일을 생성할 때 서브스크립션을 구성할 때 지정한 것과 동일한 AWS 리전을 선택해야 합니다.

4.5.5. 설치 프로그램 받기

OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

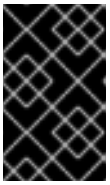
- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

프로세스

1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.
2. 인프라 공급자를 선택합니다.
3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.

중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.



중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 OpenShift Container Platform 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

4.5.6. 설치 구성 파일 만들기

AWS(Amazon Web Services)에 설치하는 OpenShift Container Platform 클러스터를 사용자 지정할 수 있습니다.

사전 요구 사항

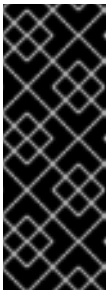
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.
- 서브스크립션 수준에서 서비스 주체 권한을 획득합니다.

절차

1. **install-config.yaml** 파일을 생성합니다.
   a. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 **<installation_directory>**는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.



중요

비어 있는 디렉터리를 지정합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

b. 화면에 나타나는 지시에 따라 클라우드에 대한 구성 세부 사항을 입력합니다.

i. 선택사항: 클러스터 시스템에 액세스하는 데 사용할 SSH 키를 선택합니다.

참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

ii. 대상 플랫폼으로 **AWS**를 선택합니다.

iii. 컴퓨터에 AWS(Amazon Web Services) 프로필이 저장되어 있지 않은 경우 설치 프로그램을 실행하도록 구성한 사용자의 AWS 액세스 키 ID와 시크릿 액세스 키를 입력합니다.

iv. 클러스터를 배포할 AWS 리전을 선택합니다.

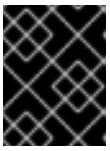
v. 클러스터에 대해 구성한 Route53 서비스의 기본 도메인을 선택합니다.

vi. 클러스터를 설명할 수 있는 이름을 입력합니다.

vii. Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 붙여넣습니다.

2. **install-config.yaml** 파일을 수정합니다. 사용 가능한 매개변수에 대한 자세한 정보는 "설치 구성 매개변수" 섹션에서 확인할 수 있습니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.

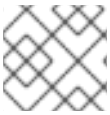


중요

install-config.yaml 파일은 설치 과정에서 사용됩니다. 이 파일을 재사용하려면 지금 백업해야 합니다.

4.5.6.1. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 매개변수 값을 제공하여 클러스터를 호스팅할 클라우드 플랫폼에서 사용자 계정을 설명하고 선택사항으로 클러스터의 플랫폼을 사용자 지정합니다. **install-config.yaml** 설치 구성 파일을 생성할 때 명령줄을 통해 필요한 매개변수 값을 제공합니다. 클러스터를 사용자 지정하면 **install-config.yaml** 파일을 수정하여 플랫폼에 대한 세부 정보를 제공할 수 있습니다.



참고

설치한 후에는 **install-config.yaml** 파일에서 이러한 매개변수를 수정할 수 없습니다.

중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

4.5.6.1.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 4.1. 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	소문자, 하이픈(**-**), 마침표(**.**)로 구성되는 문자열(예: **dev**)입니다.
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

매개변수	설명	값

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

4.5.6.1.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 4.2. 네트워크 매개변수

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체 **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.

매개변수	설명	값
networking.clusterNetwork	Pod의 IP 주소 블록입니다. 기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된 **cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16**  **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

4.5.6.1.3. 선택적 구성 매개변수

선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 4.3. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은**3**입니다.

매개변수	설명	값
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 인터넷에서 액세스할 수 없는 사설 클러스터를 배포하려면 **publish**를 **Internal**로 설정합니다. 기본값은 **External**입니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

4.5.6.1.4. 선택적 AWS 구성 매개변수

선택적 AWS 구성 매개변수는 다음 표에 설명되어 있습니다.

표 4.4. 선택적 AWS 매개변수

매개변수	설명	값
compute.platform.aws.amiID	클러스터의 컴퓨팅 머신을 부팅하는 데 사용되는 AWS AMI입니다. 이는 사용자 지정 RHCOS AMI가 필요한 리전에 필요합니다.	설정된 AWS 리전에 속하는 게시된 또는 사용자 지정 RHCOS AMI입니다.
compute.platform.aws.iamRole	컴퓨팅 머신 풀 인스턴스 프로파일에 적용되는 기존 AWS IAM 역할입니다. 이러한 필드를 사용하여 이름 지정 체계와 일치하고 IAM 역할에 대해 사전 정의된 권한 경계를 포함할 수 있습니다. 정의되지 않은 경우 설치 프로그램은 새 IAM 역할을 생성합니다.	유효한 AWS IAM 역할의 이름입니다.
compute.platform.aws.rootVolume.iops	루트 볼륨용으로 예약된 IOPS(초당 입/출력 작업)입니다.	정수(예: **4000**)
compute.platform.aws.rootVolume.size	루트 볼륨의 크기(GiB)입니다.	정수(예: **500**)
compute.platform.aws.rootVolume.type	루트 볼륨의 유형입니다.	유효한 AWS EBS 볼륨 유형 (예: **io1**)

매개변수	설명	값
compute.platform.aws.rootVolume.kmsKeyARN	KMS 키의 Amazon 리소스 이름(키 ARN)입니다. 이는 특정 KMS 키를 사용하여 작업자 노드의 OS 볼륨을 암호화하는 데 필요합니다.	유효한 키 ID 또는 키 ARN 입니다.
compute.platform.aws.type	컴퓨팅 시스템의 EC2 인스턴스 유형입니다.	유효한 AWS 인스턴스 유형 (예: **m4.2xlarge**) 다음 **머신의 인스턴스 유형**을 참조하십시오.
compute.platform.aws.zones	설치 프로그램이 컴퓨팅 머신 풀에 필요한 시스템을 생성하는 가용성 영역입니다. 자체 VPC를 제공하는 경우 해당 가용성 영역에 서브넷을 제공해야 합니다.	**us-east-1c**와 같이 YAML 시퀀스로 표시되는 유효한 AWS 가용성 영역의 목록입니다.
compute.aws.region	설치 프로그램이 컴퓨팅 리소스를 생성하는 AWS 리전입니다.	유효한 모든 AWS 리전(예: **us-east-1**)
controlPlane.platform.aws.amiID	클러스터의 컨트롤 플레인 시스템을 시작하는 데 사용되는 AWS AMI입니다. 이는 사용자 지정 RHCOS AMI가 필요한 리전에 필요합니다.	설정된 AWS 리전에 속하는 게시된 또는 사용자 지정 RHCOS AMI입니다.
controlPlane.platform.aws.iamRole	컨트롤 플레인 시스템 풀 인스턴스 프로파일에 적용되는 기존 AWS IAM 역할입니다. 이러한 필드를 사용하여 이름 지정 체계와 일치하고 IAM 역할에 대해 사전 정의된 권한 경계를 포함할 수 있습니다. 정의되지 않은 경우 설치 프로그램은 새 IAM 역할을 생성합니다.	유효한 AWS IAM 역할의 이름입니다.
controlPlane.platform.aws.rootVolume.kmsKeyARN	KMS 키의 Amazon 리소스 이름(키 ARN)입니다. 이는 특정 KMS 키를 사용하여 컨트롤 플레인 노드의 OS 볼륨을 암호화하는 데 필요합니다.	유효한 키 ID 및 키 ARN.
controlPlane.platform.aws.type	컨트롤 플레인 시스템의 EC2 인스턴스 유형입니다.	유효한 AWS 인스턴스 유형 (예: **m5.xlarge**). 다음 **머신의 인스턴스 유형**을 참조하십시오.
controlPlane.platform.aws.zones	설치 프로그램이 컨트롤 플레인 시스템 풀에 필요한 시스템을 생성하는 가용성 영역입니다.	**us-east-1c**와 같이 YAML 시퀀스로 표시되는 유효한 AWS 가용성 영역의 목록입니다.

매개변수	설명	값
controlPlane.aws.region	설치 프로그램이 컨트롤 플레인 리소스를 생성하는 AWS 리전입니다.	유효한 AWS 리전(예: **us-east-1**)
platform.aws.amiID	클러스터의 모든 머신을 부팅하는 데 사용되는 AWS AMI입니다. 설정된 경우 AMI는 클러스터와 동일한 리전에 속해 있어야 합니다. 이는 사용자 지정 RHCOS AMI가 필요한 리전에 필요합니다.	설정된 AWS 리전에 속하는 게시된 또는 사용자 지정 RHCOS AMI입니다.
platform.aws.hostedZone	클러스터의 기존 Route 53 개인 호스팅 영역입니다. 자체 VPC를 제공하는 경우에만 기존 호스팅 영역을 사용할 수 있습니다. 호스팅 영역은 설치 전에 사용자 제공 VPC와 이미 연결되어 있어야 합니다. 또한 호스팅 영역의 도메인은 클러스터 도메인 또는 클러스터 도메인의 상위 도메인이어야 합니다. 정의되지 않은 경우 설치 프로그램은 새 호스팅 영역을 생성합니다.	문자열 (예: **Z3URY6TWQ91KVV**)
platform.aws.serviceEndpoints.name	AWS 서비스 엔드 포인트 이름. 사용자 지정 엔드 포인트는 FIPS와 같은 대체 AWS 엔드 포인트를 사용해야하는 경우에만 필요합니다. EC2, S3, IAM, Elastic Load Balancing, Tagging, Route 53 및 STS AWS 서비스에 대해 사용자 지정 API 엔드 포인트를 지정할 수 있습니다.	유효한 AWS 서비스 엔드 포인트 이름.
platform.aws.serviceEndpoints.url	AWS 서비스 엔드 포인트 URL. URL은 **https** 프로토콜을 사용해야하며 호스트는 인증서를 신뢰해야 합니다.	유효한 AWS 서비스 엔드 포인트 URL.
platform.aws.userTags	설치 프로그램이 생성하는 모든 리소스에 태그로 추가하는 키와 값의 맵입니다.	유효한 YAML 맵(예: **<key>: <value>** 형식의 키 값 쌍) AWS 태그에 대한 자세한 내용은 AWS 문서의 Amazon EC2 리소스 태그 지정을 참조하십시오.

매개변수	설명	값
platform.aws.subnets	설치 프로그램이 VPC를 자동으로 생성하지 않고 VPC를 직접 제공하는 경우 사용할 클러스터의 서브넷을 지정합니다. 서브넷은 사용자가 지정하는 동일한 **machineNetwork[].cidr** 범위의 일부여야 합니다. 표준 클러스터의 경우 각 가용성 영역의 퍼블릭 및 프라이빗 서브넷을 지정합니다. 개인 클러스터의 경우 각 가용성 영역의 프라이빗 서브넷을 지정합니다.	유효한 서브넷 ID.

4.5.6.2. 클러스터 설치를 위한 최소 리소스 요구 사항

각 클러스터 시스템이 다음과 같은 최소 요구사항을 충족해야 합니다.

표 4.5. 최소 리소스 요구사항

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
부트스트랩	RHCOS	4	16GB	100GB	300
컨트롤 플레인	RHCOS	4	16GB	100GB	300
컴퓨팅	RHCOS, RHEL 7.9 또는 RHEL 8.4 [3]	2	8GB	100GB	300

1. SMT(동시 멀티스레딩) 또는 하이퍼 스레딩이 활성화되지 않은 경우 하나의 vCPU는 하나의 물리적 코어와 동일합니다. 활성화하면 다음과 같은 공식을 사용하여 해당 비율을 계산합니다. (코어당 스레드 수 × 코어 수) × 소켓 수 = vCPU 수

2. OpenShift Container Platform 및 Kubernetes는 디스크 성능에 민감하며 특히 10ms p99 fsync 기간이 필요한 컨트롤 플레인 노드의 etcd에 더 빠른 스토리지가 권장됩니다. 많은 클라우드 플랫폼에서 스토리지 크기와 IOPS를 함께 확장되므로 충분한 성능을 얻으려면 스토리지 볼륨을 과도하게 할당해야 할 수 있습니다.

3. 사용자가 프로비저닝한 모든 설치와 마찬가지로 클러스터에서 RHEL 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. RHEL 7 컴퓨팅 머신 사용은 더 이상 사용되지 않으며 향후 OpenShift Container Platform 4 릴리스에서 제거될 예정입니다.

4.5.6.3. 지원되는 AWS 머신 유형

다음 AWS(Amazon Web Services) 인스턴스 유형은 OpenShift Container Platform에서 지원됩니다.

예 4.16. 시스템의 인스턴스 유형

인스턴스 유형	부트스트랩	컨트롤 플레인	컴퓨팅
i3.large	x		
m4.large			x
m4.xlarge		x	x
m4.2xlarge		x	x
m4.4xlarge		x	x
m4.10xlarge		x	x
m4.16xlarge		x	x
m5.large			x
m5.xlarge		x	x
m5.2xlarge		x	x
m5.4xlarge		x	x
m5.8xlarge		x	x
m5.12xlarge		x	x
m5.16xlarge		x	x
m5a.large			x
m5a.xlarge		x	x
m5a.2xlarge		x	x
m5a.4xlarge		x	x
m5a.8xlarge		x	x
m5a.12xlarge		x	x
m5a.16xlarge		x	x
m6i.xlarge		x	x
m6i.2xlarge		x	x

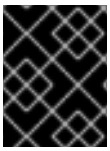
인스턴스 유형	부트스트랩	컨트롤 플레인	컴퓨팅
m6i.4xlarge		x	x
m6i.8xlarge		x	x
m6i.16xlarge		x	x
c4.2xlarge		x	x
c4.4xlarge		x	x
c4.8xlarge		x	x
c5.xlarge			x
c5.2xlarge		x	x
c5.4xlarge		x	x
c5.9xlarge		x	x
c5.12xlarge		x	x
c5.18xlarge		x	x
c5.24xlarge		x	x
c5a.xlarge			x
c5a.2xlarge		x	x
c5a.4xlarge		x	x
c5a.8xlarge		x	x
c5a.12xlarge		x	x
c5a.16xlarge		x	x
c5a.24xlarge		x	x
r4.large			x
r4.xlarge		x	x
r4.2xlarge		x	x

인스턴스 유형	부트스트랩	컨트롤 플레인	컴퓨팅
r4.4xlarge		x	x
r4.8xlarge		x	x
r4.16xlarge		x	x
r5.large			x
r5.xlarge		x	x
r5.2xlarge		x	x
r5.4xlarge		x	x
r5.8xlarge		x	x
r5.12xlarge		x	x
r5.16xlarge		x	x
r5.24xlarge		x	x
r5a.large			x
r5a.xlarge		x	x
r5a.2xlarge		x	x
r5a.4xlarge		x	x
r5a.8xlarge		x	x
r5a.12xlarge		x	x
r5a.16xlarge		x	x
r5a.24xlarge		x	x
t3.large			x
t3.xlarge			x
t3.2xlarge			x
t3a.large			x

인스턴스 유형	부트스트랩	컨트롤 플레인	컴퓨팅
t3a.xlarge			x
t3a.2xlarge			x

4.5.6.4. AWS용 샘플 사용자 지정 install-config.yaml 파일

설치 구성 파일(**install-config.yaml**)을 사용자 지정하여 OpenShift Container Platform 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.



중요

이 샘플 YAML 파일은 참조용으로만 제공됩니다. 설치 프로그램을 사용하여 **install-config.yaml** 파일을 받아서 수정해야 합니다.

```
apiVersion: v1
baseDomain: example.com (1)
credentialsMode: Mint (2)
controlPlane: (3) (4)
  hyperthreading: Enabled (5)
  name: master
  platform:
    aws:
      zones:
      - us-west-2a
      - us-west-2b
      rootVolume:
        iops: 4000
        size: 500
        type: io1 (6)
      type: m5.xlarge
  replicas: 3
compute: (7)
- hyperthreading: Enabled (8)
  name: worker
  platform:
    aws:
      rootVolume:
        iops: 2000
        size: 500
        type: io1 (9)
      type: c5.4xlarge
      zones:
      - us-west-2c
  replicas: 3
metadata:
  name: test-cluster (10)
networking:
  clusterNetwork:
```

```
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  aws:
    region: us-west-2 11
    userTags:
      adminContact: jdoe
      costCenter: 7536
    amiID: ami-96c6f8f7 12
    serviceEndpoints: 13
      - name: ec2
        url: https://vpce-id.ec2.us-west-2.vpce.amazonaws.com
fips: false 14
sshKey: ssh-ed25519 AAAA... 15
pullSecret: '{"auths": ...}' 16
```

1 10 11 16 필수 항목입니다. 설치 프로그램에서 이 값을 입력하라는 메시지를 표시합니다.

2 옵션: 이 매개 변수를 추가하여 Cloud Credential Operator (CCO)가 인증 정보의 기능을 동적으로 판별하도록 하지 않고 CCO가 지정된 모드를 사용하도록 강제합니다. CCO 모드에 대한 자세한 내용은 *Platform Operators 참조* 콘텐츠의 *Cloud Credential Operator* 항목을 참조하십시오.

3 7 이러한 매개변수와 값을 지정하지 않으면 설치 프로그램은 기본값을 적용합니다.

4 **controlPlane** 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

5 8 동시 멀티스레딩 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. 일부 클러스터 시스템에서 동시 멀티스레딩을 비활성화할 경우에는 해당 멀티스레딩을 모든 클러스터 시스템에서 비활성화해야 합니다.

중요

동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다. 동시 멀티스레딩을 비활성화하는 경우 시스템에 더 큰 인스턴스 유형(예: **m4.2xlarge** 또는 **m5.2xlarge**)을 사용합니다.

6 9 대규모의 클러스터의 경우 고속 etcd 스토리지를 구성하려면 스토리지 유형을 **IO1**로 설정하고 **iops**는 **2000**으로 설정합니다.

12 클러스터 머신을 시작하는 데 사용되는 AMI의 ID입니다. 설정된 경우 AMI는 클러스터와 동일한 리전에 속해 있어야 합니다.

13 AWS 서비스 엔드 포인트입니다. 알 수 없는 AWS 리전에 설치할 때 사용자 지정 엔드 포인트가 필요합니다. 엔드포인트 URL은 **https** 프로토콜을 사용해야하며 호스트는 인증서를 신뢰해야 합니다.

14 FIPS 모드 활성화 또는 비활성화 여부입니다. 기본적으로 FIPS 모드는 비활성화됩니다. FIPS 모드가

중요

FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.

15 선택사항으로, 클러스터의 시스템에 액세스하는 데 사용할 **sshKey** 값을 제공할 수도 있습니다.

참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

4.5.6.5. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 HTTP 또는 HTTPS 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 OpenShift Container Platform 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 API에 대한 호출을 포함하여 모든 클러스터 발신(Egress) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.

참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 Red Hat OpenStack Platform (RHOSP)에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

- **ec2.<region>.amazonaws.com,elasticloadbalancing.<region>.amazonaws.com** 및 **s3.<region>.amazonaws.com** 끝점을 VPC 끝점에 추가했습니다. 이러한 끝점은 노드에서 AWS EC2 API로 요청을 완료하는 데 필요합니다. 프록시는 노드 수준이 아닌 컨테이너 수준에서 작동하므로 이러한 요청을 AWS 사설 네트워크를 통해 AWS EC2 API로 라우팅해야 합니다. 프록시 서버의 허용 목록에 EC2 API의 공용 IP 주소를 추가하는 것만으로는 충분하지 않습니다.

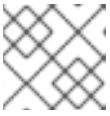
절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
```

```
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. 클러스터 외부에서 HTTP 연결을 구축하는 데 사용할 프록시 URL입니다. URL 스키마는 **http**여야 합니다.
2. 클러스터 외부에서 HTTPS 연결을 구축하는 데 사용할 프록시 URL입니다.
3. 대상 도메인 이름, IP 주소 또는 프록시에서 제외할 기타 네트워크 CIDR로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. *를 사용하여 모든 대상에 대해 프록시를 바이패스합니다.
4. 이 값을 제공하면 설치 프로그램에서 추가 CA 인증서를 보유할 **openshift -config 네임스페이스에 user-ca- bundle** 이라는 구성 맵을 생성합니다. **additionalTrustBundle** 및 하나 이상의 프록시 설정을 제공하는 경우 **프록시** 오브젝트는 **trustedCA** 필드의 **user-ca-bundle** 구성 맵을 참조하도록 구성됩니다. 그러면 Cluster Network Operator에서 **trustedCA** 매개변수에 대해 지정된 콘텐츠를 RHCOS 신뢰 번들과 병합하는 **trusted-ca-bundle** 구성 맵을 생성합니다. 프록시의 ID 인증서를 RHCOS 트러스트 번들에 있는 기관에서 서명하지 않은 경우 **additionalTrustBundle** 필드가 있어야 합니다.

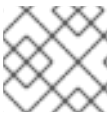


참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 OpenShift Container Platform을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 nil이 됩니다.



참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

4.5.7. 클러스터 배포

호환되는 클라우드 플랫폼에 OpenShift Container Platform을 설치할 수 있습니다.

중요

최초 설치 과정에서 설치 프로그램의 **create cluster** 명령을 한 번만 실행할 수 있습니다.

사전 요구 사항

- 클러스터를 호스팅하는 클라우드 플랫폼으로 계정을 구성합니다.
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

프로세스

1. 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터 배포를 초기화합니다.

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** 값으로 사용자 지정한 **./install-config.yaml** 파일의 위치를 지정합니다.

2 다른 설치 세부 사항을 보려면 **info** 대신 **warn**, **debug** 또는 **error**를 지정합니다.

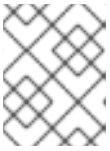
참고

호스트에 구성된 클라우드 공급자 계정에 클러스터를 배포하기에 충분한 권한이 없는 경우, 설치 프로세스가 중단되고 누락된 권한을 알리는 메시지가 표시됩니다.

클러스터 배포가 완료되면 웹 콘솔로 연결되는 링크와 **kubeadmin** 사용자의 인증 정보가 포함된 클러스터 액세스 지침이 사용자 터미널에 표시됩니다.

출력 예

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```



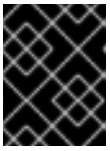
참고

설치에 성공하면 클러스터 액세스 및 인증 정보도 **<installation_directory>/.openshift_install.log**로 출력됩니다.

중요

- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.

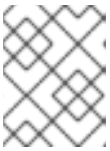
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.



중요

설치 프로그램에서 생성되는 파일이나 설치 프로그램을 삭제해서는 안 됩니다. 클러스터를 삭제하려면 두 가지가 모두 필요합니다.

2. 선택사항: 클러스터를 설치하는 데 사용한 IAM 계정에서 **AdministratorAccess** 정책을 제거하거나 비활성화합니다.

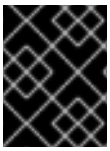


참고

AdministratorAccess 정책에서 제공하는 승격된 권한은 설치 중에만 필요합니다.

4.5.8. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift CLI(**oc**)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 **oc**를 설치할 수 있습니다.



중요

이전 버전의 **oc**를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 **oc**를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 Linux에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Linux Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. **oc** 바이너리를 **PATH**에 있는 디렉터리에 배치합니다.
PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치

다음 절차에 따라 Windows에 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.

2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Windows Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. ZIP 프로그램으로 아카이브의 압축을 풉니다.
5. **oc** 바이너리를 **PATH**에 있는 디렉터리로 이동합니다.
   PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치
다음 절차에 따라 macOS에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 MacOSX Client** 항목 옆에 있는 **지금 다운로드를** 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 해제하고 압축을 풉니다.
5. **oc** 바이너리 PATH의 디렉터리로 이동합니다.
   PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

4.5.9. CLI를 사용하여 클러스터에 로그인

클러스터 **kubeconfig** 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다.
kubeconfig 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- **oc** CLI를 설치했습니다.

프로세스

1. **kubeadmin** 인증 정보를 내보냅니다.
   -

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 **oc** 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

4.5.10. 웹 콘솔을 사용하여 클러스터에 로그인

kubeadmin 사용자는 OpenShift Container Platform 설치 후 기본적으로 존재합니다. OpenShift Container Platform 웹 콘솔을 사용하여 **kubeadmin** 사용자로 클러스터에 로그인할 수 있습니다.

사전 요구 사항

- 설치 호스트에 대한 액세스 권한이 있어야 합니다.
- 클러스터 설치를 완료했으며 모든 클러스터 Operator를 사용할 수 있습니다.

프로세스

1. 설치 호스트의 **kubeadmin-password** 파일에서 **kubeadmin** 사용자의 암호를 가져옵니다.

```
$ cat <installation_directory>/auth/kubeadmin-password
```

참고

대안으로 설치 호스트의 **<installation_directory>/.openshift_install.log** 로그 파일에서 **kubeadmin** 암호를 가져올 수 있습니다.

2. OpenShift Container Platform 웹 콘솔 경로를 나열합니다.

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```

참고

대안으로 설치 호스트의 **<installation_directory>/.openshift_install.log** 로그 파일에서 OpenShift Container Platform 경로를 가져올 수 있습니다.

출력 예

```
console     console-openshift-console.apps.<cluster_name>.<base_domain>          console
https   reencrypt/Redirect   None
```

3. 웹 브라우저의 이전 명령 출력에 자세히 설명된 경로로 이동하고 **kubeadmin** 사용자로 로그인합니다.

추가 리소스

- OpenShift Container Platform 웹 콘솔 액세스 및 이해에 대한 자세한 내용은 웹 콘솔에 액세스를 참조하십시오.

4.5.11. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

4.5.12. 다음 단계

- 설치를 확인합니다.
- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.
- 필요한 경우 클라우드 공급자 인증 정보를 제거 할 수 있습니다.

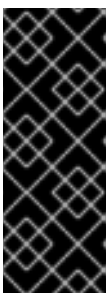
4.6. 네트워크 사용자 지정으 AWS에 클러스터 설치

OpenShift Container Platform 4.9 버전에서는 사용자 지정 네트워크 구성 옵션으로 AWS(Amazon Web Services)에 클러스터를 설치할 수 있습니다. 네트워크 구성을 사용자 지정할 경우, 클러스터가 사용자 환경의 기존 IP 주소 할당과 공존하고 기존 MTU 및 VXLAN 구성과 통합될 수 있습니다.

설치 과정에서 네트워크 구성 매개변수를 대부분 설정해야 하며, 실행 중인 클러스터에서는 **kubeProxy** 구성 매개변수만 수정할 수 있습니다.

4.6.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비 에 대한 문서를 읽습니다.
- 클러스터를 호스팅할 AWS 계정을 구성 했습니다.



중요

컴퓨터에 AWS 프로필이 저장되어 있는 경우 다단계 인증 장치를 사용하는 동안 생성한 임시 세션 토큰을 해당 프로필이 사용해서는 안 됩니다. 클러스터는 클러스터의 전체 수명 동안 현재 AWS 자격 증명을 계속 사용하여 AWS 리소스를 생성하므로 키 기반의 장기 자격 증명을 사용해야 합니다. 적절한 키를 생성하려면 AWS 문서의 IAM 사용자의 액세스 키 관리 를 참조하십시오. 설치 프로그램을 실행할 때 키를 제공할 수 있습니다.

- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성 해야 합니다.
- 사용자의 환경에서 클라우드 ID 및 액세스 관리(IAM) API에 액세스할 수 없거나 **kube-system** 네임스페이스에 관리자 수준의 인증 정보 시크릿을 저장하지 않으려면 IAM 인증 정보 수동 생성 및 유지 관리를 참조하십시오.

4.6.2. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.

중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

4.6.3. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 **core** 사용자의 **~/.ssh/authorized_keys** 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 **core**로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. **./openshift-install gather** 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

참고

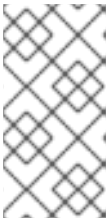
AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: **~/.ssh/id_ed25519**)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 **'~/.ssh** 디렉터리에 있는지 확인하십시오.



참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 **ed25519** 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 **rsa** 또는 **ecdsa** 알고리즘을 사용하는 키를 생성합니다.

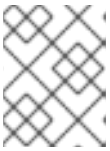
2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 **~/.ssh/id_ed25519.pub** 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 **./openshift-install gather** 명령을 사용하려는 경우 필요합니다.



참고

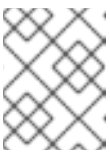
일부 배포에서는 **~/.ssh/id_rsa** 및 **~/.ssh/id_dsa**와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. **ssh-agent** 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```



참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. **ssh-agent**에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: **~/.ssh/id_ed25519**).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

4.6.4. 설치 프로그램 받기

OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

프로세스

1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.
2. 인프라 공급자를 선택합니다.
3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.

중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.

중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 OpenShift Container Platform 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

4.6.5. 네트워크 구성 단계

OpenShift Container Platform을 설치하기 전에 네트워크 구성을 사용자 지정할 수 있습니다.

1단계

매니페스트 파일을 생성하기 전에 **install-config.yaml** 파일에서 다음 네트워크 관련 필드를 사용자 지정할 수 있습니다.

- **networking.networkType**
- **networking.clusterNetwork**
- **networking.serviceNetwork**
- **networking.machineNetwork**
  이러한 필드에 대한 자세한 내용은 *설치 구성 매개 변수*에서 참조하십시오.

참고

기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

2 단계

openshift-install create manifests 를 실행하여 매니페스트 파일을 생성한 후 수정할 필드로 사용자 지정된 Cluster Network Operator 매니페스트를 정의할 수 있습니다. 매니페스트를 사용하여 고급 네트워크 구성을 지정할 수 있습니다.

install-config.yaml 파일에서 1 단계에 지정된 값을 2 단계에서 재정의할 수 없습니다. 그러나 2 단계에서 클러스터 네트워크 공급자를 추가로 사용자 지정할 수 있습니다.

4.6.6. 설치 구성 파일 만들기

AWS(Amazon Web Services)에 설치하는 OpenShift Container Platform 클러스터를 사용자 지정할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.
- 서브스크립션 수준에서 서비스 주체 권한을 획득합니다.

절차

1. **install-config.yaml** 파일을 생성합니다.
   a. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 **<installation_directory>**는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.

중요

비어 있는 디렉터리를 지정합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

b. 화면에 나타나는 지시에 따라 클라우드에 대한 구성 세부 사항을 입력합니다.

i. 선택사항: 클러스터 시스템에 액세스하는 데 사용할 SSH 키를 선택합니다.

참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

ii. 대상 플랫폼으로 **AWS**를 선택합니다.

iii. 컴퓨터에 AWS(Amazon Web Services) 프로필이 저장되어 있지 않은 경우 설치 프로그램을 실행하도록 구성한 사용자의 AWS 액세스 키 ID와 시크릿 액세스 키를 입력합니다.

iv. 클러스터를 배포할 AWS 리전을 선택합니다.

v. 클러스터에 대해 구성한 Route53 서비스의 기본 도메인을 선택합니다.

vi. 클러스터를 설명할 수 있는 이름을 입력합니다.

vii. Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 붙여넣습니다.

2. **install-config.yaml** 파일을 수정합니다. 사용 가능한 매개변수에 대한 자세한 정보는 "설치 구성 매개변수" 섹션에서 확인할 수 있습니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.

중요

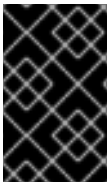
install-config.yaml 파일은 설치 과정에서 사용됩니다. 이 파일을 재사용하려면 지금 백업해야 합니다.

4.6.6.1. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 매개변수 값을 제공하여 클러스터를 호스팅할 클라우드 플랫폼에서 사용자 계정을 설명하고 선택사항으로 클러스터의 플랫폼을 사용자 지정합니다. **install-config.yaml** 설치 구성 파일을 생성할 때 명령줄을 통해 필요한 매개변수 값을 제공합니다. 클러스터를 사용자 지정하면 **install-config.yaml** 파일을 수정하여 플랫폼에 대한 세부 정보를 제공할 수 있습니다.

참고

설치한 후에는 **install-config.yaml** 파일에서 이러한 매개변수를 수정할 수 없습니다.



중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

4.6.6.1.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 4.6. 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	소문자, 하이픈(**-**), 마침표(**.**)로 구성되는 문자열(예: **dev**)입니다.
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

매개변수	설명	값

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

4.6.6.1.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 4.7. 네트워크 매개변수

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체 **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.

매개변수	설명	값
networking.clusterNetwork	Pod의 IP 주소 블록입니다. 기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된 **cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16**  **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

4.6.6.1.3. 선택적 구성 매개변수

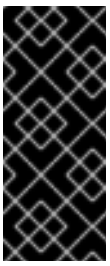
선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 4.8. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은**3**입니다.
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.

매개변수	설명	값
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 인터넷에서 액세스할 수 없는 사설 클러스터를 배포하려면 **publish**를 **Internal**로 설정합니다. 기본값은 **External**입니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

4.6.6.1.4. 선택적 AWS 구성 매개변수

선택적 AWS 구성 매개변수는 다음 표에 설명되어 있습니다.

표 4.9. 선택적 AWS 매개변수

매개변수	설명	값
compute.platform.aws.amiID	클러스터의 컴퓨팅 머신을 부팅하는 데 사용되는 AWS AMI입니다. 이는 사용자 지정 RHCOS AMI가 필요한 리전에 필요합니다.	설정된 AWS 리전에 속하는 게시된 또는 사용자 지정 RHCOS AMI입니다.
compute.platform.aws.iamRole	컴퓨팅 머신 풀 인스턴스 프로파일에 적용되는 기존 AWS IAM 역할입니다. 이러한 필드를 사용하여 이름 지정 체계와 일치하고 IAM 역할에 대해 사전 정의된 권한 경계를 포함할 수 있습니다. 정의되지 않은 경우 설치 프로그램은 새 IAM 역할을 생성합니다.	유효한 AWS IAM 역할의 이름입니다.
compute.platform.aws.rootVolume.iops	루트 볼륨용으로 예약된 IOPS(초당 입/출력 작업)입니다.	정수(예: **4000**)
compute.platform.aws.rootVolume.size	루트 볼륨의 크기(GiB)입니다.	정수(예: **500**)
compute.platform.aws.rootVolume.type	루트 볼륨의 유형입니다.	유효한 AWS EBS 볼륨 유형 (예: **io1**)

매개변수	설명	값
compute.platform.aws.rootVolume.kmsKeyARN	KMS 키의 Amazon 리소스 이름(키 ARN)입니다. 이는 특정 KMS 키를 사용하여 작업자 노드의 OS 볼륨을 암호화하는 데 필요합니다.	유효한 키 ID 또는 키 ARN 입니다.
compute.platform.aws.type	컴퓨팅 시스템의 EC2 인스턴스 유형입니다.	유효한 AWS 인스턴스 유형 (예: **m4.2xlarge**) 다음 **머신의 인스턴스 유형**을 참조하십시오.
compute.platform.aws.zones	설치 프로그램이 컴퓨팅 머신 풀에 필요한 시스템을 생성하는 가용성 영역입니다. 자체 VPC를 제공하는 경우 해당 가용성 영역에 서브넷을 제공해야 합니다.	**us-east-1c**와 같이 YAML 시퀀스로 표시되는 유효한 AWS 가용성 영역의 목록입니다.
compute.aws.region	설치 프로그램이 컴퓨팅 리소스를 생성하는 AWS 리전입니다.	유효한 모든 AWS 리전(예: **us-east-1**)
controlPlane.platform.aws.amiID	클러스터의 컨트롤 플레인 시스템을 시작하는 데 사용되는 AWS AMI입니다. 이는 사용자 지정 RHCOS AMI가 필요한 리전에 필요합니다.	설정된 AWS 리전에 속하는 게시된 또는 사용자 지정 RHCOS AMI입니다.
controlPlane.platform.aws.iamRole	컨트롤 플레인 시스템 풀 인스턴스 프로파일에 적용되는 기존 AWS IAM 역할입니다. 이러한 필드를 사용하여 이름 지정 체계와 일치하고 IAM 역할에 대해 사전 정의된 권한 경계를 포함할 수 있습니다. 정의되지 않은 경우 설치 프로그램은 새 IAM 역할을 생성합니다.	유효한 AWS IAM 역할의 이름입니다.
controlPlane.platform.aws.rootVolume.kmsKeyARN	KMS 키의 Amazon 리소스 이름(키 ARN)입니다. 이는 특정 KMS 키를 사용하여 컨트롤 플레인 노드의 OS 볼륨을 암호화하는 데 필요합니다.	유효한 키 ID 및 키 ARN.
controlPlane.platform.aws.type	컨트롤 플레인 시스템의 EC2 인스턴스 유형입니다.	유효한 AWS 인스턴스 유형 (예: **m5.xlarge**). 다음 **머신의 인스턴스 유형**을 참조하십시오.
controlPlane.platform.aws.zones	설치 프로그램이 컨트롤 플레인 시스템 풀에 필요한 시스템을 생성하는 가용성 영역입니다.	**us-east-1c**와 같이 YAML 시퀀스로 표시되는 유효한 AWS 가용성 영역의 목록입니다.

매개변수	설명	값
controlPlane.aws.region	설치 프로그램이 컨트롤 플레인 리소스를 생성하는 AWS 리전입니다.	유효한 AWS 리전(예: **us-east-1**)
platform.aws.amiID	클러스터의 모든 머신을 부팅하는 데 사용되는 AWS AMI입니다. 설정된 경우 AMI는 클러스터와 동일한 리전에 속해 있어야 합니다. 이는 사용자 지정 RHCOS AMI가 필요한 리전에 필요합니다.	설정된 AWS 리전에 속하는 게시된 또는 사용자 지정 RHCOS AMI입니다.
platform.aws.hostedZone	클러스터의 기존 Route 53 개인 호스팅 영역입니다. 자체 VPC를 제공하는 경우에만 기존 호스팅 영역을 사용할 수 있습니다. 호스팅 영역은 설치 전에 사용자 제공 VPC와 이미 연결되어 있어야 합니다. 또한 호스팅 영역의 도메인은 클러스터 도메인 또는 클러스터 도메인의 상위 도메인이어야 합니다. 정의되지 않은 경우 설치 프로그램은 새 호스팅 영역을 생성합니다.	문자열 (예: **Z3URY6TWQ91KVV**)
platform.aws.serviceEndpoints.name	AWS 서비스 엔드 포인트 이름. 사용자 지정 엔드 포인트는 FIPS와 같은 대체 AWS 엔드 포인트를 사용해야하는 경우에만 필요합니다. EC2, S3, IAM, Elastic Load Balancing, Tagging, Route 53 및 STS AWS 서비스에 대해 사용자 지정 API 엔드 포인트를 지정할 수 있습니다.	유효한 AWS 서비스 엔드 포인트 이름.
platform.aws.serviceEndpoints.url	AWS 서비스 엔드 포인트 URL. URL은 **https** 프로토콜을 사용해야하며 호스트는 인증서를 신뢰해야 합니다.	유효한 AWS 서비스 엔드 포인트 URL.
platform.aws.userTags	설치 프로그램이 생성하는 모든 리소스에 태그로 추가하는 키와 값의 맵입니다.	유효한 YAML 맵(예: **<key>: <value>** 형식의 키 값 쌍) AWS 태그에 대한 자세한 내용은 AWS 문서의 Amazon EC2 리소스 태그 지정을 참조하십시오.

매개변수	설명	값
platform.aws.subnets	설치 프로그램이 VPC를 자동으로 생성하지 않고 VPC를 직접 제공하는 경우 사용할 클러스터의 서브넷을 지정합니다. 서브넷은 사용자가 지정하는 동일한 **machineNetwork[].cidr** 범위의 일부여야 합니다. 표준 클러스터의 경우 각 가용성 영역의 퍼블릭 및 프라이빗 서브넷을 지정합니다. 개인 클러스터의 경우 각 가용성 영역의 프라이빗 서브넷을 지정합니다.	유효한 서브넷 ID.

4.6.6.2. 클러스터 설치를 위한 최소 리소스 요구 사항

각 클러스터 시스템이 다음과 같은 최소 요구사항을 충족해야 합니다.

표 4.10. 최소 리소스 요구사항

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
부트스트랩	RHCOS	4	16GB	100GB	300
컨트롤 플레인	RHCOS	4	16GB	100GB	300
컴퓨팅	RHCOS, RHEL 7.9 또는 RHEL 8.4 [3]	2	8GB	100GB	300

1. SMT(동시 멀티스레딩) 또는 하이퍼 스레딩이 활성화되지 않은 경우 하나의 vCPU는 하나의 물리적 코어와 동일합니다. 활성화하면 다음과 같은 공식을 사용하여 해당 비율을 계산합니다. (코어당 스레드 수 × 코어 수) × 소켓 수 = vCPU 수

2. OpenShift Container Platform 및 Kubernetes는 디스크 성능에 민감하며 특히 10ms p99 fsync 기간이 필요한 컨트롤 플레인 노드의 etcd에 더 빠른 스토리지가 권장됩니다. 많은 클라우드 플랫폼에서 스토리지 크기와 IOPS를 함께 확장되므로 충분한 성능을 얻으려면 스토리지 볼륨을 과도하게 할당해야 할 수 있습니다.

3. 사용자가 프로비저닝한 모든 설치와 마찬가지로 클러스터에서 RHEL 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. RHEL 7 컴퓨팅 머신 사용은 더 이상 사용되지 않으며 향후 OpenShift Container Platform 4 릴리스에서 제거될 예정입니다.

4.6.6.3. 지원되는 AWS 머신 유형

다음 AWS(Amazon Web Services) 인스턴스 유형은 OpenShift Container Platform에서 지원됩니다.

예 4.17. 시스템의 인스턴스 유형

인스턴스 유형	부트스트랩	컨트롤 플레인	컴퓨팅
i3.large	x		
m4.large			x
m4.xlarge		x	x
m4.2xlarge		x	x
m4.4xlarge		x	x
m4.10xlarge		x	x
m4.16xlarge		x	x
m5.large			x
m5.xlarge		x	x
m5.2xlarge		x	x
m5.4xlarge		x	x
m5.8xlarge		x	x
m5.12xlarge		x	x
m5.16xlarge		x	x
m5a.large			x
m5a.xlarge		x	x
m5a.2xlarge		x	x
m5a.4xlarge		x	x
m5a.8xlarge		x	x
m5a.12xlarge		x	x
m5a.16xlarge		x	x
m6i.xlarge		x	x

인스턴스 유형	부트스트랩	컨트롤 플레인	컴퓨팅
m6i.2xlarge		x	x
m6i.4xlarge		x	x
m6i.8xlarge		x	x
m6i.16xlarge		x	x
c4.2xlarge		x	x
c4.4xlarge		x	x
c4.8xlarge		x	x
c5.xlarge			x
c5.2xlarge		x	x
c5.4xlarge		x	x
c5.9xlarge		x	x
c5.12xlarge		x	x
c5.18xlarge		x	x
c5.24xlarge		x	x
c5a.xlarge			x
c5a.2xlarge		x	x
c5a.4xlarge		x	x
c5a.8xlarge		x	x
c5a.12xlarge		x	x
c5a.16xlarge		x	x
c5a.24xlarge		x	x
r4.large			x
r4.xlarge		x	x

인스턴스 유형	부트스트랩	컨트롤 플레인	컴퓨팅
r4.2xlarge		x	x
r4.4xlarge		x	x
r4.8xlarge		x	x
r4.16xlarge		x	x
r5.large			x
r5.xlarge		x	x
r5.2xlarge		x	x
r5.4xlarge		x	x
r5.8xlarge		x	x
r5.12xlarge		x	x
r5.16xlarge		x	x
r5.24xlarge		x	x
r5a.large			x
r5a.xlarge		x	x
r5a.2xlarge		x	x
r5a.4xlarge		x	x
r5a.8xlarge		x	x
r5a.12xlarge		x	x
r5a.16xlarge		x	x
r5a.24xlarge		x	x
t3.large			x
t3.xlarge			x
t3.2xlarge			x

인스턴스 유형	부트스트랩	컨트롤 플레인	컴퓨팅
t3a.large			x
t3a.xlarge			x
t3a.2xlarge			x

4.6.6.4. AWS용 샘플 사용자 지정 install-config.yaml 파일

설치 구성 파일(**install-config.yaml**)을 사용자 지정하여 OpenShift Container Platform 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.

중요

이 샘플 YAML 파일은 참조용으로만 제공됩니다. 설치 프로그램을 사용하여 **install-config.yaml** 파일을 받아서 수정해야 합니다.

```
apiVersion: v1
baseDomain: example.com (1)
credentialsMode: Mint (2)
controlPlane: (3) (4)
  hyperthreading: Enabled (5)
  name: master
  platform:
    aws:
      zones:
      - us-west-2a
      - us-west-2b
      rootVolume:
        iops: 4000
        size: 500
        type: io1 (6)
      type: m5.xlarge
  replicas: 3
compute: (7)
- hyperthreading: Enabled (8)
  name: worker
  platform:
    aws:
      rootVolume:
        iops: 2000
        size: 500
        type: io1 (9)
      type: c5.4xlarge
      zones:
      - us-west-2c
  replicas: 3
metadata:
```

```
  name: test-cluster 10
networking: 11
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  aws:
    region: us-west-2 12
    userTags:
      adminContact: jdoe
      costCenter: 7536
    amiID: ami-96c6f8f7 13
    serviceEndpoints: 14
      - name: ec2
        url: https://vpce-id.ec2.us-west-2.vpce.amazonaws.com
fips: false 15
sshKey: ssh-ed25519 AAAA... 16
pullSecret: '{"auths": ...}' 17
```

1 10 12 17 필수 항목입니다. 설치 프로그램에서 이 값을 입력하라는 메시지를 표시합니다.

2 옵션: 이 매개 변수를 추가하여 Cloud Credential Operator (CCO)가 인증 정보의 기능을 동적으로 판별하도록 하지 않고 CCO가 지정된 모드를 사용하도록 강제합니다. CCO 모드에 대한 자세한 내용은 *Platform Operators 참조* 콘텐츠의 *Cloud Credential Operator* 항목을 참조하십시오.

3 7 11 이러한 매개변수와 값을 지정하지 않으면 설치 프로그램은 기본값을 적용합니다.

4 **controlPlane** 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

5 8 동시 멀티스레딩 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. 일부 클러스터 시스템에서 동시 멀티스레딩을 비활성화할 경우에는 해당 멀티스레딩을 모든 클러스터 시스템에서 비활성화해야 합니다.

중요

동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다. 동시 멀티스레딩을 비활성화하는 경우 시스템에 더 큰 인스턴스 유형(예: **m4.2xlarge** 또는 **m5.2xlarge**)을 사용합니다.

6 9 대규모의 클러스터의 경우 고속 etcd 스토리지를 구성하려면 스토리지 유형을 **IO1**로 설정하고 **iops**는 **2000**으로 설정합니다.

13 클러스터 머신을 시작하는 데 사용되는 AMI의 ID입니다. 설정된 경우 AMI는 클러스터와 동일한 리전에 속해 있어야 합니다.

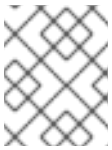
14 AWS 서비스 엔드 포인트입니다. 알 수 없는 AWS 리전에 설치할 때 사용자 지정 엔드 포인트가 필요합니다. 엔드포인트 URL은 **https** 프로토콜을 사용해야하며 호스트는 인증서를 신뢰해야 합니다.

15 FIPS 모드 활성화 또는 비활성화 여부입니다. 기본적으로 FIPS 모드는 비활성화됩니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.

중요

FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.

16 선택사항으로, 클러스터의 시스템에 액세스하는 데 사용할 **sshKey** 값을 제공할 수도 있습니다.



참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

4.6.6.5. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 HTTP 또는 HTTPS 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 OpenShift Container Platform 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 API에 대한 호출을 포함하여 모든 클러스터 발신(Egress) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.

참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 Red Hat OpenStack Platform (RHOSP)에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

- **ec2.<region>.amazonaws.com,elasticloadbalancing.<region>.amazonaws.com** 및 **s3.<region>.amazonaws.com** 끝점을 VPC 끝점에 추가했습니다. 이러한 끝점은 노드에서 AWS EC2 API로 요청을 완료하는 데 필요합니다. 프록시는 노드 수준이 아닌 컨테이너 수준에서 작동하므로 이러한 요청을 AWS 사설 네트워크를 통해 AWS EC2 API로 라우팅해야 합니다. 프록시 서버의 허용 목록에 EC2 API의 공용 IP 주소를 추가하는 것만으로는 충분하지 않습니다.

절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

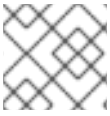
1. 클러스터 외부에서 HTTP 연결을 구축하는 데 사용할 프록시 URL입니다. URL 스키마는 **http**여야 합니다.
2. 클러스터 외부에서 HTTPS 연결을 구축하는 데 사용할 프록시 URL입니다.
3. 대상 도메인 이름, IP 주소 또는 프록시에서 제외할 기타 네트워크 CIDR로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. ***를 사용하여 모든 대상에 대해 프록시를 바이패스합니다.
4. 이 값을 제공하면 설치 프로그램에서 추가 CA 인증서를 보유할 **openshift -config 네임스페이스에 user-ca- bundle** 이라는 구성 맵을 생성합니다. **additionalTrustBundle** 및 하나 이상의 프록시 설정을 제공하는 경우 **프록시** 오브젝트는 **trustedCA** 필드의 **user-ca-bundle** 구성 맵을 참조하도록 구성됩니다. 그러면 Cluster Network Operator에서 **trustedCA** 매개변수에 대해 지정된 콘텐츠를 RHCOS 신뢰 번들과 병합하는 **trusted-ca-bundle** 구성 맵을 생성합니다. 프록시의 ID 인증서를 RHCOS 트러스트 번들에 있는 기관에서 서명하지 않은 경우 **additionalTrustBundle** 필드가 있어야 합니다.

참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 OpenShift Container Platform을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 nil이 됩니다.



참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

4.6.7. CNO(Cluster Network Operator) 구성

클러스터 네트워크의 구성은 CNO(Cluster Network Operator) 구성의 일부로 지정되며 **cluster**라는 이름의 CR(사용자 정의 리소스) 오브젝트에 저장됩니다. CR은 **operator.openshift.io** API 그룹에서 **Network** API의 필드를 지정합니다.

CNO 구성은 **Network.config.openshift.io** API 그룹의 **Network** API에서 클러스터 설치 중에 다음 필드를 상속하며 이러한 필드는 변경할 수 없습니다.

clusterNetwork

Pod IP 주소가 할당되는 IP 주소 풀입니다.

serviceNetwork

서비스를 위한 IP 주소 풀입니다.

defaultNetwork.type

OpenShift SDN 또는 OVN-Kubernetes와 같은 클러스터 네트워크 공급자입니다.

cluster라는 CNO 오브젝트에서 **defaultNetwork** 오브젝트의 필드를 설정하여 클러스터의 클러스터 네트워크 공급자 구성을 지정할 수 있습니다.

4.6.7.1. CNO(Cluster Network Operator) 구성 오브젝트

CNO(Cluster Network Operator)의 필드는 다음 표에 설명되어 있습니다.

표 4.11. CNO(Cluster Network Operator) 구성 오브젝트

필드	유형	설명
metadata.name	**string**	CNO 개체 이름입니다. 이 이름은 항상 **cluster**입니다.
spec.clusterNetwork	**array**	Pod IP 주소가 할당되는 IP 주소 블록과 클러스터의 각 개별 노드에 할당된 서브넷 접두사 길이를 지정하는 목록입니다. 예를 들면 다음과 같습니다. spec: clusterNetwork: - cidr: 10.128.0.0/19 hostPrefix: 23 - cidr: 10.128.32.0/19 hostPrefix: 23 매니페스트를 생성하기 전에 **install-config.yaml** 파일에서만 이 필드를 사용자 지정할 수 있습니다. 값은 매니페스트 파일에서 읽기 전용입니다.
spec.serviceNetwork	**array**	서비스의 IP 주소 블록입니다. OpenShift SDN 및 OVN-Kubernetes CNI(Container Network Interface) 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다. 예를 들면 다음과 같습니다. spec: serviceNetwork: - 172.30.0.0/14 매니페스트를 생성하기 전에 **install-config.yaml** 파일에서만 이 필드를 사용자 지정할 수 있습니다. 값은 매니페스트 파일에서 읽기 전용입니다.

필드	유형	설명
spec.defaultNetwork	**object**	클러스터 네트워크의 CNI(Container Network Interface) 클러스터 네트워크 공급자를 구성합니다.
spec.kubeProxyConfig	**object**	이 개체의 필드는 kube-proxy 구성을 지정합니다. OVN-Kubernetes 클러스터 네트워크 공급자를 사용하는 경우 kube-proxy 구성이 적용되지 않습니다.

defaultNetwork 오브젝트 구성
defaultNetwork 오브젝트의 값은 다음 표에 정의되어 있습니다.

표 4.12. defaultNetwork 오브젝트

필드	유형	설명
type	**string**	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, 클러스터 네트워크 공급자가 설치 중에 선택됩니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다. **참고** OpenShift Container Platform은 기본적으로 OpenShift SDN CNI(Container Network Interface) 클러스터 네트워크 공급자를 사용합니다.
openshiftSDNConfig	**object**	이 오브젝트는 OpenShift SDN 클러스터 네트워크 공급자에만 유효합니다.
ovnKubernetesConfig	**object**	이 오브젝트는 OVN-Kubernetes 클러스터 네트워크 공급자에만 유효합니다.

OpenShift SDN CNI 네트워크 공급자에 대한 구성
다음 표에서는 OpenShift SDN Container Network Interface (CNI) 클러스터 네트워크 공급자의 구성 필드를 설명합니다.

표 4.13. openshiftSDNConfig 오브젝트

필드	유형	설명
mode	**string**	OpenShift SDN의 네트워크 격리 모드를 구성합니다. 기본값은 **NetworkPolicy**입니다. **Multitenant** 및 **Subnet** 값은 OpenShift Container Platform 3.x 버전과의 역호환할 수 있지만 권장되지는 않습니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.

필드	유형	설명
mtu	**integer**	VXLAN 오버레이 네트워크의 최대 전송 단위(MTU)입니다. 이는 기본 네트워크 인터페이스의 MTU를 기준으로 자동 탐지됩니다. 일반적으로 감지된 MTU를 재정의할 필요는 없습니다. 자동 감지 값이 예상 밖인 경우 노드의 기본 네트워크 인터페이스의 MTU가 올바른지 확인합니다. 이 옵션을 사용하여 노드의 기본 네트워크 인터페이스의 MTU 값을 변경할 수 없습니다. 클러스터에 다른 노드에 대한 다른 MTU 값이 필요한 경우 이 값을 클러스터의 가장 낮은 MTU 값보다 **50** 미만으로 설정해야 합니다. 예를 들어 클러스터의 일부 노드에는 **9001**의 MTU가 있고 일부에는 **1500**의 MTU가 있는 경우 이 값을 **1450**으로 설정해야 합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
vxlanPort	**integer**	모든 VXLAN 패킷에 사용할 포트입니다. 기본값은 **4789**입니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다. 기존 노드가 다른 VXLAN 네트워크에 속하는 가상 환경에서 실행 중인 경우에는 기본값을 변경해야 할 수도 있습니다. 예를 들어 VMware NSX-T 위에서 OpenShift SDN 오버레이를 실행할 때 두 SDN이 동일한 기본 VXLAN 포트 번호를 사용하므로 VXLAN의 대체 포트를 선택해야 합니다. AWS(Amazon Web Services)에서는 포트 **9000**과 포트 **9999** 사이에서 VXLAN의 대체 포트를 선택할 수 있습니다.

OpenShift SDN 구성 예

```
defaultNetwork:
  type: OpenShiftSDN
  openshiftSDNConfig:
    mode: NetworkPolicy
    mtu: 1450
    vxlanPort: 4789
```

OVN-Kubernetes CNI 클러스터 네트워크 공급자에 대한 구성

다음 표에서는 OVN-Kubernetes CNI 클러스터 네트워크 공급자의 구성 필드를 설명합니다.

표 4.14. ovnKubernetesConfig object

필드	유형	설명

필드	유형	설명
mtu	**integer**	Geneve(Generic Network Virtualization Encapsulation) 오버레이 네트워크의 MTU(최대 전송 단위)입니다. 이는 기본 네트워크 인터페이스의 MTU를 기준으로 자동 탐지됩니다. 일반적으로 감지된 MTU를 재정의할 필요는 없습니다. 자동 감지 값이 예상 밖인 경우 노드의 기본 네트워크 인터페이스의 MTU가 올바른지 확인합니다. 이 옵션을 사용하여 노드의 기본 네트워크 인터페이스의 MTU 값을 변경할 수 없습니다. 클러스터에 다른 노드에 대한 다른 MTU 값이 필요한 경우, 이 값을 클러스터의 가장 낮은 MTU 값보다 **100** 미만으로 설정해야 합니다. 예를 들어, 클러스터의 일부 노드에 **9001**의 MTU가 있고 일부에는 **1500**의 MTU가 있는 경우 이 값을 **1400**으로 설정해야 합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
genevePort	**integer**	모든 Geneve 패킷에 사용할 포트입니다. 기본값은 **6081**입니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
ipsecConfig	**object**	IPsec 암호화를 활성화하려면 빈 오브젝트를 지정합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
policyAuditConfig	**object**	네트워크 정책 감사 로깅을 사용자 정의할 구성 오브젝트를 지정합니다. 설정되지 않으면 기본값 감사 로그 설정이 사용됩니다.

표 4.15. **policyAuditConfig** object

필드	유형	설명
rateLimit	integer	노드당 1초마다 생성할 최대 메시지 수입니다. 기본값은 초당 **20** 개의 메시지입니다.
maxFileSize	integer	감사 로그의 최대 크기(바이트)입니다. 기본값은 **50000000 또는 50** MB입니다.
대상	string	다음 추가 감사 로그 대상 중 하나입니다. **libc** 호스트에서 journald 프로세스의 libc **syslog()** 함수입니다. **udp:<host>:<port>** syslog 서버입니다. **<host>:<port>**를 syslog 서버의 호스트 및 포트로 바꿉니다. **unix:<file>** **<file>**로 지정된 Unix Domain Socket 파일입니다. **null** 감사 로그를 추가 대상으로 보내지 마십시오.

필드	유형	설명
syslogFacility	string	RFC5424에 정의된 **kern**과 같은 syslog 기능입니다. 기본값은 **local0**입니다.

OVN-Kubernetes 구성 예

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
    ipsecConfig: {}
```

kubeProxyConfig 오브젝트 구성
kubeProxyConfig 오브젝트의 값은 다음 표에 정의되어 있습니다.

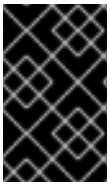
표 4.16. **kubeProxyConfig** object

필드	유형	설명
iptablesSyncPeriod	**string**	**iptables** 규칙의 새로 고침 간격입니다. 기본값은 **30s**입니다. 유효 접미사로 **s**, **m**, **h**가 있으며, 자세한 설명은 Go **time** 패키지 문서를 참조하십시오. **참고** OpenShift Container Platform 4.3 이상에서는 성능이 개선되어 더 이상 **iptablesSyncPeriod** 매개변수를 조정할 필요가 없습니다.

proxyArguments.iptables-min-sync-period	**array**	**iptables** 규칙을 새로 고치기 전 최소 기간입니다. 이 필드를 통해 새로 고침 간격이 너무 짧지 않도록 조정할 수 있습니다. 유효 접미사로 **s**, **m**, **h**가 있으며, 자세한 설명은 Go **time** 패키지를 참조하십시오. 기본값은 다음과 같습니다.
		
		`kubeProxyConfig:`
		`  proxyArguments:`
		`  iptables-min-sync-period:`
		`  - 0s`

4.6.8. 고급 네트워크 구성 지정

클러스터 네트워크 제공자의 고급 네트워크 구성을 사용하여 클러스터를 기존 네트워크 환경에 통합할 수 있습니다. 클러스터를 설치하기 전에만 고급 네트워크 구성을 지정할 수 있습니다.



중요

설치 프로그램에서 생성한 OpenShift Container Platform 매니페스트 파일을 수정하여 네트워크 구성을 사용자 정의하는 것은 지원되지 않습니다. 다음 절차에서와 같이 생성한 매니페스트 파일을 적용할 수 있습니다.

사전 요구 사항

- **install-config.yaml** 파일을 생성하고 수정 작업을 완료했습니다.

프로세스

1. 설치 프로그램이 포함된 디렉터리로 변경하고 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>**는 클러스터의 **install-config.yaml** 파일이 포함된 디렉터리의 이름을 지정합니다.

2. **<installation_directory>/ manifests/** 디렉토리에 **cluster-network-03-config.yml**이라는 stub 매니페스트 파일을 만듭니다.

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
```

3. 다음과 같이 **cluster-network-03-config.yml** 파일에서 클러스터의 고급 네트워크 구성을 지정합니다.

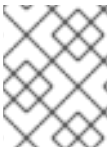
OpenShift SDN 네트워크 공급자에 대한 다른 VXLAN 포트 지정

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    openshiftSDNConfig:
      vxlanPort: 4800
```

OVN-Kubernetes 네트워크 공급자의 IPsec 활성화

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    ovnKubernetesConfig:
      ipsecConfig: {}
```

4. 선택사항: **manifests/cluster-network-03-config.yml** 파일을 백업합니다. 설치 프로그램은 Ignition 구성 파일을 생성할 때 **manifests/** 디렉터리를 사용합니다.



참고

AWS에서 NLB (Network Load Balancer)를 사용하는 방법에 대한 자세한 내용은 Network Load Balancer를 사용하여 AWS에서 Ingress 클러스터 트래픽 구성을 참조하십시오.

4.6.9. 새 AWS 클러스터에서 Ingress 컨트롤러 네트워크 로드 밸런서 생성

새 클러스터에서 AWS NLB(Network Load Balancer)가 지원하는 Ingress 컨트롤러를 생성할 수 있습니다.

사전 요구 사항

- **install-config.yaml** 파일을 생성하고 수정합니다.

프로세스

새 클러스터에서 AWS NLB가 지원하는 Ingress 컨트롤러를 생성합니다.

1. 설치 프로그램이 포함된 디렉터리로 변경하고 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>**는 클러스터의 **install-config.yaml** 파일이 포함된 디렉터리의 이름을 지정합니다.

2. **<installation_directory>/manifests/** 디렉터리에 **cluster-ingress-default-ingresscontroller.yaml**이라는 이름으로 파일을 만듭니다.

```
$ touch <installation_directory>/manifests/cluster-ingress-default-ingresscontroller.yaml 1
```

1 **<installation_directory>**는 클러스터의 **manifests** / 디렉터리가 포함된 디렉터리 이름을 지정합니다.

파일이 생성되면 다음과 같이 여러 네트워크 구성 파일이 **manifests/** 디렉토리에 나타납니다.

```
$ ls <installation_directory>/manifests/cluster-ingress-default-ingresscontroller.yaml
```

출력 예

```
cluster-ingress-default-ingresscontroller.yaml
```

3. 편집기에서 **cluster-ingress-default-ingresscontroller.yaml** 파일을 열고 원하는 운영자 구성을 설명하는 CR(사용자 정의 리소스)을 입력합니다.

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  creationTimestamp: null
  name: default
  namespace: openshift-ingress-operator
```

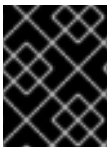
```
spec:
  endpointPublishingStrategy:
    loadBalancer:
      scope: External
      providerParameters:
        type: AWS
        aws:
          type: NLB
    type: LoadBalancerService
```

4. **cluster-ingress-default-ingresscontroller.yaml** 파일을 저장하고 텍스트 편집기를 종료합니다.

5. 선택 사항: **manifests** / **cluster-ingress-default-ingresscontroller.yaml** 파일을 백업합니다. 설치 프로그램은 클러스터를 생성할 때 **manifests/** 디렉터리를 삭제합니다.

4.6.10. OVN-Kubernetes로 하이브리드 네트워킹 구성

OVN-Kubernetes에서 하이브리드 네트워킹을 사용하도록 클러스터를 구성할 수 있습니다. 이를 통해 다양한 노드 네트워킹 구성을 지원하는 하이브리드 클러스터를 사용할 수 있습니다. 예를 들어 클러스터에서 Linux 및 Windows 노드를 모두 실행하려면 이 작업이 필요합니다.



중요

클러스터를 설치하는 동안 OVN-Kubernetes를 사용하여 하이브리드 네트워킹을 구성해야 합니다. 설치 프로세스 후에는 하이브리드 네트워킹으로 전환할 수 없습니다.

사전 요구 사항

- **install-config.yaml** 파일에 **networking.networkType**매개변수의 **OVNKubernetes**가 정의되어 있어야 합니다. 자세한 내용은 선택한 클라우드 공급자에서 OpenShift Container Platform 네트워크 사용자 정의 설정에 필요한 설치 문서를 참조하십시오.

프로세스

1. 설치 프로그램이 포함된 디렉터리로 변경하고 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory>
```

다음과 같습니다.

<installation_directory>

클러스터의 **install-config.yaml** 파일이 포함된 디렉토리의 이름을 지정합니다.

2. **<installation_directory>/ manifests/** 디렉토리에 **cluster-network-03-config.yml**이라는 stub 매니페스트 파일을 만듭니다.

```
$ cat <<EOF > <installation_directory>/manifests/cluster-network-03-config.yml
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
EOF
```

다음과 같습니다.

<installation_directory>

클러스터의 **manifests/** 디렉터리가 포함된 디렉터리 이름을 지정합니다.

3. 편집기에서 **cluster-network-03-config.yml** 파일을 열고 다음 예와 같이 하이브리드 네트워킹을 사용하여 OVN-Kubernetes를 구성합니다.

하이브리드 네트워킹 구성 지정

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    ovnKubernetesConfig:
      hybridOverlayConfig:
        hybridClusterNetwork: 1
        - cidr: 10.132.0.0/14
          hostPrefix: 23
        hybridOverlayVXLANPort: 9898 2
```

1 추가 오버레이 네트워크의 노드에 사용되는 CIDR 구성을 지정합니다. **hybridClusterNetwork** CIDR은 **clusterNetwork** CIDR과 중복될 수 없습니다.

2 추가 오버레이 네트워크에 대한 사용자 정의 VXLAN 포트를 지정합니다. 이는 vSphere에 설치된 클러스터에서 Windows 노드를 실행해야 하며 다른 클라우드 공급자에 대해 구성해서는 안 됩니다. 사용자 정의 포트는 기본 **4789** 포트를 제외한 모든 오픈 포트일 수 있습니다. 이 요구 사항에 대한 자세한 내용은 호스트 간의 포드 투 포트 연결 중단 에 대한 Microsoft 문서를 참조하십시오.

참고

Windows Server LTSC(Long-Term Servicing Channel): Windows Server 2019는 사용자 지정 **hybridOverlayVXLANPort** 값이 있는 클러스터에서 지원되지 않습니다. 이 Windows 서버 버전은 사용자 지정 VXLAN 포트를 선택하는 것을 지원하지 않기 때문입니다.

4. **cluster-network-03-config.yml** 파일을 저장하고 텍스트 편집기를 종료합니다.오.

5. 선택사항: **manifests/cluster-network-03-config.yml** 파일을 백업합니다. 설치 프로그램은 클러스터를 생성할 때 **manifests/** 디렉터리를 삭제합니다.

참고

동일한 클러스터에서 Linux 및 Windows 노드를 사용하는 방법에 대한 자세한 내용은 Windows 컨테이너 워크로드 이해 를 참조하십시오.

4.6.11. 클러스터 배포

호환되는 클라우드 플랫폼에 OpenShift Container Platform을 설치할 수 있습니다.

중요

최초 설치 과정에서 설치 프로그램의 **create cluster** 명령을 한 번만 실행할 수 있습니다.

사전 요구 사항

- 클러스터를 호스팅하는 클라우드 플랫폼으로 계정을 구성합니다.
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

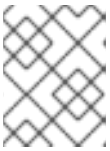
프로세스

1. 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터 배포를 초기화합니다.

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** 값으로 사용자 지정한 **./install-config.yaml** 파일의 위치를 지정합니다.

2 다른 설치 세부 사항을 보려면 **info** 대신 **warn**, **debug** 또는 **error**를 지정합니다.



참고

호스트에 구성된 클라우드 공급자 계정에 클러스터를 배포하기에 충분한 권한이 없는 경우, 설치 프로세스가 중단되고 누락된 권한을 알리는 메시지가 표시됩니다.

클러스터 배포가 완료되면 웹 콘솔로 연결되는 링크와 **kubeadmin** 사용자의 인증 정보가 포함된 클러스터 액세스 지침이 사용자 터미널에 표시됩니다.

출력 예

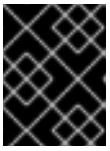
```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```

참고

설치에 성공하면 클러스터 액세스 및 인증 정보도 **<installation_directory>/.openshift_install.log**로 출력됩니다.

중요

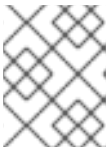
- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.



중요

설치 프로그램에서 생성되는 파일이나 설치 프로그램을 삭제해서는 안 됩니다. 클러스터를 삭제하려면 두 가지가 모두 필요합니다.

2. 선택사항: 클러스터를 설치하는 데 사용한 IAM 계정에서 **AdministratorAccess** 정책을 제거하거나 비활성화합니다.



참고

AdministratorAccess 정책에서 제공하는 승격된 권한은 설치 중에만 필요합니다.

4.6.12. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift CLI(**oc**)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 **oc**를 설치할 수 있습니다.

중요

이전 버전의 **oc**를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 **oc**를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 Linux에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Linux Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. **oc** 바이너리를 **PATH**에 있는 디렉터리에 배치합니다.

PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치
다음 절차에 따라 Windows에 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Windows Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. ZIP 프로그램으로 아카이브의 압축을 풉니다.
5. **oc** 바이너리를 **PATH**에 있는 디렉터리로 이동합니다.
   PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치
다음 절차에 따라 macOS에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 MacOSX Client** 항목 옆에 있는 **지금 다운로드를** 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 해제하고 압축을 풉니다.
5. **oc** 바이너리 PATH의 디렉터리로 이동합니다.
   PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

4.6.13. CLI를 사용하여 클러스터에 로그인

클러스터 **kubeconfig** 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. **kubeconfig** 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- **oc** CLI를 설치했습니다.

프로세스

1. **kubeadmin** 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 **oc** 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

4.6.14. 웹 콘솔을 사용하여 클러스터에 로그인

kubeadmin 사용자는 OpenShift Container Platform 설치 후 기본적으로 존재합니다. OpenShift Container Platform 웹 콘솔을 사용하여 **kubeadmin** 사용자로 클러스터에 로그인할 수 있습니다.

사전 요구 사항

- 설치 호스트에 대한 액세스 권한이 있어야 합니다.
- 클러스터 설치를 완료했으며 모든 클러스터 Operator를 사용할 수 있습니다.

프로세스

1. 설치 호스트의 **kubeadmin-password** 파일에서 **kubeadmin** 사용자의 암호를 가져옵니다.

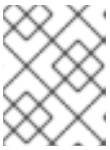
```
$ cat <installation_directory>/auth/kubeadmin-password
```

참고

대안으로 설치 호스트의 **<installation_directory>/.openshift_install.log** 로그 파일에서 **kubeadmin** 암호를 가져올 수 있습니다.

2. OpenShift Container Platform 웹 콘솔 경로를 나열합니다.

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```



참고

대안으로 설치 호스트의 **<installation_directory>/.openshift_install.log** 로그 파일에서 OpenShift Container Platform 경로를 가져올 수 있습니다.

출력 예

```
console     console-openshift-console.apps.<cluster_name>.<base_domain>          console
https   reencrypt/Redirect   None
```

3. 웹 브라우저의 이전 명령 출력에 자세히 설명된 경로로 이동하고 **kubeadmin** 사용자로 로그인합니다.

추가 리소스

- OpenShift Container Platform 웹 콘솔 액세스 및 이해에 대한 자세한 내용은 웹 콘솔에 액세스를 참조하십시오.

4.6.15. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

4.6.16. 다음 단계

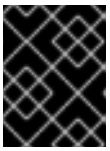
- 설치를 확인합니다.
- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.
- 필요한 경우 클라우드 공급자 인증 정보를 제거 할 수 있습니다.

4.7. 제한된 네트워크에서 AWS에 클러스터 설치

OpenShift Container Platform 버전 4.9에서는 기존 Amazon VPC(Virtual Private Cloud)에 설치 릴리스 컨텐츠의 내부 미러를 생성하여 제한된 네트워크의 AWS(Amazon Web Services)에 클러스터를 설치할 수 있습니다.

4.7.1. 사전 요구 사항

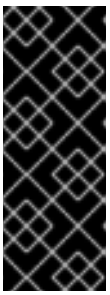
- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비 에 대한 문서를 읽습니다.
- 레지스트리에 연결이 끊긴 설치 이미지를 미러링 하고 사용 중인 OpenShift Container Platform 버전의 **imageContentSources** 데이터를 가져옵니다.



중요

미러 호스트에 설치 미디어가 있으므로 해당 컴퓨터를 사용하여 모든 설치 단계를 완료하십시오.

- AWS에 기존 VPC가 있습니다. 설치 관리자 프로비저닝 인프라를 사용하여 제한된 네트워크에 설치할 때 설치 관리자 프로비저닝 VPC를 사용할 수 없습니다. 다음 요구사항 중 하나를 충족하는 사용자 프로비저닝 VPC를 사용해야 합니다.
  - 미러 레지스트리 정보가 있습니다.
  - 방화벽 규칙 또는 피어링 연결이 다른 위치에서 호스팅되는 미러 레지스트리에 액세스할 수 있습니다.
- 클러스터를 호스팅할 AWS 계정을 구성 했습니다.



중요

컴퓨터에 AWS 프로필이 저장되어 있는 경우 다단계 인증 장치를 사용하는 동안 생성한 임시 세션 토큰을 해당 프로필이 사용해서는 안 됩니다. 클러스터는 클러스터의 전체 수명 동안 현재 AWS 자격 증명을 계속 사용하여 AWS 리소스를 생성하므로 키 기반의 장기 자격 증명을 사용해야 합니다. 적절한 키를 생성하려면 AWS 문서의 IAM 사용자의 액세스 키 관리 를 참조하십시오. 설치 프로그램을 실행할 때 키를 제공할 수 있습니다.

- AWS CLI를 다운로드하여 컴퓨터에 설치했습니다. AWS 문서의 번들 설치 관리자를 사용하여 AWS CLI 설치(Linux, macOS 또는 Unix)를 참조하십시오.
- 방화벽을 사용하며 Telemetry 서비스를 사용할 예정인 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성해야 합니다.

참고

프록시를 구성하는 경우 해당 사이트 목록도 검토해야 합니다.

- 사용자의 환경에서 클라우드 ID 및 액세스 관리(IAM) API에 액세스할 수 없거나 **kube-system** 네임스페이스에 관리자 수준의 인증 정보 시크릿을 저장하지 않으려면 IAM 인증 정보 수동 생성 및 유지 관리를 참조하십시오.

4.7.2. 네트워크가 제한된 환경에서의 설치 정보

OpenShift Container Platform 4.9에서는 소프트웨어 구성 요소를 받기 위한 인터넷 접속이 필요하지 않은 설치를 수행할 수 있습니다. 제한된 네트워크 설치는 클러스터를 설치하는 클라우드 플랫폼에 따라 설치 관리자 프로비저닝 인프라 또는 사용자 프로비저닝 인프라를 사용하여 완료할 수 있습니다.

클라우드 플랫폼에 제한된 네트워크 설치를 수행하는 방법을 선택해도 클라우드 API에 액세스는 가능해야 합니다. Amazon Web Service의 Route 53 DNS 및 IAM 서비스와 같은 일부 클라우드 기능에는 인터넷 액세스가 필요합니다. 사용 중인 네트워크에 따라 베어메탈 하드웨어 또는 VMware vSphere에 설치하기 위해 필요한 인터넷 액세스가 줄어들 수 있습니다.

제한된 네트워크 설치를 완료하려면 OpenShift Container Platform 레지스트리의 내용을 미러링하고 설치 미디어를 포함할 레지스트리를 생성해야 합니다. 인터넷과 폐쇄 네트워크에 모두 액세스하거나 제한 사항을 따르는 다른 방법을 통해 미러 호스트에 레지스트리를 생성할 수 있습니다.

4.7.2.1. 추가 제한

제한된 네트워크의 클러스터에는 다음과 같은 추가 제한이 있습니다.

- **ClusterVersion** 상태에 **사용 가능한 업데이트를 검색할 수 없음** 오류가 포함되어 있습니다.
- 기본적으로 필요한 이미지 스트림 태그에 액세스할 수 없기 때문에 개발자 카탈로그의 내용을 사용할 수 없습니다.

4.7.3. 사용자 지정 VPC 사용 정보

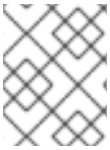
OpenShift Container Platform 4.9에서는 AWS(Amazon Web Services)의 기존 Amazon VPC(Virtual Private Cloud)에 있는 기존 서브넷에 클러스터를 배포할 수 있습니다. 기존 GCP VPC에 OpenShift Container Platform을 배포하면 새 계정의 한도 제한 적용을 받지 않거나 회사의 지침에 따른 운영 제한을 보다 쉽게 준수할 수 있습니다. VPC를 직접 생성하는 데 필요한 인프라 생성 권한을 받을 수 없는 경우 이 설치 옵션을 사용합니다.

설치 프로그램은 기존 서브넷에 있는 다른 구성 요소를 알 수 없으므로 사용자를 대신하여 서브넷 CIDR 등을 선택할 수 없습니다. 클러스터를 직접 설치하는 서브넷에 대한 네트워킹을 구성해야 합니다.

4.7.3.1. VPC 사용 요구사항

설치 프로그램은 더 이상 다음 구성 요소를 생성하지 않습니다.

- 인터넷 게이트웨이
- NAT 게이트웨이
- 서브넷
- 라우팅 테이블
- VPC
- VPC DHCP 옵션
- VPC 끝점



참고

설치 프로그램을 사용하려면 클라우드 제공 DNS 서버를 사용해야 합니다. 사용자 지정 DNS 서버 사용은 지원되지 않으며 이로 인해 설치에 실패합니다.

사용자 지정 VPC를 사용하는 경우, 사용할 클러스터와 설치 프로그램에 맞게 VPC와 해당 서브넷을 구성해야 합니다. AWS VPC 생성 및 관리에 대한 자세한 내용은 AWS 문서의 Amazon VPC 콘솔 마법사 구성 및 VPC 및 서브넷 작업을 참조하십시오.

설치 프로그램은 다음을 수행할 수 없습니다.

- 클러스터에서 사용할 네트워크 범위를 세분합니다.
- 서브넷의 라우팅 테이블을 설정합니다.
- DHCP와 같은 VPC 옵션을 설정합니다.

클러스터를 설치하기 전에 이러한 작업을 완료해야 합니다. AWS VPC의 네트워킹 구성에 대한 자세한 내용은 VPC 네트워킹 구성 요소 및 경로 테이블을 참조하십시오.

VPC는 다음 특성을 충족해야 합니다.

- VPC는 **kubernetes.io/cluster/.*: owned** 태그를 사용해서는 안 됩니다.
  설치 프로그램은 **kubernetes.io/cluster/.*: shared** 태그를 추가하도록 서브넷을 수정하므로 서브넷에 사용 가능한 여유 태그 슬롯이 하나 이상 있어야 합니다. AWS 문서의 태그 제한 사항을 참조하여 설치 프로그램이 사용자가 지정하는 각 서브넷에 태그를 추가할 수 있는지 확인합니다.
- 클러스터가 VPC에 연결된 Route 53 영역을 사용하여 클러스터의 내부 DNS 레코드를 확인할 수 있도록 VPC에서 **enableDnsSupport 및 enableDnsHostnames** 속성을 활성화해야 합니다.
  AWS 문서에서 VPC의 DNS 지원을 참조하십시오.
  자체 Route 53 호스팅 프라이빗 영역을 사용하려면 클러스터를 설치하기 전에 기존 호스팅 영역을 VPC와 연결해야 합니다. **install-config.yaml** 파일에서 **platform.aws.hostedZone** 필드를 사용하여 호스팅 영역을 정의할 수 있습니다.
- 공용 액세스 권한이 있는 클러스터를 사용하는 경우 클러스터가 사용하는 각 가용성 영역의 퍼블릭 및 프라이빗 서브넷을 만들어야 합니다. 각 가용성 영역에는 퍼블릭 서브넷과 프라이빗 서브넷이 1개 이상 포함될 수 있습니다.

연결이 끊긴 환경에서 작업 중인 경우에는 EC2 및 ELB 끝점의 공용 IP 주소에 도달할 수 없습니다. 이 문제를 해결하려면 VPC 끝점을 생성하여 클러스터가 사용 중인 서브넷에 연결해야 합니다. 올바른 끝점의 이름은 다음과 같습니다.

정부 리전

- **ec2.<region>.amazonaws.com**
- **elasticloadbalancing.<region>.amazonaws.com**
- **s3.<region>.amazonaws.com**

보안 최상위 리전

- **ec2.<region>.c2s.ic.gov**
- **elasticloadbalancing.<region>.c2s.ic.gov**
- **s3.<region>.c2s.ic.gov**

필수 VPC 구성 요소

시스템과의 통신을 허용하는 서브넷과 적합한 VPC를 제공해야 합니다.

<table>
<tr><th>구성 요소</th><th>AWS 유형</th><th>설명</th></tr>
<tr><td>VPC</td><td>AWS::EC2::VPC
AWS::EC2::VPCEndpoint</td><td>클러스터에서 사용할 공용 VPC를 제공해야 합니다. VPC는 각 서브넷의 라우팅 테이블을 참조하는 끝점을 사용하여 S3에서 호스팅되는 레지스트리와의 통신을 개선합니다.</td></tr>
<tr><td>퍼블릭 서브넷</td><td>AWS::EC2::Subnet
AWS::EC2::SubnetNetworkAclAssociation</td><td>VPC에는 1~3개의 가용성 영역에 대한 퍼블릭 서브넷이 있어야 하며 이 서브넷을 적절한 인그레스 규칙과 연결해야 합니다.</td></tr>
<tr><td>인터넷 게이트웨이</td><td>AWS::EC2::InternetGateway
AWS::EC2::VPCGatewayAttachment
AWS::EC2::RouteTable
AWS::EC2::Route
AWS::EC2::SubnetRouteTableAssociation
AWS::EC2::NatGateway
AWS::EC2::EIP</td><td>공용 경로가 있는 공용 인터넷 게이트웨이가 VPC에 연결되어 있어야 합니다. 제공된 템플릿에서 각 퍼블릭 서브넷에는 EIP 주소를 갖는 NAT 게이트웨이가 있습니다. 이러한 NAT 게이트웨이를 사용하면 프라이빗 서브넷 인스턴스와 같은 클러스터 리소스가 인터넷에 도달할 수 있으므로 일부 제한된 네트워크 또는 프록시 시나리오에는 필요하지 않습니다.</td></tr>
<tr><td>네트워크 액세스 제어</td><td>AWS::EC2::NetworkAcl
AWS::EC2::NetworkAclEntry</td><td>VPC가 다음 포트에 액세스할 수 있어야 합니다.<table><tr><th>포트</th><th>이유</th></tr><tr><td>80</td><td>인바운드 HTTP 트래픽</td></tr><tr><td>443</td><td>인바운드 HTTPS 트래픽</td></tr><tr><td>22</td><td>인바운드 SSH 트래픽</td></tr><tr><td>1024 - 65535</td><td>인바운드 임시 트래픽</td></tr><tr><td>0 - 65535</td><td>아웃바운드 임시 트래픽</td></tr></table></td></tr>
</table>

구성 요소	AWS 유형	설명
프라이빗 서브넷	• **AWS::EC2::Subnet** • **AWS::EC2::RouteTable** • **AWS::EC2::SubnetRouteTableAssociation**	VPC에 프라이빗 서브넷이 포함될 수 있습니다. 제공된 CloudFormation 템플릿은 1~3개 가용성 영역의 프라이빗 서브넷을 생성할 수 있습니다. 프라이빗 서브넷을 사용하는 경우 적절한 경로와 테이블을 제공해야 합니다.

4.7.3.2. VPC 검증

제공한 서브넷이 적합한지 확인하기 위해 설치 프로그램이 다음 데이터를 확인합니다.

- 사용자가 지정하는 모든 서브넷이 있는지 여부.
- 사용자가 프라이빗 서브넷을 제공합니다.
- 서브넷 CIDR이 사용자가 지정한 시스템 CIDR에 속합니다.
- 각 가용성 영역에 서브넷을 제공합니다. 각 가용성 영역에는 퍼블릭 서브넷과 프라이빗 서브넷이 하나씩 포함됩니다. 개인 클러스터를 사용하는 경우 각 가용성 영역에 프라이빗 서브넷만 제공합니다. 그렇지 않으면 각 가용성 영역에 퍼블릭 서브넷과 프라이빗 서브넷을 하나씩만 제공합니다.
- 각 프라이빗 서브넷 가용성 영역에 퍼블릭 서브넷을 제공합니다. 프라이빗 서브넷을 제공하지 않는 가용성 영역에서는 시스템이 프로비저닝되지 않습니다.

기존 VPC를 사용하는 클러스터를 제거해도 VPC는 삭제되지 않습니다. VPC에서 OpenShift Container Platform 클러스터를 제거하면 클러스터가 사용한 서브넷에서 **kubernetes.io/cluster/.*: shared** 태그가 제거됩니다.

4.7.3.3. 권한 분할

OpenShift Container Platform 4.3부터 클러스터를 배포하는 데 설치 프로그램에서 프로비저닝한 인프라 클러스터에 필요한 권한 중 일부가 필요하지 않게 되었습니다. 이 변경 사항은 회사의 권한 분류와 유사합니다. 즉 일부 개인의 경우 클라우드에서 다른 사람들과 다른 리소스를 생성할 수 있습니다. 예를 들어 인스턴스, 버킷, 로드 밸런서와 같은 애플리케이션 관련 항목은 생성할 수 있지만 VPC 서브넷 또는 인그레스 규칙과 같은 네트워킹 관련 구성 요소는 생성하지 못할 수 있습니다.

클러스터를 생성할 때 사용하는 GCP 자격 증명에는 서브넷, 라우팅 테이블, 인터넷 게이트웨이, NAT, VPN과 같은 VPC 내 핵심 네트워킹 구성 요소와 VPC를 생성하는 데 필요한 네트워킹 권한이 필요하지 않습니다. 하지만 ELB, 보안 그룹, S3 버킷 및 노드와 같이 클러스터 내 시스템에 필요한 애플리케이션 리소스를 생성하려면 여전히 권한이 필요합니다.

4.7.3.4. 클러스터 간 격리

OpenShift Container Platform을 기존 네트워크에 배포하면 클러스터 서비스 격리가 다음과 같은 방식으로 감소합니다.

- 동일한 VPC에 여러 OpenShift Container Platform 클러스터를 설치할 수 있습니다.
- ICMP 인그레스가 전체 네트워크에서 허용됩니다.
- TCP 22 인그레스(SSH)가 전체 네트워크에 허용됩니다.

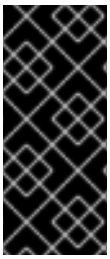
- 컨트롤 플레인 TCP 6443 인그레스(Kubernetes API)가 전체 네트워크에 허용됩니다.
- 컨트롤 플레인 TCP 22623 인그레스(MCS)가 전체 네트워크에 허용됩니다.

4.7.4. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하기 위해 필요한 이미지를 받으려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.



중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

4.7.5. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 **core** 사용자의 **~/.ssh/authorized_keys** 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 **core**로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. **./openshift-install gather** 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

참고

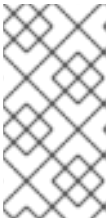
AWS 키 쌍 과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: **~/.ssh/id_ed25519**)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 **'~/.ssh** 디렉터리에 있는지 확인하십시오.



참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 **ed25519** 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 **rsa** 또는 **ecdsa** 알고리즘을 사용하는 키를 생성합니다.

2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 **~/.ssh/id_ed25519.pub** 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 **./openshift-install gather** 명령을 사용하려는 경우 필요합니다.

참고

일부 배포에서는 **~/.ssh/id_rsa** 및 **~/.ssh/id_dsa**와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. **ssh-agent** 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. **ssh-agent**에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: **~/.ssh/id_ed25519**).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

4.7.6. 설치 구성 파일 만들기

AWS(Amazon Web Services)에 설치하는 OpenShift Container Platform 클러스터를 사용자 지정할 수 있습니다.

사전 요구 사항

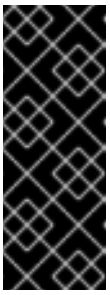
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다. 제한된 네트워크 설치의 경우, 해당 파일은 미러 호스트에 있습니다.
- 미러 레지스트리 작성 중에 생성된 **imageContentSources** 값이 있어야 합니다.
- 미러 레지스트리에 대한 인증서의 내용을 가져옵니다.
- 서브스크립션 수준에서 서비스 주체 권한을 획득합니다.

절차

1. **install-config.yaml** 파일을 생성합니다.
   a. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 **<installation_directory>**는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.



중요

비어 있는 디렉터리를 지정합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

   b. 화면에 나타나는 지시에 따라 클라우드에 대한 구성 세부 사항을 입력합니다.
      i. 선택사항: 클러스터 시스템에 액세스하는 데 사용할 SSH 키를 선택합니다.

참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

ii. 대상 플랫폼으로 **AWS**를 선택합니다.

iii. 컴퓨터에 AWS(Amazon Web Services) 프로필이 저장되어 있지 않은 경우 설치 프로그램을 실행하도록 구성한 사용자의 AWS 액세스 키 ID와 시크릿 액세스 키를 입력합니다.

iv. 클러스터를 배포할 AWS 리전을 선택합니다.

v. 클러스터에 대해 구성한 Route53 서비스의 기본 도메인을 선택합니다.

vi. 클러스터를 설명할 수 있는 이름을 입력합니다.

vii. Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 붙여넣습니다.

2. 제한된 네트워크에서의 설치에 필요한 추가 정보를 제공하려면 **install-config.yaml** 파일을 편집합니다.

a. 레지스트리의 인증 정보를 포함하도록 **pullSecret** 값을 업데이트합니다.

```
pullSecret: '{"auths":{"<mirror_host_name>:5000": {"auth": "<credentials>","email":
"you@example.com"}}}'
```

<mirror_host_name>의 경우 미러 레지스트리의 인증서에 지정한 레지스트리 도메인 이름을 지정하고 **<credentials>**의 경우 미러 레지스트리에 base64로 인코딩된 사용자 이름 및 암호를 지정합니다.

b. **additionalTrustBundle** 매개변수와 값을 추가합니다.

```
additionalTrustBundle: |
  -----BEGIN CERTIFICATE-----

ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
```

값은 미러 레지스트리에 사용한 인증서 파일의 내용이어야 하며, 신뢰할 수 있는 기존 인증 기관 또는 미러 레지스트리에 대해 생성한 자체 서명 인증서일 수 있습니다.

c. 클러스터를 설치할 VPC의 서브넷을 정의합니다.

```
subnets:
- subnet-1
- subnet-2
- subnet-3
```

d. 다음과 같은 이미지 콘텐츠 리소스를 추가합니다.

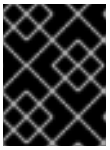
```
imageContentSources:
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: quay.io/openshift-release-dev/ocp-release
```

```
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: registry.redhat.io/ocp/release
```

이러한 값을 완료하려면 미러 레지스트리 생성 중에 기록한 **imageContentSources**를 사용하십시오.

3. 필요한 **install-config.yaml** 파일을 수정합니다. 사용 가능한 매개변수에 대한 자세한 정보는 **Installation configuration parameters** 섹션에서 확인할 수 있습니다.

4. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.



중요

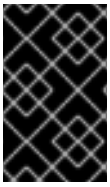
install-config.yaml 파일은 설치 과정에서 사용됩니다. 이 파일을 재사용하려면 지금 백업해야 합니다.

4.7.6.1. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 매개변수 값을 제공하여 클러스터를 호스팅할 클라우드 플랫폼에서 사용자 계정을 설명하고 선택사항으로 클러스터의 플랫폼을 사용자 지정합니다. **install-config.yaml** 설치 구성 파일을 생성할 때 명령줄을 통해 필요한 매개변수 값을 제공합니다. 클러스터를 사용자 지정하면 **install-config.yaml** 파일을 수정하여 플랫폼에 대한 세부 정보를 제공할 수 있습니다.

참고

설치한 후에는 **install-config.yaml** 파일에서 이러한 매개변수를 수정할 수 없습니다.



중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

4.7.6.1.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 4.17. 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열

매개변수	설명	값
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	소문자, 하이픈(**-**), 마침표(**.**)로 구성되는 문자열(예: **dev**)입니다.
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

4.7.6.1.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 4.18. 네트워크 매개변수

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체 **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.

networking.clusterNetwork	Pod의 IP 주소 블록입니다.
	
	기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다.
	
	여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.
		개체의 배열입니다. 예를 들면 다음과 같습니다.
		
		``
		networking:
		  clusterNetwork:
		  - cidr: 10.128.0.0/14
		    hostPrefix: 23
		``

networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된 **cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.

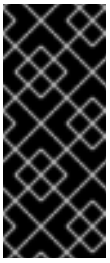
매개변수	설명	값
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16** **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

4.7.6.1.3. 선택적 구성 매개변수

선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 4.19. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열

매개변수	설명	값
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은 **3**입니다.
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열

매개변수	설명	값
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 인터넷에서 액세스할 수 없는 사설 클러스터를 배포하려면 **publish**를 **Internal**로 설정합니다. 기본값은 **External**입니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

4.7.6.1.4. 선택적 AWS 구성 매개변수

선택적 AWS 구성 매개변수는 다음 표에 설명되어 있습니다.

표 4.20. 선택적 AWS 매개변수

매개변수	설명	값
compute.platform.aws.amiID	클러스터의 컴퓨팅 머신을 부팅하는 데 사용되는 AWS AMI입니다. 이는 사용자 지정 RHCOS AMI가 필요한 리전에 필요합니다.	설정된 AWS 리전에 속하는 게시된 또는 사용자 지정 RHCOS AMI입니다.
compute.platform.aws.iamRole	컴퓨팅 머신 풀 인스턴스 프로파일에 적용되는 기존 AWS IAM 역할입니다. 이러한 필드를 사용하여 이름 지정 체계와 일치하고 IAM 역할에 대해 사전 정의된 권한 경계를 포함할 수 있습니다. 정의되지 않은 경우 설치 프로그램은 새 IAM 역할을 생성합니다.	유효한 AWS IAM 역할의 이름입니다.
compute.platform.aws.rootVolume.iops	루트 볼륨용으로 예약된 IOPS(초당 입/출력 작업)입니다.	정수(예: **4000**)
compute.platform.aws.rootVolume.size	루트 볼륨의 크기(GiB)입니다.	정수(예: **500**)
compute.platform.aws.rootVolume.type	루트 볼륨의 유형입니다.	유효한 AWS EBS 볼륨 유형 (예: **io1**)

매개변수	설명	값
compute.platform.aws.rootVolume.kmsKeyARN	KMS 키의 Amazon 리소스 이름(키 ARN)입니다. 이는 특정 KMS 키를 사용하여 작업자 노드의 OS 볼륨을 암호화하는 데 필요합니다.	유효한 키 ID 또는 키 ARN 입니다.
compute.platform.aws.type	컴퓨팅 시스템의 EC2 인스턴스 유형입니다.	유효한 AWS 인스턴스 유형 (예: **m4.2xlarge**) 다음 **머신의 인스턴스 유형**을 참조하십시오.
compute.platform.aws.zones	설치 프로그램이 컴퓨팅 머신 풀에 필요한 시스템을 생성하는 가용성 영역입니다. 자체 VPC를 제공하는 경우 해당 가용성 영역에 서브넷을 제공해야 합니다.	**us-east-1c**와 같이 YAML 시퀀스로 표시되는 유효한 AWS 가용성 영역의 목록입니다.
compute.aws.region	설치 프로그램이 컴퓨팅 리소스를 생성하는 AWS 리전입니다.	유효한 모든 AWS 리전(예: **us-east-1**)
controlPlane.platform.aws.amiID	클러스터의 컨트롤 플레인 시스템을 시작하는 데 사용되는 AWS AMI입니다. 이는 사용자 지정 RHCOS AMI가 필요한 리전에 필요합니다.	설정된 AWS 리전에 속하는 게시된 또는 사용자 지정 RHCOS AMI입니다.
controlPlane.platform.aws.iamRole	컨트롤 플레인 시스템 풀 인스턴스 프로파일에 적용되는 기존 AWS IAM 역할입니다. 이러한 필드를 사용하여 이름 지정 체계와 일치하고 IAM 역할에 대해 사전 정의된 권한 경계를 포함할 수 있습니다. 정의되지 않은 경우 설치 프로그램은 새 IAM 역할을 생성합니다.	유효한 AWS IAM 역할의 이름입니다.
controlPlane.platform.aws.rootVolume.kmsKeyARN	KMS 키의 Amazon 리소스 이름(키 ARN)입니다. 이는 특정 KMS 키를 사용하여 컨트롤 플레인 노드의 OS 볼륨을 암호화하는 데 필요합니다.	유효한 키 ID 및 키 ARN.
controlPlane.platform.aws.type	컨트롤 플레인 시스템의 EC2 인스턴스 유형입니다.	유효한 AWS 인스턴스 유형 (예: **m5.xlarge**). 다음 **머신의 인스턴스 유형**을 참조하십시오.
controlPlane.platform.aws.zones	설치 프로그램이 컨트롤 플레인 시스템 풀에 필요한 시스템을 생성하는 가용성 영역입니다.	**us-east-1c**와 같이 YAML 시퀀스로 표시되는 유효한 AWS 가용성 영역의 목록입니다.

매개변수	설명	값
controlPlane.aws.region	설치 프로그램이 컨트롤 플레인 리소스를 생성하는 AWS 리전입니다.	유효한 AWS 리전(예: **us-east-1**)
platform.aws.amiID	클러스터의 모든 머신을 부팅하는 데 사용되는 AWS AMI입니다. 설정된 경우 AMI는 클러스터와 동일한 리전에 속해 있어야 합니다. 이는 사용자 지정 RHCOS AMI가 필요한 리전에 필요합니다.	설정된 AWS 리전에 속하는 게시된 또는 사용자 지정 RHCOS AMI입니다.
platform.aws.hostedZone	클러스터의 기존 Route 53 개인 호스팅 영역입니다. 자체 VPC를 제공하는 경우에만 기존 호스팅 영역을 사용할 수 있습니다. 호스팅 영역은 설치 전에 사용자 제공 VPC와 이미 연결되어 있어야 합니다. 또한 호스팅 영역의 도메인은 클러스터 도메인 또는 클러스터 도메인의 상위 도메인이어야 합니다. 정의되지 않은 경우 설치 프로그램은 새 호스팅 영역을 생성합니다.	문자열 (예: **Z3URY6TWQ91KVV**)
platform.aws.serviceEndpoints.name	AWS 서비스 엔드 포인트 이름. 사용자 지정 엔드 포인트는 FIPS와 같은 대체 AWS 엔드 포인트를 사용해야하는 경우에만 필요합니다. EC2, S3, IAM, Elastic Load Balancing, Tagging, Route 53 및 STS AWS 서비스에 대해 사용자 지정 API 엔드 포인트를 지정할 수 있습니다.	유효한 AWS 서비스 엔드 포인트 이름.
platform.aws.serviceEndpoints.url	AWS 서비스 엔드 포인트 URL. URL은 **https** 프로토콜을 사용해야하며 호스트는 인증서를 신뢰해야 합니다.	유효한 AWS 서비스 엔드 포인트 URL.
platform.aws.userTags	설치 프로그램이 생성하는 모든 리소스에 태그로 추가하는 키와 값의 맵입니다.	유효한 YAML 맵(예: **<key>: <value>** 형식의 키 값 쌍) AWS 태그에 대한 자세한 내용은 AWS 문서의 Amazon EC2 리소스 태그 지정을 참조하십시오.

매개변수	설명	값
platform.aws.subnets	설치 프로그램이 VPC를 자동으로 생성하지 않고 VPC를 직접 제공하는 경우 사용할 클러스터의 서브넷을 지정합니다. 서브넷은 사용자가 지정하는 동일한 **machineNetwork[].cidr** 범위의 일부여야 합니다. 표준 클러스터의 경우 각 가용성 영역의 퍼블릭 및 프라이빗 서브넷을 지정합니다. 개인 클러스터의 경우 각 가용성 영역의 프라이빗 서브넷을 지정합니다.	유효한 서브넷 ID.

4.7.6.2. 클러스터 설치를 위한 최소 리소스 요구 사항

각 클러스터 시스템이 다음과 같은 최소 요구사항을 충족해야 합니다.

표 4.21. 최소 리소스 요구사항

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
부트스트랩	RHCOS	4	16GB	100GB	300
컨트롤 플레인	RHCOS	4	16GB	100GB	300
컴퓨팅	RHCOS, RHEL 7.9 또는 RHEL 8.4 [3]	2	8GB	100GB	300

1. SMT(동시 멀티스레딩) 또는 하이퍼 스레딩이 활성화되지 않은 경우 하나의 vCPU는 하나의 물리적 코어와 동일합니다. 활성화하면 다음과 같은 공식을 사용하여 해당 비율을 계산합니다. (코어당 스레드 수 × 코어 수) × 소켓 수 = vCPU 수

2. OpenShift Container Platform 및 Kubernetes는 디스크 성능에 민감하며 특히 10ms p99 fsync 기간이 필요한 컨트롤 플레인 노드의 etcd에 더 빠른 스토리지가 권장됩니다. 많은 클라우드 플랫폼에서 스토리지 크기와 IOPS를 함께 확장되므로 충분한 성능을 얻으려면 스토리지 볼륨을 과도하게 할당해야 할 수 있습니다.

3. 사용자가 프로비저닝한 모든 설치와 마찬가지로 클러스터에서 RHEL 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. RHEL 7 컴퓨팅 머신 사용은 더 이상 사용되지 않으며 향후 OpenShift Container Platform 4 릴리스에서 제거될 예정입니다.

4.7.6.3. AWS용 샘플 사용자 지정 install-config.yaml 파일

설치 구성 파일(**install-config.yaml**)을 사용자 지정하여 OpenShift Container Platform 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.

중요

이 샘플 YAML 파일은 참조용으로만 제공됩니다. 설치 프로그램을 사용하여 **install-config.yaml** 파일을 받아서 수정해야 합니다.

```
apiVersion: v1
baseDomain: example.com (1)
credentialsMode: Mint (2)
controlPlane: (3) (4)
  hyperthreading: Enabled (5)
  name: master
  platform:
    aws:
      zones:
      - us-west-2a
      - us-west-2b
      rootVolume:
        iops: 4000
        size: 500
        type: io1 (6)
      type: m5.xlarge
  replicas: 3
compute: (7)
- hyperthreading: Enabled (8)
  name: worker
  platform:
    aws:
      rootVolume:
        iops: 2000
        size: 500
        type: io1 (9)
      type: c5.4xlarge
      zones:
      - us-west-2c
  replicas: 3
metadata:
  name: test-cluster (10)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  aws:
    region: us-west-2 (11)
    userTags:
      adminContact: jdoe
```

```
      costCenter: 7536
    subnets: (12)
    - subnet-1
    - subnet-2
    - subnet-3
    amiID: ami-96c6f8f7 (13)
    serviceEndpoints: (14)
      - name: ec2
        url: https://vpce-id.ec2.us-west-2.vpce.amazonaws.com
    hostedZone: Z3URY6TWQ91KVV (15)
fips: false (16)
sshKey: ssh-ed25519 AAAA... (17)
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}}' (18)
additionalTrustBundle: | (19)
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
imageContentSources: (20)
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

(1) (10) (11) 필수 항목입니다. 설치 프로그램에서 이 값을 입력하라는 메시지를 표시합니다.

(2) 옵션: 이 매개 변수를 추가하여 Cloud Credential Operator (CCO)가 인증 정보의 기능을 동적으로 판별하도록 하지 않고 CCO가 지정된 모드를 사용하도록 강제합니다. CCO 모드에 대한 자세한 내용은 *Platform Operators 참조* 콘텐츠의 *Cloud Credential Operator* 항목을 참조하십시오.

(3) (7) 이러한 매개변수와 값을 지정하지 않으면 설치 프로그램은 기본값을 적용합니다.

(4) **controlPlane** 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

(5) (8) 동시 멀티스레딩 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. 일부 클러스터 시스템에서 동시 멀티스레딩을 비활성화할 경우에는 해당 멀티스레딩을 모든 클러스터 시스템에서 비활성화해야 합니다.

중요

동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다. 동시 멀티스레딩을 비활성화하는 경우 시스템에 더 큰 인스턴스 유형(예: **m4.2xlarge** 또는 **m5.2xlarge**)을 사용합니다.

(6) (9) 대규모의 클러스터의 경우 고속 etcd 스토리지를 구성하려면 스토리지 유형을 **IO1**로 설정하고 **iops**는 **2000**으로 설정합니다.

(12) 자체 VPC를 제공하는 경우 클러스터가 사용하는 각 가용성 영역의 서브넷을 지정합니다.

13 클러스터 머신을 시작하는 데 사용되는 AMI의 ID입니다. 설정된 경우 AMI는 클러스터와 동일한 리전에 속해 있어야 합니다.

14 AWS 서비스 엔드 포인트입니다. 알 수 없는 AWS 리전에 설치할 때 사용자 지정 엔드 포인트가 필요합니다. 엔드포인트 URL은 **https** 프로토콜을 사용해야하며 호스트는 인증서를 신뢰해야 합니다.

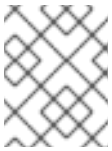
15 기존 Route 53 개인 호스팅 영역의 ID입니다. 기존 호스팅 영역을 제공하려면 클러스터를 설치하기 전에 자체 VPC를 제공하고 호스팅 영역이 VPC와 연결되어 있어야 합니다. 정의되지 않은 경우 설치 프로그램은 새 호스팅 영역을 생성합니다.

16 FIPS 모드 활성화 또는 비활성화 여부입니다. 기본적으로 FIPS 모드는 비활성화됩니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.

중요

FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.

17 선택사항으로, 클러스터의 시스템에 액세스하는 데 사용할 **sshKey** 값을 제공할 수도 있습니다.



참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

18 **<local_registry>**는 미러 레지스트리가 해당 내용을 제공하는 데 사용하는 레지스트리 도메인 이름과 포트(선택사항)를 지정합니다. 예: **registry.example.com** 또는 **registry.example.com:5000**. **<credentials>**는 미러 레지스트리의 base64 인코딩 사용자 이름과 암호를 지정합니다.

19 미러 레지스트리에 사용한 인증서 파일의 내용을 제공하십시오.

20 명령 출력에서 **imageContentSources** 섹션을 제공하여 리포지토리를 미러링하십시오.

4.7.6.4. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 HTTP 또는 HTTPS 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 OpenShift Container Platform 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 API에 대한 호출을 포함하여 모든 클러스터 발신(Egress) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.

참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 Red Hat OpenStack Platform (RHOSP)에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

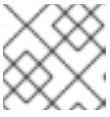
- **ec2.<region>.amazonaws.com,elasticloadbalancing.<region>.amazonaws.com** 및 **s3.<region>.amazonaws.com** 끝점을 VPC 끝점에 추가했습니다. 이러한 끝점은 노드에서 AWS EC2 API로 요청을 완료하는 데 필요합니다. 프록시는 노드 수준이 아닌 컨테이너 수준에서 작동하므로 이러한 요청을 AWS 사설 네트워크를 통해 AWS EC2 API로 라우팅해야 합니다. 프록시 서버의 허용 목록에 EC2 API의 공용 IP 주소를 추가하는 것만으로는 충분하지 않습니다.

절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. 클러스터 외부에서 HTTP 연결을 구축하는 데 사용할 프록시 URL입니다. URL 스키마는 **http**여야 합니다.
2. 클러스터 외부에서 HTTPS 연결을 구축하는 데 사용할 프록시 URL입니다.
3. 대상 도메인 이름, IP 주소 또는 프록시에서 제외할 기타 네트워크 CIDR로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. *를 사용하여 모든 대상에 대해 프록시를 바이패스합니다.
4. 이 값을 제공하면 설치 프로그램에서 추가 CA 인증서를 보유할 **openshift -config 네임스페이스에 user-ca- bundle** 이라는 구성 맵을 생성합니다. **additionalTrustBundle** 및 하나 이상의 프록시 설정을 제공하는 경우 **프록시** 오브젝트는 **trustedCA** 필드의 **user-ca-bundle** 구성 맵을 참조하도록 구성됩니다. 그러면 Cluster Network Operator에서 **trustedCA** 매개변수에 대해 지정된 콘텐츠를 RHCOS 신뢰 번들과 병합하는 **trusted-ca-bundle** 구성 맵을 생성합니다. 프록시의 ID 인증서를 RHCOS 트러스트 번들에 있는 기관에서 서명하지 않은 경우 **additionalTrustBundle** 필드가 있어야 합니다.



참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 OpenShift Container Platform을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 nil이 됩니다.

참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

4.7.7. 클러스터 배포

호환되는 클라우드 플랫폼에 OpenShift Container Platform을 설치할 수 있습니다.

중요

최초 설치 과정에서 설치 프로그램의 **create cluster** 명령을 한 번만 실행할 수 있습니다.

사전 요구 사항

- 클러스터를 호스팅하는 클라우드 플랫폼으로 계정을 구성합니다.
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

프로세스

1. 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터 배포를 초기화합니다.

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** 값으로 사용자 지정한 **./install-config.yaml** 파일의 위치를 지정합니다.

2 다른 설치 세부 사항을 보려면 **info** 대신 **warn**, **debug** 또는 **error**를 지정합니다.

참고

호스트에 구성된 클라우드 공급자 계정에 클러스터를 배포하기에 충분한 권한이 없는 경우, 설치 프로세스가 중단되고 누락된 권한을 알리는 메시지가 표시됩니다.

클러스터 배포가 완료되면 웹 콘솔로 연결되는 링크와 **kubeadmin** 사용자의 인증 정보가 포함된 클러스터 액세스 지침이 사용자 터미널에 표시됩니다.

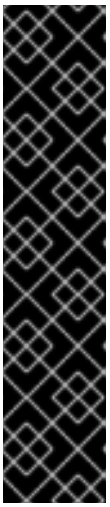
출력 예

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
```

```
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-Wt5AL"
INFO Time elapsed: 36m22s
```

참고

설치에 성공하면 클러스터 액세스 및 인증 정보도 **<installation_directory>/.openshift_install.log**로 출력됩니다.



중요

- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

중요

설치 프로그램에서 생성되는 파일이나 설치 프로그램을 삭제해서는 안 됩니다. 클러스터를 삭제하려면 두 가지가 모두 필요합니다.

2. 선택사항: 클러스터를 설치하는 데 사용한 IAM 계정에서 **AdministratorAccess** 정책을 제거하거나 비활성화합니다.

참고

AdministratorAccess 정책에서 제공하는 승격된 권한은 설치 중에만 필요합니다.

4.7.8. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift CLI(**oc**)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 **oc**를 설치할 수 있습니다.

중요

이전 버전의 **oc**를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 **oc**를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 Linux에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.

2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Linux Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. **oc** 바이너리를 **PATH**에 있는 디렉터리에 배치합니다.
PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치
다음 절차에 따라 Windows에 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Windows Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. ZIP 프로그램으로 아카이브의 압축을 풉니다.
5. **oc** 바이너리를 **PATH**에 있는 디렉터리로 이동합니다.
PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치
다음 절차에 따라 macOS에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 MacOSX Client** 항목 옆에 있는 **지금 다운로드를** 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 해제하고 압축을 풉니다.
5. **oc** 바이너리 PATH의 디렉터리로 이동합니다.
PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

4.7.9. CLI를 사용하여 클러스터에 로그인

클러스터 **kubeconfig** 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. **kubeconfig** 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- **oc** CLI를 설치했습니다.

프로세스

1. **kubeadmin** 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 **oc** 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

4.7.10. 기본 OperatorHub 소스 비활성화

Red Hat 및 커뮤니티 프로젝트에서 제공하는 콘텐츠를 소싱하는 Operator 카탈로그는 OpenShift Container Platform을 설치하는 동안 기본적으로 OperatorHub용으로 구성됩니다. 제한된 네트워크 환경에서는 클러스터 관리자로서 기본 카탈로그를 비활성화해야 합니다.

프로세스

- **OperatorHub** 오브젝트에 **disableAllDefaultSources: true**를 추가하여 기본 카탈로그의 소스를 비활성화합니다.

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

작은 정보

또는 웹 콘솔을 사용하여 카탈로그 소스를 관리할 수 있습니다. **관리 → 클러스터 설정 → 구성 → OperatorHub** 페이지에서 개별 소스를 생성, 삭제, 비활성화 및 활성화할 수 있는 소스 **탭을** 클릭합니다.

4.7.11. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보 를 참조하십시오.

4.7.12. 다음 단계

- 설치를 확인합니다.
- 클러스터를 사용자 지정합니다.
- Cluster Samples Operator 및 **must-gather** 툴의 이미지 스트림을 구성 합니다.
- 제한된 네트워크에서 Operator Lifecycle Manager (OLM) 사용 방법에 대해 살펴봅니다.
- 클러스터를 설치하는 데 사용한 미러 레지스트리에 신뢰할 수 있는 CA가 있는 경우 추가 신뢰 저장소를 구성하여 클러스터에 추가합니다.
- 필요한 경우 원격 상태 보고 옵트아웃 을 수행할 수 있습니다.

4.8. AWS의 클러스터를 기존 VPC에 설치

OpenShift Container Platform 4.9 버전에서는 AWS(Amazon Web Services)의 기존 Amazon VPC(Virtual Private Cloud)에 클러스터를 설치할 수 있습니다. 설치 프로그램이 나머지 필수 인프라를 프로비저닝하며, 이후에 추가로 사용자 지정할 수 있습니다. 설치를 사용자 지정하려면 클러스터를 설치하기 전에 **install-config.yaml** 파일에서 매개변수를 수정합니다.

4.8.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비 에 대한 문서를 읽습니다.
- 클러스터를 호스팅할 AWS 계정을 구성 했습니다.

중요

컴퓨터에 AWS 프로필이 저장되어 있는 경우 다단계 인증 장치를 사용하는 동안 생성한 임시 세션 토큰을 해당 프로필이 사용해서는 안 됩니다. 클러스터는 클러스터의 전체 수명 동안 현재 AWS 자격 증명을 계속 사용하여 AWS 리소스를 생성하므로 장기 자격 증명을 사용해야 합니다. 적절한 키를 생성하려면 AWS 문서의 IAM 사용자의 액세스 키 관리를 참조하십시오. 설치 프로그램을 실행할 때 키를 제공할 수 있습니다.

- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성 해야 합니다.
- 사용자의 환경에서 클라우드 ID 및 액세스 관리(IAM) API에 액세스할 수 없거나 **kube-system** 네임스페이스에 관리자 수준의 인증 정보 시크릿을 저장하지 않으려면 IAM 인증 정보 수동 생성 및 유지 관리를 참조하십시오.

4.8.2. 사용자 지정 VPC 사용 정보

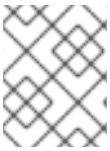
OpenShift Container Platform 4.9에서는 AWS(Amazon Web Services)의 기존 Amazon VPC(Virtual Private Cloud)에 있는 기존 서브넷에 클러스터를 배포할 수 있습니다. 기존 GCP VPC에 OpenShift Container Platform을 배포하면 새 계정의 한도 제한 적용을 받지 않거나 회사의 지침에 따른 운영 제한을 보다 쉽게 준수할 수 있습니다. VPC를 직접 생성하는 데 필요한 인프라 생성 권한을 받을 수 없는 경우 이 설치 옵션을 사용합니다.

설치 프로그램은 기존 서브넷에 있는 다른 구성 요소를 알 수 없으므로 사용자를 대신하여 서브넷 CIDR 등을 선택할 수 없습니다. 클러스터를 직접 설치하는 서브넷에 대한 네트워킹을 구성해야 합니다.

4.8.2.1. VPC 사용 요구사항

설치 프로그램은 더 이상 다음 구성 요소를 생성하지 않습니다.

- 인터넷 게이트웨이
- NAT 게이트웨이
- 서브넷
- 라우팅 테이블
- VPC
- VPC DHCP 옵션
- VPC 끝점



참고

설치 프로그램을 사용하려면 클라우드 제공 DNS 서버를 사용해야 합니다. 사용자 지정 DNS 서버 사용은 지원되지 않으며 이로 인해 설치에 실패합니다.

사용자 지정 VPC를 사용하는 경우, 사용할 클러스터와 설치 프로그램에 맞게 VPC와 해당 서브넷을 구성해야 합니다. AWS VPC 생성 및 관리에 대한 자세한 내용은 AWS 문서의 Amazon VPC 콘솔 마법사 구성 및 VPC 및 서브넷 작업을 참조하십시오.

설치 프로그램은 다음을 수행할 수 없습니다.

- 클러스터에서 사용할 네트워크 범위를 세분합니다.
- 서브넷의 라우팅 테이블을 설정합니다.
- DHCP와 같은 VPC 옵션을 설정합니다.

클러스터를 설치하기 전에 이러한 작업을 완료해야 합니다. AWS VPC의 네트워킹 구성에 대한 자세한 내용은 VPC 네트워킹 구성 요소 및 경로 테이블을 참조하십시오.

VPC는 다음 특성을 충족해야 합니다.

- 클러스터가 사용하는 각 가용성 영역에 대한 퍼블릭 및 프라이빗 서브넷을 생성합니다. 각 가용성 영역에는 퍼블릭 서브넷과 프라이빗 서브넷이 1개 이상 포함될 수 있습니다. 이 유형의 구성의 예는 AWS 문서의 퍼블릭 및 프라이빗 서브넷(NAT)이 있는 VPC 를 참조하십시오.
  각 서브넷 ID를 기록합니다. 설치를 완료하려면 **install-config.yaml** 파일의 **platform** 섹션에 이러한 값을 입력해야 합니다. AWS 문서에서 서브넷 ID 찾기를 참조하십시오.
- VPC의 CIDR 블록에는 클러스터 시스템의 IP 주소 풀인 **Networking.MachineCIDR** 범위가 포함되어야 합니다. 서브넷 CIDR 블록은 사용자가 지정하는 시스템 CIDR에 속해야 합니다.
- VPC에는 공용 인터넷 게이트웨이가 연결되어 있어야 합니다. 각 가용성 영역에 대해 다음을 수행합니다.
  - 공용 서브넷에는 인터넷 게이트웨이 경로가 필요합니다.
  - 공용 서브넷에는 EIP 주소가 있는 NAT 게이트웨이가 필요합니다.
  - 사설 서브넷에는 공용 서브넷의 NAT 게이트웨이에 대한 경로가 필요합니다.
- VPC는 **kubernetes.io/cluster/.*: owned** 태그를 사용해서는 안 됩니다.
  설치 프로그램은 **kubernetes.io/cluster/.*: shared** 태그를 추가하도록 서브넷을 수정하므로 서브넷에 사용 가능한 여유 태그 슬롯이 하나 이상 있어야 합니다. AWS 문서의 태그 제한 사항을 참조하여 설치 프로그램이 사용자가 지정하는 각 서브넷에 태그를 추가할 수 있는지 확인합니다.
- 클러스터가 VPC에 연결된 Route 53 영역을 사용하여 클러스터의 내부 DNS 레코드를 확인할 수 있도록 VPC에서 **enableDnsSupport 및 enableDnsHostnames** 속성을 활성화해야 합니다.
  AWS 문서에서 VPC의 DNS 지원을 참조하십시오.
  자체 Route 53 호스팅 프라이빗 영역을 사용하려면 클러스터를 설치하기 전에 기존 호스팅 영역을 VPC와 연결해야 합니다. **install-config.yaml** 파일에서 **platform.aws.hostedZone** 필드를 사용하여 호스팅 영역을 정의할 수 있습니다.

연결이 끊긴 환경에서 작업 중인 경우에는 EC2 및 ELB 끝점의 공용 IP 주소에 도달할 수 없습니다. 이 문제를 해결하려면 VPC 끝점을 생성하여 클러스터가 사용 중인 서브넷에 연결해야 합니다. 올바른 끝점의 이름은 다음과 같습니다.

정부 리전

- **ec2.<region>.amazonaws.com**
- **elasticloadbalancing.<region>.amazonaws.com**
- **s3.<region>.amazonaws.com**

보안 최상위 리전

- **ec2.<region>.c2s.ic.gov**

- **elasticloadbalancing.<region>.c2s.ic.gov**
- **s3.<region>.c2s.ic.gov**

필수 VPC 구성 요소

시스템과의 통신을 허용하는 서브넷과 적합한 VPC를 제공해야 합니다.

<table>
<tr><th>구성 요소</th><th>AWS 유형</th><th>설명</th></tr>
<tr><td>VPC</td><td>AWS::EC2::VPC
AWS::EC2::VPCEndpoint</td><td>클러스터에서 사용할 공용 VPC를 제공해야 합니다. VPC는 각 서브넷의 라우팅 테이블을 참조하는 끝점을 사용하여 S3에서 호스팅되는 레지스트리와의 통신을 개선합니다.</td></tr>
<tr><td>퍼블릭 서브넷</td><td>AWS::EC2::Subnet
AWS::EC2::SubnetNetworkAclAssociation</td><td>VPC에는 1~3개의 가용성 영역에 대한 퍼블릭 서브넷이 있어야 하며 이 서브넷을 적절한 인그레스 규칙과 연결해야 합니다.</td></tr>
<tr><td>인터넷 게이트웨이</td><td>AWS::EC2::InternetGateway
AWS::EC2::VPCGatewayAttachment
AWS::EC2::RouteTable
AWS::EC2::Route
AWS::EC2::SubnetRouteTableAssociation
AWS::EC2::NatGateway
AWS::EC2::EIP</td><td>공용 경로가 있는 공용 인터넷 게이트웨이가 VPC에 연결되어 있어야 합니다. 제공된 템플릿에서 각 퍼블릭 서브넷에는 EIP 주소를 갖는 NAT 게이트웨이가 있습니다. 이러한 NAT 게이트웨이를 사용하면 프라이빗 서브넷 인스턴스와 같은 클러스터 리소스가 인터넷에 도달할 수 있으므로 일부 제한된 네트워크 또는 프록시 시나리오에는 필요하지 않습니다.</td></tr>
<tr><td>네트워크 액세스 제어</td><td>AWS::EC2::NetworkAcl
AWS::EC2::NetworkAclEntry</td><td>VPC가 다음 포트에 액세스할 수 있어야 합니다.<table><tr><th>포트</th><th>이유</th></tr><tr><td>80</td><td>인바운드 HTTP 트래픽</td></tr><tr><td>443</td><td>인바운드 HTTPS 트래픽</td></tr><tr><td>22</td><td>인바운드 SSH 트래픽</td></tr></table></td></tr>
</table>

구성 요소	AWS 유형	설명	
		1024 - 65535	인바운드 임시 트래픽
		0 - 65535	아웃바운드 임시 트래픽
프라이빗 서브넷	• **AWS::EC2::Subnet** • **AWS::EC2::RouteTable** • **AWS::EC2::SubnetRouteTableAssociation**	VPC에 프라이빗 서브넷이 포함될 수 있습니다. 제공된 CloudFormation 템플릿은 1~3개 가용성 영역의 프라이빗 서브넷을 생성할 수 있습니다. 프라이빗 서브넷을 사용하는 경우 적절한 경로와 테이블을 제공해야 합니다.	

4.8.2.2. VPC 검증

제공한 서브넷이 적합한지 확인하기 위해 설치 프로그램이 다음 데이터를 확인합니다.

- 사용자가 지정하는 모든 서브넷이 있는지 여부.
- 사용자가 프라이빗 서브넷을 제공합니다.
- 서브넷 CIDR이 사용자가 지정한 시스템 CIDR에 속합니다.
- 각 가용성 영역에 서브넷을 제공합니다. 각 가용성 영역에는 퍼블릭 서브넷과 프라이빗 서브넷이 하나씩 포함됩니다. 개인 클러스터를 사용하는 경우 각 가용성 영역에 프라이빗 서브넷만 제공합니다. 그렇지 않으면 각 가용성 영역에 퍼블릭 서브넷과 프라이빗 서브넷을 하나씩만 제공합니다.
- 각 프라이빗 서브넷 가용성 영역에 퍼블릭 서브넷을 제공합니다. 프라이빗 서브넷을 제공하지 않는 가용성 영역에서는 시스템이 프로비저닝되지 않습니다.

기존 VPC를 사용하는 클러스터를 제거해도 VPC는 삭제되지 않습니다. VPC에서 OpenShift Container Platform 클러스터를 제거하면 클러스터가 사용한 서브넷에서 **kubernetes.io/cluster/.*: shared** 태그가 제거됩니다.

4.8.2.3. 권한 분할

OpenShift Container Platform 4.3부터 클러스터를 배포하는 데 설치 프로그램에서 프로비저닝한 인프라 클러스터에 필요한 권한 중 일부가 필요하지 않게 되었습니다. 이 변경 사항은 회사의 권한 분류와 유사합니다. 즉 일부 개인의 경우 클라우드에서 다른 사람들과 다른 리소스를 생성할 수 있습니다. 예를 들어 인스턴스, 버킷, 로드 밸런서와 같은 애플리케이션 관련 항목은 생성할 수 있지만 VPC 서브넷 또는 인그레스 규칙과 같은 네트워킹 관련 구성 요소는 생성하지 못할 수 있습니다.

클러스터를 생성할 때 사용하는 GCP 자격 증명에는 서브넷, 라우팅 테이블, 인터넷 게이트웨이, NAT, VPN과 같은 VPC 내 핵심 네트워킹 구성 요소와 VPC를 생성하는 데 필요한 네트워킹 권한이 필요하지 않습니다. 하지만 ELB, 보안 그룹, S3 버킷 및 노드와 같이 클러스터 내 시스템에 필요한 애플리케이션 리소스를 생성하려면 여전히 권한이 필요합니다.

4.8.2.4. 클러스터 간 격리

OpenShift Container Platform을 기존 네트워크에 배포하면 클러스터 서비스 격리가 다음과 같은 방식으로 감소합니다.

- 동일한 VPC에 여러 OpenShift Container Platform 클러스터를 설치할 수 있습니다.
- ICMP 인그레스가 전체 네트워크에서 허용됩니다.
- TCP 22 인그레스(SSH)가 전체 네트워크에 허용됩니다.
- 컨트롤 플레인 TCP 6443 인그레스(Kubernetes API)가 전체 네트워크에 허용됩니다.
- 컨트롤 플레인 TCP 22623 인그레스(MCS)가 전체 네트워크에 허용됩니다.

4.8.3. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.

중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

4.8.4. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 **core** 사용자의 **~/.ssh/authorized_keys** 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 **core**로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. **./openshift-install gather** 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

참고

AWS 키 쌍 과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: **~/.ssh/id_ed25519**)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 **'~/.ssh** 디렉터리에 있는지 확인하십시오.

참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 **ed25519** 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 **rsa** 또는 **ecdsa** 알고리즘을 사용하는 키를 생성합니다.

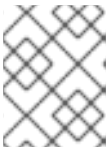
2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 **~/.ssh/id_ed25519.pub** 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 **./openshift-install gather** 명령을 사용하려는 경우 필요합니다.



참고

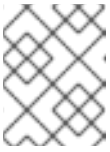
일부 배포에서는 **~/.ssh/id_rsa** 및 **~/.ssh/id_dsa**와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. **ssh-agent** 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```



참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. **ssh-agent**에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: **~/.ssh/id_ed25519**).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

4.8.5. 설치 프로그램 받기

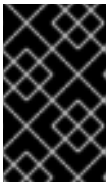
OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

프로세스

1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.
2. 인프라 공급자를 선택합니다.
3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.



중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.

중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 OpenShift Container Platform 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

4.8.6. 설치 구성 파일 만들기

AWS(Amazon Web Services)에 설치하는 OpenShift Container Platform 클러스터를 사용자 지정할 수 있습니다.

사전 요구 사항

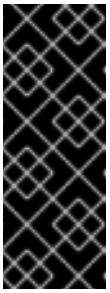
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.
- 서브스크립션 수준에서 서비스 주체 권한을 획득합니다.

절차

1. **install-config.yaml** 파일을 생성합니다.
   a. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

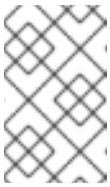
   1 **<installation_directory>**는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.



중요

비어 있는 디렉터리를 지정합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

   b. 화면에 나타나는 지시에 따라 클라우드에 대한 구성 세부 사항을 입력합니다.
      i. 선택사항: 클러스터 시스템에 액세스하는 데 사용할 SSH 키를 선택합니다.



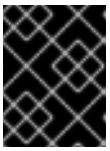
참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

      ii. 대상 플랫폼으로 **AWS**를 선택합니다.
      iii. 컴퓨터에 AWS(Amazon Web Services) 프로필이 저장되어 있지 않은 경우 설치 프로그램을 실행하도록 구성한 사용자의 AWS 액세스 키 ID와 시크릿 액세스 키를 입력합니다.
      iv. 클러스터를 배포할 AWS 리전을 선택합니다.
      v. 클러스터에 대해 구성한 Route53 서비스의 기본 도메인을 선택합니다.
      vi. 클러스터를 설명할 수 있는 이름을 입력합니다.
      vii. Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 붙여넣습니다.

2. **install-config.yaml** 파일을 수정합니다. 사용 가능한 매개변수에 대한 자세한 정보는 "설치 구성 매개변수" 섹션에서 확인할 수 있습니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.



중요

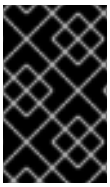
install-config.yaml 파일은 설치 과정에서 사용됩니다. 이 파일을 재사용하려면 지금 백업해야 합니다.

4.8.6.1. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 매개변수 값을 제공하여 클러스터를 호스팅할 클라우드 플랫폼에서 사용자 계정을 설명하고 선택사항으로 클러스터의 플랫폼을 사용자 지정합니다. **install-config.yaml** 설치 구성 파일을 생성할 때 명령줄을 통해 필요한 매개변수 값을 제공합니다. 클러스터를 사용자 지정하면 **install-config.yaml** 파일을 수정하여 플랫폼에 대한 세부 정보를 제공할 수 있습니다.

참고

설치한 후에는 **install-config.yaml** 파일에서 이러한 매개변수를 수정할 수 없습니다.



중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

4.8.6.1.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 4.22. 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).

매개변수	설명	값
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	소문자, 하이픈(**-**), 마침표(**.**)로 구성되는 문자열(예: **dev**)입니다.
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	{
		"auths":{
		"cloud.openshift.com":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		},
		"quay.io":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		}
		}
		}

4.8.6.1.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 4.23. 네트워크 매개변수

매개변수	설명	값

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체  **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.
networking.clusterNetwork	Pod의 IP 주소 블록입니다. 기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된 **cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16

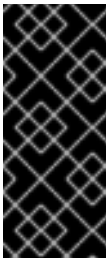
매개변수	설명	값
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16** **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

4.8.6.1.3. 선택적 구성 매개변수

선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 4.24. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열

매개변수	설명	값
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은 **3**입니다.
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열

매개변수	설명	값
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 인터넷에서 액세스할 수 없는 사설 클러스터를 배포하려면 **publish**를 **Internal**로 설정합니다. 기본값은 **External**입니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다.  **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

4.8.6.1.4. 선택적 AWS 구성 매개변수

선택적 AWS 구성 매개변수는 다음 표에 설명되어 있습니다.

표 4.25. 선택적 AWS 매개변수

매개변수	설명	값
compute.platform.aws.amiID	클러스터의 컴퓨팅 머신을 부팅하는 데 사용되는 AWS AMI입니다. 이는 사용자 지정 RHCOS AMI가 필요한 리전에 필요합니다.	설정된 AWS 리전에 속하는 게시된 또는 사용자 지정 RHCOS AMI입니다.
compute.platform.aws.iamRole	컴퓨팅 머신 풀 인스턴스 프로파일에 적용되는 기존 AWS IAM 역할입니다. 이러한 필드를 사용하여 이름 지정 체계와 일치하고 IAM 역할에 대해 사전 정의된 권한 경계를 포함할 수 있습니다. 정의되지 않은 경우 설치 프로그램은 새 IAM 역할을 생성합니다.	유효한 AWS IAM 역할의 이름입니다.
compute.platform.aws.rootVolume.iops	루트 볼륨용으로 예약된 IOPS(초당 입/출력 작업)입니다.	정수(예: **4000**)
compute.platform.aws.rootVolume.size	루트 볼륨의 크기(GiB)입니다.	정수(예: **500**)
compute.platform.aws.rootVolume.type	루트 볼륨의 유형입니다.	유효한 AWS EBS 볼륨 유형 (예: **io1**)

매개변수	설명	값
compute.platform.aws.rootVolume.kmsKeyARN	KMS 키의 Amazon 리소스 이름(키 ARN)입니다. 이는 특정 KMS 키를 사용하여 작업자 노드의 OS 볼륨을 암호화하는 데 필요합니다.	유효한 키 ID 또는 키 ARN 입니다.
compute.platform.aws.type	컴퓨팅 시스템의 EC2 인스턴스 유형입니다.	유효한 AWS 인스턴스 유형 (예: **m4.2xlarge**) 다음 **머신의 인스턴스 유형**을 참조하십시오.
compute.platform.aws.zones	설치 프로그램이 컴퓨팅 머신 풀에 필요한 시스템을 생성하는 가용성 영역입니다. 자체 VPC를 제공하는 경우 해당 가용성 영역에 서브넷을 제공해야 합니다.	**us-east-1c**와 같이 YAML 시퀀스로 표시되는 유효한 AWS 가용성 영역의 목록입니다.
compute.aws.region	설치 프로그램이 컴퓨팅 리소스를 생성하는 AWS 리전입니다.	유효한 모든 AWS 리전(예: **us-east-1**)
controlPlane.platform.aws.amiID	클러스터의 컨트롤 플레인 시스템을 시작하는 데 사용되는 AWS AMI입니다. 이는 사용자 지정 RHCOS AMI가 필요한 리전에 필요합니다.	설정된 AWS 리전에 속하는 게시된 또는 사용자 지정 RHCOS AMI입니다.
controlPlane.platform.aws.iamRole	컨트롤 플레인 시스템 풀 인스턴스 프로파일에 적용되는 기존 AWS IAM 역할입니다. 이러한 필드를 사용하여 이름 지정 체계와 일치하고 IAM 역할에 대해 사전 정의된 권한 경계를 포함할 수 있습니다. 정의되지 않은 경우 설치 프로그램은 새 IAM 역할을 생성합니다.	유효한 AWS IAM 역할의 이름입니다.
controlPlane.platform.aws.rootVolume.kmsKeyARN	KMS 키의 Amazon 리소스 이름(키 ARN)입니다. 이는 특정 KMS 키를 사용하여 컨트롤 플레인 노드의 OS 볼륨을 암호화하는 데 필요합니다.	유효한 키 ID 및 키 ARN.
controlPlane.platform.aws.type	컨트롤 플레인 시스템의 EC2 인스턴스 유형입니다.	유효한 AWS 인스턴스 유형 (예: **m5.xlarge**). 다음 **머신의 인스턴스 유형**을 참조하십시오.
controlPlane.platform.aws.zones	설치 프로그램이 컨트롤 플레인 시스템 풀에 필요한 시스템을 생성하는 가용성 영역입니다.	**us-east-1c**와 같이 YAML 시퀀스로 표시되는 유효한 AWS 가용성 영역의 목록입니다.

매개변수	설명	값
controlPlane.aws.region	설치 프로그램이 컨트롤 플레인 리소스를 생성하는 AWS 리전입니다.	유효한 AWS 리전(예: **us-east-1**)
platform.aws.amiID	클러스터의 모든 머신을 부팅하는 데 사용되는 AWS AMI입니다. 설정된 경우 AMI는 클러스터와 동일한 리전에 속해 있어야 합니다. 이는 사용자 지정 RHCOS AMI가 필요한 리전에 필요합니다.	설정된 AWS 리전에 속하는 게시된 또는 사용자 지정 RHCOS AMI입니다.
platform.aws.hostedZone	클러스터의 기존 Route 53 개인 호스팅 영역입니다. 자체 VPC를 제공하는 경우에만 기존 호스팅 영역을 사용할 수 있습니다. 호스팅 영역은 설치 전에 사용자 제공 VPC와 이미 연결되어 있어야 합니다. 또한 호스팅 영역의 도메인은 클러스터 도메인 또는 클러스터 도메인의 상위 도메인이어야 합니다. 정의되지 않은 경우 설치 프로그램은 새 호스팅 영역을 생성합니다.	문자열 (예: **Z3URY6TWQ91KVV**)
platform.aws.serviceEndpoints.name	AWS 서비스 엔드 포인트 이름. 사용자 지정 엔드 포인트는 FIPS와 같은 대체 AWS 엔드 포인트를 사용해야하는 경우에만 필요합니다. EC2, S3, IAM, Elastic Load Balancing, Tagging, Route 53 및 STS AWS 서비스에 대해 사용자 지정 API 엔드 포인트를 지정할 수 있습니다.	유효한 AWS 서비스 엔드 포인트 이름.
platform.aws.serviceEndpoints.url	AWS 서비스 엔드 포인트 URL. URL은 **https** 프로토콜을 사용해야하며 호스트는 인증서를 신뢰해야 합니다.	유효한 AWS 서비스 엔드 포인트 URL.
platform.aws.userTags	설치 프로그램이 생성하는 모든 리소스에 태그로 추가하는 키와 값의 맵입니다.	유효한 YAML 맵(예: **<key>: <value>** 형식의 키 값 쌍) AWS 태그에 대한 자세한 내용은 AWS 문서의 Amazon EC2 리소스 태그 지정을 참조하십시오.

매개변수	설명	값
platform.aws.subnets	설치 프로그램이 VPC를 자동으로 생성하지 않고 VPC를 직접 제공하는 경우 사용할 클러스터의 서브넷을 지정합니다. 서브넷은 사용자가 지정하는 동일한 **machineNetwork[].cidr** 범위의 일부여야 합니다. 표준 클러스터의 경우 각 가용성 영역의 퍼블릭 및 프라이빗 서브넷을 지정합니다. 개인 클러스터의 경우 각 가용성 영역의 프라이빗 서브넷을 지정합니다.	유효한 서브넷 ID.

4.8.6.2. 클러스터 설치를 위한 최소 리소스 요구 사항

각 클러스터 시스템이 다음과 같은 최소 요구사항을 충족해야 합니다.

표 4.26. 최소 리소스 요구사항

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
부트스트랩	RHCOS	4	16GB	100GB	300
컨트롤 플레인	RHCOS	4	16GB	100GB	300
컴퓨팅	RHCOS, RHEL 7.9 또는 RHEL 8.4 [3]	2	8GB	100GB	300

1. SMT(동시 멀티스레딩) 또는 하이퍼 스레딩이 활성화되지 않은 경우 하나의 vCPU는 하나의 물리적 코어와 동일합니다. 활성화하면 다음과 같은 공식을 사용하여 해당 비율을 계산합니다. (코어당 스레드 수 × 코어 수) × 소켓 수 = vCPU 수

2. OpenShift Container Platform 및 Kubernetes는 디스크 성능에 민감하며 특히 10ms p99 fsync 기간이 필요한 컨트롤 플레인 노드의 etcd에 더 빠른 스토리지가 권장됩니다. 많은 클라우드 플랫폼에서 스토리지 크기와 IOPS를 함께 확장되므로 충분한 성능을 얻으려면 스토리지 볼륨을 과도하게 할당해야 할 수 있습니다.

3. 사용자가 프로비저닝한 모든 설치와 마찬가지로 클러스터에서 RHEL 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. RHEL 7 컴퓨팅 머신 사용은 더 이상 사용되지 않으며 향후 OpenShift Container Platform 4 릴리스에서 제거될 예정입니다.

4.8.6.3. 지원되는 AWS 머신 유형

다음 AWS(Amazon Web Services) 인스턴스 유형은 OpenShift Container Platform에서 지원됩니다.

예 4.18. 시스템의 인스턴스 유형

인스턴스 유형	부트스트랩	컨트롤 플레인	컴퓨팅
i3.large	x		
m4.large			x
m4.xlarge		x	x
m4.2xlarge		x	x
m4.4xlarge		x	x
m4.10xlarge		x	x
m4.16xlarge		x	x
m5.large			x
m5.xlarge		x	x
m5.2xlarge		x	x
m5.4xlarge		x	x
m5.8xlarge		x	x
m5.12xlarge		x	x
m5.16xlarge		x	x
m5a.large			x
m5a.xlarge		x	x
m5a.2xlarge		x	x
m5a.4xlarge		x	x
m5a.8xlarge		x	x
m5a.12xlarge		x	x
m5a.16xlarge		x	x
m6i.xlarge		x	x

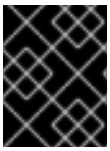
인스턴스 유형	부트스트랩	컨트롤 플레인	컴퓨팅
m6i.2xlarge		x	x
m6i.4xlarge		x	x
m6i.8xlarge		x	x
m6i.16xlarge		x	x
c4.2xlarge		x	x
c4.4xlarge		x	x
c4.8xlarge		x	x
c5.xlarge			x
c5.2xlarge		x	x
c5.4xlarge		x	x
c5.9xlarge		x	x
c5.12xlarge		x	x
c5.18xlarge		x	x
c5.24xlarge		x	x
c5a.xlarge			x
c5a.2xlarge		x	x
c5a.4xlarge		x	x
c5a.8xlarge		x	x
c5a.12xlarge		x	x
c5a.16xlarge		x	x
c5a.24xlarge		x	x
r4.large			x
r4.xlarge		x	x

인스턴스 유형	부트스트랩	컨트롤 플레인	컴퓨팅
r4.2xlarge		x	x
r4.4xlarge		x	x
r4.8xlarge		x	x
r4.16xlarge		x	x
r5.large			x
r5.xlarge		x	x
r5.2xlarge		x	x
r5.4xlarge		x	x
r5.8xlarge		x	x
r5.12xlarge		x	x
r5.16xlarge		x	x
r5.24xlarge		x	x
r5a.large			x
r5a.xlarge		x	x
r5a.2xlarge		x	x
r5a.4xlarge		x	x
r5a.8xlarge		x	x
r5a.12xlarge		x	x
r5a.16xlarge		x	x
r5a.24xlarge		x	x
t3.large			x
t3.xlarge			x
t3.2xlarge			x

인스턴스 유형	부트스트랩	컨트롤 플레인	컴퓨팅
t3a.large			x
t3a.xlarge			x
t3a.2xlarge			x

4.8.6.4. AWS용 샘플 사용자 지정 install-config.yaml 파일

설치 구성 파일(**install-config.yaml**)을 사용자 지정하여 OpenShift Container Platform 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.



중요

이 샘플 YAML 파일은 참조용으로만 제공됩니다. 설치 프로그램을 사용하여 **install-config.yaml** 파일을 받아서 수정해야 합니다.

```
apiVersion: v1
baseDomain: example.com (1)
credentialsMode: Mint (2)
controlPlane: (3) (4)
  hyperthreading: Enabled (5)
  name: master
  platform:
    aws:
      zones:
      - us-west-2a
      - us-west-2b
      rootVolume:
        iops: 4000
        size: 500
        type: io1 (6)
      type: m5.xlarge
  replicas: 3
compute: (7)
- hyperthreading: Enabled (8)
  name: worker
  platform:
    aws:
      rootVolume:
        iops: 2000
        size: 500
        type: io1 (9)
      type: c5.4xlarge
      zones:
      - us-west-2c
  replicas: 3
metadata:
```

```
  name: test-cluster 10
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  aws:
    region: us-west-2 11
    userTags:
      adminContact: jdoe
      costCenter: 7536
    subnets: 12
    - subnet-1
    - subnet-2
    - subnet-3
    amiID: ami-96c6f8f7 13
    serviceEndpoints: 14
      - name: ec2
        url: https://vpce-id.ec2.us-west-2.vpce.amazonaws.com
    hostedZone: Z3URY6TWQ91KVV 15
fips: false 16
sshKey: ssh-ed25519 AAAA... 17
pullSecret: '{"auths": ...}' 18
```

1 10 11 18 필수 항목입니다. 설치 프로그램에서 이 값을 입력하라는 메시지를 표시합니다.

2 옵션: 이 매개 변수를 추가하여 Cloud Credential Operator (CCO)가 인증 정보의 기능을 동적으로 판별하도록 하지 않고 CCO가 지정된 모드를 사용하도록 강제합니다. CCO 모드에 대한 자세한 내용은 *Platform Operators 참조* 콘텐츠의 *Cloud Credential Operator* 항목을 참조하십시오.

3 7 이러한 매개변수와 값을 지정하지 않으면 설치 프로그램은 기본값을 적용합니다.

4 **controlPlane** 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

5 8 동시 멀티스레딩 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. 일부 클러스터 시스템에서 동시 멀티스레딩을 비활성화할 경우에는 해당 멀티스레딩을 모든 클러스터 시스템에서 비활성화해야 합니다.

중요

동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다. 동시 멀티스레딩을 비활성화하는 경우 시스템에 더 큰 인스턴스 유형(예: **m4.2xlarge** 또는 **m5.2xlarge**)을 사용합니다.

6 9 대규모의 클러스터의 경우 고속 etcd 스토리지를 구성하려면 스토리지 유형을 **IO1**로 설정하고 **iops**는 **2000**으로 설정합니다.

12 자체 VPC를 제공하는 경우 클러스터가 사용하는 각 가용성 영역의 서브넷을 지정합니다.

13 클러스터 머신을 시작하는 데 사용되는 AMI의 ID입니다. 설정된 경우 AMI는 클러스터와 동일한 리전에 속해 있어야 합니다.

14 AWS 서비스 엔드 포인트입니다. 알 수 없는 AWS 리전에 설치할 때 사용자 지정 엔드 포인트가 필요합니다. 엔드포인트 URL은 **https** 프로토콜을 사용해야하며 호스트는 인증서를 신뢰해야 합니다.

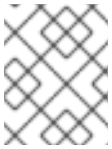
15 기존 Route 53 개인 호스팅 영역의 ID입니다. 기존 호스팅 영역을 제공하려면 클러스터를 설치하기 전에 자체 VPC를 제공하고 호스팅 영역이 VPC와 연결되어 있어야 합니다. 정의되지 않은 경우 설치 프로그램은 새 호스팅 영역을 생성합니다.

16 FIPS 모드 활성화 또는 비활성화 여부입니다. 기본적으로 FIPS 모드는 비활성화됩니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.

중요

FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.

17 선택사항으로, 클러스터의 시스템에 액세스하는 데 사용할 **sshKey** 값을 제공할 수도 있습니다.



참고

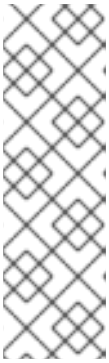
설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

4.8.6.5. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 HTTP 또는 HTTPS 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 OpenShift Container Platform 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 API에 대한 호출을 포함하여 모든 클러스터 발신(Egress) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.



참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 Red Hat OpenStack Platform (RHOSP)에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

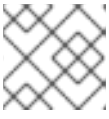
- **ec2.<region>.amazonaws.com,elasticloadbalancing.<region>.amazonaws.com** 및 **s3.<region>.amazonaws.com** 끝점을 VPC 끝점에 추가했습니다. 이러한 끝점은 노드에서 AWS EC2 API로 요청을 완료하는 데 필요합니다. 프록시는 노드 수준이 아닌 컨테이너 수준에서 작동하므로 이러한 요청을 AWS 사설 네트워크를 통해 AWS EC2 API로 라우팅해야 합니다. 프록시 서버의 허용 목록에 EC2 API의 공용 IP 주소를 추가하는 것만으로는 충분하지 않습니다.

절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. 클러스터 외부에서 HTTP 연결을 구축하는 데 사용할 프록시 URL입니다. URL 스키마는 **http**여야 합니다.
2. 클러스터 외부에서 HTTPS 연결을 구축하는 데 사용할 프록시 URL입니다.
3. 대상 도메인 이름, IP 주소 또는 프록시에서 제외할 기타 네트워크 CIDR로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. *를 사용하여 모든 대상에 대해 프록시를 바이패스합니다.
4. 이 값을 제공하면 설치 프로그램에서 추가 CA 인증서를 보유할 **openshift -config 네임스페이스에 user-ca- bundle** 이라는 구성 맵을 생성합니다. **additionalTrustBundle** 및 하나 이상의 프록시 설정을 제공하는 경우 **프록시** 오브젝트는 **trustedCA** 필드의 **user-ca-bundle** 구성 맵을 참조하도록 구성됩니다. 그러면 Cluster Network Operator에서 **trustedCA** 매개변수에 대해 지정된 콘텐츠를 RHCOS 신뢰 번들과 병합하는 **trusted-ca-bundle** 구성 맵을 생성합니다. 프록시의 ID 인증서를 RHCOS 트러스트 번들에 있는 기관에서 서명하지 않은 경우 **additionalTrustBundle** 필드가 있어야 합니다.



참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 OpenShift Container Platform을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 nil이 됩니다.

참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

4.8.7. 클러스터 배포

호환되는 클라우드 플랫폼에 OpenShift Container Platform을 설치할 수 있습니다.

중요

최초 설치 과정에서 설치 프로그램의 **create cluster** 명령을 한 번만 실행할 수 있습니다.

사전 요구 사항

- 클러스터를 호스팅하는 클라우드 플랫폼으로 계정을 구성합니다.
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

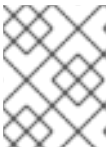
프로세스

1. 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터 배포를 초기화합니다.

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** 값으로 사용자 지정한 **./install-config.yaml** 파일의 위치를 지정합니다.

2 다른 설치 세부 사항을 보려면 **info** 대신 **warn**, **debug** 또는 **error**를 지정합니다.



참고

호스트에 구성된 클라우드 공급자 계정에 클러스터를 배포하기에 충분한 권한이 없는 경우, 설치 프로세스가 중단되고 누락된 권한을 알리는 메시지가 표시됩니다.

클러스터 배포가 완료되면 웹 콘솔로 연결되는 링크와 **kubeadmin** 사용자의 인증 정보가 포함된 클러스터 액세스 지침이 사용자 터미널에 표시됩니다.

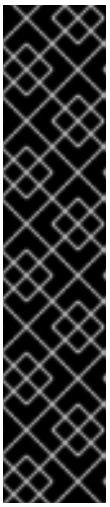
출력 예

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
```

```
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-Wt5AL"
INFO Time elapsed: 36m22s
```

참고

설치에 성공하면 클러스터 액세스 및 인증 정보도 **<installation_directory>/.openshift_install.log**로 출력됩니다.



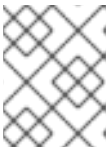
중요

- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

중요

설치 프로그램에서 생성되는 파일이나 설치 프로그램을 삭제해서는 안 됩니다. 클러스터를 삭제하려면 두 가지가 모두 필요합니다.

2. 선택사항: 클러스터를 설치하는 데 사용한 IAM 계정에서 **AdministratorAccess** 정책을 제거하거나 비활성화합니다.



참고

AdministratorAccess 정책에서 제공하는 승격된 권한은 설치 중에만 필요합니다.

4.8.8. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift CLI(**oc**)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 **oc**를 설치할 수 있습니다.

중요

이전 버전의 **oc**를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 **oc**를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 Linux에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.

2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Linux Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. **oc** 바이너리를 **PATH**에 있는 디렉터리에 배치합니다.
PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치
다음 절차에 따라 Windows에 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Windows Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. ZIP 프로그램으로 아카이브의 압축을 풉니다.
5. **oc** 바이너리를 **PATH**에 있는 디렉터리로 이동합니다.
PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치
다음 절차에 따라 macOS에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 MacOSX Client** 항목 옆에 있는 **지금 다운로드를** 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 해제하고 압축을 풉니다.
5. **oc** 바이너리 PATH의 디렉터리로 이동합니다.
PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

4.8.9. CLI를 사용하여 클러스터에 로그인

클러스터 **kubeconfig** 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. **kubeconfig** 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- **oc** CLI를 설치했습니다.

프로세스

1. **kubeadmin** 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 **oc** 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

4.8.10. 웹 콘솔을 사용하여 클러스터에 로그인

kubeadmin 사용자는 OpenShift Container Platform 설치 후 기본적으로 존재합니다. OpenShift Container Platform 웹 콘솔을 사용하여 **kubeadmin** 사용자로 클러스터에 로그인할 수 있습니다.

사전 요구 사항

- 설치 호스트에 대한 액세스 권한이 있어야 합니다.
- 클러스터 설치를 완료했으며 모든 클러스터 Operator를 사용할 수 있습니다.

프로세스

1. 설치 호스트의 **kubeadmin-password** 파일에서 **kubeadmin** 사용자의 암호를 가져옵니다.

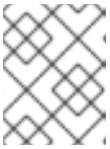
```
$ cat <installation_directory>/auth/kubeadmin-password
```

참고

대안으로 설치 호스트의 **<installation_directory>/.openshift_install.log** 로그 파일에서 **kubeadmin** 암호를 가져올 수 있습니다.

2. OpenShift Container Platform 웹 콘솔 경로를 나열합니다.

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```



참고

대안으로 설치 호스트의 **<installation_directory>/.openshift_install.log** 로그 파일에서 OpenShift Container Platform 경로를 가져올 수 있습니다.

출력 예

```
console     console-openshift-console.apps.<cluster_name>.<base_domain>           console
https   reencrypt/Redirect   None
```

3. 웹 브라우저의 이전 명령 출력에 자세히 설명된 경로로 이동하고 **kubeadmin** 사용자로 로그인합니다.

추가 리소스

- OpenShift Container Platform 웹 콘솔 액세스 및 이해에 대한 자세한 내용은 웹 콘솔에 액세스를 참조하십시오.

4.8.11. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

4.8.12. 다음 단계

- 설치를 확인합니다.
- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.
- 필요한 경우 클라우드 공급자 인증 정보를 제거 할 수 있습니다.

4.9. AWS에 개인 클러스터 설치

OpenShift Container Platform 4.9 버전에서는 AWS(Amazon Web Services)의 기존 VPC에 프라이빗 클러스터를 설치할 수 있습니다. 설치 프로그램이 나머지 필수 인프라를 프로비저닝하며, 이후에 추가로 사용자 지정할 수 있습니다. 설치를 사용자 지정하려면 클러스터를 설치하기 전에 **install-config.yaml** 파일에서 매개변수를 수정합니다.

4.9.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비 에 대한 문서를 읽습니다.
- 클러스터를 호스팅할 AWS 계정을 구성했습니다.

중요

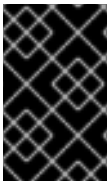
컴퓨터에 AWS 프로필이 저장되어 있는 경우 다단계 인증 장치를 사용하는 동안 생성한 임시 세션 토큰을 해당 프로필이 사용해서는 안 됩니다. 클러스터는 클러스터의 전체 수명 동안 현재 AWS 자격 증명을 계속 사용하여 AWS 리소스를 생성하므로 장기 자격 증명을 사용해야 합니다. 적절한 키를 생성하려면 AWS 문서의 IAM 사용자의 액세스 키 관리를 참조하십시오. 설치 프로그램을 실행할 때 키를 제공할 수 있습니다.

- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성해야 합니다.
- 사용자의 환경에서 클라우드 ID 및 액세스 관리(IAM) API에 액세스할 수 없거나 **kube-system** 네임스페이스에 관리자 수준의 인증 정보 시크릿을 저장하지 않으려면 IAM 인증 정보 수동 생성 및 유지 관리를 참조하십시오.

4.9.2. 프라이빗 클러스터

외부 엔드 포인트를 노출하지 않는 비공개 OpenShift Container Platform 클러스터를 배포할 수 있습니다. 프라이빗 클러스터는 내부 네트워크에서만 액세스할 수 있으며 인터넷에 표시되지 않습니다.

기본적으로 OpenShift Container Platform은 공개적으로 액세스 가능한 DNS 및 끝점을 사용하여 프로비저닝됩니다. 따라서 개인 클러스터를 배포할 때 클러스터에서 DNS, Ingress Controller 및 API 서버를 비공개로 설정할 수 있습니다. 즉 클러스터 리소스는 내부 네트워크에서만 액세스할 수 있고 인터넷에는 노출되지 않습니다.



중요

클러스터에 퍼블릭 서브넷이 있는 경우 관리자가 생성한 로드 밸런서 서비스에 공개적으로 액세스할 수 있습니다. 클러스터 보안을 보장하기 위해 이러한 서비스에 명시적으로 주석이 지정되었는지 확인합니다.

프라이빗 클러스터를 배포하려면 다음을 수행해야 합니다.

- 요구 사항을 충족하는 기존 네트워킹을 사용합니다. 네트워크의 다른 클러스터 사이에 클러스터 리소스를 공유할 수 있습니다.
- 다음에 액세스할 수 있는 머신에서 배포합니다.

- 프로비저닝하는 클라우드용 API 서비스
- 프로비저닝하는 네트워크의 호스트
- 설치 미디어를 가져올 인터넷

이러한 액세스 요구사항을 충족하고 회사의 지침을 따르는 모든 시스템을 사용할 수 있습니다. 클라우드 네트워크의 배스천 호스트 또는 VPN을 통해 네트워크에 액세스할 수 있는 시스템 등을 예로 들 수 있습니다.

4.9.2.1. AWS의 개인 클러스터

AWS(Amazon Web Services)에 개인 클러스터를 생성하려면 클러스터를 호스팅할 기존 프라이빗 VPC와 서브넷을 제공해야 합니다. 또한 설치 프로그램에서 클러스터에 필요한 DNS 레코드를 확인할 수 있어야 합니다. 설치 프로그램은 개인 네트워크에서만 액세스할 수 있도록 Ingress Operator 및 API 서버를 구성합니다.

클러스터가 AWS API에 액세스하려면 여전히 인터넷 접속이 필요합니다.

다음은 프라이빗 클러스터를 설치할 때 필요하지 않거나 생성되지 않는 항목들입니다.

- 퍼블릭 서브넷
- 공용 인그레스를 지원하는 공용 로드 밸런서
- 클러스터의 **baseDomain**과 일치하는 공용 Route 53 영역

설치 프로그램은 사용자가 지정하는 **baseDomain**을 사용하여 프라이빗 Route 53 영역과 클러스터에 필요한 레코드를 생성합니다. Operator가 클러스터에 대한 공용 레코드를 생성하지 않고 사용자가 지정하는 프라이빗 서브넷에 모든 클러스터 시스템이 배치되도록 클러스터가 구성됩니다.

4.9.2.1.1. 제한

프라이빗 클러스터에 공용 기능을 추가하는 기능은 제한됩니다.

- 설치 후에는 VPC에서 사용 중인 각 가용성 영역의 퍼블릭 서브넷 생성, 공용 로드 밸런서 생성, 6443(Kubernetes API 포트)의 인터넷 트래픽을 허용하도록 컨트롤 플레인 보안 그룹 구성 등 추가 조치 없이 Kubernetes API 엔드포인트를 공개할 수 없습니다.
- 공용 서비스 유형 로드 밸런서를 사용하는 경우 AWS가 공용 로드 밸런서를 생성하는 데 사용할 수 있도록 각 가용성 영역의 퍼블릭 서브넷에 **kubernetes.io/cluster/<cluster-infra-id>: shared** 태그를 지정해야 합니다.

4.9.3. 사용자 지정 VPC 사용 정보

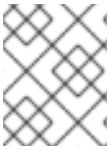
OpenShift Container Platform 4.9에서는 AWS(Amazon Web Services)의 기존 Amazon VPC(Virtual Private Cloud)에 있는 기존 서브넷에 클러스터를 배포할 수 있습니다. 기존 GCP VPC에 OpenShift Container Platform을 배포하면 새 계정의 한도 제한 적용을 받지 않거나 회사의 지침에 따른 운영 제한을 보다 쉽게 준수할 수 있습니다. VPC를 직접 생성하는 데 필요한 인프라 생성 권한을 받을 수 없는 경우 이 설치 옵션을 사용합니다.

설치 프로그램은 기존 서브넷에 있는 다른 구성 요소를 알 수 없으므로 사용자를 대신하여 서브넷 CIDR 등을 선택할 수 없습니다. 클러스터를 직접 설치하는 서브넷에 대한 네트워킹을 구성해야 합니다.

4.9.3.1. VPC 사용 요구사항

설치 프로그램은 더 이상 다음 구성 요소를 생성하지 않습니다.

- 인터넷 게이트웨이
- NAT 게이트웨이
- 서브넷
- 라우팅 테이블
- VPC
- VPC DHCP 옵션
- VPC 끝점



참고

설치 프로그램을 사용하려면 클라우드 제공 DNS 서버를 사용해야 합니다. 사용자 지정 DNS 서버 사용은 지원되지 않으며 이로 인해 설치에 실패합니다.

사용자 지정 VPC를 사용하는 경우, 사용할 클러스터와 설치 프로그램에 맞게 VPC와 해당 서브넷을 구성해야 합니다. AWS VPC 생성 및 관리에 대한 자세한 내용은 AWS 문서의 Amazon VPC 콘솔 마법사 구성 및 VPC 및 서브넷 작업을 참조하십시오.

설치 프로그램은 다음을 수행할 수 없습니다.

- 클러스터에서 사용할 네트워크 범위를 세분합니다.
- 서브넷의 라우팅 테이블을 설정합니다.
- DHCP와 같은 VPC 옵션을 설정합니다.

클러스터를 설치하기 전에 이러한 작업을 완료해야 합니다. AWS VPC의 네트워킹 구성에 대한 자세한 내용은 VPC 네트워킹 구성 요소 및 경로 테이블을 참조하십시오.

VPC는 다음 특성을 충족해야 합니다.

- VPC는 **kubernetes.io/cluster/.*: owned** 태그를 사용해서는 안 됩니다.
  설치 프로그램은 **kubernetes.io/cluster/.*: shared** 태그를 추가하도록 서브넷을 수정하므로 서브넷에 사용 가능한 여유 태그 슬롯이 하나 이상 있어야 합니다. AWS 문서의 태그 제한 사항을 참조하여 설치 프로그램이 사용자가 지정하는 각 서브넷에 태그를 추가할 수 있는지 확인합니다.

- 클러스터가 VPC에 연결된 Route 53 영역을 사용하여 클러스터의 내부 DNS 레코드를 확인할 수 있도록 VPC에서 **enableDnsSupport 및 enableDnsHostnames** 속성을 활성화해야 합니다.
  AWS 문서에서 VPC의 DNS 지원을 참조하십시오.
  자체 Route 53 호스팅 프라이빗 영역을 사용하려면 클러스터를 설치하기 전에 기존 호스팅 영역을 VPC와 연결해야 합니다. **install-config.yaml** 파일에서 **platform.aws.hostedZone** 필드를 사용하여 호스팅 영역을 정의할 수 있습니다.

- 공용 액세스 권한이 있는 클러스터를 사용하는 경우 클러스터가 사용하는 각 가용성 영역의 퍼블릭 및 프라이빗 서브넷을 만들어야 합니다. 각 가용성 영역에는 퍼블릭 서브넷과 프라이빗 서브넷이 1개 이상 포함될 수 있습니다.

연결이 끊긴 환경에서 작업 중인 경우에는 EC2 및 ELB 끝점의 공용 IP 주소에 도달할 수 없습니다. 이 문제를 해결하려면 VPC 끝점을 생성하여 클러스터가 사용 중인 서브넷에 연결해야 합니다. 올바른 끝점의 이름은 다음과 같습니다.

정부 리전

- **ec2.<region>.amazonaws.com**
- **elasticloadbalancing.<region>.amazonaws.com**
- **s3.<region>.amazonaws.com**

보안 최상위 리전

- **ec2.<region>.c2s.ic.gov**
- **elasticloadbalancing.<region>.c2s.ic.gov**
- **s3.<region>.c2s.ic.gov**

필수 VPC 구성 요소

시스템과의 통신을 허용하는 서브넷과 적합한 VPC를 제공해야 합니다.

구성 요소	AWS 유형	설명
VPC	• **AWS::EC2::VPC** • **AWS::EC2::VPCEndpoint**	클러스터에서 사용할 공용 VPC를 제공해야 합니다. VPC는 각 서브넷의 라우팅 테이블을 참조하는 끝점을 사용하여 S3에서 호스팅되는 레지스트리와의 통신을 개선합니다.
퍼블릭 서브넷	• **AWS::EC2::Subnet** • **AWS::EC2::SubnetNetworkAclAssociation**	VPC에는 1~3개의 가용성 영역에 대한 퍼블릭 서브넷이 있어야 하며 이 서브넷을 적절한 인그레스 규칙과 연결해야 합니다.
인터넷 게이트웨이	• **AWS::EC2::InternetGateway** • **AWS::EC2::VPCGatewayAttachment** • **AWS::EC2::RouteTable** • **AWS::EC2::Route** • **AWS::EC2::SubnetRouteTableAssociation** • **AWS::EC2::NatGateway** • **AWS::EC2::EIP**	공용 경로가 있는 공용 인터넷 게이트웨이가 VPC에 연결되어 있어야 합니다. 제공된 템플릿에서 각 퍼블릭 서브넷에는 EIP 주소를 갖는 NAT 게이트웨이가 있습니다. 이러한 NAT 게이트웨이를 사용하면 프라이빗 서브넷 인스턴스와 같은 클러스터 리소스가 인터넷에 도달할 수 있으므로 일부 제한된 네트워크 또는 프록시 시나리오에는 필요하지 않습니다.

<table>
<tr><th>구성 요소</th><th>AWS 유형</th><th colspan="2">설명</th></tr>
<tr><td rowspan="7">네트워크 액세스 제어</td><td rowspan="7">AWS::EC2::NetworkAcl
AWS::EC2::NetworkAclEntry</td><td colspan="2">VPC가 다음 포트에 액세스할 수 있어야 합니다.</td></tr>
<tr><td>포트</td><td>이유</td></tr>
<tr><td>80</td><td>인바운드 HTTP 트래픽</td></tr>
<tr><td>443</td><td>인바운드 HTTPS 트래픽</td></tr>
<tr><td>22</td><td>인바운드 SSH 트래픽</td></tr>
<tr><td>1024 - 65535</td><td>인바운드 임시 트래픽</td></tr>
<tr><td>0 - 65535</td><td>아웃바운드 임시 트래픽</td></tr>
<tr><td>프라이빗 서브넷</td><td>AWS::EC2::Subnet
AWS::EC2::RouteTable
AWS::EC2::SubnetRouteTableAssociation</td><td colspan="2">VPC에 프라이빗 서브넷이 포함될 수 있습니다. 제공된 CloudFormation 템플릿은 1~3개 가용성 영역의 프라이빗 서브넷을 생성할 수 있습니다. 프라이빗 서브넷을 사용하는 경우 적절한 경로와 테이블을 제공해야 합니다.</td></tr>
</table>

4.9.3.2. VPC 검증

제공한 서브넷이 적합한지 확인하기 위해 설치 프로그램이 다음 데이터를 확인합니다.

- 사용자가 지정하는 모든 서브넷이 있는지 여부.
- 사용자가 프라이빗 서브넷을 제공합니다.
- 서브넷 CIDR이 사용자가 지정한 시스템 CIDR에 속합니다.
- 각 가용성 영역에 서브넷을 제공합니다. 각 가용성 영역에는 퍼블릭 서브넷과 프라이빗 서브넷이 하나씩 포함됩니다. 개인 클러스터를 사용하는 경우 각 가용성 영역에 프라이빗 서브넷만 제공합니다. 그렇지 않으면 각 가용성 영역에 퍼블릭 서브넷과 프라이빗 서브넷을 하나씩만 제공합니다.
- 각 프라이빗 서브넷 가용성 영역에 퍼블릭 서브넷을 제공합니다. 프라이빗 서브넷을 제공하지 않는 가용성 영역에서는 시스템이 프로비저닝되지 않습니다.

기존 VPC를 사용하는 클러스터를 제거해도 VPC는 삭제되지 않습니다. VPC에서 OpenShift Container Platform 클러스터를 제거하면 클러스터가 사용한 서브넷에서 **kubernetes.io/cluster/.*: shared** 태그가 제거됩니다.

4.9.3.3. 권한 분할

OpenShift Container Platform 4.3부터 클러스터를 배포하는 데 설치 프로그램에서 프로비저닝한 인프라 클러스터에 필요한 권한 중 일부가 필요하지 않게 되었습니다. 이 변경 사항은 회사의 권한 분류와 유사합니다. 즉 일부 개인의 경우 클라우드에서 다른 사람들과 다른 리소스를 생성할 수 있습니다. 예를 들어 인스턴스, 버킷, 로드 밸런서와 같은 애플리케이션 관련 항목은 생성할 수 있지만 VPC 서브넷 또는 인그레스 규칙과 같은 네트워킹 관련 구성 요소는 생성하지 못할 수 있습니다.

클러스터를 생성할 때 사용하는 GCP 자격 증명에는 서브넷, 라우팅 테이블, 인터넷 게이트웨이, NAT, VPN과 같은 VPC 내 핵심 네트워킹 구성 요소와 VPC를 생성하는 데 필요한 네트워킹 권한이 필요하지 않습니다. 하지만 ELB, 보안 그룹, S3 버킷 및 노드와 같이 클러스터 내 시스템에 필요한 애플리케이션 리소스를 생성하려면 여전히 권한이 필요합니다.

4.9.3.4. 클러스터 간 격리

OpenShift Container Platform을 기존 네트워크에 배포하면 클러스터 서비스 격리가 다음과 같은 방식으로 감소합니다.

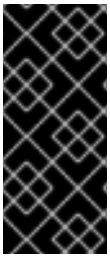
- 동일한 VPC에 여러 OpenShift Container Platform 클러스터를 설치할 수 있습니다.
- ICMP 인그레스가 전체 네트워크에서 허용됩니다.
- TCP 22 인그레스(SSH)가 전체 네트워크에 허용됩니다.
- 컨트롤 플레인 TCP 6443 인그레스(Kubernetes API)가 전체 네트워크에 허용됩니다.
- 컨트롤 플레인 TCP 22623 인그레스(MCS)가 전체 네트워크에 허용됩니다.

4.9.4. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.



중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

4.9.5. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 **core** 사용자의 **~/.ssh/authorized_keys** 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 **core**로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. **./openshift-install gather** 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

참고

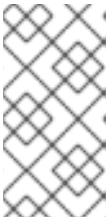
AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: **~/.ssh/id_ed25519**)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 **'~/.ssh** 디렉터리에 있는지 확인하십시오.



참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 **ed25519** 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 **rsa** 또는 **ecdsa** 알고리즘을 사용하는 키를 생성합니다.

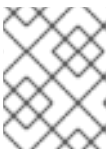
2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 **~/.ssh/id_ed25519.pub** 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 **./openshift-install gather** 명령을 사용하려는 경우 필요합니다.



참고

일부 배포에서는 **~/.ssh/id_rsa** 및 **~/.ssh/id_dsa**와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. **ssh-agent** 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

-

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. **ssh-agent**에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: **~/.ssh/id_ed25519**).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

4.9.6. 설치 프로그램 받기

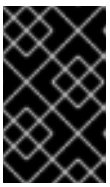
OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

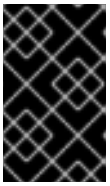
프로세스

1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.
2. 인프라 공급자를 선택합니다.
3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.



중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.



중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 OpenShift Container Platform 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

4.9.7. 수동으로 설치 구성 파일 만들기

내부 네트워크에서만 액세스할 수 있고 인터넷에 표시되지 않는 프라이빗 OpenShift Container Platform 클러스터 설치의 경우 설치 구성 파일을 수동으로 생성해야 합니다.

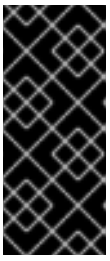
사전 요구 사항

- 로컬 시스템에 설치 프로그램에 제공할 SSH 공개 키가 있습니다. 키는 디버깅 및 재해 복구를 위해 클러스터 노드에 대한 SSH 인증에 사용됩니다.
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿이 있습니다.

프로세스

1. 필요한 설치 자산을 저장할 설치 디렉터리를 만듭니다.

```
$ mkdir <installation_directory>
```



중요

디렉터리를 만들어야 합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

2. 샘플 **install-config.yaml** 파일 템플릿을 사용자 지정하여 **<installation_directory>**에 저장합니다.

참고

이 설정 파일의 이름을 **install-config.yaml**로 지정해야 합니다.

참고

일부 플랫폼 유형의 경우 대체로 **./openshift-install create install-config --dir <installation_directory>** 를 실행하여 **install-config.yaml** 파일을 생성할 수 있습니다. 프롬프트에서 클러스터 구성에 대한 세부 정보를 제공할 수 있습니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.

중요

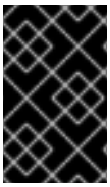
install-config.yaml 파일은 설치 과정의 다음 단계에서 사용됩니다. 이 시점에서 이를 백업해야 합니다.

4.9.7.1. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 매개변수 값을 제공하여 클러스터를 호스팅할 클라우드 플랫폼에서 사용자 계정을 설명하고 선택사항으로 클러스터의 플랫폼을 사용자 지정합니다. **install-config.yaml** 설치 구성 파일을 생성할 때 명령줄을 통해 필요한 매개변수 값을 제공합니다. 클러스터를 사용자 지정하면 **install-config.yaml** 파일을 수정하여 플랫폼에 대한 세부 정보를 제공할 수 있습니다.

참고

설치한 후에는 **install-config.yaml** 파일에서 이러한 매개변수를 수정할 수 없습니다.



중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

4.9.7.1.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 4.27. 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).

매개변수	설명	값
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	소문자, 하이픈(**-**), 마침표(**.**)로 구성되는 문자열(예: **dev**)입니다.
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

4.9.7.1.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 4.28. 네트워크 매개변수

매개변수	설명	값

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체 **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.
networking.clusterNetwork	Pod의 IP 주소 블록입니다. 기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된**cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16

매개변수	설명	값
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16** **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

4.9.7.1.3. 선택적 구성 매개변수

선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 4.29. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열

매개변수	설명	값
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은 **3**입니다.
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열

매개변수	설명	값
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 인터넷에서 액세스할 수 없는 사설 클러스터를 배포하려면 **publish**를 **Internal**로 설정합니다. 기본값은 **External**입니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

4.9.7.1.4. 선택적 AWS 구성 매개변수

선택적 AWS 구성 매개변수는 다음 표에 설명되어 있습니다.

표 4.30. 선택적 AWS 매개변수

매개변수	설명	값
compute.platform.aws.amiID	클러스터의 컴퓨팅 머신을 부팅하는 데 사용되는 AWS AMI입니다. 이는 사용자 지정 RHCOS AMI가 필요한 리전에 필요합니다.	설정된 AWS 리전에 속하는 게시된 또는 사용자 지정 RHCOS AMI입니다.
compute.platform.aws.iamRole	컴퓨팅 머신 풀 인스턴스 프로파일에 적용되는 기존 AWS IAM 역할입니다. 이러한 필드를 사용하여 이름 지정 체계와 일치하고 IAM 역할에 대해 사전 정의된 권한 경계를 포함할 수 있습니다. 정의되지 않은 경우 설치 프로그램은 새 IAM 역할을 생성합니다.	유효한 AWS IAM 역할의 이름입니다.
compute.platform.aws.rootVolume.iops	루트 볼륨용으로 예약된 IOPS(초당 입/출력 작업)입니다.	정수(예: **4000**)
compute.platform.aws.rootVolume.size	루트 볼륨의 크기(GiB)입니다.	정수(예: **500**)
compute.platform.aws.rootVolume.type	루트 볼륨의 유형입니다.	유효한 AWS EBS 볼륨 유형 (예: **io1**)

매개변수	설명	값
compute.platform.aws.rootVolume.kmsKeyARN	KMS 키의 Amazon 리소스 이름(키 ARN)입니다. 이는 특정 KMS 키를 사용하여 작업자 노드의 OS 볼륨을 암호화하는 데 필요합니다.	유효한 키 ID 또는 키 ARN 입니다.
compute.platform.aws.type	컴퓨팅 시스템의 EC2 인스턴스 유형입니다.	유효한 AWS 인스턴스 유형 (예: **m4.2xlarge**) 다음 **머신의 인스턴스 유형**을 참조하십시오.
compute.platform.aws.zones	설치 프로그램이 컴퓨팅 머신 풀에 필요한 시스템을 생성하는 가용성 영역입니다. 자체 VPC를 제공하는 경우 해당 가용성 영역에 서브넷을 제공해야 합니다.	**us-east-1c**와 같이 YAML 시퀀스로 표시되는 유효한 AWS 가용성 영역의 목록입니다.
compute.aws.region	설치 프로그램이 컴퓨팅 리소스를 생성하는 AWS 리전입니다.	유효한 모든 AWS 리전(예: **us-east-1**)
controlPlane.platform.aws.amiID	클러스터의 컨트롤 플레인 시스템을 시작하는 데 사용되는 AWS AMI입니다. 이는 사용자 지정 RHCOS AMI가 필요한 리전에 필요합니다.	설정된 AWS 리전에 속하는 게시된 또는 사용자 지정 RHCOS AMI입니다.
controlPlane.platform.aws.iamRole	컨트롤 플레인 시스템 풀 인스턴스 프로파일에 적용되는 기존 AWS IAM 역할입니다. 이러한 필드를 사용하여 이름 지정 체계와 일치하고 IAM 역할에 대해 사전 정의된 권한 경계를 포함할 수 있습니다. 정의되지 않은 경우 설치 프로그램은 새 IAM 역할을 생성합니다.	유효한 AWS IAM 역할의 이름입니다.
controlPlane.platform.aws.rootVolume.kmsKeyARN	KMS 키의 Amazon 리소스 이름(키 ARN)입니다. 이는 특정 KMS 키를 사용하여 컨트롤 플레인 노드의 OS 볼륨을 암호화하는 데 필요합니다.	유효한 키 ID 및 키 ARN.
controlPlane.platform.aws.type	컨트롤 플레인 시스템의 EC2 인스턴스 유형입니다.	유효한 AWS 인스턴스 유형 (예: **m5.xlarge**). 다음 **머신의 인스턴스 유형**을 참조하십시오.
controlPlane.platform.aws.zones	설치 프로그램이 컨트롤 플레인 시스템 풀에 필요한 시스템을 생성하는 가용성 영역입니다.	**us-east-1c**와 같이 YAML 시퀀스로 표시되는 유효한 AWS 가용성 영역의 목록입니다.

매개변수	설명	값
controlPlane.aws.region	설치 프로그램이 컨트롤 플레인 리소스를 생성하는 AWS 리전입니다.	유효한 AWS 리전(예: **us-east-1**)
platform.aws.amiID	클러스터의 모든 머신을 부팅하는 데 사용되는 AWS AMI입니다. 설정된 경우 AMI는 클러스터와 동일한 리전에 속해 있어야 합니다. 이는 사용자 지정 RHCOS AMI가 필요한 리전에 필요합니다.	설정된 AWS 리전에 속하는 게시된 또는 사용자 지정 RHCOS AMI입니다.
platform.aws.hostedZone	클러스터의 기존 Route 53 개인 호스팅 영역입니다. 자체 VPC를 제공하는 경우에만 기존 호스팅 영역을 사용할 수 있습니다. 호스팅 영역은 설치 전에 사용자 제공 VPC와 이미 연결되어 있어야 합니다. 또한 호스팅 영역의 도메인은 클러스터 도메인 또는 클러스터 도메인의 상위 도메인이어야 합니다. 정의되지 않은 경우 설치 프로그램은 새 호스팅 영역을 생성합니다.	문자열 (예: **Z3URY6TWQ91KVV**)
platform.aws.serviceEndpoints.name	AWS 서비스 엔드 포인트 이름. 사용자 지정 엔드 포인트는 FIPS와 같은 대체 AWS 엔드 포인트를 사용해야하는 경우에만 필요합니다. EC2, S3, IAM, Elastic Load Balancing, Tagging, Route 53 및 STS AWS 서비스에 대해 사용자 지정 API 엔드 포인트를 지정할 수 있습니다.	유효한 AWS 서비스 엔드 포인트 이름.
platform.aws.serviceEndpoints.url	AWS 서비스 엔드 포인트 URL. URL은 **https** 프로토콜을 사용해야하며 호스트는 인증서를 신뢰해야 합니다.	유효한 AWS 서비스 엔드 포인트 URL.
platform.aws.userTags	설치 프로그램이 생성하는 모든 리소스에 태그로 추가하는 키와 값의 맵입니다.	유효한 YAML 맵(예: **<key>: <value>** 형식의 키 값 쌍) AWS 태그에 대한 자세한 내용은 AWS 문서의 Amazon EC2 리소스 태그 지정을 참조하십시오.

매개변수	설명	값
platform.aws.subnets	설치 프로그램이 VPC를 자동으로 생성하지 않고 VPC를 직접 제공하는 경우 사용할 클러스터의 서브넷을 지정합니다. 서브넷은 사용자가 지정하는 동일한 **machineNetwork[].cidr** 범위의 일부여야 합니다. 표준 클러스터의 경우 각 가용성 영역의 퍼블릭 및 프라이빗 서브넷을 지정합니다. 개인 클러스터의 경우 각 가용성 영역의 프라이빗 서브넷을 지정합니다.	유효한 서브넷 ID.

4.9.7.2. 클러스터 설치를 위한 최소 리소스 요구 사항

각 클러스터 시스템이 다음과 같은 최소 요구사항을 충족해야 합니다.

표 4.31. 최소 리소스 요구사항

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
부트스트랩	RHCOS	4	16GB	100GB	300
컨트롤 플레인	RHCOS	4	16GB	100GB	300
컴퓨팅	RHCOS, RHEL 7.9 또는 RHEL 8.4 [3]	2	8GB	100GB	300

1. SMT(동시 멀티스레딩) 또는 하이퍼 스레딩이 활성화되지 않은 경우 하나의 vCPU는 하나의 물리적 코어와 동일합니다. 활성화하면 다음과 같은 공식을 사용하여 해당 비율을 계산합니다. (코어당 스레드 수 × 코어 수) × 소켓 수 = vCPU 수

2. OpenShift Container Platform 및 Kubernetes는 디스크 성능에 민감하며 특히 10ms p99 fsync 기간이 필요한 컨트롤 플레인 노드의 etcd에 더 빠른 스토리지가 권장됩니다. 많은 클라우드 플랫폼에서 스토리지 크기와 IOPS를 함께 확장되므로 충분한 성능을 얻으려면 스토리지 볼륨을 과도하게 할당해야 할 수 있습니다.

3. 사용자가 프로비저닝한 모든 설치와 마찬가지로 클러스터에서 RHEL 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. RHEL 7 컴퓨팅 머신 사용은 더 이상 사용되지 않으며 향후 OpenShift Container Platform 4 릴리스에서 제거될 예정입니다.

4.9.7.3. 지원되는 AWS 머신 유형

다음 AWS(Amazon Web Services) 인스턴스 유형은 OpenShift Container Platform에서 지원됩니다.

예 4.19. 시스템의 인스턴스 유형

인스턴스 유형	부트스트랩	컨트롤 플레인	컴퓨팅
i3.large	x		
m4.large			x
m4.xlarge		x	x
m4.2xlarge		x	x
m4.4xlarge		x	x
m4.10xlarge		x	x
m4.16xlarge		x	x
m5.large			x
m5.xlarge		x	x
m5.2xlarge		x	x
m5.4xlarge		x	x
m5.8xlarge		x	x
m5.12xlarge		x	x
m5.16xlarge		x	x
m5a.large			x
m5a.xlarge		x	x
m5a.2xlarge		x	x
m5a.4xlarge		x	x
m5a.8xlarge		x	x
m5a.12xlarge		x	x
m5a.16xlarge		x	x
m6i.xlarge		x	x

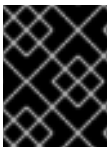
인스턴스 유형	부트스트랩	컨트롤 플레인	컴퓨팅
m6i.2xlarge		x	x
m6i.4xlarge		x	x
m6i.8xlarge		x	x
m6i.16xlarge		x	x
c4.2xlarge		x	x
c4.4xlarge		x	x
c4.8xlarge		x	x
c5.xlarge			x
c5.2xlarge		x	x
c5.4xlarge		x	x
c5.9xlarge		x	x
c5.12xlarge		x	x
c5.18xlarge		x	x
c5.24xlarge		x	x
c5a.xlarge			x
c5a.2xlarge		x	x
c5a.4xlarge		x	x
c5a.8xlarge		x	x
c5a.12xlarge		x	x
c5a.16xlarge		x	x
c5a.24xlarge		x	x
r4.large			x
r4.xlarge		x	x

인스턴스 유형	부트스트랩	컨트롤 플레인	컴퓨팅
r4.2xlarge		x	x
r4.4xlarge		x	x
r4.8xlarge		x	x
r4.16xlarge		x	x
r5.large			x
r5.xlarge		x	x
r5.2xlarge		x	x
r5.4xlarge		x	x
r5.8xlarge		x	x
r5.12xlarge		x	x
r5.16xlarge		x	x
r5.24xlarge		x	x
r5a.large			x
r5a.xlarge		x	x
r5a.2xlarge		x	x
r5a.4xlarge		x	x
r5a.8xlarge		x	x
r5a.12xlarge		x	x
r5a.16xlarge		x	x
r5a.24xlarge		x	x
t3.large			x
t3.xlarge			x
t3.2xlarge			x

인스턴스 유형	부트스트랩	컨트롤 플레인	컴퓨팅
t3a.large			x
t3a.xlarge			x
t3a.2xlarge			x

4.9.7.4. AWS용 샘플 사용자 지정 install-config.yaml 파일

설치 구성 파일(**install-config.yaml**)을 사용자 지정하여 OpenShift Container Platform 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.



중요

이 샘플 YAML 파일은 참조용으로만 제공됩니다. 설치 프로그램을 사용하여 **install-config.yaml** 파일을 받아서 수정해야 합니다.

```
apiVersion: v1
baseDomain: example.com (1)
credentialsMode: Mint (2)
controlPlane: (3) (4)
  hyperthreading: Enabled (5)
  name: master
  platform:
    aws:
      zones:
      - us-west-2a
      - us-west-2b
      rootVolume:
        iops: 4000
        size: 500
        type: io1 (6)
      type: m5.xlarge
  replicas: 3
compute: (7)
- hyperthreading: Enabled (8)
  name: worker
  platform:
    aws:
      rootVolume:
        iops: 2000
        size: 500
        type: io1 (9)
      type: c5.4xlarge
      zones:
      - us-west-2c
  replicas: 3
metadata:
```

```
  name: test-cluster 10
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  aws:
    region: us-west-2 11
    userTags:
      adminContact: jdoe
      costCenter: 7536
    subnets: 12
    - subnet-1
    - subnet-2
    - subnet-3
    amiID: ami-96c6f8f7 13
    serviceEndpoints: 14
      - name: ec2
        url: https://vpce-id.ec2.us-west-2.vpce.amazonaws.com
    hostedZone: Z3URY6TWQ91KVV 15
fips: false 16
sshKey: ssh-ed25519 AAAA... 17
publish: Internal 18
pullSecret: '{"auths": ...}' 19
```

1 10 11 19 필수 항목입니다. 설치 프로그램에서 이 값을 입력하라는 메시지를 표시합니다.

2 옵션: 이 매개 변수를 추가하여 Cloud Credential Operator (CCO)가 인증 정보의 기능을 동적으로 판별하도록 하지 않고 CCO가 지정된 모드를 사용하도록 강제합니다. CCO 모드에 대한 자세한 내용은 *Platform Operators 참조* 콘텐츠의 *Cloud Credential Operator* 항목을 참조하십시오.

3 7 이러한 매개변수와 값을 지정하지 않으면 설치 프로그램은 기본값을 적용합니다.

4 **controlPlane** 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

5 8 동시 멀티스레딩 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. 일부 클러스터 시스템에서 동시 멀티스레딩을 비활성화할 경우에는 해당 멀티스레딩을 모든 클러스터 시스템에서 비활성화해야 합니다.

중요

동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다. 동시 멀티스레딩을 비활성화하는 경우 시스템에 더 큰 인스턴스 유형(예: **m4.2xlarge** 또는 **m5.2xlarge**)을 사용합니다.

6 9 대규모의 클러스터의 경우 고속 etcd 스토리지를 구성하려면 스토리지 유형을 **IO1**로 설정하고 **iops**는 **2000**으로 설정합니다.

12 자체 VPC를 제공하는 경우 클러스터가 사용하는 각 가용성 영역의 서브넷을 지정합니다.

13 클러스터 머신을 시작하는 데 사용되는 AMI의 ID입니다. 설정된 경우 AMI는 클러스터와 동일한 리전에 속해 있어야 합니다.

14 AWS 서비스 엔드 포인트입니다. 알 수 없는 AWS 리전에 설치할 때 사용자 지정 엔드 포인트가 필요합니다. 엔드포인트 URL은 **https** 프로토콜을 사용해야하며 호스트는 인증서를 신뢰해야 합니다.

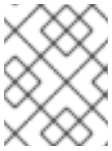
15 기존 Route 53 개인 호스팅 영역의 ID입니다. 기존 호스팅 영역을 제공하려면 클러스터를 설치하기 전에 자체 VPC를 제공하고 호스팅 영역이 VPC와 연결되어 있어야 합니다. 정의되지 않은 경우 설치 프로그램은 새 호스팅 영역을 생성합니다.

16 FIPS 모드 활성화 또는 비활성화 여부입니다. 기본적으로 FIPS 모드는 비활성화됩니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.

중요

FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.

17 선택사항으로, 클러스터의 시스템에 액세스하는 데 사용할 **sshKey** 값을 제공할 수도 있습니다.



참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

18 클러스터의 사용자 끝점을 게시하는 방법. 인터넷에서 액세스할 수 없는 프라이빗 클러스터를 배포하려면 **publish**를 **Internal**로 설정합니다. 기본값은 **External**입니다.

4.9.7.5. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 HTTP 또는 HTTPS 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 OpenShift Container Platform 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 API에 대한 호출을 포함하여 모든 클러스터 발신(Egress) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.

참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 Red Hat OpenStack Platform (RHOSP)에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

- **ec2.<region>.amazonaws.com,elasticloadbalancing.<region>.amazonaws.com** 및 **s3.<region>.amazonaws.com** 끝점을 VPC 끝점에 추가했습니다. 이러한 끝점은 노드에서 AWS EC2 API로 요청을 완료하는 데 필요합니다. 프록시는 노드 수준이 아닌 컨테이너 수준에서 작동하므로 이러한 요청을 AWS 사설 네트워크를 통해 AWS EC2 API로 라우팅해야 합니다. 프록시 서버의 허용 목록에 EC2 API의 공용 IP 주소를 추가하는 것만으로는 충분하지 않습니다.

절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

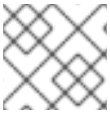
1. 클러스터 외부에서 HTTP 연결을 구축하는 데 사용할 프록시 URL입니다. URL 스키마는 **http**여야 합니다.
2. 클러스터 외부에서 HTTPS 연결을 구축하는 데 사용할 프록시 URL입니다.
3. 대상 도메인 이름, IP 주소 또는 프록시에서 제외할 기타 네트워크 CIDR로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. *를 사용하여 모든 대상에 대해 프록시를 바이패스합니다.
4. 이 값을 제공하면 설치 프로그램에서 추가 CA 인증서를 보유할 **openshift -config 네임스페이스에 user-ca- bundle** 이라는 구성 맵을 생성합니다. **additionalTrustBundle** 및 하나 이상의 프록시 설정을 제공하는 경우 **프록시** 오브젝트는 **trustedCA** 필드의 **user-ca-bundle** 구성 맵을 참조하도록 구성됩니다. 그러면 Cluster Network Operator에서 **trustedCA** 매개변수에 대해 지정된 콘텐츠를 RHCOS 신뢰 번들과 병합하는 **trusted-ca-bundle** 구성 맵을 생성합니다. 프록시의 ID 인증서를 RHCOS 트러스트 번들에 있는 기관에서 서명하지 않은 경우 **additionalTrustBundle** 필드가 있어야 합니다.

참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 OpenShift Container Platform을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 nil이 됩니다.



참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

4.9.8. 클러스터 배포

호환되는 클라우드 플랫폼에 OpenShift Container Platform을 설치할 수 있습니다.

중요

최초 설치 과정에서 설치 프로그램의 **create cluster** 명령을 한 번만 실행할 수 있습니다.

사전 요구 사항

- 클러스터를 호스팅하는 클라우드 플랫폼으로 계정을 구성합니다.
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

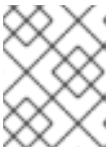
프로세스

1. 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터 배포를 초기화합니다.

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>**는 다음을 지정합니다.

2 다른 설치 세부 사항을 보려면 **info** 대신 **warn**, **debug** 또는 **error**를 지정합니다.



참고

호스트에 구성된 클라우드 공급자 계정에 클러스터를 배포하기에 충분한 권한이 없는 경우, 설치 프로세스가 중단되고 누락된 권한을 알리는 메시지가 표시됩니다.

클러스터 배포가 완료되면 웹 콘솔로 연결되는 링크와 **kubeadmin** 사용자의 인증 정보가 포함된 클러스터 액세스 지침이 사용자 터미널에 표시됩니다.

출력 예

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
```

```
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-Wt5AL"
INFO Time elapsed: 36m22s
```

참고

설치에 성공하면 클러스터 액세스 및 인증 정보도 **<installation_directory>/.openshift_install.log**로 출력됩니다.

중요

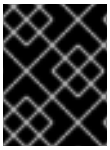
- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

중요

설치 프로그램에서 생성되는 파일이나 설치 프로그램을 삭제해서는 안 됩니다. 클러스터를 삭제하려면 두 가지가 모두 필요합니다.

4.9.9. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift CLI(**oc**)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 **oc**를 설치할 수 있습니다.



중요

이전 버전의 **oc**를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 **oc**를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 Linux에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Linux Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. **oc** 바이너리를 **PATH**에 있는 디렉터리에 배치합니다.
PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치
다음 절차에 따라 Windows에 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Windows Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. ZIP 프로그램으로 아카이브의 압축을 풉니다.
5. **oc** 바이너리를 **PATH**에 있는 디렉터리로 이동합니다.
PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치
다음 절차에 따라 macOS에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 MacOSX Client** 항목 옆에 있는 **지금 다운로드를** 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 해제하고 압축을 풉니다.
5. **oc** 바이너리 PATH의 디렉터리로 이동합니다.
PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

4.9.10. CLI를 사용하여 클러스터에 로그인

클러스터 **kubeconfig** 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. **kubeconfig** 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- **oc** CLI를 설치했습니다.

프로세스

1. **kubeadmin** 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 **oc** 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

4.9.11. 웹 콘솔을 사용하여 클러스터에 로그인

kubeadmin 사용자는 OpenShift Container Platform 설치 후 기본적으로 존재합니다. OpenShift Container Platform 웹 콘솔을 사용하여 **kubeadmin** 사용자로 클러스터에 로그인할 수 있습니다.

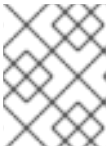
사전 요구 사항

- 설치 호스트에 대한 액세스 권한이 있어야 합니다.
- 클러스터 설치를 완료했으며 모든 클러스터 Operator를 사용할 수 있습니다.

프로세스

1. 설치 호스트의 **kubeadmin-password** 파일에서 **kubeadmin** 사용자의 암호를 가져옵니다.

```
$ cat <installation_directory>/auth/kubeadmin-password
```



참고

대안으로 설치 호스트의 **<installation_directory>/.openshift_install.log** 로그 파일에서 **kubeadmin** 암호를 가져올 수 있습니다.

2. OpenShift Container Platform 웹 콘솔 경로를 나열합니다.

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```

참고

대안으로 설치 호스트의 **<installation_directory>/.openshift_install.log** 로그 파일에서 OpenShift Container Platform 경로를 가져올 수 있습니다.

출력 예

```
console     console-openshift-console.apps.<cluster_name>.<base_domain>          console
https   reencrypt/Redirect   None
```

3. 웹 브라우저의 이전 명령 출력에 자세히 설명된 경로로 이동하고 **kubeadmin** 사용자로 로그인합니다.

추가 리소스

- OpenShift Container Platform 웹 콘솔 액세스 및 이해에 대한 자세한 내용은 웹 콘솔에 액세스를 참조하십시오.

4.9.12. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

4.9.13. 다음 단계

- 설치를 확인합니다.
- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.
- 필요한 경우 클라우드 공급자 인증 정보를 제거 할 수 있습니다.

4.10. 정부 리전 또는 시크릿 리전에 AWS의 클러스터 설치

OpenShift Container Platform 4.9 버전에서는 정부 리전 또는 시크릿 리전에 AWS(Amazon Web Services)의 클러스터를 설치할 수 있습니다. 리전을 구성하려면 클러스터를 설치하기 전에 **install-config.yaml** 파일에서 매개변수를 수정합니다.

4.10.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비 에 대한 문서를 읽습니다.
- 클러스터를 호스팅할 AWS 계정을 구성 했습니다.

중요

컴퓨터에 AWS 프로필이 저장되어 있는 경우 다단계 인증 장치를 사용하는 동안 생성한 임시 세션 토큰을 해당 프로필이 사용해서는 안 됩니다. 클러스터는 클러스터의 전체 수명 동안 현재 AWS 자격 증명을 계속 사용하여 AWS 리소스를 생성하므로 장기 자격 증명을 사용해야 합니다. 적절한 키를 생성하려면 AWS 문서의 IAM 사용자의 액세스 키 관리를 참조하십시오. 설치 프로그램을 실행할 때 키를 제공할 수 있습니다.

- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성 해야 합니다.
- 사용자의 환경에서 클라우드 ID 및 액세스 관리(IAM) API에 액세스할 수 없거나 **kube-system** 네임스페이스에 관리자 수준의 인증 정보 시크릿을 저장하지 않으려면 IAM 인증 정보 수동 생성 및 유지 관리를 참조하십시오.

4.10.2. AWS 정부 및 시크릿 리전

OpenShift Container Platform은 AWS GovCloud (US) 리전 및 AWS Commercial Cloud Services (C2S) Top Secret Region 에 클러스터 배포를 지원합니다. 이러한 리전은 연방, 주, 지역 수준의 미국 정부 기관은 물론 클라우드에서 중요한 워크로드를 실행해야 하는 미국 정부 기관, 계약자, 교육 기관 및 기타 미국 고객을 위해 설계되었습니다.

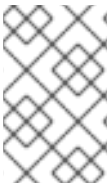
이러한 리전에는 선택할 수 있는Red Hat Enterprise Linux CoreOS (RHCOS) Amazon 머신 이미지(AMI)가 게시되지 않았으므로 해당 리전에 속하는 사용자 지정 AMI를 업로드해야 합니다.

다음 AWS GovCloud 파티션이 지원됩니다.

- **us-gov-west-1**
- **us-gov-east-1**

다음 AWS 최상위 시크릿 리전 파티션이 지원됩니다.

- **us-iso-east-1**



참고

AWS 최상위 시크릿 리전에서 지원되는 최대 MTU는 AWS 상용과 동일하지 않습니다. 설치 중 MTU 구성에 대한 자세한 내용은 *네트워크 사용자 지정을 사용하여 AWS에 클러스터 설치* 의 *Cluster Network Operator 구성 오브젝트* 섹션을 참조하십시오.

4.10.3. 설치 요구 사항

Red Hat은 AWS 정부 또는 시크릿 리전의 RHCOS(Red Hat Enterprise Linux CoreOS) Amzaon 머신 이미지를 게시하지 않습니다.

클러스터를 설치하려면 먼저 다음을 수행해야 합니다.

- 사용자 지정 RHCOS AMI를 업로드합니다.
- 수동으로 설치 구성 파일 (**install-config.yaml**)을 생성합니다.
- 설치 구성 파일에서 AWS 리전 및 관련 사용자 지정 AMI를 지정합니다.

OpenShift Container Platform 설치 프로그램을 사용하여 설치 구성 파일을 생성할 수 없습니다. 설치 프로그램에서 RHCOS AMI에 대한 기본 지원 없이 AWS 리전을 나열하지 않습니다.

중요

C2S Top Secret Region에 배포하는 경우 AWS API에 사용자 정의 CA 신뢰 번들이 필요하므로 **install-config.yaml** 파일의 **additionalTrustBundle** 필드에 사용자 정의 CA 인증서도 정의해야 합니다. 설치 프로그램이 AWS API에 액세스할 수 있도록 하려면 설치 프로그램을 실행하는 머신에 CA 인증서를 정의해야 합니다. 머신의 신뢰 저장소에 CA 번들을 추가하고 **AWS_CA_BUNDLE** 환경 변수를 사용하거나 AWS 구성 파일의 **ca_bundle** 필드에 CA 번들을 정의해야 합니다.

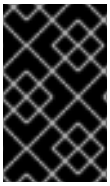
4.10.4. 프라이빗 클러스터

외부 엔드 포인트를 노출하지 않는 비공개 OpenShift Container Platform 클러스터를 배포할 수 있습니다. 프라이빗 클러스터는 내부 네트워크에서만 액세스할 수 있으며 인터넷에 표시되지 않습니다.

참고

퍼블릭 영역은 AWS GovCloud 또는 Top Secret Region의 Route 53에서 지원되지 않습니다. 따라서 클러스터가 AWS 정부 또는 시크릿 리전에 배포된 경우 프라이빗으로 설정되어야 합니다.

기본적으로 OpenShift Container Platform은 공개적으로 액세스 가능한 DNS 및 끝점을 사용하여 프로비저닝됩니다. 따라서 개인 클러스터를 배포할 때 클러스터에서 DNS, Ingress Controller 및 API 서버를 비공개로 설정할 수 있습니다. 즉 클러스터 리소스는 내부 네트워크에서만 액세스할 수 있고 인터넷에는 노출되지 않습니다.



중요

클러스터에 퍼블릭 서브넷이 있는 경우 관리자가 생성한 로드 밸런서 서비스에 공개적으로 액세스할 수 있습니다. 클러스터 보안을 보장하기 위해 이러한 서비스에 명시적으로 주석이 지정되었는지 확인합니다.

프라이빗 클러스터를 배포하려면 다음을 수행해야 합니다.

- 요구 사항을 충족하는 기존 네트워킹을 사용합니다. 네트워크의 다른 클러스터 사이에 클러스터 리소스를 공유할 수 있습니다.
- 다음에 액세스할 수 있는 머신에서 배포합니다.
  - 프로비저닝하는 클라우드용 API 서비스
  - 프로비저닝하는 네트워크의 호스트
  - 설치 미디어를 가져올 인터넷

이러한 액세스 요구사항을 충족하고 회사의 지침을 따르는 모든 시스템을 사용할 수 있습니다. 클라우드 네트워크의 배스천 호스트 또는 VPN을 통해 네트워크에 액세스할 수 있는 시스템 등을 예로 들 수 있습니다.

4.10.4.1. AWS의 개인 클러스터

AWS(Amazon Web Services)에 개인 클러스터를 생성하려면 클러스터를 호스팅할 기존 프라이빗 VPC와 서브넷을 제공해야 합니다. 또한 설치 프로그램에서 클러스터에 필요한 DNS 레코드를 확인할 수 있어야 합니다. 설치 프로그램은 개인 네트워크에서만 액세스할 수 있도록 Ingress Operator 및 API 서버를 구성합니다.

클러스터가 AWS API에 액세스하려면 여전히 인터넷 접속이 필요합니다.

다음은 프라이빗 클러스터를 설치할 때 필요하지 않거나 생성되지 않는 항목들입니다.

- 퍼블릭 서브넷
- 공용 인그레스를 지원하는 공용 로드 밸런서
- 클러스터의 **baseDomain**과 일치하는 공용 Route 53 영역

설치 프로그램은 사용자가 지정하는 **baseDomain**을 사용하여 프라이빗 Route 53 영역과 클러스터에 필요한 레코드를 생성합니다. Operator가 클러스터에 대한 공용 레코드를 생성하지 않고 사용자가 지정하는 프라이빗 서브넷에 모든 클러스터 시스템이 배치되도록 클러스터가 구성됩니다.

4.10.4.1.1. 제한

프라이빗 클러스터에 공용 기능을 추가하는 기능은 제한됩니다.

- 설치 후에는 VPC에서 사용 중인 각 가용성 영역의 퍼블릭 서브넷 생성, 공용 로드 밸런서 생성, 6443(Kubernetes API 포트)의 인터넷 트래픽을 허용하도록 컨트롤 플레인 보안 그룹 구성 등 추가 조치 없이 Kubernetes API 엔드포인트를 공개할 수 없습니다.
- 공용 서비스 유형 로드 밸런서를 사용하는 경우 AWS가 공용 로드 밸런서를 생성하는 데 사용할 수 있도록 각 가용성 영역의 퍼블릭 서브넷에 **kubernetes.io/cluster/<cluster-infra-id>: shared** 태그를 지정해야 합니다.

4.10.5. 사용자 지정 VPC 사용 정보

OpenShift Container Platform 4.9에서는 AWS(Amazon Web Services)의 기존 Amazon VPC(Virtual Private Cloud)에 있는 기존 서브넷에 클러스터를 배포할 수 있습니다. 기존 GCP VPC에 OpenShift Container Platform을 배포하면 새 계정의 한도 제한 적용을 받지 않거나 회사의 지침에 따른 운영 제한을 보다 쉽게 준수할 수 있습니다. VPC를 직접 생성하는 데 필요한 인프라 생성 권한을 받을 수 없는 경우 이 설치 옵션을 사용합니다.

설치 프로그램은 기존 서브넷에 있는 다른 구성 요소를 알 수 없으므로 사용자를 대신하여 서브넷 CIDR 등을 선택할 수 없습니다. 클러스터를 직접 설치하는 서브넷에 대한 네트워킹을 구성해야 합니다.

4.10.5.1. VPC 사용 요구사항

설치 프로그램은 더 이상 다음 구성 요소를 생성하지 않습니다.

- 인터넷 게이트웨이
- NAT 게이트웨이

- 서브넷
- 라우팅 테이블
- VPC
- VPC DHCP 옵션
- VPC 끝점

참고

설치 프로그램을 사용하려면 클라우드 제공 DNS 서버를 사용해야 합니다. 사용자 지정 DNS 서버 사용은 지원되지 않으며 이로 인해 설치에 실패합니다.

사용자 지정 VPC를 사용하는 경우, 사용할 클러스터와 설치 프로그램에 맞게 VPC와 해당 서브넷을 구성해야 합니다. AWS VPC 생성 및 관리에 대한 자세한 내용은 AWS 문서의 Amazon VPC 콘솔 마법사 구성 및 VPC 및 서브넷 작업을 참조하십시오.

설치 프로그램은 다음을 수행할 수 없습니다.

- 클러스터에서 사용할 네트워크 범위를 세분합니다.
- 서브넷의 라우팅 테이블을 설정합니다.
- DHCP와 같은 VPC 옵션을 설정합니다.

클러스터를 설치하기 전에 이러한 작업을 완료해야 합니다. AWS VPC의 네트워킹 구성에 대한 자세한 내용은 VPC 네트워킹 구성 요소 및 경로 테이블을 참조하십시오.

VPC는 다음 특성을 충족해야 합니다.

- VPC는 **kubernetes.io/cluster/.*: owned** 태그를 사용해서는 안 됩니다.
  설치 프로그램은 **kubernetes.io/cluster/.*: shared** 태그를 추가하도록 서브넷을 수정하므로 서브넷에 사용 가능한 여유 태그 슬롯이 하나 이상 있어야 합니다. AWS 문서의 태그 제한 사항을 참조하여 설치 프로그램이 사용자가 지정하는 각 서브넷에 태그를 추가할 수 있는지 확인합니다.

- 클러스터가 VPC에 연결된 Route 53 영역을 사용하여 클러스터의 내부 DNS 레코드를 확인할 수 있도록 VPC에서 **enableDnsSupport 및 enableDnsHostnames** 속성을 활성화해야 합니다.
  AWS 문서에서 VPC의 DNS 지원을 참조하십시오.
  자체 Route 53 호스팅 프라이빗 영역을 사용하려면 클러스터를 설치하기 전에 기존 호스팅 영역을 VPC와 연결해야 합니다. **install-config.yaml** 파일에서 **platform.aws.hostedZone** 필드를 사용하여 호스팅 영역을 정의할 수 있습니다.

- 공용 액세스 권한이 있는 클러스터를 사용하는 경우 클러스터가 사용하는 각 가용성 영역의 퍼블릭 및 프라이빗 서브넷을 만들어야 합니다. 각 가용성 영역에는 퍼블릭 서브넷과 프라이빗 서브넷이 1개 이상 포함될 수 있습니다.

연결이 끊긴 환경에서 작업 중인 경우에는 EC2 및 ELB 끝점의 공용 IP 주소에 도달할 수 없습니다. 이 문제를 해결하려면 VPC 끝점을 생성하여 클러스터가 사용 중인 서브넷에 연결해야 합니다. 올바른 끝점의 이름은 다음과 같습니다.

정부 리전

- **ec2.<region>.amazonaws.com**

- **elasticloadbalancing.<region>.amazonaws.com**
- **s3.<region>.amazonaws.com**

보안 최상위 리전

- **ec2.<region>.c2s.ic.gov**
- **elasticloadbalancing.<region>.c2s.ic.gov**
- **s3.<region>.c2s.ic.gov**

필수 VPC 구성 요소

시스템과의 통신을 허용하는 서브넷과 적합한 VPC를 제공해야 합니다.

구성 요소	AWS 유형	설명
VPC	• **AWS::EC2::VPC** • **AWS::EC2::VPCEndpoint**	클러스터에서 사용할 공용 VPC를 제공해야 합니다. VPC는 각 서브넷의 라우팅 테이블을 참조하는 끝점을 사용하여 S3에서 호스팅되는 레지스트리와의 통신을 개선합니다.
퍼블릭 서브넷	• **AWS::EC2::Subnet** • **AWS::EC2::SubnetNetworkAclAssociation**	VPC에는 1~3개의 가용성 영역에 대한 퍼블릭 서브넷이 있어야 하며 이 서브넷을 적절한 인그레스 규칙과 연결해야 합니다.
인터넷 게이트웨이	• **AWS::EC2::InternetGateway** • **AWS::EC2::VPCGatewayAttachment** • **AWS::EC2::RouteTable** • **AWS::EC2::Route** • **AWS::EC2::SubnetRouteTableAssociation** • **AWS::EC2::NatGateway** • **AWS::EC2::EIP**	공용 경로가 있는 공용 인터넷 게이트웨이가 VPC에 연결되어 있어야 합니다. 제공된 템플릿에서 각 퍼블릭 서브넷에는 EIP 주소를 갖는 NAT 게이트웨이가 있습니다. 이러한 NAT 게이트웨이를 사용하면 프라이빗 서브넷 인스턴스와 같은 클러스터 리소스가 인터넷에 도달할 수 있으므로 일부 제한된 네트워크 또는 프록시 시나리오에는 필요하지 않습니다.
네트워크 액세스 제어	• **AWS::EC2::NetworkAcl** • **AWS::EC2::NetworkAclEntry**	VPC가 다음 포트에 액세스할 수 있어야 합니다. **포트** / **이유** **80** / 인바운드 HTTP 트래픽

구성 요소	AWS 유형	설명	
		443	인바운드 HTTPS 트래픽
		22	인바운드 SSH 트래픽
		1024 - 65535	인바운드 임시 트래픽
		0 - 65535	아웃바운드 임시 트래픽
프라이빗 서브넷	• **AWS::EC2::Subnet** • **AWS::EC2::RouteTable** • **AWS::EC2::SubnetRouteTableAssociation**	VPC에 프라이빗 서브넷이 포함될 수 있습니다. 제공된 CloudFormation 템플릿은 1~3개 가용성 영역의 프라이빗 서브넷을 생성할 수 있습니다. 프라이빗 서브넷을 사용하는 경우 적절한 경로와 테이블을 제공해야 합니다.	

4.10.5.2. VPC 검증

제공한 서브넷이 적합한지 확인하기 위해 설치 프로그램이 다음 데이터를 확인합니다.

- 사용자가 지정하는 모든 서브넷이 있는지 여부.
- 사용자가 프라이빗 서브넷을 제공합니다.
- 서브넷 CIDR이 사용자가 지정한 시스템 CIDR에 속합니다.
- 각 가용성 영역에 서브넷을 제공합니다. 각 가용성 영역에는 퍼블릭 서브넷과 프라이빗 서브넷이 하나씩 포함됩니다. 개인 클러스터를 사용하는 경우 각 가용성 영역에 프라이빗 서브넷만 제공합니다. 그렇지 않으면 각 가용성 영역에 퍼블릭 서브넷과 프라이빗 서브넷을 하나씩만 제공합니다.
- 각 프라이빗 서브넷 가용성 영역에 퍼블릭 서브넷을 제공합니다. 프라이빗 서브넷을 제공하지 않는 가용성 영역에서는 시스템이 프로비저닝되지 않습니다.

기존 VPC를 사용하는 클러스터를 제거해도 VPC는 삭제되지 않습니다. VPC에서 OpenShift Container Platform 클러스터를 제거하면 클러스터가 사용한 서브넷에서 **kubernetes.io/cluster/.*: shared** 태그가 제거됩니다.

4.10.5.3. 권한 분할

OpenShift Container Platform 4.3부터 클러스터를 배포하는 데 설치 프로그램에서 프로비저닝한 인프라 클러스터에 필요한 권한 중 일부가 필요하지 않게 되었습니다. 이 변경 사항은 회사의 권한 분류와 유사합니다. 즉 일부 개인의 경우 클라우드에서 다른 사람들과 다른 리소스를 생성할 수 있습니다. 예를 들어 인스턴스, 버킷, 로드 밸런서와 같은 애플리케이션 관련 항목은 생성할 수 있지만 VPC 서브넷 또는 인그레스 규칙과 같은 네트워킹 관련 구성 요소는 생성하지 못할 수 있습니다.

클러스터를 생성할 때 사용하는 GCP 자격 증명에는 서브넷, 라우팅 테이블, 인터넷 게이트웨이, NAT, VPN과 같은 VPC 내 핵심 네트워킹 구성 요소와 VPC를 생성하는 데 필요한 네트워킹 권한이 필요하지 않

습니다. 하지만 ELB, 보안 그룹, S3 버킷 및 노드와 같이 클러스터 내 시스템에 필요한 애플리케이션 리소스를 생성하려면 여전히 권한이 필요합니다.

4.10.5.4. 클러스터 간 격리

OpenShift Container Platform을 기존 네트워크에 배포하면 클러스터 서비스 격리가 다음과 같은 방식으로 감소합니다.

- 동일한 VPC에 여러 OpenShift Container Platform 클러스터를 설치할 수 있습니다.
- ICMP 인그레스가 전체 네트워크에서 허용됩니다.
- TCP 22 인그레스(SSH)가 전체 네트워크에 허용됩니다.
- 컨트롤 플레인 TCP 6443 인그레스(Kubernetes API)가 전체 네트워크에 허용됩니다.
- 컨트롤 플레인 TCP 22623 인그레스(MCS)가 전체 네트워크에 허용됩니다.

4.10.6. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.

중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

4.10.7. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 **core** 사용자의 **~/.ssh/authorized_keys** 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 **core**로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. **./openshift-install gather** 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

참고

AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: **~/.ssh/id_ed25519**)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 **'~/.ssh** 디렉터리에 있는지 확인하십시오.

참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 **ed25519** 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 **rsa** 또는 **ecdsa** 알고리즘을 사용하는 키를 생성합니다.

2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 **~/.ssh/id_ed25519.pub** 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 **./openshift-install gather** 명령을 사용하려는 경우 필요합니다.

참고

일부 배포에서는 **~/.ssh/id_rsa** 및 **~/.ssh/id_dsa**와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. **ssh-agent** 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. **ssh-agent**에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: **~/.ssh/id_ed25519**).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

4.10.8. AWS에서 사용자 지정 RHCOS AMI 업로드

사용자 지정 Amazon Web Services (AWS) 리전에 배포하는 경우 해당 리전에 속하는 사용자 지정 Red Hat Enterprise Linux CoreOS (RHCOS) Amazon 머신 이미지 (AMI)를 업로드해야 합니다.

사전 요구 사항

- AWS 계정을 구성했습니다.
- 필요한 IAM 서비스 역할로 Amazon S3 버킷을 생성했습니다.
- RHCOS VMDK 파일을 Amazon S3에 업로드했습니다. RHCOS VMDK 파일은 설치하는 OpenShift Container Platform 버전과 같거나 그 이하의 버전이어야 합니다.
- AWS CLI를 다운로드하여 컴퓨터에 설치했습니다. Install the AWS CLI Using the Bundled Installer를 참조하십시오.

프로세스

1. AWS 프로필을 환경 변수로 내보냅니다.

```
$ export AWS_PROFILE=<aws_profile> 1
```

1 **govcloud**와 같이 AWS 인증 정보를 보유하는 AWS 프로필 이름입니다.

2. 사용자 지정 AMI와 연결할 리전을 환경 변수로 내보냅니다.

```
$ export AWS_DEFAULT_REGION=<aws_region> 1
```

1 **us-gov-east-1**과 같은 AWS 리전.

3. Amazon S3에 업로드한 RHCOS 버전을 환경 변수로 내보냅니다.

```
$ export RHCOS_VERSION=<version> 1
```

1 **4.9.0** 과 같은 RHCOS VMDK 버전입니다.

4. Amazon S3 버킷 이름을 환경 변수로 내보냅니다.

```
$ export VMIMPORT_BUCKET_NAME=<s3_bucket_name>
```

5. **container.json** 파일을 만들고 RHCOS VMDK 파일을 정의합니다.

```
$ cat <<EOF > containers.json
{
  "Description": "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64",
  "Format": "vmdk",
  "UserBucket": {
    "S3Bucket": "${VMIMPORT_BUCKET_NAME}",
    "S3Key": "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64.vmdk"
  }
}
EOF
```

6. RHCOS 디스크를 Amazon EBS 스냅샷으로 가져옵니다.

```
$ aws ec2 import-snapshot --region ${AWS_DEFAULT_REGION} \
    --description "<description>" \ 1
    --disk-container "file://<file_path>/containers.json" 2
```

1 가져온 RHCOS 디스크에 대한 설명 (예: **rhcos-$ {RHCOS_VERSION} -x86_64-aws.x86_64**)입니다.

2 RHCOS 디스크를 설명하는 JSON 파일의 파일 경로입니다. JSON 파일에는 Amazon S3 버킷 이름과 키가 포함되어 있어야합니다.

7. 이미지 가져 오기 상태를 확인합니다.

```
$ watch -n 5 aws ec2 describe-import-snapshot-tasks --region ${AWS_DEFAULT_REGION}
```

출력 예

```
{
    "ImportSnapshotTasks": [
        {
            "Description": "rhcos-4.7.0-x86_64-aws.x86_64",
            "ImportTaskId": "import-snap-fh6i8uil",
            "SnapshotTaskDetail": {
                "Description": "rhcos-4.7.0-x86_64-aws.x86_64",
                "DiskImageSize": 819056640.0,
                "Format": "VMDK",
                "SnapshotId": "snap-06331325870076318",
                "Status": "completed",
```

```
            "UserBucket": {
                "S3Bucket": "external-images",
                "S3Key": "rhcos-4.7.0-x86_64-aws.x86_64.vmdk"
            }
        }
    }
  ]
}
```

SnapshotId를 복사하여 이미지를 등록합니다.

8. RHCOS 스냅 샷에서 사용자 지정 RHCOS AMI를 생성합니다.

```
$ aws ec2 register-image \
  --region ${AWS_DEFAULT_REGION} \
  --architecture x86_64 \ 1
  --description "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64" \ 2
  --ena-support \
  --name "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64" \ 3
  --virtualization-type hvm \
  --root-device-name '/dev/xvda' \
  --block-device-mappings 'DeviceName=/dev/xvda,Ebs=
{DeleteOnTermination=true,SnapshotId=<snapshot_ID>}' 4
```

1. **x86_64**, **s390x** 또는 **ppc64le**과 같은 RHCOS VMDK 아키텍처 유형입니다.
2. 가져온 스냅샷의 **Description**입니다.
3. RHCOS AMI의 이름입니다.
4. 가져온 스냅샷의 **SnapshotID**입니다.

이러한 API에 대한 자세한 내용은 importing snapshots 및 creating EBS-backed AMIs에서 참조하십시오.

4.10.9. AWS Marketplace 이미지 가져오기

AWS Marketplace 이미지를 사용하여 OpenShift Container Platform 클러스터를 배포하는 경우 먼저 AWS를 통해 구독해야 합니다. 이 제공을 구독하면 설치 프로그램이 작업자 노드를 배포하는 데 사용하는 AMI ID를 제공합니다.

참고

AWS Marketplace 이미지를 사용하여 OpenShift Container Platform 클러스터를 배포하는 것은 시크릿 리전 또는 중국 지역에서 지원되지 않습니다.

사전 요구 사항

- 이 제안을 구매할 수 있는 AWS 계정이 있어야 합니다. 이 계정은 클러스터를 설치하는 데 사용되는 계정과 같을 필요는 없습니다.

프로세스

1. AWS Marketplace 에서 OpenShift Container Platform 서브스크립션을 완료합니다.

2. 특정 지역에 대한 AMI ID를 기록합니다. 설치 프로세스의 일부로 클러스터를 배포하기 전에 **install-config.yaml** 파일을 이 값으로 업데이트해야 합니다.

AWS Marketplace 작업자 노드가 있는 샘플 install-config.yaml 파일

```
apiVersion: v1
baseDomain: example.com
compute:
- hyperthreading: Enabled
  name: worker
  platform:
    aws:
      amiID: ami-06c4d345f7c207239 (1)
      type: m5.4xlarge
  replicas: 3
metadata:
  name: test-cluster
platform:
  aws:
    region: us-gov-west-1 (2)
sshKey: ssh-ed25519 AAAA...
pullSecret: '{"auths": ...}'
```

(1) AWS Marketplace 서브스크립션의 AMI ID입니다.

(2) AMI ID는 특정 AWS 리전과 연결되어 있습니다. 설치 구성 파일을 생성할 때 서브스크립션을 구성할 때 지정한 것과 동일한 AWS 리전을 선택해야 합니다.

4.10.10. 설치 프로그램 받기

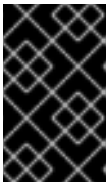
OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

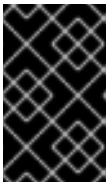
프로세스

1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.
2. 인프라 공급자를 선택합니다.
3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.



중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.



중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 OpenShift Container Platform 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

4.10.11. 수동으로 설치 구성 파일 만들기

사용자 지정 Red Hat Enterprise Linux CoreOS (RHCOS) AMI가 필요한 리전에 Amazon Web Services (AWS)에서 OpenShift Container Platform을 설치할 때 설치 구성 파일을 수동으로 생성해야 합니다.

사전 요구 사항

- 사용자 정의 RHCOS AMI를 업로드했습니다.
- 로컬 시스템에 설치 프로그램에 제공할 SSH 공개 키가 있습니다. 키는 디버깅 및 재해 복구를 위해 클러스터 노드에 대한 SSH 인증에 사용됩니다.
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿이 있습니다.

프로세스

1. 필요한 설치 자산을 저장할 설치 디렉터리를 만듭니다.

```
$ mkdir <installation_directory>
```

중요

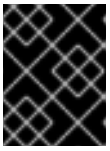
디렉터리를 만들어야 합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

2. 샘플 **install-config.yaml** 파일 템플릿을 사용자 지정하여 **<installation_directory>**에 저장합니다.

참고

이 설정 파일의 이름을 **install-config.yaml**로 지정해야 합니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.



중요

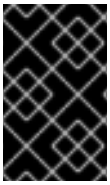
install-config.yaml 파일은 설치 과정의 다음 단계에서 사용됩니다. 이 시점에서 이를 백업해야 합니다.

4.10.11.1. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 매개변수 값을 제공하여 클러스터를 호스팅할 클라우드 플랫폼에서 사용자 계정을 설명하고 선택사항으로 클러스터의 플랫폼을 사용자 지정합니다. **install-config.yaml** 설치 구성 파일을 생성할 때 명령줄을 통해 필요한 매개변수 값을 제공합니다. 클러스터를 사용자 지정하면 **install-config.yaml** 파일을 수정하여 플랫폼에 대한 세부 정보를 제공할 수 있습니다.

참고

설치한 후에는 **install-config.yaml** 파일에서 이러한 매개변수를 수정할 수 없습니다.



중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

4.10.11.1.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 4.32. 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체

매개변수	설명	값
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	소문자, 하이픈(**-**), 마침표(**.**)로 구성되는 문자열(예: **dev**)입니다.
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	{
		"auths":{
		"cloud.openshift.com":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		},
		"quay.io":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		}
		}
		}

4.10.11.1.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 4.33. 네트워크 매개변수

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체 **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.

매개변수	설명	값
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.

networking.clusterNetwork	Pod의 IP 주소 블록입니다.
	
	기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다.
	
	여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.
		개체의 배열입니다. 예를 들면 다음과 같습니다.
		
		`networking:`
		`  clusterNetwork:`
		`  - cidr: 10.128.0.0/14`
		`    hostPrefix: 23`

networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된**cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.

networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다.
	
	OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.
		CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다.
		
		`networking:`
		`  serviceNetwork:`
		`   - 172.30.0.0/16`

networking.machineNetwork	시스템의 IP 주소 블록입니다.
	
	여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.
		개체의 배열입니다. 예를 들면 다음과 같습니다.
		
		`networking:`
		`  machineNetwork:`
		`  - cidr: 10.0.0.0/16`

매개변수	설명	값
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16**  **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

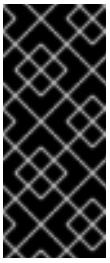
4.10.11.1.3. 선택적 구성 매개변수

선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 4.34. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열

매개변수	설명	값
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은 **3**입니다.
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열

매개변수	설명	값
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 인터넷에서 액세스할 수 없는 사설 클러스터를 배포하려면 **publish**를 **Internal**로 설정합니다. 기본값은 **External**입니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

4.10.11.1.4. 선택적 AWS 구성 매개변수

선택적 AWS 구성 매개변수는 다음 표에 설명되어 있습니다.

표 4.35. 선택적 AWS 매개변수

매개변수	설명	값
compute.platform.aws.amiID	클러스터의 컴퓨팅 머신을 부팅하는 데 사용되는 AWS AMI입니다. 이는 사용자 지정 RHCOS AMI가 필요한 리전에 필요합니다.	설정된 AWS 리전에 속하는 게시된 또는 사용자 지정 RHCOS AMI입니다.
compute.platform.aws.iamRole	컴퓨팅 머신 풀 인스턴스 프로파일에 적용되는 기존 AWS IAM 역할입니다. 이러한 필드를 사용하여 이름 지정 체계와 일치하고 IAM 역할에 대해 사전 정의된 권한 경계를 포함할 수 있습니다. 정의되지 않은 경우 설치 프로그램은 새 IAM 역할을 생성합니다.	유효한 AWS IAM 역할의 이름입니다.
compute.platform.aws.rootVolume.iops	루트 볼륨용으로 예약된 IOPS(초당 입/출력 작업)입니다.	정수(예: **4000**)
compute.platform.aws.rootVolume.size	루트 볼륨의 크기(GiB)입니다.	정수(예: **500**)
compute.platform.aws.rootVolume.type	루트 볼륨의 유형입니다.	유효한 AWS EBS 볼륨 유형 (예: **io1**)

매개변수	설명	값
compute.platform.aws.rootVolume.kmsKeyARN	KMS 키의 Amazon 리소스 이름(키 ARN)입니다. 이는 특정 KMS 키를 사용하여 작업자 노드의 OS 볼륨을 암호화하는 데 필요합니다.	유효한 키 ID 또는 키 ARN 입니다.
compute.platform.aws.type	컴퓨팅 시스템의 EC2 인스턴스 유형입니다.	유효한 AWS 인스턴스 유형 (예: **m4.2xlarge**) 다음 **머신의 인스턴스 유형**을 참조하십시오.
compute.platform.aws.zones	설치 프로그램이 컴퓨팅 머신 풀에 필요한 시스템을 생성하는 가용성 영역입니다. 자체 VPC를 제공하는 경우 해당 가용성 영역에 서브넷을 제공해야 합니다.	**us-east-1c**와 같이 YAML 시퀀스로 표시되는 유효한 AWS 가용성 영역의 목록입니다.
compute.aws.region	설치 프로그램이 컴퓨팅 리소스를 생성하는 AWS 리전입니다.	유효한 모든 AWS 리전(예: **us-east-1**)
controlPlane.platform.aws.amiID	클러스터의 컨트롤 플레인 시스템을 시작하는 데 사용되는 AWS AMI입니다. 이는 사용자 지정 RHCOS AMI가 필요한 리전에 필요합니다.	설정된 AWS 리전에 속하는 게시된 또는 사용자 지정 RHCOS AMI입니다.
controlPlane.platform.aws.iamRole	컨트롤 플레인 시스템 풀 인스턴스 프로파일에 적용되는 기존 AWS IAM 역할입니다. 이러한 필드를 사용하여 이름 지정 체계와 일치하고 IAM 역할에 대해 사전 정의된 권한 경계를 포함할 수 있습니다. 정의되지 않은 경우 설치 프로그램은 새 IAM 역할을 생성합니다.	유효한 AWS IAM 역할의 이름입니다.
controlPlane.platform.aws.rootVolume.kmsKeyARN	KMS 키의 Amazon 리소스 이름(키 ARN)입니다. 이는 특정 KMS 키를 사용하여 컨트롤 플레인 노드의 OS 볼륨을 암호화하는 데 필요합니다.	유효한 키 ID 및 키 ARN.
controlPlane.platform.aws.type	컨트롤 플레인 시스템의 EC2 인스턴스 유형입니다.	유효한 AWS 인스턴스 유형 (예: **m5.xlarge**). 다음 **머신의 인스턴스 유형**을 참조하십시오.
controlPlane.platform.aws.zones	설치 프로그램이 컨트롤 플레인 시스템 풀에 필요한 시스템을 생성하는 가용성 영역입니다.	**us-east-1c**와 같이 YAML 시퀀스로 표시되는 유효한 AWS 가용성 영역의 목록입니다.

매개변수	설명	값
controlPlane.aws.region	설치 프로그램이 컨트롤 플레인 리소스를 생성하는 AWS 리전입니다.	유효한 AWS 리전(예: **us-east-1**)
platform.aws.amiID	클러스터의 모든 머신을 부팅하는 데 사용되는 AWS AMI입니다. 설정된 경우 AMI는 클러스터와 동일한 리전에 속해 있어야 합니다. 이는 사용자 지정 RHCOS AMI가 필요한 리전에 필요합니다.	설정된 AWS 리전에 속하는 게시된 또는 사용자 지정 RHCOS AMI입니다.
platform.aws.hostedZone	클러스터의 기존 Route 53 개인 호스팅 영역입니다. 자체 VPC를 제공하는 경우에만 기존 호스팅 영역을 사용할 수 있습니다. 호스팅 영역은 설치 전에 사용자 제공 VPC와 이미 연결되어 있어야 합니다. 또한 호스팅 영역의 도메인은 클러스터 도메인 또는 클러스터 도메인의 상위 도메인이어야 합니다. 정의되지 않은 경우 설치 프로그램은 새 호스팅 영역을 생성합니다.	문자열 (예: **Z3URY6TWQ91KVV**)
platform.aws.serviceEndpoints.name	AWS 서비스 엔드 포인트 이름. 사용자 지정 엔드 포인트는 FIPS와 같은 대체 AWS 엔드 포인트를 사용해야하는 경우에만 필요합니다. EC2, S3, IAM, Elastic Load Balancing, Tagging, Route 53 및 STS AWS 서비스에 대해 사용자 지정 API 엔드 포인트를 지정할 수 있습니다.	유효한 AWS 서비스 엔드 포인트 이름.
platform.aws.serviceEndpoints.url	AWS 서비스 엔드 포인트 URL. URL은 **https** 프로토콜을 사용해야하며 호스트는 인증서를 신뢰해야 합니다.	유효한 AWS 서비스 엔드 포인트 URL.
platform.aws.userTags	설치 프로그램이 생성하는 모든 리소스에 태그로 추가하는 키와 값의 맵입니다.	유효한 YAML 맵(예: **<key>: <value>** 형식의 키 값 쌍) AWS 태그에 대한 자세한 내용은 AWS 문서의 Amazon EC2 리소스 태그 지정을 참조하십시오.

매개변수	설명	값
platform.aws.subnets	설치 프로그램이 VPC를 자동으로 생성하지 않고 VPC를 직접 제공하는 경우 사용할 클러스터의 서브넷을 지정합니다. 서브넷은 사용자가 지정하는 동일한 **machineNetwork[].cidr** 범위의 일부여야 합니다. 표준 클러스터의 경우 각 가용성 영역의 퍼블릭 및 프라이빗 서브넷을 지정합니다. 개인 클러스터의 경우 각 가용성 영역의 프라이빗 서브넷을 지정합니다.	유효한 서브넷 ID.

4.10.11.2. 클러스터 설치를 위한 최소 리소스 요구 사항

각 클러스터 시스템이 다음과 같은 최소 요구사항을 충족해야 합니다.

표 4.36. 최소 리소스 요구사항

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
부트스트랩	RHCOS	4	16GB	100GB	300
컨트롤 플레인	RHCOS	4	16GB	100GB	300
컴퓨팅	RHCOS, RHEL 7.9 또는 RHEL 8.4 [3]	2	8GB	100GB	300

1. SMT(동시 멀티스레딩) 또는 하이퍼 스레딩이 활성화되지 않은 경우 하나의 vCPU는 하나의 물리적 코어와 동일합니다. 활성화하면 다음과 같은 공식을 사용하여 해당 비율을 계산합니다. (코어당 스레드 수 × 코어 수) × 소켓 수 = vCPU 수

2. OpenShift Container Platform 및 Kubernetes는 디스크 성능에 민감하며 특히 10ms p99 fsync 기간이 필요한 컨트롤 플레인 노드의 etcd에 더 빠른 스토리지가 권장됩니다. 많은 클라우드 플랫폼에서 스토리지 크기와 IOPS를 함께 확장되므로 충분한 성능을 얻으려면 스토리지 볼륨을 과도하게 할당해야 할 수 있습니다.

3. 사용자가 프로비저닝한 모든 설치와 마찬가지로 클러스터에서 RHEL 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. RHEL 7 컴퓨팅 머신 사용은 더 이상 사용되지 않으며 향후 OpenShift Container Platform 4 릴리스에서 제거될 예정입니다.

4.10.11.3. 지원되는 AWS 머신 유형

다음 AWS(Amazon Web Services) 인스턴스 유형은 OpenShift Container Platform에서 지원됩니다.

예 4.20. 시스템의 인스턴스 유형

인스턴스 유형	부트스트랩	컨트롤 플레인	컴퓨팅
i3.large	x		
m4.large			x
m4.xlarge		x	x
m4.2xlarge		x	x
m4.4xlarge		x	x
m4.10xlarge		x	x
m4.16xlarge		x	x
m5.large			x
m5.xlarge		x	x
m5.2xlarge		x	x
m5.4xlarge		x	x
m5.8xlarge		x	x
m5.12xlarge		x	x
m5.16xlarge		x	x
m5a.large			x
m5a.xlarge		x	x
m5a.2xlarge		x	x
m5a.4xlarge		x	x
m5a.8xlarge		x	x
m5a.12xlarge		x	x
m5a.16xlarge		x	x
m6i.xlarge		x	x

인스턴스 유형	부트스트랩	컨트롤 플레인	컴퓨팅
m6i.2xlarge		x	x
m6i.4xlarge		x	x
m6i.8xlarge		x	x
m6i.16xlarge		x	x
c4.2xlarge		x	x
c4.4xlarge		x	x
c4.8xlarge		x	x
c5.xlarge			x
c5.2xlarge		x	x
c5.4xlarge		x	x
c5.9xlarge		x	x
c5.12xlarge		x	x
c5.18xlarge		x	x
c5.24xlarge		x	x
c5a.xlarge			x
c5a.2xlarge		x	x
c5a.4xlarge		x	x
c5a.8xlarge		x	x
c5a.12xlarge		x	x
c5a.16xlarge		x	x
c5a.24xlarge		x	x
r4.large			x
r4.xlarge		x	x

인스턴스 유형	부트스트랩	컨트롤 플레인	컴퓨팅
r4.2xlarge		x	x
r4.4xlarge		x	x
r4.8xlarge		x	x
r4.16xlarge		x	x
r5.large			x
r5.xlarge		x	x
r5.2xlarge		x	x
r5.4xlarge		x	x
r5.8xlarge		x	x
r5.12xlarge		x	x
r5.16xlarge		x	x
r5.24xlarge		x	x
r5a.large			x
r5a.xlarge		x	x
r5a.2xlarge		x	x
r5a.4xlarge		x	x
r5a.8xlarge		x	x
r5a.12xlarge		x	x
r5a.16xlarge		x	x
r5a.24xlarge		x	x
t3.large			x
t3.xlarge			x
t3.2xlarge			x

인스턴스 유형	부트스트랩	컨트롤 플레인	컴퓨팅
t3a.large			x
t3a.xlarge			x
t3a.2xlarge			x

4.10.11.4. AWS용 샘플 사용자 지정 install-config.yaml 파일

설치 구성 파일(**install-config.yaml**)을 사용자 지정하여 OpenShift Container Platform 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.

중요

이 샘플 YAML 파일은 참조용으로만 제공됩니다. 수동으로 생성한 설치 구성 파일에 매개변수 값을 입력하기 위한 리소스로 사용합니다.

```
apiVersion: v1
baseDomain: example.com (1)
credentialsMode: Mint (2)
controlPlane: (3) (4)
  hyperthreading: Enabled (5)
  name: master
  platform:
    aws:
      zones:
      - us-gov-west-1a
      - us-gov-west-1b
      rootVolume:
        iops: 4000
        size: 500
        type: io1 (6)
      type: m5.xlarge
  replicas: 3
compute: (7)
- hyperthreading: Enabled (8)
  name: worker
  platform:
    aws:
      rootVolume:
        iops: 2000
        size: 500
        type: io1 (9)
      type: c5.4xlarge
      zones:
      - us-gov-west-1c
  replicas: 3
metadata:
```

```
  name: test-cluster 10
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  aws:
    region: us-gov-west-1 11
    userTags:
      adminContact: jdoe
      costCenter: 7536
    subnets: 12
    - subnet-1
    - subnet-2
    - subnet-3
    amiID: ami-96c6f8f7 13 14
    serviceEndpoints: 15
      - name: ec2
        url: https://vpce-id.ec2.us-west-2.vpce.amazonaws.com
    hostedZone: Z3URY6TWQ91KVV 16
fips: false 17
sshKey: ssh-ed25519 AAAA... 18
publish: Internal 19
pullSecret: '{"auths": ...}' 20
additionalTrustBundle: | 21
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
```

1 10 11 13 20 필수 항목입니다.

2 옵션: 이 매개 변수를 추가하여 Cloud Credential Operator (CCO)가 인증 정보의 기능을 동적으로 판별하도록 하지 않고 CCO가 지정된 모드를 사용하도록 강제합니다. CCO 모드에 대한 자세한 내용은 *Platform Operators 참조* 콘텐츠의 *Cloud Credential Operator* 항목을 참조하십시오.

3 7 이러한 매개변수와 값을 지정하지 않으면 설치 프로그램은 기본값을 적용합니다.

4 **controlPlane** 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

5 8 동시 멀티스레딩 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. 일부 클러스터 시스템에서 동시 멀티스레딩을 비활성화할 경우에는 해당 멀티스레딩을 모든 클러스터 시스템에서 비활성화해야 합니다.

중요

동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다. 동시 멀티스레딩을 비활성화하는 경우 시스템에 더 큰 인스턴스 유형(예: **m4.2xlarge** 또는 **m5.2xlarge**)을 사용합니다.

6 9 대규모의 클러스터의 경우 고속 etcd 스토리지를 구성하려면 스토리지 유형을 **IO1**로 설정하고 **iops**는 **2000**으로 설정합니다.

12 자체 VPC를 제공하는 경우 클러스터가 사용하는 각 가용성 영역의 서브넷을 지정합니다.

14 클러스터 머신을 시작하는 데 사용되는 AMI의 ID입니다. 설정된 경우 AMI는 클러스터와 동일한 리전에 속해 있어야 합니다.

15 AWS 서비스 엔드 포인트입니다. 알 수 없는 AWS 리전에 설치할 때 사용자 지정 엔드 포인트가 필요합니다. 엔드포인트 URL은 **https** 프로토콜을 사용해야하며 호스트는 인증서를 신뢰해야 합니다.

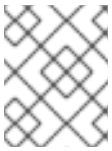
16 기존 Route 53 개인 호스팅 영역의 ID입니다. 기존 호스팅 영역을 제공하려면 클러스터를 설치하기 전에 자체 VPC를 제공하고 호스팅 영역이 VPC와 연결되어 있어야 합니다. 정의되지 않은 경우 설치 프로그램은 새 호스팅 영역을 생성합니다.

17 FIPS 모드 활성화 또는 비활성화 여부입니다. 기본적으로 FIPS 모드는 비활성화됩니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.

중요

FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.

18 선택사항으로, 클러스터의 시스템에 액세스하는 데 사용할 **sshKey** 값을 제공할 수도 있습니다.



참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

19 클러스터의 사용자 끝점을 게시하는 방법. 인터넷에서 액세스할 수 없는 프라이빗 클러스터를 배포하려면 **publish**를 **Internal**로 설정합니다. 기본값은 **External**입니다.

21 사용자 정의 CA 인증서입니다. 이는 AWS API에 사용자 정의 CA 신뢰 번들이 필요하기 때문에 AWS C2S Top Secret Region에 배포할 때 필요합니다.

4.10.11.5. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 HTTP 또는 HTTPS 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 OpenShift Container Platform 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.

- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 API에 대한 호출을 포함하여 모든 클러스터 발신(Egress) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.

참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 Red Hat OpenStack Platform (RHOSP)에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

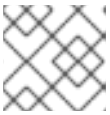
- **ec2.<region>.amazonaws.com,elasticloadbalancing.<region>.amazonaws.com** 및 **s3.<region>.amazonaws.com** 끝점을 VPC 끝점에 추가했습니다. 이러한 끝점은 노드에서 AWS EC2 API로 요청을 완료하는 데 필요합니다. 프록시는 노드 수준이 아닌 컨테이너 수준에서 작동하므로 이러한 요청을 AWS 사설 네트워크를 통해 AWS EC2 API로 라우팅해야 합니다. 프록시 서버의 허용 목록에 EC2 API의 공용 IP 주소를 추가하는 것만으로는 충분하지 않습니다.

절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. 클러스터 외부에서 HTTP 연결을 구축하는 데 사용할 프록시 URL입니다. URL 스키마는 **http**여야 합니다.
2. 클러스터 외부에서 HTTPS 연결을 구축하는 데 사용할 프록시 URL입니다.
3. 대상 도메인 이름, IP 주소 또는 프록시에서 제외할 기타 네트워크 CIDR로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. ***를 사용하여 모든 대상에 대해 프록시를 바이패스합니다.
4. 이 값을 제공하면 설치 프로그램에서 추가 CA 인증서를 보유할 **openshift -config 네임스페이스에 user-ca- bundle** 이라는 구성 맵을 생성합니다. **additionalTrustBundle** 및 하나 이상의 프록시 설정을 제공하는 경우 **프록시** 오브젝트는 **trustedCA** 필드의 **user-ca-bundle** 구성 맵을 참조하도록 구성됩니다. 그러면 Cluster Network Operator에서 **trustedCA** 매개변수에 대해 지정된 콘텐츠를 RHCOS 신뢰 번들과 병합하는 **trusted-ca-bundle** 구성 맵을 생성합니다. 프록시의 ID 인증서를 RHCOS 트러스트 번들에 있는 기관에서 서명하지 않은 경우 **additionalTrustBundle** 필드가 있어야 합니다.



참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 OpenShift Container Platform을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 nil이 됩니다.

참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

4.10.12. 클러스터 배포

호환되는 클라우드 플랫폼에 OpenShift Container Platform을 설치할 수 있습니다.

중요

최초 설치 과정에서 설치 프로그램의 **create cluster** 명령을 한 번만 실행할 수 있습니다.

사전 요구 사항

- 클러스터를 호스팅하는 클라우드 플랫폼으로 계정을 구성합니다.
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

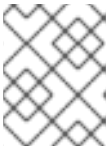
프로세스

1. 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터 배포를 초기화합니다.

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** 값으로 사용자 지정한 **./install-config.yaml** 파일의 위치를 지정합니다.

2 다른 설치 세부 사항을 보려면 **info** 대신 **warn**, **debug** 또는 **error**를 지정합니다.



참고

호스트에 구성된 클라우드 공급자 계정에 클러스터를 배포하기에 충분한 권한이 없는 경우, 설치 프로세스가 중단되고 누락된 권한을 알리는 메시지가 표시됩니다.

클러스터 배포가 완료되면 웹 콘솔로 연결되는 링크와 **kubeadmin** 사용자의 인증 정보가 포함된 클러스터 액세스 지침이 사용자 터미널에 표시됩니다.

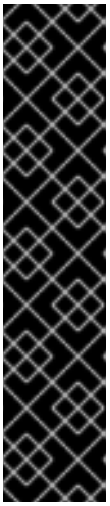
출력 예

```
...
INFO Install complete!
```

```
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```

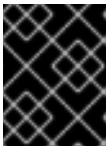
참고

설치에 성공하면 클러스터 액세스 및 인증 정보도 **<installation_directory>/.openshift_install.log**로 출력됩니다.



중요

- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.



중요

설치 프로그램에서 생성되는 파일이나 설치 프로그램을 삭제해서는 안 됩니다. 클러스터를 삭제하려면 두 가지가 모두 필요합니다.

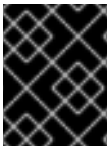
2. 선택사항: 클러스터를 설치하는 데 사용한 IAM 계정에서 **AdministratorAccess** 정책을 제거하거나 비활성화합니다.

참고

AdministratorAccess 정책에서 제공하는 승격된 권한은 설치 중에만 필요합니다.

4.10.13. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift CLI(**oc**)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 **oc**를 설치할 수 있습니다.



중요

이전 버전의 **oc**를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 **oc**를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 Linux에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Linux Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. **oc** 바이너리를 **PATH**에 있는 디렉터리에 배치합니다.
PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치
다음 절차에 따라 Windows에 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Windows Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. ZIP 프로그램으로 아카이브의 압축을 풉니다.
5. **oc** 바이너리를 **PATH**에 있는 디렉터리로 이동합니다.
PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치
다음 절차에 따라 macOS에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 MacOSX Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.

4. 아카이브의 압축을 해제하고 압축을 풉니다.

5. **oc** 바이너리 PATH의 디렉터리로 이동합니다.
PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

4.10.14. CLI를 사용하여 클러스터에 로그인

클러스터 **kubeconfig** 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. **kubeconfig** 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- **oc** CLI를 설치했습니다.

프로세스

1. **kubeadmin** 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 **oc** 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

4.10.15. 웹 콘솔을 사용하여 클러스터에 로그인

kubeadmin 사용자는 OpenShift Container Platform 설치 후 기본적으로 존재합니다. OpenShift Container Platform 웹 콘솔을 사용하여 **kubeadmin** 사용자로 클러스터에 로그인할 수 있습니다.

사전 요구 사항

- 설치 호스트에 대한 액세스 권한이 있어야 합니다.
- 클러스터 설치를 완료했으며 모든 클러스터 Operator를 사용할 수 있습니다.

프로세스

1. 설치 호스트의 **kubeadmin-password** 파일에서 **kubeadmin** 사용자의 암호를 가져옵니다.

```
$ cat <installation_directory>/auth/kubeadmin-password
```

참고

대안으로 설치 호스트의 **<installation_directory>/.openshift_install.log** 로그 파일에서 **kubeadmin** 암호를 가져올 수 있습니다.

2. OpenShift Container Platform 웹 콘솔 경로를 나열합니다.

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```

참고

대안으로 설치 호스트의 **<installation_directory>/.openshift_install.log** 로그 파일에서 OpenShift Container Platform 경로를 가져올 수 있습니다.

출력 예

```
console     console-openshift-console.apps.<cluster_name>.<base_domain>          console
https   reencrypt/Redirect   None
```

3. 웹 브라우저의 이전 명령 출력에 자세히 설명된 경로로 이동하고 **kubeadmin** 사용자로 로그인합니다.

추가 리소스

- OpenShift Container Platform 웹 콘솔 액세스 및 이해에 대한 자세한 내용은 웹 콘솔에 액세스를 참조하십시오.

4.10.16. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

4.10.17. 다음 단계

- 설치를 확인합니다.
- 클러스터를 사용자 지정합니다.

- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.
- 필요한 경우 클라우드 공급자 인증 정보를 제거 할 수 있습니다.

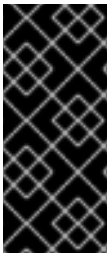
4.11. AWS 중국 리전에 클러스터 설치

OpenShift Container Platform 버전 4.9에서는 다음과 같은 AWS(Amazon Web Services) 중국 리전에 클러스터를 설치할 수 있습니다.

- **cn-north-1(** 베이징)
- **cn-northwest-1** (닝샤)

4.11.1. 사전 요구 사항

- ICP(Internet Content Provider) 라이센스가 있습니다.
- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비 에 대한 문서를 읽습니다.
- 클러스터를 호스팅할 AWS 계정을 구성 했습니다.
- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성 해야 합니다.
- 사용자의 환경에서 클라우드 ID 및 액세스 관리(IAM) API에 액세스할 수 없거나 **kube-system** 네임스페이스에 관리자 수준의 인증 정보 시크릿을 저장하지 않으려면 IAM 인증 정보 수동 생성 및 유지 관리를 참조하십시오.



중요

컴퓨터에 AWS 프로필이 저장되어 있는 경우 다단계 인증 장치를 사용하는 동안 생성한 임시 세션 토큰을 해당 프로필이 사용해서는 안 됩니다. 클러스터는 클러스터의 전체 수명 동안 현재 AWS 자격 증명을 계속 사용하여 AWS 리소스를 생성하므로 장기 자격 증명을 사용해야 합니다. 적절한 키를 생성하려면 AWS 문서의 IAM 사용자의 액세스 키 관리를 참조하십시오. 설치 프로그램을 실행할 때 키를 제공할 수 있습니다.

4.11.2. 설치 요구 사항

Red Hat은 AWS 중국 리전의 RHCOS(Red Hat Enterprise Linux CoreOS) Amazon 머신 이미지(AMI)를 게시하지 않습니다.

클러스터를 설치하려면 먼저 다음을 수행해야 합니다.

- 사용자 지정 RHCOS AMI를 업로드합니다.
- 수동으로 설치 구성 파일 (**install-config.yaml**)을 생성합니다.
- 설치 구성 파일에서 AWS 리전 및 관련 사용자 지정 AMI를 지정합니다.

OpenShift Container Platform 설치 프로그램을 사용하여 설치 구성 파일을 생성할 수 없습니다. 설치 프로그램에서 RHCOS AMI에 대한 기본 지원 없이 AWS 리전을 나열하지 않습니다.

4.11.3. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.

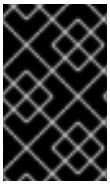
중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

4.11.4. 프라이빗 클러스터

외부 엔드 포인트를 노출하지 않는 비공개 OpenShift Container Platform 클러스터를 배포할 수 있습니다. 프라이빗 클러스터는 내부 네트워크에서만 액세스할 수 있으며 인터넷에 표시되지 않습니다.

기본적으로 OpenShift Container Platform은 공개적으로 액세스 가능한 DNS 및 끝점을 사용하여 프로비저닝됩니다. 따라서 개인 클러스터를 배포할 때 클러스터에서 DNS, Ingress Controller 및 API 서버를 비공개로 설정할 수 있습니다. 즉 클러스터 리소스는 내부 네트워크에서만 액세스할 수 있고 인터넷에는 노출되지 않습니다.



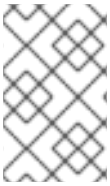
중요

클러스터에 퍼블릭 서브넷이 있는 경우 관리자가 생성한 로드 밸런서 서비스에 공개적으로 액세스할 수 있습니다. 클러스터 보안을 보장하기 위해 이러한 서비스에 명시적으로 주석이 지정되었는지 확인합니다.

프라이빗 클러스터를 배포하려면 다음을 수행해야 합니다.

- 요구 사항을 충족하는 기존 네트워킹을 사용합니다. 네트워크의 다른 클러스터 사이에 클러스터 리소스를 공유할 수 있습니다.
- 다음에 액세스할 수 있는 머신에서 배포합니다.
  - 프로비저닝하는 클라우드용 API 서비스
  - 프로비저닝하는 네트워크의 호스트
  - 설치 미디어를 가져올 인터넷

이러한 액세스 요구사항을 충족하고 회사의 지침을 따르는 모든 시스템을 사용할 수 있습니다. 예를 들어, 이 시스템은 클라우드 네트워크의 배스천 호스트가 될 수 있습니다.



참고

AWS 중국 리전은 VPC와 네트워크 간의 VPN 연결을 지원하지 않습니다. 베이징 및 닝샤 리전의 Amazon VPC 서비스에 대한 자세한 내용은 AWS 중국 리전 설명서의 Amazon Virtual Private Cloud를 참조하십시오.

4.11.4.1. AWS의 개인 클러스터

AWS(Amazon Web Services)에 개인 클러스터를 생성하려면 클러스터를 호스팅할 기존 프라이빗 VPC와 서브넷을 제공해야 합니다. 또한 설치 프로그램에서 클러스터에 필요한 DNS 레코드를 확인할 수 있어야 합니다. 설치 프로그램은 개인 네트워크에서만 액세스할 수 있도록 Ingress Operator 및 API 서버를 구성합니다.

클러스터가 AWS API에 액세스하려면 여전히 인터넷 접속이 필요합니다.

다음은 프라이빗 클러스터를 설치할 때 필요하지 않거나 생성되지 않는 항목들입니다.

- 퍼블릭 서브넷
- 공용 인그레스를 지원하는 공용 로드 밸런서
- 클러스터의 **baseDomain**과 일치하는 공용 Route 53 영역

설치 프로그램은 사용자가 지정하는 **baseDomain**을 사용하여 프라이빗 Route 53 영역과 클러스터에 필요한 레코드를 생성합니다. Operator가 클러스터에 대한 공용 레코드를 생성하지 않고 사용자가 지정하는 프라이빗 서브넷에 모든 클러스터 시스템이 배치되도록 클러스터가 구성됩니다.

4.11.4.1.1. 제한

프라이빗 클러스터에 공용 기능을 추가하는 기능은 제한됩니다.

- 설치 후에는 VPC에서 사용 중인 각 가용성 영역의 퍼블릭 서브넷 생성, 공용 로드 밸런서 생성, 6443(Kubernetes API 포트)의 인터넷 트래픽을 허용하도록 컨트롤 플레인 보안 그룹 구성 등 추가 조치 없이 Kubernetes API 엔드포인트를 공개할 수 없습니다.
- 공용 서비스 유형 로드 밸런서를 사용하는 경우 AWS가 공용 로드 밸런서를 생성하는 데 사용할 수 있도록 각 가용성 영역의 퍼블릭 서브넷에 **kubernetes.io/cluster/<cluster-infra-id>: shared** 태그를 지정해야 합니다.

4.11.5. 사용자 지정 VPC 사용 정보

OpenShift Container Platform 4.9에서는 AWS(Amazon Web Services)의 기존 Amazon VPC(Virtual Private Cloud)에 있는 기존 서브넷에 클러스터를 배포할 수 있습니다. 기존 GCP VPC에 OpenShift Container Platform을 배포하면 새 계정의 한도 제한 적용을 받지 않거나 회사의 지침에 따른 운영 제한을 보다 쉽게 준수할 수 있습니다. VPC를 직접 생성하는 데 필요한 인프라 생성 권한을 받을 수 없는 경우 이 설치 옵션을 사용합니다.

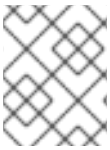
설치 프로그램은 기존 서브넷에 있는 다른 구성 요소를 알 수 없으므로 사용자를 대신하여 서브넷 CIDR 등을 선택할 수 없습니다. 클러스터를 직접 설치하는 서브넷에 대한 네트워킹을 구성해야 합니다.

4.11.5.1. VPC 사용 요구사항

설치 프로그램은 더 이상 다음 구성 요소를 생성하지 않습니다.

- 인터넷 게이트웨이

- NAT 게이트웨이
- 서브넷
- 라우팅 테이블
- VPC
- VPC DHCP 옵션
- VPC 끝점



참고

설치 프로그램을 사용하려면 클라우드 제공 DNS 서버를 사용해야 합니다. 사용자 지정 DNS 서버 사용은 지원되지 않으며 이로 인해 설치에 실패합니다.

사용자 지정 VPC를 사용하는 경우, 사용할 클러스터와 설치 프로그램에 맞게 VPC와 해당 서브넷을 구성해야 합니다. AWS VPC 생성 및 관리에 대한 자세한 내용은 AWS 문서의 Amazon VPC 콘솔 마법사 구성 및 VPC 및 서브넷 작업을 참조하십시오.

설치 프로그램은 다음을 수행할 수 없습니다.

- 클러스터에서 사용할 네트워크 범위를 세분합니다.
- 서브넷의 라우팅 테이블을 설정합니다.
- DHCP와 같은 VPC 옵션을 설정합니다.

클러스터를 설치하기 전에 이러한 작업을 완료해야 합니다. AWS VPC의 네트워킹 구성에 대한 자세한 내용은 VPC 네트워킹 구성 요소 및 경로 테이블을 참조하십시오.

VPC는 다음 특성을 충족해야 합니다.

- VPC는 **kubernetes.io/cluster/.*: owned** 태그를 사용해서는 안 됩니다.
  설치 프로그램은 **kubernetes.io/cluster/.*: shared** 태그를 추가하도록 서브넷을 수정하므로 서브넷에 사용 가능한 여유 태그 슬롯이 하나 이상 있어야 합니다. AWS 문서의 태그 제한 사항을 참조하여 설치 프로그램이 사용자가 지정하는 각 서브넷에 태그를 추가할 수 있는지 확인합니다.

- 클러스터가 VPC에 연결된 Route 53 영역을 사용하여 클러스터의 내부 DNS 레코드를 확인할 수 있도록 VPC에서 **enableDnsSupport 및 enableDnsHostnames** 속성을 활성화해야 합니다.
  AWS 문서에서 VPC의 DNS 지원을 참조하십시오.
  자체 Route 53 호스팅 프라이빗 영역을 사용하려면 클러스터를 설치하기 전에 기존 호스팅 영역을 VPC와 연결해야 합니다. **install-config.yaml** 파일에서 **platform.aws.hostedZone** 필드를 사용하여 호스팅 영역을 정의할 수 있습니다.

- 공용 액세스 권한이 있는 클러스터를 사용하는 경우 클러스터가 사용하는 각 가용성 영역의 퍼블릭 및 프라이빗 서브넷을 만들어야 합니다. 각 가용성 영역에는 퍼블릭 서브넷과 프라이빗 서브넷이 1개 이상 포함될 수 있습니다.

연결이 끊긴 환경에서 작업 중인 경우에는 EC2 및 ELB 끝점의 공용 IP 주소에 도달할 수 없습니다. 이 문제를 해결하려면 VPC 끝점을 생성하여 클러스터가 사용 중인 서브넷에 연결해야 합니다. 올바른 끝점의 이름은 다음과 같습니다.

- **ec2.<region>.amazonaws.com.cn**

- **elasticloadbalancing.<region>.amazonaws.com**
- **s3.<region>.amazonaws.com**

필수 VPC 구성 요소

시스템과의 통신을 허용하는 서브넷과 적합한 VPC를 제공해야 합니다.

<table>
<tr><th>구성 요소</th><th>AWS 유형</th><th>설명</th></tr>
<tr><td>VPC</td><td>AWS::EC2::VPC
AWS::EC2::VPCEndpoint</td><td>클러스터에서 사용할 공용 VPC를 제공해야 합니다. VPC는 각 서브넷의 라우팅 테이블을 참조하는 끝점을 사용하여 S3에서 호스팅되는 레지스트리와의 통신을 개선합니다.</td></tr>
<tr><td>퍼블릭 서브넷</td><td>AWS::EC2::Subnet
AWS::EC2::SubnetNetworkAclAssociation</td><td>VPC에는 1~3개의 가용성 영역에 대한 퍼블릭 서브넷이 있어야 하며 이 서브넷을 적절한 인그레스 규칙과 연결해야 합니다.</td></tr>
<tr><td>인터넷 게이트웨이</td><td>AWS::EC2::InternetGateway
AWS::EC2::VPCGatewayAttachment
AWS::EC2::RouteTable
AWS::EC2::Route
AWS::EC2::SubnetRouteTableAssociation
AWS::EC2::NatGateway
AWS::EC2::EIP</td><td>공용 경로가 있는 공용 인터넷 게이트웨이가 VPC에 연결되어 있어야 합니다. 제공된 템플릿에서 각 퍼블릭 서브넷에는 EIP 주소를 갖는 NAT 게이트웨이가 있습니다. 이러한 NAT 게이트웨이를 사용하면 프라이빗 서브넷 인스턴스와 같은 클러스터 리소스가 인터넷에 도달할 수 있으므로 일부 제한된 네트워크 또는 프록시 시나리오에는 필요하지 않습니다.</td></tr>
<tr><td>네트워크 액세스 제어</td><td>AWS::EC2::NetworkAcl
AWS::EC2::NetworkAclEntry</td><td>VPC가 다음 포트에 액세스할 수 있어야 합니다.<table><tr><th>포트</th><th>이유</th></tr><tr><td>80</td><td>인바운드 HTTP 트래픽</td></tr><tr><td>443</td><td>인바운드 HTTPS 트래픽</td></tr><tr><td>22</td><td>인바운드 SSH 트래픽</td></tr></table></td></tr>
</table>

구성 요소	AWS 유형	설명	
		1024 - 65535	인바운드 임시 트래픽
		0 - 65535	아웃바운드 임시 트래픽
프라이빗 서브넷	• **AWS::EC2::Subnet** • **AWS::EC2::RouteTable** • **AWS::EC2::SubnetRouteTableAssociation**	VPC에 프라이빗 서브넷이 포함될 수 있습니다. 제공된 CloudFormation 템플릿은 1~3개 가용성 영역의 프라이빗 서브넷을 생성할 수 있습니다. 프라이빗 서브넷을 사용하는 경우 적절한 경로와 테이블을 제공해야 합니다.	

4.11.5.2. VPC 검증

제공한 서브넷이 적합한지 확인하기 위해 설치 프로그램이 다음 데이터를 확인합니다.

- 사용자가 지정하는 모든 서브넷이 있는지 여부.
- 사용자가 프라이빗 서브넷을 제공합니다.
- 서브넷 CIDR이 사용자가 지정한 시스템 CIDR에 속합니다.
- 각 가용성 영역에 서브넷을 제공합니다. 각 가용성 영역에는 퍼블릭 서브넷과 프라이빗 서브넷이 하나씩 포함됩니다. 개인 클러스터를 사용하는 경우 각 가용성 영역에 프라이빗 서브넷만 제공합니다. 그렇지 않으면 각 가용성 영역에 퍼블릭 서브넷과 프라이빗 서브넷을 하나씩만 제공합니다.
- 각 프라이빗 서브넷 가용성 영역에 퍼블릭 서브넷을 제공합니다. 프라이빗 서브넷을 제공하지 않는 가용성 영역에서는 시스템이 프로비저닝되지 않습니다.

기존 VPC를 사용하는 클러스터를 제거해도 VPC는 삭제되지 않습니다. VPC에서 OpenShift Container Platform 클러스터를 제거하면 클러스터가 사용한 서브넷에서 **kubernetes.io/cluster/.*: shared** 태그가 제거됩니다.

4.11.5.3. 권한 분할

OpenShift Container Platform 4.3부터 클러스터를 배포하는 데 설치 프로그램에서 프로비저닝한 인프라 클러스터에 필요한 권한 중 일부가 필요하지 않게 되었습니다. 이 변경 사항은 회사의 권한 분류와 유사합니다. 즉 일부 개인의 경우 클라우드에서 다른 사람들과 다른 리소스를 생성할 수 있습니다. 예를 들어 인스턴스, 버킷, 로드 밸런서와 같은 애플리케이션 관련 항목은 생성할 수 있지만 VPC 서브넷 또는 인그레스 규칙과 같은 네트워킹 관련 구성 요소는 생성하지 못할 수 있습니다.

클러스터를 생성할 때 사용하는 GCP 자격 증명에는 서브넷, 라우팅 테이블, 인터넷 게이트웨이, NAT, VPN과 같은 VPC 내 핵심 네트워킹 구성 요소와 VPC를 생성하는 데 필요한 네트워킹 권한이 필요하지 않습니다. 하지만 ELB, 보안 그룹, S3 버킷 및 노드와 같이 클러스터 내 시스템에 필요한 애플리케이션 리소스를 생성하려면 여전히 권한이 필요합니다.

4.11.5.4. 클러스터 간 격리

OpenShift Container Platform을 기존 네트워크에 배포하면 클러스터 서비스 격리가 다음과 같은 방식으로 감소합니다.

- 동일한 VPC에 여러 OpenShift Container Platform 클러스터를 설치할 수 있습니다.
- ICMP 인그레스가 전체 네트워크에서 허용됩니다.
- TCP 22 인그레스(SSH)가 전체 네트워크에 허용됩니다.
- 컨트롤 플레인 TCP 6443 인그레스(Kubernetes API)가 전체 네트워크에 허용됩니다.
- 컨트롤 플레인 TCP 22623 인그레스(MCS)가 전체 네트워크에 허용됩니다.

4.11.6. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

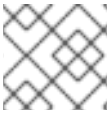
OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 **core** 사용자의 **~/.ssh/authorized_keys** 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 **core**로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다. SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. **./openshift-install gather** 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.



참고

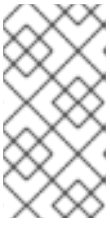
AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: **~/.ssh/id_ed25519**)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 **'~/.ssh** 디렉터리에 있는지 확인하십시오.



참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 **ed25519** 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 **rsa** 또는 **ecdsa** 알고리즘을 사용하는 키를 생성합니다.

2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 **~/.ssh/id_ed25519.pub** 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 **./openshift-install gather** 명령을 사용하려는 경우 필요합니다.

참고

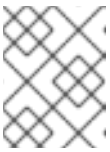
일부 배포에서는 **~/.ssh/id_rsa** 및 **~/.ssh/id_dsa**와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. **ssh-agent** 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```



참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. **ssh-agent**에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: **~/.ssh/id_ed25519**).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

4.11.7. AWS에서 사용자 지정 RHCOS AMI 업로드

사용자 지정 Amazon Web Services (AWS) 리전에 배포하는 경우 해당 리전에 속하는 사용자 지정 Red Hat Enterprise Linux CoreOS (RHCOS) Amazon 머신 이미지 (AMI)를 업로드해야 합니다.

사전 요구 사항

- AWS 계정을 구성했습니다.
- 필요한 IAM 서비스 역할로 Amazon S3 버킷을 생성했습니다.
- RHCOS VMDK 파일을 Amazon S3에 업로드했습니다. RHCOS VMDK 파일은 설치하는 OpenShift Container Platform 버전과 같거나 그 이하의 버전이어야 합니다.
- AWS CLI를 다운로드하여 컴퓨터에 설치했습니다. Install the AWS CLI Using the Bundled Installer를 참조하십시오.

프로세스

1. AWS 프로필을 환경 변수로 내보냅니다.

```
$ export AWS_PROFILE=<aws_profile> 1
```

1 **beijingadmin**과 같이 AWS 인증 정보를 보유하는 AWS 프로필 이름입니다.

2. 사용자 지정 AMI와 연결할 리전을 환경 변수로 내보냅니다.

```
$ export AWS_DEFAULT_REGION=<aws_region> 1
```

1 **cn-north-1**과 같은 AWS 리전.

3. Amazon S3에 업로드한 RHCOS 버전을 환경 변수로 내보냅니다.

```
$ export RHCOS_VERSION=<version> 1
```

1 **4.9.0** 과 같은 RHCOS VMDK 버전입니다.

4. Amazon S3 버킷 이름을 환경 변수로 내보냅니다.

```
$ export VMIMPORT_BUCKET_NAME=<s3_bucket_name>
```

5. **container.json** 파일을 만들고 RHCOS VMDK 파일을 정의합니다.

```
$ cat <<EOF > containers.json
{
  "Description": "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64",
  "Format": "vmdk",
  "UserBucket": {
    "S3Bucket": "${VMIMPORT_BUCKET_NAME}",
    "S3Key": "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64.vmdk"
  }
}
EOF
```

6. RHCOS 디스크를 Amazon EBS 스냅샷으로 가져옵니다.

```
$ aws ec2 import-snapshot --region ${AWS_DEFAULT_REGION} \
    --description "<description>" \ 1
    --disk-container "file://<file_path>/containers.json" 2
```

1 가져온 RHCOS 디스크에 대한 설명 (예: **rhcos-$ {RHCOS_VERSION} -x86_64-aws.x86_64**)입니다.

2 RHCOS 디스크를 설명하는 JSON 파일의 파일 경로입니다. JSON 파일에는 Amazon S3 버킷 이름과 키가 포함되어 있어야합니다.

7. 이미지 가져 오기 상태를 확인합니다.

```
$ watch -n 5 aws ec2 describe-import-snapshot-tasks --region ${AWS_DEFAULT_REGION}
```

출력 예

```
{
    "ImportSnapshotTasks": [
        {
            "Description": "rhcos-4.7.0-x86_64-aws.x86_64",
            "ImportTaskId": "import-snap-fh6i8uil",
            "SnapshotTaskDetail": {
                "Description": "rhcos-4.7.0-x86_64-aws.x86_64",
                "DiskImageSize": 819056640.0,
                "Format": "VMDK",
                "SnapshotId": "snap-06331325870076318",
                "Status": "completed",
                "UserBucket": {
                    "S3Bucket": "external-images",
                    "S3Key": "rhcos-4.7.0-x86_64-aws.x86_64.vmdk"
                }
            }
        }
    ]
}
```

SnapshotId를 복사하여 이미지를 등록합니다.

8. RHCOS 스냅 샷에서 사용자 지정 RHCOS AMI를 생성합니다.

```
$ aws ec2 register-image \
  --region ${AWS_DEFAULT_REGION} \
  --architecture x86_64 \ 1
  --description "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64" \ 2
  --ena-support \
  --name "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64" \ 3
  --virtualization-type hvm \
  --root-device-name '/dev/xvda' \
  --block-device-mappings 'DeviceName=/dev/xvda,Ebs=
{DeleteOnTermination=true,SnapshotId=<snapshot_ID>}' 4
```

1 **x86_64**, **s390x** 또는 **ppc64le**과 같은 RHCOS VMDK 아키텍처 유형입니다.

2 가져온 스냅샷의 **Description**입니다.

3 RHCOS AMI의 이름입니다.

4 가져온 스냅샷의 **SnapshotID**입니다.

이러한 API에 대한 자세한 내용은 importing snapshots 및 creating EBS-backed AMIs에서 참조하십시오.

4.11.8. 설치 프로그램 받기

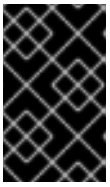
OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

프로세스

1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.
2. 인프라 공급자를 선택합니다.
3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.



중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.

중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 OpenShift Container Platform 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

4.11.9. 수동으로 설치 구성 파일 만들기

사용자 지정 Red Hat Enterprise Linux CoreOS (RHCOS) AMI가 필요한 리전에 Amazon Web Services (AWS)에서 OpenShift Container Platform을 설치할 때 설치 구성 파일을 수동으로 생성해야 합니다.

사전 요구 사항

- 사용자 정의 RHCOS AMI를 업로드했습니다.
- 로컬 시스템에 설치 프로그램에 제공할 SSH 공개 키가 있습니다. 키는 디버깅 및 재해 복구를 위해 클러스터 노드에 대한 SSH 인증에 사용됩니다.
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿이 있습니다.

프로세스

1. 필요한 설치 자산을 저장할 설치 디렉터리를 만듭니다.

```
$ mkdir <installation_directory>
```

중요

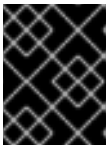
디렉터리를 만들어야 합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

2. 샘플 **install-config.yaml** 파일 템플릿을 사용자 지정하여 **<installation_directory>**에 저장합니다.

참고

이 설정 파일의 이름을 **install-config.yaml**로 지정해야 합니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.



중요

install-config.yaml 파일은 설치 과정의 다음 단계에서 사용됩니다. 이 시점에서 이를 백업해야 합니다.

4.11.9.1. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 매개변수 값을 제공하여 클러스터를 호스팅할 클라우드 플랫폼에서 사용자 계정을 설명하고 선택사항으로 클러스터의 플랫폼을 사용자 지정합니다. **install-config.yaml** 설치 구성 파일을 생성할 때 명령줄을 통해 필요한 매개변수 값을 제공합니다. 클러스터를 사용자 지정하면 **install-config.yaml** 파일을 수정하여 플랫폼에 대한 세부 정보를 제공할 수 있습니다.

참고

설치한 후에는 **install-config.yaml** 파일에서 이러한 매개변수를 수정할 수 없습니다.

중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

4.11.9.1.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 4.37. 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	소문자, 하이픈(**-**), 마침표(**.**)로 구성되는 문자열(예: **dev**)입니다.
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

매개변수	설명	값

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

4.11.9.1.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 4.38. 네트워크 매개변수

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체 **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.

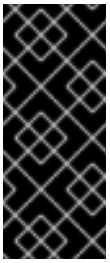
매개변수	설명	값
networking.clusterNetwork	Pod의 IP 주소 블록입니다. 기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된**cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16

매개변수	설명	값
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16**  **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

4.11.9.1.3. 선택적 구성 매개변수

선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 4.39. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**

매개변수	설명	값
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은**3**입니다.
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 기본값은 **External**입니다. 이 필드를 **Internal** 로 설정하는 것은 클라우드 이외의 플랫폼에서는 지원되지 않습니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

4.11.9.2. AWS용 샘플 사용자 지정 install-config.yaml 파일

설치 구성 파일(**install-config.yaml**)을 사용자 지정하여 OpenShift Container Platform 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.

중요

이 샘플 YAML 파일은 참조용으로만 제공됩니다. 수동으로 생성한 설치 구성 파일에 매개변수 값을 입력하기 위한 리소스로 사용합니다.

```
apiVersion: v1
baseDomain: example.com (1)
credentialsMode: Mint (2)
controlPlane: (3) (4)
  hyperthreading: Enabled (5)
  name: master
  platform:
    aws:
      zones:
      - cn-north-1a
      - cn-north-1b
      rootVolume:
        iops: 4000
        size: 500
        type: io1 (6)
      type: m5.xlarge
  replicas: 3
compute: (7)
- hyperthreading: Enabled (8)
  name: worker
  platform:
    aws:
      rootVolume:
        iops: 2000
        size: 500
        type: io1 (9)
```

```
      type: c5.4xlarge
      zones:
      - cn-north-1a
  replicas: 3
metadata:
  name: test-cluster (10)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  aws:
    region: cn-north-1 (11)
    userTags:
      adminContact: jdoe
      costCenter: 7536
    subnets: (12)
    - subnet-1
    - subnet-2
    - subnet-3
    amiID: ami-96c6f8f7 (13) (14)
    serviceEndpoints: (15)
      - name: ec2
        url: https://vpce-id.ec2.cn-north-1.vpce.amazonaws.com.cn
    hostedZone: Z3URY6TWQ91KVV (16)
fips: false (17)
sshKey: ssh-ed25519 AAAA... (18)
publish: Internal (19)
pullSecret: '{"auths": ...}' (20)
```

(1) (10) (11) (13) (20) 필수 항목입니다.

(2) 옵션: 이 매개 변수를 추가하여 Cloud Credential Operator (CCO)가 인증 정보의 기능을 동적으로 판별하도록 하지 않고 CCO가 지정된 모드를 사용하도록 강제합니다. CCO 모드에 대한 자세한 내용은 *Platform Operators 참조* 콘텐츠의 *Cloud Credential Operator* 항목을 참조하십시오.

(3) (7) 이러한 매개변수와 값을 지정하지 않으면 설치 프로그램은 기본값을 적용합니다.

(4) **controlPlane** 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

(5) (8) 동시 멀티스레딩 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. 일부 클러스터 시스템에서 동시 멀티스레딩을 비활성화할 경우에는 해당 멀티스레딩을 모든 클러스터 시스템에서 비활성화해야 합니다.

중요

동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다. 동시 멀티스레딩을 비활성화하는 경우 시스템에 더 큰 인스턴스 유형(예: **m4.2xlarge** 또는 **m5.2xlarge**)을 사용합니다.

6 9 대규모의 클러스터의 경우 고속 etcd 스토리지를 구성하려면 스토리지 유형을 **IO1**로 설정하고 **iops**는 **2000**으로 설정합니다.

12 자체 VPC를 제공하는 경우 클러스터가 사용하는 각 가용성 영역의 서브넷을 지정합니다.

14 클러스터 머신을 시작하는 데 사용되는 AMI의 ID입니다. 설정된 경우 AMI는 클러스터와 동일한 리전에 속해 있어야 합니다.

15 AWS 서비스 엔드 포인트입니다. 알 수 없는 AWS 리전에 설치할 때 사용자 지정 엔드 포인트가 필요합니다. 엔드포인트 URL은 **https** 프로토콜을 사용해야하며 호스트는 인증서를 신뢰해야 합니다.

16 기존 Route 53 개인 호스팅 영역의 ID입니다. 기존 호스팅 영역을 제공하려면 클러스터를 설치하기 전에 자체 VPC를 제공하고 호스팅 영역이 VPC와 연결되어 있어야 합니다. 정의되지 않은 경우 설치 프로그램은 새 호스팅 영역을 생성합니다.

17 FIPS 모드 활성화 또는 비활성화 여부입니다. 기본적으로 FIPS 모드는 비활성화됩니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.

중요

FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.

18 선택사항으로, 클러스터의 시스템에 액세스하는 데 사용할 **sshKey** 값을 제공할 수도 있습니다.

참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

19 클러스터의 사용자 끝점을 게시하는 방법. 인터넷에서 액세스할 수 없는 프라이빗 클러스터를 배포하려면 **publish**를 **Internal**로 설정합니다. 기본값은 **External**입니다.

4.11.9.3. 클러스터 설치를 위한 최소 리소스 요구 사항

각 클러스터 시스템이 다음과 같은 최소 요구사항을 충족해야 합니다.

표 4.40. 최소 리소스 요구사항

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
부트스트랩	RHCOS	4	16GB	100GB	300
컨트롤 플레인	RHCOS	4	16GB	100GB	300

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
컴퓨팅	RHCOS, RHEL 7.9 또는 RHEL 8.4 [3]	2	8GB	100GB	300

1. SMT(동시 멀티스레딩) 또는 하이퍼 스레딩이 활성화되지 않은 경우 하나의 vCPU는 하나의 물리적 코어와 동일합니다. 활성화하면 다음과 같은 공식을 사용하여 해당 비율을 계산합니다. (코어당 스레드 수 × 코어 수) × 소켓 수 = vCPU 수

2. OpenShift Container Platform 및 Kubernetes는 디스크 성능에 민감하며 특히 10ms p99 fsync 기간이 필요한 컨트롤 플레인 노드의 etcd에 더 빠른 스토리지가 권장됩니다. 많은 클라우드 플랫폼에서 스토리지 크기와 IOPS를 함께 확장되므로 충분한 성능을 얻으려면 스토리지 볼륨을 과도하게 할당해야 할 수 있습니다.

3. 사용자가 프로비저닝한 모든 설치와 마찬가지로 클러스터에서 RHEL 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. RHEL 7 컴퓨팅 머신 사용은 더 이상 사용되지 않으며 향후 OpenShift Container Platform 4 릴리스에서 제거될 예정입니다.

4.11.9.4. 지원되는 AWS 머신 유형

다음 AWS(Amazon Web Services) 인스턴스 유형은 OpenShift Container Platform에서 지원됩니다.

예 4.21. 시스템의 인스턴스 유형

인스턴스 유형	부트스트랩	컨트롤 플레인	컴퓨팅
i3.large	x		
m4.large			x
m4.xlarge		x	x
m4.2xlarge		x	x
m4.4xlarge		x	x
m4.10xlarge		x	x
m4.16xlarge		x	x
m5.large			x
m5.xlarge		x	x
m5.2xlarge		x	x

인스턴스 유형	부트스트랩	컨트롤 플레인	컴퓨팅
m5.4xlarge		x	x
m5.8xlarge		x	x
m5.12xlarge		x	x
m5.16xlarge		x	x
m5a.large			x
m5a.xlarge		x	x
m5a.2xlarge		x	x
m5a.4xlarge		x	x
m5a.8xlarge		x	x
m5a.12xlarge		x	x
m5a.16xlarge		x	x
m6i.xlarge		x	x
m6i.2xlarge		x	x
m6i.4xlarge		x	x
m6i.8xlarge		x	x
m6i.16xlarge		x	x
c4.2xlarge		x	x
c4.4xlarge		x	x
c4.8xlarge		x	x
c5.xlarge			x
c5.2xlarge		x	x
c5.4xlarge		x	x
c5.9xlarge		x	x

인스턴스 유형	부트스트랩	컨트롤 플레인	컴퓨팅
c5.12xlarge		x	x
c5.18xlarge		x	x
c5.24xlarge		x	x
c5a.xlarge			x
c5a.2xlarge		x	x
c5a.4xlarge		x	x
c5a.8xlarge		x	x
c5a.12xlarge		x	x
c5a.16xlarge		x	x
c5a.24xlarge		x	x
r4.large			x
r4.xlarge		x	x
r4.2xlarge		x	x
r4.4xlarge		x	x
r4.8xlarge		x	x
r4.16xlarge		x	x
r5.large			x
r5.xlarge		x	x
r5.2xlarge		x	x
r5.4xlarge		x	x
r5.8xlarge		x	x
r5.12xlarge		x	x
r5.16xlarge		x	x

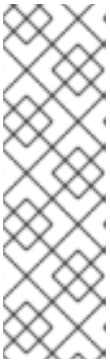
인스턴스 유형	부트스트랩	컨트롤 플레인	컴퓨팅
r5.24xlarge		x	x
r5a.large			x
r5a.xlarge		x	x
r5a.2xlarge		x	x
r5a.4xlarge		x	x
r5a.8xlarge		x	x
r5a.12xlarge		x	x
r5a.16xlarge		x	x
r5a.24xlarge		x	x
t3.large			x
t3.xlarge			x
t3.2xlarge			x
t3a.large			x
t3a.xlarge			x
t3a.2xlarge			x

4.11.9.5. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 HTTP 또는 HTTPS 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 OpenShift Container Platform 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 API에 대한 호출을 포함하여 모든 클러스터 발신(Egress) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.



참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 Red Hat OpenStack Platform (RHOSP)에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

- **ec2.<region>.amazonaws.com.cn,elasticloadbalancing.<region>.amazonaws.com**, **s3.<region>.amazonaws.com** 끝점을 VPC 엔드포인트에 추가했습니다. 이러한 끝점은 노드에서 AWS EC2 API로 요청을 완료하는 데 필요합니다. 프록시는 노드 수준이 아닌 컨테이너 수준에서 작동하므로 이러한 요청을 AWS 사설 네트워크를 통해 AWS EC2 API로 라우팅해야 합니다. 프록시 서버의 허용 목록에 EC2 API의 공용 IP 주소를 추가하는 것만으로는 충분하지 않습니다.

절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. 클러스터 외부에서 HTTP 연결을 구축하는 데 사용할 프록시 URL입니다. URL 스키마는 **http**여야 합니다.
2. 클러스터 외부에서 HTTPS 연결을 구축하는 데 사용할 프록시 URL입니다.
3. 대상 도메인 이름, IP 주소 또는 프록시에서 제외할 기타 네트워크 CIDR로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. *를 사용하여 모든 대상에 대해 프록시를 바이패스합니다.
4. 이 값을 제공하면 설치 프로그램에서 추가 CA 인증서를 보유할 **openshift -config 네임스페이스에 user-ca- bundle** 이라는 구성 맵을 생성합니다. **additionalTrustBundle** 및 하나 이상의 프록시 설정을 제공하는 경우 **프록시** 오브젝트는 **trustedCA** 필드의 **user-ca-bundle** 구성 맵을 참조하도록 구성됩니다. 그러면 Cluster Network Operator에서 **trustedCA** 매개변수에 대해 지정된 콘텐츠를 RHCOS 신뢰 번들과 병합하는 **trusted-ca-bundle** 구성 맵을 생성합니다. 프록시의 ID 인증서를 RHCOS 트러스트 번들에 있는 기관에서 서명하지 않은 경우 **additionalTrustBundle** 필드가 있어야 합니다.

참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 OpenShift Container Platform을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 nil이 됩니다.

참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

4.11.10. 클러스터 배포

호환되는 클라우드 플랫폼에 OpenShift Container Platform을 설치할 수 있습니다.

중요

최초 설치 과정에서 설치 프로그램의 **create cluster** 명령을 한 번만 실행할 수 있습니다.

사전 요구 사항

- 클러스터를 호스팅하는 클라우드 플랫폼으로 계정을 구성합니다.
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

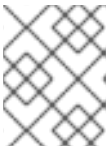
프로세스

1. 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터 배포를 초기화합니다.

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** 값으로 사용자 지정한 **./install-config.yaml** 파일의 위치를 지정합니다.

2 다른 설치 세부 사항을 보려면 **info** 대신 **warn**, **debug** 또는 **error**를 지정합니다.



참고

호스트에 구성된 클라우드 공급자 계정에 클러스터를 배포하기에 충분한 권한이 없는 경우, 설치 프로세스가 중단되고 누락된 권한을 알리는 메시지가 표시됩니다.

클러스터 배포가 완료되면 웹 콘솔로 연결되는 링크와 **kubeadmin** 사용자의 인증 정보가 포함된 클러스터 액세스 지침이 사용자 터미널에 표시됩니다.

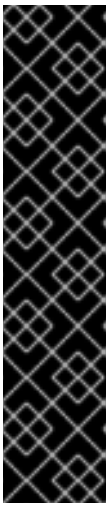
출력 예

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
```

```
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-Wt5AL"
INFO Time elapsed: 36m22s
```

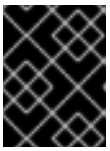
참고

설치에 성공하면 클러스터 액세스 및 인증 정보도 **<installation_directory>/.openshift_install.log**로 출력됩니다.



중요

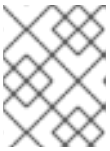
- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.



중요

설치 프로그램에서 생성되는 파일이나 설치 프로그램을 삭제해서는 안 됩니다. 클러스터를 삭제하려면 두 가지가 모두 필요합니다.

2. 선택사항: 클러스터를 설치하는 데 사용한 IAM 계정에서 **AdministratorAccess** 정책을 제거하거나 비활성화합니다.



참고

AdministratorAccess 정책에서 제공하는 승격된 권한은 설치 중에만 필요합니다.

4.11.11. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift CLI(**oc**)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 **oc**를 설치할 수 있습니다.

중요

이전 버전의 **oc**를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 **oc**를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 Linux에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.

2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Linux Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. **oc** 바이너리를 **PATH**에 있는 디렉터리에 배치합니다.
PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치
다음 절차에 따라 Windows에 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Windows Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. ZIP 프로그램으로 아카이브의 압축을 풉니다.
5. **oc** 바이너리를 **PATH**에 있는 디렉터리로 이동합니다.
PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치
다음 절차에 따라 macOS에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 MacOSX Client** 항목 옆에 있는 **지금 다운로드를** 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 해제하고 압축을 풉니다.
5. **oc** 바이너리 PATH의 디렉터리로 이동합니다.
PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

4.11.12. CLI를 사용하여 클러스터에 로그인

클러스터 **kubeconfig** 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. **kubeconfig** 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- **oc** CLI를 설치했습니다.

프로세스

1. **kubeadmin** 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 **oc** 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

4.11.13. 웹 콘솔을 사용하여 클러스터에 로그인

kubeadmin 사용자는 OpenShift Container Platform 설치 후 기본적으로 존재합니다. OpenShift Container Platform 웹 콘솔을 사용하여 **kubeadmin** 사용자로 클러스터에 로그인할 수 있습니다.

사전 요구 사항

- 설치 호스트에 대한 액세스 권한이 있어야 합니다.
- 클러스터 설치를 완료했으며 모든 클러스터 Operator를 사용할 수 있습니다.

프로세스

1. 설치 호스트의 **kubeadmin-password** 파일에서 **kubeadmin** 사용자의 암호를 가져옵니다.

```
$ cat <installation_directory>/auth/kubeadmin-password
```

참고

대안으로 설치 호스트의 **<installation_directory>/.openshift_install.log** 로그 파일에서 **kubeadmin** 암호를 가져올 수 있습니다.

2. OpenShift Container Platform 웹 콘솔 경로를 나열합니다.

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```

참고

대안으로 설치 호스트의 **<installation_directory>/.openshift_install.log** 로그 파일에서 OpenShift Container Platform 경로를 가져올 수 있습니다.

출력 예

```
console     console-openshift-console.apps.<cluster_name>.<base_domain>           console
https   reencrypt/Redirect   None
```

3. 웹 브라우저의 이전 명령 출력에 자세히 설명된 경로로 이동하고 **kubeadmin** 사용자로 로그인합니다.

4.11.14. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- OpenShift Container Platform 웹 콘솔 액세스 및 이해에 대한 자세한 내용은 웹 콘솔에 액세스를 참조하십시오.
- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

4.11.15. 다음 단계

- 설치를 확인합니다.
- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.
- 필요한 경우 클라우드 공급자 인증 정보를 제거 할 수 있습니다.

4.12. CLOUDFORMATION 템플릿을 사용하여 사용자 프로비저닝 인프라에 클러스터 설치

OpenShift Container Platform 4.9 버전에서는 사용자가 제공하는 인프라를 사용하는 클러스터를 AWS(Amazon Web Services)에 설치할 수 있습니다.

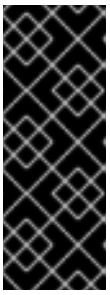
이 인프라를 생성하는 한 가지 방법은 제공된 CloudFormation 템플릿을 사용하는 것입니다. 템플릿을 수정하여 인프라를 사용자 지정하거나 포함된 정보를 사용하여 회사 정책에 따라 AWS 개체를 생성할 수 있습니다.

중요

사용자가 프로비저닝한 인프라 설치를 수행하는 단계는 예시용으로만 제공됩니다. 사용자가 제공하는 인프라를 사용하여 클러스터를 설치하려면 클라우드 공급자 및 OpenShift Container Platform 설치 프로세스에 대한 정보가 필요합니다. 안내된 단계를 수행하거나 자체 모델링에 유용한 몇 가지 CloudFormation 템플릿이 제공됩니다. 여러 다른 방법을 통해 필요한 리소스를 자유롭게 생성할 수도 있습니다. 템플릿은 예시일 뿐입니다.

4.12.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비 에 대한 문서를 읽습니다.
- 클러스터를 호스팅할 AWS 계정을 구성했습니다.



중요

컴퓨터에 AWS 프로필이 저장되어 있는 경우 다단계 인증 장치를 사용하는 동안 생성한 임시 세션 토큰을 해당 프로필이 사용해서는 안 됩니다. 클러스터는 클러스터의 전체 수명 동안 현재 AWS 자격 증명을 계속 사용하여 AWS 리소스를 생성하므로 키 기반의 장기 자격 증명을 사용해야 합니다. 적절한 키를 생성하려면 AWS 문서의 IAM 사용자의 액세스 키 관리 를 참조하십시오. 설치 프로그램을 실행할 때 키를 제공할 수 있습니다.

- AWS CLI를 다운로드하여 컴퓨터에 설치했습니다. AWS 문서의 번들 설치 관리자(Linux, macOS 또는 UNIX)를 사용하여 AWS CLI 설치를 참조하십시오.
- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성 해야 합니다.

참고

프록시를 구성하는 경우에도 해당 사이트 목록을 검토하십시오.

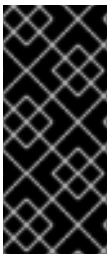
- 사용자의 환경에서 클라우드 ID 및 액세스 관리(IAM) API에 액세스할 수 없거나 **kube-system** 네임스페이스에 관리자 수준의 인증 정보 시크릿을 저장하지 않으려면 IAM 인증 정보 수동 생성 및 유지 관리를 참조하십시오.

4.12.2. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.



중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

4.12.3. 사용자 프로비저닝 인프라를 포함한 클러스터의 시스템 요구사항

사용자 프로비저닝 인프라가 포함된 클러스터의 경우, 필요한 모든 시스템을 배포해야 합니다.

이 섹션에서는 사용자 프로비저닝 인프라에 OpenShift Container Platform을 배포해야 하는 요구 사항에 대해 설명합니다.

4.12.3.1. 클러스터 설치에 필요한 시스템

최소 OpenShift Container Platform 클러스터에 다음과 같은 호스트가 필요합니다.

표 4.41. 최소 필수 호스트

호스트	설명
임시 부트스트랩 시스템 한 개	컨트롤 플레인 시스템 세 개에 OpenShift Container Platform 클러스터를 배포하기 위한 부트스트랩 시스템이 클러스터에 필요합니다. 클러스터를 설치한 후 부트스트랩 시스템을 제거할 수 있습니다.
컨트롤 플레인 시스템 세 개	컨트롤 플레인 시스템은 컨트롤 플레인을 구성하는 Kubernetes 및 OpenShift Container Platform 서비스를 실행합니다.
두 개 이상의 컴퓨팅 시스템(작업자 시스템이라고도 함).	OpenShift Container Platform 사용자가 요청한 워크로드는 컴퓨팅 머신에서 실행됩니다.

중요

클러스터의 고가용성을 유지하려면 이러한 클러스터 시스템에 대해 별도의 물리적 호스트를 사용하십시오.

부트스트랩, 컨트롤 플레인 시스템은 운영 체제로 RHCOS (Red Hat Enterprise Linux CoreOS)를 사용해야 합니다. 그러나 컴퓨팅 머신은 RHCOS(Red Hat Enterprise Linux CoreOS), RHEL (Red Hat Enterprise Linux) 7.9 또는 RHEL 8.4 중에서 선택할 수 있습니다.

RHCOS는 RHEL 8(Red Hat Enterprise Linux)을 기반으로하며 모든 하드웨어 인증 및 요구사항을 모두 이어받습니다. Red Hat Enterprise Linux 기술 기능 및 제한 을 참조하십시오.

4.12.3.2. 클러스터 설치를 위한 최소 리소스 요구 사항

각 클러스터 시스템이 다음과 같은 최소 요구사항을 충족해야 합니다.

표 4.42. 최소 리소스 요구사항

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
부트스트랩	RHCOS	4	16GB	100GB	300
컨트롤 플레인	RHCOS	4	16GB	100GB	300
컴퓨팅	RHCOS, RHEL 7.9 또는 RHEL 8.4 [3]	2	8GB	100GB	300

1. SMT(동시 멀티스레딩) 또는 하이퍼 스레딩이 활성화되지 않은 경우 하나의 vCPU는 하나의 물리적 코어와 동일합니다. 활성화하면 다음과 같은 공식을 사용하여 해당 비율을 계산합니다. (코어당 스레드 수 × 코어 수) × 소켓 수 = vCPU 수

2. OpenShift Container Platform 및 Kubernetes는 디스크 성능에 민감하며 특히 10ms p99 fsync 기간이 필요한 컨트롤 플레인 노드의 etcd에 더 빠른 스토리지가 권장됩니다. 많은 클라우드 플랫폼에서 스토리지 크기와 IOPS를 함께 확장되므로 충분한 성능을 얻으려면 스토리지 볼륨을 과도하게 할당해야 할 수 있습니다.

3. 사용자가 프로비저닝한 모든 설치와 마찬가지로 클러스터에서 RHEL 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. RHEL 7 컴퓨팅 머신 사용은 더 이상 사용되지 않으며 향후 OpenShift Container Platform 4 릴리스에서 제거될 예정입니다.

4.12.3.3. 인증서 서명 요청 관리

사용자가 프로비저닝하는 인프라를 사용하는 경우 자동 시스템 관리 기능으로 인해 클러스터의 액세스가 제한되므로 설치한 후 클러스터 인증서 서명 요청(CSR)을 승인하는 메커니즘을 제공해야 합니다. **kube-controller-manager**는 kubelet 클라이언트 CSR만 승인합니다. **machine-approver**는 올바른 시스템에서 발행한 요청인지 확인할 수 없기 때문에 kubelet 자격 증명을 사용하여 요청하는 서비스 인증서의 유효성을 보장할 수 없습니다. kubelet 서빙 인증서 요청의 유효성을 확인하고 요청을 승인하는 방법을 결정하여 구현해야 합니다.

4.12.3.4. 지원되는 AWS 머신 유형

다음 AWS(Amazon Web Services) 인스턴스 유형은 OpenShift Container Platform에서 지원됩니다.

예 4.22. 시스템의 인스턴스 유형

인스턴스 유형	부트스트랩	컨트롤 플레인	컴퓨팅
i3.large	x		

인스턴스 유형	부트스트랩	컨트롤 플레인	컴퓨팅
m4.large			x
m4.xlarge		x	x
m4.2xlarge		x	x
m4.4xlarge		x	x
m4.10xlarge		x	x
m4.16xlarge		x	x
m5.large			x
m5.xlarge		x	x
m5.2xlarge		x	x
m5.4xlarge		x	x
m5.8xlarge		x	x
m5.12xlarge		x	x
m5.16xlarge		x	x
m5a.large			x
m5a.xlarge		x	x
m5a.2xlarge		x	x
m5a.4xlarge		x	x
m5a.8xlarge		x	x
m5a.12xlarge		x	x
m5a.16xlarge		x	x
m6i.xlarge		x	x
m6i.2xlarge		x	x

인스턴스 유형	부트스트랩	컨트롤 플레인	컴퓨팅
m6i.4xlarge		x	x
m6i.8xlarge		x	x
m6i.16xlarge		x	x
c4.2xlarge		x	x
c4.4xlarge		x	x
c4.8xlarge		x	x
c5.xlarge			x
c5.2xlarge		x	x
c5.4xlarge		x	x
c5.9xlarge		x	x
c5.12xlarge		x	x
c5.18xlarge		x	x
c5.24xlarge		x	x
c5a.xlarge			x
c5a.2xlarge		x	x
c5a.4xlarge		x	x
c5a.8xlarge		x	x
c5a.12xlarge		x	x
c5a.16xlarge		x	x
c5a.24xlarge		x	x
r4.large			x
r4.xlarge		x	x

인스턴스 유형	부트스트랩	컨트롤 플레인	컴퓨팅
r4.2xlarge		x	x
r4.4xlarge		x	x
r4.8xlarge		x	x
r4.16xlarge		x	x
r5.large			x
r5.xlarge		x	x
r5.2xlarge		x	x
r5.4xlarge		x	x
r5.8xlarge		x	x
r5.12xlarge		x	x
r5.16xlarge		x	x
r5.24xlarge		x	x
r5a.large			x
r5a.xlarge		x	x
r5a.2xlarge		x	x
r5a.4xlarge		x	x
r5a.8xlarge		x	x
r5a.12xlarge		x	x
r5a.16xlarge		x	x
r5a.24xlarge		x	x
t3.large			x
t3.xlarge			x

인스턴스 유형	부트스트랩	컨트롤 플레인	컴퓨팅
t3.2xlarge			x
t3a.large			x
t3a.xlarge			x
t3a.2xlarge			x

4.12.4. 필수 AWS 인프라 구성 요소

AWS(Amazon Web Services)의 사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치하려면 시스템과 지원 인프라를 모두 수동으로 생성해야 합니다.

다른 플랫폼의 통합 테스트에 대한 자세한 내용은 OpenShift Container Platform 4.x 통합 테스트 페이지를 참조하십시오.

제공된 CloudFormation 템플릿을 사용하면 다음 구성 요소를 나타내는 AWS 리소스 스택을 생성할 수 있습니다.

- AWS 가상 사설 클라우드(VPC)
- 네트워킹 및 로드 밸런싱 구성 요소
- 보안 그룹 및 역할
- OpenShift Container Platform 부트스트랩 노드
- OpenShift Container Platform 컨트롤 플레인 노드
- OpenShift Container Platform 컴퓨팅 노드

대안으로, 구성 요소를 수동으로 생성하거나 클러스터 요구 사항을 충족하는 기존 인프라를 재사용할 수 있습니다. 구성 요소의 상호 관계에 대한 자세한 내용은 CloudFormation 템플릿을 검토하십시오.

4.12.4.1. 기타 인프라 구성 요소

- A VPC
- DNS 항목
- 로드 밸런서(클래식 또는 네트워크) 및 리스너
- 공개 및 개인 Route 53 영역
- 보안 그룹
- IAM 역할
- S3 버킷

연결이 끊긴 환경에서 작업하거나 프록시를 사용하는 경우 EC2 및 ELB 끝점의 공용 IP 주소에 연결할 수 없습니다. 이러한 끝점에 연결하려면 VPC 끝점을 생성하여 클러스터가 사용 중인 서브넷에 연결해야 합니다. 다음 끝점을 생성합니다.

- **ec2.<region>.amazonaws.com**
- **elasticloadbalancing.<region>.amazonaws.com**
- **s3.<region>.amazonaws.com**

필수 VPC 구성 요소

시스템과의 통신을 허용하는 서브넷과 적합한 VPC를 제공해야 합니다.

<table>
<tr><th>구성 요소</th><th>AWS 유형</th><th colspan="2">설명</th></tr>
<tr><td>VPC</td><td>AWS::EC2::VPC
AWS::EC2::VPCEndpoint</td><td colspan="2">클러스터에서 사용할 공용 VPC를 제공해야 합니다. VPC는 각 서브넷의 라우팅 테이블을 참조하는 끝점을 사용하여 S3에서 호스팅되는 레지스트리와의 통신을 개선합니다.</td></tr>
<tr><td>퍼블릭 서브넷</td><td>AWS::EC2::Subnet
AWS::EC2::SubnetNetworkAclAssociation</td><td colspan="2">VPC에는 1~3개의 가용성 영역에 대한 퍼블릭 서브넷이 있어야 하며 이 서브넷을 적절한 인그레스 규칙과 연결해야 합니다.</td></tr>
<tr><td>인터넷 게이트웨이</td><td>AWS::EC2::InternetGateway
AWS::EC2::VPCGatewayAttachment
AWS::EC2::RouteTable
AWS::EC2::Route
AWS::EC2::SubnetRouteTableAssociation
AWS::EC2::NatGateway
AWS::EC2::EIP</td><td colspan="2">공용 경로가 있는 공용 인터넷 게이트웨이가 VPC에 연결되어 있어야 합니다. 제공된 템플릿에서 각 퍼블릭 서브넷에는 EIP 주소를 갖는 NAT 게이트웨이가 있습니다. 이러한 NAT 게이트웨이를 사용하면 프라이빗 서브넷 인스턴스와 같은 클러스터 리소스가 인터넷에 도달할 수 있으므로 일부 제한된 네트워크 또는 프록시 시나리오에는 필요하지 않습니다.</td></tr>
<tr><td>네트워크 액세스 제어</td><td>AWS::EC2::NetworkAcl
AWS::EC2::NetworkAclEntry</td><td colspan="2">VPC가 다음 포트에 액세스할 수 있어야 합니다.</td></tr>
<tr><td></td><td></td><td>포트</td><td>이유</td></tr>
<tr><td></td><td></td><td>80</td><td>인바운드 HTTP 트래픽</td></tr>
<tr><td></td><td></td><td>443</td><td>인바운드 HTTPS 트래픽</td></tr>
</table>

구성 요소	AWS 유형	설명	
		22	인바운드 SSH 트래픽
		1024 - 65535	인바운드 임시 트래픽
		0 - 65535	아웃바운드 임시 트래픽
프라이빗 서브넷	• **AWS::EC2::Subnet** • **AWS::EC2::RouteTable** • **AWS::EC2::SubnetRouteTableAssociation**	VPC에 프라이빗 서브넷이 포함될 수 있습니다. 제공된 CloudFormation 템플릿은 1~3개 가용성 영역의 프라이빗 서브넷을 생성할 수 있습니다. 프라이빗 서브넷을 사용하는 경우 적절한 경로와 테이블을 제공해야 합니다.	

필수 DNS 및 로드 밸런싱 구성 요소

DNS 및 로드 밸런서 구성은 공개 호스팅 영역을 사용해야 하며 클러스터 인프라를 프로비저닝하는 경우 설치 프로그램이 사용하는 것과 유사한 개인 호스팅 영역을 사용할 수 있습니다. 로드 밸런서로 확인되는 DNS 항목을 생성해야 합니다. **api.<cluster_name>.<domain>**에 대한 항목은 외부 로드 밸런서를 가리켜야 하고 **api-int.<cluster_name>.<domain>**에 대한 항목은 내부 로드 밸런서를 가리켜야 합니다.

클러스터에는 또한 Kubernetes API 및 해당 확장에 필요한 포트 6443 및 새 시스템의 Ignition 구성 파일에 필요한 포트 22623용 로드 밸런서와 리스너가 필요합니다. 대상은 컨트롤 플레인 노드가 됩니다. 포트 6443은 클러스터 외부의 클라이언트와 클러스터 내부의 노드에서 모두 액세스할 수 있어야 합니다. 포트 22623은 클러스터 내부 노드에서 액세스할 수 있어야 합니다.

구성 요소	AWS 유형	설명
DNS	**AWS::Route53::HostedZone**	내부 DNS의 호스팅 영역입니다.
etcd 레코드 세트	**AWS::Route53::RecordSet**	컨트롤 플레인 시스템의 etcd 등록 레코드입니다.
공용 로드 밸런서	**AWS::ElasticLoadBalancingV2::LoadBalancer**	퍼블릭 서브넷의 로드 밸런서입니다.
외부 API 서버 레코드	**AWS::Route53::RecordSetGroup**	외부 API 서버의 별칭 레코드입니다.

구성 요소	AWS 유형	설명
외부 리스너	**AWS::ElasticLoadBalancingV2::Listener**	외부 로드 밸런서용 포트 6443의 리스너입니다.
외부 대상 그룹	**AWS::ElasticLoadBalancingV2::TargetGroup**	외부 로드 밸런서의 대상 그룹입니다.
프라이빗 로드 밸런서	**AWS::ElasticLoadBalancingV2::LoadBalancer**	프라이빗 서브넷의 로드 밸런서입니다.
내부 API 서버 레코드	**AWS::Route53::RecordSetGroup**	내부 API 서버의 별칭 레코드입니다.
내부 리스너	**AWS::ElasticLoadBalancingV2::Listener**	내부 로드 밸런서용 포트 22623의 리스너입니다.
내부 대상 그룹	**AWS::ElasticLoadBalancingV2::TargetGroup**	내부 로드 밸런서의 대상 그룹입니다.
내부 리스너	**AWS::ElasticLoadBalancingV2::Listener**	내부 로드 밸런서용 포트 6443의 리스너입니다.
내부 대상 그룹	**AWS::ElasticLoadBalancingV2::TargetGroup**	내부 로드 밸런서의 대상 그룹입니다.

보안 그룹

컨트롤 플레인 및 작업자 시스템에는 다음 포트에 대한 액세스 권한이 필요합니다.

그룹	유형	IP 프로토콜	포트 범위
MasterSecurityGroup	**AWS::EC2::SecurityGroup**	**icmp**	**0**

그룹	유형	IP 프로토콜	포트 범위
		tcp	22
		tcp	6443
		tcp	22623
WorkerSecurityGroup	AWS::EC2::SecurityGroup	icmp	0
		tcp	22
BootstrapSecurityGroup	AWS::EC2::SecurityGroup	tcp	22
		tcp	19531

컨트롤 플레인 인그레스

컨트롤 플레인 시스템에는 다음과 같은 인그레스 그룹이 필요합니다. 각 인그레스 그룹은 **AWS::EC2::SecurityGroupIngress** 리소스입니다.

인그레스 그룹	설명	IP 프로토콜	포트 범위
MasterIngressEtcd	etcd	tcp	2379- 2380
MasterIngressVxlan	Vxlan 패킷	udp	4789
MasterIngressWorkerVxlan	Vxlan 패킷	udp	4789
MasterIngressInternal	내부 클러스터 통신 및 Kubernetes 프록시 메트릭	tcp	9000 - 9999
MasterIngressWorkerInternal	내부 클러스터 통신	tcp	9000 - 9999
MasterIngressKube	Kubernetes kubelet, 스케줄러 및 컨트롤러 관리자	tcp	10250 - 10259
MasterIngressWorkerKube	Kubernetes kubelet, 스케줄러 및 컨트롤러 관리자	tcp	10250 - 10259

인그레스 그룹	설명	IP 프로토콜	포트 범위
MasterIngressIngressServices	Kubernetes 인그레스 서비스	**tcp**	**30000 - 32767**
MasterIngressWorkerIngressServices	Kubernetes 인그레스 서비스	**tcp**	**30000 - 32767**
MasterIngressGeneve	Geneve 패킷	**udp**	**6081**
MasterIngressWorkerGeneve	Geneve 패킷	**udp**	**6081**
MasterIngressIpsecIke	IPsec IKE 패킷	**udp**	**500**
MasterIngressWorkerIpsecIke	IPsec IKE 패킷	**udp**	**500**
MasterIngressIpsecNat	IPsec NAT-T 패킷	**udp**	**4500**
MasterIngressWorkerIpsecNat	IPsec NAT-T 패킷	**udp**	**4500**
MasterIngressIpsecEsp	IPsec ESP 패킷	**50**	모두
MasterIngressWorkerIpsecEsp	IPsec ESP 패킷	**50**	모두
MasterIngressInternalUDP	내부 클러스터 통신	**udp**	**9000 - 9999**
MasterIngressWorkerInternalUDP	내부 클러스터 통신	**udp**	**9000 - 9999**
MasterIngressIngressServicesUDP	Kubernetes 인그레스 서비스	**udp**	**30000 - 32767**

인그레스 그룹	설명	IP 프로토콜	포트 범위
MasterIngressWorkerIngressServicesUDP	Kubernetes 인그레스 서비스	**udp**	**30000 - 32767**

작업자 인그레스

작업자 시스템에는 다음과 같은 인그레스 그룹이 필요합니다. 각 인그레스 그룹은 **AWS::EC2::SecurityGroupIngress** 리소스입니다.

인그레스 그룹	설명	IP 프로토콜	포트 범위
WorkerIngressVxlan	Vxlan 패킷	**udp**	**4789**
WorkerIngressWorkerVxlan	Vxlan 패킷	**udp**	**4789**
WorkerIngressInternal	내부 클러스터 통신	**tcp**	**9000 - 9999**
WorkerIngressWorkerInternal	내부 클러스터 통신	**tcp**	**9000 - 9999**
WorkerIngressKube	Kubernetes kubelet, 스케줄러 및 컨트롤러 관리자	**tcp**	**10250**
WorkerIngressWorkerKube	Kubernetes kubelet, 스케줄러 및 컨트롤러 관리자	**tcp**	**10250**
WorkerIngressIngressServices	Kubernetes 인그레스 서비스	**tcp**	**30000 - 32767**
WorkerIngressWorkerIngressServices	Kubernetes 인그레스 서비스	**tcp**	**30000 - 32767**
WorkerIngressGeneve	Geneve 패킷	**udp**	**6081**
WorkerIngressMasterGeneve	Geneve 패킷	**udp**	**6081**
WorkerIngressIpsecIke	IPsec IKE 패킷	**udp**	**500**

인그레스 그룹	설명	IP 프로토콜	포트 범위
WorkerIngressMasterIpsecIke	IPsec IKE 패킷	**udp**	**500**
WorkerIngressIpsecNat	IPsec NAT-T 패킷	**udp**	**4500**
WorkerIngressMasterIpsecNat	IPsec NAT-T 패킷	**udp**	**4500**
WorkerIngressIpsecEsp	IPsec ESP 패킷	**50**	**모두**
WorkerIngressMasterIpsecEsp	IPsec ESP 패킷	**50**	**모두**
WorkerIngressInternalUDP	내부 클러스터 통신	**udp**	**9000 - 9999**
WorkerIngressMasterInternalUDP	내부 클러스터 통신	**udp**	**9000 - 9999**
WorkerIngressIngressServicesUDP	Kubernetes 인그레스 서비스	**udp**	**30000 - 32767**
WorkerIngressMasterIngressServicesUDP	Kubernetes 인그레스 서비스	**udp**	**30000 - 32767**

역할 및 인스턴스 프로필

AWS에서 시스템 권한을 부여해야 합니다. 제공된 CloudFormation 템플릿은 다음 **AWS::IAM::Role** 오브젝트에 대한 **허용** 권한을 머신에 부여하며 각 역할 세트의 **AWS::IAM::InstanceProfile**을 제공합니다. 템플릿을 사용하지 않는 경우 다음과 같은 광범위한 권한 또는 다음과 같은 개별 권한을 시스템에 부여할 수 있습니다.

역할	효과	동작	리소스 이름
Master	**허용**	**ec2:***	*
	허용	**elasticloadbalancing:***	*

역할	효과	동작	리소스 이름
	허용	**iam:PassRole**	*
	허용	**s3:GetObject**	*
Worker	허용	**ec2:Describe***	*
부트스트랩	허용	**ec2:Describe***	*
	허용	**ec2:AttachVolume**	*
	허용	**ec2:DetachVolume**	*

4.12.4.2. 클러스터 시스템

다음 시스템의 **AWS:: EC2:: Instance** 개체가 필요합니다.

- 부트스트랩 시스템. 이 시스템은 설치 중에 필요하지만 클러스터가 배포된 후 제거할 수 있습니다.
- 컨트롤 플레인 시스템 세 개 컨트롤 플레인 시스템은 머신 세트에 의해 관리되지 않습니다.
- 컴퓨팅 시스템. 설치 중에 두 개 이상의 컴퓨팅 시스템(작업자 시스템이라고도 함)을 생성해야 합니다. 이러한 시스템은 머신 세트에 의해 관리되지 않습니다.

4.12.4.3. IAM 사용자에게 필요한 AWS 권한

참고

기본 클러스터 리소스를 삭제하려면 IAM 사용자에게 **us-east-1** 리전에 권한 **태그:GetResources** 가 있어야 합니다. AWS API 요구 사항의 일부로 OpenShift Container Platform 설치 프로그램은 이 리전에서 다양한 작업을 수행합니다.

AWS(Amazon Web Services)에서 생성되는 IAM 사용자에게 **AdministratorAccess** 정책을 연결하면 해당 사용자에게 필요한 모든 권한이 부여됩니다. OpenShift Container Platform 클러스터의 모든 구성 요소를 배포하려면 IAM 사용자에게 다음과 같은 권한이 필요합니다.

예 4.23. 설치에 필요한 EC2 권한

- **ec2:AuthorizeSecurityGroupEgress**
- **ec2:AuthorizeSecurityGroupIngress**
- **ec2:CopyImage**
- **ec2:CreateNetworkInterface**
- **ec2:AttachNetworkInterface**
- **ec2:CreateSecurityGroup**

- **ec2:CreateTags**
- **ec2:CreateVolume**
- **ec2:DeleteSecurityGroup**
- **ec2:DeleteSnapshot**
- **ec2:DeleteTags**
- **ec2:DeregisterImage**
- **ec2:DescribeAccountAttributes**
- **ec2:DescribeAddresses**
- **ec2:DescribeAvailabilityZones**
- **ec2:DescribeDhcpOptions**
- **ec2:DescribeImages**
- **ec2:DescribeInstanceAttribute**
- **ec2:DescribeInstanceCreditSpecifications**
- **ec2:DescribeInstances**
- **ec2:DescribeInstanceTypes**
- **ec2:DescribeInternetGateways**
- **ec2:DescribeKeyPairs**
- **ec2:DescribeNatGateways**
- **ec2:DescribeNetworkAcls**
- **ec2:DescribeNetworkInterfaces**
- **ec2:DescribePrefixLists**
- **ec2:DescribeRegions**
- **ec2:DescribeRouteTables**
- **ec2:DescribeSecurityGroups**
- **ec2:DescribeSubnets**
- **ec2:DescribeTags**
- **ec2:DescribeVolumes**
- **ec2:DescribeVpcAttribute**
- **ec2:DescribeVpcClassicLink**

- **ec2:DescribeVpcClassicLinkDnsSupport**
- **ec2:DescribeVpcEndpoints**
- **ec2:DescribeVpcs**
- **ec2:GetEbsDefaultKmsKeyId**
- **ec2:ModifyInstanceAttribute**
- **ec2:ModifyNetworkInterfaceAttribute**
- **ec2:RevokeSecurityGroupEgress**
- **ec2:RevokeSecurityGroupIngress**
- **ec2:RunInstances**
- **ec2:TerminateInstances**

예 4.24. 설치 과정에서 네트워크 리소스를 생성하는 데 필요한 권한

- **ec2:AllocateAddress**
- **ec2:AssociateAddress**
- **ec2:AssociateDhcpOptions**
- **ec2:AssociateRouteTable**
- **ec2:AttachInternetGateway**
- **ec2:CreateDhcpOptions**
- **ec2:CreateInternetGateway**
- **ec2:CreateNatGateway**
- **ec2:CreateRoute**
- **ec2:CreateRouteTable**
- **ec2:CreateSubnet**
- **ec2:CreateVpc**
- **ec2:CreateVpcEndpoint**
- **ec2:ModifySubnetAttribute**
- **ec2:ModifyVpcAttribute**

참고

기존 VPC를 사용하는 경우 네트워크 리소스를 생성하기 위해 계정에 이러한 권한이 필요하지 않습니다.

예 4.25. 설치에 필요한 Elastic Load Balancing 권한(ELB)

- **elasticloadbalancing:AddTags**
- **elasticloadbalancing:ApplySecurityGroupsToLoadBalancer**
- **elasticloadbalancing:AttachLoadBalancerToSubnets**
- **elasticloadbalancing:ConfigureHealthCheck**
- **elasticloadbalancing:CreateLoadBalancer**
- **elasticloadbalancing:CreateLoadBalancerListeners**
- **elasticloadbalancing:DeleteLoadBalancer**
- **elasticloadbalancing:DeregisterInstancesFromLoadBalancer**
- **elasticloadbalancing:DescribeInstanceHealth**
- **elasticloadbalancing:DescribeLoadBalancerAttributes**
- **elasticloadbalancing:DescribeLoadBalancers**
- **elasticloadbalancing:DescribeTags**
- **elasticloadbalancing:ModifyLoadBalancerAttributes**
- **elasticloadbalancing:RegisterInstancesWithLoadBalancer**
- **elasticloadbalancing:SetLoadBalancerPoliciesOfListener**

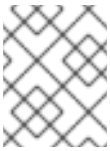
예 4.26. 설치에 필요한 Elastic Load Balancing 권한(ELBv2)

- **elasticloadbalancing:AddTags**
- **elasticloadbalancing:CreateListener**
- **elasticloadbalancing:CreateLoadBalancer**
- **elasticloadbalancing:CreateTargetGroup**
- **elasticloadbalancing:DeleteLoadBalancer**
- **elasticloadbalancing:DeregisterTargets**
- **elasticloadbalancing:DescribeListeners**
- **elasticloadbalancing:DescribeLoadBalancerAttributes**
- **elasticloadbalancing:DescribeLoadBalancers**
- **elasticloadbalancing:DescribeTargetGroupAttributes**

- **elasticloadbalancing:DescribeTargetHealth**
- **elasticloadbalancing:ModifyLoadBalancerAttributes**
- **elasticloadbalancing:ModifyTargetGroup**
- **elasticloadbalancing:ModifyTargetGroupAttributes**
- **elasticloadbalancing:RegisterTargets**

예 4.27. 설치에 필요한 IAM 권한

- **iam:AddRoleToInstanceProfile**
- **iam:CreateInstanceProfile**
- **iam:CreateRole**
- **iam:DeleteInstanceProfile**
- **iam:DeleteRole**
- **iam:DeleteRolePolicy**
- **iam:GetInstanceProfile**
- **iam:GetRole**
- **iam:GetRolePolicy**
- **iam:GetUser**
- **iam:ListInstanceProfilesForRole**
- **iam:ListRoles**
- **iam:ListUsers**
- **iam:PassRole**
- **iam:PutRolePolicy**
- **iam:RemoveRoleFromInstanceProfile**
- **iam:SimulatePrincipalPolicy**
- **iam:TagRole**



참고

AWS 계정에서 탄력적 로드 밸런서 (ELB)를 생성하지 않은 경우 IAM 사용자에게 **iam:CreateServiceLinkedRole** 권한이 필요합니다.

예 4.28. 설치에 필요한 Route 53 권한

- **route53:ChangeResourceRecordSets**
- **route53:ChangeTagsForResource**
- **route53:CreateHostedZone**
- **route53:DeleteHostedZone**
- **route53:GetChange**
- **route53:GetHostedZone**
- **route53:ListHostedZones**
- **route53:ListHostedZonesByName**
- **route53:ListResourceRecordSets**
- **route53:ListTagsForResource**
- **route53:UpdateHostedZoneComment**

예 4.29. 설치에 필요한 S3 권한

- **s3:CreateBucket**
- **s3:DeleteBucket**
- **s3:GetAccelerateConfiguration**
- **s3:GetBucketAcl**
- **s3:GetBucketCors**
- **s3:GetBucketLocation**
- **s3:GetBucketLogging**
- **s3:GetBucketObjectLockConfiguration**
- **s3:GetBucketReplication**
- **s3:GetBucketRequestPayment**
- **s3:GetBucketTagging**
- **s3:GetBucketVersioning**
- **s3:GetBucketWebsite**
- **s3:GetEncryptionConfiguration**
- **s3:GetLifecycleConfiguration**
- **s3:GetReplicationConfiguration**
- **s3:ListBucket**

- **s3:PutBucketAcl**
- **s3:PutBucketTagging**
- **s3:PutEncryptionConfiguration**

예 4.30. 클러스터 Operator에 필요한 S3 권한

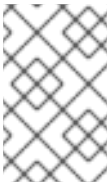
- **s3:DeleteObject**
- **s3:GetObject**
- **s3:GetObjectAcl**
- **s3:GetObjectTagging**
- **s3:GetObjectVersion**
- **s3:PutObject**
- **s3:PutObjectAcl**
- **s3:PutObjectTagging**

예 4.31. 기본 클러스터 리소스를 삭제하는 데 필요한 권한

- **autoscaling:DescribeAutoScalingGroups**
- **ec2:DeleteNetworkInterface**
- **ec2:DeleteVolume**
- **elasticloadbalancing:DeleteTargetGroup**
- **elasticloadbalancing:DescribeTargetGroups**
- **iam:DeleteAccessKey**
- **iam:DeleteUser**
- **iam:ListAttachedRolePolicies**
- **iam:ListInstanceProfiles**
- **iam:ListRolePolicies**
- **iam:ListUserPolicies**
- **s3:DeleteObject**
- **s3:ListBucketVersions**
- **tag:GetResources**

예 4.32. 네트워크 리소스를 삭제하는 데 필요한 권한

- **ec2:DeleteDhcpOptions**
- **ec2:DeleteInternetGateway**
- **ec2:DeleteNatGateway**
- **ec2:DeleteRoute**
- **ec2:DeleteRouteTable**
- **ec2:DeleteSubnet**
- **ec2:DeleteVpc**
- **ec2:DeleteVpcEndpoints**
- **ec2:DetachInternetGateway**
- **ec2:DisassociateRouteTable**
- **ec2:ReleaseAddress**
- **ec2:ReplaceRouteTableAssociation**



참고

기존 VPC를 사용하는 경우 네트워크 리소스를 삭제하기 위해 계정에 이러한 권한이 필요하지 않습니다. 대신 사용자 계정에는 네트워크 리소스를 삭제하기 위한 **tag:UntagResources** 권한만 필요합니다.

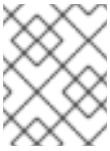
예 4.33. 공유 인스턴스 역할이 있는 클러스터를 삭제하는 데 필요한 권한

- **iam:UntagRole**

예 4.34. 매니페스트를 생성하는 데 필요한 추가 IAM 및 S3 권한

- **iam:DeleteAccessKey**
- **iam:DeleteUser**
- **iam:DeleteUserPolicy**
- **iam:GetUserPolicy**
- **iam:ListAccessKeys**
- **iam:PutUserPolicy**
- **iam:TagUser**
- **s3:PutBucketPublicAccessBlock**

- **s3:GetBucketPublicAccessBlock**
- **s3:PutLifecycleConfiguration**
- **s3:HeadBucket**
- **s3:ListBucketMultipartUploads**
- **s3:AbortMultipartUpload**



참고

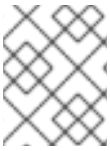
mint 모드를 사용하여 클라우드 공급자 인증 정보를 관리하는 경우 IAM 사용자에게 **iam:CreateAccessKey** 및 **iam:CreateUser** 권한이 필요합니다.

예 4.35. 인스턴스에 대한 선택적 권한 및 설치에 대한 할당량 검사

- **ec2:DescribeInstanceTypeOfferings**
- **servicequotas:ListAWSDefaultServiceQuotas**

4.12.5. AWS Marketplace 이미지 가져오기

AWS Marketplace 이미지를 사용하여 OpenShift Container Platform 클러스터를 배포하는 경우 먼저 AWS를 통해 구독해야 합니다. 이 제공을 구독하면 설치 프로그램이 작업자 노드를 배포하는 데 사용하는 AMI ID를 제공합니다.



참고

AWS Marketplace 이미지를 사용하여 OpenShift Container Platform 클러스터를 배포하는 것은 시크릿 리전 또는 중국 지역에서 지원되지 않습니다.

사전 요구 사항

- 이 제안을 구매할 수 있는 AWS 계정이 있어야 합니다. 이 계정은 클러스터를 설치하는 데 사용되는 계정과 같을 필요는 없습니다.

프로세스

1. AWS Marketplace 에서 OpenShift Container Platform 서브스크립션을 완료합니다.
2. 특정 지역에 대한 AMI ID를 기록합니다. CloudFormation 템플릿을 사용하여 작업자 노드를 배포하는 경우 이 값으로 **worker0.type.properties.ImageID** 매개변수를 업데이트해야 합니다.

4.12.6. 설치 프로그램 받기

OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

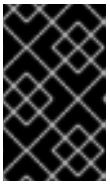
- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

프로세스

1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.
2. 인프라 공급자를 선택합니다.
3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.

중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.



중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 OpenShift Container Platform 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

4.12.7. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 **core** 사용자의 **~/.ssh/authorized_keys** 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 **core**로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. **./openshift-install gather** 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

참고

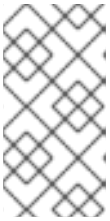
AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: **~/.ssh/id_ed25519**)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 **'~/.ssh** 디렉터리에 있는지 확인하십시오.



참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 **ed25519** 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 **rsa** 또는 **ecdsa** 알고리즘을 사용하는 키를 생성합니다.

2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 **~/.ssh/id_ed25519.pub** 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 **./openshift-install gather** 명령을 사용하려는 경우 필요합니다.

참고

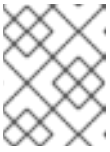
일부 배포에서는 **~/.ssh/id_rsa** 및 **~/.ssh/id_dsa**와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. **ssh-agent** 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```



참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. **ssh-agent**에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: **~/.ssh/id_ed25519**).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다. 프로비저닝하는 인프라에 클러스터를 설치하는 경우 설치 프로그램에 키를 제공해야 합니다.

4.12.8. AWS용 설치 파일 생성

사용자 프로비저닝 인프라를 사용하는 OpenShift Container Platform을 AWS(Amazon Web Services)에 설치하려면 설치 프로그램에서 클러스터를 배포하는 데 필요한 파일을 생성하고 클러스터가 사용할 시스템만을 생성하도록 파일을 수정해야 합니다. **install-config.yaml** 파일, Kubernetes 매니페스트 및 Ignition 구성 파일을 생성하고 사용자 지정합니다. 또한 설치 준비 단계에서 별도의 **var** 파티션을 먼저 설정할 수 있는 옵션이 있습니다.

4.12.8.1. 선택 사항: 별도의 /var 파티션 만들기

OpenShift Container Platform의 디스크 파티션 설정은 설치 프로그램에 맡기는 것이 좋습니다. 그러나 확장하려는 파일 시스템의 일부에 별도의 파티션을 생성해야 하는 경우가 있습니다.

OpenShift 컨테이너 플랫폼은 **/var** 파티션 또는 **/var**의 하위 디렉터리 중 하나에 스토리지를 연결하는 단일 파티션의 추가를 지원합니다. 예를 들면 다음과 같습니다.

- **/var/lib/containers**: 시스템에 더 많은 이미지와 컨테이너가 추가됨에 따라 확장될 수 있는 컨테이너 관련 콘텐츠를 보관합니다.
- **/var/lib/etcd**: etcd 스토리지의 성능 최적화와 같은 목적으로 별도로 보관할 데이터를 보관합니다.
- **/var**: 감사 등의 목적에 맞게 별도로 분리하여 보관해야 하는 데이터를 보관합니다.

/var 디렉터리의 콘텐츠를 별도로 저장하면 필요에 따라 해당 영역에 대한 스토리지 확장을 보다 용이하게 하고 나중에 OpenShift Container Platform을 다시 설치하여 해당 데이터를 그대로 보존할 수 있습니다. 이 방법을 사용하면 모든 컨테이너를 다시 가져올 필요가 없으며 시스템을 업데이트할 때 대용량 로그 파일을 복사할 필요도 없습니다.

RHCOS(Red Hat Enterprise Linux CoreOS)를 새로 설치하기 전에 **/var가 있어야 하므로 다음 절차에서는 OpenShift Container Platform 설치의 openshift-install 준비 단계 중에 삽입되는 머신 구성 매니페스트를 생성하여 별도의 /var** 파티션을 설정합니다.

중요

이 절차에서 별도의 **/var** 파티션을 생성하기 위해 단계에 따라 이 섹션의 뒷부분에서 설명한 대로 Kubernetes 매니페스트 및 Ignition 구성 파일을 다시 생성할 필요가 없습니다.

프로세스

1. OpenShift Container Platform 설치 파일을 저장할 디렉터리를 만듭니다.

```
$ mkdir $HOME/clusterconfig
```

2. **openshift-install**을 실행하여 **manifest** 및 **openshift** 하위 디렉터리에 파일 세트를 만듭니다. 프롬프트가 표시되면 시스템 질문에 대답합니다.

```
$ openshift-install create manifests --dir $HOME/clusterconfig
```

출력 예

```
? SSH Public Key ...
INFO Credentials loaded from the "myprofile" profile in file "/home/myuser/.aws/credentials"
INFO Consuming Install Config from target directory
INFO Manifests created in: $HOME/clusterconfig/manifests and
$HOME/clusterconfig/openshift
```

3. 선택 사항: 설치 프로그램이 **clusterconfig/openshift** 디렉터리에 매니페스트를 생성했는지 확인합니다.

```
$ ls $HOME/clusterconfig/openshift/
```

출력 예

```
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

4. 추가 파티션을 구성하는 Butane 구성을 생성합니다. 예를 들어 파일 이름을 **$HOME/clusterconfig/98-var-partition.bu** 로 지정하고, 디스크 장치 이름을 **worker** 시스템의 스토리지 장치 이름으로 변경하고 스토리지 크기를 적절하게 설정합니다. 이 예에서는 **/var** 디렉토리를 별도의 파티션에 배치합니다.

```
variant: openshift
version: 4.9.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1 파티션을 설정해야하는 디스크 저장 장치 이름입니다.

2 데이터 파티션을 부트 디스크에 추가할 때 최소 25000MiB(메비 바이트)가 권장됩니다. 루트 파일 시스템은 지정된 오프셋까지 사용 가능한 모든 공간을 채우기 위해 자동으로 크기가

3 데이터 파티션의 크기(MB)입니다.

4 컨테이너 스토리지에 사용되는 파일 시스템에 **prjquota** 마운트 옵션을 활성화해야 합니다.

참고

별도의 **/var** 파티션을 만들 때 다른 인스턴스 유형에 동일한 장치 이름이 없는 경우 작업자 노드에 다른 인스턴스 유형을 사용할 수 없습니다.

5. Butane 구성에서 매니페스트를 생성하여 **clusterconfig/openshift** 디렉터리에 저장합니다. 예를 들어 다음 명령을 실행합니다.

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

6. **openshift-install**을 다시 실행하여 **manifest** 및 **openshift** 하위 디렉터리의 파일 세트에서 Ignition 구성을 만듭니다.

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

이제 Ignition 구성 파일을 설치 절차에 대한 입력으로 사용하여 RHCOS (Red Hat Enterprise Linux CoreOS) 시스템을 설치할 수 있습니다.

4.12.8.2. 설치 구성 파일 만들기

설치 프로그램이 클러스터를 배포하는 데 필요한 설치 구성 파일을 생성하고 사용자 지정합니다.

사전 요구 사항

- 사용자가 프로비저닝한 인프라의 OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받으셨습니다.
- Red Hat에서 게시한 Red Hat Enterprise Linux CoreOS (RHCOS) AMI와 함께 리전에 클러스터를 배포하고 있는지 확인하셨습니다. AWS GovCloud 리전과 같이 사용자 지정 AMI가 필요한 리전에 배포하는 경우 **install-config.yaml** 파일을 수동으로 생성해야 합니다.

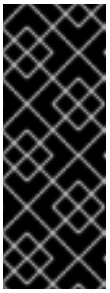
프로세스

1. **install-config.yaml** 파일을 생성합니다.

   a. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 **<installation_directory>**는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.



중요

비어 있는 디렉터리를 지정합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

b. 화면에 나타나는 지시에 따라 클라우드에 대한 구성 세부 사항을 입력합니다.

i. 선택사항: 클러스터 시스템에 액세스하는 데 사용할 SSH 키를 선택합니다.

참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

ii. 대상 플랫폼으로 **aws**를 선택합니다.

iii. 컴퓨터에 AWS 프로필이 저장되어 있지 않은 경우 설치 프로그램을 실행하도록 구성한 사용자의 AWS 액세스 키 ID와 시크릿 액세스 키를 입력합니다.

참고

AWS 액세스 키 ID와 시크릿 액세스 키는 설치 호스트에 있는 현재 사용자의 홈 디렉터리에서 **~/.aws/credentials**에 저장됩니다. 내보낸 프로필의 인증 정보가 파일에 없으면 설치 프로그램에서 인증 정보에 대한 메시지를 표시합니다. 설치 프로그램에 사용자가 제공하는 인증 정보는 파일에 저장됩니다.

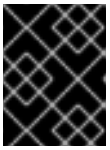
iv. 클러스터를 배포할 AWS 리전을 선택합니다.

v. 클러스터에 대해 구성한 Route53 서비스의 기본 도메인을 선택합니다.

vi. 클러스터를 설명할 수 있는 이름을 입력합니다.

vii. Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 붙여넣습니다.

2. 선택사항: **install-config.yaml** 파일을 백업합니다.



중요

install-config.yaml 파일은 설치 과정에서 사용됩니다. 이 파일을 재사용하려면 지금 백업해야 합니다.

추가 리소스

- AWS 프로필 및 인증 정보 구성에 대한 자세한 내용은 AWS 문서의 구성 및 인증 정보 파일 설정을 참조하십시오.

4.12.8.3. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 HTTP 또는 HTTPS 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 OpenShift Container Platform 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 API에 대한 호출을 포함하여 모든 클러스터 발신(Egress) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.

참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 Red Hat OpenStack Platform (RHOSP)에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

- **ec2.<region>.amazonaws.com,elasticloadbalancing.<region>.amazonaws.com** 및 **s3.<region>.amazonaws.com** 끝점을 VPC 끝점에 추가했습니다. 이러한 끝점은 노드에서 AWS EC2 API로 요청을 완료하는 데 필요합니다. 프록시는 노드 수준이 아닌 컨테이너 수준에서 작동하므로 이러한 요청을 AWS 사설 네트워크를 통해 AWS EC2 API로 라우팅해야 합니다. 프록시 서버의 허용 목록에 EC2 API의 공용 IP 주소를 추가하는 것만으로는 충분하지 않습니다.

절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. 클러스터 외부에서 HTTP 연결을 구축하는 데 사용할 프록시 URL입니다. URL 스키마는 **http**여야 합니다.
2. 클러스터 외부에서 HTTPS 연결을 구축하는 데 사용할 프록시 URL입니다.
3. 대상 도메인 이름, IP 주소 또는 프록시에서 제외할 기타 네트워크 CIDR로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. *를 사용하여 모든 대상에 대

해 프록시를 바이패스합니다.

4 이 값을 제공하면 설치 프로그램에서 추가 CA 인증서를 보유할 **openshift -config 네임스페이스에 user-ca- bundle** 이라는 구성 맵을 생성합니다. **additionalTrustBundle** 및 하나 이상의 프록시 설정을 제공하는 경우 **프록시** 오브젝트는 **trustedCA** 필드의 **user-ca-bundle** 구성 맵을 참조하도록 구성됩니다. 그러면 Cluster Network Operator에서 **trustedCA** 매개변수에 대해 지정된 콘텐츠를 RHCOS 신뢰 번들과 병합하는 **trusted-ca-bundle** 구성 맵을 생성합니다. 프록시의 ID 인증서를 RHCOS 트러스트 번들에 있는 기관에서 서명하지 않은 경우 **additionalTrustBundle** 필드가 있어야 합니다.

참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 OpenShift Container Platform을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 nil이 됩니다.

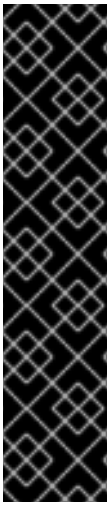
참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

4.12.8.4. Kubernetes 매니페스트 및 Ignition 설정 파일 생성

일부 클러스터 정의 파일을 수정하고 클러스터 시스템을 수동으로 시작해야 하므로 클러스터가 시스템을 구성하는 데 필요한 Kubernetes 매니페스트 및 Ignition 구성 파일을 사용자가 생성해야 합니다.

설치 구성 파일은 Kubernetes 매니페스트로 변환됩니다. 매니페스트는 나중에 클러스터 머신을 구성하는 데 사용되는 Ignition 구성 파일로 래핑됩니다.



중요

- OpenShift Container Platform 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램을 가져오셨습니다.
- **install-config.yaml** 설치 구성 파일을 생성하셨습니다.

프로세스

1. OpenShift Container Platform 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터에 대한 Kubernetes 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>**는 사용자가 만든 **install-config.yaml** 파일이 포함된 설치 디렉터리를 지정합니다.

2. 컨트롤 플레인 시스템을 정의하는 Kubernetes 매니페스트 파일을 제거합니다.

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

이러한 파일을 제거하면 클러스터가 컨트롤 플레인 시스템을 자동으로 생성하지 못합니다.

3. 작업자 시스템을 정의하는 Kubernetes 매니페스트 파일을 제거합니다.

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

작업자 시스템은 사용자가 직접 생성하고 관리하기 때문에 초기화할 필요가 없습니다.

4. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes 매니페스트 파일의 **mastersSchedulable** 매개변수가 **false**로 설정되어 있는지 확인합니다. 이 설정으로 인해 컨트롤 플레인 머신에서 포드가 예약되지 않습니다.

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** 파일을 엽니다.

   b. **mastersSchedulable** 매개변수를 찾아서 값을 **False**로 설정되어 있는지 확인합니다.

   c. 파일을 저장하고 종료합니다.

5. 선택사항: Ingress Operator가 사용자 대신 DNS 레코드를 생성하지 못하도록 하려면 **<installation_directory>/manifests/cluster-dns-02-config.yml** DNS 구성 파일에서 **privateZone** 및 **publicZone** 섹션을 제거합니다.

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

1 2 이 섹션을 완전히 제거합니다.

제거한 경우 나중에 인그레스 DNS 레코드를 수동으로 추가해야 합니다.

6. Ignition 구성 파일을 생성하려면 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행합니다.

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>**는 동일한 설치 디렉터리를 지정합니다.

설치 디렉터리의 부트스트랩, 컨트롤 플레인 및 컴퓨팅 노드에 대한 Ignition 구성 파일이 생성됩니다. **kubeadmin-password** 및 **kubeconfig** 파일은 **./<installation_directory>/auth** 디렉터리에 생성됩니다.

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

4.12.9. 인프라 이름 추출

Ignition 구성 파일에는 AWS(Amazon Web Services)에서 클러스터를 고유하게 식별하는 데 사용할 수 있는 고유한 클러스터 ID가 포함되어 있습니다. 인프라 이름은 OpenShift Container Platform 설치 중에 적절한 AWS 리소스를 찾는 데도 사용됩니다. 제공된 CloudFormation 템플릿에 이 인프라 이름에 대한 참조가 포함되어 있으므로 이름을 추출해야 합니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받으셨습니다.
- 클러스터에 대한 Ignition 구성 파일을 생성하셨습니다.
- **jq** CLI를 설치하셨습니다.

프로세스

- Ignition 구성 파일 메타데이터에서 인프라 이름을 추출하여 확인하려면 다음 명령을 실행하십시오.

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

출력 예

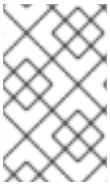
```
openshift-vw9j6 1
```

1 이 명령의 출력은 클러스터 이름과 임의의 문자열입니다.

4.12.10. AWS에서 VPC 생성

OpenShift Container Platform 클러스터에서 사용할 VPC(Virtual Private Cloud)를 AWS(Amazon Web Services)에서 생성해야 합니다. VPN 및 라우팅 테이블 등 요구사항에 맞게 VPC를 사용자 지정할 수 있습니다.

제공된 CloudFormation 템플릿과 사용자 정의 매개변수 파일을 사용하여 VPC를 나타내는 AWS 리소스 스택을 생성할 수 있습니다.



참고

AWS 인프라를 생성하는 데 제공된 CloudFormation 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- AWS 계정을 구성했습니다.
- **aws 구성**을 실행하여 AWS 키와 리전을 로컬 AWS 프로필에 추가하셨습니다.
- 클러스터에 대한 Ignition 구성 파일을 생성하셨습니다.

프로세스

1. 템플릿에 필요한 매개변수 값이 포함된 JSON 파일을 생성합니다.

```
[
  {
    "ParameterKey": "VpcCidr", 1
    "ParameterValue": "10.0.0.0/16" 2
  },
  {
    "ParameterKey": "AvailabilityZoneCount", 3
    "ParameterValue": "1" 4
  },
  {
    "ParameterKey": "SubnetBits", 5
    "ParameterValue": "12" 6
  }
]
```

1. VPC의 CIDR 블록입니다.
2. **xxxx/16-24** 형식으로 CIDR 블록을 지정합니다.
3. VPC를 배포할 가용성 영역의 수입니다.
4. **1**과 **3** 사이의 정수를 지정합니다.
5. 각 가용성 영역에 있는 각 서브넷의 크기입니다.
6. **5**와 **13** 사이의 정수를 지정합니다. 여기서 **5**는 **/27**이고 **13**은 **/19**입니다.

2. 이 항목의 **VPC 섹션에 대한 CloudFormation 템플릿**섹션에서 템플릿을 복사하여 컴퓨터에 YAML 파일로 저장합니다. 이 템플릿은 클러스터에 필요한 VPC를 설명합니다.

3. CloudFormation 템플릿을 시작하여 VPC를 나타내는 AWS 리소스 스택을 생성합니다.

중요

명령은 한 줄로 입력해야 합니다.

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
```

1. **<name>**은 CloudFormation 스택의 이름입니다(예: **cluster-vpc**). 클러스터를 제거하는 경우 이 스택의 이름이 필요합니다.
2. **<template>**은 저장한 CloudFormation 템플릿 YAML 파일의 상대 경로 및 이름입니다.
3. **<parameters>**는 CloudFormation 매개변수 JSON 파일의 상대 경로 및 이름입니다.

출력 예

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-vpc/dbedae40-2fd3-11eb-820e-12a48460849f
```

4. 템플릿 구성 요소가 있는지 확인합니다.

```
$ aws cloudformation describe-stacks --stack-name <name>
```

StackStatus에 **CREATE_COMPLETE**이 표시된 후 다음 매개변수의 출력 값이 표시됩니다. 클러스터를 생성하기 위해 실행하는 다른 CloudFormation 템플릿에 이러한 매개변수 값을 제공해야 합니다.

VpcId	VPC의 ID입니다.
PublicSubnetIds	새 퍼블릭 서브넷의 ID입니다.
PrivateSubnetIds	새 프라이빗 서브넷의 ID입니다.

4.12.10.1. VPC용 CloudFormation 템플릿

다음 CloudFormation 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 VPC를 배포할 수 있습니다.

예 4.36. VPC용 CloudFormation 템플릿

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for Best Practice VPC with 1-3 AZs

Parameters:
  VpcCidr:
```

```
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])(\/(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.0.0/16
    Description: CIDR block for VPC.
    Type: String
  AvailabilityZoneCount:
    ConstraintDescription: "The number of availability zones. (Min: 1, Max: 3)"
    MinValue: 1
    MaxValue: 3
    Default: 1
    Description: "How many AZs to create VPC subnets for. (Min: 1, Max: 3)"
    Type: Number
  SubnetBits:
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/19-27.
    MinValue: 5
    MaxValue: 13
    Default: 12
    Description: "Size of each subnet to create within the availability zones. (Min: 5 = /27, Max: 13 = /19)"
    Type: Number

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcCidr
      - SubnetBits
    - Label:
        default: "Availability Zones"
      Parameters:
      - AvailabilityZoneCount
    ParameterLabels:
      AvailabilityZoneCount:
        default: "Availability Zone Count"
      VpcCidr:
        default: "VPC CIDR"
      SubnetBits:
        default: "Bits Per Subnet"

Conditions:
  DoAz3: !Equals [3, !Ref AvailabilityZoneCount]
  DoAz2: !Or [!Equals [2, !Ref AvailabilityZoneCount], Condition: DoAz3]

Resources:
  VPC:
    Type: "AWS::EC2::VPC"
    Properties:
      EnableDnsSupport: "true"
      EnableDnsHostnames: "true"
      CidrBlock: !Ref VpcCidr
  PublicSubnet:
    Type: "AWS::EC2::Subnet"
    Properties:
```

```
      VpcId: !Ref VPC
      CidrBlock: !Select [0, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 0
      - Fn::GetAZs: !Ref "AWS::Region"
  PublicSubnet2:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz2
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [1, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 1
      - Fn::GetAZs: !Ref "AWS::Region"
  PublicSubnet3:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz3
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [2, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 2
      - Fn::GetAZs: !Ref "AWS::Region"
  InternetGateway:
    Type: "AWS::EC2::InternetGateway"
  GatewayToInternet:
    Type: "AWS::EC2::VPCGatewayAttachment"
    Properties:
      VpcId: !Ref VPC
      InternetGatewayId: !Ref InternetGateway
  PublicRouteTable:
    Type: "AWS::EC2::RouteTable"
    Properties:
      VpcId: !Ref VPC
  PublicRoute:
    Type: "AWS::EC2::Route"
    DependsOn: GatewayToInternet
    Properties:
      RouteTableId: !Ref PublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref InternetGateway
  PublicSubnetRouteTableAssociation:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PublicSubnet
      RouteTableId: !Ref PublicRouteTable
  PublicSubnetRouteTableAssociation2:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Condition: DoAz2
    Properties:
      SubnetId: !Ref PublicSubnet2
      RouteTableId: !Ref PublicRouteTable
  PublicSubnetRouteTableAssociation3:
    Condition: DoAz3
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
```

```
      SubnetId: !Ref PublicSubnet3
      RouteTableId: !Ref PublicRouteTable
  PrivateSubnet:
    Type: "AWS::EC2::Subnet"
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [3, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 0
      - Fn::GetAZs: !Ref "AWS::Region"
  PrivateRouteTable:
    Type: "AWS::EC2::RouteTable"
    Properties:
      VpcId: !Ref VPC
  PrivateSubnetRouteTableAssociation:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PrivateSubnet
      RouteTableId: !Ref PrivateRouteTable
  NAT:
    DependsOn:
    - GatewayToInternet
    Type: "AWS::EC2::NatGateway"
    Properties:
      AllocationId:
        "Fn::GetAtt":
        - EIP
        - AllocationId
      SubnetId: !Ref PublicSubnet
  EIP:
    Type: "AWS::EC2::EIP"
    Properties:
      Domain: vpc
  Route:
    Type: "AWS::EC2::Route"
    Properties:
      RouteTableId:
        Ref: PrivateRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId:
        Ref: NAT
  PrivateSubnet2:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz2
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [4, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 1
      - Fn::GetAZs: !Ref "AWS::Region"
  PrivateRouteTable2:
    Type: "AWS::EC2::RouteTable"
    Condition: DoAz2
    Properties:
      VpcId: !Ref VPC
  PrivateSubnetRouteTableAssociation2:
```

```
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Condition: DoAz2
  Properties:
    SubnetId: !Ref PrivateSubnet2
    RouteTableId: !Ref PrivateRouteTable2
NAT2:
  DependsOn:
  - GatewayToInternet
  Type: "AWS::EC2::NatGateway"
  Condition: DoAz2
  Properties:
    AllocationId:
      "Fn::GetAtt":
      - EIP2
      - AllocationId
    SubnetId: !Ref PublicSubnet2
EIP2:
  Type: "AWS::EC2::EIP"
  Condition: DoAz2
  Properties:
    Domain: vpc
Route2:
  Type: "AWS::EC2::Route"
  Condition: DoAz2
  Properties:
    RouteTableId:
      Ref: PrivateRouteTable2
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId:
      Ref: NAT2
PrivateSubnet3:
  Type: "AWS::EC2::Subnet"
  Condition: DoAz3
  Properties:
    VpcId: !Ref VPC
    CidrBlock: !Select [5, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
    AvailabilityZone: !Select
    - 2
    - Fn::GetAZs: !Ref "AWS::Region"
PrivateRouteTable3:
  Type: "AWS::EC2::RouteTable"
  Condition: DoAz3
  Properties:
    VpcId: !Ref VPC
PrivateSubnetRouteTableAssociation3:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Condition: DoAz3
  Properties:
    SubnetId: !Ref PrivateSubnet3
    RouteTableId: !Ref PrivateRouteTable3
NAT3:
  DependsOn:
  - GatewayToInternet
  Type: "AWS::EC2::NatGateway"
  Condition: DoAz3
  Properties:
```

```
      AllocationId:
        "Fn::GetAtt":
        - EIP3
        - AllocationId
      SubnetId: !Ref PublicSubnet3
  EIP3:
    Type: "AWS::EC2::EIP"
    Condition: DoAz3
    Properties:
      Domain: vpc
  Route3:
    Type: "AWS::EC2::Route"
    Condition: DoAz3
    Properties:
      RouteTableId:
        Ref: PrivateRouteTable3
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId:
        Ref: NAT3
  S3Endpoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      PolicyDocument:
        Version: 2012-10-17
        Statement:
        - Effect: Allow
          Principal: '*'
          Action:
          - '*'
          Resource:
          - '*'
      RouteTableIds:
      - !Ref PublicRouteTable
      - !Ref PrivateRouteTable
      - !If [DoAz2, !Ref PrivateRouteTable2, !Ref "AWS::NoValue"]
      - !If [DoAz3, !Ref PrivateRouteTable3, !Ref "AWS::NoValue"]
      ServiceName: !Join
      - ''
      - - com.amazonaws.
        - !Ref 'AWS::Region'
        - .s3
      VpcId: !Ref VPC

Outputs:
  VpcId:
    Description: ID of the new VPC.
    Value: !Ref VPC
  PublicSubnetIds:
    Description: Subnet IDs of the public subnets.
    Value:
      !Join [
        ",",
        [!Ref PublicSubnet, !If [DoAz2, !Ref PublicSubnet2, !Ref "AWS::NoValue"], !If [DoAz3, !Ref
PublicSubnet3, !Ref "AWS::NoValue"]]
      ]
  PrivateSubnetIds:
```

```
    Description: Subnet IDs of the private subnets.
    Value:
      !Join [
        ",",
        [!Ref PrivateSubnet, !If [DoAz2, !Ref PrivateSubnet2, !Ref "AWS::NoValue"], !If [DoAz3, !Ref
PrivateSubnet3, !Ref "AWS::NoValue"]]
        ]
```

추가 리소스

- AWS CloudFormation 콘솔로 이동하여 생성하는 CloudFormation 스택에 대한 세부 정보를 볼 수 있습니다.

4.12.11. AWS에서 네트워킹 및 로드 밸런싱 구성 요소 생성

AWS(Amazon Web Services)에서 OpenShift Container Platform 클러스터가 사용할 수 있는 네트워킹 및 클래식 또는 네트워크 로드 밸런싱을 구성해야 합니다.

제공된 CloudFormation 템플릿과 사용자 정의 매개변수 파일을 사용하여 AWS 리소스 스택을 생성할 수 있습니다. 스택은 OpenShift Container Platform 클러스터에 필요한 네트워킹 및 로드 밸런싱 구성 요소를 나타냅니다. 템플릿은 호스팅 영역 및 서브넷 태그도 생성합니다.

단일 VPC(Virtual Private Cloud)에서 템플릿을 여러 번 실행할 수 있습니다.

참고

AWS 인프라를 생성하는 데 제공된 CloudFormation 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- AWS 계정을 구성했습니다.
- **aws 구성**을 실행하여 AWS 키와 리전을 로컬 AWS 프로필에 추가하셨습니다.
- 클러스터에 대한 Ignition 구성 파일을 생성하셨습니다.
- AWS에서 VPC 및 관련 서브넷을 생성하고 구성하셨습니다.

프로세스

1. 클러스터의 **install-config.yaml** 파일에서 지정한 Route 53 기본 도메인의 호스팅 영역 ID를 가져옵니다. 다음 명령을 실행하여 호스팅 영역에 대한 세부 정보를 얻을 수 있습니다.

```
$ aws route53 list-hosted-zones-by-name --dns-name <route53_domain> 1
```

1 **<route53_domain>**은 클러스터의 **install-config.yaml** 파일을 생성할 때 사용한 Route 53 기본 도메인을 지정합니다.

출력 예

```
mycluster.example.com. False 100
HOSTEDZONES 65F8F38E-2268-B835-E15C-AB55336FCBFA
/hostedzone/Z21IXYZABCZ2A4 mycluster.example.com. 10
```

예제 출력에서 호스팅 영역 ID는 **Z21IXYZABCZ2A4**입니다.

2. 템플릿에 필요한 매개변수 값이 포함된 JSON 파일을 생성합니다.

```
[
  {
    "ParameterKey": "ClusterName", 1
    "ParameterValue": "mycluster" 2
  },
  {
    "ParameterKey": "InfrastructureName", 3
    "ParameterValue": "mycluster-<random_string>" 4
  },
  {
    "ParameterKey": "HostedZoneId", 5
    "ParameterValue": "<random_string>" 6
  },
  {
    "ParameterKey": "HostedZoneName", 7
    "ParameterValue": "example.com" 8
  },
  {
    "ParameterKey": "PublicSubnets", 9
    "ParameterValue": "subnet-<random_string>" 10
  },
  {
    "ParameterKey": "PrivateSubnets", 11
    "ParameterValue": "subnet-<random_string>" 12
  },
  {
    "ParameterKey": "VpcId", 13
    "ParameterValue": "vpc-<random_string>" 14
  }
]
```

1. 호스트 이름에 사용할 짧은 대표 클러스터 이름입니다.
2. 클러스터의 **install-config.yaml** 파일을 생성할 때 사용한 클러스터 이름을 지정합니다.
3. 클러스터에 대해 Ignition 구성 파일에 인코딩되는 클러스터 인프라의 이름입니다.
4. Ignition 구성 파일 메타데이터에서 추출한 인프라 이름을 **<cluster-name>-<random-string>** 형식으로 지정합니다.
5. 대상을 등록할 Route 53 공용 영역 ID입니다.
6. Route 53 공용 영역 ID를 **Z21IXYZABCZ2A4**와 유사한 형식으로 지정합니다. 이 값은 AWS 콘솔에서 가져올 수 있습니다.
7. 대상을 등록할 Route 53 영역입니다.

8 클러스터의 **install-config.yaml** 파일을 생성할 때 사용한 Route 53 기본 도메인을 지정합니다. AWS 콘솔에 표시되는 후행 마침표(.)는 포함하지 마십시오.

9 VPC용으로 만든 퍼블릭 서브넷입니다.

10 VPC에 대한 CloudFormation 템플릿의 출력에서 **PublicSubnetIds** 값을 지정합니다.

11 VPC용으로 만든 프라이빗 서브넷입니다.

12 VPC에 대한 CloudFormation 템플릿의 출력에서 **PrivateSubnetIds** 값을 지정합니다.

13 클러스터용으로 만든 VPC입니다.

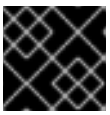
14 VPC에 대한 CloudFormation 템플릿의 출력에서 **VpcId** 값을 지정합니다.

3. 이 항목의 **네트워크 및 로드 밸런서에 대한 CloudFormation 템플릿**섹션에서 템플릿을 복사하여 사용자 컴퓨터에 YAML 파일로 저장합니다. 이 템플릿은 클러스터에 필요한 네트워킹 및 로드 밸런싱 개체를 설명합니다.

중요

클러스터를 AWS 정부 또는 시크릿 리전에 배포하는 경우 **CNAME** 레코드를 사용하여 CloudFormation 템플릿에서 **InternalApiServerRecord**를 업데이트해야 합니다. **ALIAS** 유형의 레코드는 AWS 정부 리전에서 지원되지 않습니다.

4. CloudFormation 템플릿을 시작하여 네트워킹 및 로드 밸런싱 구성 요소를 제공하는 AWS 리소스 스택을 생성합니다.



중요

명령은 한 줄로 입력해야 합니다.

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
    --capabilities CAPABILITY_NAMED_IAM 4
```

1 **<name>**은 CloudFormation 스택의 이름입니다(예: **cluster-dns**). 클러스터를 제거하는 경우 이 스택의 이름이 필요합니다.

2 **<template>**은 저장한 CloudFormation 템플릿 YAML 파일의 상대 경로 및 이름입니다.

3 **<parameters>**는 CloudFormation 매개변수 JSON 파일의 상대 경로 및 이름입니다.

4 제공된 템플릿에서 일부 **AWS::IAM::Role** 리소스를 생성하므로 **CAPABILITY_NAMED_IAM** 기능을 명시적으로 선언해야 합니다.

출력 예

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-dns/cd3e5de0-2fd4-11eb-5cf0-12be5c33a183
```

5. 템플릿 구성 요소가 있는지 확인합니다.

```
$ aws cloudformation describe-stacks --stack-name <name>
```

StackStatus에 **CREATE_COMPLETE**이 표시된 후 다음 매개변수의 출력 값이 표시됩니다. 클러스터를 생성하기 위해 실행하는 다른 CloudFormation 템플릿에 이러한 매개변수 값을 제공해야 합니다.

PrivateHostedZoneId	프라이빗 DNS의 호스팅 영역 ID입니다.
ExternalApiLoadBalancerName	외부 API 로드 밸런서의 전체 이름입니다.
InternalApiLoadBalancerName	내부 API 로드 밸런서의 전체 이름입니다.
ApiServerDnsName	API 서버의 전체 호스트 이름입니다.
RegisterNlbIpTargetsLambda	이러한 로드 밸런서의 IP 대상 등록/등록 취소에 유용한 Lambda ARN입니다.
ExternalApiTargetGroupArn	외부 API 대상 그룹의 ARN입니다.
InternalApiTargetGroupArn	내부 API 대상 그룹의 ARN입니다.
InternalServiceTargetGroupArn	내부 서비스 대상 그룹의 ARN입니다.

4.12.11.1. 네트워크 및 로드 밸런서에 대한 CloudFormation 템플릿

다음 CloudFormation 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 네트워킹 개체 및 로드 밸런서를 배포할 수 있습니다.

예 4.37. 네트워크 및 로드 밸런서에 대한 CloudFormation 템플릿

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Network Elements (Route53 & LBs)
```

```
Parameters:
  ClusterName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Cluster name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, representative cluster name to use for host names and other identifying
names.
    Type: String
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag cloud resources and identify items owned or
used by the cluster.
    Type: String
  HostedZoneId:
    Description: The Route53 public zone ID to register the targets with, such as
Z21IXYZABCZ2A4.
    Type: String
  HostedZoneName:
    Description: The Route53 zone to register the targets with, such as example.com. Omit the
trailing period.
    Type: String
    Default: "example.com"
  PublicSubnets:
    Description: The internet-facing subnets.
    Type: List<AWS::EC2::Subnet::Id>
  PrivateSubnets:
    Description: The internal subnets.
    Type: List<AWS::EC2::Subnet::Id>
  VpcId:
    Description: The VPC-scoped resources will belong to this VPC.
    Type: AWS::EC2::VPC::Id

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - ClusterName
      - InfrastructureName
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcId
      - PublicSubnets
      - PrivateSubnets
    - Label:
        default: "DNS"
      Parameters:
      - HostedZoneName
```

```
      - HostedZoneId
    ParameterLabels:
      ClusterName:
        default: "Cluster Name"
      InfrastructureName:
        default: "Infrastructure Name"
      VpcId:
        default: "VPC ID"
      PublicSubnets:
        default: "Public Subnets"
      PrivateSubnets:
        default: "Private Subnets"
      HostedZoneName:
        default: "Public Hosted Zone Name"
      HostedZoneId:
        default: "Public Hosted Zone ID"

Resources:
  ExtApiElb:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Name: !Join ["-", [!Ref InfrastructureName, "ext"]]
      IpAddressType: ipv4
      Subnets: !Ref PublicSubnets
      Type: network

  IntApiElb:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Name: !Join ["-", [!Ref InfrastructureName, "int"]]
      Scheme: internal
      IpAddressType: ipv4
      Subnets: !Ref PrivateSubnets
      Type: network

  IntDns:
    Type: "AWS::Route53::HostedZone"
    Properties:
      HostedZoneConfig:
        Comment: "Managed by CloudFormation"
      Name: !Join [".", [!Ref ClusterName, !Ref HostedZoneName]]
      HostedZoneTags:
      - Key: Name
        Value: !Join ["-", [!Ref InfrastructureName, "int"]]
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "owned"
      VPCs:
      - VPCId: !Ref VpcId
        VPCRegion: !Ref "AWS::Region"

  ExternalApiServerRecord:
    Type: AWS::Route53::RecordSetGroup
    Properties:
      Comment: Alias record for the API server
      HostedZoneId: !Ref HostedZoneId
      RecordSets:
```

```
      - Name:
          !Join [
            ".",
            ["api", !Ref ClusterName, !Join ["", [!Ref HostedZoneName, "."]]],
          ]
        Type: A
        AliasTarget:
          HostedZoneId: !GetAtt ExtApiElb.CanonicalHostedZoneID
          DNSName: !GetAtt ExtApiElb.DNSName

  InternalApiServerRecord:
    Type: AWS::Route53::RecordSetGroup
    Properties:
      Comment: Alias record for the API server
      HostedZoneId: !Ref IntDns
      RecordSets:
      - Name:
          !Join [
            ".",
            ["api", !Ref ClusterName, !Join ["", [!Ref HostedZoneName, "."]]],
          ]
        Type: A
        AliasTarget:
          HostedZoneId: !GetAtt IntApiElb.CanonicalHostedZoneID
          DNSName: !GetAtt IntApiElb.DNSName
      - Name:
          !Join [
            ".",
            ["api-int", !Ref ClusterName, !Join ["", [!Ref HostedZoneName, "."]]],
          ]
        Type: A
        AliasTarget:
          HostedZoneId: !GetAtt IntApiElb.CanonicalHostedZoneID
          DNSName: !GetAtt IntApiElb.DNSName

  ExternalApiListener:
    Type: AWS::ElasticLoadBalancingV2::Listener
    Properties:
      DefaultActions:
      - Type: forward
        TargetGroupArn:
          Ref: ExternalApiTargetGroup
      LoadBalancerArn:
        Ref: ExtApiElb
      Port: 6443
      Protocol: TCP

  ExternalApiTargetGroup:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      HealthCheckIntervalSeconds: 10
      HealthCheckPath: "/readyz"
      HealthCheckPort: 6443
      HealthCheckProtocol: HTTPS
      HealthyThresholdCount: 2
      UnhealthyThresholdCount: 2
```

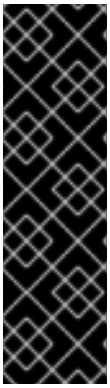
```
    Port: 6443
    Protocol: TCP
    TargetType: ip
    VpcId:
      Ref: VpcId
    TargetGroupAttributes:
    - Key: deregistration_delay.timeout_seconds
      Value: 60

  InternalApiListener:
    Type: AWS::ElasticLoadBalancingV2::Listener
    Properties:
      DefaultActions:
      - Type: forward
        TargetGroupArn:
          Ref: InternalApiTargetGroup
      LoadBalancerArn:
        Ref: IntApiElb
      Port: 6443
      Protocol: TCP

  InternalApiTargetGroup:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      HealthCheckIntervalSeconds: 10
      HealthCheckPath: "/readyz"
      HealthCheckPort: 6443
      HealthCheckProtocol: HTTPS
      HealthyThresholdCount: 2
      UnhealthyThresholdCount: 2
      Port: 6443
      Protocol: TCP
      TargetType: ip
      VpcId:
        Ref: VpcId
      TargetGroupAttributes:
      - Key: deregistration_delay.timeout_seconds
        Value: 60

  InternalServiceInternalListener:
    Type: AWS::ElasticLoadBalancingV2::Listener
    Properties:
      DefaultActions:
      - Type: forward
        TargetGroupArn:
          Ref: InternalServiceTargetGroup
      LoadBalancerArn:
        Ref: IntApiElb
      Port: 22623
      Protocol: TCP

  InternalServiceTargetGroup:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      HealthCheckIntervalSeconds: 10
      HealthCheckPath: "/healthz"
```

```
    HealthCheckPort: 22623
    HealthCheckProtocol: HTTPS
    HealthyThresholdCount: 2
    UnhealthyThresholdCount: 2
    Port: 22623
    Protocol: TCP
    TargetType: ip
    VpcId:
      Ref: VpcId
    TargetGroupAttributes:
    - Key: deregistration_delay.timeout_seconds
      Value: 60

  RegisterTargetLambdaIamRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: !Join ["-", [!Ref InfrastructureName, "nlb", "lambda", "role"]]
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
            - "lambda.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Path: "/"
      Policies:
      - PolicyName: !Join ["-", [!Ref InfrastructureName, "master", "policy"]]
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Action:
              [
                "elasticloadbalancing:RegisterTargets",
                "elasticloadbalancing:DeregisterTargets",
              ]
            Resource: !Ref InternalApiTargetGroup
          - Effect: "Allow"
            Action:
              [
                "elasticloadbalancing:RegisterTargets",
                "elasticloadbalancing:DeregisterTargets",
              ]
            Resource: !Ref InternalServiceTargetGroup
          - Effect: "Allow"
            Action:
              [
                "elasticloadbalancing:RegisterTargets",
                "elasticloadbalancing:DeregisterTargets",
              ]
            Resource: !Ref ExternalApiTargetGroup

  RegisterNlbIpTargets:
    Type: "AWS::Lambda::Function"
```

```
    Properties:
      Handler: "index.handler"
      Role:
        Fn::GetAtt:
        - "RegisterTargetLambdaIamRole"
        - "Arn"
      Code:
        ZipFile: |
          import json
          import boto3
          import cfnresponse
          def handler(event, context):
            elb = boto3.client('elbv2')
            if event['RequestType'] == 'Delete':
              elb.deregister_targets(TargetGroupArn=event['ResourceProperties']
['TargetArn'],Targets=[{'Id': event['ResourceProperties']['TargetIp']}])
            elif event['RequestType'] == 'Create':
              elb.register_targets(TargetGroupArn=event['ResourceProperties']['TargetArn'],Targets=
[{'Id': event['ResourceProperties']['TargetIp']}])
            responseData = {}
            cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData,
event['ResourceProperties']['TargetArn']+event['ResourceProperties']['TargetIp'])
      Runtime: "python3.7"
      Timeout: 120

  RegisterSubnetTagsLambdaIamRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: !Join ["-", [!Ref InfrastructureName, "subnet-tags-lambda-role"]]
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
            - "lambda.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Path: "/"
      Policies:
      - PolicyName: !Join ["-", [!Ref InfrastructureName, "subnet-tagging-policy"]]
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Action:
              [
                "ec2:DeleteTags",
                "ec2:CreateTags"
              ]
            Resource: "arn:aws:ec2:*:*:subnet/*"
          - Effect: "Allow"
            Action:
              [
                "ec2:DescribeSubnets",
                "ec2:DescribeTags"
```

```
            ]
          Resource: "*"

  RegisterSubnetTags:
    Type: "AWS::Lambda::Function"
    Properties:
      Handler: "index.handler"
      Role:
        Fn::GetAtt:
        - "RegisterSubnetTagsLambdaIamRole"
        - "Arn"
      Code:
        ZipFile: |
          import json
          import boto3
          import cfnresponse
          def handler(event, context):
            ec2_client = boto3.client('ec2')
            if event['RequestType'] == 'Delete':
              for subnet_id in event['ResourceProperties']['Subnets']:
                ec2_client.delete_tags(Resources=[subnet_id], Tags=[{'Key': 'kubernetes.io/cluster/' +
event['ResourceProperties']['InfrastructureName']}]);
            elif event['RequestType'] == 'Create':
              for subnet_id in event['ResourceProperties']['Subnets']:
                ec2_client.create_tags(Resources=[subnet_id], Tags=[{'Key': 'kubernetes.io/cluster/' +
event['ResourceProperties']['InfrastructureName'], 'Value': 'shared'}]);
            responseData = {}
            cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData,
event['ResourceProperties']['InfrastructureName']+event['ResourceProperties']['Subnets'][0])
      Runtime: "python3.7"
      Timeout: 120

  RegisterPublicSubnetTags:
    Type: Custom::SubnetRegister
    Properties:
      ServiceToken: !GetAtt RegisterSubnetTags.Arn
      InfrastructureName: !Ref InfrastructureName
      Subnets: !Ref PublicSubnets

  RegisterPrivateSubnetTags:
    Type: Custom::SubnetRegister
    Properties:
      ServiceToken: !GetAtt RegisterSubnetTags.Arn
      InfrastructureName: !Ref InfrastructureName
      Subnets: !Ref PrivateSubnets

Outputs:
  PrivateHostedZoneId:
    Description: Hosted zone ID for the private DNS, which is required for private records.
    Value: !Ref IntDns
  ExternalApiLoadBalancerName:
    Description: Full name of the external API load balancer.
    Value: !GetAtt ExtApiElb.LoadBalancerFullName
  InternalApiLoadBalancerName:
    Description: Full name of the internal API load balancer.
    Value: !GetAtt IntApiElb.LoadBalancerFullName
```

```
  ApiServerDnsName:
    Description: Full hostname of the API server, which is required for the Ignition config files.
    Value: !Join [".", ["api-int", !Ref ClusterName, !Ref HostedZoneName]]
  RegisterNlbIpTargetsLambda:
    Description: Lambda ARN useful to help register or deregister IP targets for these load
balancers.
    Value: !GetAtt RegisterNlbIpTargets.Arn
  ExternalApiTargetGroupArn:
    Description: ARN of the external API target group.
    Value: !Ref ExternalApiTargetGroup
  InternalApiTargetGroupArn:
    Description: ARN of the internal API target group.
    Value: !Ref InternalApiTargetGroup
  InternalServiceTargetGroupArn:
    Description: ARN of the internal service target group.
    Value: !Ref InternalServiceTargetGroup
```



중요

클러스터를 AWS 정부 또는 시크릿 리전에 배포하는 경우 **CNAME** 레코드를 사용하도록 **InternalApiServerRecord**를 업데이트해야 합니다. **ALIAS** 유형의 레코드는 AWS 정부 리전에서 지원되지 않습니다. 예를 들면 다음과 같습니다.

```
Type: CNAME
TTL: 10
ResourceRecords:
- !GetAtt IntApiElb.DNSName
```

추가 리소스

- AWS CloudFormation 콘솔로 이동하여 생성하는 CloudFormation 스택에 대한 세부 정보를 볼 수 있습니다.
- AWS Route 53 콘솔로 이동하여 호스팅 영역에 대한 세부 정보를 볼 수 있습니다.
- 공개 호스팅 영역 나열에 대한 자세한 내용은 AWS 문서의 공개 호스팅 영역 나열을 참조하십시오.

4.12.12. AWS에서 보안 그룹 및 역할 생성

OpenShift Container Platform 클러스터에서 사용할 보안 그룹 및 역할을 AWS(Amazon Web Services)에서 생성해야 합니다.

제공된 CloudFormation 템플릿과 사용자 정의 매개변수 파일을 사용하여 AWS 리소스 스택을 생성할 수 있습니다. 스택은 OpenShift Container Platform 클러스터에 필요한 보안 그룹 및 역할을 나타냅니다.

참고

AWS 인프라를 생성하는 데 제공된 CloudFormation 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- AWS 계정을 구성했습니다.
- **aws 구성**을 실행하여 AWS 키와 리전을 로컬 AWS 프로필에 추가하셨습니다.
- 클러스터에 대한 Ignition 구성 파일을 생성하셨습니다.
- AWS에서 VPC 및 관련 서브넷을 생성하고 구성하셨습니다.

프로세스

1. 템플릿에 필요한 매개변수 값이 포함된 JSON 파일을 생성합니다.

```
[
  {
    "ParameterKey": "InfrastructureName", (1)
    "ParameterValue": "mycluster-<random_string>" (2)
  },
  {
    "ParameterKey": "VpcCidr", (3)
    "ParameterValue": "10.0.0.0/16" (4)
  },
  {
    "ParameterKey": "PrivateSubnets", (5)
    "ParameterValue": "subnet-<random_string>" (6)
  },
  {
    "ParameterKey": "VpcId", (7)
    "ParameterValue": "vpc-<random_string>" (8)
  }
]
```

(1) 클러스터에 대해 Ignition 구성 파일에 인코딩되는 클러스터 인프라의 이름입니다.

(2) Ignition 구성 파일 메타데이터에서 추출한 인프라 이름을 **<cluster-name>-<random-string>** 형식으로 지정합니다.

(3) VPC의 CIDR 블록입니다.

(4) **xxxx/16-24** 양식으로 정의하고 VPC에 사용한 CIDR 블록 매개변수를 지정합니다.

(5) VPC용으로 만든 프라이빗 서브넷입니다.

(6) VPC에 대한 CloudFormation 템플릿의 출력에서 **PrivateSubnetIds** 값을 지정합니다.

(7) 클러스터용으로 만든 VPC입니다.

(8) VPC에 대한 CloudFormation 템플릿의 출력에서 **VpcId** 값을 지정합니다.

2. 이 항목의 **보안 개체에 대한 CloudFormation 템플릿** 섹션에서 템플릿을 복사하여 사용자 컴퓨터에 YAML 파일로 저장합니다. 이 템플릿은 클러스터에 필요한 보안 그룹 및 역할을 설명합니다.

3. CloudFormation 템플릿을 시작하여 보안 그룹 및 역할을 나타내는 AWS 리소스 스택을 생성합니다.

중요

명령은 한 줄로 입력해야 합니다.

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
    --capabilities CAPABILITY_NAMED_IAM 4
```

1. **<name>**은 CloudFormation 스택의 이름입니다(예: **cluster-sec**). 클러스터를 제거하는 경우 이 스택의 이름이 필요합니다.
2. **<template>**은 저장한 CloudFormation 템플릿 YAML 파일의 상대 경로 및 이름입니다.
3. **<parameters>**는 CloudFormation 매개변수 JSON 파일의 상대 경로 및 이름입니다.
4. 제공된 템플릿에서 일부 **AWS::IAM::Role** 및 **AWS::IAM::InstanceProfile** 리소스를 생성하므로 **CAPABILITY_NAMED_IAM** 기능을 명시적으로 선언해야 합니다.

출력 예

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-sec/03bd4210-2ed7-11eb-6d7a-13fc0b61e9db
```

4. 템플릿 구성 요소가 있는지 확인합니다.

```
$ aws cloudformation describe-stacks --stack-name <name>
```

StackStatus에 **CREATE_COMPLETE**이 표시된 후 다음 매개변수의 출력 값이 표시됩니다. 클러스터를 생성하기 위해 실행하는 다른 CloudFormation 템플릿에 이러한 매개변수 값을 제공해야 합니다.

MasterSecurityGroupId	마스터 보안 그룹 ID
WorkerSecurityGroupId	작업자 보안 그룹 ID
MasterInstanceProfile	마스터 IAM 인스턴스 프로필
WorkerInstanceProfile	작업자 IAM 인스턴스 프로필

4.12.12.1. 보안 개체의 CloudFormation 템플릿

다음 CloudFormation 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 보안 개체를 배포할 수 있습니다.

예 4.38. 보안 개체의 CloudFormation 템플릿

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Security Elements (Security Groups & IAM)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag cloud resources and identify items owned or
used by the cluster.
    Type: String
  VpcCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-
4][0-9]|25[0-5])(\/(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.0.0/16
    Description: CIDR block for VPC.
    Type: String
  VpcId:
    Description: The VPC-scoped resources will belong to this VPC.
    Type: AWS::EC2::VPC::Id
  PrivateSubnets:
    Description: The internal subnets.
    Type: List<AWS::EC2::Subnet::Id>

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - InfrastructureName
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcId
      - VpcCidr
      - PrivateSubnets
    ParameterLabels:
      InfrastructureName:
        default: "Infrastructure Name"
      VpcId:
        default: "VPC ID"
      VpcCidr:
        default: "VPC CIDR"
      PrivateSubnets:
        default: "Private Subnets"

Resources:
```

```
MasterSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Cluster Master Security Group
    SecurityGroupIngress:
    - IpProtocol: icmp
      FromPort: 0
      ToPort: 0
      CidrIp: !Ref VpcCidr
    - IpProtocol: tcp
      FromPort: 22
      ToPort: 22
      CidrIp: !Ref VpcCidr
    - IpProtocol: tcp
      ToPort: 6443
      FromPort: 6443
      CidrIp: !Ref VpcCidr
    - IpProtocol: tcp
      FromPort: 22623
      ToPort: 22623
      CidrIp: !Ref VpcCidr
    VpcId: !Ref VpcId

WorkerSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Cluster Worker Security Group
    SecurityGroupIngress:
    - IpProtocol: icmp
      FromPort: 0
      ToPort: 0
      CidrIp: !Ref VpcCidr
    - IpProtocol: tcp
      FromPort: 22
      ToPort: 22
      CidrIp: !Ref VpcCidr
    VpcId: !Ref VpcId

MasterIngressEtcd:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: etcd
    FromPort: 2379
    ToPort: 2380
    IpProtocol: tcp

MasterIngressVxlan:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Vxlan packets
    FromPort: 4789
    ToPort: 4789
```

```
      IpProtocol: udp

  MasterIngressWorkerVxlan:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Vxlan packets
      FromPort: 4789
      ToPort: 4789
      IpProtocol: udp

  MasterIngressGeneve:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Geneve packets
      FromPort: 6081
      ToPort: 6081
      IpProtocol: udp

  MasterIngressWorkerGeneve:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Geneve packets
      FromPort: 6081
      ToPort: 6081
      IpProtocol: udp

  MasterIngressIpsecIke:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: IPsec IKE packets
      FromPort: 500
      ToPort: 500
      IpProtocol: udp

  MasterIngressIpsecNat:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: IPsec NAT-T packets
      FromPort: 4500
      ToPort: 4500
      IpProtocol: udp

  MasterIngressIpsecEsp:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
```

```
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: IPsec ESP packets
    IpProtocol: 50

MasterIngressWorkerIpsecIke:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: IPsec IKE packets
    FromPort: 500
    ToPort: 500
    IpProtocol: udp

MasterIngressWorkerIpsecNat:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: IPsec NAT-T packets
    FromPort: 4500
    ToPort: 4500
    IpProtocol: udp

MasterIngressWorkerIpsecEsp:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: IPsec ESP packets
    IpProtocol: 50

MasterIngressInternal:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Internal cluster communication
    FromPort: 9000
    ToPort: 9999
    IpProtocol: tcp

MasterIngressWorkerInternal:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Internal cluster communication
    FromPort: 9000
    ToPort: 9999
    IpProtocol: tcp

MasterIngressInternalUDP:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
```

```
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: udp

  MasterIngressWorkerInternalUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: udp

  MasterIngressKube:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Kubernetes kubelet, scheduler and controller manager
      FromPort: 10250
      ToPort: 10259
      IpProtocol: tcp

  MasterIngressWorkerKube:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Kubernetes kubelet, scheduler and controller manager
      FromPort: 10250
      ToPort: 10259
      IpProtocol: tcp

  MasterIngressIngressServices:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: tcp

  MasterIngressWorkerIngressServices:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: tcp
```

```
MasterIngressIngressServicesUDP:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: udp

MasterIngressWorkerIngressServicesUDP:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: udp

WorkerIngressVxlan:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Vxlan packets
    FromPort: 4789
    ToPort: 4789
    IpProtocol: udp

WorkerIngressMasterVxlan:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Vxlan packets
    FromPort: 4789
    ToPort: 4789
    IpProtocol: udp

WorkerIngressGeneve:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Geneve packets
    FromPort: 6081
    ToPort: 6081
    IpProtocol: udp

WorkerIngressMasterGeneve:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Geneve packets
```

```
      FromPort: 6081
      ToPort: 6081
      IpProtocol: udp

  WorkerIngressIpsecIke:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: IPsec IKE packets
      FromPort: 500
      ToPort: 500
      IpProtocol: udp

  WorkerIngressIpsecNat:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: IPsec NAT-T packets
      FromPort: 4500
      ToPort: 4500
      IpProtocol: udp

  WorkerIngressIpsecEsp:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: IPsec ESP packets
      IpProtocol: 50

  WorkerIngressMasterIpsecIke:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: IPsec IKE packets
      FromPort: 500
      ToPort: 500
      IpProtocol: udp

  WorkerIngressMasterIpsecNat:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: IPsec NAT-T packets
      FromPort: 4500
      ToPort: 4500
      IpProtocol: udp

  WorkerIngressMasterIpsecEsp:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
```

```
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: IPsec ESP packets
      IpProtocol: 50

  WorkerIngressInternal:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: tcp

  WorkerIngressMasterInternal:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: tcp

  WorkerIngressInternalUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: udp

  WorkerIngressMasterInternalUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: udp

  WorkerIngressKube:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Kubernetes secure kubelet port
      FromPort: 10250
      ToPort: 10250
      IpProtocol: tcp

  WorkerIngressWorkerKube:
    Type: AWS::EC2::SecurityGroupIngress
```

```
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Internal Kubernetes communication
    FromPort: 10250
    ToPort: 10250
    IpProtocol: tcp

WorkerIngressIngressServices:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: tcp

WorkerIngressMasterIngressServices:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: tcp

WorkerIngressIngressServicesUDP:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: udp

WorkerIngressMasterIngressServicesUDP:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: udp

MasterIamRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Version: "2012-10-17"
      Statement:
      - Effect: "Allow"
        Principal:
```

```
          Service:
          - "ec2.amazonaws.com"
        Action:
        - "sts:AssumeRole"
    Policies:
    - PolicyName: !Join ["-", [!Ref InfrastructureName, "master", "policy"]]
      PolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Action:
          - "ec2:AttachVolume"
          - "ec2:AuthorizeSecurityGroupIngress"
          - "ec2:CreateSecurityGroup"
          - "ec2:CreateTags"
          - "ec2:CreateVolume"
          - "ec2:DeleteSecurityGroup"
          - "ec2:DeleteVolume"
          - "ec2:Describe*"
          - "ec2:DetachVolume"
          - "ec2:ModifyInstanceAttribute"
          - "ec2:ModifyVolume"
          - "ec2:RevokeSecurityGroupIngress"
          - "elasticloadbalancing:AddTags"
          - "elasticloadbalancing:AttachLoadBalancerToSubnets"
          - "elasticloadbalancing:ApplySecurityGroupsToLoadBalancer"
          - "elasticloadbalancing:CreateListener"
          - "elasticloadbalancing:CreateLoadBalancer"
          - "elasticloadbalancing:CreateLoadBalancerPolicy"
          - "elasticloadbalancing:CreateLoadBalancerListeners"
          - "elasticloadbalancing:CreateTargetGroup"
          - "elasticloadbalancing:ConfigureHealthCheck"
          - "elasticloadbalancing:DeleteListener"
          - "elasticloadbalancing:DeleteLoadBalancer"
          - "elasticloadbalancing:DeleteLoadBalancerListeners"
          - "elasticloadbalancing:DeleteTargetGroup"
          - "elasticloadbalancing:DeregisterInstancesFromLoadBalancer"
          - "elasticloadbalancing:DeregisterTargets"
          - "elasticloadbalancing:Describe*"
          - "elasticloadbalancing:DetachLoadBalancerFromSubnets"
          - "elasticloadbalancing:ModifyListener"
          - "elasticloadbalancing:ModifyLoadBalancerAttributes"
          - "elasticloadbalancing:ModifyTargetGroup"
          - "elasticloadbalancing:ModifyTargetGroupAttributes"
          - "elasticloadbalancing:RegisterInstancesWithLoadBalancer"
          - "elasticloadbalancing:RegisterTargets"
          - "elasticloadbalancing:SetLoadBalancerPoliciesForBackendServer"
          - "elasticloadbalancing:SetLoadBalancerPoliciesOfListener"
          - "kms:DescribeKey"
          Resource: "*"

  MasterInstanceProfile:
    Type: "AWS::IAM::InstanceProfile"
    Properties:
      Roles:
      - Ref: "MasterIamRole"
```

```
  WorkerIamRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
            - "ec2.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Policies:
      - PolicyName: !Join ["-", [!Ref InfrastructureName, "worker", "policy"]]
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Action:
            - "ec2:DescribeInstances"
            - "ec2:DescribeRegions"
            Resource: "*"

  WorkerInstanceProfile:
    Type: "AWS::IAM::InstanceProfile"
    Properties:
      Roles:
      - Ref: "WorkerIamRole"

Outputs:
  MasterSecurityGroupId:
    Description: Master Security Group ID
    Value: !GetAtt MasterSecurityGroup.GroupId

  WorkerSecurityGroupId:
    Description: Worker Security Group ID
    Value: !GetAtt WorkerSecurityGroup.GroupId

  MasterInstanceProfile:
    Description: Master IAM Instance Profile
    Value: !Ref MasterInstanceProfile

  WorkerInstanceProfile:
    Description: Worker IAM Instance Profile
    Value: !Ref WorkerInstanceProfile
```

추가 리소스

- AWS CloudFormation 콘솔로 이동하여 생성하는 CloudFormation 스택에 대한 세부 정보를 볼 수 있습니다.

4.12.13. 스트림 메타데이터를 사용하여 RHCOS AMI에 액세스

OpenShift Container Platform에서 *스트림 메타데이터*는 JSON 형식으로 RHCOS에 대한 표준화된 메타데이터를 제공하고 클러스터에 메타데이터를 삽입합니다. 스트림 메타데이터는 여러 아키텍처를 지원하는 안정적인 형식이며 자동화 유지하기 위해 자체 문서화하도록 설계되었습니다.

openshift-install의 **coreos print-stream-json** 하위 명령을 사용하여 스트림 메타데이터 형식의 부트 이미지에 대한 정보에 액세스할 수 있습니다. 이 명령은 스트림 메타데이터를 스크립트가 가능하고 컴퓨터가 읽을 수 있는 형식으로 인쇄하는 방법을 제공합니다.

사용자 프로비저닝 설치의 경우 **openshift-install** 바이너리에는 AWS AMI와 같은 OpenShift Container Platform과 함께 사용하기 위해 테스트된 RHCOS 부팅 이미지에 대한 참조가 포함되어 있습니다.

절차

스트림 메타데이터를 구문 분석하려면 다음 방법 중 하나를 사용합니다.

- Go 프로그램에서는 https://github.com/coreos/stream-metadata-go에서의 공식 **stream-metadata-go** 라이브러리를 사용합니다. 라이브러리에서 예제 코드를 볼 수도 있습니다.
- Python 또는 Ruby와 같은 다른 프로그래밍 언어에서 선호하는 프로그래밍 언어의 JSON 라이브러리를 사용합니다.
- **jq**와 같은 JSON 데이터를 처리하는 명령줄 유틸리티에서 다음을 수행합니다.
  - AWS 리전의 현재 **x86_64** AMI(예: **us-west-1**)를 출력합니다.

```
$ openshift-install coreos print-stream-json | jq -r
'.architectures.x86_64.images.aws.regions["us-west-1"].image'
```

출력 예

```
ami-0d3e625f84626bbda
```

이 명령의 출력은 **us-west-1** 리전의 AWS AMI ID입니다. AMI는 클러스터와 동일한 리전에 속해 있어야 합니다.

4.12.14. AWS 인프라용 RHCOS AMI

Red Hat은 OpenShift Container Platform 노드에 수동으로 지정할 수 있는 다양한 AWS 영역에 유효한 RHCOS(Red Hat Enterprise Linux CoreOS) AMI를 제공합니다.

참고

사용자 고유의 AMI를 가져 와서 RHCOS AMI가 게시되지 않은 리전에 설치할 수도 있습니다.

표 4.43. RHCOS AMI

AWS 영역	AWS AMI
af-south-1	**ami-0f3804f5a2f913dcc**
ap-east-1	**ami-0de1febb30a83da66**

AWS 영역	AWS AMI
ap-northeast-1	**ami-0183df96a3e002687**
ap-northeast-2	**ami-06b8798cd60242798**
ap-northeast-3	**ami-00b16b33aa0951016**
ap-south-1	**ami-007243f8ff78e8294**
ap-southeast-1	**ami-079dfdacb5ab5a0d1**
ap-southeast-2	**ami-03882e39cb7785c32**
ca-central-1	**ami-05cba1f80cc8b1dbe**
eu-central-1	**ami-073c775bbe9cd434e**
eu-north-1	**ami-0763e6e75b681acc5**
eu-south-1	**ami-00d023f19775fb64b**
eu-west-1	**ami-0033e3f2331a530c4**
eu-west-2	**ami-00d8a741ebe74f0c4**
eu-west-3	**ami-09b04e7f60e3374a7**
me-south-1	**ami-0f8039330b6e54010**
sa-east-1	**ami-01af22f821b470ad1**
us-east-1	**ami-0c72f473496a7b1c2**
us-east-2	**ami-09e637fc5885c13cc**
us-west-1	**ami-0fa0f6fce7e63dd26**
us-west-2	**ami-084fb1316cd1ed4cc**

4.12.14.1. 게시된 RHCOS AMI가 없는 AWS 리전

Red Hat Enterprise Linux CoreOS (RHCOS) Amazon Machine Image (AMI) 또는 AWS 소프트웨어 개발 키트 (SDK)에 대한 기본 지원없이 Amazon Web Services (AWS) 리전에 OpenShift Container Platform 클러스터를 배포할 수 있습니다. 게시된 AMI를 AWS 리전에서 사용할 수 없는 경우 클러스터를 설치하기 전에 사용자 지정 AMI를 업로드할 수 있습니다.

AWS SDK에서 지원하지 않는 리전에 배포하고 사용자 지정 AMI를 지정하지 않은 경우 설치 프로그램은

us-east-1 AMI를 사용자 계정에 자동으로 복사합니다. 그 다음 설치 프로그램은 기본 또는 사용자 지정 KMS (Key Management Service) 키를 사용하여 암호화 된 EBS 볼륨에서 컨트롤 플레인 시스템을 생성합니다. 이를 통해 AMI는 게 된 RHCOS AMI와 동일한 프로세스 워크 플로를 실행할 수 있습니다.

RHCOS AMI에 대한 기본 지원이 없는 리전은 게시되지 않기 때문에 클러스터 생성 중에 터미널에서 선택할 수 없습니다. 그러나 **install-config.yaml** 파일에서 사용자 지정 AMI를 구성하여 이 리전에 설치할 수 있습니다.

4.12.14.2. AWS에서 사용자 지정 RHCOS AMI 업로드

사용자 지정 Amazon Web Services (AWS) 리전에 배포하는 경우 해당 리전에 속하는 사용자 지정 Red Hat Enterprise Linux CoreOS (RHCOS) Amazon 머신 이미지 (AMI)를 업로드해야 합니다.

사전 요구 사항

- AWS 계정을 구성했습니다.
- 필요한 IAM 서비스 역할로 Amazon S3 버킷을 생성했습니다.
- RHCOS VMDK 파일을 Amazon S3에 업로드했습니다. RHCOS VMDK 파일은 설치하는 OpenShift Container Platform 버전과 같거나 그 이하의 버전이어야 합니다.
- AWS CLI를 다운로드하여 컴퓨터에 설치했습니다. Install the AWS CLI Using the Bundled Installer를 참조하십시오.

프로세스

1. AWS 프로필을 환경 변수로 내보냅니다.

```
$ export AWS_PROFILE=<aws_profile> 1
```

2. 사용자 지정 AMI와 연결할 리전을 환경 변수로 내보냅니다.

```
$ export AWS_DEFAULT_REGION=<aws_region> 1
```

3. Amazon S3에 업로드한 RHCOS 버전을 환경 변수로 내보냅니다.

```
$ export RHCOS_VERSION=<version> 1
```

1 1 1 **4.9.0** 과 같은 RHCOS VMDK 버전입니다.

4. Amazon S3 버킷 이름을 환경 변수로 내보냅니다.

```
$ export VMIMPORT_BUCKET_NAME=<s3_bucket_name>
```

5. **container.json** 파일을 만들고 RHCOS VMDK 파일을 정의합니다.

```
$ cat <<EOF > containers.json
{
  "Description": "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64",
  "Format": "vmdk",
  "UserBucket": {
    "S3Bucket": "${VMIMPORT_BUCKET_NAME}",
```

```
      "S3Key": "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64.vmdk"
    }
}
EOF
```

6. RHCOS 디스크를 Amazon EBS 스냅샷으로 가져옵니다.

```
$ aws ec2 import-snapshot --region ${AWS_DEFAULT_REGION} \
    --description "<description>" \ 1
    --disk-container "file://<file_path>/containers.json" 2
```

1 가져온 RHCOS 디스크에 대한 설명 (예: **rhcos-$ {RHCOS_VERSION} -x86_64-aws.x86_64**)입니다.

2 RHCOS 디스크를 설명하는 JSON 파일의 파일 경로입니다. JSON 파일에는 Amazon S3 버킷 이름과 키가 포함되어 있어야합니다.

7. 이미지 가져 오기 상태를 확인합니다.

```
$ watch -n 5 aws ec2 describe-import-snapshot-tasks --region ${AWS_DEFAULT_REGION}
```

출력 예

```
{
    "ImportSnapshotTasks": [
        {
            "Description": "rhcos-4.7.0-x86_64-aws.x86_64",
            "ImportTaskId": "import-snap-fh6i8uil",
            "SnapshotTaskDetail": {
                "Description": "rhcos-4.7.0-x86_64-aws.x86_64",
                "DiskImageSize": 819056640.0,
                "Format": "VMDK",
                "SnapshotId": "snap-06331325870076318",
                "Status": "completed",
                "UserBucket": {
                    "S3Bucket": "external-images",
                    "S3Key": "rhcos-4.7.0-x86_64-aws.x86_64.vmdk"
                }
            }
        }
    ]
}
```

SnapshotId를 복사하여 이미지를 등록합니다.

8. RHCOS 스냅 샷에서 사용자 지정 RHCOS AMI를 생성합니다.

```
$ aws ec2 register-image \
  --region ${AWS_DEFAULT_REGION} \
  --architecture x86_64 \ 1
  --description "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64" \ 2
  --ena-support \
  --name "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64" \ 3
```

```
    --virtualization-type hvm \
    --root-device-name '/dev/xvda' \
    --block-device-mappings 'DeviceName=/dev/xvda,Ebs=
{DeleteOnTermination=true,SnapshotId=<snapshot_ID>}' 4
```

1. **x86_64**, **s390x** 또는 **ppc64le**과 같은 RHCOS VMDK 아키텍처 유형입니다.
2. 가져온 스냅샷의 **Description**입니다.
3. RHCOS AMI의 이름입니다.
4. 가져온 스냅샷의 **SnapshotID**입니다.

이러한 API에 대한 자세한 내용은 importing snapshots 및 creating EBS-backed AMIs에서 참조하십시오.

4.12.15. AWS에서 부트스트랩 노드 생성

OpenShift Container Platform 클러스터 초기화 과정에서 사용할 부트스트랩 노드를 AWS(Amazon Web Services)에서 생성해야 합니다.

제공된 CloudFormation 템플릿과 사용자 정의 매개변수 파일을 사용하여 AWS 리소스 스택을 생성할 수 있습니다. 스택은 OpenShift Container Platform 설치에 필요한 부트스트랩 노드를 나타냅니다.

참고

부트스트랩 노드를 생성하는 데 제공된 CloudFormation 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다.. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

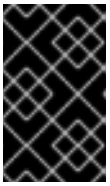
- AWS 계정을 구성했습니다.
- **aws 구성**을 실행하여 AWS 키와 리전을 로컬 AWS 프로필에 추가하셨습니다.
- 클러스터에 대한 Ignition 구성 파일을 생성하셨습니다.
- AWS에서 VPC 및 관련 서브넷을 생성하고 구성하셨습니다.
- AWS에서 DNS, 로드 밸런서 및 리스너를 생성하고 구성하셨습니다.
- AWS에서 클러스터에 필요한 보안 그룹 및 역할을 생성하셨습니다.

프로세스

1. **bootstrap.ign** Ignition 구성 파일을 클러스터에 제공할 위치를 지정합니다. 이 파일은 설치 디렉터리에 있습니다. 한 가지 방법은 클러스터의 리전에 S3 버킷을 생성하고 여기에 Ignition 구성 파일을 업로드하는 것입니다.

중요

제공된 CloudFormation 템플릿은 클러스터의 Ignition 구성 파일이 S3 버킷에서 제공되는 것으로 가정합니다. 다른 위치에서 파일을 제공하려면 템플릿을 수정해야 합니다.



중요

AWS SDK와 다른 엔드 포인트가 있는 리전에 배포하거나 자체 사용자 지정 엔드 포인트를 제공하는 경우 **s3://** 스키마 대신 사전에 서명된 URL을 S3 버킷에 사용해야 합니다.

참고

부트스트랩 Ignition 구성 파일에는 X.509 키와 같은 시크릿이 포함되어 있습니다. 다음 단계는 S3 버킷에 대한 기본 보안을 제공합니다. 추가 보안을 제공하기 위해 OpenShift IAM 사용자와 같은 특정 사용자만 버킷에 포함된 개체에 액세스할 수 있도록 S3 버킷 정책을 활성화할 수 있습니다. S3과 관계없이 부트스트랩 시스템이 도달할 수 있는 모든 주소에서 부트스트랩 Ignition 구성 파일을 제공할 수 있습니다.

a. 버킷을 만듭니다.

```
$ aws s3 mb s3://<cluster-name>-infra 1
```

1 **<cluster-name>-infra**는 버킷 이름입니다. **install-config.yaml** 파일을 생성할 때 **<cluster-name>**을 클러스터에 지정된 이름으로 교체합니다.

b. **bootstrap.ign** Ignition 구성 파일을 버킷에 업로드합니다.

```
$ aws s3 cp <installation_directory>/bootstrap.ign s3://<cluster-name>-
infra/bootstrap.ign 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

c. 파일이 업로드되었는지 확인합니다.

```
$ aws s3 ls s3://<cluster-name>-infra/
```

출력 예

```
2019-04-03 16:15:16     314878 bootstrap.ign
```

2. 템플릿에 필요한 매개변수 값이 포함된 JSON 파일을 생성합니다.

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "RhcosAmi", 3
    "ParameterValue": "ami-<random_string>" 4
  },
  {
    "ParameterKey": "AllowedBootstrapSshCidr", 5
    "ParameterValue": "0.0.0.0/0" 6
```

```
  },
  {
    "ParameterKey": "PublicSubnet", 7
    "ParameterValue": "subnet-<random_string>" 8
  },
  {
    "ParameterKey": "MasterSecurityGroupId", 9
    "ParameterValue": "sg-<random_string>" 10
  },
  {
    "ParameterKey": "VpcId", 11
    "ParameterValue": "vpc-<random_string>" 12
  },
  {
    "ParameterKey": "BootstrapIgnitionLocation", 13
    "ParameterValue": "s3://<bucket_name>/bootstrap.ign" 14
  },
  {
    "ParameterKey": "AutoRegisterELB", 15
    "ParameterValue": "yes" 16
  },
  {
    "ParameterKey": "RegisterNlbIpTargetsLambdaArn", 17
    "ParameterValue": "arn:aws:lambda:<region>:<account_number>:function:
<dns_stack_name>-RegisterNlbIpTargets-<random_string>" 18
  },
  {
    "ParameterKey": "ExternalApiTargetGroupArn", 19
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Exter-<random_string>" 20
  },
  {
    "ParameterKey": "InternalApiTargetGroupArn", 21
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 22
  },
  {
    "ParameterKey": "InternalServiceTargetGroupArn", 23
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 24
  }
]
```

1. 클러스터에 대해 Ignition 구성 파일에 인코딩되는 클러스터 인프라의 이름입니다.
2. Ignition 구성 파일 메타데이터에서 추출한 인프라 이름을 **<cluster-name>-<random-string>** 형식으로 지정합니다.
3. 부트스트랩 노드에 사용할 현재 RHCOS(Red Hat Enterprise Linux CoreOS) AMI입니다.
4. 유효한 **AWS::EC2::Image::Id** 값을 지정합니다.
5. 부트스트랩 노드에 대한 SSH 액세스를 허용하는 CIDR 블록입니다.

6. **xxxx/16-24** 형식으로 CIDR 블록을 지정합니다.
7. 부트스트랩 노드를 시작하기 위해 VPC와 연결되는 퍼블릭 서브넷입니다.
8. VPC에 대한 CloudFormation 템플릿의 출력에서 **PublicSubnetIds** 값을 지정합니다.
9. 마스터 보안 그룹 ID(임시 규칙 등록용)입니다.
10. 보안 그룹 및 역할에 대한 CloudFormation 템플릿 출력에서 **MasterSecurityGroupId** 값을 지정합니다.
11. VPC 생성 리소스가 속합니다.
12. VPC에 대한 CloudFormation 템플릿의 출력에서 **VpcId** 값을 지정합니다.
13. 부트스트랩 Ignition 구성 파일을 가져올 위치입니다.
14. S3 버킷과 파일 이름을 **s3://<bucket_name>/bootstrap.ign** 형식으로 지정합니다.
15. 네트워크 로드 밸런서(NLB) 등록 여부입니다.
16. **yes** 또는 **no**를 지정합니다. **yes**를 지정하는 경우 Lambda ARN(Amazon Resource Name) 값을 제공해야 합니다.
17. NLB IP 대상 등록 lambda 그룹의 ARN입니다.
18. DNS 및 로드 밸런싱을 위해 CloudFormation 템플릿의 출력에서 **RegisterNlbIpTargetsLambda** 값을 지정합니다. 클러스터를 AWS GovCloud 리전에 배포하는 경우 **arn:aws-us-gov**를 사용합니다.
19. 외부 API 로드 밸런서 대상 그룹의 ARN입니다.
20. DNS 및 로드 밸런싱의 CloudFormation 템플릿 출력에서 **ExternalApiTargetGroupArn** 값을 지정합니다. 클러스터를 AWS GovCloud 리전에 배포하는 경우 **arn:aws-us-gov**를 사용합니다.
21. 내부 API 로드 밸런서 대상 그룹의 ARN입니다.
22. DNS 및 로드 밸런싱의 CloudFormation 템플릿 출력에서 **InternalApiTargetGroupArn** 값을 지정합니다. 클러스터를 AWS GovCloud 리전에 배포하는 경우 **arn:aws-us-gov**를 사용합니다.
23. 내부 서비스 로드 밸런서 대상 그룹의 ARN입니다.
24. DNS 및 로드 밸런싱의 CloudFormation 템플릿 출력에서 **InternalServiceTargetGroupArn** 값을 지정합니다. 클러스터를 AWS GovCloud 리전에 배포하는 경우 **arn:aws-us-gov**를 사용합니다.

3. 이 항목의 **부트스트랩 시스템의 CloudFormation 템플릿** 섹션에서 템플릿을 복사하여 사용자 시스템에 YAML 파일로 저장합니다. 이 템플릿은 클러스터에 필요한 부트스트랩 시스템을 설명합니다.
4. CloudFormation 템플릿을 시작하여 부트스트랩 노드를 나타내는 AWS 리소스 스택을 생성합니다.

중요

명령은 한 줄로 입력해야 합니다.

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
    --capabilities CAPABILITY_NAMED_IAM 4
```

1. **<name>**은 CloudFormation 스택의 이름입니다(예: **cluster-bootstrap**). 클러스터를 제거하는 경우 이 스택의 이름이 필요합니다.
2. **<template>**은 저장한 CloudFormation 템플릿 YAML 파일의 상대 경로 및 이름입니다.
3. **<parameters>**는 CloudFormation 매개변수 JSON 파일의 상대 경로 및 이름입니다.
4. 제공된 템플릿에서 일부 **AWS::IAM::Role** 및 **AWS::IAM::InstanceProfile** 리소스를 생성하므로 **CAPABILITY_NAMED_IAM** 기능을 명시적으로 선언해야 합니다.

출력 예

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-bootstrap/12944486-2add-11eb-9dee-12dace8e3a83
```

5. 템플릿 구성 요소가 있는지 확인합니다.

```
$ aws cloudformation describe-stacks --stack-name <name>
```

StackStatus에 **CREATE_COMPLETE**이 표시된 후 다음 매개변수의 출력 값이 표시됩니다. 클러스터를 생성하기 위해 실행하는 다른 CloudFormation 템플릿에 이러한 매개변수 값을 제공해야 합니다.

Bootstrap InstanceId	부트스트랩 인스턴스 ID입니다.
Bootstrap PublicIp	부트스트랩 노드 공용 IP 주소입니다.
Bootstrap PrivateIp	부트스트랩 노드 개인 IP 주소입니다.

4.12.15.1. 부트스트랩 시스템용 CloudFormation 템플릿

다음 CloudFormation 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 부트스트랩 시스템을 배포할 수 있습니다.

예 4.39. 부트스트랩 시스템용 CloudFormation 템플릿

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Bootstrap (EC2 Instance, Security Groups and IAM)
```

```
Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag cloud resources and identify items owned or
used by the cluster.
    Type: String
  RhcosAmi:
    Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.
    Type: AWS::EC2::Image::Id
  AllowedBootstrapSshCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-
4][0-9]|25[0-5])(\/([0-9]|1[0-9]|2[0-9]|3[0-2]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/0-32.
    Default: 0.0.0.0/0
    Description: CIDR block to allow SSH access to the bootstrap node.
    Type: String
  PublicSubnet:
    Description: The public subnet to launch the bootstrap node into.
    Type: AWS::EC2::Subnet::Id
  MasterSecurityGroupId:
    Description: The master security group ID for registering temporary rules.
    Type: AWS::EC2::SecurityGroup::Id
  VpcId:
    Description: The VPC-scoped resources will belong to this VPC.
    Type: AWS::EC2::VPC::Id
  BootstrapIgnitionLocation:
    Default: s3://my-s3-bucket/bootstrap.ign
    Description: Ignition config file location.
    Type: String
  AutoRegisterELB:
    Default: "yes"
    AllowedValues:
    - "yes"
    - "no"
    Description: Do you want to invoke NLB registration, which requires a Lambda ARN parameter?
    Type: String
  RegisterNlbIpTargetsLambdaArn:
    Description: ARN for NLB IP target registration lambda.
    Type: String
  ExternalApiTargetGroupArn:
    Description: ARN for external API load balancer target group.
    Type: String
  InternalApiTargetGroupArn:
    Description: ARN for internal API load balancer target group.
    Type: String
  InternalServiceTargetGroupArn:
    Description: ARN for internal service load balancer target group.
    Type: String

Metadata:
  AWS::CloudFormation::Interface:
```

```
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - InfrastructureName
    - Label:
        default: "Host Information"
      Parameters:
      - RhcosAmi
      - BootstrapIgnitionLocation
      - MasterSecurityGroupId
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcId
      - AllowedBootstrapSshCidr
      - PublicSubnet
    - Label:
        default: "Load Balancer Automation"
      Parameters:
      - AutoRegisterELB
      - RegisterNlbIpTargetsLambdaArn
      - ExternalApiTargetGroupArn
      - InternalApiTargetGroupArn
      - InternalServiceTargetGroupArn
    ParameterLabels:
      InfrastructureName:
        default: "Infrastructure Name"
      VpcId:
        default: "VPC ID"
      AllowedBootstrapSshCidr:
        default: "Allowed SSH Source"
      PublicSubnet:
        default: "Public Subnet"
      RhcosAmi:
        default: "Red Hat Enterprise Linux CoreOS AMI ID"
      BootstrapIgnitionLocation:
        default: "Bootstrap Ignition Source"
      MasterSecurityGroupId:
        default: "Master Security Group ID"
      AutoRegisterELB:
        default: "Use Provided ELB Automation"

Conditions:
  DoRegistration: !Equals ["yes", !Ref AutoRegisterELB]

Resources:
  BootstrapIamRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
```

```
          - "ec2.amazonaws.com"
        Action:
        - "sts:AssumeRole"
    Path: "/"
    Policies:
    - PolicyName: !Join ["-", [!Ref InfrastructureName, "bootstrap", "policy"]]
      PolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Action: "ec2:Describe*"
          Resource: "*"
        - Effect: "Allow"
          Action: "ec2:AttachVolume"
          Resource: "*"
        - Effect: "Allow"
          Action: "ec2:DetachVolume"
          Resource: "*"
        - Effect: "Allow"
          Action: "s3:GetObject"
          Resource: "*"

  BootstrapInstanceProfile:
    Type: "AWS::IAM::InstanceProfile"
    Properties:
      Path: "/"
      Roles:
      - Ref: "BootstrapIamRole"

  BootstrapSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Cluster Bootstrap Security Group
      SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: !Ref AllowedBootstrapSshCidr
      - IpProtocol: tcp
        ToPort: 19531
        FromPort: 19531
        CidrIp: 0.0.0.0/0
      VpcId: !Ref VpcId

  BootstrapInstance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      IamInstanceProfile: !Ref BootstrapInstanceProfile
      InstanceType: "i3.large"
      NetworkInterfaces:
      - AssociatePublicIpAddress: "true"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "BootstrapSecurityGroup"
        - !Ref "MasterSecurityGroupId"
```

```
      SubnetId: !Ref "PublicSubnet"
    UserData:
      Fn::Base64: !Sub
      - '{"ignition":{"config":{"replace":{"source":"${S3Loc}"}},"version":"3.1.0"}}'
      - {
        S3Loc: !Ref BootstrapIgnitionLocation
      }

  RegisterBootstrapApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref ExternalApiTargetGroupArn
      TargetIp: !GetAtt BootstrapInstance.PrivateIp

  RegisterBootstrapInternalApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalApiTargetGroupArn
      TargetIp: !GetAtt BootstrapInstance.PrivateIp

  RegisterBootstrapInternalServiceTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalServiceTargetGroupArn
      TargetIp: !GetAtt BootstrapInstance.PrivateIp

Outputs:
  BootstrapInstanceId:
    Description: Bootstrap Instance ID.
    Value: !Ref BootstrapInstance

  BootstrapPublicIp:
    Description: The bootstrap node public IP address.
    Value: !GetAtt BootstrapInstance.PublicIp

  BootstrapPrivateIp:
    Description: The bootstrap node private IP address.
    Value: !GetAtt BootstrapInstance.PrivateIp
```

추가 리소스

- AWS CloudFormation 콘솔로 이동하여 생성하는 CloudFormation 스택에 대한 세부 정보를 볼 수 있습니다.
- AWS 영역의 RHCOS(Red Hat Enterprise Linux CoreOS) AMI에 대한 자세한 내용은 AWS 인프라의 RHCOS AMI를 참조하십시오.

4.12.16. AWS에서 컨트롤 플레인 시스템 생성

클러스터가 사용할 컨트롤 플레인 시스템을 AWS(Amazon Web Services)에서 생성해야 합니다.

제공된 CloudFormation 템플릿과 사용자 정의 매개변수 파일을 사용하여 컨트롤 플레인 노드를 나타내는 AWS 리소스 스택을 생성할 수 있습니다.

중요

CloudFormation 템플릿은 세 개의 컨트롤 플레인 노드를 나타내는 스택을 생성합니다.

참고

컨트롤 플레인 노드를 생성하는 데 제공된 CloudFormation 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다.. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- AWS 계정을 구성했습니다.
- **aws 구성**을 실행하여 AWS 키와 리전을 로컬 AWS 프로필에 추가하셨습니다.
- 클러스터에 대한 Ignition 구성 파일을 생성하셨습니다.
- AWS에서 VPC 및 관련 서브넷을 생성하고 구성하셨습니다.
- AWS에서 DNS, 로드 밸런서 및 리스너를 생성하고 구성하셨습니다.
- AWS에서 클러스터에 필요한 보안 그룹 및 역할을 생성하셨습니다.
- 부트스트랩 시스템을 생성하셨습니다.

프로세스

1. 템플릿에 필요한 매개변수 값이 포함된 JSON 파일을 생성합니다.

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "RhcosAmi", 3
    "ParameterValue": "ami-<random_string>" 4
  },
  {
    "ParameterKey": "AutoRegisterDNS", 5
    "ParameterValue": "yes" 6
  },
  {
    "ParameterKey": "PrivateHostedZoneId", 7
    "ParameterValue": "<random_string>" 8
  },
  {
    "ParameterKey": "PrivateHostedZoneName", 9
```

```
    "ParameterValue": "mycluster.example.com" 10
  },
  {
    "ParameterKey": "Master0Subnet", 11
    "ParameterValue": "subnet-<random_string>" 12
  },
  {
    "ParameterKey": "Master1Subnet", 13
    "ParameterValue": "subnet-<random_string>" 14
  },
  {
    "ParameterKey": "Master2Subnet", 15
    "ParameterValue": "subnet-<random_string>" 16
  },
  {
    "ParameterKey": "MasterSecurityGroupId", 17
    "ParameterValue": "sg-<random_string>" 18
  },
  {
    "ParameterKey": "IgnitionLocation", 19
    "ParameterValue": "https://api-int.<cluster_name>.<domain_name>:22623/config/master"
20
  },
  {
    "ParameterKey": "CertificateAuthorities", 21
    "ParameterValue": "data:text/plain;charset=utf-8;base64,ABC...xYz==" 22
  },
  {
    "ParameterKey": "MasterInstanceProfileName", 23
    "ParameterValue": "<roles_stack>-MasterInstanceProfile-<random_string>" 24
  },
  {
    "ParameterKey": "MasterInstanceType", 25
    "ParameterValue": "m5.xlarge" 26
  },
  {
    "ParameterKey": "AutoRegisterELB", 27
    "ParameterValue": "yes" 28
  },
  {
    "ParameterKey": "RegisterNlbIpTargetsLambdaArn", 29
    "ParameterValue": "arn:aws:lambda:<region>:<account_number>:function:
<dns_stack_name>-RegisterNlbIpTargets-<random_string>" 30
  },
  {
    "ParameterKey": "ExternalApiTargetGroupArn", 31
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Exter-<random_string>" 32
  },
  {
    "ParameterKey": "InternalApiTargetGroupArn", 33
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 34
```

```
  },
  {
    "ParameterKey": "InternalServiceTargetGroupArn", 35
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 36
  }
]
```

1. 클러스터에 대해 Ignition 구성 파일에 인코딩되는 클러스터 인프라의 이름입니다.
2. Ignition 구성 파일 메타데이터에서 추출한 인프라 이름을 **<cluster-name>-<random-string>** 형식으로 지정합니다.
3. 컨트롤 플레인 시스템에 사용할 현재 RHCOS(Red Hat Enterprise Linux CoreOS) AMI입니다.
4. 유효한 **AWS::EC2::Image::Id** 값을 지정합니다.
5. DNS etcd 등록을 수행할지 여부입니다.
6. **yes** 또는 **no**를 지정합니다. **yes**를 지정하는 경우 호스팅 영역 정보를 제공해야 합니다.
7. etcd 대상을 등록할 Route 53 프라이빗 영역 ID입니다.
8. DNS 및 로드 밸런싱을 위해 CloudFormation 템플릿의 출력에서 **PrivateHostedZoneId** 값을 지정합니다.
9. 대상을 등록할 Route 53 영역입니다.
10. **<cluster_name>.<domain_name>**을 지정합니다. 여기서 **<domain_name>**은 클러스터에 대한 **install-config.yaml** 파일을 생성할 때 사용한 Route 53 기본 도메인입니다. AWS 콘솔에 표시되는 후행 마침표(.)는 포함하지 마십시오.
11. 13. 15. 컨트롤 플레인 시스템을 시작하기 위한 서브넷(프라이빗)입니다.
12. 14. 16. DNS 및 로드 밸런싱을 위해 CloudFormation 템플릿의 출력에서 **PrivateSubnets** 값의 서브넷을 지정합니다.
17. 컨트롤 플레인 노드와 연결할 마스터 보안 그룹 ID입니다.
18. 보안 그룹 및 역할에 대한 CloudFormation 템플릿 출력에서 **MasterSecurityGroupId** 값을 지정합니다.
19. 컨트롤 플레인 Ignition 구성 파일을 가져올 위치입니다.
20. 생성된 Ignition 구성 파일 위치(**https://api-int.<cluster_name>.<domain_name>:22623/config/master**)를 지정합니다.
21. 사용할 base64로 인코딩 인증 기관 문자열입니다.
22. 설치 디렉터리에 있는 **master.ign** 파일에서 값을 지정합니다. 이 값은 **data:text/plain;charset=utf-8;base64,ABC...xYz==** 형식의 긴 문자열입니다.
23. 컨트롤 플레인 노드와 연결할 IAM 프로필입니다.
24. 보안 그룹 및 역할에 대한 CloudFormation 템플릿 출력에서 **MasterInstanceProfile** 매개변수 값을 지정합니다.

25 컨트롤 플레인 시스템에 사용할 AWS 인스턴스 유형입니다.

26 허용되는 값:

- **m4.xlarge**
- **m4.2xlarge**
- **m4.4xlarge**
- **m4.10xlarge**
- **m4.16xlarge**
- **m5.xlarge**
- **m5.2xlarge**
- **m5.4xlarge**
- **m5.8xlarge**
- **m5.12xlarge**
- **m5.16xlarge**
- **m5a.xlarge**
- **m5a.2xlarge**
- **m5a.4xlarge**
- **m5a.8xlarge**
- **m5a.12xlarge**
- **m5a.16xlarge**
- **c4.2xlarge**
- **c4.4xlarge**
- **c4.8xlarge**
- **c5.2xlarge**
- **c5.4xlarge**
- **c5.9xlarge**
- **c5.12xlarge**
- **c5.18xlarge**
- **c5.24xlarge**
- **c5a.2xlarge**

- **c5a.4xlarge**
- **c5a.8xlarge**
- **c5a.12xlarge**
- **c5a.16xlarge**
- **c5a.24xlarge**
- **r4.xlarge**
- **r4.2xlarge**
- **r4.4xlarge**
- **r4.8xlarge**
- **r4.16xlarge**
- **r5.xlarge**
- **r5.2xlarge**
- **r5.4xlarge**
- **r5.8xlarge**
- **r5.12xlarge**
- **r5.16xlarge**
- **r5.24xlarge**
- **r5a.xlarge**
- **r5a.2xlarge**
- **r5a.4xlarge**
- **r5a.8xlarge**
- **r5a.12xlarge**
- **r5a.16xlarge**
- **r5a.24xlarge**

27 네트워크 로드 밸런서(NLB) 등록 여부입니다.

28 **yes** 또는 **no**를 지정합니다. **yes**를 지정하는 경우 Lambda ARN(Amazon Resource Name) 값을 제공해야 합니다.

29 NLB IP 대상 등록 lambda 그룹의 ARN입니다.

30 DNS 및 로드 밸런싱을 위해 CloudFormation 템플릿의 출력에서 **RegisterNlbIpTargetsLambda** 값을 지정합니다. 클러스터를 AWS GovCloud 리전에 배포하는 경우 **arn:aws-us-gov**를 사용합니다.

31 외부 API 로드 밸런서 대상 그룹의 ARN입니다.

32 DNS 및 로드 밸런싱의 CloudFormation 템플릿 출력에서 **ExternalApiTargetGroupArn** 값을 지정합니다. 클러스터를 AWS GovCloud 리전에 배포하는 경우 **arn:aws-us-gov**를 사용합니다.

33 내부 API 로드 밸런서 대상 그룹의 ARN입니다.

34 DNS 및 로드 밸런싱의 CloudFormation 템플릿 출력에서 **InternalApiTargetGroupArn** 값을 지정합니다. 클러스터를 AWS GovCloud 리전에 배포하는 경우 **arn:aws-us-gov**를 사용합니다.

35 내부 서비스 로드 밸런서 대상 그룹의 ARN입니다.

36 DNS 및 로드 밸런싱의 CloudFormation 템플릿 출력에서 **InternalServiceTargetGroupArn** 값을 지정합니다. 클러스터를 AWS GovCloud 리전에 배포하는 경우 **arn:aws-us-gov**를 사용합니다.

2. 이 항목의 **컨트롤 플레인 시스템에 대한 CloudFormation 템플릿** 섹션에서 템플릿을 복사하여 사용자 시스템에 YAML 파일로 저장합니다. 이 템플릿은 클러스터에 필요한 컨트롤 플레인 시스템을 설명합니다.

3. **MasterInstanceType** 값으로 **M5** 인스턴스 유형을 지정한 경우 CloudFormation 템플릿의 **MasterInstanceType.AllowedValues** 매개변수에 해당 인스턴스 유형을 추가합니다.

4. CloudFormation 템플릿을 시작하여 컨트롤 플레인 노드를 나타내는 AWS 리소스 스택을 생성합니다.

중요

명령은 한 줄로 입력해야 합니다.

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
```

1 **<name>**은 CloudFormation 스택의 이름입니다(예: **cluster-control-plane**). 클러스터를 제거하는 경우 이 스택의 이름이 필요합니다.

2 **<template>**은 저장한 CloudFormation 템플릿 YAML 파일의 상대 경로 및 이름입니다.

3 **<parameters>**는 CloudFormation 매개변수 JSON 파일의 상대 경로 및 이름입니다.

출력 예

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-control-plane/21c7e2b0-2ee2-11eb-c6f6-0aa34627df4b
```

참고

CloudFormation 템플릿은 세 개의 컨트롤 플레인 노드를 나타내는 스택을 생성합니다.

5. 템플릿 구성 요소가 있는지 확인합니다.

```
$ aws cloudformation describe-stacks --stack-name <name>
```

4.12.16.1. 컨트롤 플레인 시스템에 대한 CloudFormation 템플릿

다음 CloudFormation 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 컨트롤 플레인 시스템을 배포할 수 있습니다.

예 4.40. 컨트롤 플레인 시스템에 대한 CloudFormation 템플릿

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Node Launch (EC2 master instances)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag nodes for the kubelet cloud provider.
    Type: String
  RhcosAmi:
    Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.
    Type: AWS::EC2::Image::Id
  AutoRegisterDNS:
    Default: "yes"
    AllowedValues:
    - "yes"
    - "no"
    Description: Do you want to invoke DNS etcd registration, which requires Hosted Zone
information?
    Type: String
  PrivateHostedZoneId:
    Description: The Route53 private zone ID to register the etcd targets with, such as
Z21IXYZABCZ2A4.
    Type: String
  PrivateHostedZoneName:
    Description: The Route53 zone to register the targets with, such as cluster.example.com. Omit
the trailing period.
    Type: String
  Master0Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  Master1Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  Master2Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  MasterSecurityGroupId:
    Description: The master security group ID to associate with master nodes.
    Type: AWS::EC2::SecurityGroup::Id
  IgnitionLocation:
```

```
    Default: https://api-int.$CLUSTER_NAME.$DOMAIN:22623/config/master
    Description: Ignition config file location.
    Type: String
  CertificateAuthorities:
    Default: data:text/plain;charset=utf-8;base64,ABC...xYz==
    Description: Base64 encoded certificate authority string to use.
    Type: String
  MasterInstanceProfileName:
    Description: IAM profile to associate with master nodes.
    Type: String
  MasterInstanceType:
    Default: m5.xlarge
    Type: String

  AutoRegisterELB:
    Default: "yes"
    AllowedValues:
    - "yes"
    - "no"
    Description: Do you want to invoke NLB registration, which requires a Lambda ARN parameter?
    Type: String
  RegisterNlbIpTargetsLambdaArn:
    Description: ARN for NLB IP target registration lambda. Supply the value from the cluster
infrastructure or select "no" for AutoRegisterELB.
    Type: String
  ExternalApiTargetGroupArn:
    Description: ARN for external API load balancer target group. Supply the value from the cluster
infrastructure or select "no" for AutoRegisterELB.
    Type: String
  InternalApiTargetGroupArn:
    Description: ARN for internal API load balancer target group. Supply the value from the cluster
infrastructure or select "no" for AutoRegisterELB.
    Type: String
  InternalServiceTargetGroupArn:
    Description: ARN for internal service load balancer target group. Supply the value from the
cluster infrastructure or select "no" for AutoRegisterELB.
    Type: String

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - InfrastructureName
    - Label:
        default: "Host Information"
      Parameters:
      - MasterInstanceType
      - RhcosAmi
      - IgnitionLocation
      - CertificateAuthorities
      - MasterSecurityGroupId
      - MasterInstanceProfileName
    - Label:
        default: "Network Configuration"
```

```
      Parameters:
      - VpcId
      - AllowedBootstrapSshCidr
      - Master0Subnet
      - Master1Subnet
      - Master2Subnet
    - Label:
        default: "DNS"
      Parameters:
      - AutoRegisterDNS
      - PrivateHostedZoneName
      - PrivateHostedZoneId
    - Label:
        default: "Load Balancer Automation"
      Parameters:
      - AutoRegisterELB
      - RegisterNlbIpTargetsLambdaArn
      - ExternalApiTargetGroupArn
      - InternalApiTargetGroupArn
      - InternalServiceTargetGroupArn
    ParameterLabels:
      InfrastructureName:
        default: "Infrastructure Name"
      VpcId:
        default: "VPC ID"
      Master0Subnet:
        default: "Master-0 Subnet"
      Master1Subnet:
        default: "Master-1 Subnet"
      Master2Subnet:
        default: "Master-2 Subnet"
      MasterInstanceType:
        default: "Master Instance Type"
      MasterInstanceProfileName:
        default: "Master Instance Profile Name"
      RhcosAmi:
        default: "Red Hat Enterprise Linux CoreOS AMI ID"
      BootstrapIgnitionLocation:
        default: "Master Ignition Source"
      CertificateAuthorities:
        default: "Ignition CA String"
      MasterSecurityGroupId:
        default: "Master Security Group ID"
      AutoRegisterDNS:
        default: "Use Provided DNS Automation"
      AutoRegisterELB:
        default: "Use Provided ELB Automation"
      PrivateHostedZoneName:
        default: "Private Hosted Zone Name"
      PrivateHostedZoneId:
        default: "Private Hosted Zone ID"

Conditions:
  DoRegistration: !Equals ["yes", !Ref AutoRegisterELB]
  DoDns: !Equals ["yes", !Ref AutoRegisterDNS]
```

```
Resources:
  Master0:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref MasterInstanceProfileName
      InstanceType: !Ref MasterInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "Master0Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"merge":[{"source":"${SOURCE}"}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}"}]}},"version":"3.1.0"}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

  RegisterMaster0:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref ExternalApiTargetGroupArn
      TargetIp: !GetAtt Master0.PrivateIp

  RegisterMaster0InternalApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalApiTargetGroupArn
      TargetIp: !GetAtt Master0.PrivateIp

  RegisterMaster0InternalServiceTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalServiceTargetGroupArn
      TargetIp: !GetAtt Master0.PrivateIp

  Master1:
    Type: AWS::EC2::Instance
```

```
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref MasterInstanceProfileName
      InstanceType: !Ref MasterInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "Master1Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"merge":[{"source":"${SOURCE}"}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}"}]}},"version":"3.1.0"}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

  RegisterMaster1:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref ExternalApiTargetGroupArn
      TargetIp: !GetAtt Master1.PrivateIp

  RegisterMaster1InternalApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalApiTargetGroupArn
      TargetIp: !GetAtt Master1.PrivateIp

  RegisterMaster1InternalServiceTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalServiceTargetGroupArn
      TargetIp: !GetAtt Master1.PrivateIp

  Master2:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
```

```
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref MasterInstanceProfileName
      InstanceType: !Ref MasterInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "Master2Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"merge":[{"source":"${SOURCE}"}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}"}]}},"version":"3.1.0"}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

  RegisterMaster2:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref ExternalApiTargetGroupArn
      TargetIp: !GetAtt Master2.PrivateIp

  RegisterMaster2InternalApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalApiTargetGroupArn
      TargetIp: !GetAtt Master2.PrivateIp

  RegisterMaster2InternalServiceTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalServiceTargetGroupArn
      TargetIp: !GetAtt Master2.PrivateIp

  EtcdSrvRecords:
    Condition: DoDns
    Type: AWS::Route53::RecordSet
    Properties:
      HostedZoneId: !Ref PrivateHostedZoneId
      Name: !Join [".", ["_etcd-server-ssl._tcp", !Ref PrivateHostedZoneName]]
      ResourceRecords:
      - !Join [
```

```
        " ",
        ["0 10 2380", !Join [".", ["etcd-0", !Ref PrivateHostedZoneName]]],
      ]
      - !Join [
        " ",
        ["0 10 2380", !Join [".", ["etcd-1", !Ref PrivateHostedZoneName]]],
      ]
      - !Join [
        " ",
        ["0 10 2380", !Join [".", ["etcd-2", !Ref PrivateHostedZoneName]]],
      ]
      TTL: 60
      Type: SRV

  Etcd0Record:
    Condition: DoDns
    Type: AWS::Route53::RecordSet
    Properties:
      HostedZoneId: !Ref PrivateHostedZoneId
      Name: !Join [".", ["etcd-0", !Ref PrivateHostedZoneName]]
      ResourceRecords:
      - !GetAtt Master0.PrivateIp
      TTL: 60
      Type: A

  Etcd1Record:
    Condition: DoDns
    Type: AWS::Route53::RecordSet
    Properties:
      HostedZoneId: !Ref PrivateHostedZoneId
      Name: !Join [".", ["etcd-1", !Ref PrivateHostedZoneName]]
      ResourceRecords:
      - !GetAtt Master1.PrivateIp
      TTL: 60
      Type: A

  Etcd2Record:
    Condition: DoDns
    Type: AWS::Route53::RecordSet
    Properties:
      HostedZoneId: !Ref PrivateHostedZoneId
      Name: !Join [".", ["etcd-2", !Ref PrivateHostedZoneName]]
      ResourceRecords:
      - !GetAtt Master2.PrivateIp
      TTL: 60
      Type: A

Outputs:
  PrivateIPs:
    Description: The control-plane node private IP addresses.
    Value:
      !Join [
        ",",
        [!GetAtt Master0.PrivateIp, !GetAtt Master1.PrivateIp, !GetAtt Master2.PrivateIp]
      ]
```

추가 리소스

- AWS CloudFormation 콘솔로 이동하여 생성하는 CloudFormation 스택에 대한 세부 정보를 볼 수 있습니다.

4.12.17. AWS에서 작업자 노드 생성

클러스터가 사용할 작업자 노드를 AWS(Amazon Web Services)에서 생성할 수 있습니다.

제공된 CloudFormation 템플릿과 사용자 정의 매개변수 파일을 사용하여 작업자 노드를 나타내는 AWS 리소스 스택을 생성할 수 있습니다.

중요

CloudFormation 템플릿은 하나의 작업자 노드를 나타내는 스택을 생성합니다. 각 작업자 노드의 스택을 생성해야 합니다.

참고

작업자 노드를 생성하는 데 제공된 CloudFormation 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다.. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- AWS 계정을 구성했습니다.
- **aws 구성**을 실행하여 AWS 키와 리전을 로컬 AWS 프로필에 추가하셨습니다.
- 클러스터에 대한 Ignition 구성 파일을 생성하셨습니다.
- AWS에서 VPC 및 관련 서브넷을 생성하고 구성하셨습니다.
- AWS에서 DNS, 로드 밸런서 및 리스너를 생성하고 구성하셨습니다.
- AWS에서 클러스터에 필요한 보안 그룹 및 역할을 생성하셨습니다.
- 부트스트랩 시스템을 생성하셨습니다.
- 컨트롤 플레인 시스템을 생성하셨습니다.

프로세스

1. CloudFormation 템플릿에 필요한 매개변수 값이 포함된 JSON 파일을 만듭니다.

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "RhcosAmi", 3
    "ParameterValue": "ami-<random_string>" 4
  },
  {
```

```
  "ParameterKey": "Subnet", 5
  "ParameterValue": "subnet-<random_string>" 6
 },
 {
  "ParameterKey": "WorkerSecurityGroupId", 7
  "ParameterValue": "sg-<random_string>" 8
 },
 {
  "ParameterKey": "IgnitionLocation", 9
  "ParameterValue": "https://api-int.<cluster_name>.<domain_name>:22623/config/worker"
10
 },
 {
  "ParameterKey": "CertificateAuthorities", 11
  "ParameterValue": "" 12
 },
 {
  "ParameterKey": "WorkerInstanceProfileName", 13
  "ParameterValue": "" 14
 },
 {
  "ParameterKey": "WorkerInstanceType", 15
  "ParameterValue": "m4.2xlarge" 16
 }
]
```

1. 클러스터에 대해 Ignition 구성 파일에 인코딩되는 클러스터 인프라의 이름입니다.
2. Ignition 구성 파일 메타데이터에서 추출한 인프라 이름을 **<cluster-name>-<random-string>** 형식으로 지정합니다.
3. 작업자 노드에 사용할 현재 RHCOS(Red Hat Enterprise Linux CoreOS) AMI입니다.
4. 유효한 **AWS::EC2::Image::Id** 값을 지정합니다.
5. 작업자 노드를 시작하기 위한 서브넷(프라이빗)입니다.
6. DNS 및 로드 밸런싱을 위해 CloudFormation 템플릿의 출력에서 **PrivateSubnets** 값의 서브넷을 지정합니다.
7. 작업자 노드와 연결할 작업자 보안 그룹 ID입니다.
8. 보안 그룹 및 역할에 대한 CloudFormation 템플릿 출력에서 **WorkerSecurityGroupId** 값을 지정합니다.
9. 부트스트랩 Ignition 구성 파일을 가져올 위치입니다.
10. 생성된 Ignition 구성 위치(**https://api-int.<cluster_name>.<domain_name>:22623/config/worker**)를 지정합니다.
11. 사용할 base64로 인코딩 인증 기관 문자열입니다.
12. 설치 디렉터리에 있는 **worker.ign** 파일에서 값을 지정합니다. 이 값은 **data:text/plain;charset=utf-8;base64,ABC...xYz==** 형식의 긴 문자열입니다.

13 작업자 노드와 연결할 IAM 프로필입니다.

14 보안 그룹 및 역할에 대한 CloudFormation 템플릿 출력에서 **WorkerInstanceProfile** 매개변수 값을 지정합니다.

15 컨트롤 플레인 시스템에 사용할 AWS 인스턴스 유형입니다.

16 허용되는 값:

- **m4.large**
- **m4.xlarge**
- **m4.2xlarge**
- **m4.4xlarge**
- **m4.10xlarge**
- **m4.16xlarge**
- **m5.large**
- **m5.xlarge**
- **m5.2xlarge**
- **m5.4xlarge**
- **m5.8xlarge**
- **m5.12xlarge**
- **m5.16xlarge**
- **m5a.large**
- **m5a.xlarge**
- **m5a.2xlarge**
- **m5a.4xlarge**
- **m5a.8xlarge**
- **m5a.12xlarge**
- **m5a.16xlarge**
- **c4.large**
- **c4.xlarge**
- **c4.2xlarge**
- **c4.4xlarge**
- **c4.8xlarge**

- **c5.large**
- **c5.xlarge**
- **c5.2xlarge**
- **c5.4xlarge**
- **c5.9xlarge**
- **c5.12xlarge**
- **c5.18xlarge**
- **c5.24xlarge**
- **c5a.large**
- **c5a.xlarge**
- **c5a.2xlarge**
- **c5a.4xlarge**
- **c5a.8xlarge**
- **c5a.12xlarge**
- **c5a.16xlarge**
- **c5a.24xlarge**
- **r4.large**
- **r4.xlarge**
- **r4.2xlarge**
- **r4.4xlarge**
- **r4.8xlarge**
- **r4.16xlarge**
- **r5.large**
- **r5.xlarge**
- **r5.2xlarge**
- **r5.4xlarge**
- **r5.8xlarge**
- **r5.12xlarge**
- **r5.16xlarge**

- **r5.24xlarge**
- **r5a.large**
- **r5a.xlarge**
- **r5a.2xlarge**
- **r5a.4xlarge**
- **r5a.8xlarge**
- **r5a.12xlarge**
- **r5a.16xlarge**
- **r5a.24xlarge**
- **t3.large**
- **t3.xlarge**
- **t3.2xlarge**
- **t3a.large**
- **t3a.xlarge**
- **t3a.2xlarge**

2. 이 항목의 **작업자 시스템에 대한 CloudFormation 템플릿** 섹션에서 템플릿을 복사하여 사용자 시스템에 YAML 파일로 저장합니다. 이 템플릿은 클러스터에 필요한 네트워킹 개체 및 로드 밸런서를 설명합니다.

3. 선택 사항: **WorkerInstanceType** 값으로 M **5** 인스턴스 유형을 지정한 경우 CloudFormation 템플릿의 **WorkerInstanceType.AllowedValues** 매개변수에 해당 인스턴스 유형을 추가합니다.

4. 선택 사항: AWS Marketplace 이미지로 배포하는 경우 서브스크립션에서 얻은 AMI ID로 **Worker0.type.properties.ImageID** 매개변수를 업데이트합니다.

5. CloudFormation 템플릿을 사용하여 작업자 노드를 나타내는 AWS 리소스 스택을 생성합니다.

중요

명령은 한 줄로 입력해야 합니다.

```
$ aws cloudformation create-stack --stack-name <name> (1)
    --template-body file://<template>.yaml \ (2)
    --parameters file://<parameters>.json (3)
```

(1) **<name>**은 CloudFormation 스택의 이름입니다(예: **cluster-worker-1**). 클러스터를 제거하는 경우 이 스택의 이름이 필요합니다.

(2) **<template>**은 저장한 CloudFormation 템플릿 YAML 파일의 상대 경로 및 이름입니다.

3 **<parameters>**는 CloudFormation 매개변수 JSON 파일의 상대 경로 및 이름입니다.

출력 예

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-worker-1/729ee301-1c2a-11eb-348f-sd9888c65b59
```

참고

CloudFormation 템플릿은 하나의 작업자 노드를 나타내는 스택을 생성합니다.

6. 템플릿 구성 요소가 있는지 확인합니다.

```
$ aws cloudformation describe-stacks --stack-name <name>
```

7. 클러스터에 충분한 작업자 시스템을 생성할 때까지 계속해서 작업자 스택을 생성합니다. 동일한 템플릿 및 매개변수 파일을 참조하고 다른 스택 이름을 지정하여 추가 작업자 스택을 생성할 수 있습니다.

중요

작업자 시스템을 두 개 이상 생성해야 하므로 이 CloudFormation 템플릿을 사용하는 스택을 두 개 이상 생성해야 합니다.

4.12.17.1. 작업자 시스템용 CloudFormation 템플릿

다음 CloudFormation 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 작업자 시스템을 배포할 수 있습니다.

예 4.41. 작업자 시스템용 CloudFormation 템플릿

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Node Launch (EC2 worker instance)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag nodes for the kubelet cloud provider.
    Type: String
  RhcosAmi:
    Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.
    Type: AWS::EC2::Image::Id
  Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  WorkerSecurityGroupId:
    Description: The master security group ID to associate with master nodes.
    Type: AWS::EC2::SecurityGroup::Id
```

```
  IgnitionLocation:
    Default: https://api-int.$CLUSTER_NAME.$DOMAIN:22623/config/worker
    Description: Ignition config file location.
    Type: String
  CertificateAuthorities:
    Default: data:text/plain;charset=utf-8;base64,ABC...xYz==
    Description: Base64 encoded certificate authority string to use.
    Type: String
  WorkerInstanceProfileName:
    Description: IAM profile to associate with master nodes.
    Type: String
  WorkerInstanceType:
    Default: m5.large
    Type: String

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - InfrastructureName
    - Label:
        default: "Host Information"
      Parameters:
      - WorkerInstanceType
      - RhcosAmi
      - IgnitionLocation
      - CertificateAuthorities
      - WorkerSecurityGroupId
      - WorkerInstanceProfileName
    - Label:
        default: "Network Configuration"
      Parameters:
      - Subnet
    ParameterLabels:
      Subnet:
        default: "Subnet"
      InfrastructureName:
        default: "Infrastructure Name"
      WorkerInstanceType:
        default: "Worker Instance Type"
      WorkerInstanceProfileName:
        default: "Worker Instance Profile Name"
      RhcosAmi:
        default: "Red Hat Enterprise Linux CoreOS AMI ID"
      IgnitionLocation:
        default: "Worker Ignition Source"
      CertificateAuthorities:
        default: "Ignition CA String"
      WorkerSecurityGroupId:
        default: "Worker Security Group ID"

Resources:
  Worker0:
    Type: AWS::EC2::Instance
```

```
      Properties:
        ImageId: !Ref RhcosAmi
        BlockDeviceMappings:
        - DeviceName: /dev/xvda
          Ebs:
            VolumeSize: "120"
            VolumeType: "gp2"
        IamInstanceProfile: !Ref WorkerInstanceProfileName
        InstanceType: !Ref WorkerInstanceType
        NetworkInterfaces:
        - AssociatePublicIpAddress: "false"
          DeviceIndex: "0"
          GroupSet:
          - !Ref "WorkerSecurityGroupId"
          SubnetId: !Ref "Subnet"
        UserData:
          Fn::Base64: !Sub
          - '{"ignition":{"config":{"merge":[{"source":"${SOURCE}"}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}"}]}},"version":"3.1.0"}}'
          - {
            SOURCE: !Ref IgnitionLocation,
            CA_BUNDLE: !Ref CertificateAuthorities,
          }
        Tags:
        - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
          Value: "shared"

Outputs:
  PrivateIP:
    Description: The compute node private IP address.
    Value: !GetAtt Worker0.PrivateIp
```

추가 리소스

- AWS CloudFormation 콘솔로 이동하여 생성하는 CloudFormation 스택에 대한 세부 정보를 볼 수 있습니다.

4.12.18. 사용자 프로비저닝 인프라로 AWS에서 부트스트랩 시퀀스 초기화

AWS(Amazon Web Services)에서 필요한 인프라를 모두 생성한 후 OpenShift Container Platform 컨트롤 플레인을 초기화하는 부트스트랩 시퀀스를 시작할 수 있습니다.

사전 요구 사항

- AWS 계정을 구성했습니다.
- **aws 구성**을 실행하여 AWS 키와 리전을 로컬 AWS 프로필에 추가하셨습니다.
- 클러스터에 대한 Ignition 구성 파일을 생성하셨습니다.
- AWS에서 VPC 및 관련 서브넷을 생성하고 구성하셨습니다.
- AWS에서 DNS, 로드 밸런서 및 리스너를 생성하고 구성하셨습니다.

- AWS에서 클러스터에 필요한 보안 그룹 및 역할을 생성하셨습니다.
- 부트스트랩 시스템을 생성하셨습니다.
- 컨트롤 플레인 시스템을 생성하셨습니다.
- 작업자 노드를 생성하셨습니다.

프로세스

1. 설치 프로그램이 포함된 디렉터리로 변경하고 OpenShift Container Platform 컨트롤 플레인을 초기화하는 부트스트랩 프로세스를 시작합니다.

```
$ ./openshift-install wait-for bootstrap-complete --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2 다른 설치 세부 사항을 보려면 **info** 대신 **warn**, **debug** 또는 **error**를 지정합니다.

출력 예

```
INFO Waiting up to 20m0s for the Kubernetes API at
https://api.mycluster.example.com:6443...
INFO API v1.22.1 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
INFO Time elapsed: 1s
```

FATAL 경고 없이 명령이 종료되면 OpenShift Container Platform 컨트롤 플레인이 초기화된 것입니다.

참고

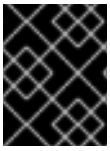
컨트롤 플레인이 초기화된 후 컴퓨팅 노드를 설정하고 Operator 형태로 추가 서비스를 설치합니다.

추가 리소스

- OpenShift Container Platform 설치가 진행되는 동안 설치, 부트스트랩 및 컨트롤 플레인 로그를 모니터링하는 데 대한 자세한 내용은 설치 진행 상황 모니터링 을 참조하십시오.
- 부트스트랩 프로세스와 관련된 문제 해결에 대한 정보는 부트스트랩 노드 진단 데이터 수집 을 참조하십시오.
- AWS EC2 콘솔 을 사용하여 생성된 실행 중인 인스턴스에 대한 세부 정보를 볼 수 있습니다.

4.12.19. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift CLI(**oc**)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 **oc**를 설치할 수 있습니다.



중요

이전 버전의 **oc**를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 **oc**를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치
다음 절차를 사용하여 Linux에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Linux Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. **oc** 바이너리를 **PATH**에 있는 디렉터리에 배치합니다.
PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치
다음 절차에 따라 Windows에 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Windows Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. ZIP 프로그램으로 아카이브의 압축을 풉니다.
5. **oc** 바이너리를 **PATH**에 있는 디렉터리로 이동합니다.
PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치
다음 절차에 따라 macOS에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 MacOSX Client** 항목 옆에 있는 **지금 다운로드를** 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 해제하고 압축을 풉니다.
5. **oc** 바이너리 PATH의 디렉터리로 이동합니다.
   PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

4.12.20. CLI를 사용하여 클러스터에 로그인

클러스터 **kubeconfig** 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. **kubeconfig** 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- **oc** CLI를 설치했습니다.

프로세스

1. **kubeadmin** 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 **oc** 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

4.12.21. 머신의 인증서 서명 요청 승인

클러스터에 시스템을 추가하면 추가한 시스템별로 보류 중인 인증서 서명 요청(CSR)이 두 개씩 생성됩니다. 이러한 CSR이 승인되었는지 확인해야 하며, 필요한 경우 이를 직접 승인해야 합니다. 클라이언트 요청을 먼저 승인한 다음 서버 요청을 승인해야 합니다.

사전 요구 사항

- 클러스터에 시스템을 추가했습니다.

프로세스

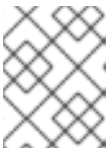
1. 클러스터가 시스템을 인식하는지 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0 Ready    master  63m  v1.22.1
master-1 Ready    master  63m  v1.22.1
master-2 Ready    master  64m  v1.22.1
```

출력에 생성된 모든 시스템이 나열됩니다.



참고

이전 출력에는 일부 CSR이 승인될 때까지 컴퓨팅 노드(작업자 노드라고도 함)가 포함되지 않을 수 있습니다.

2. 보류 중인 CSR을 검토하고 클러스터에 추가한 각 시스템에 대해 **Pending** 또는 **Approved** 상태의 클라이언트 및 서버 요청이 표시되는지 확인합니다.

```
$ oc get csr
```

출력 예

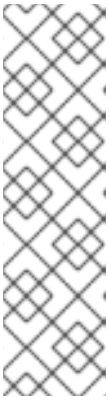
```
NAME        AGE    REQUESTOR                                                    CONDITION
csr-8b2br   15m    system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
csr-8vnps   15m    system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
...
```

예에서는 두 시스템이 클러스터에 참여하고 있습니다. 목록에는 승인된 CSR이 더 많이 나타날 수도 있습니다.

3. CSR이 승인되지 않은 경우, 추가된 시스템에 대한 모든 보류 중인 CSR이 **Pending** 상태로 전환된 후 클러스터 시스템의 CSR을 승인합니다.

참고

CSR은 교체 주기가 자동으로 만료되므로 클러스터에 시스템을 추가한 후 1시간 이내에 CSR을 승인하십시오. 한 시간 내에 승인하지 않으면 인증서가 교체되고 각 노드에 대해 두 개 이상의 인증서가 표시됩니다. 이러한 인증서를 모두 승인해야 합니다. 클라이언트 CSR이 승인되면 Kubelet은 인증서에 대한 보조 CSR을 생성하므로 수동 승인이 필요합니다. 그러면 Kubelet에서 동일한 매개변수를 사용하여 새 인증서를 요청하는 경우 인증서 갱신 요청은 **machine-approver**에 의해 자동으로 승인됩니다.



참고

베어 메탈 및 기타 사용자 프로비저닝 인프라와 같이 머신 API를 사용하도록 활성화되지 않는 플랫폼에서 실행되는 클러스터의 경우 CSR(Kubelet service Certificate Request)을 자동으로 승인하는 방법을 구현해야 합니다. 요청이 승인되지 않으면 API 서버가 kubelet에 연결될 때 서비스 인증서가 필요하므로 **oc exec**, **oc rsh**, **oc logs** 명령을 성공적으로 수행할 수 없습니다. Kubelet 엔드 포인트에 연결하는 모든 작업을 수행하려면 이 인증서 승인이 필요합니다. 이 방법은 새 CSR을 감시하고 CSR이 **system:node** 또는 **system:admin** 그룹의 **node-bootstrapper** 서비스 계정에 의해 제출되었는지 확인하고 노드의 ID를 확인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>**은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

참고

일부 Operator는 일부 CSR이 승인될 때까지 사용할 수 없습니다.

4. 이제 클라이언트 요청이 승인되었으므로 클러스터에 추가한 각 머신의 서버 요청을 검토해야 합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 나머지 CSR이 승인되지 않고 **Pending** 상태인 경우 클러스터 머신의 CSR을 승인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>**은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. 모든 클라이언트 및 서버 CSR이 승인된 후 머신은 **Ready** 상태가 됩니다. 다음 명령을 실행하여 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0 Ready    master  73m  v1.22.1
master-1 Ready    master  73m  v1.22.1
master-2 Ready    master  74m  v1.22.1
worker-0 Ready    worker  11m  v1.22.1
worker-1 Ready    worker  11m  v1.22.1
```

참고

머신이 **Ready** 상태로 전환하는 데 서버 CSR의 승인 후 몇 분이 걸릴 수 있습니다.

추가 정보

- CSR에 대한 자세한 내용은 인증서 서명 요청을 참조하십시오.

4.12.22. Operator의 초기 설정

컨트롤 플레인이 초기화된 후 일부 Operator를 즉시 구성하여 모두 사용 가능하도록 해야 합니다.

사전 요구 사항

- 컨트롤 플레인이 초기화되어 있습니다.

프로세스

1. 클러스터 구성 요소가 온라인 상태인지 확인합니다.

```
$ watch -n5 oc get clusteroperators
```

출력 예

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
```

```
authentication                             4.9.0     True       False       False      19m
baremetal                                  4.9.0     True       False       False      37m
cloud-credential                           4.9.0     True       False       False      40m
cluster-autoscaler                         4.9.0     True       False       False      37m
config-operator                            4.9.0     True       False       False      38m
console                                    4.9.0     True       False       False      26m
csi-snapshot-controller                    4.9.0     True       False       False      37m
dns                                        4.9.0     True       False       False      37m
etcd                                       4.9.0     True       False       False      36m
image-registry                             4.9.0     True       False       False      31m
ingress                                    4.9.0     True       False       False      30m
insights                                   4.9.0     True       False       False      31m
kube-apiserver                             4.9.0     True       False       False      26m
kube-controller-manager                    4.9.0     True       False       False      36m
kube-scheduler                             4.9.0     True       False       False      36m
kube-storage-version-migrator              4.9.0     True       False       False      37m
machine-api                                4.9.0     True       False       False      29m
machine-approver                           4.9.0     True       False       False      37m
machine-config                             4.9.0     True       False       False      36m
marketplace                                4.9.0     True       False       False      37m
monitoring                                 4.9.0     True       False       False      29m
network                                    4.9.0     True       False       False      38m
node-tuning                                4.9.0     True       False       False      37m
openshift-apiserver                        4.9.0     True       False       False      32m
openshift-controller-manager               4.9.0     True       False       False      30m
openshift-samples                          4.9.0     True       False       False      32m
operator-lifecycle-manager                 4.9.0     True       False       False      37m
operator-lifecycle-manager-catalog         4.9.0     True       False       False      37m
operator-lifecycle-manager-packageserver   4.9.0     True       False       False      32m
service-ca                                 4.9.0     True       False       False      38m
storage                                    4.9.0     True       False       False      37m
```

2. 사용할 수 없는 Operator를 구성합니다.

4.12.22.1. 이미지 레지스트리 스토리지 구성

Amazon Web Services는 기본 스토리지를 제공하므로 설치 후 Image Registry Operator를 사용할 수 있습니다. 그러나 Registry Operator가 S3 버킷을 생성하고 스토리지를 자동으로 구성할 수 없는 경우에는 레지스트리 스토리지를 수동으로 구성해야 합니다.

프로덕션 클러스터에 필요한 영구 볼륨을 구성하는 과정의 지침이 표시됩니다. 해당하는 경우, 프로덕션 환경 외 클러스터에서만 사용할 수 있는 저장 위치로서 빈 디렉터리를 구성하는 과정의 지침이 표시됩니다.

업그레이드 중에 **Recreate** 롤아웃 전략을 사용하여 이미지 레지스트리의 블록 스토리지 유형 사용을 허용하기 위한 추가 지침이 제공됩니다.

숨겨진 리전에 OpenShift Container Platform을 배포하도록 AWS에서 사용자 프로비저닝 인프라에 대한 레지스트리 스토리지를 구성할 수 있습니다. 자세한 내용은 AWS 사용자 프로비저닝 인프라용 레지스트리 구성을 참조하십시오.

4.12.22.1.1. 사용자 프로비저닝 인프라로 AWS용 레지스트리 스토리지 구성

설치하는 동안 클라우드 자격 증명만으로도 Amazon S3 버킷을 생성할 수 있으며 Registry Operator가 자동으로 스토리지를 구성합니다.

Registry Operator가 S3 버킷을 생성하고 스토리지를 자동으로 구성할수 없는 경우 다음 프로시저에 따라 S3 버킷을 생성하고 스토리지를 구성할 수 있습니다.

사전 요구 사항

- AWS에 사용자 프로비저닝된 인프라가 있는 클러스터가 있어야 합니다.
- Amazon S3 스토리지의 경우 시크릿에는 두 개의 키가 포함되어야 합니다.
  - **REGISTRY_STORAGE_S3_ACCESSKEY**
  - **REGISTRY_STORAGE_S3_SECRETKEY**

프로세스

Registry Operator가 S3 버킷을 생성하고 스토리지를 자동으로 구성할 수 없는 경우 다음 프로시저를 사용합니다.

1. 1일이 지난 완료되지 않은 다중 파트 업로드를 중단하도록 Bucket Lifecycle Policy 를 설정합니다.
2. **configs.imageregistry.operator.openshift.io/cluster**에 스토리지 설정을 입력합니다.

```
$ oc edit configs.imageregistry.operator.openshift.io/cluster
```

설정 예

```
storage:
  s3:
    bucket: <bucket-name>
    region: <region-name>
```

주의

AWS에서 레지스트리 이미지를 보안을 위해 S3 버킷에 공용 액세스를 차단합니다.

4.12.22.1.2. 프로덕션 환경 외 클러스터에서 이미지 레지스트리의 스토리지 구성

이미지 레지스트리 Operator에 대한 스토리지를 구성해야 합니다. 프로덕션 환경 외 클러스터의 경우, 이미지 레지스트리를 빈 디렉터리로 설정할 수 있습니다. 이렇게 하는 경우 레지스트리를 다시 시작하면 모든 이미지가 손실됩니다.

프로세스

- 이미지 레지스트리 스토리지를 빈 디렉터리로 설정하려면 다음을 수행하십시오.

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
{"storage":{"emptyDir":{}}}}'
```

주의

프로덕션 환경 외 클러스터에 대해서만 이 옵션을 구성하십시오.

Image Registry Operator가 구성 요소를 초기화하기 전에 이 명령을 실행하면 **oc patch** 명령이 실패하며 다음 오류가 발생합니다.

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

몇 분 후에 명령을 다시 실행하십시오.

4.12.23. 부트스트랩 리소스 삭제

클러스터에 대한 초기 Operator 구성을 완료한 후 AWS(Amazon Web Services)에서 부트스트랩 리소스를 제거합니다.

사전 요구 사항

- 클러스터에 대한 초기 Operator 구성을 완료했습니다.

프로세스

1. 부트스트랩 리소스를 삭제합니다. CloudFormation 템플릿을 사용한 경우 해당 스택을 삭제 합니다.
   - AWS CLI를 사용하여 스택 삭제:

     ```
     $ aws cloudformation delete-stack --stack-name <name> 1
     ```

     1 **<name>**은 부트스트랩 스택의 이름입니다.

   - AWS CloudFormation 콘솔을 사용하여 스택을 삭제합니다.

4.12.24. 인그레스 DNS 레코드 생성

DNS 영역 구성을 제거한 경우 인그레스 로드 밸런서를 가리키는 DNS 레코드를 수동으로 만듭니다. 와일드카드 레코드 또는 특정 레코드를 만들 수 있습니다. 다음 프로시저에서는 A 레코드를 사용하지만 CNAME 또는 별칭과 같이 필요한 다른 레코드 유형을 사용할 수 있습니다.

사전 요구 사항

- 프로비저닝한 인프라를 사용하는 AWS(Amazon Web Services)에 OpenShift Container Platform 클러스터를 배포했습니다.
- OpenShift CLI(**oc**)를 설치합니다.
- **jq** CLI를 설치하셨습니다.

- AWS CLI를 다운로드하여 컴퓨터에 설치했습니다. 번들 설치 관리자를 사용하여 AWS CLI 설치(Linux, macOS 또는 Unix)를 참조하십시오.

프로세스

1. 생성할 경로를 결정합니다.
   - 와일드카드 레코드를 만들려면 ***.apps.<cluster_name>. <domain_name>**을 사용합니다. 여기서 **<cluster_name>**은 클러스터 이름이고 **<domain_name>**은 OpenShift Container Platform 클러스터의 Route 53 기본 도메인입니다.
   - 특정 레코드를 만들려면 다음 명령의 출력에 표시된 대로 클러스터가 사용하는 각 경로에 대한 레코드를 만들어야 합니다.

     ```
     $ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}
     {"\n"}{end}{end}' routes
     ```

     출력 예

     ```
     oauth-openshift.apps.<cluster_name>.<domain_name>
     console-openshift-console.apps.<cluster_name>.<domain_name>
     downloads-openshift-console.apps.<cluster_name>.<domain_name>
     alertmanager-main-openshift-monitoring.apps.<cluster_name>.<domain_name>
     grafana-openshift-monitoring.apps.<cluster_name>.<domain_name>
     prometheus-k8s-openshift-monitoring.apps.<cluster_name>.<domain_name>
     ```

2. Ingress Operator 로드 밸런서 상태를 검색하고 **EXTERNAL-IP** 열에 표시된 외부 IP 주소의 값을 확인합니다.

   ```
   $ oc -n openshift-ingress get service router-default
   ```

   출력 예

   ```
   NAME             TYPE           CLUSTER-IP      EXTERNAL-IP                            PORT(S)
   AGE
   router-default   LoadBalancer   172.30.62.215   ab3...28.us-east-2.elb.amazonaws.com
   80:31499/TCP,443:30693/TCP   5m
   ```

3. 로드 밸런서의 호스팅 영역 ID를 찾습니다.

   ```
   $ aws elb describe-load-balancers | jq -r '.LoadBalancerDescriptions[] | select(.DNSName ==
   "<external_ip>").CanonicalHostedZoneNameID' 1
   ```

   1 **<external_ip>**는 가져온 Ingress Operator 로드 밸런서의 외부 IP 주소값을 지정합니다.

   출력 예

   ```
   Z3AADJGX6KTTL2
   ```

   이 명령의 출력은 로드 밸런서 호스팅 영역 ID입니다.

4. 클러스터 도메인의 공개 호스팅 영역 ID를 가져옵니다.

```
$ aws route53 list-hosted-zones-by-name \
        --dns-name "<domain_name>" \ 1
        --query 'HostedZones[? Config.PrivateZone != `true` && Name ==
`<domain_name>.`].Id' 2
        --output text
```

1 2 **<domain_name>**은 OpenShift Container Platform 클러스터의 Route 53 기본 도메인을 지정합니다.

출력 예

```
/hostedzone/Z3URY6TWQ91KVV
```

도메인의 공개 호스팅 영역 ID가 명령 출력에 표시됩니다. 이 예에서는 **Z3URY6TWQ91KVV**입니다.

5. 프라이빗 영역에 별칭 레코드를 추가합니다.

```
$ aws route53 change-resource-record-sets --hosted-zone-id "<private_hosted_zone_id>" --
change-batch '{ 1
>   "Changes": [
>     {
>       "Action": "CREATE",
>       "ResourceRecordSet": {
>         "Name": "\\052.apps.<cluster_domain>", 2
>         "Type": "A",
>         "AliasTarget":{
>           "HostedZoneId": "<hosted_zone_id>", 3
>           "DNSName": "<external_ip>.", 4
>           "EvaluateTargetHealth": false
>         }
>       }
>     }
>   ]
> }'
```

1 **<private_hosted_zone_id>**는 DNS 및 로드 밸런싱에 대한 CloudFormation 템플릿 출력의 값을 지정합니다.

2 **<cluster_domain>**은 OpenShift Container Platform 클러스터와 함께 사용하는 도메인 또는 하위 도메인을 지정합니다.

3 **<hosted_zone_id>**는 가져온 로드 밸런서의 공개 호스팅 영역 ID를 지정합니다.

4 **<external_ip>**는 Ingress Operator 로드 밸런서의 외부 IP 주소값을 지정합니다. 이 매개변수 값에 후행 마침표(**.**)을 포함시켜야 합니다.

6. 퍼블릭 영역에 레코드를 추가합니다.

```
$ aws route53 change-resource-record-sets --hosted-zone-id "<public_hosted_zone_id>"" --
change-batch '{ 1
>   "Changes": [
```

```
>    {
>      "Action": "CREATE",
>      "ResourceRecordSet": {
>        "Name": "\\052.apps.<cluster_domain>", 2
>        "Type": "A",
>        "AliasTarget":{
>          "HostedZoneId": "<hosted_zone_id>", 3
>          "DNSName": "<external_ip>.", 4
>          "EvaluateTargetHealth": false
>        }
>      }
>    }
>  ]
>}'
```

1. **<public_hosted_zone_id>**는 도메인의 퍼블릭 호스팅 영역을 지정합니다.
2. **<cluster_domain>**은 OpenShift Container Platform 클러스터와 함께 사용하는 도메인 또는 하위 도메인을 지정합니다.
3. **<hosted_zone_id>**는 가져온 로드 밸런서의 공개 호스팅 영역 ID를 지정합니다.
4. **<external_ip>**는 Ingress Operator 로드 밸런서의 외부 IP 주소값을 지정합니다. 이 매개변수 값에 후행 마침표(**.**)을 포함시켜야 합니다.

4.12.25. 사용자 프로비저닝 인프라에서 AWS 설치 완료

AWS(Amazon Web Service) 사용자 프로비저닝 인프라에서 OpenShift Container Platform 설치를 시작한 후 배포가 완료될 때까지 모니터링합니다.

사전 요구 사항

- 사용자 프로비저닝 AWS 인프라에서 OpenShift Container Platform 클러스터에 대한 부트스트랩 노드를 제거하셨습니다.
- **oc** CLI를 설치했습니다.

프로세스

- 설치 프로그램이 포함된 디렉터리에서 클러스터 설치를 완료합니다.

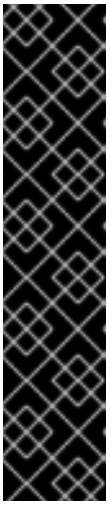
```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1. **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

출력 예

```
INFO Waiting up to 40m0s for the cluster at https://api.mycluster.example.com:6443 to
initialize...
INFO Waiting up to 10m0s for the openshift-console route to be created...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
```

```
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Fe5en-ymBEc-
Wt6NL"
INFO Time elapsed: 1s
```



중요

- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

4.12.26. 웹 콘솔을 사용하여 클러스터에 로그인

kubeadmin 사용자는 OpenShift Container Platform 설치 후 기본적으로 존재합니다. OpenShift Container Platform 웹 콘솔을 사용하여 **kubeadmin** 사용자로 클러스터에 로그인할 수 있습니다.

사전 요구 사항

- 설치 호스트에 대한 액세스 권한이 있어야 합니다.
- 클러스터 설치를 완료했으며 모든 클러스터 Operator를 사용할 수 있습니다.

프로세스

1. 설치 호스트의 **kubeadmin-password** 파일에서 **kubeadmin** 사용자의 암호를 가져옵니다.

```
$ cat <installation_directory>/auth/kubeadmin-password
```

참고

대안으로 설치 호스트의 **<installation_directory>/.openshift_install.log** 로그 파일에서 **kubeadmin** 암호를 가져올 수 있습니다.

2. OpenShift Container Platform 웹 콘솔 경로를 나열합니다.

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```

참고

대안으로 설치 호스트의 **<installation_directory>/.openshift_install.log** 로그 파일에서 OpenShift Container Platform 경로를 가져올 수 있습니다.

출력 예

```
console    console-openshift-console.apps.<cluster_name>.<base_domain>          console
https   reencrypt/Redirect   None
```

3. 웹 브라우저의 이전 명령 출력에 자세히 설명된 경로로 이동하고 **kubeadmin** 사용자로 로그인합니다.

추가 리소스

- OpenShift Container Platform 웹 콘솔 액세스 및 이해에 대한 자세한 내용은 웹 콘솔에 액세스를 참조하십시오.

4.12.27. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

4.12.28. 추가 리소스

- AWS CloudFormation 스택에 대한 자세한 내용은 AWS 문서의 스택 작업을 참조하십시오.

4.12.29. 다음 단계

- 설치를 확인합니다.
- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.
- 필요한 경우 클라우드 공급자 인증 정보를 제거 할 수 있습니다.

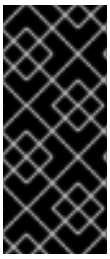
4.13. 사용자 프로비저닝 인프라가 있는 제한된 네트워크에서 AWS에 클러스터 설치

OpenShift Container Platform 4.9 버전에서는 사용자가 제공하는 인프라 및 설치 릴리스 콘텐츠의 내부 미러를 사용하는 AWS(Amazon Web Services)에 클러스터를 설치할 수 있습니다.

중요

미러링된 설치 릴리스 컨텐츠를 사용하여 OpenShift Container Platform 클러스터를 설치할 수는 있지만 AWS API를 사용하려면 여전히 클러스터에 인터넷 액세스가 필요합니다.

이 인프라를 생성하는 한 가지 방법은 제공된 CloudFormation 템플릿을 사용하는 것입니다. 템플릿을 수정하여 인프라를 사용자 지정하거나 포함된 정보를 사용하여 회사 정책에 따라 AWS 개체를 생성할 수 있습니다.



중요

사용자가 프로비저닝한 인프라 설치를 수행하는 단계는 예시용으로만 제공됩니다. 사용자가 제공하는 인프라를 사용하여 클러스터를 설치하려면 클라우드 공급자 및 OpenShift Container Platform 설치 프로세스에 대한 정보가 필요합니다. 안내된 단계를 수행하거나 자체 모델링에 유용한 몇 가지 CloudFormation 템플릿이 제공됩니다. 여러 다른 방법을 통해 필요한 리소스를 자유롭게 생성할 수도 있습니다. 템플릿은 예시일 뿐입니다.

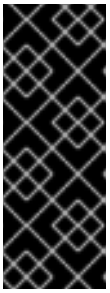
4.13.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비 에 대한 문서를 읽습니다.
- 미러 호스트에 미러 레지스트리를 생성 하고 사용 중인 OpenShift Container Platform 버전의 **imageContentSources** 데이터를 가져옵니다.

중요

미러 호스트에 설치 미디어가 있으므로 해당 컴퓨터를 사용하여 모든 설치 단계를 완료하십시오.

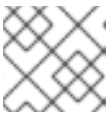
- 클러스터를 호스팅할 AWS 계정을 구성 했습니다.



중요

컴퓨터에 AWS 프로필이 저장되어 있는 경우 다단계 인증 장치를 사용하는 동안 생성한 임시 세션 토큰을 해당 프로필이 사용해서는 안 됩니다. 클러스터는 클러스터의 전체 수명 동안 현재 AWS 자격 증명을 계속 사용하여 AWS 리소스를 생성하므로 키 기반의 장기 자격 증명을 사용해야 합니다. 적절한 키를 생성하려면 AWS 문서의 IAM 사용자의 액세스 키 관리 를 참조하십시오. 설치 프로그램을 실행할 때 키를 제공할 수 있습니다.

- AWS CLI를 다운로드하여 컴퓨터에 설치했습니다. AWS 문서의 번들 설치 관리자를 사용하여 AWS CLI 설치(Linux, macOS 또는 Unix)를 참조하십시오.
- 방화벽을 사용하며 Telemetry 서비스를 사용할 예정인 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성해야 합니다.



참고

프록시를 구성하는 경우에도 해당 사이트 목록을 검토하십시오.

- 사용자의 환경에서 클라우드 ID 및 액세스 관리(IAM) API에 액세스할 수 없거나 **kube-system** 네임스페이스에 관리자 수준의 인증 정보 시크릿을 저장하지 않으려면 IAM 인증 정보 수동 생성 및 유지 관리를 참조하십시오.

4.13.2. 네트워크가 제한된 환경에서의 설치 정보

OpenShift Container Platform 4.9에서는 소프트웨어 구성 요소를 받기 위한 인터넷 접속이 필요하지 않은 설치를 수행할 수 있습니다. 제한된 네트워크 설치는 클러스터를 설치하는 클라우드 플랫폼에 따라 설치 관리자 프로비저닝 인프라 또는 사용자 프로비저닝 인프라를 사용하여 완료할 수 있습니다.

클라우드 플랫폼에 제한된 네트워크 설치를 수행하는 방법을 선택해도 클라우드 API에 액세스는 가능해야 합니다. Amazon Web Service의 Route 53 DNS 및 IAM 서비스와 같은 일부 클라우드 기능에는 인터넷 액세스가 필요합니다. 사용 중인 네트워크에 따라 베어메탈 하드웨어 또는 VMware vSphere에 설치하기 위해 필요한 인터넷 액세스가 줄어들 수 있습니다.

제한된 네트워크 설치를 완료하려면 OpenShift Container Platform 레지스트리의 내용을 미러링하고 설치 미디어를 포함할 레지스트리를 생성해야 합니다. 인터넷과 폐쇄 네트워크에 모두 액세스하거나 제한 사항을 따르는 다른 방법을 통해 미러 호스트에 레지스트리를 생성할 수 있습니다.

중요

사용자 프로비저닝 설치의 구성이 복잡하므로 사용자 프로비저닝 인프라를 사용하여 제한된 네트워크 설치를 시도하기 전에 표준 사용자 프로비저닝 인프라 설치를 완료하는 것이 좋습니다. 이 테스트 설치를 완료하면 제한된 네트워크에 설치하는 동안 발생할 수 있는 문제를 보다 쉽게 파악 및 해결할 수 있습니다.

4.13.2.1. 추가 제한

제한된 네트워크의 클러스터에는 다음과 같은 추가 제한이 있습니다.

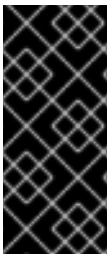
- **ClusterVersion** 상태에 **사용 가능한 업데이트를 검색할 수 없음** 오류가 포함되어 있습니다.
- 기본적으로 필요한 이미지 스트림 태그에 액세스할 수 없기 때문에 개발자 카탈로그의 내용을 사용할 수 없습니다.

4.13.3. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하기 위해 필요한 이미지를 받으려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.



중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

4.13.4. 사용자 프로비저닝 인프라를 포함한 클러스터의 시스템 요구사항

사용자 프로비저닝 인프라가 포함된 클러스터의 경우, 필요한 모든 시스템을 배포해야 합니다.

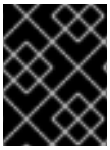
이 섹션에서는 사용자 프로비저닝 인프라에 OpenShift Container Platform을 배포해야 하는 요구 사항에 대해 설명합니다.

4.13.4.1. 클러스터 설치에 필요한 시스템

최소 OpenShift Container Platform 클러스터에 다음과 같은 호스트가 필요합니다.

표 4.44. 최소 필수 호스트

호스트	설명
임시 부트스트랩 시스템 한 개	컨트롤 플레인 시스템 세 개에 OpenShift Container Platform 클러스터를 배포하기 위한 부트스트랩 시스템이 클러스터에 필요합니다. 클러스터를 설치한 후 부트스트랩 시스템을 제거할 수 있습니다.
컨트롤 플레인 시스템 세 개	컨트롤 플레인 시스템은 컨트롤 플레인을 구성하는 Kubernetes 및 OpenShift Container Platform 서비스를 실행합니다.
두 개 이상의 컴퓨팅 시스템(작업자 시스템이라고도 함).	OpenShift Container Platform 사용자가 요청한 워크로드는 컴퓨팅 머신에서 실행됩니다.



중요

클러스터의 고가용성을 유지하려면 이러한 클러스터 시스템에 대해 별도의 물리적 호스트를 사용하십시오.

부트스트랩, 컨트롤 플레인 시스템은 운영 체제로 RHCOS (Red Hat Enterprise Linux CoreOS)를 사용해야 합니다. 그러나 컴퓨팅 머신은 RHCOS(Red Hat Enterprise Linux CoreOS), RHEL (Red Hat Enterprise Linux) 7.9 또는 RHEL 8.4 중에서 선택할 수 있습니다.

RHCOS는 RHEL 8(Red Hat Enterprise Linux)을 기반으로하며 모든 하드웨어 인증 및 요구사항을 모두 이어받습니다. Red Hat Enterprise Linux 기술 기능 및 제한 을 참조하십시오.

4.13.4.2. 클러스터 설치를 위한 최소 리소스 요구 사항

각 클러스터 시스템이 다음과 같은 최소 요구사항을 충족해야 합니다.

표 4.45. 최소 리소스 요구사항

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
부트스트랩	RHCOS	4	16GB	100GB	300
컨트롤 플레인	RHCOS	4	16GB	100GB	300
컴퓨팅	RHCOS, RHEL 7.9 또는 RHEL 8.4 [3]	2	8GB	100GB	300

1. SMT(동시 멀티스레딩) 또는 하이퍼 스레딩이 활성화되지 않은 경우 하나의 vCPU는 하나의 물리적 코어와 동일합니다. 활성화하면 다음과 같은 공식을 사용하여 해당 비율을 계산합니다. (코어당 스레드 수 × 코어 수) × 소켓 수 = vCPU 수

2. OpenShift Container Platform 및 Kubernetes는 디스크 성능에 민감하며 특히 10ms p99 fsync 기간이 필요한 컨트롤 플레인 노드의 etcd에 더 빠른 스토리지가 권장됩니다. 많은 클라우드 플랫폼에서 스토리지 크기와 IOPS를 함께 확장되므로 충분한 성능을 얻으려면 스토리지 볼륨을 과도하게 할당해야 할 수 있습니다.

3. 사용자가 프로비저닝한 모든 설치와 마찬가지로 클러스터에서 RHEL 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. RHEL 7 컴퓨팅 머신 사용은 더 이상 사용되지 않으며 향후 OpenShift Container Platform 4 릴리스에서 제거될 예정입니다.

4.13.4.3. 인증서 서명 요청 관리

사용자가 프로비저닝하는 인프라를 사용하는 경우 자동 시스템 관리 기능으로 인해 클러스터의 액세스가 제한되므로 설치한 후 클러스터 인증서 서명 요청(CSR)을 승인하는 메커니즘을 제공해야 합니다. **kube-controller-manager**는 kubelet 클라이언트 CSR만 승인합니다. **machine-approver**는 올바른 시스템에서 발행한 요청인지 확인할 수 없기 때문에 kubelet 자격 증명을 사용하여 요청하는 서비스 인증서의 유효성을 보장할 수 없습니다. kubelet 서빙 인증서 요청의 유효성을 확인하고 요청을 승인하는 방법을 결정하여 구현해야 합니다.

4.13.4.4. 지원되는 AWS 머신 유형

다음 AWS(Amazon Web Services) 인스턴스 유형은 OpenShift Container Platform에서 지원됩니다.

예 4.42. 시스템의 인스턴스 유형

인스턴스 유형	부트스트랩	컨트롤 플레인	컴퓨팅
i3.large	x		
m4.large			x
m4.xlarge		x	x
m4.2xlarge		x	x
m4.4xlarge		x	x
m4.10xlarge		x	x
m4.16xlarge		x	x
m5.large			x
m5.xlarge		x	x
m5.2xlarge		x	x

인스턴스 유형	부트스트랩	컨트롤 플레인	컴퓨팅
m5.4xlarge		x	x
m5.8xlarge		x	x
m5.12xlarge		x	x
m5.16xlarge		x	x
m5a.large			x
m5a.xlarge		x	x
m5a.2xlarge		x	x
m5a.4xlarge		x	x
m5a.8xlarge		x	x
m5a.12xlarge		x	x
m5a.16xlarge		x	x
m6i.xlarge		x	x
m6i.2xlarge		x	x
m6i.4xlarge		x	x
m6i.8xlarge		x	x
m6i.16xlarge		x	x
c4.2xlarge		x	x
c4.4xlarge		x	x
c4.8xlarge		x	x
c5.xlarge			x
c5.2xlarge		x	x
c5.4xlarge		x	x

인스턴스 유형	부트스트랩	컨트롤 플레인	컴퓨팅
c5.9xlarge		x	x
c5.12xlarge		x	x
c5.18xlarge		x	x
c5.24xlarge		x	x
c5a.xlarge			x
c5a.2xlarge		x	x
c5a.4xlarge		x	x
c5a.8xlarge		x	x
c5a.12xlarge		x	x
c5a.16xlarge		x	x
c5a.24xlarge		x	x
r4.large			x
r4.xlarge		x	x
r4.2xlarge		x	x
r4.4xlarge		x	x
r4.8xlarge		x	x
r4.16xlarge		x	x
r5.large			x
r5.xlarge		x	x
r5.2xlarge		x	x
r5.4xlarge		x	x
r5.8xlarge		x	x
r5.12xlarge		x	x

인스턴스 유형	부트스트랩	컨트롤 플레인	컴퓨팅
r5.16xlarge		x	x
r5.24xlarge		x	x
r5a.large			x
r5a.xlarge		x	x
r5a.2xlarge		x	x
r5a.4xlarge		x	x
r5a.8xlarge		x	x
r5a.12xlarge		x	x
r5a.16xlarge		x	x
r5a.24xlarge		x	x
t3.large			x
t3.xlarge			x
t3.2xlarge			x
t3a.large			x
t3a.xlarge			x
t3a.2xlarge			x

4.13.5. 필수 AWS 인프라 구성 요소

AWS(Amazon Web Services)의 사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치하려면 시스템과 지원 인프라를 모두 수동으로 생성해야 합니다.

다른 플랫폼의 통합 테스트에 대한 자세한 내용은 OpenShift Container Platform 4.x 통합 테스트 페이지를 참조하십시오.

제공된 CloudFormation 템플릿을 사용하면 다음 구성 요소를 나타내는 AWS 리소스 스택을 생성할 수 있습니다.

- AWS 가상 사설 클라우드(VPC)

- 네트워킹 및 로드 밸런싱 구성 요소
- 보안 그룹 및 역할
- OpenShift Container Platform 부트스트랩 노드
- OpenShift Container Platform 컨트롤 플레인 노드
- OpenShift Container Platform 컴퓨팅 노드

대안으로, 구성 요소를 수동으로 생성하거나 클러스터 요구 사항을 충족하는 기존 인프라를 재사용할 수 있습니다. 구성 요소의 상호 관계에 대한 자세한 내용은 CloudFormation 템플릿을 검토하십시오.

4.13.5.1. 기타 인프라 구성 요소

- A VPC
- DNS 항목
- 로드 밸런서(클래식 또는 네트워크) 및 리스너
- 공개 및 개인 Route 53 영역
- 보안 그룹
- IAM 역할
- S3 버킷

연결이 끊긴 환경에서 작업하거나 프록시를 사용하는 경우 EC2 및 ELB 끝점의 공용 IP 주소에 연결할 수 없습니다. 이러한 끝점에 연결하려면 VPC 끝점을 생성하여 클러스터가 사용 중인 서브넷에 연결해야 합니다. 다음 끝점을 생성합니다.

- **ec2.<region>.amazonaws.com**
- **elasticloadbalancing.<region>.amazonaws.com**
- **s3.<region>.amazonaws.com**

필수 VPC 구성 요소

시스템과의 통신을 허용하는 서브넷과 적합한 VPC를 제공해야 합니다.

구성 요소	AWS 유형	설명
VPC	• **AWS::EC2::VPC** • **AWS::EC2::VPCEndpoint**	클러스터에서 사용할 공용 VPC를 제공해야 합니다. VPC는 각 서브넷의 라우팅 테이블을 참조하는 끝점을 사용하여 S3에서 호스팅되는 레지스트리와의 통신을 개선합니다.

<table>
<tr><th>구성 요소</th><th>AWS 유형</th><th>설명</th></tr>
<tr><td>퍼블릭 서브넷</td><td>AWS::EC2::Subnet
AWS::EC2::SubnetNetworkAclAssociation</td><td>VPC에는 1~3개의 가용성 영역에 대한 퍼블릭 서브넷이 있어야 하며 이 서브넷을 적절한 인그레스 규칙과 연결해야 합니다.</td></tr>
<tr><td>인터넷 게이트웨이</td><td>AWS::EC2::InternetGateway
AWS::EC2::VPCGatewayAttachment
AWS::EC2::RouteTable
AWS::EC2::Route
AWS::EC2::SubnetRouteTableAssociation
AWS::EC2::NatGateway
AWS::EC2::EIP</td><td>공용 경로가 있는 공용 인터넷 게이트웨이가 VPC에 연결되어 있어야 합니다. 제공된 템플릿에서 각 퍼블릭 서브넷에는 EIP 주소를 갖는 NAT 게이트웨이가 있습니다. 이러한 NAT 게이트웨이를 사용하면 프라이빗 서브넷 인스턴스와 같은 클러스터 리소스가 인터넷에 도달할 수 있으므로 일부 제한된 네트워크 또는 프록시 시나리오에는 필요하지 않습니다.</td></tr>
<tr><td>네트워크 액세스 제어</td><td>AWS::EC2::NetworkAcl
AWS::EC2::NetworkAclEntry</td><td>VPC가 다음 포트에 액세스할 수 있어야 합니다.<table><tr><th>포트</th><th>이유</th></tr><tr><td>80</td><td>인바운드 HTTP 트래픽</td></tr><tr><td>443</td><td>인바운드 HTTPS 트래픽</td></tr><tr><td>22</td><td>인바운드 SSH 트래픽</td></tr><tr><td>1024 - 65535</td><td>인바운드 임시 트래픽</td></tr><tr><td>0 - 65535</td><td>아웃바운드 임시 트래픽</td></tr></table></td></tr>
<tr><td>프라이빗 서브넷</td><td>AWS::EC2::Subnet
AWS::EC2::RouteTable
AWS::EC2::SubnetRouteTableAssociation</td><td>VPC에 프라이빗 서브넷이 포함될 수 있습니다. 제공된 CloudFormation 템플릿은 1~3개 가용성 영역의 프라이빗 서브넷을 생성할 수 있습니다. 프라이빗 서브넷을 사용하는 경우 적절한 경로와 테이블을 제공해야 합니다.</td></tr>
</table>

필수 DNS 및 로드 밸런싱 구성 요소

DNS 및 로드 밸런서 구성은 공개 호스팅 영역을 사용해야 하며 클러스터 인프라를 프로비저닝하는 경우 설치 프로그램이 사용하는 것과 유사한 개인 호스팅 영역을 사용할 수 있습니다. 로드 밸런서로 확인되는 DNS 항목을 생성해야 합니다. **api.<cluster_name>.<domain>**에 대한 항목은 외부 로드 밸런서를 가리켜야 하고 **api-int.<cluster_name>.<domain>**에 대한 항목은 내부 로드 밸런서를 가리켜야 합니다.

클러스터에는 또한 Kubernetes API 및 해당 확장에 필요한 포트 6443 및 새 시스템의 Ignition 구성 파일에 필요한 포트 22623용 로드 밸런서와 리스너가 필요합니다. 대상은 컨트롤 플레인 노드가 됩니다. 포트 6443은 클러스터 외부의 클라이언트와 클러스터 내부의 노드에서 모두 액세스할 수 있어야 합니다. 포트 22623은 클러스터 내부 노드에서 액세스할 수 있어야 합니다.

구성 요소	AWS 유형	설명
DNS	**AWS::Route53::HostedZone**	내부 DNS의 호스팅 영역입니다.
etcd 레코드 세트	**AWS::Route53::RecordSet**	컨트롤 플레인 시스템의 etcd 등록 레코드입니다.
공용 로드 밸런서	**AWS::ElasticLoadBalancingV2::LoadBalancer**	퍼블릭 서브넷의 로드 밸런서입니다.
외부 API 서버 레코드	**AWS::Route53::RecordSetGroup**	외부 API 서버의 별칭 레코드입니다.
외부 리스너	**AWS::ElasticLoadBalancingV2::Listener**	외부 로드 밸런서용 포트 6443의 리스너입니다.
외부 대상 그룹	**AWS::ElasticLoadBalancingV2::TargetGroup**	외부 로드 밸런서의 대상 그룹입니다.
프라이빗 로드 밸런서	**AWS::ElasticLoadBalancingV2::LoadBalancer**	프라이빗 서브넷의 로드 밸런서입니다.
내부 API 서버 레코드	**AWS::Route53::RecordSetGroup**	내부 API 서버의 별칭 레코드입니다.
내부 리스너	**AWS::ElasticLoadBalancingV2::Listener**	내부 로드 밸런서용 포트 22623의 리스너입니다.

구성 요소	AWS 유형	설명
내부 대상 그룹	**AWS::ElasticLoadBalancingV2::TargetGroup**	내부 로드 밸런서의 대상 그룹입니다.
내부 리스너	**AWS::ElasticLoadBalancingV2::Listener**	내부 로드 밸런서용 포트 6443의 리스너입니다.
내부 대상 그룹	**AWS::ElasticLoadBalancingV2::TargetGroup**	내부 로드 밸런서의 대상 그룹입니다.

보안 그룹

컨트롤 플레인 및 작업자 시스템에는 다음 포트에 대한 액세스 권한이 필요합니다.

그룹	유형	IP 프로토콜	포트 범위
MasterSecurityGroup	**AWS::EC2::SecurityGroup**	**icmp**	**0**
		tcp	**22**
		tcp	**6443**
		tcp	**22623**
WorkerSecurityGroup	**AWS::EC2::SecurityGroup**	**icmp**	**0**
		tcp	**22**
BootstrapSecurityGroup	**AWS::EC2::SecurityGroup**	**tcp**	**22**
		tcp	**19531**

컨트롤 플레인 인그레스

컨트롤 플레인 시스템에는 다음과 같은 인그레스 그룹이 필요합니다. 각 인그레스 그룹은 **AWS::EC2::SecurityGroupIngress** 리소스입니다.

인그레스 그룹	설명	IP 프로토콜	포트 범위
MasterIngressEtcd	etcd	**tcp**	**2379- 2380**

인그레스 그룹	설명	IP 프로토콜	포트 범위
MasterIngressVxlan	Vxlan 패킷	**udp**	**4789**
MasterIngressWorkerVxlan	Vxlan 패킷	**udp**	**4789**
MasterIngressInternal	내부 클러스터 통신 및 Kubernetes 프록시 메트릭	**tcp**	**9000 - 9999**
MasterIngressWorkerInternal	내부 클러스터 통신	**tcp**	**9000 - 9999**
MasterIngressKube	Kubernetes kubelet, 스케줄러 및 컨트롤러 관리자	**tcp**	**10250 - 10259**
MasterIngressWorkerKube	Kubernetes kubelet, 스케줄러 및 컨트롤러 관리자	**tcp**	**10250 - 10259**
MasterIngressIngressServices	Kubernetes 인그레스 서비스	**tcp**	**30000 - 32767**
MasterIngressWorkerIngressServices	Kubernetes 인그레스 서비스	**tcp**	**30000 - 32767**
MasterIngressGeneve	Geneve 패킷	**udp**	**6081**
MasterIngressWorkerGeneve	Geneve 패킷	**udp**	**6081**
MasterIngressIpsecIke	IPsec IKE 패킷	**udp**	**500**
MasterIngressWorkerIpsecIke	IPsec IKE 패킷	**udp**	**500**
MasterIngressIpsecNat	IPsec NAT-T 패킷	**udp**	**4500**
MasterIngressWorkerIpsecNat	IPsec NAT-T 패킷	**udp**	**4500**

인그레스 그룹	설명	IP 프로토콜	포트 범위
MasterIngressIpsecEsp	IPsec ESP 패킷	**50**	모두
MasterIngressWorkerIpsecEsp	IPsec ESP 패킷	**50**	모두
MasterIngressInternalUDP	내부 클러스터 통신	**udp**	**9000 - 9999**
MasterIngressWorkerInternalUDP	내부 클러스터 통신	**udp**	**9000 - 9999**
MasterIngressIngressServicesUDP	Kubernetes 인그레스 서비스	**udp**	**30000 - 32767**
MasterIngressWorkerIngressServicesUDP	Kubernetes 인그레스 서비스	**udp**	**30000 - 32767**

작업자 인그레스

작업자 시스템에는 다음과 같은 인그레스 그룹이 필요합니다. 각 인그레스 그룹은 **AWS::EC2::SecurityGroupIngress** 리소스입니다.

인그레스 그룹	설명	IP 프로토콜	포트 범위
WorkerIngressVxlan	Vxlan 패킷	**udp**	**4789**
WorkerIngressWorkerVxlan	Vxlan 패킷	**udp**	**4789**
WorkerIngressInternal	내부 클러스터 통신	**tcp**	**9000 - 9999**
WorkerIngressWorkerInternal	내부 클러스터 통신	**tcp**	**9000 - 9999**
WorkerIngressKube	Kubernetes kubelet, 스케줄러 및 컨트롤러 관리자	**tcp**	**10250**
WorkerIngressWorkerKube	Kubernetes kubelet, 스케줄러 및 컨트롤러 관리자	**tcp**	**10250**

인그레스 그룹	설명	IP 프로토콜	포트 범위
WorkerIngressIngressServices	Kubernetes 인그레스 서비스	**tcp**	**30000 - 32767**
WorkerIngressWorkerIngressServices	Kubernetes 인그레스 서비스	**tcp**	**30000 - 32767**
WorkerIngressGeneve	Geneve 패킷	**udp**	**6081**
WorkerIngressMasterGeneve	Geneve 패킷	**udp**	**6081**
WorkerIngressIpsecIke	IPsec IKE 패킷	**udp**	**500**
WorkerIngressMasterIpsecIke	IPsec IKE 패킷	**udp**	**500**
WorkerIngressIpsecNat	IPsec NAT-T 패킷	**udp**	**4500**
WorkerIngressMasterIpsecNat	IPsec NAT-T 패킷	**udp**	**4500**
WorkerIngressIpsecEsp	IPsec ESP 패킷	**50**	모두
WorkerIngressMasterIpsecEsp	IPsec ESP 패킷	**50**	모두
WorkerIngressInternalUDP	내부 클러스터 통신	**udp**	**9000 - 9999**
WorkerIngressMasterInternalUDP	내부 클러스터 통신	**udp**	**9000 - 9999**
WorkerIngressIngressServicesUDP	Kubernetes 인그레스 서비스	**udp**	**30000 - 32767**

인그레스 그룹	설명	IP 프로토콜	포트 범위
WorkerIngress MasterIngress ServicesUDP	Kubernetes 인그레스 서비스	**udp**	**30000 - 32767**

역할 및 인스턴스 프로필

AWS에서 시스템 권한을 부여해야 합니다. 제공된 CloudFormation 템플릿은 다음 **AWS::IAM::Role** 오브젝트에 대한 **허용** 권한을 머신에 부여하며 각 역할 세트의 **AWS::IAM::InstanceProfile**을 제공합니다. 템플릿을 사용하지 않는 경우 다음과 같은 광범위한 권한 또는 다음과 같은 개별 권한을 시스템에 부여할 수 있습니다.

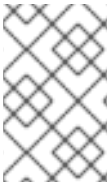
역할	효과	동작	리소스 이름
Master	**허용**	**ec2:***	*
	허용	**elasticloadbalancing:***	*
	허용	**iam:PassRole**	*
	허용	**s3:GetObject**	*
Worker	**허용**	**ec2:Describe***	*
부트스트랩	**허용**	**ec2:Describe***	*
	허용	**ec2:AttachVolume**	*
	허용	**ec2:DetachVolume**	*

4.13.5.2. 클러스터 시스템

다음 시스템의 **AWS:: EC2:: Instance** 개체가 필요합니다.

- 부트스트랩 시스템. 이 시스템은 설치 중에 필요하지만 클러스터가 배포된 후 제거할 수 있습니다.
- 컨트롤 플레인 시스템 세 개 컨트롤 플레인 시스템은 머신 세트에 의해 관리되지 않습니다.
- 컴퓨팅 시스템. 설치 중에 두 개 이상의 컴퓨팅 시스템(작업자 시스템이라고도 함)을 생성해야 합니다. 이러한 시스템은 머신 세트에 의해 관리되지 않습니다.

4.13.5.3. IAM 사용자에게 필요한 AWS 권한



참고

기본 클러스터 리소스를 삭제하려면 IAM 사용자에게 **us-east-1** 리전에 권한 **태그:GetResources** 가 있어야 합니다. AWS API 요구 사항의 일부로 OpenShift Container Platform 설치 프로그램은 이 리전에서 다양한 작업을 수행합니다.

AWS(Amazon Web Services)에서 생성되는 IAM 사용자에게 **AdministratorAccess** 정책을 연결하면 해당 사용자에게 필요한 모든 권한이 부여됩니다. OpenShift Container Platform 클러스터의 모든 구성 요소를 배포하려면 IAM 사용자에게 다음과 같은 권한이 필요합니다.

예 4.43. 설치에 필요한 EC2 권한

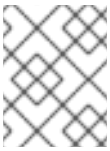
- **ec2:AuthorizeSecurityGroupEgress**
- **ec2:AuthorizeSecurityGroupIngress**
- **ec2:CopyImage**
- **ec2:CreateNetworkInterface**
- **ec2:AttachNetworkInterface**
- **ec2:CreateSecurityGroup**
- **ec2:CreateTags**
- **ec2:CreateVolume**
- **ec2:DeleteSecurityGroup**
- **ec2:DeleteSnapshot**
- **ec2:DeleteTags**
- **ec2:DeregisterImage**
- **ec2:DescribeAccountAttributes**
- **ec2:DescribeAddresses**
- **ec2:DescribeAvailabilityZones**
- **ec2:DescribeDhcpOptions**
- **ec2:DescribeImages**
- **ec2:DescribeInstanceAttribute**
- **ec2:DescribeInstanceCreditSpecifications**
- **ec2:DescribeInstances**
- **ec2:DescribeInstanceTypes**
- **ec2:DescribeInternetGateways**
- **ec2:DescribeKeyPairs**

- ec2:DescribeNatGateways
- ec2:DescribeNetworkAcls
- ec2:DescribeNetworkInterfaces
- ec2:DescribePrefixLists
- ec2:DescribeRegions
- ec2:DescribeRouteTables
- ec2:DescribeSecurityGroups
- ec2:DescribeSubnets
- ec2:DescribeTags
- ec2:DescribeVolumes
- ec2:DescribeVpcAttribute
- ec2:DescribeVpcClassicLink
- ec2:DescribeVpcClassicLinkDnsSupport
- ec2:DescribeVpcEndpoints
- ec2:DescribeVpcs
- ec2:GetEbsDefaultKmsKeyId
- ec2:ModifyInstanceAttribute
- ec2:ModifyNetworkInterfaceAttribute
- ec2:RevokeSecurityGroupEgress
- ec2:RevokeSecurityGroupIngress
- ec2:RunInstances
- ec2:TerminateInstances

예 4.44. 설치 과정에서 네트워크 리소스를 생성하는 데 필요한 권한

- ec2:AllocateAddress
- ec2:AssociateAddress
- ec2:AssociateDhcpOptions
- ec2:AssociateRouteTable
- ec2:AttachInternetGateway
- ec2:CreateDhcpOptions

- **ec2:CreateInternetGateway**
- **ec2:CreateNatGateway**
- **ec2:CreateRoute**
- **ec2:CreateRouteTable**
- **ec2:CreateSubnet**
- **ec2:CreateVpc**
- **ec2:CreateVpcEndpoint**
- **ec2:ModifySubnetAttribute**
- **ec2:ModifyVpcAttribute**



참고

기존 VPC를 사용하는 경우 네트워크 리소스를 생성하기 위해 계정에 이러한 권한이 필요하지 않습니다.

예 4.45. 설치에 필요한 Elastic Load Balancing 권한(ELB)

- **elasticloadbalancing:AddTags**
- **elasticloadbalancing:ApplySecurityGroupsToLoadBalancer**
- **elasticloadbalancing:AttachLoadBalancerToSubnets**
- **elasticloadbalancing:ConfigureHealthCheck**
- **elasticloadbalancing:CreateLoadBalancer**
- **elasticloadbalancing:CreateLoadBalancerListeners**
- **elasticloadbalancing:DeleteLoadBalancer**
- **elasticloadbalancing:DeregisterInstancesFromLoadBalancer**
- **elasticloadbalancing:DescribeInstanceHealth**
- **elasticloadbalancing:DescribeLoadBalancerAttributes**
- **elasticloadbalancing:DescribeLoadBalancers**
- **elasticloadbalancing:DescribeTags**
- **elasticloadbalancing:ModifyLoadBalancerAttributes**
- **elasticloadbalancing:RegisterInstancesWithLoadBalancer**
- **elasticloadbalancing:SetLoadBalancerPoliciesOfListener**

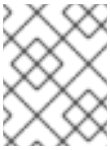
예 4.46. 설치에 필요한 Elastic Load Balancing 권한(ELBv2)

- **elasticloadbalancing:AddTags**
- **elasticloadbalancing:CreateListener**
- **elasticloadbalancing:CreateLoadBalancer**
- **elasticloadbalancing:CreateTargetGroup**
- **elasticloadbalancing:DeleteLoadBalancer**
- **elasticloadbalancing:DeregisterTargets**
- **elasticloadbalancing:DescribeListeners**
- **elasticloadbalancing:DescribeLoadBalancerAttributes**
- **elasticloadbalancing:DescribeLoadBalancers**
- **elasticloadbalancing:DescribeTargetGroupAttributes**
- **elasticloadbalancing:DescribeTargetHealth**
- **elasticloadbalancing:ModifyLoadBalancerAttributes**
- **elasticloadbalancing:ModifyTargetGroup**
- **elasticloadbalancing:ModifyTargetGroupAttributes**
- **elasticloadbalancing:RegisterTargets**

예 4.47. 설치에 필요한 IAM 권한

- **iam:AddRoleToInstanceProfile**
- **iam:CreateInstanceProfile**
- **iam:CreateRole**
- **iam:DeleteInstanceProfile**
- **iam:DeleteRole**
- **iam:DeleteRolePolicy**
- **iam:GetInstanceProfile**
- **iam:GetRole**
- **iam:GetRolePolicy**
- **iam:GetUser**
- **iam:ListInstanceProfilesForRole**

- **iam:ListRoles**
- **iam:ListUsers**
- **iam:PassRole**
- **iam:PutRolePolicy**
- **iam:RemoveRoleFromInstanceProfile**
- **iam:SimulatePrincipalPolicy**
- **iam:TagRole**



참고

AWS 계정에서 탄력적 로드 밸런서 (ELB)를 생성하지 않은 경우 IAM 사용자에게 **iam:CreateServiceLinkedRole** 권한이 필요합니다.

예 4.48. 설치에 필요한 Route 53 권한

- **route53:ChangeResourceRecordSets**
- **route53:ChangeTagsForResource**
- **route53:CreateHostedZone**
- **route53:DeleteHostedZone**
- **route53:GetChange**
- **route53:GetHostedZone**
- **route53:ListHostedZones**
- **route53:ListHostedZonesByName**
- **route53:ListResourceRecordSets**
- **route53:ListTagsForResource**
- **route53:UpdateHostedZoneComment**

예 4.49. 설치에 필요한 S3 권한

- **s3:CreateBucket**
- **s3:DeleteBucket**
- **s3:GetAccelerateConfiguration**
- **s3:GetBucketAcl**
- **s3:GetBucketCors**

- **s3:GetBucketLocation**
- **s3:GetBucketLogging**
- **s3:GetBucketObjectLockConfiguration**
- **s3:GetBucketReplication**
- **s3:GetBucketRequestPayment**
- **s3:GetBucketTagging**
- **s3:GetBucketVersioning**
- **s3:GetBucketWebsite**
- **s3:GetEncryptionConfiguration**
- **s3:GetLifecycleConfiguration**
- **s3:GetReplicationConfiguration**
- **s3:ListBucket**
- **s3:PutBucketAcl**
- **s3:PutBucketTagging**
- **s3:PutEncryptionConfiguration**

예 4.50. 클러스터 Operator에 필요한 S3 권한

- **s3:DeleteObject**
- **s3:GetObject**
- **s3:GetObjectAcl**
- **s3:GetObjectTagging**
- **s3:GetObjectVersion**
- **s3:PutObject**
- **s3:PutObjectAcl**
- **s3:PutObjectTagging**

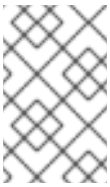
예 4.51. 기본 클러스터 리소스를 삭제하는 데 필요한 권한

- **autoscaling:DescribeAutoScalingGroups**
- **ec2:DeleteNetworkInterface**
- **ec2:DeleteVolume**

- **elasticloadbalancing:DeleteTargetGroup**
- **elasticloadbalancing:DescribeTargetGroups**
- **iam:DeleteAccessKey**
- **iam:DeleteUser**
- **iam:ListAttachedRolePolicies**
- **iam:ListInstanceProfiles**
- **iam:ListRolePolicies**
- **iam:ListUserPolicies**
- **s3:DeleteObject**
- **s3:ListBucketVersions**
- **tag:GetResources**

예 4.52. 네트워크 리소스를 삭제하는 데 필요한 권한

- **ec2:DeleteDhcpOptions**
- **ec2:DeleteInternetGateway**
- **ec2:DeleteNatGateway**
- **ec2:DeleteRoute**
- **ec2:DeleteRouteTable**
- **ec2:DeleteSubnet**
- **ec2:DeleteVpc**
- **ec2:DeleteVpcEndpoints**
- **ec2:DetachInternetGateway**
- **ec2:DisassociateRouteTable**
- **ec2:ReleaseAddress**
- **ec2:ReplaceRouteTableAssociation**



참고

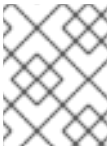
기존 VPC를 사용하는 경우 네트워크 리소스를 삭제하기 위해 계정에 이러한 권한이 필요하지 않습니다. 대신 사용자 계정에는 네트워크 리소스를 삭제하기 위한 **tag:UntagResources** 권한만 필요합니다.

예 4.53. 공유 인스턴스 역할이 있는 클러스터를 삭제하는 데 필요한 권한

- **iam:UntagRole**

예 4.54. 매니페스트를 생성하는 데 필요한 추가 IAM 및 S3 권한

- **iam:DeleteAccessKey**
- **iam:DeleteUser**
- **iam:DeleteUserPolicy**
- **iam:GetUserPolicy**
- **iam:ListAccessKeys**
- **iam:PutUserPolicy**
- **iam:TagUser**
- **s3:PutBucketPublicAccessBlock**
- **s3:GetBucketPublicAccessBlock**
- **s3:PutLifecycleConfiguration**
- **s3:HeadBucket**
- **s3:ListBucketMultipartUploads**
- **s3:AbortMultipartUpload**



참고

mint 모드를 사용하여 클라우드 공급자 인증 정보를 관리하는 경우 IAM 사용자에게 **iam:CreateAccessKey** 및 **iam:CreateUser** 권한이 필요합니다.

예 4.55. 인스턴스에 대한 선택적 권한 및 설치에 대한 할당량 검사

- **ec2:DescribeInstanceTypeOfferings**
- **servicequotas:ListAWSDefaultServiceQuotas**

4.13.6. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 **core** 사용자의 **~/.ssh/authorized_keys** 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 **core**로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. **./openshift-install gather** 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

참고

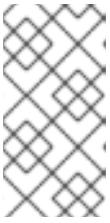
AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: **~/.ssh/id_ed25519**)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 **'~/.ssh** 디렉터리에 있는지 확인하십시오.



참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 **ed25519** 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 **rsa** 또는 **ecdsa** 알고리즘을 사용하는 키를 생성합니다.

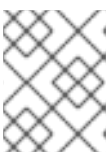
2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 **~/.ssh/id_ed25519.pub** 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 **./openshift-install gather** 명령을 사용하려는 경우 필요합니다.



참고

일부 배포에서는 **~/.ssh/id_rsa** 및 **~/.ssh/id_dsa**와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. **ssh-agent** 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. **ssh-agent**에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: **~/.ssh/id_ed25519**).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다. 프로비저닝하는 인프라에 클러스터를 설치하는 경우 설치 프로그램에 키를 제공해야 합니다.

4.13.7. AWS용 설치 파일 생성

사용자 프로비저닝 인프라를 사용하는 OpenShift Container Platform을 AWS(Amazon Web Services)에 설치하려면 설치 프로그램에서 클러스터를 배포하는 데 필요한 파일을 생성하고 클러스터가 사용할 시스템만을 생성하도록 파일을 수정해야 합니다. **install-config.yaml** 파일, Kubernetes 매니페스트 및 Ignition 구성 파일을 생성하고 사용자 지정합니다. 또한 설치 준비 단계에서 별도의 **var** 파티션을 먼저 설정할 수 있는 옵션이 있습니다.

4.13.7.1. 선택 사항: 별도의 /var 파티션 만들기

OpenShift Container Platform의 디스크 파티션 설정은 설치 프로그램에 맡기는 것이 좋습니다. 그러나 확장하려는 파일 시스템의 일부에 별도의 파티션을 생성해야 하는 경우가 있습니다.

OpenShift 컨테이너 플랫폼은 **/var** 파티션 또는 **/var**의 하위 디렉터리 중 하나에 스토리지를 연결하는 단일 파티션의 추가를 지원합니다. 예를 들면 다음과 같습니다.

- **/var/lib/containers**: 시스템에 더 많은 이미지와 컨테이너가 추가됨에 따라 확장될 수 있는 컨테이너 관련 콘텐츠를 보관합니다.
- **/var/lib/etcd**: etcd 스토리지의 성능 최적화와 같은 목적으로 별도로 보관할 데이터를 보관합니다.
- **/var**: 감사 등의 목적에 맞게 별도로 분리하여 보관해야 하는 데이터를 보관합니다.

/var 디렉터리의 콘텐츠를 별도로 저장하면 필요에 따라 해당 영역에 대한 스토리지 확장을 보다 용이하게 하고 나중에 OpenShift Container Platform을 다시 설치하여 해당 데이터를 그대로 보존할 수 있습니다. 이 방법을 사용하면 모든 컨테이너를 다시 가져올 필요가 없으며 시스템을 업데이트할 때 대용량 로그 파일을 복사할 필요도 없습니다.

RHCOS(Red Hat Enterprise Linux CoreOS)를 새로 설치하기 전에 **/var가 있어야 하므로 다음 절차에서는 OpenShift Container Platform 설치의 openshift-install 준비 단계 중에 삽입되는 머신 구성 매니페스트를 생성하여 별도의 /var** 파티션을 설정합니다.

중요

이 절차에서 별도의 **/var** 파티션을 생성하기 위해 단계에 따라 이 섹션의 뒷부분에서 설명한 대로 Kubernetes 매니페스트 및 Ignition 구성 파일을 다시 생성할 필요가 없습니다.

프로세스

1. OpenShift Container Platform 설치 파일을 저장할 디렉터리를 만듭니다.

```
$ mkdir $HOME/clusterconfig
```

2. **openshift-install**을 실행하여 **manifest** 및 **openshift** 하위 디렉터리에 파일 세트를 만듭니다. 프롬프트가 표시되면 시스템 질문에 대답합니다.

```
$ openshift-install create manifests --dir $HOME/clusterconfig
```

출력 예

```
? SSH Public Key ...
INFO Credentials loaded from the "myprofile" profile in file "/home/myuser/.aws/credentials"
INFO Consuming Install Config from target directory
INFO Manifests created in: $HOME/clusterconfig/manifests and
$HOME/clusterconfig/openshift
```

3. 선택 사항: 설치 프로그램이 **clusterconfig/openshift** 디렉터리에 매니페스트를 생성했는지 확인합니다.

```
$ ls $HOME/clusterconfig/openshift/
```

출력 예

```
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

4. 추가 파티션을 구성하는 Butane 구성을 생성합니다. 예를 들어 파일 이름을 **$HOME/clusterconfig/98-var-partition.bu** 로 지정하고, 디스크 장치 이름을 **worker** 시스템의 스토리지 장치 이름으로 변경하고 스토리지 크기를 적절하게 설정합니다. 이 예에서는 **/var** 디렉토리를 별도의 파티션에 배치합니다.

```
variant: openshift
version: 4.9.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
```

```
disks:
- device: /dev/<device_name> 1
  partitions:
  - label: var
    start_mib: <partition_start_offset> 2
    size_mib: <partition_size> 3
filesystems:
  - device: /dev/disk/by-partlabel/var
    path: /var
    format: xfs
    mount_options: [defaults, prjquota] 4
    with_mount_unit: true
```

1 파티션을 설정해야하는 디스크 저장 장치 이름입니다.

2 데이터 파티션을 부트 디스크에 추가할 때 최소 25000MiB(메비 바이트)가 권장됩니다. 루트 파일 시스템은 지정된 오프셋까지 사용 가능한 모든 공간을 채우기 위해 자동으로 크기가 조정됩니다. 값이 지정되지 않거나 지정된 값이 권장 최소값보다 작으면 생성되는 루트 파일 시스템의 크기가 너무 작아지고 RHCOS를 나중에 다시 설치할 때 데이터 파티션의 첫 번째 부분을 덮어 쓸 수 있습니다.

3 데이터 파티션의 크기(MB)입니다.

4 컨테이너 스토리지에 사용되는 파일 시스템에 **prjquota** 마운트 옵션을 활성화해야 합니다.

참고

별도의 **/var** 파티션을 만들 때 다른 인스턴스 유형에 동일한 장치 이름이 없는 경우 작업자 노드에 다른 인스턴스 유형을 사용할 수 없습니다.

5. Butane 구성에서 매니페스트를 생성하여 **clusterconfig/openshift** 디렉터리에 저장합니다. 예를 들어 다음 명령을 실행합니다.

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

6. **openshift-install**을 다시 실행하여 **manifest** 및 **openshift** 하위 디렉터리의 파일 세트에서 Ignition 구성을 만듭니다.

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

이제 Ignition 구성 파일을 설치 절차에 대한 입력으로 사용하여 RHCOS (Red Hat Enterprise Linux CoreOS) 시스템을 설치할 수 있습니다.

4.13.7.2. 설치 구성 파일 만들기

설치 프로그램이 클러스터를 배포하는 데 필요한 설치 구성 파일을 생성하고 사용자 지정합니다.

사전 요구 사항

- 사용자가 프로비저닝한 인프라의 OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받으셨습니다. 제한된 네트워크 설치의 경우, 해당 파일은 미러 호스트에 있습니다.
- Red Hat에서 게시한 Red Hat Enterprise Linux CoreOS (RHCOS) AMI와 함께 리전에 클러스터를 배포하고 있는지 확인하셨습니다. AWS GovCloud 리전과 같이 사용자 지정 AMI가 필요한 리전에 배포하는 경우 **install-config.yaml** 파일을 수동으로 생성해야 합니다.

프로세스

1. **install-config.yaml** 파일을 생성합니다.
   a. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

   1 **<installation_directory>**는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.

중요

비어 있는 디렉터리를 지정합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

   b. 화면에 나타나는 지시에 따라 클라우드에 대한 구성 세부 사항을 입력합니다.
      i. 선택사항: 클러스터 시스템에 액세스하는 데 사용할 SSH 키를 선택합니다.

참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

      ii. 대상 플랫폼으로 **aws**를 선택합니다.
      iii. 컴퓨터에 AWS 프로필이 저장되어 있지 않은 경우 설치 프로그램을 실행하도록 구성한 사용자의 AWS 액세스 키 ID와 시크릿 액세스 키를 입력합니다.

참고

AWS 액세스 키 ID와 시크릿 액세스 키는 설치 호스트에 있는 현재 사용자의 홈 디렉터리에서 **~/.aws/credentials**에 저장됩니다. 내보낸 프로필의 인증 정보가 파일에 없으면 설치 프로그램에서 인증 정보에 대한 메시지를 표시합니다. 설치 프로그램에 사용자가 제공하는 인증 정보는 파일에 저장됩니다.

      iv. 클러스터를 배포할 AWS 리전을 선택합니다.
      v. 클러스터에 대해 구성한 Route53 서비스의 기본 도메인을 선택합니다.

vi. 클러스터를 설명할 수 있는 이름을 입력합니다.

vii. Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 붙여넣습니다.

2. 제한된 네트워크에서의 설치에 필요한 추가 정보를 제공하려면 **install-config.yaml** 파일을 편집합니다.

a. 레지스트리의 인증 정보를 포함하도록 **pullSecret** 값을 업데이트합니다.

```
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email":
"you@example.com"}}}'
```

<local_registry>는 미러 레지스트리가 해당 내용을 제공하는 데 사용하는 레지스트리 도메인 이름과 포트(선택사항)를 지정합니다. 예: **registry.example.com** 또는 **registry.example.com:5000**. **<credentials>**는 미러 레지스트리의 base64 인코딩 사용자 이름과 암호를 지정합니다.

b. **additionalTrustBundle** 매개변수와 값을 추가합니다. 값은 미러 레지스트리에 사용한 인증서 파일의 내용이어야 하며, 신뢰할 수 있는 기존 인증 기관 또는 미러 레지스트리에 대해 생성한 자체 서명 인증서일 수 있습니다.

```
additionalTrustBundle: |
  -----BEGIN CERTIFICATE-----

  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
```

c. 이미지 내용 리소스를 추가합니다.

```
imageContentSources:
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

명령 출력에서 **imageContentSources** 섹션을 사용하여 제한된 네트워크로 가져온 매체에서 내용을 미러링할 때 사용한 값 또는 리포지토리를 미러링합니다.

d. 선택사항: 게시 전략을 **Internal**로 설정합니다.

```
publish: Internal
```

이 옵션을 설정하여 내부 Ingress Controller 및 프라이빗 로드 밸런서를 생성합니다.

3. 선택사항: **install-config.yaml** 파일을 백업합니다.

중요

install-config.yaml 파일은 설치 과정에서 사용됩니다. 이 파일을 재사용하려면 지금 백업해야 합니다.

추가 리소스

- AWS 프로필 및 인증 정보 구성에 대한 자세한 내용은 AWS 문서의 구성 및 인증 정보 파일 설정을 참조하십시오.

4.13.7.3. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 HTTP 또는 HTTPS 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 OpenShift Container Platform 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 API에 대한 호출을 포함하여 모든 클러스터 발신(Egress) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.

참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 Red Hat OpenStack Platform (RHOSP)에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

- **ec2.<region>.amazonaws.com,elasticloadbalancing.<region>.amazonaws.com** 및 **s3.<region>.amazonaws.com** 끝점을 VPC 끝점에 추가했습니다. 이러한 끝점은 노드에서 AWS EC2 API로 요청을 완료하는 데 필요합니다. 프록시는 노드 수준이 아닌 컨테이너 수준에서 작동하므로 이러한 요청을 AWS 사설 네트워크를 통해 AWS EC2 API로 라우팅해야 합니다. 프록시 서버의 허용 목록에 EC2 API의 공용 IP 주소를 추가하는 것만으로는 충분하지 않습니다.

절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. 클러스터 외부에서 HTTP 연결을 구축하는 데 사용할 프록시 URL입니다. URL 스키마는 **http**여야 합니다.

2 클러스터 외부에서 HTTPS 연결을 구축하는 데 사용할 프록시 URL입니다.

3 대상 도메인 이름, IP 주소 또는 프록시에서 제외할 기타 네트워크 CIDR로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. ***를 사용하여 모든 대상에 대해 프록시를 바이패스합니다.

4 이 값을 제공하면 설치 프로그램에서 추가 CA 인증서를 보유할 **openshift -config 네임스페이스에 user-ca- bundle** 이라는 구성 맵을 생성합니다. **additionalTrustBundle** 및 하나 이상의 프록시 설정을 제공하는 경우 **프록시** 오브젝트는 **trustedCA** 필드의 **user-ca-bundle** 구성 맵을 참조하도록 구성됩니다. 그러면 Cluster Network Operator에서 **trustedCA** 매개변수에 대해 지정된 콘텐츠를 RHCOS 신뢰 번들과 병합하는 **trusted-ca-bundle** 구성 맵을 생성합니다. 프록시의 ID 인증서를 RHCOS 트러스트 번들에 있는 기관에서 서명하지 않은 경우 **additionalTrustBundle** 필드가 있어야 합니다.

참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 OpenShift Container Platform을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 nil이 됩니다.

참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

4.13.7.4. Kubernetes 매니페스트 및 Ignition 설정 파일 생성

일부 클러스터 정의 파일을 수정하고 클러스터 시스템을 수동으로 시작해야 하므로 클러스터가 시스템을 구성하는 데 필요한 Kubernetes 매니페스트 및 Ignition 구성 파일을 사용자가 생성해야 합니다.

설치 구성 파일은 Kubernetes 매니페스트로 변환됩니다. 매니페스트는 나중에 클러스터 머신을 구성하는 데 사용되는 Ignition 구성 파일로 래핑됩니다.

중요

- OpenShift Container Platform 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램을 가져오셨습니다. 제한된 네트워크 설치의 경우, 해당 파일은 미러 호스트에 있습니다.
- **install-config.yaml** 설치 구성 파일을 생성하셨습니다.

프로세스

1. OpenShift Container Platform 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터에 대한 Kubernetes 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>**는 사용자가 만든 **install-config.yaml** 파일이 포함된 설치 디렉터리를 지정합니다.

2. 컨트롤 플레인 시스템을 정의하는 Kubernetes 매니페스트 파일을 제거합니다.

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

이러한 파일을 제거하면 클러스터가 컨트롤 플레인 시스템을 자동으로 생성하지 못합니다.

3. 작업자 시스템을 정의하는 Kubernetes 매니페스트 파일을 제거합니다.

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

작업자 시스템은 사용자가 직접 생성하고 관리하기 때문에 초기화할 필요가 없습니다.

4. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes 매니페스트 파일의 **mastersSchedulable** 매개변수가 **false**로 설정되어 있는지 확인합니다. 이 설정으로 인해 컨트롤 플레인 머신에서 포드가 예약되지 않습니다.
   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** 파일을 엽니다.
   b. **mastersSchedulable** 매개변수를 찾아서 값을 **False**로 설정되어 있는지 확인합니다.
   c. 파일을 저장하고 종료합니다.

5. 선택사항: Ingress Operator가 사용자 대신 DNS 레코드를 생성하지 못하도록 하려면 **<installation_directory>/manifests/cluster-dns-02-config.yml** DNS 구성 파일에서 **privateZone** 및 **publicZone** 섹션을 제거합니다.

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

1 2 이 섹션을 완전히 제거합니다.

제거한 경우 나중에 인그레스 DNS 레코드를 수동으로 추가해야 합니다.

6. Ignition 구성 파일을 생성하려면 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행합니다.

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>**는 동일한 설치 디렉터리를 지정합니다.

설치 디렉터리의 부트스트랩, 컨트롤 플레인 및 컴퓨팅 노드에 대한 Ignition 구성 파일이 생성됩니다. **kubeadmin-password** 및 **kubeconfig** 파일은 **./<installation_directory>/auth** 디렉터리에 생성됩니다.

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

4.13.8. 인프라 이름 추출

Ignition 구성 파일에는 AWS(Amazon Web Services)에서 클러스터를 고유하게 식별하는 데 사용할 수 있는 고유한 클러스터 ID가 포함되어 있습니다. 인프라 이름은 OpenShift Container Platform 설치 중에 적절한 AWS 리소스를 찾는 데도 사용됩니다. 제공된 CloudFormation 템플릿에 이 인프라 이름에 대한 참조가 포함되어 있으므로 이름을 추출해야 합니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받으셨습니다.
- 클러스터에 대한 Ignition 구성 파일을 생성하셨습니다.
- **jq** CLI를 설치하셨습니다.

프로세스

- Ignition 구성 파일 메타데이터에서 인프라 이름을 추출하여 확인하려면 다음 명령을 실행하십시오.

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

출력 예

```
openshift-vw9j6 1
```

1 이 명령의 출력은 클러스터 이름과 임의의 문자열입니다.

4.13.9. AWS에서 VPC 생성

OpenShift Container Platform 클러스터에서 사용할 VPC(Virtual Private Cloud)를 AWS(Amazon Web Services)에서 생성해야 합니다. VPN 및 라우팅 테이블 등 요구사항에 맞게 VPC를 사용자 지정할 수 있습니다.

제공된 CloudFormation 템플릿과 사용자 정의 매개변수 파일을 사용하여 VPC를 나타내는 AWS 리소스 스택을 생성할 수 있습니다.

참고

AWS 인프라를 생성하는 데 제공된 CloudFormation 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- AWS 계정을 구성했습니다.
- **aws 구성**을 실행하여 AWS 키와 리전을 로컬 AWS 프로필에 추가하셨습니다.
- 클러스터에 대한 Ignition 구성 파일을 생성하셨습니다.

프로세스

1. 템플릿에 필요한 매개변수 값이 포함된 JSON 파일을 생성합니다.

```
[
  {
    "ParameterKey": "VpcCidr", 1
    "ParameterValue": "10.0.0.0/16" 2
  },
  {
    "ParameterKey": "AvailabilityZoneCount", 3
    "ParameterValue": "1" 4
  },
  {
    "ParameterKey": "SubnetBits", 5
    "ParameterValue": "12" 6
  }
]
```

1 VPC의 CIDR 블록입니다.

2 **xxxx/16-24** 형식으로 CIDR 블록을 지정합니다.

3 VPC를 배포할 가용성 영역의 수입니다.

4 **1**과 **3** 사이의 정수를 지정합니다.

5 각 가용성 영역에 있는 각 서브넷의 크기입니다.

6 **5**와 **13** 사이의 정수를 지정합니다. 여기서 **5**는 **/27**이고 **13**은 **/19**입니다.

2. 이 항목의 **VPC 섹션에 대한 CloudFormation 템플릿**섹션에서 템플릿을 복사하여 컴퓨터에 YAML 파일로 저장합니다. 이 템플릿은 클러스터에 필요한 VPC를 설명합니다.

3. CloudFormation 템플릿을 시작하여 VPC를 나타내는 AWS 리소스 스택을 생성합니다.

중요

명령은 한 줄로 입력해야 합니다.

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
```

1 **<name>**은 CloudFormation 스택의 이름입니다(예: **cluster-vpc**). 클러스터를 제거하는 경우 이 스택의 이름이 필요합니다.

2 **<template>**은 저장한 CloudFormation 템플릿 YAML 파일의 상대 경로 및 이름입니다.

3 **<parameters>**는 CloudFormation 매개변수 JSON 파일의 상대 경로 및 이름입니다.

출력 예

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-vpc/dbedae40-2fd3-11eb-820e-12a48460849f
```

4. 템플릿 구성 요소가 있는지 확인합니다.

```
$ aws cloudformation describe-stacks --stack-name <name>
```

StackStatus에 **CREATE_COMPLETE**이 표시된 후 다음 매개변수의 출력 값이 표시됩니다. 클러스터를 생성하기 위해 실행하는 다른 CloudFormation 템플릿에 이러한 매개변수 값을 제공해야 합니다.

VpcId	VPC의 ID입니다.
PublicSubnetIds	새 퍼블릭 서브넷의 ID입니다.
PrivateSubnetIds	새 프라이빗 서브넷의 ID입니다.

4.13.9.1. VPC용 CloudFormation 템플릿

다음 CloudFormation 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 VPC를 배포할 수 있습니다.

예 4.56. VPC용 CloudFormation 템플릿

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for Best Practice VPC with 1-3 AZs

Parameters:
  VpcCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])(\/(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.0.0/16
    Description: CIDR block for VPC.
    Type: String
  AvailabilityZoneCount:
    ConstraintDescription: "The number of availability zones. (Min: 1, Max: 3)"
    MinValue: 1
    MaxValue: 3
    Default: 1
    Description: "How many AZs to create VPC subnets for. (Min: 1, Max: 3)"
    Type: Number
  SubnetBits:
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/19-27.
    MinValue: 5
    MaxValue: 13
    Default: 12
    Description: "Size of each subnet to create within the availability zones. (Min: 5 = /27, Max: 13 = /19)"
    Type: Number

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcCidr
      - SubnetBits
    - Label:
        default: "Availability Zones"
      Parameters:
      - AvailabilityZoneCount
    ParameterLabels:
      AvailabilityZoneCount:
        default: "Availability Zone Count"
      VpcCidr:
        default: "VPC CIDR"
      SubnetBits:
        default: "Bits Per Subnet"

Conditions:
  DoAz3: !Equals [3, !Ref AvailabilityZoneCount]
  DoAz2: !Or [!Equals [2, !Ref AvailabilityZoneCount], Condition: DoAz3]

Resources:
  VPC:
    Type: "AWS::EC2::VPC"
    Properties:
      EnableDnsSupport: "true"
```

```
      EnableDnsHostnames: "true"
      CidrBlock: !Ref VpcCidr
  PublicSubnet:
    Type: "AWS::EC2::Subnet"
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [0, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 0
      - Fn::GetAZs: !Ref "AWS::Region"
  PublicSubnet2:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz2
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [1, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 1
      - Fn::GetAZs: !Ref "AWS::Region"
  PublicSubnet3:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz3
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [2, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 2
      - Fn::GetAZs: !Ref "AWS::Region"
  InternetGateway:
    Type: "AWS::EC2::InternetGateway"
  GatewayToInternet:
    Type: "AWS::EC2::VPCGatewayAttachment"
    Properties:
      VpcId: !Ref VPC
      InternetGatewayId: !Ref InternetGateway
  PublicRouteTable:
    Type: "AWS::EC2::RouteTable"
    Properties:
      VpcId: !Ref VPC
  PublicRoute:
    Type: "AWS::EC2::Route"
    DependsOn: GatewayToInternet
    Properties:
      RouteTableId: !Ref PublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref InternetGateway
  PublicSubnetRouteTableAssociation:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PublicSubnet
      RouteTableId: !Ref PublicRouteTable
  PublicSubnetRouteTableAssociation2:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Condition: DoAz2
    Properties:
      SubnetId: !Ref PublicSubnet2
```

```
      RouteTableId: !Ref PublicRouteTable
  PublicSubnetRouteTableAssociation3:
    Condition: DoAz3
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PublicSubnet3
      RouteTableId: !Ref PublicRouteTable
  PrivateSubnet:
    Type: "AWS::EC2::Subnet"
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [3, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 0
      - Fn::GetAZs: !Ref "AWS::Region"
  PrivateRouteTable:
    Type: "AWS::EC2::RouteTable"
    Properties:
      VpcId: !Ref VPC
  PrivateSubnetRouteTableAssociation:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PrivateSubnet
      RouteTableId: !Ref PrivateRouteTable
  NAT:
    DependsOn:
    - GatewayToInternet
    Type: "AWS::EC2::NatGateway"
    Properties:
      AllocationId:
        "Fn::GetAtt":
        - EIP
        - AllocationId
      SubnetId: !Ref PublicSubnet
  EIP:
    Type: "AWS::EC2::EIP"
    Properties:
      Domain: vpc
  Route:
    Type: "AWS::EC2::Route"
    Properties:
      RouteTableId:
        Ref: PrivateRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId:
        Ref: NAT
  PrivateSubnet2:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz2
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [4, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 1
      - Fn::GetAZs: !Ref "AWS::Region"
  PrivateRouteTable2:
```

```
  Type: "AWS::EC2::RouteTable"
  Condition: DoAz2
  Properties:
    VpcId: !Ref VPC
PrivateSubnetRouteTableAssociation2:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Condition: DoAz2
  Properties:
    SubnetId: !Ref PrivateSubnet2
    RouteTableId: !Ref PrivateRouteTable2
NAT2:
  DependsOn:
  - GatewayToInternet
  Type: "AWS::EC2::NatGateway"
  Condition: DoAz2
  Properties:
    AllocationId:
      "Fn::GetAtt":
      - EIP2
      - AllocationId
    SubnetId: !Ref PublicSubnet2
EIP2:
  Type: "AWS::EC2::EIP"
  Condition: DoAz2
  Properties:
    Domain: vpc
Route2:
  Type: "AWS::EC2::Route"
  Condition: DoAz2
  Properties:
    RouteTableId:
      Ref: PrivateRouteTable2
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId:
      Ref: NAT2
PrivateSubnet3:
  Type: "AWS::EC2::Subnet"
  Condition: DoAz3
  Properties:
    VpcId: !Ref VPC
    CidrBlock: !Select [5, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
    AvailabilityZone: !Select
    - 2
    - Fn::GetAZs: !Ref "AWS::Region"
PrivateRouteTable3:
  Type: "AWS::EC2::RouteTable"
  Condition: DoAz3
  Properties:
    VpcId: !Ref VPC
PrivateSubnetRouteTableAssociation3:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Condition: DoAz3
  Properties:
    SubnetId: !Ref PrivateSubnet3
    RouteTableId: !Ref PrivateRouteTable3
NAT3:
```

```
    DependsOn:
    - GatewayToInternet
    Type: "AWS::EC2::NatGateway"
    Condition: DoAz3
    Properties:
      AllocationId:
        "Fn::GetAtt":
        - EIP3
        - AllocationId
      SubnetId: !Ref PublicSubnet3
  EIP3:
    Type: "AWS::EC2::EIP"
    Condition: DoAz3
    Properties:
      Domain: vpc
  Route3:
    Type: "AWS::EC2::Route"
    Condition: DoAz3
    Properties:
      RouteTableId:
        Ref: PrivateRouteTable3
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId:
        Ref: NAT3
  S3Endpoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      PolicyDocument:
        Version: 2012-10-17
        Statement:
        - Effect: Allow
          Principal: '*'
          Action:
          - '*'
          Resource:
          - '*'
      RouteTableIds:
      - !Ref PublicRouteTable
      - !Ref PrivateRouteTable
      - !If [DoAz2, !Ref PrivateRouteTable2, !Ref "AWS::NoValue"]
      - !If [DoAz3, !Ref PrivateRouteTable3, !Ref "AWS::NoValue"]
      ServiceName: !Join
      - ''
      - - com.amazonaws.
        - !Ref 'AWS::Region'
        - .s3
      VpcId: !Ref VPC

Outputs:
  VpcId:
    Description: ID of the new VPC.
    Value: !Ref VPC
  PublicSubnetIds:
    Description: Subnet IDs of the public subnets.
    Value:
      !Join [
```

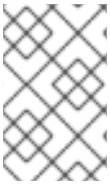
```
        ",",
        [!Ref PublicSubnet, !If [DoAz2, !Ref PublicSubnet2, !Ref "AWS::NoValue"], !If [DoAz3, !Ref
PublicSubnet3, !Ref "AWS::NoValue"]]
      ]
  PrivateSubnetIds:
    Description: Subnet IDs of the private subnets.
    Value:
      !Join [
        ",",
        [!Ref PrivateSubnet, !If [DoAz2, !Ref PrivateSubnet2, !Ref "AWS::NoValue"], !If [DoAz3, !Ref
PrivateSubnet3, !Ref "AWS::NoValue"]]
      ]
```

4.13.10. AWS에서 네트워킹 및 로드 밸런싱 구성 요소 생성

AWS(Amazon Web Services)에서 OpenShift Container Platform 클러스터가 사용할 수 있는 네트워킹 및 클래식 또는 네트워크 로드 밸런싱을 구성해야 합니다.

제공된 CloudFormation 템플릿과 사용자 정의 매개변수 파일을 사용하여 AWS 리소스 스택을 생성할 수 있습니다. 스택은 OpenShift Container Platform 클러스터에 필요한 네트워킹 및 로드 밸런싱 구성 요소를 나타냅니다. 템플릿은 호스팅 영역 및 서브넷 태그도 생성합니다.

단일 VPC(Virtual Private Cloud)에서 템플릿을 여러 번 실행할 수 있습니다.



참고

AWS 인프라를 생성하는 데 제공된 CloudFormation 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- AWS 계정을 구성했습니다.
- **aws 구성**을 실행하여 AWS 키와 리전을 로컬 AWS 프로필에 추가하셨습니다.
- 클러스터에 대한 Ignition 구성 파일을 생성하셨습니다.
- AWS에서 VPC 및 관련 서브넷을 생성하고 구성하셨습니다.

프로세스

1. 클러스터의 **install-config.yaml** 파일에서 지정한 Route 53 기본 도메인의 호스팅 영역 ID를 가져옵니다. 다음 명령을 실행하여 호스팅 영역에 대한 세부 정보를 얻을 수 있습니다.

```
$ aws route53 list-hosted-zones-by-name --dns-name <route53_domain> 1
```

1 **<route53_domain>**은 클러스터의 **install-config.yaml** 파일을 생성할 때 사용한 Route 53 기본 도메인을 지정합니다.

출력 예

```
mycluster.example.com. False 100
HOSTEDZONES 65F8F38E-2268-B835-E15C-AB55336FCBFA
/hostedzone/Z21IXYZABCZ2A4 mycluster.example.com. 10
```

예제 출력에서 호스팅 영역 ID는 **Z21IXYZABCZ2A4**입니다.

2. 템플릿에 필요한 매개변수 값이 포함된 JSON 파일을 생성합니다.

```
[
  {
    "ParameterKey": "ClusterName", 1
    "ParameterValue": "mycluster" 2
  },
  {
    "ParameterKey": "InfrastructureName", 3
    "ParameterValue": "mycluster-<random_string>" 4
  },
  {
    "ParameterKey": "HostedZoneId", 5
    "ParameterValue": "<random_string>" 6
  },
  {
    "ParameterKey": "HostedZoneName", 7
    "ParameterValue": "example.com" 8
  },
  {
    "ParameterKey": "PublicSubnets", 9
    "ParameterValue": "subnet-<random_string>" 10
  },
  {
    "ParameterKey": "PrivateSubnets", 11
    "ParameterValue": "subnet-<random_string>" 12
  },
  {
    "ParameterKey": "VpcId", 13
    "ParameterValue": "vpc-<random_string>" 14
  }
]
```

1. 호스트 이름에 사용할 짧은 대표 클러스터 이름입니다.
2. 클러스터의 **install-config.yaml** 파일을 생성할 때 사용한 클러스터 이름을 지정합니다.
3. 클러스터에 대해 Ignition 구성 파일에 인코딩되는 클러스터 인프라의 이름입니다.
4. Ignition 구성 파일 메타데이터에서 추출한 인프라 이름을 **<cluster-name>-<random-string>** 형식으로 지정합니다.
5. 대상을 등록할 Route 53 공용 영역 ID입니다.
6. Route 53 공용 영역 ID를 **Z21IXYZABCZ2A4**와 유사한 형식으로 지정합니다. 이 값은 AWS 콘솔에서 가져올 수 있습니다.
7. 대상을 등록할 Route 53 영역입니다.

8 클러스터의 **install-config.yaml** 파일을 생성할 때 사용한 Route 53 기본 도메인을 지정합니다. AWS 콘솔에 표시되는 후행 마침표(.)는 포함하지 마십시오.

9 VPC용으로 만든 퍼블릭 서브넷입니다.

10 VPC에 대한 CloudFormation 템플릿의 출력에서 **PublicSubnetIds** 값을 지정합니다.

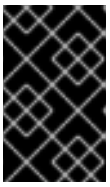
11 VPC용으로 만든 프라이빗 서브넷입니다.

12 VPC에 대한 CloudFormation 템플릿의 출력에서 **PrivateSubnetIds** 값을 지정합니다.

13 클러스터용으로 만든 VPC입니다.

14 VPC에 대한 CloudFormation 템플릿의 출력에서 **VpcId** 값을 지정합니다.

3. 이 항목의 **네트워크 및 로드 밸런서에 대한 CloudFormation 템플릿**섹션에서 템플릿을 복사하여 사용자 컴퓨터에 YAML 파일로 저장합니다. 이 템플릿은 클러스터에 필요한 네트워킹 및 로드 밸런싱 개체를 설명합니다.



중요

클러스터를 AWS 정부 또는 시크릿 리전에 배포하는 경우 **CNAME** 레코드를 사용하여 CloudFormation 템플릿에서 **InternalApiServerRecord**를 업데이트해야 합니다. **ALIAS** 유형의 레코드는 AWS 정부 리전에서 지원되지 않습니다.

4. CloudFormation 템플릿을 시작하여 네트워킹 및 로드 밸런싱 구성 요소를 제공하는 AWS 리소스 스택을 생성합니다.

중요

명령은 한 줄로 입력해야 합니다.

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
    --capabilities CAPABILITY_NAMED_IAM 4
```

1 **<name>**은 CloudFormation 스택의 이름입니다(예: **cluster-dns**). 클러스터를 제거하는 경우 이 스택의 이름이 필요합니다.

2 **<template>**은 저장한 CloudFormation 템플릿 YAML 파일의 상대 경로 및 이름입니다.

3 **<parameters>**는 CloudFormation 매개변수 JSON 파일의 상대 경로 및 이름입니다.

4 제공된 템플릿에서 일부 **AWS::IAM::Role** 리소스를 생성하므로 **CAPABILITY_NAMED_IAM** 기능을 명시적으로 선언해야 합니다.

출력 예

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-dns/cd3e5de0-2fd4-11eb-5cf0-12be5c33a183
```

5. 템플릿 구성 요소가 있는지 확인합니다.

```
$ aws cloudformation describe-stacks --stack-name <name>
```

StackStatus에 **CREATE_COMPLETE**이 표시된 후 다음 매개변수의 출력 값이 표시됩니다. 클러스터를 생성하기 위해 실행하는 다른 CloudFormation 템플릿에 이러한 매개변수 값을 제공해야 합니다.

PrivateHostedZoneId	프라이빗 DNS의 호스팅 영역 ID입니다.
ExternalApiLoadBalancerName	외부 API 로드 밸런서의 전체 이름입니다.
InternalApiLoadBalancerName	내부 API 로드 밸런서의 전체 이름입니다.
ApiServerDnsName	API 서버의 전체 호스트 이름입니다.
RegisterNlbIpTargetsLambda	이러한 로드 밸런서의 IP 대상 등록/등록 취소에 유용한 Lambda ARN입니다.
ExternalApiTargetGroupArn	외부 API 대상 그룹의 ARN입니다.
InternalApiTargetGroupArn	내부 API 대상 그룹의 ARN입니다.
InternalServiceTargetGroupArn	내부 서비스 대상 그룹의 ARN입니다.

4.13.10.1. 네트워크 및 로드 밸런서에 대한 CloudFormation 템플릿

다음 CloudFormation 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 네트워킹 개체 및 로드 밸런서를 배포할 수 있습니다.

예 4.57. 네트워크 및 로드 밸런서에 대한 CloudFormation 템플릿

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Network Elements (Route53 & LBs)
```

```
Parameters:
  ClusterName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Cluster name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, representative cluster name to use for host names and other identifying
names.
    Type: String
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag cloud resources and identify items owned or
used by the cluster.
    Type: String
  HostedZoneId:
    Description: The Route53 public zone ID to register the targets with, such as
Z21IXYZABCZ2A4.
    Type: String
  HostedZoneName:
    Description: The Route53 zone to register the targets with, such as example.com. Omit the
trailing period.
    Type: String
    Default: "example.com"
  PublicSubnets:
    Description: The internet-facing subnets.
    Type: List<AWS::EC2::Subnet::Id>
  PrivateSubnets:
    Description: The internal subnets.
    Type: List<AWS::EC2::Subnet::Id>
  VpcId:
    Description: The VPC-scoped resources will belong to this VPC.
    Type: AWS::EC2::VPC::Id

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - ClusterName
      - InfrastructureName
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcId
      - PublicSubnets
      - PrivateSubnets
    - Label:
        default: "DNS"
      Parameters:
      - HostedZoneName
```

```
      - HostedZoneId
    ParameterLabels:
      ClusterName:
        default: "Cluster Name"
      InfrastructureName:
        default: "Infrastructure Name"
      VpcId:
        default: "VPC ID"
      PublicSubnets:
        default: "Public Subnets"
      PrivateSubnets:
        default: "Private Subnets"
      HostedZoneName:
        default: "Public Hosted Zone Name"
      HostedZoneId:
        default: "Public Hosted Zone ID"

Resources:
  ExtApiElb:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Name: !Join ["-", [!Ref InfrastructureName, "ext"]]
      IpAddressType: ipv4
      Subnets: !Ref PublicSubnets
      Type: network

  IntApiElb:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Name: !Join ["-", [!Ref InfrastructureName, "int"]]
      Scheme: internal
      IpAddressType: ipv4
      Subnets: !Ref PrivateSubnets
      Type: network

  IntDns:
    Type: "AWS::Route53::HostedZone"
    Properties:
      HostedZoneConfig:
        Comment: "Managed by CloudFormation"
      Name: !Join [".", [!Ref ClusterName, !Ref HostedZoneName]]
      HostedZoneTags:
      - Key: Name
        Value: !Join ["-", [!Ref InfrastructureName, "int"]]
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "owned"
      VPCs:
      - VPCId: !Ref VpcId
        VPCRegion: !Ref "AWS::Region"

  ExternalApiServerRecord:
    Type: AWS::Route53::RecordSetGroup
    Properties:
      Comment: Alias record for the API server
      HostedZoneId: !Ref HostedZoneId
      RecordSets:
```

```
    - Name:
        !Join [
          ".",
          ["api", !Ref ClusterName, !Join ["", [!Ref HostedZoneName, "."]]],
        ]
      Type: A
      AliasTarget:
        HostedZoneId: !GetAtt ExtApiElb.CanonicalHostedZoneID
        DNSName: !GetAtt ExtApiElb.DNSName

InternalApiServerRecord:
  Type: AWS::Route53::RecordSetGroup
  Properties:
    Comment: Alias record for the API server
    HostedZoneId: !Ref IntDns
    RecordSets:
    - Name:
        !Join [
          ".",
          ["api", !Ref ClusterName, !Join ["", [!Ref HostedZoneName, "."]]],
        ]
      Type: A
      AliasTarget:
        HostedZoneId: !GetAtt IntApiElb.CanonicalHostedZoneID
        DNSName: !GetAtt IntApiElb.DNSName
    - Name:
        !Join [
          ".",
          ["api-int", !Ref ClusterName, !Join ["", [!Ref HostedZoneName, "."]]],
        ]
      Type: A
      AliasTarget:
        HostedZoneId: !GetAtt IntApiElb.CanonicalHostedZoneID
        DNSName: !GetAtt IntApiElb.DNSName

ExternalApiListener:
  Type: AWS::ElasticLoadBalancingV2::Listener
  Properties:
    DefaultActions:
    - Type: forward
      TargetGroupArn:
        Ref: ExternalApiTargetGroup
    LoadBalancerArn:
      Ref: ExtApiElb
    Port: 6443
    Protocol: TCP

ExternalApiTargetGroup:
  Type: AWS::ElasticLoadBalancingV2::TargetGroup
  Properties:
    HealthCheckIntervalSeconds: 10
    HealthCheckPath: "/readyz"
    HealthCheckPort: 6443
    HealthCheckProtocol: HTTPS
    HealthyThresholdCount: 2
    UnhealthyThresholdCount: 2
```

```
    Port: 6443
    Protocol: TCP
    TargetType: ip
    VpcId:
      Ref: VpcId
    TargetGroupAttributes:
    - Key: deregistration_delay.timeout_seconds
      Value: 60

  InternalApiListener:
    Type: AWS::ElasticLoadBalancingV2::Listener
    Properties:
      DefaultActions:
      - Type: forward
        TargetGroupArn:
          Ref: InternalApiTargetGroup
      LoadBalancerArn:
        Ref: IntApiElb
      Port: 6443
      Protocol: TCP

  InternalApiTargetGroup:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      HealthCheckIntervalSeconds: 10
      HealthCheckPath: "/readyz"
      HealthCheckPort: 6443
      HealthCheckProtocol: HTTPS
      HealthyThresholdCount: 2
      UnhealthyThresholdCount: 2
      Port: 6443
      Protocol: TCP
      TargetType: ip
      VpcId:
        Ref: VpcId
      TargetGroupAttributes:
      - Key: deregistration_delay.timeout_seconds
        Value: 60

  InternalServiceInternalListener:
    Type: AWS::ElasticLoadBalancingV2::Listener
    Properties:
      DefaultActions:
      - Type: forward
        TargetGroupArn:
          Ref: InternalServiceTargetGroup
      LoadBalancerArn:
        Ref: IntApiElb
      Port: 22623
      Protocol: TCP

  InternalServiceTargetGroup:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      HealthCheckIntervalSeconds: 10
      HealthCheckPath: "/healthz"
```

```
    HealthCheckPort: 22623
    HealthCheckProtocol: HTTPS
    HealthyThresholdCount: 2
    UnhealthyThresholdCount: 2
    Port: 22623
    Protocol: TCP
    TargetType: ip
    VpcId:
      Ref: VpcId
    TargetGroupAttributes:
    - Key: deregistration_delay.timeout_seconds
      Value: 60

  RegisterTargetLambdaIamRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: !Join ["-", [!Ref InfrastructureName, "nlb", "lambda", "role"]]
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
            - "lambda.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Path: "/"
      Policies:
      - PolicyName: !Join ["-", [!Ref InfrastructureName, "master", "policy"]]
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Action:
              [
                "elasticloadbalancing:RegisterTargets",
                "elasticloadbalancing:DeregisterTargets",
              ]
            Resource: !Ref InternalApiTargetGroup
          - Effect: "Allow"
            Action:
              [
                "elasticloadbalancing:RegisterTargets",
                "elasticloadbalancing:DeregisterTargets",
              ]
            Resource: !Ref InternalServiceTargetGroup
          - Effect: "Allow"
            Action:
              [
                "elasticloadbalancing:RegisterTargets",
                "elasticloadbalancing:DeregisterTargets",
              ]
            Resource: !Ref ExternalApiTargetGroup

  RegisterNlbIpTargets:
    Type: "AWS::Lambda::Function"
```

```
    Properties:
      Handler: "index.handler"
      Role:
        Fn::GetAtt:
        - "RegisterTargetLambdaIamRole"
        - "Arn"
      Code:
        ZipFile: |
          import json
          import boto3
          import cfnresponse
          def handler(event, context):
            elb = boto3.client('elbv2')
            if event['RequestType'] == 'Delete':
              elb.deregister_targets(TargetGroupArn=event['ResourceProperties']
['TargetArn'],Targets=[{'Id': event['ResourceProperties']['TargetIp']}])
            elif event['RequestType'] == 'Create':
              elb.register_targets(TargetGroupArn=event['ResourceProperties']['TargetArn'],Targets=
[{'Id': event['ResourceProperties']['TargetIp']}])
            responseData = {}
            cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData,
event['ResourceProperties']['TargetArn']+event['ResourceProperties']['TargetIp'])
      Runtime: "python3.7"
      Timeout: 120

  RegisterSubnetTagsLambdaIamRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: !Join ["-", [!Ref InfrastructureName, "subnet-tags-lambda-role"]]
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
            - "lambda.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Path: "/"
      Policies:
      - PolicyName: !Join ["-", [!Ref InfrastructureName, "subnet-tagging-policy"]]
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Action:
              [
                "ec2:DeleteTags",
                "ec2:CreateTags"
              ]
            Resource: "arn:aws:ec2:*:*:subnet/*"
          - Effect: "Allow"
            Action:
              [
                "ec2:DescribeSubnets",
                "ec2:DescribeTags"
```

```
            ]
          Resource: "*"

  RegisterSubnetTags:
    Type: "AWS::Lambda::Function"
    Properties:
      Handler: "index.handler"
      Role:
        Fn::GetAtt:
        - "RegisterSubnetTagsLambdaIamRole"
        - "Arn"
      Code:
        ZipFile: |
          import json
          import boto3
          import cfnresponse
          def handler(event, context):
            ec2_client = boto3.client('ec2')
            if event['RequestType'] == 'Delete':
              for subnet_id in event['ResourceProperties']['Subnets']:
                ec2_client.delete_tags(Resources=[subnet_id], Tags=[{'Key': 'kubernetes.io/cluster/' +
event['ResourceProperties']['InfrastructureName']}]);
            elif event['RequestType'] == 'Create':
              for subnet_id in event['ResourceProperties']['Subnets']:
                ec2_client.create_tags(Resources=[subnet_id], Tags=[{'Key': 'kubernetes.io/cluster/' +
event['ResourceProperties']['InfrastructureName'], 'Value': 'shared'}]);
            responseData = {}
            cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData,
event['ResourceProperties']['InfrastructureName']+event['ResourceProperties']['Subnets'][0])
      Runtime: "python3.7"
      Timeout: 120

  RegisterPublicSubnetTags:
    Type: Custom::SubnetRegister
    Properties:
      ServiceToken: !GetAtt RegisterSubnetTags.Arn
      InfrastructureName: !Ref InfrastructureName
      Subnets: !Ref PublicSubnets

  RegisterPrivateSubnetTags:
    Type: Custom::SubnetRegister
    Properties:
      ServiceToken: !GetAtt RegisterSubnetTags.Arn
      InfrastructureName: !Ref InfrastructureName
      Subnets: !Ref PrivateSubnets

Outputs:
  PrivateHostedZoneId:
    Description: Hosted zone ID for the private DNS, which is required for private records.
    Value: !Ref IntDns
  ExternalApiLoadBalancerName:
    Description: Full name of the external API load balancer.
    Value: !GetAtt ExtApiElb.LoadBalancerFullName
  InternalApiLoadBalancerName:
    Description: Full name of the internal API load balancer.
    Value: !GetAtt IntApiElb.LoadBalancerFullName
```

```
  ApiServerDnsName:
    Description: Full hostname of the API server, which is required for the Ignition config files.
    Value: !Join ["", ["api-int", !Ref ClusterName, !Ref HostedZoneName]]
  RegisterNlbIpTargetsLambda:
    Description: Lambda ARN useful to help register or deregister IP targets for these load
balancers.
    Value: !GetAtt RegisterNlbIpTargets.Arn
  ExternalApiTargetGroupArn:
    Description: ARN of the external API target group.
    Value: !Ref ExternalApiTargetGroup
  InternalApiTargetGroupArn:
    Description: ARN of the internal API target group.
    Value: !Ref InternalApiTargetGroup
  InternalServiceTargetGroupArn:
    Description: ARN of the internal service target group.
    Value: !Ref InternalServiceTargetGroup
```

중요

클러스터를 AWS 정부 또는 시크릿 리전에 배포하는 경우 **CNAME** 레코드를 사용하도록 **InternalApiServerRecord**를 업데이트해야 합니다. **ALIAS** 유형의 레코드는 AWS 정부 리전에서 지원되지 않습니다. 예를 들면 다음과 같습니다.

```
Type: CNAME
TTL: 10
ResourceRecords:
- !GetAtt IntApiElb.DNSName
```

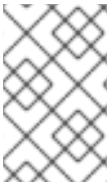
추가 리소스

- 공개 호스팅 영역 나열에 대한 자세한 내용은 AWS 문서의 공개 호스팅 영역 나열을 참조하십시오.

4.13.11. AWS에서 보안 그룹 및 역할 생성

OpenShift Container Platform 클러스터에서 사용할 보안 그룹 및 역할을 AWS(Amazon Web Services)에서 생성해야 합니다.

제공된 CloudFormation 템플릿과 사용자 정의 매개변수 파일을 사용하여 AWS 리소스 스택을 생성할 수 있습니다. 스택은 OpenShift Container Platform 클러스터에 필요한 보안 그룹 및 역할을 나타냅니다.



참고

AWS 인프라를 생성하는 데 제공된 CloudFormation 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- AWS 계정을 구성했습니다.
- **aws 구성**을 실행하여 AWS 키와 리전을 로컬 AWS 프로필에 추가하셨습니다.

- 클러스터에 대한 Ignition 구성 파일을 생성하셨습니다.
- AWS에서 VPC 및 관련 서브넷을 생성하고 구성하셨습니다.

프로세스

1. 템플릿에 필요한 매개변수 값이 포함된 JSON 파일을 생성합니다.

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "VpcCidr", 3
    "ParameterValue": "10.0.0.0/16" 4
  },
  {
    "ParameterKey": "PrivateSubnets", 5
    "ParameterValue": "subnet-<random_string>" 6
  },
  {
    "ParameterKey": "VpcId", 7
    "ParameterValue": "vpc-<random_string>" 8
  }
]
```

1. 클러스터에 대해 Ignition 구성 파일에 인코딩되는 클러스터 인프라의 이름입니다.
2. Ignition 구성 파일 메타데이터에서 추출한 인프라 이름을 **<cluster-name>-<random-string>** 형식으로 지정합니다.
3. VPC의 CIDR 블록입니다.
4. **xxxx/16-24** 양식으로 정의하고 VPC에 사용한 CIDR 블록 매개변수를 지정합니다.
5. VPC용으로 만든 프라이빗 서브넷입니다.
6. VPC에 대한 CloudFormation 템플릿의 출력에서 **PrivateSubnetIds** 값을 지정합니다.
7. 클러스터용으로 만든 VPC입니다.
8. VPC에 대한 CloudFormation 템플릿의 출력에서 **VpcId** 값을 지정합니다.

2. 이 항목의 **보안 개체에 대한 CloudFormation 템플릿** 섹션에서 템플릿을 복사하여 사용자 컴퓨터에 YAML 파일로 저장합니다. 이 템플릿은 클러스터에 필요한 보안 그룹 및 역할을 설명합니다.
3. CloudFormation 템플릿을 시작하여 보안 그룹 및 역할을 나타내는 AWS 리소스 스택을 생성합니다.

중요

명령은 한 줄로 입력해야 합니다.

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
    --capabilities CAPABILITY_NAMED_IAM 4
```

1. **<name>**은 CloudFormation 스택의 이름입니다(예: **cluster-sec**). 클러스터를 제거하는 경우 이 스택의 이름이 필요합니다.
2. **<template>**은 저장한 CloudFormation 템플릿 YAML 파일의 상대 경로 및 이름입니다.
3. **<parameters>**는 CloudFormation 매개변수 JSON 파일의 상대 경로 및 이름입니다.
4. 제공된 템플릿에서 일부 **AWS::IAM::Role** 및 **AWS::IAM::InstanceProfile** 리소스를 생성하므로 **CAPABILITY_NAMED_IAM** 기능을 명시적으로 선언해야 합니다.

출력 예

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-sec/03bd4210-2ed7-11eb-6d7a-13fc0b61e9db
```

4. 템플릿 구성 요소가 있는지 확인합니다.

```
$ aws cloudformation describe-stacks --stack-name <name>
```

StackStatus에 **CREATE_COMPLETE**이 표시된 후 다음 매개변수의 출력 값이 표시됩니다. 클러스터를 생성하기 위해 실행하는 다른 CloudFormation 템플릿에 이러한 매개변수 값을 제공해야 합니다.

MasterSecurityGroupId	마스터 보안 그룹 ID
WorkerSecurityGroupId	작업자 보안 그룹 ID
MasterInstanceProfile	마스터 IAM 인스턴스 프로필
WorkerInstanceProfile	작업자 IAM 인스턴스 프로필

4.13.11.1. 보안 개체의 CloudFormation 템플릿

다음 CloudFormation 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 보안 개체를 배포할 수 있습니다.

예 4.58. 보안 개체의 CloudFormation 템플릿

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Security Elements (Security Groups & IAM)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag cloud resources and identify items owned or
used by the cluster.
    Type: String
  VpcCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-
4][0-9]|25[0-5])(\/(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.0.0/16
    Description: CIDR block for VPC.
    Type: String
  VpcId:
    Description: The VPC-scoped resources will belong to this VPC.
    Type: AWS::EC2::VPC::Id
  PrivateSubnets:
    Description: The internal subnets.
    Type: List<AWS::EC2::Subnet::Id>

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - InfrastructureName
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcId
      - VpcCidr
      - PrivateSubnets
    ParameterLabels:
      InfrastructureName:
        default: "Infrastructure Name"
      VpcId:
        default: "VPC ID"
      VpcCidr:
        default: "VPC CIDR"
      PrivateSubnets:
        default: "Private Subnets"

Resources:
  MasterSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Cluster Master Security Group
      SecurityGroupIngress:
```

```
      - IpProtocol: icmp
        FromPort: 0
        ToPort: 0
        CidrIp: !Ref VpcCidr
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: !Ref VpcCidr
      - IpProtocol: tcp
        ToPort: 6443
        FromPort: 6443
        CidrIp: !Ref VpcCidr
      - IpProtocol: tcp
        FromPort: 22623
        ToPort: 22623
        CidrIp: !Ref VpcCidr
      VpcId: !Ref VpcId

  WorkerSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Cluster Worker Security Group
      SecurityGroupIngress:
      - IpProtocol: icmp
        FromPort: 0
        ToPort: 0
        CidrIp: !Ref VpcCidr
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: !Ref VpcCidr
      VpcId: !Ref VpcId

  MasterIngressEtcd:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: etcd
      FromPort: 2379
      ToPort: 2380
      IpProtocol: tcp

  MasterIngressVxlan:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Vxlan packets
      FromPort: 4789
      ToPort: 4789
      IpProtocol: udp

  MasterIngressWorkerVxlan:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
```

```
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Vxlan packets
    FromPort: 4789
    ToPort: 4789
    IpProtocol: udp

MasterIngressGeneve:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Geneve packets
    FromPort: 6081
    ToPort: 6081
    IpProtocol: udp

MasterIngressWorkerGeneve:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Geneve packets
    FromPort: 6081
    ToPort: 6081
    IpProtocol: udp

MasterIngressIpsecIke:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: IPsec IKE packets
    FromPort: 500
    ToPort: 500
    IpProtocol: udp

MasterIngressIpsecNat:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: IPsec NAT-T packets
    FromPort: 4500
    ToPort: 4500
    IpProtocol: udp

MasterIngressIpsecEsp:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: IPsec ESP packets
    IpProtocol: 50

MasterIngressWorkerIpsecIke:
```

```
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: IPsec IKE packets
    FromPort: 500
    ToPort: 500
    IpProtocol: udp

MasterIngressWorkerIpsecNat:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: IPsec NAT-T packets
    FromPort: 4500
    ToPort: 4500
    IpProtocol: udp

MasterIngressWorkerIpsecEsp:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: IPsec ESP packets
    IpProtocol: 50

MasterIngressInternal:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Internal cluster communication
    FromPort: 9000
    ToPort: 9999
    IpProtocol: tcp

MasterIngressWorkerInternal:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Internal cluster communication
    FromPort: 9000
    ToPort: 9999
    IpProtocol: tcp

MasterIngressInternalUDP:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Internal cluster communication
    FromPort: 9000
    ToPort: 9999
    IpProtocol: udp
```

```
MasterIngressWorkerInternalUDP:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Internal cluster communication
    FromPort: 9000
    ToPort: 9999
    IpProtocol: udp

MasterIngressKube:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Kubernetes kubelet, scheduler and controller manager
    FromPort: 10250
    ToPort: 10259
    IpProtocol: tcp

MasterIngressWorkerKube:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Kubernetes kubelet, scheduler and controller manager
    FromPort: 10250
    ToPort: 10259
    IpProtocol: tcp

MasterIngressIngressServices:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: tcp

MasterIngressWorkerIngressServices:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: tcp

MasterIngressIngressServicesUDP:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
```

```
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: udp

  MasterIngressWorkerIngressServicesUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: udp

  WorkerIngressVxlan:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Vxlan packets
      FromPort: 4789
      ToPort: 4789
      IpProtocol: udp

  WorkerIngressMasterVxlan:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Vxlan packets
      FromPort: 4789
      ToPort: 4789
      IpProtocol: udp

  WorkerIngressGeneve:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Geneve packets
      FromPort: 6081
      ToPort: 6081
      IpProtocol: udp

  WorkerIngressMasterGeneve:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Geneve packets
      FromPort: 6081
      ToPort: 6081
      IpProtocol: udp

  WorkerIngressIpsecIke:
```

```
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: IPsec IKE packets
    FromPort: 500
    ToPort: 500
    IpProtocol: udp

WorkerIngressIpsecNat:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: IPsec NAT-T packets
    FromPort: 4500
    ToPort: 4500
    IpProtocol: udp

WorkerIngressIpsecEsp:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: IPsec ESP packets
    IpProtocol: 50

WorkerIngressMasterIpsecIke:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: IPsec IKE packets
    FromPort: 500
    ToPort: 500
    IpProtocol: udp

WorkerIngressMasterIpsecNat:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: IPsec NAT-T packets
    FromPort: 4500
    ToPort: 4500
    IpProtocol: udp

WorkerIngressMasterIpsecEsp:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: IPsec ESP packets
    IpProtocol: 50

WorkerIngressInternal:
```

```
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Internal cluster communication
    FromPort: 9000
    ToPort: 9999
    IpProtocol: tcp

WorkerIngressMasterInternal:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Internal cluster communication
    FromPort: 9000
    ToPort: 9999
    IpProtocol: tcp

WorkerIngressInternalUDP:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Internal cluster communication
    FromPort: 9000
    ToPort: 9999
    IpProtocol: udp

WorkerIngressMasterInternalUDP:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Internal cluster communication
    FromPort: 9000
    ToPort: 9999
    IpProtocol: udp

WorkerIngressKube:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Kubernetes secure kubelet port
    FromPort: 10250
    ToPort: 10250
    IpProtocol: tcp

WorkerIngressWorkerKube:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Internal Kubernetes communication
    FromPort: 10250
```

```
    ToPort: 10250
    IpProtocol: tcp

  WorkerIngressIngressServices:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: tcp

  WorkerIngressMasterIngressServices:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: tcp

  WorkerIngressIngressServicesUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: udp

  WorkerIngressMasterIngressServicesUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: udp

  MasterIamRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
            - "ec2.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Policies:
```

```
    - PolicyName: !Join ["-", [!Ref InfrastructureName, "master", "policy"]]
      PolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Action:
          - "ec2:AttachVolume"
          - "ec2:AuthorizeSecurityGroupIngress"
          - "ec2:CreateSecurityGroup"
          - "ec2:CreateTags"
          - "ec2:CreateVolume"
          - "ec2:DeleteSecurityGroup"
          - "ec2:DeleteVolume"
          - "ec2:Describe*"
          - "ec2:DetachVolume"
          - "ec2:ModifyInstanceAttribute"
          - "ec2:ModifyVolume"
          - "ec2:RevokeSecurityGroupIngress"
          - "elasticloadbalancing:AddTags"
          - "elasticloadbalancing:AttachLoadBalancerToSubnets"
          - "elasticloadbalancing:ApplySecurityGroupsToLoadBalancer"
          - "elasticloadbalancing:CreateListener"
          - "elasticloadbalancing:CreateLoadBalancer"
          - "elasticloadbalancing:CreateLoadBalancerPolicy"
          - "elasticloadbalancing:CreateLoadBalancerListeners"
          - "elasticloadbalancing:CreateTargetGroup"
          - "elasticloadbalancing:ConfigureHealthCheck"
          - "elasticloadbalancing:DeleteListener"
          - "elasticloadbalancing:DeleteLoadBalancer"
          - "elasticloadbalancing:DeleteLoadBalancerListeners"
          - "elasticloadbalancing:DeleteTargetGroup"
          - "elasticloadbalancing:DeregisterInstancesFromLoadBalancer"
          - "elasticloadbalancing:DeregisterTargets"
          - "elasticloadbalancing:Describe*"
          - "elasticloadbalancing:DetachLoadBalancerFromSubnets"
          - "elasticloadbalancing:ModifyListener"
          - "elasticloadbalancing:ModifyLoadBalancerAttributes"
          - "elasticloadbalancing:ModifyTargetGroup"
          - "elasticloadbalancing:ModifyTargetGroupAttributes"
          - "elasticloadbalancing:RegisterInstancesWithLoadBalancer"
          - "elasticloadbalancing:RegisterTargets"
          - "elasticloadbalancing:SetLoadBalancerPoliciesForBackendServer"
          - "elasticloadbalancing:SetLoadBalancerPoliciesOfListener"
          - "kms:DescribeKey"
          Resource: "*"

  MasterInstanceProfile:
    Type: "AWS::IAM::InstanceProfile"
    Properties:
      Roles:
      - Ref: "MasterIamRole"

  WorkerIamRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
```

```
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
            - "ec2.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Policies:
      - PolicyName: !Join ["-", [!Ref InfrastructureName, "worker", "policy"]]
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Action:
            - "ec2:DescribeInstances"
            - "ec2:DescribeRegions"
            Resource: "*"

  WorkerInstanceProfile:
    Type: "AWS::IAM::InstanceProfile"
    Properties:
      Roles:
      - Ref: "WorkerIamRole"

Outputs:
  MasterSecurityGroupId:
    Description: Master Security Group ID
    Value: !GetAtt MasterSecurityGroup.GroupId

  WorkerSecurityGroupId:
    Description: Worker Security Group ID
    Value: !GetAtt WorkerSecurityGroup.GroupId

  MasterInstanceProfile:
    Description: Master IAM Instance Profile
    Value: !Ref MasterInstanceProfile

  WorkerInstanceProfile:
    Description: Worker IAM Instance Profile
    Value: !Ref WorkerInstanceProfile
```

4.13.12. 스트림 메타데이터를 사용하여 RHCOS AMI에 액세스

OpenShift Container Platform에서 *스트림 메타데이터*는 JSON 형식으로 RHCOS에 대한 표준화된 메타데이터를 제공하고 클러스터에 메타데이터를 삽입합니다. 스트림 메타데이터는 여러 아키텍처를 지원하는 안정적인 형식이며 자동화 유지하기 위해 자체 문서화하도록 설계되었습니다.

openshift-install의 **coreos print-stream-json** 하위 명령을 사용하여 스트림 메타데이터 형식의 부트 이미지에 대한 정보에 액세스할 수 있습니다. 이 명령은 스트림 메타데이터를 스크립트가 가능하고 컴퓨터가 읽을 수 있는 형식으로 인쇄하는 방법을 제공합니다.

사용자 프로비저닝 설치의 경우 **openshift-install** 바이너리에는 AWS AMI와 같은 OpenShift Container Platform과 함께 사용하기 위해 테스트된 RHCOS 부팅 이미지에 대한 참조가 포함되어 있습니다.

절차

스트림 메타데이터를 구문 분석하려면 다음 방법 중 하나를 사용합니다.

- Go 프로그램에서는 https://github.com/coreos/stream-metadata-go에서의 공식 **stream-metadata-go** 라이브러리를 사용합니다. 라이브러리에서 예제 코드를 볼 수도 있습니다.
- Python 또는 Ruby와 같은 다른 프로그래밍 언어에서 선호하는 프로그래밍 언어의 JSON 라이브러리를 사용합니다.
- **jq**와 같은 JSON 데이터를 처리하는 명령줄 유틸리티에서 다음을 수행합니다.
  - AWS 리전의 현재 **x86_64** AMI(예: **us-west-1**)를 출력합니다.

```
$ openshift-install coreos print-stream-json | jq -r
'.architectures.x86_64.images.aws.regions["us-west-1"].image'
```

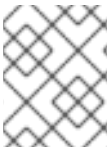
출력 예

```
ami-0d3e625f84626bbda
```

이 명령의 출력은 **us-west-1** 리전의 AWS AMI ID입니다. AMI는 클러스터와 동일한 리전에 속해 있어야 합니다.

4.13.13. AWS 인프라용 RHCOS AMI

Red Hat은 OpenShift Container Platform 노드에 수동으로 지정할 수 있는 다양한 AWS 영역에 유효한 RHCOS(Red Hat Enterprise Linux CoreOS) AMI를 제공합니다.



참고

사용자 고유의 AMI를 가져 와서 RHCOS AMI가 게시되지 않은 리전에 설치할 수도 있습니다.

표 4.46. RHCOS AMI

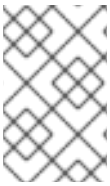
AWS 영역	AWS AMI
af-south-1	**ami-0f3804f5a2f913dcc**
ap-east-1	**ami-0de1febb30a83da66**
ap-northeast-1	**ami-0183df96a3e002687**
ap-northeast-2	**ami-06b8798cd60242798**
ap-northeast-3	**ami-00b16b33aa0951016**
ap-south-1	**ami-007243f8ff78e8294**
ap-southeast-1	**ami-079dfdacb5ab5a0d1**

AWS 영역	AWS AMI
ap-southeast-2	**ami-03882e39cb7785c32**
ca-central-1	**ami-05cba1f80cc8b1dbe**
eu-central-1	**ami-073c775bbe9cd434e**
eu-north-1	**ami-0763e6e75b681acc5**
eu-south-1	**ami-00d023f19775fb64b**
eu-west-1	**ami-0033e3f2331a530c4**
eu-west-2	**ami-00d8a741ebe74f0c4**
eu-west-3	**ami-09b04e7f60e3374a7**
me-south-1	**ami-0f8039330b6e54010**
sa-east-1	**ami-01af22f821b470ad1**
us-east-1	**ami-0c72f473496a7b1c2**
us-east-2	**ami-09e637fc5885c13cc**
us-west-1	**ami-0fa0f6fce7e63dd26**
us-west-2	**ami-084fb1316cd1ed4cc**

4.13.14. AWS에서 부트스트랩 노드 생성

OpenShift Container Platform 클러스터 초기화 과정에서 사용할 부트스트랩 노드를 AWS(Amazon Web Services)에서 생성해야 합니다.

제공된 CloudFormation 템플릿과 사용자 정의 매개변수 파일을 사용하여 AWS 리소스 스택을 생성할 수 있습니다. 스택은 OpenShift Container Platform 설치에 필요한 부트스트랩 노드를 나타냅니다.



참고

부트스트랩 노드를 생성하는 데 제공된 CloudFormation 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다.. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

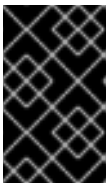
사전 요구 사항

- AWS 계정을 구성했습니다.

- **aws 구성**을 실행하여 AWS 키와 리전을 로컬 AWS 프로필에 추가하셨습니다.
- 클러스터에 대한 Ignition 구성 파일을 생성하셨습니다.
- AWS에서 VPC 및 관련 서브넷을 생성하고 구성하셨습니다.
- AWS에서 DNS, 로드 밸런서 및 리스너를 생성하고 구성하셨습니다.
- AWS에서 클러스터에 필요한 보안 그룹 및 역할을 생성하셨습니다.

프로세스

1. **bootstrap.ign** Ignition 구성 파일을 클러스터에 제공할 위치를 지정합니다. 이 파일은 설치 디렉터리에 있습니다. 한 가지 방법은 클러스터의 리전에 S3 버킷을 생성하고 여기에 Ignition 구성 파일을 업로드하는 것입니다.

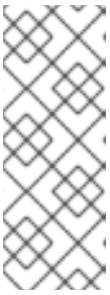


중요

제공된 CloudFormation 템플릿은 클러스터의 Ignition 구성 파일이 S3 버킷에서 제공되는 것으로 가정합니다. 다른 위치에서 파일을 제공하려면 템플릿을 수정해야 합니다.

중요

AWS SDK와 다른 엔드 포인트가 있는 리전에 배포하거나 자체 사용자 지정 엔드 포인트를 제공하는 경우 **s3://** 스키마 대신 사전에 서명된 URL을 S3 버킷에 사용해야 합니다.



참고

부트스트랩 Ignition 구성 파일에는 X.509 키와 같은 시크릿이 포함되어 있습니다. 다음 단계는 S3 버킷에 대한 기본 보안을 제공합니다. 추가 보안을 제공하기 위해 OpenShift IAM 사용자와 같은 특정 사용자만 버킷에 포함된 개체에 액세스할 수 있도록 S3 버킷 정책을 활성화할 수 있습니다. S3과 관계없이 부트스트랩 시스템이 도달할 수 있는 모든 주소에서 부트스트랩 Ignition 구성 파일을 제공할 수 있습니다.

a. 버킷을 만듭니다.

```
$ aws s3 mb s3://<cluster-name>-infra 1
```

1 **<cluster-name>-infra**는 버킷 이름입니다. **install-config.yaml** 파일을 생성할 때 **<cluster-name>**을 클러스터에 지정된 이름으로 교체합니다.

b. **bootstrap.ign** Ignition 구성 파일을 버킷에 업로드합니다.

```
$ aws s3 cp <installation_directory>/bootstrap.ign s3://<cluster-name>-
infra/bootstrap.ign 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

c. 파일이 업로드되었는지 확인합니다.

```
$ aws s3 ls s3://<cluster-name>-infra/
```

출력 예

```
2019-04-03 16:15:16     314878 bootstrap.ign
```

2. 템플릿에 필요한 매개변수 값이 포함된 JSON 파일을 생성합니다.

```
[
 {
  "ParameterKey": "InfrastructureName", 1
  "ParameterValue": "mycluster-<random_string>" 2
 },
 {
  "ParameterKey": "RhcosAmi", 3
  "ParameterValue": "ami-<random_string>" 4
 },
 {
  "ParameterKey": "AllowedBootstrapSshCidr", 5
  "ParameterValue": "0.0.0.0/0" 6
 },
 {
  "ParameterKey": "PublicSubnet", 7
  "ParameterValue": "subnet-<random_string>" 8
 },
 {
  "ParameterKey": "MasterSecurityGroupId", 9
  "ParameterValue": "sg-<random_string>" 10
 },
 {
  "ParameterKey": "VpcId", 11
  "ParameterValue": "vpc-<random_string>" 12
 },
 {
  "ParameterKey": "BootstrapIgnitionLocation", 13
  "ParameterValue": "s3://<bucket_name>/bootstrap.ign" 14
 },
 {
  "ParameterKey": "AutoRegisterELB", 15
  "ParameterValue": "yes" 16
 },
 {
  "ParameterKey": "RegisterNlbIpTargetsLambdaArn", 17
  "ParameterValue": "arn:aws:lambda:<region>:<account_number>:function:
<dns_stack_name>-RegisterNlbIpTargets-<random_string>" 18
 },
 {
  "ParameterKey": "ExternalApiTargetGroupArn", 19
  "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Exter-<random_string>" 20
 },
 {
```

```
    "ParameterKey": "InternalApiTargetGroupArn", 21
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 22
  },
  {
    "ParameterKey": "InternalServiceTargetGroupArn", 23
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 24
  }
]
```

1. 클러스터에 대해 Ignition 구성 파일에 인코딩되는 클러스터 인프라의 이름입니다.
2. Ignition 구성 파일 메타데이터에서 추출한 인프라 이름을 **<cluster-name>-<random-string>** 형식으로 지정합니다.
3. 부트스트랩 노드에 사용할 현재 RHCOS(Red Hat Enterprise Linux CoreOS) AMI입니다.
4. 유효한 **AWS::EC2::Image::Id** 값을 지정합니다.
5. 부트스트랩 노드에 대한 SSH 액세스를 허용하는 CIDR 블록입니다.
6. **xxxx/16-24** 형식으로 CIDR 블록을 지정합니다.
7. 부트스트랩 노드를 시작하기 위해 VPC와 연결되는 퍼블릭 서브넷입니다.
8. VPC에 대한 CloudFormation 템플릿의 출력에서 **PublicSubnetIds** 값을 지정합니다.
9. 마스터 보안 그룹 ID(임시 규칙 등록용)입니다.
10. 보안 그룹 및 역할에 대한 CloudFormation 템플릿 출력에서 **MasterSecurityGroupId** 값을 지정합니다.
11. VPC 생성 리소스가 속합니다.
12. VPC에 대한 CloudFormation 템플릿의 출력에서 **VpcId** 값을 지정합니다.
13. 부트스트랩 Ignition 구성 파일을 가져올 위치입니다.
14. S3 버킷과 파일 이름을 **s3://<bucket_name>/bootstrap.ign** 형식으로 지정합니다.
15. 네트워크 로드 밸런서(NLB) 등록 여부입니다.
16. **yes** 또는 **no**를 지정합니다. **yes**를 지정하는 경우 Lambda ARN(Amazon Resource Name) 값을 제공해야 합니다.
17. NLB IP 대상 등록 lambda 그룹의 ARN입니다.
18. DNS 및 로드 밸런싱을 위해 CloudFormation 템플릿의 출력에서 **RegisterNlbIpTargetsLambda** 값을 지정합니다. 클러스터를 AWS GovCloud 리전에 배포하는 경우 **arn:aws-us-gov**를 사용합니다.
19. 외부 API 로드 밸런서 대상 그룹의 ARN입니다.
20. DNS 및 로드 밸런싱의 CloudFormation 템플릿 출력에서 **ExternalApiTargetGroupArn** 값을 지정합니다. 클러스터를 AWS GovCloud 리전에 배포하는 경우 **arn:aws-us-gov**를 사용합니다.

21 내부 API 로드 밸런서 대상 그룹의 ARN입니다.

22 DNS 및 로드 밸런싱의 CloudFormation 템플릿 출력에서 **InternalApiTargetGroupArn** 값을 지정합니다. 클러스터를 AWS GovCloud 리전에 배포하는 경우 **arn:aws-us-gov**를 사용합니다.

23 내부 서비스 로드 밸런서 대상 그룹의 ARN입니다.

24 DNS 및 로드 밸런싱의 CloudFormation 템플릿 출력에서 **InternalServiceTargetGroupArn** 값을 지정합니다. 클러스터를 AWS GovCloud 리전에 배포하는 경우 **arn:aws-us-gov**를 사용합니다.

3. 이 항목의 **부트스트랩 시스템의 CloudFormation 템플릿** 섹션에서 템플릿을 복사하여 사용자 시스템에 YAML 파일로 저장합니다. 이 템플릿은 클러스터에 필요한 부트스트랩 시스템을 설명합니다.

4. CloudFormation 템플릿을 시작하여 부트스트랩 노드를 나타내는 AWS 리소스 스택을 생성합니다.

중요

명령은 한 줄로 입력해야 합니다.

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
    --capabilities CAPABILITY_NAMED_IAM 4
```

1 **<name>**은 CloudFormation 스택의 이름입니다(예: **cluster-bootstrap**). 클러스터를 제거하는 경우 이 스택의 이름이 필요합니다.

2 **<template>**은 저장한 CloudFormation 템플릿 YAML 파일의 상대 경로 및 이름입니다.

3 **<parameters>**는 CloudFormation 매개변수 JSON 파일의 상대 경로 및 이름입니다.

4 제공된 템플릿에서 일부 **AWS::IAM::Role** 및 **AWS::IAM::InstanceProfile** 리소스를 생성하므로 **CAPABILITY_NAMED_IAM** 기능을 명시적으로 선언해야 합니다.

출력 예

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-bootstrap/12944486-2add-11eb-9dee-12dace8e3a83
```

5. 템플릿 구성 요소가 있는지 확인합니다.

```
$ aws cloudformation describe-stacks --stack-name <name>
```

StackStatus에 **CREATE_COMPLETE**이 표시된 후 다음 매개변수의 출력 값이 표시됩니다. 클러스터를 생성하기 위해 실행하는 다른 CloudFormation 템플릿에 이러한 매개변수 값을 제공해야 합니다.

Bootstrap InstanceId	부트스트랩 인스턴스 ID입니다.
Bootstrap PublicIp	부트스트랩 노드 공용 IP 주소입니다.
Bootstrap PrivateIp	부트스트랩 노드 개인 IP 주소입니다.

4.13.14.1. 부트스트랩 시스템용 CloudFormation 템플릿

다음 CloudFormation 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 부트스트랩 시스템을 배포할 수 있습니다.

예 4.59. 부트스트랩 시스템용 CloudFormation 템플릿

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Bootstrap (EC2 Instance, Security Groups and IAM)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag cloud resources and identify items owned or
used by the cluster.
    Type: String
  RhcosAmi:
    Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.
    Type: AWS::EC2::Image::Id
  AllowedBootstrapSshCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-
4][0-9]|25[0-5])(\/([0-9]|1[0-9]|2[0-9]|3[0-2]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/0-32.
    Default: 0.0.0.0/0
    Description: CIDR block to allow SSH access to the bootstrap node.
    Type: String
  PublicSubnet:
    Description: The public subnet to launch the bootstrap node into.
    Type: AWS::EC2::Subnet::Id
  MasterSecurityGroupId:
    Description: The master security group ID for registering temporary rules.
    Type: AWS::EC2::SecurityGroup::Id
  VpcId:
    Description: The VPC-scoped resources will belong to this VPC.
    Type: AWS::EC2::VPC::Id
  BootstrapIgnitionLocation:
    Default: s3://my-s3-bucket/bootstrap.ign
    Description: Ignition config file location.
    Type: String
  AutoRegisterELB:
```

```
    Default: "yes"
    AllowedValues:
    - "yes"
    - "no"
    Description: Do you want to invoke NLB registration, which requires a Lambda ARN parameter?
    Type: String
  RegisterNlbIpTargetsLambdaArn:
    Description: ARN for NLB IP target registration lambda.
    Type: String
  ExternalApiTargetGroupArn:
    Description: ARN for external API load balancer target group.
    Type: String
  InternalApiTargetGroupArn:
    Description: ARN for internal API load balancer target group.
    Type: String
  InternalServiceTargetGroupArn:
    Description: ARN for internal service load balancer target group.
    Type: String

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - InfrastructureName
    - Label:
        default: "Host Information"
      Parameters:
      - RhcosAmi
      - BootstrapIgnitionLocation
      - MasterSecurityGroupId
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcId
      - AllowedBootstrapSshCidr
      - PublicSubnet
    - Label:
        default: "Load Balancer Automation"
      Parameters:
      - AutoRegisterELB
      - RegisterNlbIpTargetsLambdaArn
      - ExternalApiTargetGroupArn
      - InternalApiTargetGroupArn
      - InternalServiceTargetGroupArn
    ParameterLabels:
      InfrastructureName:
        default: "Infrastructure Name"
      VpcId:
        default: "VPC ID"
      AllowedBootstrapSshCidr:
        default: "Allowed SSH Source"
      PublicSubnet:
        default: "Public Subnet"
      RhcosAmi:
```

```
      default: "Red Hat Enterprise Linux CoreOS AMI ID"
    BootstrapIgnitionLocation:
      default: "Bootstrap Ignition Source"
    MasterSecurityGroupId:
      default: "Master Security Group ID"
    AutoRegisterELB:
      default: "Use Provided ELB Automation"

Conditions:
  DoRegistration: !Equals ["yes", !Ref AutoRegisterELB]

Resources:
  BootstrapIamRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
            - "ec2.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Path: "/"
      Policies:
      - PolicyName: !Join ["-", [!Ref InfrastructureName, "bootstrap", "policy"]]
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Action: "ec2:Describe*"
            Resource: "*"
          - Effect: "Allow"
            Action: "ec2:AttachVolume"
            Resource: "*"
          - Effect: "Allow"
            Action: "ec2:DetachVolume"
            Resource: "*"
          - Effect: "Allow"
            Action: "s3:GetObject"
            Resource: "*"

  BootstrapInstanceProfile:
    Type: "AWS::IAM::InstanceProfile"
    Properties:
      Path: "/"
      Roles:
      - Ref: "BootstrapIamRole"

  BootstrapSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Cluster Bootstrap Security Group
      SecurityGroupIngress:
      - IpProtocol: tcp
```

```
      FromPort: 22
      ToPort: 22
      CidrIp: !Ref AllowedBootstrapSshCidr
    - IpProtocol: tcp
      ToPort: 19531
      FromPort: 19531
      CidrIp: 0.0.0.0/0
    VpcId: !Ref VpcId

BootstrapInstance:
  Type: AWS::EC2::Instance
  Properties:
    ImageId: !Ref RhcosAmi
    IamInstanceProfile: !Ref BootstrapInstanceProfile
    InstanceType: "i3.large"
    NetworkInterfaces:
    - AssociatePublicIpAddress: "true"
      DeviceIndex: "0"
      GroupSet:
      - !Ref "BootstrapSecurityGroup"
      - !Ref "MasterSecurityGroupId"
      SubnetId: !Ref "PublicSubnet"
    UserData:
      Fn::Base64: !Sub
      - '{"ignition":{"config":{"replace":{"source":"${S3Loc}"}},"version":"3.1.0"}}'
      - {
        S3Loc: !Ref BootstrapIgnitionLocation
      }

RegisterBootstrapApiTarget:
  Condition: DoRegistration
  Type: Custom::NLBRegister
  Properties:
    ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
    TargetArn: !Ref ExternalApiTargetGroupArn
    TargetIp: !GetAtt BootstrapInstance.PrivateIp

RegisterBootstrapInternalApiTarget:
  Condition: DoRegistration
  Type: Custom::NLBRegister
  Properties:
    ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
    TargetArn: !Ref InternalApiTargetGroupArn
    TargetIp: !GetAtt BootstrapInstance.PrivateIp

RegisterBootstrapInternalServiceTarget:
  Condition: DoRegistration
  Type: Custom::NLBRegister
  Properties:
    ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
    TargetArn: !Ref InternalServiceTargetGroupArn
    TargetIp: !GetAtt BootstrapInstance.PrivateIp

Outputs:
  BootstrapInstanceId:
    Description: Bootstrap Instance ID.
```

```
    Value: !Ref BootstrapInstance

BootstrapPublicIp:
  Description: The bootstrap node public IP address.
  Value: !GetAtt BootstrapInstance.PublicIp

BootstrapPrivateIp:
  Description: The bootstrap node private IP address.
  Value: !GetAtt BootstrapInstance.PrivateIp
```

추가 리소스

- AWS 영역의 RHCOS(Red Hat Enterprise Linux CoreOS) AMI에 대한 자세한 내용은 AWS 인프라의 RHCOS AMI를 참조하십시오.

4.13.15. AWS에서 컨트롤 플레인 시스템 생성

클러스터가 사용할 컨트롤 플레인 시스템을 AWS(Amazon Web Services)에서 생성해야 합니다.

제공된 CloudFormation 템플릿과 사용자 정의 매개변수 파일을 사용하여 컨트롤 플레인 노드를 나타내는 AWS 리소스 스택을 생성할 수 있습니다.

중요

CloudFormation 템플릿은 세 개의 컨트롤 플레인 노드를 나타내는 스택을 생성합니다.

참고

컨트롤 플레인 노드를 생성하는 데 제공된 CloudFormation 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다.. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- AWS 계정을 구성했습니다.
- **aws 구성**을 실행하여 AWS 키와 리전을 로컬 AWS 프로필에 추가하셨습니다.
- 클러스터에 대한 Ignition 구성 파일을 생성하셨습니다.
- AWS에서 VPC 및 관련 서브넷을 생성하고 구성하셨습니다.
- AWS에서 DNS, 로드 밸런서 및 리스너를 생성하고 구성하셨습니다.
- AWS에서 클러스터에 필요한 보안 그룹 및 역할을 생성하셨습니다.
- 부트스트랩 시스템을 생성하셨습니다.

프로세스

1. 템플릿에 필요한 매개변수 값이 포함된 JSON 파일을 생성합니다.

```
[
```

```
{
  "ParameterKey": "InfrastructureName", (1)
  "ParameterValue": "mycluster-<random_string>" (2)
},
{
  "ParameterKey": "RhcosAmi", (3)
  "ParameterValue": "ami-<random_string>" (4)
},
{
  "ParameterKey": "AutoRegisterDNS", (5)
  "ParameterValue": "yes" (6)
},
{
  "ParameterKey": "PrivateHostedZoneId", (7)
  "ParameterValue": "<random_string>" (8)
},
{
  "ParameterKey": "PrivateHostedZoneName", (9)
  "ParameterValue": "mycluster.example.com" (10)
},
{
  "ParameterKey": "Master0Subnet", (11)
  "ParameterValue": "subnet-<random_string>" (12)
},
{
  "ParameterKey": "Master1Subnet", (13)
  "ParameterValue": "subnet-<random_string>" (14)
},
{
  "ParameterKey": "Master2Subnet", (15)
  "ParameterValue": "subnet-<random_string>" (16)
},
{
  "ParameterKey": "MasterSecurityGroupId", (17)
  "ParameterValue": "sg-<random_string>" (18)
},
{
  "ParameterKey": "IgnitionLocation", (19)
  "ParameterValue": "https://api-int.<cluster_name>.<domain_name>:22623/config/master"
(20)
},
{
  "ParameterKey": "CertificateAuthorities", (21)
  "ParameterValue": "data:text/plain;charset=utf-8;base64,ABC...xYz==" (22)
},
{
  "ParameterKey": "MasterInstanceProfileName", (23)
  "ParameterValue": "<roles_stack>-MasterInstanceProfile-<random_string>" (24)
},
{
  "ParameterKey": "MasterInstanceType", (25)
  "ParameterValue": "m5.xlarge" (26)
```

```
  },
  {
    "ParameterKey": "AutoRegisterELB", 27
    "ParameterValue": "yes" 28
  },
  {
    "ParameterKey": "RegisterNlbIpTargetsLambdaArn", 29
    "ParameterValue": "arn:aws:lambda:<region>:<account_number>:function:
<dns_stack_name>-RegisterNlbIpTargets-<random_string>" 30
  },
  {
    "ParameterKey": "ExternalApiTargetGroupArn", 31
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Exter-<random_string>" 32
  },
  {
    "ParameterKey": "InternalApiTargetGroupArn", 33
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 34
  },
  {
    "ParameterKey": "InternalServiceTargetGroupArn", 35
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 36
  }
]
```

1. 클러스터에 대해 Ignition 구성 파일에 인코딩되는 클러스터 인프라의 이름입니다.
2. Ignition 구성 파일 메타데이터에서 추출한 인프라 이름을 **<cluster-name>-<random-string>** 형식으로 지정합니다.
3. 컨트롤 플레인 시스템에 사용할 현재 RHCOS(Red Hat Enterprise Linux CoreOS) AMI입니다.
4. 유효한 **AWS::EC2::Image::Id** 값을 지정합니다.
5. DNS etcd 등록을 수행할지 여부입니다.
6. **yes** 또는 **no**를 지정합니다. **yes**를 지정하는 경우 호스팅 영역 정보를 제공해야 합니다.
7. etcd 대상을 등록할 Route 53 프라이빗 영역 ID입니다.
8. DNS 및 로드 밸런싱을 위해 CloudFormation 템플릿의 출력에서 **PrivateHostedZoneId** 값을 지정합니다.
9. 대상을 등록할 Route 53 영역입니다.
10. **<cluster_name>.<domain_name>**을 지정합니다. 여기서 **<domain_name>**은 클러스터에 대한 **install-config.yaml** 파일을 생성할 때 사용한 Route 53 기본 도메인입니다. AWS 콘솔에 표시되는 후행 마침표(.)는 포함하지 마십시오.

11 13 15 컨트롤 플레인 시스템을 시작하기 위한 서브넷(프라이빗)입니다.

12 14 16

DNS 및 로드 밸런싱을 위해 CloudFormation 템플릿의 출력에서 **PrivateSubnets** 값의 서브넷을 지정합니다.

17 컨트롤 플레인 노드와 연결할 마스터 보안 그룹 ID입니다.

18 보안 그룹 및 역할에 대한 CloudFormation 템플릿 출력에서 **MasterSecurityGroupId** 값을 지정합니다.

19 컨트롤 플레인 Ignition 구성 파일을 가져올 위치입니다.

20 생성된 Ignition 구성 파일 위치(**https://api-int.<cluster_name>.<domain_name>:22623/config/master**)를 지정합니다.

21 사용할 base64로 인코딩 인증 기관 문자열입니다.

22 설치 디렉터리에 있는 **master.ign** 파일에서 값을 지정합니다. 이 값은 **data:text/plain;charset=utf-8;base64,ABC...xYz==** 형식의 긴 문자열입니다.

23 컨트롤 플레인 노드와 연결할 IAM 프로필입니다.

24 보안 그룹 및 역할에 대한 CloudFormation 템플릿 출력에서 **MasterInstanceProfile** 매개변수 값을 지정합니다.

25 컨트롤 플레인 시스템에 사용할 AWS 인스턴스 유형입니다.

26 허용되는 값:

- **m4.xlarge**
- **m4.2xlarge**
- **m4.4xlarge**
- **m4.10xlarge**
- **m4.16xlarge**
- **m5.xlarge**
- **m5.2xlarge**
- **m5.4xlarge**
- **m5.8xlarge**
- **m5.12xlarge**
- **m5.16xlarge**
- **m5a.xlarge**
- **m5a.2xlarge**
- **m5a.4xlarge**
- **m5a.8xlarge**
- **m5a.12xlarge**

- **m5a.16xlarge**
- **c4.2xlarge**
- **c4.4xlarge**
- **c4.8xlarge**
- **c5.2xlarge**
- **c5.4xlarge**
- **c5.9xlarge**
- **c5.12xlarge**
- **c5.18xlarge**
- **c5.24xlarge**
- **c5a.2xlarge**
- **c5a.4xlarge**
- **c5a.8xlarge**
- **c5a.12xlarge**
- **c5a.16xlarge**
- **c5a.24xlarge**
- **r4.xlarge**
- **r4.2xlarge**
- **r4.4xlarge**
- **r4.8xlarge**
- **r4.16xlarge**
- **r5.xlarge**
- **r5.2xlarge**
- **r5.4xlarge**
- **r5.8xlarge**
- **r5.12xlarge**
- **r5.16xlarge**
- **r5.24xlarge**
- **r5a.xlarge**

- **r5a.2xlarge**
- **r5a.4xlarge**
- **r5a.8xlarge**
- **r5a.12xlarge**
- **r5a.16xlarge**
- **r5a.24xlarge**

27 네트워크 로드 밸런서(NLB) 등록 여부입니다.

28 **yes** 또는 **no**를 지정합니다. **yes**를 지정하는 경우 Lambda ARN(Amazon Resource Name) 값을 제공해야 합니다.

29 NLB IP 대상 등록 lambda 그룹의 ARN입니다.

30 DNS 및 로드 밸런싱을 위해 CloudFormation 템플릿의 출력에서 **RegisterNlbIpTargetsLambda** 값을 지정합니다. 클러스터를 AWS GovCloud 리전에 배포하는 경우 **arn:aws-us-gov**를 사용합니다.

31 외부 API 로드 밸런서 대상 그룹의 ARN입니다.

32 DNS 및 로드 밸런싱의 CloudFormation 템플릿 출력에서 **ExternalApiTargetGroupArn** 값을 지정합니다. 클러스터를 AWS GovCloud 리전에 배포하는 경우 **arn:aws-us-gov**를 사용합니다.

33 내부 API 로드 밸런서 대상 그룹의 ARN입니다.

34 DNS 및 로드 밸런싱의 CloudFormation 템플릿 출력에서 **InternalApiTargetGroupArn** 값을 지정합니다. 클러스터를 AWS GovCloud 리전에 배포하는 경우 **arn:aws-us-gov**를 사용합니다.

35 내부 서비스 로드 밸런서 대상 그룹의 ARN입니다.

36 DNS 및 로드 밸런싱의 CloudFormation 템플릿 출력에서 **InternalServiceTargetGroupArn** 값을 지정합니다. 클러스터를 AWS GovCloud 리전에 배포하는 경우 **arn:aws-us-gov**를 사용합니다.

2. 이 항목의 **컨트롤 플레인 시스템에 대한 CloudFormation 템플릿** 섹션에서 템플릿을 복사하여 사용자 시스템에 YAML 파일로 저장합니다. 이 템플릿은 클러스터에 필요한 컨트롤 플레인 시스템을 설명합니다.

3. **MasterInstanceType** 값으로 **M5** 인스턴스 유형을 지정한 경우 CloudFormation 템플릿의 **MasterInstanceType.AllowedValues** 매개변수에 해당 인스턴스 유형을 추가합니다.

4. CloudFormation 템플릿을 시작하여 컨트롤 플레인 노드를 나타내는 AWS 리소스 스택을 생성합니다.

중요

명령은 한 줄로 입력해야 합니다.

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
```

1 **<name>**은 CloudFormation 스택의 이름입니다(예: **cluster-control-plane**). 클러스터를 제거하는 경우 이 스택의 이름이 필요합니다.

2 **<template>**은 저장한 CloudFormation 템플릿 YAML 파일의 상대 경로 및 이름입니다.

3 **<parameters>**는 CloudFormation 매개변수 JSON 파일의 상대 경로 및 이름입니다.

출력 예

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-control-plane/21c7e2b0-2ee2-11eb-c6f6-0aa34627df4b
```

참고

CloudFormation 템플릿은 세 개의 컨트롤 플레인 노드를 나타내는 스택을 생성합니다.

5. 템플릿 구성 요소가 있는지 확인합니다.

```
$ aws cloudformation describe-stacks --stack-name <name>
```

4.13.15.1. 컨트롤 플레인 시스템에 대한 CloudFormation 템플릿

다음 CloudFormation 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 컨트롤 플레인 시스템을 배포할 수 있습니다.

예 4.60. 컨트롤 플레인 시스템에 대한 CloudFormation 템플릿

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Node Launch (EC2 master instances)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag nodes for the kubelet cloud provider.
    Type: String
  RhcosAmi:
    Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.
    Type: AWS::EC2::Image::Id
  AutoRegisterDNS:
    Default: "yes"
    AllowedValues:
    - "yes"
    - "no"
```

```
    Description: Do you want to invoke DNS etcd registration, which requires Hosted Zone
information?
    Type: String
  PrivateHostedZoneId:
    Description: The Route53 private zone ID to register the etcd targets with, such as
Z21IXYZABCZ2A4.
    Type: String
  PrivateHostedZoneName:
    Description: The Route53 zone to register the targets with, such as cluster.example.com. Omit
the trailing period.
    Type: String
  Master0Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  Master1Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  Master2Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  MasterSecurityGroupId:
    Description: The master security group ID to associate with master nodes.
    Type: AWS::EC2::SecurityGroup::Id
  IgnitionLocation:
    Default: https://api-int.$CLUSTER_NAME.$DOMAIN:22623/config/master
    Description: Ignition config file location.
    Type: String
  CertificateAuthorities:
    Default: data:text/plain;charset=utf-8;base64,ABC...xYz==
    Description: Base64 encoded certificate authority string to use.
    Type: String
  MasterInstanceProfileName:
    Description: IAM profile to associate with master nodes.
    Type: String
  MasterInstanceType:
    Default: m5.xlarge
    Type: String

  AutoRegisterELB:
    Default: "yes"
    AllowedValues:
    - "yes"
    - "no"
    Description: Do you want to invoke NLB registration, which requires a Lambda ARN parameter?
    Type: String
  RegisterNlbIpTargetsLambdaArn:
    Description: ARN for NLB IP target registration lambda. Supply the value from the cluster
infrastructure or select "no" for AutoRegisterELB.
    Type: String
  ExternalApiTargetGroupArn:
    Description: ARN for external API load balancer target group. Supply the value from the cluster
infrastructure or select "no" for AutoRegisterELB.
    Type: String
  InternalApiTargetGroupArn:
    Description: ARN for internal API load balancer target group. Supply the value from the cluster
infrastructure or select "no" for AutoRegisterELB.
```

```
    Type: String
  InternalServiceTargetGroupArn:
    Description: ARN for internal service load balancer target group. Supply the value from the
cluster infrastructure or select "no" for AutoRegisterELB.
    Type: String

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - InfrastructureName
    - Label:
        default: "Host Information"
      Parameters:
      - MasterInstanceType
      - RhcosAmi
      - IgnitionLocation
      - CertificateAuthorities
      - MasterSecurityGroupId
      - MasterInstanceProfileName
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcId
      - AllowedBootstrapSshCidr
      - Master0Subnet
      - Master1Subnet
      - Master2Subnet
    - Label:
        default: "DNS"
      Parameters:
      - AutoRegisterDNS
      - PrivateHostedZoneName
      - PrivateHostedZoneId
    - Label:
        default: "Load Balancer Automation"
      Parameters:
      - AutoRegisterELB
      - RegisterNlbIpTargetsLambdaArn
      - ExternalApiTargetGroupArn
      - InternalApiTargetGroupArn
      - InternalServiceTargetGroupArn
    ParameterLabels:
      InfrastructureName:
        default: "Infrastructure Name"
      VpcId:
        default: "VPC ID"
      Master0Subnet:
        default: "Master-0 Subnet"
      Master1Subnet:
        default: "Master-1 Subnet"
      Master2Subnet:
        default: "Master-2 Subnet"
      MasterInstanceType:
```

```
      default: "Master Instance Type"
    MasterInstanceProfileName:
      default: "Master Instance Profile Name"
    RhcosAmi:
      default: "Red Hat Enterprise Linux CoreOS AMI ID"
    BootstrapIgnitionLocation:
      default: "Master Ignition Source"
    CertificateAuthorities:
      default: "Ignition CA String"
    MasterSecurityGroupId:
      default: "Master Security Group ID"
    AutoRegisterDNS:
      default: "Use Provided DNS Automation"
    AutoRegisterELB:
      default: "Use Provided ELB Automation"
    PrivateHostedZoneName:
      default: "Private Hosted Zone Name"
    PrivateHostedZoneId:
      default: "Private Hosted Zone ID"

Conditions:
  DoRegistration: !Equals ["yes", !Ref AutoRegisterELB]
  DoDns: !Equals ["yes", !Ref AutoRegisterDNS]

Resources:
  Master0:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref MasterInstanceProfileName
      InstanceType: !Ref MasterInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "Master0Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"merge":[{"source":"${SOURCE}"}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}"}]}},"version":"3.1.0"}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

  RegisterMaster0:
    Condition: DoRegistration
```

```
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref ExternalApiTargetGroupArn
      TargetIp: !GetAtt Master0.PrivateIp

  RegisterMaster0InternalApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalApiTargetGroupArn
      TargetIp: !GetAtt Master0.PrivateIp

  RegisterMaster0InternalServiceTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalServiceTargetGroupArn
      TargetIp: !GetAtt Master0.PrivateIp

  Master1:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref MasterInstanceProfileName
      InstanceType: !Ref MasterInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "Master1Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"merge":[{"source":"${SOURCE}"}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}"}]}},"version":"3.1.0"}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

  RegisterMaster1:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
```

```
      TargetArn: !Ref ExternalApiTargetGroupArn
      TargetIp: !GetAtt Master1.PrivateIp

  RegisterMaster1InternalApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalApiTargetGroupArn
      TargetIp: !GetAtt Master1.PrivateIp

  RegisterMaster1InternalServiceTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalServiceTargetGroupArn
      TargetIp: !GetAtt Master1.PrivateIp

  Master2:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref MasterInstanceProfileName
      InstanceType: !Ref MasterInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "Master2Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"merge":[{"source":"${SOURCE}"}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}"}]}},"version":"3.1.0"}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

  RegisterMaster2:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref ExternalApiTargetGroupArn
      TargetIp: !GetAtt Master2.PrivateIp
```

```
  RegisterMaster2InternalApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalApiTargetGroupArn
      TargetIp: !GetAtt Master2.PrivateIp

  RegisterMaster2InternalServiceTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalServiceTargetGroupArn
      TargetIp: !GetAtt Master2.PrivateIp

  EtcdSrvRecords:
    Condition: DoDns
    Type: AWS::Route53::RecordSet
    Properties:
      HostedZoneId: !Ref PrivateHostedZoneId
      Name: !Join [".", ["_etcd-server-ssl._tcp", !Ref PrivateHostedZoneName]]
      ResourceRecords:
      - !Join [
        " ",
        ["0 10 2380", !Join [".", ["etcd-0", !Ref PrivateHostedZoneName]]],
      ]
      - !Join [
        " ",
        ["0 10 2380", !Join [".", ["etcd-1", !Ref PrivateHostedZoneName]]],
      ]
      - !Join [
        " ",
        ["0 10 2380", !Join [".", ["etcd-2", !Ref PrivateHostedZoneName]]],
      ]
      TTL: 60
      Type: SRV

  Etcd0Record:
    Condition: DoDns
    Type: AWS::Route53::RecordSet
    Properties:
      HostedZoneId: !Ref PrivateHostedZoneId
      Name: !Join [".", ["etcd-0", !Ref PrivateHostedZoneName]]
      ResourceRecords:
      - !GetAtt Master0.PrivateIp
      TTL: 60
      Type: A

  Etcd1Record:
    Condition: DoDns
    Type: AWS::Route53::RecordSet
    Properties:
      HostedZoneId: !Ref PrivateHostedZoneId
      Name: !Join [".", ["etcd-1", !Ref PrivateHostedZoneName]]
      ResourceRecords:
```

```
      - !GetAtt Master1.PrivateIp
      TTL: 60
      Type: A

  Etcd2Record:
    Condition: DoDns
    Type: AWS::Route53::RecordSet
    Properties:
      HostedZoneId: !Ref PrivateHostedZoneId
      Name: !Join [".", ["etcd-2", !Ref PrivateHostedZoneName]]
      ResourceRecords:
      - !GetAtt Master2.PrivateIp
      TTL: 60
      Type: A

Outputs:
  PrivateIPs:
    Description: The control-plane node private IP addresses.
    Value:
      !Join [
        ",",
        [!GetAtt Master0.PrivateIp, !GetAtt Master1.PrivateIp, !GetAtt Master2.PrivateIp]
      ]
```

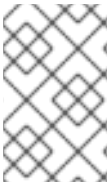
4.13.16. AWS에서 작업자 노드 생성

클러스터가 사용할 작업자 노드를 AWS(Amazon Web Services)에서 생성할 수 있습니다.

제공된 CloudFormation 템플릿과 사용자 정의 매개변수 파일을 사용하여 작업자 노드를 나타내는 AWS 리소스 스택을 생성할 수 있습니다.

중요

CloudFormation 템플릿은 하나의 작업자 노드를 나타내는 스택을 생성합니다. 각 작업자 노드의 스택을 생성해야 합니다.



참고

작업자 노드를 생성하는 데 제공된 CloudFormation 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다.. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- AWS 계정을 구성했습니다.
- **aws 구성**을 실행하여 AWS 키와 리전을 로컬 AWS 프로필에 추가하셨습니다.
- 클러스터에 대한 Ignition 구성 파일을 생성하셨습니다.
- AWS에서 VPC 및 관련 서브넷을 생성하고 구성하셨습니다.
- AWS에서 DNS, 로드 밸런서 및 리스너를 생성하고 구성하셨습니다.

- AWS에서 클러스터에 필요한 보안 그룹 및 역할을 생성하셨습니다.
- 부트스트랩 시스템을 생성하셨습니다.
- 컨트롤 플레인 시스템을 생성하셨습니다.

프로세스

1. CloudFormation 템플릿에 필요한 매개변수 값이 포함된 JSON 파일을 만듭니다.

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "RhcosAmi", 3
    "ParameterValue": "ami-<random_string>" 4
  },
  {
    "ParameterKey": "Subnet", 5
    "ParameterValue": "subnet-<random_string>" 6
  },
  {
    "ParameterKey": "WorkerSecurityGroupId", 7
    "ParameterValue": "sg-<random_string>" 8
  },
  {
    "ParameterKey": "IgnitionLocation", 9
    "ParameterValue": "https://api-int.<cluster_name>.<domain_name>:22623/config/worker"
10
  },
  {
    "ParameterKey": "CertificateAuthorities", 11
    "ParameterValue": "" 12
  },
  {
    "ParameterKey": "WorkerInstanceProfileName", 13
    "ParameterValue": "" 14
  },
  {
    "ParameterKey": "WorkerInstanceType", 15
    "ParameterValue": "m4.2xlarge" 16
  }
]
```

1. 클러스터에 대해 Ignition 구성 파일에 인코딩되는 클러스터 인프라의 이름입니다.
2. Ignition 구성 파일 메타데이터에서 추출한 인프라 이름을 **<cluster-name>-<random-string>** 형식으로 지정합니다.
3. 작업자 노드에 사용할 현재 RHCOS(Red Hat Enterprise Linux CoreOS) AMI입니다.

4 유효한 **AWS::EC2::Image::Id** 값을 지정합니다.

5 작업자 노드를 시작하기 위한 서브넷(프라이빗)입니다.

6 DNS 및 로드 밸런싱을 위해 CloudFormation 템플릿의 출력에서 **PrivateSubnets** 값의 서브넷을 지정합니다.

7 작업자 노드와 연결할 작업자 보안 그룹 ID입니다.

8 보안 그룹 및 역할에 대한 CloudFormation 템플릿 출력에서 **WorkerSecurityGroupId** 값을 지정합니다.

9 부트스트랩 Ignition 구성 파일을 가져올 위치입니다.

10 생성된 Ignition 구성 위치(**https://api-int.<cluster_name>.<domain_name>:22623/config/worker**)를 지정합니다.

11 사용할 base64로 인코딩 인증 기관 문자열입니다.

12 설치 디렉터리에 있는 **worker.ign** 파일에서 값을 지정합니다. 이 값은 **data:text/plain;charset=utf-8;base64,ABC...xYz==** 형식의 긴 문자열입니다.

13 작업자 노드와 연결할 IAM 프로필입니다.

14 보안 그룹 및 역할에 대한 CloudFormation 템플릿 출력에서 **WorkerInstanceProfile** 매개변수 값을 지정합니다.

15 컨트롤 플레인 시스템에 사용할 AWS 인스턴스 유형입니다.

16 허용되는 값:

- **m4.large**
- **m4.xlarge**
- **m4.2xlarge**
- **m4.4xlarge**
- **m4.10xlarge**
- **m4.16xlarge**
- **m5.large**
- **m5.xlarge**
- **m5.2xlarge**
- **m5.4xlarge**
- **m5.8xlarge**
- **m5.12xlarge**
- **m5.16xlarge**
- **m5a.large**

- **m5a.xlarge**
- **m5a.2xlarge**
- **m5a.4xlarge**
- **m5a.8xlarge**
- **m5a.12xlarge**
- **m5a.16xlarge**
- **c4.large**
- **c4.xlarge**
- **c4.2xlarge**
- **c4.4xlarge**
- **c4.8xlarge**
- **c5.large**
- **c5.xlarge**
- **c5.2xlarge**
- **c5.4xlarge**
- **c5.9xlarge**
- **c5.12xlarge**
- **c5.18xlarge**
- **c5.24xlarge**
- **c5a.large**
- **c5a.xlarge**
- **c5a.2xlarge**
- **c5a.4xlarge**
- **c5a.8xlarge**
- **c5a.12xlarge**
- **c5a.16xlarge**
- **c5a.24xlarge**
- **r4.large**
- **r4.xlarge**

- **r4.2xlarge**
- **r4.4xlarge**
- **r4.8xlarge**
- **r4.16xlarge**
- **r5.large**
- **r5.xlarge**
- **r5.2xlarge**
- **r5.4xlarge**
- **r5.8xlarge**
- **r5.12xlarge**
- **r5.16xlarge**
- **r5.24xlarge**
- **r5a.large**
- **r5a.xlarge**
- **r5a.2xlarge**
- **r5a.4xlarge**
- **r5a.8xlarge**
- **r5a.12xlarge**
- **r5a.16xlarge**
- **r5a.24xlarge**
- **t3.large**
- **t3.xlarge**
- **t3.2xlarge**
- **t3a.large**
- **t3a.xlarge**
- **t3a.2xlarge**

2. 이 항목의 **작업자 시스템에 대한 CloudFormation 템플릿** 섹션에서 템플릿을 복사하여 사용자 시스템에 YAML 파일로 저장합니다. 이 템플릿은 클러스터에 필요한 네트워킹 개체 및 로드 밸런서를 설명합니다.

3. 선택 사항: **WorkerInstanceType** 값으로 M **5** 인스턴스 유형을 지정한 경우 CloudFormation 템플릿의 **WorkerInstanceType.AllowedValues** 매개변수에 해당 인스턴스 유형을 추가합니다.

4. 선택 사항: AWS Marketplace 이미지로 배포하는 경우 서브스크립션에서 얻은 AMI ID로 **Worker0.type.properties.ImageID** 매개변수를 업데이트합니다.

5. CloudFormation 템플릿을 사용하여 작업자 노드를 나타내는 AWS 리소스 스택을 생성합니다.

중요

명령은 한 줄로 입력해야 합니다.

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml \ 2
    --parameters file://<parameters>.json 3
```

1 **<name>**은 CloudFormation 스택의 이름입니다(예: **cluster-worker-1**). 클러스터를 제거하는 경우 이 스택의 이름이 필요합니다.

2 **<template>**은 저장한 CloudFormation 템플릿 YAML 파일의 상대 경로 및 이름입니다.

3 **<parameters>**는 CloudFormation 매개변수 JSON 파일의 상대 경로 및 이름입니다.

출력 예

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-worker-1/729ee301-1c2a-11eb-348f-sd9888c65b59
```

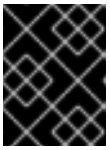
참고

CloudFormation 템플릿은 하나의 작업자 노드를 나타내는 스택을 생성합니다.

6. 템플릿 구성 요소가 있는지 확인합니다.

```
$ aws cloudformation describe-stacks --stack-name <name>
```

7. 클러스터에 충분한 작업자 시스템을 생성할 때까지 계속해서 작업자 스택을 생성합니다. 동일한 템플릿 및 매개변수 파일을 참조하고 다른 스택 이름을 지정하여 추가 작업자 스택을 생성할 수 있습니다.



중요

작업자 시스템을 두 개 이상 생성해야 하므로 이 CloudFormation 템플릿을 사용하는 스택을 두 개 이상 생성해야 합니다.

4.13.16.1. 작업자 시스템용 CloudFormation 템플릿

다음 CloudFormation 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 작업자 시스템을 배포할 수 있습니다.

예 4.61. 작업자 시스템용 CloudFormation 템플릿

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Node Launch (EC2 worker instance)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag nodes for the kubelet cloud provider.
    Type: String
  RhcosAmi:
    Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.
    Type: AWS::EC2::Image::Id
  Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  WorkerSecurityGroupId:
    Description: The master security group ID to associate with master nodes.
    Type: AWS::EC2::SecurityGroup::Id
  IgnitionLocation:
    Default: https://api-int.$CLUSTER_NAME.$DOMAIN:22623/config/worker
    Description: Ignition config file location.
    Type: String
  CertificateAuthorities:
    Default: data:text/plain;charset=utf-8;base64,ABC...xYz==
    Description: Base64 encoded certificate authority string to use.
    Type: String
  WorkerInstanceProfileName:
    Description: IAM profile to associate with master nodes.
    Type: String
  WorkerInstanceType:
    Default: m5.large
    Type: String

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - InfrastructureName
    - Label:
        default: "Host Information"
      Parameters:
      - WorkerInstanceType
      - RhcosAmi
      - IgnitionLocation
      - CertificateAuthorities
      - WorkerSecurityGroupId
      - WorkerInstanceProfileName
    - Label:
        default: "Network Configuration"
      Parameters:
      - Subnet
```

```
    ParameterLabels:
      Subnet:
        default: "Subnet"
      InfrastructureName:
        default: "Infrastructure Name"
      WorkerInstanceType:
        default: "Worker Instance Type"
      WorkerInstanceProfileName:
        default: "Worker Instance Profile Name"
      RhcosAmi:
        default: "Red Hat Enterprise Linux CoreOS AMI ID"
      IgnitionLocation:
        default: "Worker Ignition Source"
      CertificateAuthorities:
        default: "Ignition CA String"
      WorkerSecurityGroupId:
        default: "Worker Security Group ID"

Resources:
  Worker0:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref WorkerInstanceProfileName
      InstanceType: !Ref WorkerInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "WorkerSecurityGroupId"
        SubnetId: !Ref "Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"merge":[{"source":"${SOURCE}"}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}"}]}},"version":"3.1.0"}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

Outputs:
  PrivateIP:
    Description: The compute node private IP address.
    Value: !GetAtt Worker0.PrivateIp
```

4.13.17. 사용자 프로비저닝 인프라로 AWS에서 부트스트랩 시퀀스 초기화

AWS(Amazon Web Services)에서 필요한 인프라를 모두 생성한 후 OpenShift Container Platform 컨트롤 플레인을 초기화하는 부트스트랩 시퀀스를 시작할 수 있습니다.

사전 요구 사항

- AWS 계정을 구성했습니다.
- **aws 구성**을 실행하여 AWS 키와 리전을 로컬 AWS 프로필에 추가하셨습니다.
- 클러스터에 대한 Ignition 구성 파일을 생성하셨습니다.
- AWS에서 VPC 및 관련 서브넷을 생성하고 구성하셨습니다.
- AWS에서 DNS, 로드 밸런서 및 리스너를 생성하고 구성하셨습니다.
- AWS에서 클러스터에 필요한 보안 그룹 및 역할을 생성하셨습니다.
- 부트스트랩 시스템을 생성하셨습니다.
- 컨트롤 플레인 시스템을 생성하셨습니다.
- 작업자 노드를 생성하셨습니다.

프로세스

1. 설치 프로그램이 포함된 디렉터리로 변경하고 OpenShift Container Platform 컨트롤 플레인을 초기화하는 부트스트랩 프로세스를 시작합니다.

```
$ ./openshift-install wait-for bootstrap-complete --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2 다른 설치 세부 사항을 보려면 **info** 대신 **warn**, **debug** 또는 **error**를 지정합니다.

출력 예

```
INFO Waiting up to 20m0s for the Kubernetes API at
https://api.mycluster.example.com:6443...
INFO API v1.22.1 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
INFO Time elapsed: 1s
```

FATAL 경고 없이 명령이 종료되면 OpenShift Container Platform 컨트롤 플레인이 초기화된 것입니다.

참고

컨트롤 플레인이 초기화된 후 컴퓨팅 노드를 설정하고 Operator 형태로 추가 서비스를 설치합니다.

추가 리소스

- OpenShift Container Platform 설치가 진행되는 동안 설치, 부트스트랩 및 컨트롤 플레인 로그를 모니터링하는 데 대한 자세한 내용은 설치 진행 상황 모니터링 을 참조하십시오.
- 부트스트랩 프로세스와 관련된 문제 해결에 대한 정보는 부트스트랩 노드 진단 데이터 수집 을 참조하십시오.

4.13.18. CLI를 사용하여 클러스터에 로그인

클러스터 **kubeconfig** 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. **kubeconfig** 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- **oc** CLI를 설치했습니다.

프로세스

1. **kubeadmin** 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 **oc** 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

4.13.19. 머신의 인증서 서명 요청 승인

클러스터에 시스템을 추가하면 추가한 시스템별로 보류 중인 인증서 서명 요청(CSR)이 두 개씩 생성됩니다. 이러한 CSR이 승인되었는지 확인해야 하며, 필요한 경우 이를 직접 승인해야 합니다. 클라이언트 요청을 먼저 승인한 다음 서버 요청을 승인해야 합니다.

사전 요구 사항

- 클러스터에 시스템을 추가했습니다.

프로세스

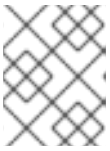
1. 클러스터가 시스템을 인식하는지 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0 Ready    master  63m  v1.22.1
master-1 Ready    master  63m  v1.22.1
master-2 Ready    master  64m  v1.22.1
```

출력에 생성된 모든 시스템이 나열됩니다.



참고

이전 출력에는 일부 CSR이 승인될 때까지 컴퓨팅 노드(작업자 노드라고도 함)가 포함되지 않을 수 있습니다.

2. 보류 중인 CSR을 검토하고 클러스터에 추가한 각 시스템에 대해 **Pending** 또는 **Approved** 상태의 클라이언트 및 서버 요청이 표시되는지 확인합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE    REQUESTOR                                                                   CONDITION
csr-8b2br   15m    system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
csr-8vnps   15m    system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
...
```

예에서는 두 시스템이 클러스터에 참여하고 있습니다. 목록에는 승인된 CSR이 더 많이 나타날 수도 있습니다.

3. CSR이 승인되지 않은 경우, 추가된 시스템에 대한 모든 보류 중인 CSR이 **Pending** 상태로 전환된 후 클러스터 시스템의 CSR을 승인합니다.

참고

CSR은 교체 주기가 자동으로 만료되므로 클러스터에 시스템을 추가한 후 1시간 이내에 CSR을 승인하십시오. 한 시간 내에 승인하지 않으면 인증서가 교체되고 각 노드에 대해 두 개 이상의 인증서가 표시됩니다. 이러한 인증서를 모두 승인해야 합니다. 클라이언트 CSR이 승인되면 Kubelet은 인증서에 대한 보조 CSR을 생성하므로 수동 승인이 필요합니다. 그러면 Kubelet에서 동일한 매개변수를 사용하여 새 인증서를 요청하는 경우 인증서 갱신 요청은 **machine-approver**에 의해 자동으로 승인됩니다.

참고

베어 메탈 및 기타 사용자 프로비저닝 인프라와 같이 머신 API를 사용하도록 활성화되지 않는 플랫폼에서 실행되는 클러스터의 경우 CSR(Kubelet service Certificate Request)을 자동으로 승인하는 방법을 구현해야 합니다. 요청이 승인되지 않으면 API 서버가 kubelet에 연결될 때 서비스 인증서가 필요하므로 **oc exec**, **oc rsh**, **oc logs** 명령을 성공적으로 수행할 수 없습니다. Kubelet 엔드 포인트에 연결하는 모든 작업을 수행하려면 이 인증서 승인이 필요합니다. 이 방법은 새 CSR을 감시하고 CSR이 **system:node** 또는 **system:admin** 그룹의 **node-bootstrapper** 서비스 계정에 의해 제출되었는지 확인하고 노드의 ID를 확인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>**은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

참고

일부 Operator는 일부 CSR이 승인될 때까지 사용할 수 없습니다.

4. 이제 클라이언트 요청이 승인되었으므로 클러스터에 추가한 각 머신의 서버 요청을 검토해야 합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 나머지 CSR이 승인되지 않고 **Pending** 상태인 경우 클러스터 머신의 CSR을 승인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>**은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. 모든 클라이언트 및 서버 CSR이 승인된 후 머신은 **Ready** 상태가 됩니다. 다음 명령을 실행하여 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME      STATUS    ROLES   AGE  VERSION
```

```
master-0  Ready    master  73m  v1.22.1
master-1  Ready    master  73m  v1.22.1
master-2  Ready    master  74m  v1.22.1
worker-0  Ready    worker  11m  v1.22.1
worker-1  Ready    worker  11m  v1.22.1
```

참고

머신이 **Ready** 상태로 전환하는 데 서버 CSR의 승인 후 몇 분이 걸릴 수 있습니다.

추가 정보

- CSR에 대한 자세한 내용은 인증서 서명 요청을 참조하십시오.

4.13.20. Operator의 초기 설정

컨트롤 플레인이 초기화된 후 일부 Operator를 즉시 구성하여 모두 사용 가능하도록 해야 합니다.

사전 요구 사항

- 컨트롤 플레인이 초기화되어 있습니다.

프로세스

1. 클러스터 구성 요소가 온라인 상태인지 확인합니다.

```
$ watch -n5 oc get clusteroperators
```

출력 예

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.9.0     True        False         False      19m
baremetal                                  4.9.0     True        False         False      37m
cloud-credential                           4.9.0     True        False         False      40m
cluster-autoscaler                         4.9.0     True        False         False      37m
config-operator                            4.9.0     True        False         False      38m
console                                    4.9.0     True        False         False      26m
csi-snapshot-controller                    4.9.0     True        False         False      37m
dns                                        4.9.0     True        False         False      37m
etcd                                       4.9.0     True        False         False      36m
image-registry                             4.9.0     True        False         False      31m
ingress                                    4.9.0     True        False         False      30m
insights                                   4.9.0     True        False         False      31m
kube-apiserver                             4.9.0     True        False         False      26m
kube-controller-manager                    4.9.0     True        False         False      36m
kube-scheduler                             4.9.0     True        False         False      36m
kube-storage-version-migrator              4.9.0     True        False         False      37m
machine-api                                4.9.0     True        False         False      29m
machine-approver                           4.9.0     True        False         False      37m
machine-config                             4.9.0     True        False         False      36m
marketplace                                4.9.0     True        False         False      37m
monitoring                                 4.9.0     True        False         False      29m
network                                    4.9.0     True        False         False      38m
```

```
node-tuning                                4.9.0     True        False         False      37m
openshift-apiserver                        4.9.0     True        False         False      32m
openshift-controller-manager               4.9.0     True        False         False      30m
openshift-samples                          4.9.0     True        False         False      32m
operator-lifecycle-manager                 4.9.0     True        False         False      37m
operator-lifecycle-manager-catalog         4.9.0     True        False         False      37m
operator-lifecycle-manager-packageserver   4.9.0     True        False         False      32m
service-ca                                 4.9.0     True        False         False      38m
storage                                    4.9.0     True        False         False      37m
```

2. 사용할 수 없는 Operator를 구성합니다.

4.13.20.1. 기본 OperatorHub 소스 비활성화

Red Hat 및 커뮤니티 프로젝트에서 제공하는 콘텐츠를 소싱하는 Operator 카탈로그는 OpenShift Container Platform을 설치하는 동안 기본적으로 OperatorHub용으로 구성됩니다. 제한된 네트워크 환경에서는 클러스터 관리자로서 기본 카탈로그를 비활성화해야 합니다.

프로세스

- **OperatorHub** 오브젝트에 **disableAllDefaultSources: true**를 추가하여 기본 카탈로그의 소스를 비활성화합니다.

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

작은 정보

또는 웹 콘솔을 사용하여 카탈로그 소스를 관리할 수 있습니다. **관리 → 클러스터 설정 → 구성 → OperatorHub** 페이지에서 개별 소스를 생성, 삭제, 비활성화 및 활성화할 수 있는 소스 **탭을** 클릭합니다.

4.13.20.2. 이미지 레지스트리 스토리지 구성

Amazon Web Services는 기본 스토리지를 제공하므로 설치 후 Image Registry Operator를 사용할 수 있습니다. 그러나 Registry Operator가 S3 버킷을 생성하고 스토리지를 자동으로 구성할 수 없는 경우에는 레지스트리 스토리지를 수동으로 구성해야 합니다.

프로덕션 클러스터에 필요한 영구 볼륨을 구성하는 과정의 지침이 표시됩니다. 해당하는 경우, 프로덕션 환경 외 클러스터에서만 사용할 수 있는 저장 위치로서 빈 디렉터리를 구성하는 과정의 지침이 표시됩니다.

업그레이드 중에 **Recreate** 롤아웃 전략을 사용하여 이미지 레지스트리의 블록 스토리지 유형 사용을 허용하기 위한 추가 지침이 제공됩니다.

4.13.20.2.1. 사용자 프로비저닝 인프라로 AWS용 레지스트리 스토리지 구성

설치하는 동안 클라우드 자격 증명만으로도 Amazon S3 버킷을 생성할 수 있으며 Registry Operator가 자동으로 스토리지를 구성합니다.

Registry Operator가 S3 버킷을 생성하고 스토리지를 자동으로 구성할수 없는 경우 다음 프로시저에 따라 S3 버킷을 생성하고 스토리지를 구성할 수 있습니다.

사전 요구 사항

- AWS에 사용자 프로비저닝된 인프라가 있는 클러스터가 있어야 합니다.
- Amazon S3 스토리지의 경우 시크릿에는 두 개의 키가 포함되어야 합니다.
  - **REGISTRY_STORAGE_S3_ACCESSKEY**
  - **REGISTRY_STORAGE_S3_SECRETKEY**

프로세스

Registry Operator가 S3 버킷을 생성하고 스토리지를 자동으로 구성할 수 없는 경우 다음 프로시저를 사용합니다.

1. 1일이 지난 완료되지 않은 다중 파트 업로드를 중단하도록 Bucket Lifecycle Policy 를 설정합니다.
2. **configs.imageregistry.operator.openshift.io/cluster**에 스토리지 설정을 입력합니다.

```
$ oc edit configs.imageregistry.operator.openshift.io/cluster
```

설정 예

```
storage:
  s3:
    bucket: <bucket-name>
    region: <region-name>
```

주의

AWS에서 레지스트리 이미지를 보안을 위해 S3 버킷에 공용 액세스를 차단합니다.

4.13.20.2.2. 프로덕션 환경 외 클러스터에서 이미지 레지스트리의 스토리지 구성

이미지 레지스트리 Operator에 대한 스토리지를 구성해야 합니다. 프로덕션 환경 외 클러스터의 경우, 이미지 레지스트리를 빈 디렉터리로 설정할 수 있습니다. 이렇게 하는 경우 레지스트리를 다시 시작하면 모든 이미지가 손실됩니다.

프로세스

- 이미지 레지스트리 스토리지를 빈 디렉터리로 설정하려면 다음을 수행하십시오.

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"storage":{"emptyDir":{}}}}'
```

주의

프로덕션 환경 외 클러스터에 대해서만 이 옵션을 구성하십시오.

Image Registry Operator가 구성 요소를 초기화하기 전에 이 명령을 실행하면 **oc patch** 명령이 실패하며 다음 오류가 발생합니다.

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

몇 분 후에 명령을 다시 실행하십시오.

4.13.21. 부트스트랩 리소스 삭제

클러스터에 대한 초기 Operator 구성을 완료한 후 AWS(Amazon Web Services)에서 부트스트랩 리소스를 제거합니다.

사전 요구 사항

- 클러스터에 대한 초기 Operator 구성을 완료했습니다.

프로세스

1. 부트스트랩 리소스를 삭제합니다. CloudFormation 템플릿을 사용한 경우 해당 스택을 삭제 합니다.
   - AWS CLI를 사용하여 스택 삭제:

     ```
     $ aws cloudformation delete-stack --stack-name <name> 1
     ```

     1 **<name>**은 부트스트랩 스택의 이름입니다.

   - AWS CloudFormation 콘솔을 사용하여 스택을 삭제합니다.

4.13.22. 인그레스 DNS 레코드 생성

DNS 영역 구성을 제거한 경우 인그레스 로드 밸런서를 가리키는 DNS 레코드를 수동으로 만듭니다. 와일드카드 레코드 또는 특정 레코드를 만들 수 있습니다. 다음 프로시저에서는 A 레코드를 사용하지만 CNAME 또는 별칭과 같이 필요한 다른 레코드 유형을 사용할 수 있습니다.

사전 요구 사항

- 프로비저닝한 인프라를 사용하는 AWS(Amazon Web Services)에 OpenShift Container Platform 클러스터를 배포했습니다.
- OpenShift CLI(**oc**)를 설치합니다.
- **jq** CLI를 설치하셨습니다.

- AWS CLI를 다운로드하여 컴퓨터에 설치했습니다. 번들 설치 관리자를 사용하여 AWS CLI 설치 (Linux, macOS 또는 Unix)를 참조하십시오.

프로세스

1. 생성할 경로를 결정합니다.
   - 와일드카드 레코드를 만들려면 ***.apps.<cluster_name>. <domain_name>**을 사용합니다. 여기서 **<cluster_name>**은 클러스터 이름이고 **<domain_name>**은 OpenShift Container Platform 클러스터의 Route 53 기본 도메인입니다.
   - 특정 레코드를 만들려면 다음 명령의 출력에 표시된 대로 클러스터가 사용하는 각 경로에 대한 레코드를 만들어야 합니다.

     ```
     $ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}
     {"\n"}{end}{end}' routes
     ```

     출력 예

     ```
     oauth-openshift.apps.<cluster_name>.<domain_name>
     console-openshift-console.apps.<cluster_name>.<domain_name>
     downloads-openshift-console.apps.<cluster_name>.<domain_name>
     alertmanager-main-openshift-monitoring.apps.<cluster_name>.<domain_name>
     grafana-openshift-monitoring.apps.<cluster_name>.<domain_name>
     prometheus-k8s-openshift-monitoring.apps.<cluster_name>.<domain_name>
     ```

2. Ingress Operator 로드 밸런서 상태를 검색하고 **EXTERNAL-IP** 열에 표시된 외부 IP 주소의 값을 확인합니다.

   ```
   $ oc -n openshift-ingress get service router-default
   ```

   출력 예

   ```
   NAME             TYPE           CLUSTER-IP      EXTERNAL-IP                            PORT(S)
   AGE
   router-default   LoadBalancer   172.30.62.215   ab3...28.us-east-2.elb.amazonaws.com
   80:31499/TCP,443:30693/TCP   5m
   ```

3. 로드 밸런서의 호스팅 영역 ID를 찾습니다.

   ```
   $ aws elb describe-load-balancers | jq -r '.LoadBalancerDescriptions[] | select(.DNSName ==
   "<external_ip>").CanonicalHostedZoneNameID' 1
   ```

   1 **<external_ip>**는 가져온 Ingress Operator 로드 밸런서의 외부 IP 주소값을 지정합니다.

   출력 예

   ```
   Z3AADJGX6KTTL2
   ```

   이 명령의 출력은 로드 밸런서 호스팅 영역 ID입니다.

4. 클러스터 도메인의 공개 호스팅 영역 ID를 가져옵니다.

```
$ aws route53 list-hosted-zones-by-name \
        --dns-name "<domain_name>" \ 1
        --query 'HostedZones[? Config.PrivateZone != `true` && Name ==
`<domain_name>.`].Id' 2
        --output text
```

1 2 **<domain_name>**은 OpenShift Container Platform 클러스터의 Route 53 기본 도메인을 지정합니다.

출력 예

```
/hostedzone/Z3URY6TWQ91KVV
```

도메인의 공개 호스팅 영역 ID가 명령 출력에 표시됩니다. 이 예에서는 **Z3URY6TWQ91KVV**입니다.

5. 프라이빗 영역에 별칭 레코드를 추가합니다.

```
$ aws route53 change-resource-record-sets --hosted-zone-id "<private_hosted_zone_id>" --
change-batch '{ 1
>   "Changes": [
>     {
>       "Action": "CREATE",
>       "ResourceRecordSet": {
>         "Name": "\\052.apps.<cluster_domain>", 2
>         "Type": "A",
>         "AliasTarget":{
>           "HostedZoneId": "<hosted_zone_id>", 3
>           "DNSName": "<external_ip>.", 4
>           "EvaluateTargetHealth": false
>         }
>       }
>     }
>   ]
> }'
```

1 **<private_hosted_zone_id>**는 DNS 및 로드 밸런싱에 대한 CloudFormation 템플릿 출력의 값을 지정합니다.

2 **<cluster_domain>**은 OpenShift Container Platform 클러스터와 함께 사용하는 도메인 또는 하위 도메인을 지정합니다.

3 **<hosted_zone_id>**는 가져온 로드 밸런서의 공개 호스팅 영역 ID를 지정합니다.

4 **<external_ip>**는 Ingress Operator 로드 밸런서의 외부 IP 주소값을 지정합니다. 이 매개변수 값에 후행 마침표(**.**)을 포함시켜야 합니다.

6. 퍼블릭 영역에 레코드를 추가합니다.

```
$ aws route53 change-resource-record-sets --hosted-zone-id "<public_hosted_zone_id>"" --
change-batch '{ 1
>   "Changes": [
```

```
>    {
>      "Action": "CREATE",
>      "ResourceRecordSet": {
>        "Name": "\\052.apps.<cluster_domain>", 2
>        "Type": "A",
>        "AliasTarget":{
>          "HostedZoneId": "<hosted_zone_id>", 3
>          "DNSName": "<external_ip>.", 4
>          "EvaluateTargetHealth": false
>        }
>      }
>    }
>  ]
>}'
```

1. **<public_hosted_zone_id>**는 도메인의 퍼블릭 호스팅 영역을 지정합니다.
2. **<cluster_domain>**은 OpenShift Container Platform 클러스터와 함께 사용하는 도메인 또는 하위 도메인을 지정합니다.
3. **<hosted_zone_id>**는 가져온 로드 밸런서의 공개 호스팅 영역 ID를 지정합니다.
4. **<external_ip>**는 Ingress Operator 로드 밸런서의 외부 IP 주소값을 지정합니다. 이 매개변수 값에 후행 마침표(**.**)을 포함시켜야 합니다.

4.13.23. 사용자 프로비저닝 인프라에서 AWS 설치 완료

AWS(Amazon Web Service) 사용자 프로비저닝 인프라에서 OpenShift Container Platform 설치를 시작한 후 배포가 완료될 때까지 모니터링합니다.

사전 요구 사항

- 사용자 프로비저닝 AWS 인프라에서 OpenShift Container Platform 클러스터에 대한 부트스트랩 노드를 제거하셨습니다.
- **oc** CLI를 설치했습니다.

프로세스

1. 설치 프로그램이 포함된 디렉터리에서 클러스터 설치를 완료합니다.

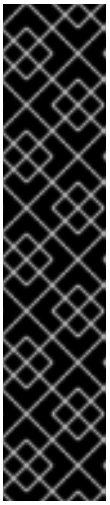
```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1. **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

출력 예

```
INFO Waiting up to 40m0s for the cluster at https://api.mycluster.example.com:6443 to
initialize...
INFO Waiting up to 10m0s for the openshift-console route to be created...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
```

```
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Fe5en-ymBEc-
Wt6NL"
INFO Time elapsed: 1s
```



중요

- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

2. 클러스터 등록 페이지에서 클러스터를 등록합니다.

4.13.24. 웹 콘솔을 사용하여 클러스터에 로그인

kubeadmin 사용자는 OpenShift Container Platform 설치 후 기본적으로 존재합니다. OpenShift Container Platform 웹 콘솔을 사용하여 **kubeadmin** 사용자로 클러스터에 로그인할 수 있습니다.

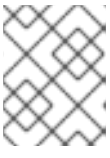
사전 요구 사항

- 설치 호스트에 대한 액세스 권한이 있어야 합니다.
- 클러스터 설치를 완료했으며 모든 클러스터 Operator를 사용할 수 있습니다.

프로세스

1. 설치 호스트의 **kubeadmin-password** 파일에서 **kubeadmin** 사용자의 암호를 가져옵니다.

```
$ cat <installation_directory>/auth/kubeadmin-password
```

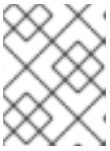


참고

대안으로 설치 호스트의 **<installation_directory>/.openshift_install.log** 로그 파일에서 **kubeadmin** 암호를 가져올 수 있습니다.

2. OpenShift Container Platform 웹 콘솔 경로를 나열합니다.

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```



참고

대안으로 설치 호스트의 **<installation_directory>/.openshift_install.log** 로그 파일에서 OpenShift Container Platform 경로를 가져올 수 있습니다.

출력 예

```
console     console-openshift-console.apps.<cluster_name>.<base_domain>          console
https   reencrypt/Redirect   None
```

3. 웹 브라우저의 이전 명령 출력에 자세히 설명된 경로로 이동하고 **kubeadmin** 사용자로 로그인합니다.

추가 리소스

- OpenShift Container Platform 웹 콘솔 액세스 및 이해에 대한 자세한 내용은 웹 콘솔에 액세스를 참조하십시오.

4.13.25. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

4.13.26. 추가 리소스

- AWS CloudFormation 스택에 대한 자세한 내용은 AWS 문서의 스택 작업을 참조하십시오.

4.13.27. 다음 단계

- 설치를 확인합니다.
- 클러스터를 사용자 지정합니다.
- Cluster Samples Operator 및 **must-gather** 툴의 이미지 스트림을 구성합니다.
- 제한된 네트워크에서 Operator Lifecycle Manager (OLM) 사용 방법에 대해 살펴봅니다.
- 클러스터를 설치하는 데 사용한 미러 레지스트리에 신뢰할 수 있는 CA가 있는 경우 추가 신뢰 저장소를 구성하여 클러스터에 추가합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.
- 필요한 경우 클라우드 공급자 인증 정보를 제거 할 수 있습니다.

4.14. AWS의 클러스터 설치 제거

AWS(Amazon Web Services)에 배포한 클러스터를 제거할 수 있습니다.

4.14.1. 설치 관리자가 프로비저닝한 인프라를 사용하는 클러스터 제거

클라우드에서 설치 관리자 프로비저닝 인프라를 사용하는 클러스터를 제거할 수 있습니다.

참고

설치 제거 후 특히 UPI(User Provisioned Infrastructure) 클러스터에서 제거되지 않은 리소스에 대해 클라우드 공급자를 확인합니다. 설치 관리자가 생성하지 않았거나 설치 프로그램이 액세스할 수 없는 리소스가 있을 수 있습니다.

사전 요구 사항

- 클러스터를 배포하는 데 사용한 설치 프로그램의 사본을 준비합니다.
- 클러스터를 만들 때 설치 프로그램에 의해 생성된 파일을 준비합니다.

프로세스

1. 클러스터를 설치하는 데 사용한 컴퓨터에 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행합니다.

```
$ ./openshift-install destroy cluster \
--dir <installation_directory> --log-level info 1 2
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2 다른 세부 사항을 보려면 **info** 대신 **warn**, **debug** 또는 **error**를 지정합니다.

참고

클러스터의 클러스터 정의 파일이 포함되어 있는 디렉터리를 지정해야 합니다. 설치 프로그램이 클러스터를 삭제하려면 이 디렉터리에 있는 **metadata.json** 파일이 필요합니다.

2. 선택사항: **<installation_directory>** 디렉터리와 OpenShift Container Platform 설치 프로그램을 삭제합니다.

4.14.2. Cloud Credential Operator 유틸리티를 사용하여 AWS 리소스 삭제

STS를 사용하여 수동 모드에서 Cloud Credential Operator (CCO)를 사용하여 OpenShift Container Platform 클러스터를 설치 제거한 후 리소스를 정리하려면 CCO 유틸리티(**ccoctl**)를 사용하여 설치 중에 **ccoctl**이 생성한 AWS 리소스를 제거할 수 있습니다.

사전 요구 사항

- **ccoctl** 바이너리를 추출하고 준비합니다.

- STS를 사용하여 수동 모드에서 CCO와 함께 OpenShift Container Platform 클러스터를 설치합니다.

프로세스

- **ccoctl**에서 생성한 AWS 리소스를 삭제합니다.

```
$ ccoctl aws delete --name=<name> --region=<aws_region>
```

다음과 같습니다.

  - ***<name>***은 원래 클라우드 리소스를 만들고 태그하는 데 사용되는 이름과 일치합니다.
  - ***<AWS-region>***은 클라우드 리소스가 삭제될 AWS 리전입니다.

출력 예:

```
2021/04/08 17:50:41 Identity Provider object .well-known/openid-configuration deleted from the bucket <name>-oidc
2021/04/08 17:50:42 Identity Provider object keys.json deleted from the bucket <name>-oidc
2021/04/08 17:50:43 Identity Provider bucket <name>-oidc deleted
2021/04/08 17:51:05 Policy <name>-openshift-cloud-credential-operator-cloud-credential-o associated with IAM Role <name>-openshift-cloud-credential-operator-cloud-credential-o deleted
2021/04/08 17:51:05 IAM Role <name>-openshift-cloud-credential-operator-cloud-credential-o deleted
2021/04/08 17:51:07 Policy <name>-openshift-cluster-csi-drivers-ebs-cloud-credentials associated with IAM Role <name>-openshift-cluster-csi-drivers-ebs-cloud-credentials deleted
2021/04/08 17:51:07 IAM Role <name>-openshift-cluster-csi-drivers-ebs-cloud-credentials deleted
2021/04/08 17:51:08 Policy <name>-openshift-image-registry-installer-cloud-credentials associated with IAM Role <name>-openshift-image-registry-installer-cloud-credentials deleted
2021/04/08 17:51:08 IAM Role <name>-openshift-image-registry-installer-cloud-credentials deleted
2021/04/08 17:51:09 Policy <name>-openshift-ingress-operator-cloud-credentials associated with IAM Role <name>-openshift-ingress-operator-cloud-credentials deleted
2021/04/08 17:51:10 IAM Role <name>-openshift-ingress-operator-cloud-credentials deleted
2021/04/08 17:51:11 Policy <name>-openshift-machine-api-aws-cloud-credentials associated with IAM Role <name>-openshift-machine-api-aws-cloud-credentials deleted
2021/04/08 17:51:11 IAM Role <name>-openshift-machine-api-aws-cloud-credentials deleted
2021/04/08 17:51:39 Identity Provider with ARN arn:aws:iam::<aws_account_id>:oidc-provider/<name>-oidc.s3.<aws_region>.amazonaws.com deleted
```

검증

AWS를 쿼리하여 리소스가 삭제되었는지 확인할 수 있습니다. 자세한 내용은 AWS 설명서를 참조하십시오.

5장. AZURE에 설치

5.1. AZURE에 설치 준비

5.1.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비 에 대한 문서를 읽습니다.

5.1.2. Azure에 OpenShift Container Platform을 설치하기 위한 요구사항

Microsoft Azure에 OpenShift Container Platform을 설치하려면 먼저 Azure 계정을 구성해야 합니다. 계정 구성, 계정 제한, 퍼블릭 DNS 영역 구성, 필수 역할, 서비스 주체 생성 및 지원되는 Azure 리전에 대한 자세한 내용은 Azure 계정 구성을 참조하십시오.

사용자의 환경에서 클라우드 ID 및 액세스 관리(IAM) API에 액세스할 수 없거나 **kube-system** 네임스페이스에 관리자 수준의 인증 정보 시크릿을 저장하지 않으려면 다른 옵션에 대해 Azure용 IAM 수동 생성을 참조하십시오.

5.1.3. Azure에 OpenShift Container Platform을 설치할 방법 선택

설치 관리자 프로비저닝 또는 사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치할 수 있습니다. 기본 설치 유형은 설치 프로그램이 클러스터의 기본 인프라를 프로비저닝하는 설치 관리자 프로비저닝 인프라를 사용합니다. 제공하는 인프라에 OpenShift Container Platform도 설치할 수 있습니다. 설치 프로그램에서 프로비저닝한 인프라를 사용하지 않는 경우 클러스터 리소스를 직접 관리하고 유지보수해야 합니다.

설치 관리자 프로비저닝 및 사용자 프로비저닝 설치 프로세스에 대한 자세한 내용은 설치 프로세스를 참조하십시오.

5.1.3.1. 설치 관리자 프로비저닝 인프라를 사용하여 클러스터 설치

다음 방법 중 하나를 사용하여 OpenShift Container Platform 설치 프로그램에서 프로비저닝한 Azure 인프라에 컨테이너를 설치할 수 있습니다.

- **Azure에서 클러스터 빠른 설치:** OpenShift Container Platform 설치 프로그램에서 프로비저닝한 Azure 인프라에 OpenShift Container Platform을 설치할 수 있습니다. 기본 구성 옵션을 사용하여 빠르게 클러스터를 설치할 수 있습니다.
- **Azure에 사용자 지정 클러스터 설치**: 설치 프로그램이 프로비저닝하는 Azure 인프라에 사용자 지정 클러스터를 설치할 수 있습니다. 설치 프로그램을 통해 설치 단계에서 일부 사용자 지정을 적용할 수 있습니다. 다른 많은 사용자 정의 옵션은 설치 후사용할 수 있습니다.
- **네트워크 사용자 지정으로 Azure에 클러스터 설치**: 설치 중에 OpenShift Container Platform 네트워크 구성을 사용자 지정할 수 있으므로 클러스터가 기존 IP 주소 할당과 공존하고 네트워크 요구사항을 준수할 수 있습니다.
- **기존 VNet에 Azure의 클러스터 설치**: Azure의 기존 VNet(Azure Virtual Network)에 OpenShift Container Platform을 설치할 수 있습니다. 새 계정 또는 인프라를 생성할 때 제한과 같이 회사의 지침에 따라 설정되는 제약 조건이 있는 경우 이 설치 방법을 사용할 수 있습니다.

- **Azure에 프라이빗 클러스터 설치**: 프라이빗 클러스터를 Azure의 기존 Azure Virtual Network(VNet)에 설치할 수 있습니다. 이 방법을 사용하여 인터넷에 표시되지 않는 내부 네트워크에 OpenShift Container Platform을 배포할 수 있습니다.

- **Azure에 있는 클러스터를 정부 리전에 설치**: OpenShift Container Platform은 연방, 주, 지역 수준의 정부 기관은 물론 Azure에서 중요한 워크로드를 실행해야 하는 미국 정부 기관, 계약자, 교육 기관 및 기타 미국 고객을 위해 설계된 Microsoft Azure Government (MAG) 리전에 배포할 수 있습니다.

5.1.3.2. 사용자 프로비저닝 인프라를 사용하여 클러스터 설치

다음 방법을 사용하여 프로비저닝한 Azure 인프라에 클러스터를 설치할 수 있습니다.

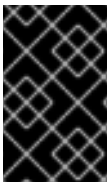
- **ARM 템플릿을 사용하여 Azure에 클러스터 설치**: 사용자가 제공하는 인프라를 사용하여 Azure에 OpenShift Container Platform을 설치할 수 있습니다. 제공된 ARM(Azure Resource Manager) 템플릿을 사용하여 설치를 지원할 수 있습니다.

5.1.4. 다음 단계

- Azure 계정 구성

5.2. AZURE 계정 구성

OpenShift Container Platform을 설치하려면 먼저 Microsoft Azure 계정을 구성해야 합니다.



중요

공용 끝점을 통해 사용할 수 있는 모든 Azure 리소스에는 리소스 이름 제한 사항이 적용되며, 특정 용어를 사용하는 리소스를 생성할 수 없습니다. Azure가 제한하는 용어 목록은 Azure 문서의 예약된 리소스 이름 오류 해결 을 참조하십시오.

5.2.1. Azure 계정 제한

OpenShift Container Platform 클러스터는 수많은 Microsoft Azure 구성 요소를 사용하며, 기본 Azure 서브크립션 및 서비스 제한, 할당량 및 제약 조건은 OpenShift Container Platform 클러스터 설치에 영향을 미칩니다.

중요

기본 제한값은 무료 평가판 및 종량 과금제와 같은 상품 카테고리 유형과 Dv2, F, G 등 시리즈에 따라 다릅니다. 예를 들어 기업 계약 서브스크립션의 기본값은 350개 코어입니다.

서브스크립션 유형에 대한 제한값을 확인하고 필요한 경우 Azure에 기본 클러스터를 설치하기 전에 계정에 대한 할당량 제한값을 늘리십시오.

다음 표에 OpenShift Container Platform 클러스터를 설치하고 실행하는 데 영향을 미칠 수 있는 Azure 구성 요소 제한값이 요약되어 있습니다.

구성 요소	기본적으로 필요한 구성 요소 수	기본 Azure 제한값	설명

구성 요소	기본적으로 필요한 구성 요소 수	기본 Azure 제한값	설명
vCPU	40	리전당 20개	기본 클러스터의 경우 40개의 vCPU가 필요하므로 계정 제한을 늘려야 합니다. 기본적으로 각 클러스터는 다음 인스턴스를 생성합니다. ● 설치 후 제거되는 하나의 부트스트랩 시스템 ● 컨트롤 플레인 시스템 세 개 ● 컴퓨팅 시스템 세 개 부트스트랩 시스템은 4개의 vCPU를 사용하는 **Standard_D4s_v3** 시스템을 사용하고, 컨트롤 플레인 시스템은 8개의 vCPU를 사용하는 **Standard_D8s_v3** 가상 머신을 사용하고, 작업자 시스템은 4개의 vCPU를 사용하는 **Standard_D4s_v3** 가상 머신을 사용하므로, 기본 클러스터에는 40개의 vCPU가 필요합니다. 4개의 vCPU를 사용하는 부트스트랩 노드 VM은 설치 중에만 사용됩니다. 더 많은 작업자 노드를 배포하거나, 자동 크기 조정을 활성화하거나, 대규모 워크로드를 배포하거나, 다른 인스턴스 유형을 사용하려면 필요한 시스템을 클러스터가 배포할 수 있도록 계정의 vCPU 제한값을 더 늘려야 합니다. 기본적으로 설치 프로그램은 컨트롤 플레인과 컴퓨팅 시스템을 한 리전 내의 모든 가용성 영역에 분배합니다. 클러스터의 고가용성을 보장하기 위해서는 가용성 영역이 세 개 이상인 리전을 선택하십시오. 각 리전에 가용성 영역이 세 개 미만인 경우에는 설치 프로그램이 사용 가능한 영역에 둘 이상의 컨트롤 플레인 시스템을 배치합니다.
OS 디스크	7		VM OS 디스크는 컨트롤 플레인 시스템에 대해 테스트되고 권장되는 최소 처리량 5000 IOPS/200MBps를 유지할 수 있어야 합니다. 이 처리량은 최소 1 TiB Premium SSD (P30)를 보유하여 제공할 수 있습니다. Azure에서는 디스크 성능은 SSD 디스크 크기에 직접적으로 의존하므로 **Standard_D8s_v3** 또는 기타 유사한 시스템 유형에서 지원되는 처리량과 5000 IOPS 목표를 달성하려면 최소 P30 디스크가 필요합니다. 읽기 대기 시간이 짧고 읽기 IOPS 및 처리량을 높이려면 호스트 캐싱을 **ReadOnly**로 설정해야 합니다. VM 메모리 또는 로컬 SSD 디스크에 있는 캐시에서 수행된 읽기는 Blob 스토리지에 있는 데이터 디스크에서 읽기보다 훨씬 빠릅니다.
VNet	1	리전당 1000개	각 기본 클러스터에는 두 개의 서브넷이 포함된 하나의 가상 네트워크(VNet)가 필요합니다.

구성 요소	기본적으로 필요한 구성 요소 수	기본 Azure 제한 값	설명
네트워크 인터페이스	7	리전당 65,536개	각 기본 클러스터에는 7개의 네트워크 인터페이스가 필요합니다. 더 많은 시스템을 생성하거나 배포된 워크로드가 로드 밸런서를 생성하는 경우 클러스터는 더 많은 네트워크 인터페이스를 사용합니다.
네트워크 보안 그룹	2	5000	각 클러스터는 VNet의 각 서브넷에 대한 네트워크 보안 그룹을 생성합니다. 기본 클러스터는 컨트롤 플레인과 컴퓨팅 노드 서브넷에 대한 네트워크 보안 그룹을 생성합니다: **controlplane**: 어디에서나 포트 6443에서 컨트롤 플레인 시스템에 도달할 수 있습니다. **node**: 포트 80 및 443을 통해 인터넷에서 작업자 노드에 도달할 수 있습니다.
네트워크 로드 밸런서	3	리전당 1000개	각 클러스터는 다음 로드 밸런서를 생성합니다. **default**: 작업자 시스템에서 포트 80과 443에 대한 요청을 로드 밸런싱하는 공용 IP 주소 **internal**: 컨트롤 플레인 시스템에서 포트 6443과 22623에 대한 요청을 로드 밸런싱하는 개인 IP 주소 **external**: 컨트롤 플레인 시스템에서 포트 6443에 대한 요청을 로드 밸런싱하는 공용 IP 주소 애플리케이션이 더 많은 Kubernetes **LoadBalancer** Service 개체를 생성하면 클러스터가 더 많은 로드 밸런서를 사용합니다.
공용 IP 주소	3		두 개의 공용 로드 밸런서 각각이 공용 IP 주소를 사용합니다. 또한 부트스트랩 시스템은 공용 IP 주소를 사용하므로 설치 중 문제를 해결하기 위해 시스템으로 SSH를 실행할 수 있습니다. 부트스트랩 노드의 IP 주소는 설치 중에만 사용됩니다.
개인 IP 주소	7		내부 로드 밸런서, 3개의 컨트롤 플레인 시스템 및 3개의 각 작업자 시스템은 각각 개인 IP 주소를 사용합니다.

구성 요소	기본적으로 필요한 구성 요소 수	기본 Azure 제한 값	설명
spot VM vCPU (선택 사항)	0 스팟 VM을 구성하는 경우 클러스터에 모든 컴퓨팅 노드에 대해 두 개의 스팟 VM vCPU가 있어야 합니다.	리전당 20개	선택적 구성 요소입니다. Spot 가상 머신을 사용하려면 Azure 기본 제한을 클러스터의 컴퓨팅 노드 수보다 두 배 이상 늘려야 합니다. **참고** 컨트롤 플레인 노드에 스팟 VM을 사용하는 것은 권장되지 않습니다.

5.2.2. Azure에서 퍼블릭 DNS 영역 구성

OpenShift Container Platform을 설치하려면 사용하는 Microsoft Azure 계정에 전용의 퍼블릭 호스팅 DNS 영역이 있어야 합니다. 도메인에 대한 권한도 이 영역에 있어야 합니다. 이 서비스는 클러스터와 외부 연결에 필요한 클러스터 DNS 확인 및 이름 조회 기능을 제공합니다.

프로세스

1. 도메인 또는 하위 도메인과 등록 기관을 식별합니다. 기존 도메인 및 등록 기관을 이전하거나 Azure 또는 다른 소스를 통해 새 도메인과 등록 기관을 받을 수 있습니다.

   참고

   Azure를 통한 도메인 구매에 대한 자세한 내용은 Azure 문서에서 Azure App Service의 사용자 지정 도메인 이름 구매를 참조하십시오.

2. 기존 도메인과 등록 기관을 사용하는 경우 해당 DNS를 Azure로 마이그레이션합니다. Azure 문서의 활성 DNS 이름을 Azure App Service로 마이그레이션 을 참조하십시오.

3. 도메인에 맞게 DNS를 구성합니다. Azure 문서의 자습서: Azure DNS에서 도메인 호스팅 에 나온 단계에 따라 도메인 또는 하위 도메인에 대한 퍼블릭 호스팅 영역을 생성하고, 권한 있는 새 이름 서버를 추출하고, 도메인이 사용하는 이름 서버의 등록 기관 레코드를 업데이트합니다.
   적절한 루트 도메인(예: **openshiftcorp.com**) 또는 하위 도메인(예: **clusters.openshiftcorp.com**)을 사용합니다.

4. 하위 도메인을 사용하는 경우, 회사의 프로시저에 따라 상위 도메인에 위임 레코드를 추가합니다.

5.2.3. Azure 계정 제한 늘리기

계정 제한을 늘리려면 Azure 포털에서 지원 요청을 제출하십시오.

참고

지원 요청당 한 가지 유형의 할당량만 늘릴 수 있습니다.

프로세스

1. Azure 포털의 왼쪽 하단 모서리에서 **Help + support**를 클릭합니다.

2. **New support request**를 클릭한 후 필요한 값을 선택합니다.
   a. **Issue type** 목록에서 **Service and subscription limits (quotas)**을 선택합니다.
   b. **Subscription** 목록에서 수정할 서브스크립션을 선택합니다.
   c. **Quota type** 목록에서 증가시킬 할당량을 선택합니다. 예를 들어 클러스터를 설치하는 데 필요해서 vCPU 수를 늘리려면 **Compute-VM (cores-vCPUs) subscription limit increases**를 선택합니다.
   d. **Next: Solutions**를 클릭합니다.
3. **Problem Details** 페이지에서 할당량 증가에 필요한 정보를 제공합니다.
   a. **Provide details**를 클릭하고 **Quota details** 창에서 필요한 세부 사항을 제공합니다.
   b. SUPPORT METHOD 및 CONTACT INFO 섹션에서 문제 심각도와 연락처 정보를 제공합니다.
4. **Next: Review + create**을 클릭한 후 **Create**을 클릭합니다.

5.2.4. 필수 Azure 역할

OpenShift Container Platform에는 Microsoft Azure 리소스를 관리할 수 있도록 서비스 주체가 필요합니다. 서비스 주체를 생성하려면 Azure 계정 서브스크립션에 다음 역할이 있어야 합니다.

- **User Access Administrator**
- **기여자**

Azure 포털에서 역할을 설정하려면 Azure 문서의 RBAC 및 Azure 포털을 사용하여 Azure 리소스에 대한 액세스 관리를 참조하십시오.

5.2.5. 서비스 주체 생성

OpenShift Container Platform 및 해당 설치 프로그램은 Azure Resource Manager를 사용하여 Microsoft Azure 리소스를 생성하므로 이를 나타내는 서비스 주체를 생성해야 합니다.

사전 요구 사항

- Azure CLI를 설치 또는 업데이트합니다.
- Azure 계정은 사용하는 서브스크립션에 대한 필요한 역할을 갖습니다.

프로세스

1. Azure CLI에 로그인합니다.

```
$ az login
```

2. Azure 계정이 서브스크립션을 사용하는 경우 올바른 서브스크립션을 사용하고 있는지 확인합니다.
   a. 사용 가능한 계정 목록을 보고 클러스터에 사용하려는 서브스크립션의 **tenantId** 값을 기록합니다.

```
$ az account list --refresh
```

출력 예

```
[
  {
    "cloudName": "AzureCloud",
    "id": "9bab1460-96d5-40b3-a78e-17b15e978a80",
    "isDefault": true,
    "name": "Subscription Name",
    "state": "Enabled",
    "tenantId": "6057c7e9-b3ae-489d-a54e-de3f6bf6a8ee",
    "user": {
      "name": "you@example.com",
      "type": "user"
    }
  }
]
```

b. 활성 계정 세부 사항을 보고 **tenantId** 값이 사용하려는 서브스크립션과 일치하는지 확인합니다.

```
$ az account show
```

출력 예

```
{
  "environmentName": "AzureCloud",
  "id": "9bab1460-96d5-40b3-a78e-17b15e978a80",
  "isDefault": true,
  "name": "Subscription Name",
  "state": "Enabled",
  "tenantId": "6057c7e9-b3ae-489d-a54e-de3f6bf6a8ee", (1)
  "user": {
    "name": "you@example.com",
    "type": "user"
  }
}
```

(1) **tenantId** 매개변수 값이 올바른 서브스크립션 ID인지 확인합니다.

c. 올바른 서브스크립션을 사용하지 않는 경우, 활성 서브스크립션을 변경합니다.

```
$ az account set -s <subscription_id> (1)
```

(1) 서브스크립션 ID를 지정합니다.

d. 서브스크립션 ID 업데이트를 확인합니다.

```
$ az account show
```

출력 예

```
{
  "environmentName": "AzureCloud",
  "id": "33212d16-bdf6-45cb-b038-f6565b61edda",
  "isDefault": true,
  "name": "Subscription Name",
  "state": "Enabled",
  "tenantId": "8049c7e9-c3de-762d-a54e-dc3f6be6a7ee",
  "user": {
    "name": "you@example.com",
    "type": "user"
  }
}
```

3. 출력의 **tenantId** 및 **id** 매개변수 값을 기록합니다. OpenShift Container Platform 설치 중에 이러한 값이 필요합니다.

4. 계정에 대한 서비스 주체를 생성합니다.

```
$ az ad sp create-for-rbac --role Contributor --name <service_principal> \ 1
  --scopes /subscriptions/<subscription_id> 2
  --years <years> 3
```

1 서비스 주체 이름을 지정합니다.

2 서브스크립션 ID를 지정합니다.

3 년 수를 지정합니다. 기본적으로 서비스 주체는 1년 후에 만료됩니다. **--years** 옵션을 사용하면 서비스 주체의 유효성을 확장할 수 있습니다.

출력 예

```
Creating 'Contributor' role assignment under scope '/subscriptions/<subscription_id>'
The output includes credentials that you must protect. Be sure that you do not
include these credentials in your code or check the credentials into your source
control. For more information, see https://aka.ms/azadsp-cli
{
  "appId": "ac461d78-bf4b-4387-ad16-7e32e328aec6",
  "displayName": <service_principal>",
  "password": "00000000-0000-0000-0000-000000000000",
  "tenantId": "8049c7e9-c3de-762d-a54e-dc3f6be6a7ee"
}
```

5. 이전 출력의 **appId** 및 **password** 매개변수 값을 기록합니다. OpenShift Container Platform 설치 중에 이러한 값이 필요합니다.

6. 다음 명령을 실행하여 **User Access Administrator** 역할을 할당합니다.

```
$ az role assignment create --role "User Access Administrator" \
  --assignee-object-id $(az ad sp show --id <appId> --query id -o tsv) 1
```

1 서비스 주체의 **appId** 매개변수 값을 지정합니다.

추가 리소스

- CCO 모드에 대한 자세한 내용은 Cloud Credential Operator 정보를 참조하십시오.

5.2.6. 지원되는 Azure Marketplace 리전

Azure Marketplace 이미지를 사용하여 클러스터를 설치하면 북미 및 EMEA에서 제안을 구매하는 고객이 사용할 수 있습니다.

이 제안은 북미 또는 EMEA에서 구입해야 하지만 OpenShift Container Platform에서 지원하는 Azure 공용 파티션에 클러스터를 배포할 수 있습니다.

참고

Azure Marketplace 이미지를 사용하여 클러스터를 배포하는 것은 Azure Government 리전에서 지원되지 않습니다.

5.2.7. 지원되는 Azure 리전

설치 프로그램은 서브스크립션을 기반으로 사용 가능한 Microsoft Azure 리전 목록을 동적으로 생성합니다.

지원되는 Azure 리전

- **australiacentral** (호주 중부)
- **australiaeast** (호주 동부)
- **australiasoutheast** (호주 남동부)
- **brazilsouth** (브라질 남부)
- **canadacentral** (캐나다 중부)
- **canadaeast** (캐나다 동부)
- **centralindia** (인도 중부)
- **centralus** (미국 중부)
- **eastasia** (동아시아)
- **eastus** (미국 동부)
- **eastus2** (미국 동부 2)
- **francecentral** (프랑스 중부)
- **germanywestcentral** (독일 중서부)
- **japaneast** (일본 동부)
- **japanwest** (일본 서부)
- **koreacentral** (한국 중부)
- **koreasouth** (한국 남부)

- **northcentralus** (미국 중북부)
- **northeurope** (북유럽)
- **norwayeast** (노르웨이 동부)
- **CloudEventtarcentral** (Qatar Central)
- **southafricanorth** (남아프리카 북부)
- **southcentralus** (미국 중남부)
- **southeastasia** (동남아시아)
- **southindia** (인도 남부)
- **switzerlandnorth** (스위스 북부)
- **uaenorth** (UAE 북부)
- **uksouth** (영국 남부)
- **ukwest** (영국 서부)
- **westcentralus** (미국 중서부)
- **westeurope** (서유럽)
- **westindia** (인도 서부)
- **westus** (미국 서부)
- **westus2** (미국 서부 2)

지원되는 Azure Government 리전
OpenShift Container Platform 버전 4.6에는 다음과 같은 MAG (Microsoft Azure Government) 리전에 대한 지원이 추가되어 있습니다.

- **usgovtexas** (US Gov Texas)
- **usgovvirginia** (US Gov Virginia)

사용 가능한 모든 MAG 리전은 Azure 설명서에서 확인할 수 있습니다. 제공되는 다른 MAG 리전은 OpenShift Container Platform에서 작동할 것으로 예상되지만 아직 테스트되지 않았습니다.

5.2.8. 다음 단계

- OpenShift Container Platform 클러스터를 Azure에 설치합니다. 사용자 지정 클러스터를 설치 하거나 기본 옵션을 적용하여 클러스터를 빠르게 설치 할 수 있습니다.

5.3. 수동으로 AZURE 용 IAM 생성

클라우드 ID 및 액세스 관리(IAM) API에 연결할 수 없는 환경에서 또는 관리자가 클러스터 **kube-system** 네임 스페이스에 관리자 수준의 인증 정보 시크릿을 저장하지 않는 것을 선호하는 경우 클러스터를 설치하기 전에 CCO(Cloud Credential Operator)를 수동 모드로 설정할 수 있습니다.

5.3.1. kube-system 프로젝트에 관리자 수준 시크릿을 저장하는 대안

CCO(Cloud Credential Operator)는 클라우드 공급자 인증 정보를 Kubernetes CRD(사용자 지정 리소스 정의)로 관리합니다. **install-config.yaml** 파일에서 **credentialsMode** 매개변수의 다른 값을 설정하여 조직의 보안 요구 사항에 맞게 CCO를 구성할 수 있습니다.

클러스터 **kube-system** 프로젝트에 관리자 수준 인증 정보 시크릿을 저장하지 않으려면 OpenShift Container Platform을 설치할 때 CCO의 **credentialsMode** 매개변수를 **수동**으로 관리할 수 있습니다.

수동 모드를 사용하면 각 클러스터 구성 요소에는 클러스터에 관리자 수준 인증 정보를 저장하지 않고 필요한 권한만 보유할 수 있습니다. 환경이 클라우드 공급자 공용 IAM 끝점에 연결되지 않은 경우 이 모드를 사용할 수도 있습니다. 그러나 업그레이드할 때마다 새 릴리스 이미지로 권한을 수동으로 조정해야 합니다. 또한 요청하는 모든 구성 요소에 대한 인증 정보를 수동으로 제공해야 합니다.

추가 리소스

- 사용 가능한 모든 CCO 인증 정보 모드 및 지원되는 플랫폼에 대한 자세한 내용은 Cloud Credential Operator 정보를 참조하십시오.

5.3.2. 수동으로 IAM 생성

Cloud Credential Operator (CCO)는 클라우드 아이덴티티 및 액세스 관리 (IAM) API에 연결할 수 없는 환경에서 설치하기 전에 수동 모드로 전환할 수 있습니다. 또는 관리자가 클러스터 **kube-system** 네임 스페이스에 관리자 수준의 인증 정보 시크릿을 저장하지 않도록 합니다.

프로세스

1. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행하여 **install-config.yaml** 파일을 생성합니다.

```
$ openshift-install create install-config --dir <installation_directory>
```

   여기서 **<installation_directory>** 는 설치 프로그램이 파일을 생성하는 디렉터리입니다.

2. **install-config.yaml** 구성 파일을 편집하여 **credentialsMode** 매개 변수가 **Manual**로 설정되도록 합니다.

   install-config.yaml 설정 파일 예

```
apiVersion: v1
baseDomain: cluster1.example.com
credentialsMode: Manual (1)
compute:
- architecture: amd64
  hyperthreading: Enabled
...
```

   (1) 이 행은 **credentialsMode** 매개변수를 **Manual**로 설정하기 위해 추가됩니다.

3. 매니페스트를 생성하려면 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행합니다.

```
$ openshift-install create manifests --dir <installation_directory>
```

여기서 **<installation_directory>** 는 설치 프로그램이 파일을 생성하는 디렉터리입니다.

4. 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행하여 **openshift-install** 바이너리가 사용하도록 빌드된 OpenShift Container Platform 릴리스 이미지에 대한 세부 정보를 가져옵니다.

```
$ openshift-install version
```

출력 예

```
release image quay.io/openshift-release-dev/ocp-release:4.y.z-x86_64
```

5. 다음 명령을 실행하여 배포 중인 클라우드를 대상으로 하는 이 릴리스 이미지에서 모든 **CredentialsRequest** 오브젝트를 찾습니다.

```
$ oc adm release extract quay.io/openshift-release-dev/ocp-release:4.y.z-x86_64 \
  --credentials-requests \
  --cloud=azure
```

이 명령을 수행하면 각 **CredentialsRequest** 오브젝트에 대해 YAML 파일이 생성됩니다.

샘플 CredentialsRequest 개체

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component-credentials-request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AzureProviderSpec
    roleBindings:
    - role: Contributor
  ...
```

6. 이전에 생성한 **openshift-install** 매니페스트 디렉터리에 시크릿 YAML 파일을 만듭니다. 시크릿은 각 **CredentialsRequest** 오브젝트의 **spec.secretRef**에 정의된 네임 스페이스 및 시크릿 이름을 사용하여 저장해야 합니다.

보안이 포함된 샘플 CredentialsRequest 오브젝트

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component-credentials-request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AzureProviderSpec
    roleBindings:
    - role: Contributor
```

```
    ...
  secretRef:
    name: <component-secret>
    namespace: <component-namespace>
  ...
```

샘플 Secret 오브젝트

```
apiVersion: v1
kind: Secret
metadata:
  name: <component-secret>
  namespace: <component-namespace>
data:
  azure_subscription_id: <base64_encoded_azure_subscription_id>
  azure_client_id: <base64_encoded_azure_client_id>
  azure_client_secret: <base64_encoded_azure_client_secret>
  azure_tenant_id: <base64_encoded_azure_tenant_id>
  azure_resource_prefix: <base64_encoded_azure_resource_prefix>
  azure_resourcegroup: <base64_encoded_azure_resourcegroup>
  azure_region: <base64_encoded_azure_region>
```

중요

수동으로 유지 관리되는 인증 정보를 사용하는 클러스터를 업그레이드하기 전에 CCO가 업그레이드 가능한 상태인지 확인해야 합니다. 자세한 내용은 클라우드 공급자에 대한 설치 콘텐츠의 "수동으로 유지 관리되는 인증 정보를 사용하여 클러스터 업그레이드" 섹션을 참조하십시오.

5.3.3. 수동으로 유지 관리되는 인증 정보로 클러스터 업그레이드

CCO(Cloud Credential Operator) 수동으로 유지 관리되는 인증 정보가 있는 클러스터의 **Upgradable** 상태는 기본적으로 **False** 입니다.

- 마이너 릴리스(예: 4.8에서 4.9로)의 경우 이 상태는 업데이트된 권한을 처리하고 **CloudCredential** 리소스에 주석을 달아 권한이 다음 버전에 필요에 따라 업데이트되었음을 나타낼 때까지 업그레이드되지 않도록 합니다. 이 주석은 **Upgradable** 상태를 **True**로 변경합니다.
- 예를 들어 4.9.0에서 4.9.1으로 z-stream 릴리스의 경우 권한이 추가되거나 변경되지 않으므로 업그레이드가 차단되지 않습니다.

수동으로 유지 관리되는 인증 정보로 클러스터를 업그레이드하기 전에 업그레이드할 릴리스 이미지에 대한 새 인증 정보를 생성해야 합니다. 또한 기존 인증 정보에 필요한 권한을 검토하고 해당 구성 요소에 대한 새 릴리스에 새 권한 요구 사항을 수용해야 합니다.

절차

1. 새 릴리스에 대한 **CredentialsRequest** 사용자 지정 리소스를 추출하고 검사합니다.
   클라우드 공급자용 설치 콘텐츠의 "수동으로 IAM 생성" 섹션에서는 클라우드에 필요한 인증 정보를 획득하고 사용하는 방법을 설명합니다.
2. 클러스터에서 수동으로 유지 관리되는 인증 정보를 업데이트합니다.
   - 새 릴리스 이미지에서 추가한 **CredentialsRequest** 사용자 정의 리소스에 대한 새 시크릿을 생성합니다.

- 시크릿에 저장된 기존 인증 정보에 대한 **CredentialsRequest** 사용자 정의 리소스가 권한 요구 사항이 변경된 경우 필요에 따라 권한을 업데이트합니다.

3. 모든 보안이 새 릴리스에 대해 올바른 경우 클러스터를 업그레이드할 준비가 되었음을 나타냅니다.

   a. **cluster-admin** 역할의 사용자로 OpenShift Container Platform CLI에 로그인합니다.

   b. **CloudCredential** 리소스를 편집하여 **metadata** 필드 내에 **upgradeable-to** 주석을 추가합니다.

   ```
   $ oc edit cloudcredential cluster
   ```

   추가할 텍스트

   ```
   ...
     metadata:
       annotations:
         cloudcredential.openshift.io/upgradeable-to: <version_number>
   ...
   ```

   여기서 **<version_number>**는 **x.y.z** 형식으로 업그레이드할 버전입니다. 예를 들어 OpenShift Container Platform 4.8.2 의 경우 **4.8.2**입니다.

   주석을 추가한 후 업그레이드 가능 상태가 변경되는 데 몇 분이 소요될 수 있습니다.

4. CCO를 업그레이드할 수 있는지 확인합니다.

   a. 웹 콘솔의 **관리자** 화면에서 **관리자 → 클러스터 설정**으로 이동합니다.

   b. CCO 상태 세부 정보를 보려면 **Cluster Operators** 목록에서 **cloud-credential**을 클릭합니다.

   c. **Conditions** 섹션의 **Upgradeable** 상태가 **False**인 경우 **upgradeable-to 주석**에 오타 오류가 없는지 확인합니다.

Conditions 섹션의 **Upgradeable** 상태가 **True** 이면 OpenShift Container Platform 업그레이드를 시작할 수 있습니다.

5.3.4. 다음 단계

- OpenShift Container Platform 클러스터 설치:
  - 설치 관리자가 프로비저닝한 인프라에 기본 옵션으로 Azure에 빠르게 클러스터 설치
  - 설치 관리자가 프로비저닝한 인프라에 클라우드 사용자 지정으로 클러스터 설치
  - 설치 관리자가 프로비저닝한 인프라에 네트워크 사용자 지정으로 클러스터 설치

5.4. AZURE에 빠르게 클러스터 설치

OpenShift Container Platform 버전 4.9에서는 Microsoft Azure에 기본 구성 옵션을 사용하는 클러스터를 설치할 수 있습니다.

5.4.1. 사전 요구 사항

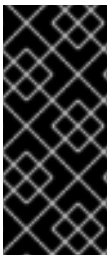
- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비 에 대한 문서를 읽습니다.
- 클러스터를 호스팅하고, 클러스터를 배포할 테스트 및 검증된 리전을 결정하도록 Azure 계정을 구성합니다.
- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성 해야 합니다.
- 사용자의 환경에서 클라우드 ID 및 액세스 관리(IAM) API에 액세스할 수 없거나 **kube-system** 네임스페이스에 관리자 수준의 인증 정보 시크릿을 저장하지 않으려면 IAM 인증 정보 수동 생성 및 유지 관리를 참조하십시오.

5.4.2. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.



중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

5.4.3. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

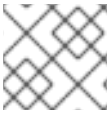
OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 **core** 사용자의 **~/.ssh/authorized_keys** 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 **core**로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. **./openshift-install gather** 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.



참고

AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: **~/.ssh/id_ed25519**)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 **'~/.ssh** 디렉터리에 있는지 확인하십시오.

참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 **ed25519** 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 **rsa** 또는 **ecdsa** 알고리즘을 사용하는 키를 생성합니다.

2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 **~/.ssh/id_ed25519.pub** 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 **./openshift-install gather** 명령을 사용하려는 경우 필요합니다.

참고

일부 배포에서는 **~/.ssh/id_rsa** 및 **~/.ssh/id_dsa**와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. **ssh-agent** 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. **ssh-agent**에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: **~/.ssh/id_ed25519**).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

5.4.4. 설치 프로그램 받기

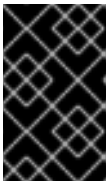
OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

프로세스

1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.
2. 인프라 공급자를 선택합니다.
3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.



중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.

중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 OpenShift Container Platform 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

5.4.5. 클러스터 배포

호환되는 클라우드 플랫폼에 OpenShift Container Platform을 설치할 수 있습니다.

중요

최초 설치 과정에서 설치 프로그램의 **create cluster** 명령을 한 번만 실행할 수 있습니다.

사전 요구 사항

- 클러스터를 호스팅하는 클라우드 플랫폼으로 계정을 구성합니다.
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

프로세스

1. 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터 배포를 초기화합니다.

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>**는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.

2 다른 설치 세부 사항을 보려면 **info** 대신 **warn**, **debug** 또는 **error**를 지정합니다.

중요

비어 있는 디렉터리를 지정합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

화면에 나타나는 지시에 따라 필요한 값을 입력합니다.

a. 선택사항: 클러스터 시스템에 액세스하는 데 사용할 SSH 키를 선택합니다.

참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

b. 대상 플랫폼으로 **azure**를 선택합니다.

c. 컴퓨터에 Microsoft Azure 프로필이 저장되어 있지 않은 경우 서비스크립션 및 서비스 주체에 대해 다음 Azure 매개변수 값을 지정합니다.

- **azure subscription id** 클러스터에 사용할 서브스크립션 ID입니다. 계정 출력의 **id** 값을 지정하십시오.

- **azure tenant id** 테넌트 ID. 계정 출력의 **tenantId** 값을 지정하십시오.
- **azure service principal client id** 서비스 주체의 **appId** 매개변수 값.
- **azure service principal client secret** 서비스 주체의 **password** 매개변수 값.

d. 클러스터를 배포할 리전을 선택합니다.

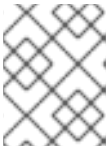
e. 클러스터를 배포할 기본 도메인을 선택합니다. 기본 도메인은 클러스터용으로 생성한 Azure DNS 영역에 해당합니다.

f. 클러스터를 설명할 수 있는 이름을 입력합니다.

중요

공용 끝점을 통해 사용할 수 있는 모든 Azure 리소스에는 리소스 이름 제한 사항이 적용되며, 특정 용어를 사용하는 리소스를 생성할 수 없습니다. Azure가 제한하는 용어 목록은 Azure 문서의 예약된 리소스 이름 오류 해결 을 참조하십시오.

g. Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 붙여넣습니다.



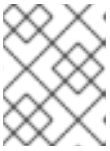
참고

호스트에 구성된 클라우드 공급자 계정에 클러스터를 배포하기에 충분한 권한이 없는 경우, 설치 프로세스가 중단되고 누락된 권한을 알리는 메시지가 표시됩니다.

클러스터 배포가 완료되면 웹 콘솔로 연결되는 링크와 **kubeadmin** 사용자의 인증 정보가 포함된 클러스터 액세스 지침이 사용자 터미널에 표시됩니다.

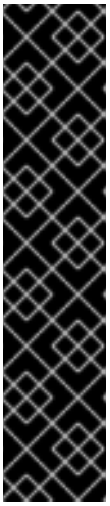
출력 예

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```



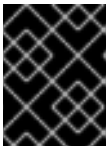
참고

설치에 성공하면 클러스터 액세스 및 인증 정보도 **<installation_directory>/.openshift_install.log**로 출력됩니다.



중요

- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

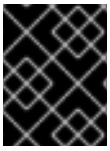


중요

설치 프로그램에서 생성되는 파일이나 설치 프로그램을 삭제해서는 안 됩니다. 클러스터를 삭제하려면 두 가지가 모두 필요합니다.

5.4.6. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift CLI(**oc**)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 **oc**를 설치할 수 있습니다.



중요

이전 버전의 **oc**를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 **oc**를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 Linux에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Linux Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. **oc** 바이너리를 **PATH**에 있는 디렉터리에 배치합니다.
PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치
다음 절차에 따라 Windows에 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Windows Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. ZIP 프로그램으로 아카이브의 압축을 풉니다.
5. **oc** 바이너리를 **PATH**에 있는 디렉터리로 이동합니다.
   PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치
다음 절차에 따라 macOS에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 MacOSX Client** 항목 옆에 있는 **지금 다운로드를** 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 해제하고 압축을 풉니다.
5. **oc** 바이너리 PATH의 디렉터리로 이동합니다.
   PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

5.4.7. CLI를 사용하여 클러스터에 로그인

클러스터 **kubeconfig** 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다.
kubeconfig 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- **oc** CLI를 설치했습니다.

프로세스

1. **kubeadmin** 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 **oc** 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

추가 리소스

- OpenShift Container Platform 웹 콘솔 액세스 및 이해에 대한 자세한 내용은 웹 콘솔에 액세스를 참조하십시오.

5.4.8. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

5.4.9. 다음 단계

- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.

5.5. 사용자 지정 설정을 사용하여 AZURE에 클러스터 설치

OpenShift Container Platform 버전 4.9에서는 설치 프로그램이 Microsoft Azure에 프로비저닝하는 인프라에 사용자 지정 클러스터를 설치할 수 있습니다. 설치를 사용자 지정하려면 클러스터를 설치하기 전에 **install-config.yaml** 파일에서 매개변수를 수정합니다.

5.5.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비 에 대한 문서를 읽습니다.
- 클러스터를 호스팅하고, 클러스터를 배포할 테스트 및 검증된 리전을 결정하도록 Azure 계정을 구성합니다.
- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성 해야 합니다.
- 사용자의 환경에서 클라우드 ID 및 액세스 관리(IAM) API에 액세스할 수 없거나 **kube-system** 네임스페이스에 관리자 수준의 인증 정보 시크릿을 저장하지 않으려면 IAM 인증 정보 수동 생성 및 유지 관리를 참조하십시오.

5.5.2. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.

중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

5.5.3. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 **core** 사용자의 **~/.ssh/authorized_keys** 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 **core**로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. **./openshift-install gather** 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

참고

AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: **~/.ssh/id_ed25519**)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 **'~/.ssh** 디렉터리에 있는지 확인하십시오.

참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 **ed25519** 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 **rsa** 또는 **ecdsa** 알고리즘을 사용하는 키를 생성합니다.

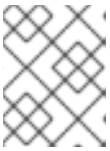
2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 **~/.ssh/id_ed25519.pub** 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 **./openshift-install gather** 명령을 사용하려는 경우 필요합니다.



참고

일부 배포에서는 **~/.ssh/id_rsa** 및 **~/.ssh/id_dsa**와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. **ssh-agent** 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. **ssh-agent**에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: **~/.ssh/id_ed25519**).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

5.5.4. Azure Marketplace 이미지 선택

Azure Marketplace 오퍼링을 사용하여 OpenShift Container Platform 클러스터를 배포하는 경우 먼저 Azure Marketplace 이미지를 가져와야 합니다. 설치 프로그램은 이 이미지를 사용하여 작업자 노드를 배포합니다. 이미지를 가져올 때 다음 사항을 고려하십시오.

- 이미지가 동일하지만 Azure Marketplace 게시자는 지역에 따라 다릅니다. 북미에 있는 경우 게시자로 **redhat** 을 지정합니다. EMEA에 있는 경우 게시자로 **redhat-limited** 를 지정합니다.
- 이 프로모션에는 **rh-ocp-worker** SKU 및 **rh-ocp-worker-gen1** SKU가 포함되어 있습니다. **rh-ocp-worker** SKU는 Hyper-V generation 버전 2 VM 이미지를 나타냅니다. OpenShift Container Platform에서 사용되는 기본 인스턴스 유형은 버전 2와 호환됩니다. 버전 1 호환 가능한 인스턴스 유형을 사용하려는 경우 **rh-ocp-worker-gen1** SKU와 연결된 이미지를 사용합니다. **rh-ocp-worker-gen1** SKU는 Hyper-V 버전 1 VM 이미지를 나타냅니다.

사전 요구 사항

- Azure CLI 클라이언트 **(az)** 를 설치했습니다.
- Azure 계정에는 제공 권한이 있으며 Azure CLI 클라이언트를 사용하여 이 계정에 로그인했습니다.

절차

1. 다음 명령 중 하나를 실행하여 사용 가능한 모든 OpenShift Container Platform 이미지를 표시합니다.
   - 북미:

```
$  az vm image list --all --offer rh-ocp-worker --publisher redhat -o table
```

출력 예

```
Offer          Publisher      Sku               Urn                                                                  Version
-------------  -------------  ----------------  -------------------------------------------------------------------  ---------------
rh-ocp-worker  RedHat         rh-ocp-worker     RedHat:rh-ocp-worker:rh-
```

```
ocpworker:4.8.2021122100               4.8.2021122100
rh-ocp-worker  RedHat          rh-ocp-worker-gen1  RedHat:rh-ocp-worker:rh-ocp-worker-
gen1:4.8.2021122100          4.8.2021122100
```

- EMEA:

```
$  az vm image list --all --offer rh-ocp-worker --publisher redhat-limited -o table
```

출력 예

```
Offer          Publisher       Sku                 Urn                                                                  Version
-------------  --------------  ------------------  -------------------------------------------------------------------
----  --------------
rh-ocp-worker  redhat-limited  rh-ocp-worker       redhat-limited:rh-ocp-worker:rh-ocp-
worker:4.8.2021122100       4.8.2021122100
rh-ocp-worker  redhat-limited  rh-ocp-worker-gen1  redhat-limited:rh-ocp-worker:rh-ocp-
worker-gen1:4.8.2021122100  4.8.2021122100
```

참고

설치하는 OpenShift Container Platform 버전에 관계없이 사용할 올바른 버전의 Azure Marketplace 이미지는 4.8.x입니다. 필요한 경우 설치 프로세스의 일부로 VM이 자동으로 업그레이드됩니다.

2. 다음 명령 중 하나를 실행하여 제안의 이미지를 검사합니다.

- 북미:

```
$ az vm image show --urn redhat:rh-ocp-worker:rh-ocp-worker:<version>
```

- EMEA:

```
$ az vm image show --urn redhat-limited:rh-ocp-worker:rh-ocp-worker:<version>
```

3. 다음 명령 중 하나를 실행하여 제안 조건을 검토합니다.

- 북미:

```
$ az vm image terms show --urn redhat:rh-ocp-worker:rh-ocp-worker:<version>
```

- EMEA:

```
$ az vm image terms show --urn redhat-limited:rh-ocp-worker:rh-ocp-worker:<version>
```

4. 다음 명령 중 하나를 실행하여 제품 약관에 동의합니다.

- 북미:

```
$ az vm image terms accept --urn redhat:rh-ocp-worker:rh-ocp-worker:<version>
```

- EMEA:

```
$ az vm image terms accept --urn redhat-limited:rh-ocp-worker:rh-ocp-worker:<version>
```

5. 제안의 이미지 세부 정보를 기록합니다. 설치를 완료하기 전에 " Marketplace Installation" 섹션에 있는 **99_openshift-cluster-api_worker-machineset-[0-2].yaml** 파일을 업데이트해야 합니다.

5.5.5. 설치 프로그램 받기

OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

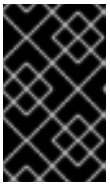
- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

프로세스

1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.
2. 인프라 공급자를 선택합니다.
3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.

중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.



중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 OpenShift Container Platform 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

5.5.6. 설치 구성 파일 만들기

Microsoft Azure에 설치하는 OpenShift Container Platform 클러스터를 사용자 지정할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.
- 서브스크립션 수준에서 서비스 주체 권한을 획득합니다.

절차

1. **install-config.yaml** 파일을 생성합니다.

   a. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

   ```
   $ ./openshift-install create install-config --dir <installation_directory> 1
   ```

   1 **<installation_directory>**는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.

   중요

   비어 있는 디렉터리를 지정합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

   b. 화면에 나타나는 지시에 따라 클라우드에 대한 구성 세부 사항을 입력합니다.

      i. 선택사항: 클러스터 시스템에 액세스하는 데 사용할 SSH 키를 선택합니다.

      참고

      설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

      ii. 대상 플랫폼으로 **azure**를 선택합니다.

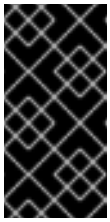
      iii. 컴퓨터에 Microsoft Azure 프로필이 저장되어 있지 않은 경우 서비스크립션 및 서비스 주체에 대해 다음 Azure 매개변수 값을 지정합니다.

      - **azure subscription id** 클러스터에 사용할 서브스크립션 ID입니다. 계정 출력의 **id** 값을 지정하십시오.
      - **azure tenant id** 테넌트 ID. 계정 출력의 **tenantId** 값을 지정하십시오.
      - **azure service principal client id** 서비스 주체의 **appId** 매개변수 값.
      - **azure service principal client secret** 서비스 주체의 **password** 매개변수 값.

      iv. 클러스터를 배포할 리전을 선택합니다.

      v. 클러스터를 배포할 기본 도메인을 선택합니다. 기본 도메인은 클러스터용으로 생성한 Azure DNS 영역에 해당합니다.

      vi. 클러스터를 설명할 수 있는 이름을 입력합니다.



중요

공용 끝점을 통해 사용할 수 있는 모든 Azure 리소스에는 리소스 이름 제한 사항이 적용되며, 특정 용어를 사용하는 리소스를 생성할 수 없습니다. Azure가 제한하는 용어 목록은 Azure 문서의 예약된 리소스 이름 오류 해결을 참조하십시오.

vii. Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 붙여넣습니다.

2. **install-config.yaml** 파일을 수정합니다. 사용 가능한 매개변수에 대한 자세한 정보는 "설치 구성 매개변수" 섹션에서 확인할 수 있습니다.

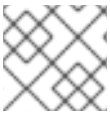
3. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.

중요

install-config.yaml 파일은 설치 과정에서 사용됩니다. 이 파일을 재사용하려면 지금 백업해야 합니다.

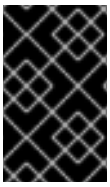
5.5.6.1. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 매개변수 값을 제공하여 클러스터를 호스팅할 클라우드 플랫폼에서 사용자 계정을 설명하고 선택사항으로 클러스터의 플랫폼을 사용자 지정합니다. **install-config.yaml** 설치 구성 파일을 생성할 때 명령줄을 통해 필요한 매개변수 값을 제공합니다. 클러스터를 사용자 지정하면 **install-config.yaml** 파일을 수정하여 플랫폼에 대한 세부 정보를 제공할 수 있습니다.



참고

설치한 후에는 **install-config.yaml** 파일에서 이러한 매개변수를 수정할 수 없습니다.



중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

5.5.6.1.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 5.1. 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열

매개변수	설명	값
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	소문자, 하이픈(**-**), 마침표(**.**)로 구성되는 문자열(예: **dev**)입니다.
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

5.5.6.1.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 5.2. 네트워크 매개변수

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체 **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.
networking.clusterNetwork	Pod의 IP 주소 블록입니다. 기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된 **cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16

매개변수	설명	값
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16** **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

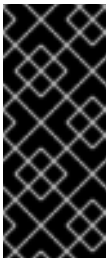
5.5.6.1.3. 선택적 구성 매개변수

선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 5.3. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열

매개변수	설명	값
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은 **3**입니다.
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열

매개변수	설명	값
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 인터넷에서 액세스할 수 없는 사설 클러스터를 배포하려면 **publish**를 **Internal**로 설정합니다. 기본값은 **External**입니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

5.5.6.1.4. 추가 Azure 구성 매개 변수

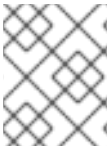
추가 Azure 구성 매개변수는 다음 표에 설명되어 있습니다.

표 5.4. 추가 Azure 매개변수

매개변수	설명	값
compute.platform.azure.osDisk.diskSizeGB	VM의 Azure 디스크 크기입니다.	디스크 크기(GB)를 나타내는 정수입니다. 기본값은 **128** 입니다.
compute.platform.azure.osDisk.diskType	디스크 유형을 정의합니다.	**standard_LRS**, **premium_LRS**, or **standardSSD_LRS**. 기본값은 **premium_LRS** 입니다.
controlPlane.platform.azure.osDisk.diskSizeGB	VM의 Azure 디스크 크기입니다.	디스크 크기(GB)를 나타내는 정수입니다. 기본값은 **1024** 입니다.
controlPlane.platform.azure.osDisk.diskType	디스크 유형을 정의합니다.	**Premium_LRS** 또는 **표준SSD_LRS**. 기본값은 **premium_LRS** 입니다.
platform.azure.baseDomainResourceGroupName	기본 도메인의 DNS 영역을 포함하는 리소스 그룹의 이름입니다.	문자열(예: **production_cluster**).

매개변수	설명	값
platform.azure.resourceGroupName	클러스터를 설치할 기존 리소스 그룹의 이름입니다. 이 리소스 그룹은 비어 있어야 하며 특정 클러스터에만 사용해야 합니다. 클러스터 구성 요소는 리소스 그룹의 모든 리소스에 대한 소유권을 가정합니다. 설치 프로그램의 서비스 주체 범위를 이 리소스 그룹으로 제한하는 경우 해당 환경에서 설치 프로그램에서 사용하는 기타 모든 리소스에 퍼블릭 DNS 영역 및 가상 네트워크와 같은 필수 권한이 있는지 확인해야 합니다. 설치 프로그램을 사용하여 클러스터를 삭제하면 이 리소스 그룹이 삭제됩니다.	문자열(예: **existing_resource_group**)
platform.azure.outboundType	클러스터를 인터넷에 연결하는 데 사용되는 아웃 바운드 라우팅 전략입니다. 사용자 정의 라우팅을 사용하는 경우 클러스터를 설치하기 전에 아웃 바운드 라우팅이 이미 구성된 경우 기존 네트워킹을 사용할 수 있어야 합니다. 설치 프로그램은 사용자 정의 라우팅 설정을 하지 않습니다.	**LoadBalancer** 또는 **UserDefinedRouting**. 기본값은 **LoadBalancer** 입니다.
platform.azure.region	클러스터를 호스팅하는 Azure 리전의 이름입니다.	유효한 리전 이름(예: **centralus**).
platform.azure.zone	시스템을 배치하려는 가용성 영역 목록. 고가용성을 위해 최소한 두 개의 영역을 지정하십시오.	영역 목록(예: **["1", "2", "3"]**)
platform.azure.networkResourceGroupName	클러스터를 배포하려는 기존 VNet이 포함된 리소스 그룹의 이름입니다. 이 이름은 **platform.azure.baseDomainResourceGroupName**과 같을 수 없습니다.	문자열.
platform.azure.virtualNetwork	클러스터를 배포할 기존 VNet의 이름입니다.	문자열.
platform.azure.controlPlaneSubnet	컨트롤 플레인 시스템을 배포할 VNet의 기존 서브넷 이름입니다.	유효한 CIDR(예: **10.0.0.0/16**)
platform.azure.computeSubnet	컴퓨팅 시스템을 배포할 VNet의 기존 서브넷 이름입니다.	유효한 CIDR(예: **10.0.0.0/16**)

매개변수	설명	값
platform.azure.cloudName	적절한 Azure API 엔드포인트에서 Azure SDK를 구성하는데 사용되는 Azure 클라우드 환경의 이름입니다. 비어있는 경우 기본값 **AzurePublicCloud**가 사용됩니다.	**AzurePublicCloud** 또는 **AzureUSGovernmentCloud** 와 같은 유효한 클라우드 환경.



참고

Azure 가용성 영역을 사용자 지정하거나 Azure 클러스터와 함께 태그를 사용하여 Azure 리소스를 구성할 수 없습니다.

5.5.6.2. 클러스터 설치를 위한 최소 리소스 요구 사항

각 클러스터 시스템이 다음과 같은 최소 요구사항을 충족해야 합니다.

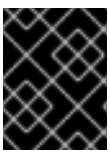
표 5.5. 최소 리소스 요구사항

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
부트스트랩	RHCOS	4	16GB	100GB	300
컨트롤 플레인	RHCOS	4	16GB	100GB	300
컴퓨팅	RHCOS, RHEL 7.9 또는 RHEL 8.4 [3]	2	8GB	100GB	300

1. SMT(동시 멀티스레딩) 또는 하이퍼 스레딩이 활성화되지 않은 경우 하나의 vCPU는 하나의 물리적 코어와 동일합니다. 활성화하면 다음과 같은 공식을 사용하여 해당 비율을 계산합니다. (코어당 스레드 수 × 코어 수) × 소켓 수 = vCPU 수

2. OpenShift Container Platform 및 Kubernetes는 디스크 성능에 민감하며 특히 10ms p99 fsync 기간이 필요한 컨트롤 플레인 노드의 etcd에 더 빠른 스토리지가 권장됩니다. 많은 클라우드 플랫폼에서 스토리지 크기와 IOPS를 함께 확장되므로 충분한 성능을 얻으려면 스토리지 볼륨을 과도하게 할당해야 할 수 있습니다.

3. 사용자가 프로비저닝한 모든 설치와 마찬가지로 클러스터에서 RHEL 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. RHEL 7 컴퓨팅 머신 사용은 더 이상 사용되지 않으며 향후 OpenShift Container Platform 4 릴리스에서 제거될 예정입니다.



중요

Premium **IO** 가 **true** 로 설정된 Azure 가상 머신을 사용해야 합니다. 머신에는 **하이퍼 VGeneration** 속성에 **V1** 이 포함되어 있어야 합니다.

5.5.6.3. Azure용 샘플 사용자 지정 install-config.yaml 파일

install-config.yaml 파일을 사용자 지정하여 OpenShift Container Platform 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.

중요

이 샘플 YAML 파일은 참조용으로만 제공됩니다. 설치 프로그램을 사용하여 **install-config.yaml** 파일을 받아서 수정해야 합니다.

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2)
  hyperthreading: Enabled (3) (4)
  name: master
  platform:
    azure:
      osDisk:
        diskSizeGB: 1024 (5)
        diskType: Premium_LRS
      type: Standard_D8s_v3
  replicas: 3
compute: (6)
- hyperthreading: Enabled (7)
  name: worker
  platform:
    azure:
      type: Standard_D2s_v3
      osDisk:
        diskSizeGB: 512 (8)
        diskType: Standard_LRS
      zones: (9)
      - "1"
      - "2"
      - "3"
  replicas: 5
metadata:
  name: test-cluster (10)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  azure:
    baseDomainResourceGroupName: resource_group (11)
    region: centralus (12)
    resourceGroupName: existing_resource_group (13)
    outboundType: Loadbalancer
    cloudName: AzurePublicCloud
```

```
pullSecret: '{"auths": ...}' 14
fips: false 15
sshKey: ssh-ed25519 AAAA... 16
```

1 10 12 14 필수 항목입니다. 설치 프로그램에서 이 값을 입력하라는 메시지를 표시합니다.

2 6 이러한 매개변수와 값을 지정하지 않으면 설치 프로그램은 기본값을 적용합니다.

3 7 **controlPlane** 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

4 동시 멀티스레딩 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. 일부 클러스터 시스템에서 동시 멀티스레딩을 비활성화할 경우에는 해당 멀티스레딩을 모든 클러스터 시스템에서 비활성화해야 합니다.

중요

동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다. 동시 멀티스레딩을 비활성화하는 경우, 사용자 시스템에 더 큰 가상 머신 유형(예: **Standard_D8s_v3**)을 사용하십시오.

5 8 사용할 디스크 크기는 GB 단위로 지정할 수 있습니다. 컨트롤 플레인 노드의 최소 권장 크기는 1024GB입니다.

9 시스템을 배포할 영역 목록을 지정합니다. 고가용성을 위해 최소한 두 개의 영역을 지정하십시오.

11 기본 도메인의 DNS 영역을 포함하는 리소스 그룹의 이름을 지정합니다.

13 클러스터를 설치할 기존 리소스 그룹의 이름을 지정합니다. 정의되지 않은 경우 클러스터에 새 리소스 그룹이 생성됩니다.

15 FIPS 모드 활성화 또는 비활성화 여부입니다. 기본적으로 FIPS 모드는 비활성화됩니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.

중요

FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.

16 선택사항으로, 클러스터의 시스템에 액세스하는 데 사용할 **sshKey** 값을 제공할 수도 있습니다.

참고

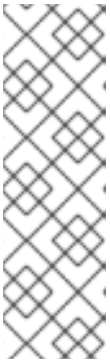
설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

5.5.6.4. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 HTTP 또는 HTTPS 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 OpenShift Container Platform 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 API에 대한 호출을 포함하여 모든 클러스터 발신(Egress) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.



참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 Red Hat OpenStack Platform (RHOSP)에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

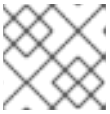
절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. 클러스터 외부에서 HTTP 연결을 구축하는 데 사용할 프록시 URL입니다. URL 스키마는 **http**여야 합니다.
2. 클러스터 외부에서 HTTPS 연결을 구축하는 데 사용할 프록시 URL입니다.
3. 대상 도메인 이름, IP 주소 또는 프록시에서 제외할 기타 네트워크 CIDR로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. ***를 사용하여 모든 대상에 대해 프록시를 바이패스합니다.
4. 이 값을 제공하면 설치 프로그램에서 추가 CA 인증서를 보유할 **openshift -config 네임스페이스에 user-ca- bundle** 이라는 구성 맵을 생성합니다. **additionalTrustBundle** 및 하나 이상의 프록시 설정을 제공하는 경우 **프록시** 오브젝트는 **trustedCA** 필드의 **user-ca-bundle** 구성 맵을 참조하도록 구성됩니다. 그러면 Cluster Network Operator에서 **trustedCA** 매개변수에 대해 지정된 콘텐츠를 RHCOS 신뢰 번들과 병합하는 **trusted-ca-bundle** 구성 맵을

생성합니다. 프록시의 ID 인증서를 RHCOS 트러스트 번들에 있는 기관에서 서명하지 않은 경우 **additionalTrustBundle** 필드가 있어야 합니다.



참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 OpenShift Container Platform을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 nil이 됩니다.

참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

5.5.7. Marketplace 설치에 대한 매니페스트 업데이트

설치를 위해 Marketplace 이미지를 선택한 경우 Marketplace 이미지를 사용하도록 매니페스트를 생성하고 수정해야 합니다.

사전 요구 사항

- **install-config.yaml** 파일을 생성하고 수정 작업을 완료했습니다.

프로세스

1. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행하여 매니페스트를 생성합니다.

```
$ openshift-install create manifests --dir <installation_dir>
```

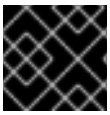
2. 컴퓨팅 머신 세트 정의의 **.spec.spec.providerSpec.value.image** 속성을 편집하여 **offer**,**publisher**,**sku**, **version** 값을 "Azure Marketplace 이미지 선택" 섹션에 수집된 세부 정보로 교체합니다. 다음은 업데이트해야 하는 세 개의 파일입니다.
   - **<installation_dir>/openshift/99_openshift-cluster-api_worker-machineset-0.yaml**
   - **<installation_dir>/openshift/99_openshift-cluster-api_worker-machineset-1.yaml**
   - **<installation_dir>/openshift/99_openshift-cluster-api_worker-machineset-2.yaml**
3. 각 파일에서 .spec. **template.spec.providerSpec.value.resourceID** 속성 값을 빈 값(**""**)으로 바꿉니다.
4. 각 파일에서 **type** 속성을 **MarketplaceWithPlan** 로 설정합니다.
5. 첫 번째 머신 세트 파일을 예로 사용하면 **.spec.template.spec.providerSpec.value.image** 섹션은 다음 예와 유사해야 합니다.

```
image:
  offer: rh-ocp-worker
  publisher: redhat
```

```
resourceID: ""
sku: rh-ocp-worker
version: 4.8.2021122100
type: MarketplaceWithPlan
```

5.5.8. 클러스터 배포

호환되는 클라우드 플랫폼에 OpenShift Container Platform을 설치할 수 있습니다.



중요

최초 설치 과정에서 설치 프로그램의 **create cluster** 명령을 한 번만 실행할 수 있습니다.

사전 요구 사항

- 클러스터를 호스팅하는 클라우드 플랫폼으로 계정을 구성합니다.
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

프로세스

1. 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터 배포를 초기화합니다.

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** 값으로 사용자 지정한 **./install-config.yaml** 파일의 위치를 지정합니다.

2 다른 설치 세부 사항을 보려면 **info** 대신 **warn**, **debug** 또는 **error**를 지정합니다.

참고

호스트에 구성된 클라우드 공급자 계정에 클러스터를 배포하기에 충분한 권한이 없는 경우, 설치 프로세스가 중단되고 누락된 권한을 알리는 메시지가 표시됩니다.

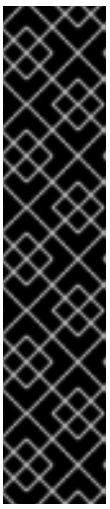
클러스터 배포가 완료되면 웹 콘솔로 연결되는 링크와 **kubeadmin** 사용자의 인증 정보가 포함된 클러스터 액세스 지침이 사용자 터미널에 표시됩니다.

출력 예

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```

참고

설치에 성공하면 클러스터 액세스 및 인증 정보도 **<installation_directory>/.openshift_install.log**로 출력됩니다.



중요

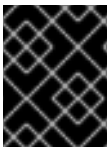
- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

중요

설치 프로그램에서 생성되는 파일이나 설치 프로그램을 삭제해서는 안 됩니다. 클러스터를 삭제하려면 두 가지가 모두 필요합니다.

5.5.9. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift CLI(**oc**)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 **oc**를 설치할 수 있습니다.



중요

이전 버전의 **oc**를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 **oc**를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 Linux에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Linux Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. **oc** 바이너리를 **PATH**에 있는 디렉터리에 배치합니다.
PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

-

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치
다음 절차에 따라 Windows에 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Windows Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. ZIP 프로그램으로 아카이브의 압축을 풉니다.
5. **oc** 바이너리를 **PATH**에 있는 디렉터리로 이동합니다.
   PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

   ```
   C:\> path
   ```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치
다음 절차에 따라 macOS에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 MacOSX Client** 항목 옆에 있는 **지금 다운로드를** 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 해제하고 압축을 풉니다.
5. **oc** 바이너리 PATH의 디렉터리로 이동합니다.
   PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

   ```
   $ echo $PATH
   ```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

5.5.10. CLI를 사용하여 클러스터에 로그인

클러스터 **kubeconfig** 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다.
kubeconfig 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러

스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- **oc** CLI를 설치했습니다.

프로세스

1. **kubeadmin** 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 **oc** 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

추가 리소스

- OpenShift Container Platform 웹 콘솔 액세스 및 이해에 대한 자세한 내용은 웹 콘솔에 액세스를 참조하십시오.

5.5.11. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

5.5.12. 다음 단계

- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.

5.6. 네트워크 사용자 지정 설정을 사용하여 AZURE에 클러스터 설치

OpenShift Container Platform 버전 4.9에서는 설치 프로그램이 Microsoft Azure에 프로비저닝하는 인프라에 사용자 지정 네트워크 구성으로 클러스터를 설치할 수 있습니다. 네트워크 구성을 사용자 지정할 경우, 클러스터가 사용자 환경의 기존 IP 주소 할당과 공존하고 기존 MTU 및 VXLAN 구성과 통합될 수 있습니다.

설치 과정에서 네트워크 구성 매개변수를 대부분 설정해야 하며, 실행 중인 클러스터에서는 **kubeProxy** 구성 매개변수만 수정할 수 있습니다.

5.6.1. 사전 요구 사항

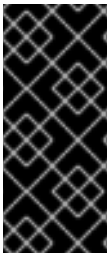
- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비 에 대한 문서를 읽습니다.
- 클러스터를 호스팅하고, 클러스터를 배포할 테스트 및 검증된 리전을 결정하도록 Azure 계정을 구성합니다.
- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성 해야 합니다.
- 사용자의 환경에서 클라우드 ID 및 액세스 관리(IAM) API에 액세스할 수 없거나 **kube-system** 네임스페이스에 관리자 수준의 인증 정보 시크릿을 저장하지 않으려면 IAM 인증 정보 수동 생성 및 유지 관리를 참조하십시오. 수동 모드는 클라우드 IAM API에 연결할 수 없는 환경에서도 사용할 수 있습니다.

5.6.2. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.



중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

5.6.3. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 **core** 사용자의 **~/.ssh/authorized_keys** 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 **core**로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다. SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. **./openshift-install gather** 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

참고

AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: **~/.ssh/id_ed25519**)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 **'~/.ssh** 디렉터리에 있는지 확인하십시오.

참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 **ed25519** 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 **rsa** 또는 **ecdsa** 알고리즘을 사용하는 키를 생성합니다.

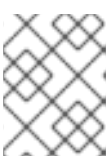
2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 **~/.ssh/id_ed25519.pub** 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 **./openshift-install gather** 명령을 사용하려는 경우 필요합니다.



참고

일부 배포에서는 **~/.ssh/id_rsa** 및 **~/.ssh/id_dsa**와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. **ssh-agent** 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. **ssh-agent**에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: **~/.ssh/id_ed25519**).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

5.6.4. 설치 프로그램 받기

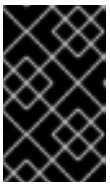
OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

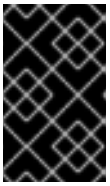
프로세스

1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.
2. 인프라 공급자를 선택합니다.
3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.



중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.



중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 OpenShift Container Platform 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

5.6.5. 설치 구성 파일 만들기

Microsoft Azure에 설치하는 OpenShift Container Platform 클러스터를 사용자 지정할 수 있습니다.

사전 요구 사항

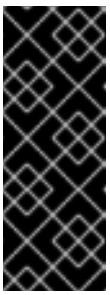
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.
- 서브스크립션 수준에서 서비스 주체 권한을 획득합니다.

절차

1. **install-config.yaml** 파일을 생성합니다.
   a. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 **<installation_directory>**는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.



중요

비어 있는 디렉터리를 지정합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

   b. 화면에 나타나는 지시에 따라 클라우드에 대한 구성 세부 사항을 입력합니다.
      i. 선택사항: 클러스터 시스템에 액세스하는 데 사용할 SSH 키를 선택합니다.

참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

ii. 대상 플랫폼으로 **azure**를 선택합니다.

iii. 컴퓨터에 Microsoft Azure 프로필이 저장되어 있지 않은 경우 서비스크립션 및 서비스 주체에 대해 다음 Azure 매개변수 값을 지정합니다.

- **azure subscription id** 클러스터에 사용할 서브스크립션 ID입니다. 계정 출력의 **id** 값을 지정하십시오.
- **azure tenant id** 테넌트 ID. 계정 출력의 **tenantId** 값을 지정하십시오.
- **azure service principal client id** 서비스 주체의 **appId** 매개변수 값.
- **azure service principal client secret** 서비스 주체의 **password** 매개변수 값.

iv. 클러스터를 배포할 리전을 선택합니다.

v. 클러스터를 배포할 기본 도메인을 선택합니다. 기본 도메인은 클러스터용으로 생성한 Azure DNS 영역에 해당합니다.

vi. 클러스터를 설명할 수 있는 이름을 입력합니다.

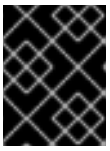
중요

공용 끝점을 통해 사용할 수 있는 모든 Azure 리소스에는 리소스 이름 제한 사항이 적용되며, 특정 용어를 사용하는 리소스를 생성할 수 없습니다. Azure가 제한하는 용어 목록은 Azure 문서의 예약된 리소스 이름 오류 해결을 참조하십시오.

vii. Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 붙여넣습니다.

2. **install-config.yaml** 파일을 수정합니다. 사용 가능한 매개변수에 대한 자세한 정보는 "설치 구성 매개변수" 섹션에서 확인할 수 있습니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.



중요

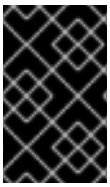
install-config.yaml 파일은 설치 과정에서 사용됩니다. 이 파일을 재사용하려면 지금 백업해야 합니다.

5.6.5.1. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 매개변수 값을 제공하여 클러스터를 호스팅할 클라우드 플랫폼에서 사용자 계정을 설명하고 선택사항으로 클러스터의 플랫폼을 사용자 지정합니다. **install-config.yaml** 설치 구성 파일을 생성할 때 명령줄을 통해 필요한 매개변수 값을 제공합니다. 클러스터를 사용자 지정하면 **install-config.yaml** 파일을 수정하여 플랫폼에 대한 세부 정보를 제공할 수 있습니다.

참고

설치한 후에는 **install-config.yaml** 파일에서 이러한 매개변수를 수정할 수 없습니다.



중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

5.6.5.1.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 5.6. 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	소문자, 하이픈(**-**), 마침표(**.**)로 구성되는 문자열(예: **dev**)입니다.

매개변수	설명	값
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

5.6.5.1.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 5.7. 네트워크 매개변수

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체 **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.

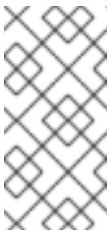
매개변수	설명	값
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.

networking.clusterNetwork	Pod의 IP 주소 블록입니다.
	
	기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다.
	
	여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.
		개체의 배열입니다. 예를 들면 다음과 같습니다.
		
		`networking:`
		`  clusterNetwork:`
		`  - cidr: 10.128.0.0/14`
		`    hostPrefix: 23`

networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된**cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.

networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다.
	
	OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.
		CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다.
		
		`networking:`
		`  serviceNetwork:`
		`   - 172.30.0.0/16`

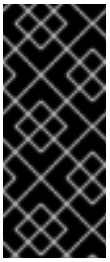
networking.machineNetwork	시스템의 IP 주소 블록입니다.
	
	여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.
		개체의 배열입니다. 예를 들면 다음과 같습니다.
		
		`networking:`
		`  machineNetwork:`
		`  - cidr: 10.0.0.0/16`

매개변수	설명	값
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16**  **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

5.6.5.1.3. 선택적 구성 매개변수

선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 5.8. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**

매개변수	설명	값
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은**3**입니다.
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 인터넷에서 액세스할 수 없는 사설 클러스터를 배포하려면 **publish**를 **Internal**로 설정합니다. 기본값은 **External**입니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

5.6.5.1.4. 추가 Azure 구성 매개 변수

추가 Azure 구성 매개변수는 다음 표에 설명되어 있습니다.

표 5.9. 추가 Azure 매개변수

매개변수	설명	값
compute.platform.azure.osDisk.diskSizeGB	VM의 Azure 디스크 크기입니다.	디스크 크기(GB)를 나타내는 정수입니다. 기본값은 **128** 입니다.
compute.platform.azure.osDisk.diskType	디스크 유형을 정의합니다.	**standard_LRS**, **premium_LRS**, or **standardSSD_LRS**. 기본값은 **premium_LRS** 입니다.
controlPlane.platform.azure.osDisk.diskSizeGB	VM의 Azure 디스크 크기입니다.	디스크 크기(GB)를 나타내는 정수입니다. 기본값은 **1024** 입니다.
controlPlane.platform.azure.osDisk.diskType	디스크 유형을 정의합니다.	**Premium_LRS** 또는 **표준SSD_LRS**. 기본값은 **premium_LRS** 입니다.
platform.azure.baseDomainResourceGroupName	기본 도메인의 DNS 영역을 포함하는 리소스 그룹의 이름입니다.	문자열(예: **production_cluster**).

매개변수	설명	값
platform.azure.resourceGroupName	클러스터를 설치할 기존 리소스 그룹의 이름입니다. 이 리소스 그룹은 비어 있어야 하며 특정 클러스터에만 사용해야 합니다. 클러스터 구성 요소는 리소스 그룹의 모든 리소스에 대한 소유권을 가정합니다. 설치 프로그램의 서비스 주체 범위를 이 리소스 그룹으로 제한하는 경우 해당 환경에서 설치 프로그램에서 사용하는 기타 모든 리소스에 퍼블릭 DNS 영역 및 가상 네트워크와 같은 필수 권한이 있는지 확인해야 합니다. 설치 프로그램을 사용하여 클러스터를 삭제하면 이 리소스 그룹이 삭제됩니다.	문자열(예: **existing_resource_group**)
platform.azure.outboundType	클러스터를 인터넷에 연결하는 데 사용되는 아웃 바운드 라우팅 전략입니다. 사용자 정의 라우팅을 사용하는 경우 클러스터를 설치하기 전에 아웃 바운드 라우팅이 이미 구성된 경우 기존 네트워킹을 사용할 수 있어야 합니다. 설치 프로그램은 사용자 정의 라우팅 설정을 하지 않습니다.	**LoadBalancer** 또는 **UserDefinedRouting**. 기본값은 **LoadBalancer** 입니다.
platform.azure.region	클러스터를 호스팅하는 Azure 리전의 이름입니다.	유효한 리전 이름(예: **centralus**).
platform.azure.zone	시스템을 배치하려는 가용성 영역 목록. 고가용성을 위해 최소한 두 개의 영역을 지정하십시오.	영역 목록(예: **["1", "2", "3"]**)
platform.azure.networkResourceGroupName	클러스터를 배포하려는 기존 VNet이 포함된 리소스 그룹의 이름입니다. 이 이름은 **platform.azure.baseDomainResourceGroupName**과 같을 수 없습니다.	문자열.
platform.azure.virtualNetwork	클러스터를 배포할 기존 VNet의 이름입니다.	문자열.
platform.azure.controlPlaneSubnet	컨트롤 플레인 시스템을 배포할 VNet의 기존 서브넷 이름입니다.	유효한 CIDR(예: **10.0.0.0/16**)
platform.azure.computeSubnet	컴퓨팅 시스템을 배포할 VNet의 기존 서브넷 이름입니다.	유효한 CIDR(예: **10.0.0.0/16**)

매개변수	설명	값
platform.azure.cloudName	적절한 Azure API 엔드포인트에서 Azure SDK를 구성하는데 사용되는 Azure 클라우드 환경의 이름입니다. 비어있는 경우 기본값 **AzurePublicCloud**가 사용됩니다.	**AzurePublicCloud** 또는 **AzureUSGovernmentCloud** 와 같은 유효한 클라우드 환경.

참고

Azure 가용성 영역을 사용자 지정하거나 Azure 클러스터와 함께 태그를 사용하여 Azure 리소스를 구성할 수 없습니다.

5.6.5.2. 클러스터 설치를 위한 최소 리소스 요구 사항

각 클러스터 시스템이 다음과 같은 최소 요구사항을 충족해야 합니다.

표 5.10. 최소 리소스 요구사항

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
부트스트랩	RHCOS	4	16GB	100GB	300
컨트롤 플레인	RHCOS	4	16GB	100GB	300
컴퓨팅	RHCOS, RHEL 7.9 또는 RHEL 8.4 [3]	2	8GB	100GB	300

1. SMT(동시 멀티스레딩) 또는 하이퍼 스레딩이 활성화되지 않은 경우 하나의 vCPU는 하나의 물리적 코어와 동일합니다. 활성화하면 다음과 같은 공식을 사용하여 해당 비율을 계산합니다. (코어당 스레드 수 × 코어 수) × 소켓 수 = vCPU 수

2. OpenShift Container Platform 및 Kubernetes는 디스크 성능에 민감하며 특히 10ms p99 fsync 기간이 필요한 컨트롤 플레인 노드의 etcd에 더 빠른 스토리지가 권장됩니다. 많은 클라우드 플랫폼에서 스토리지 크기와 IOPS를 함께 확장되므로 충분한 성능을 얻으려면 스토리지 볼륨을 과도하게 할당해야 할 수 있습니다.

3. 사용자가 프로비저닝한 모든 설치와 마찬가지로 클러스터에서 RHEL 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. RHEL 7 컴퓨팅 머신 사용은 더 이상 사용되지 않으며 향후 OpenShift Container Platform 4 릴리스에서 제거될 예정입니다.

중요

Premium **IO** 가 **true** 로 설정된 Azure 가상 머신을 사용해야 합니다. 머신에는 **하이퍼 VGeneration** 속성에 **V1** 이 포함되어 있어야 합니다.

5.6.5.3. Azure용 샘플 사용자 지정 install-config.yaml 파일

install-config.yaml 파일을 사용자 지정하여 OpenShift Container Platform 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.

중요

이 샘플 YAML 파일은 참조용으로만 제공됩니다. 설치 프로그램을 사용하여 **install-config.yaml** 파일을 받아서 수정해야 합니다.

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2)
  hyperthreading: Enabled (3) (4)
  name: master
  platform:
    azure:
      osDisk:
        diskSizeGB: 1024 (5)
        diskType: Premium_LRS
      type: Standard_D8s_v3
  replicas: 3
compute: (6)
- hyperthreading: Enabled (7)
  name: worker
  platform:
    azure:
      type: Standard_D2s_v3
      osDisk:
        diskSizeGB: 512 (8)
        diskType: Standard_LRS
      zones: (9)
      - "1"
      - "2"
      - "3"
  replicas: 5
metadata:
  name: test-cluster (10)
networking: (11)
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  azure:
    baseDomainResourceGroupName: resource_group (12)
    region: centralus (13)
    resourceGroupName: existing_resource_group (14)
    outboundType: Loadbalancer
    cloudName: AzurePublicCloud
```

```
pullSecret: '{"auths": ...}' 15
fips: false 16
sshKey: ssh-ed25519 AAAA... 17
```

1 10 13 15 필수 항목입니다. 설치 프로그램에서 이 값을 입력하라는 메시지를 표시합니다.

2 6 11 이러한 매개변수와 값을 지정하지 않으면 설치 프로그램은 기본값을 적용합니다.

3 7 **controlPlane** 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

4 동시 멀티스레딩 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. 일부 클러스터 시스템에서 동시 멀티스레딩을 비활성화할 경우에는 해당 멀티스레딩을 모든 클러스터 시스템에서 비활성화해야 합니다.

중요

동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다. 동시 멀티스레딩을 비활성화하는 경우, 사용자 시스템에 더 큰 가상 머신 유형(예: **Standard_D8s_v3**)을 사용하십시오.

5 8 사용할 디스크 크기는 GB 단위로 지정할 수 있습니다. 컨트롤 플레인 노드의 최소 권장 크기는 1024GB입니다.

9 시스템을 배포할 영역 목록을 지정합니다. 고가용성을 위해 최소한 두 개의 영역을 지정하십시오.

12 기본 도메인의 DNS 영역을 포함하는 리소스 그룹의 이름을 지정합니다.

14 클러스터를 설치할 기존 리소스 그룹의 이름을 지정합니다. 정의되지 않은 경우 클러스터에 새 리소스 그룹이 생성됩니다.

16 FIPS 모드 활성화 또는 비활성화 여부입니다. 기본적으로 FIPS 모드는 비활성화됩니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.

중요

FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.

17 선택사항으로, 클러스터의 시스템에 액세스하는 데 사용할 **sshKey** 값을 제공할 수도 있습니다.

참고

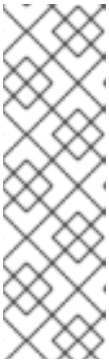
설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

5.6.5.4. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 HTTP 또는 HTTPS 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 OpenShift Container Platform 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 API에 대한 호출을 포함하여 모든 클러스터 발신(Egress) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.



참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 Red Hat OpenStack Platform (RHOSP)에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. 클러스터 외부에서 HTTP 연결을 구축하는 데 사용할 프록시 URL입니다. URL 스키마는 **http**여야 합니다.
2. 클러스터 외부에서 HTTPS 연결을 구축하는 데 사용할 프록시 URL입니다.
3. 대상 도메인 이름, IP 주소 또는 프록시에서 제외할 기타 네트워크 CIDR로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. ***를 사용하여 모든 대상에 대해 프록시를 바이패스합니다.
4. 이 값을 제공하면 설치 프로그램에서 추가 CA 인증서를 보유할 **openshift -config 네임스페이스에 user-ca- bundle** 이라는 구성 맵을 생성합니다. **additionalTrustBundle** 및 하나 이상의 프록시 설정을 제공하는 경우 **프록시** 오브젝트는 **trustedCA** 필드의 **user-ca-bundle** 구성 맵을 참조하도록 구성됩니다. 그러면 Cluster Network Operator에서 **trustedCA** 매개변수에 대해 지정된 콘텐츠를 RHCOS 신뢰 번들과 병합하는 **trusted-ca-bundle** 구성 맵을

생성합니다. 프록시의 ID 인증서를 RHCOS 트러스트 번들에 있는 기관에서 서명하지 않은 경우 **additionalTrustBundle** 필드가 있어야 합니다.

참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 OpenShift Container Platform을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 nil이 됩니다.

참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

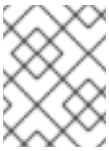
5.6.6. 네트워크 구성 단계

OpenShift Container Platform을 설치하기 전에 네트워크 구성을 사용자 지정할 수 있습니다.

1 단계

매니페스트 파일을 생성하기 전에 **install-config.yaml** 파일에서 다음 네트워크 관련 필드를 사용자 지정할 수 있습니다.

- **networking.networkType**
- **networking.clusterNetwork**
- **networking.serviceNetwork**
- **networking.machineNetwork**
  이러한 필드에 대한 자세한 내용은 *설치 구성 매개 변수*에서 참조하십시오.



참고

기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

2 단계

openshift-install create manifests 를 실행하여 매니페스트 파일을 생성한 후 수정할 필드로 사용자 지정된 Cluster Network Operator 매니페스트를 정의할 수 있습니다. 매니페스트를 사용하여 고급 네트워크 구성을 지정할 수 있습니다.

install-config.yaml 파일에서 1 단계에 지정된 값을 2 단계에서 재정의할 수 없습니다. 그러나 2 단계에서 클러스터 네트워크 공급자를 추가로 사용자 지정할 수 있습니다.

5.6.7. 고급 네트워크 구성 지정

클러스터 네트워크 제공자의 고급 네트워크 구성을 사용하여 클러스터를 기존 네트워크 환경에 통합할 수 있습니다. 클러스터를 설치하기 전에만 고급 네트워크 구성을 지정할 수 있습니다.

중요

설치 프로그램에서 생성한 OpenShift Container Platform 매니페스트 파일을 수정하여 네트워크 구성을 사용자 정의하는 것은 지원되지 않습니다. 다음 절차에서와 같이 생성한 매니페스트 파일을 적용할 수 있습니다.

사전 요구 사항

- **install-config.yaml** 파일을 생성하고 수정 작업을 완료했습니다.

프로세스

1. 설치 프로그램이 포함된 디렉터리로 변경하고 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>**는 클러스터의 **install-config.yaml** 파일이 포함된 디렉터리의 이름을 지정합니다.

2. **<installation_directory>/ manifests/** 디렉토리에 **cluster-network-03-config.yml**이라는 stub 매니페스트 파일을 만듭니다.

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
```

3. 다음과 같이 **cluster-network-03-config.yml** 파일에서 클러스터의 고급 네트워크 구성을 지정합니다.

OpenShift SDN 네트워크 공급자에 대한 다른 VXLAN 포트 지정

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    openshiftSDNConfig:
      vxlanPort: 4800
```

OVN-Kubernetes 네트워크 공급자의 IPsec 활성화

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    ovnKubernetesConfig:
      ipsecConfig: {}
```

4. 선택사항: **manifests/cluster-network-03-config.yml** 파일을 백업합니다. 설치 프로그램은 Ignition 구성 파일을 생성할 때 **manifests/** 디렉터리를 사용합니다.

5.6.8. CNO(Cluster Network Operator) 구성

클러스터 네트워크의 구성은 CNO(Cluster Network Operator) 구성의 일부로 지정되며 **cluster**라는 이름의 CR(사용자 정의 리소스) 오브젝트에 저장됩니다. CR은 **operator.openshift.io** API 그룹에서 **Network** API의 필드를 지정합니다.

CNO 구성은 **Network.config.openshift.io** API 그룹의 **Network** API에서 클러스터 설치 중에 다음 필드를 상속하며 이러한 필드는 변경할 수 없습니다.

clusterNetwork

Pod IP 주소가 할당되는 IP 주소 풀입니다.

serviceNetwork

서비스를 위한 IP 주소 풀입니다.

defaultNetwork.type

OpenShift SDN 또는 OVN-Kubernetes와 같은 클러스터 네트워크 공급자입니다.

cluster라는 CNO 오브젝트에서 **defaultNetwork** 오브젝트의 필드를 설정하여 클러스터의 클러스터 네트워크 공급자 구성을 지정할 수 있습니다.

5.6.8.1. CNO(Cluster Network Operator) 구성 오브젝트

CNO(Cluster Network Operator)의 필드는 다음 표에 설명되어 있습니다.

표 5.11. CNO(Cluster Network Operator) 구성 오브젝트

필드	유형	설명
metadata.name	**string**	CNO 개체 이름입니다. 이 이름은 항상 **cluster**입니다.
spec.clusterNetwork	**array**	Pod IP 주소가 할당되는 IP 주소 블록과 클러스터의 각 개별 노드에 할당된 서브넷 접두사 길이를 지정하는 목록입니다. 예를 들면 다음과 같습니다. spec: clusterNetwork: - cidr: 10.128.0.0/19 hostPrefix: 23 - cidr: 10.128.32.0/19 hostPrefix: 23 매니페스트를 생성하기 전에 **install-config.yaml** 파일에서만 이 필드를 사용자 지정할 수 있습니다. 값은 매니페스트 파일에서 읽기 전용입니다.

필드	유형	설명
spec.serviceNetwork	**array**	서비스의 IP 주소 블록입니다. OpenShift SDN 및 OVN-Kubernetes CNI(Container Network Interface) 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다. 예를 들면 다음과 같습니다. spec: serviceNetwork: - 172.30.0.0/14 매니페스트를 생성하기 전에 **install-config.yaml** 파일에서만 이 필드를 사용자 지정할 수 있습니다. 값은 매니페스트 파일에서 읽기 전용입니다.
spec.defaultNetwork	**object**	클러스터 네트워크의 CNI(Container Network Interface) 클러스터 네트워크 공급자를 구성합니다.
spec.kubeProxyConfig	**object**	이 개체의 필드는 kube-proxy 구성을 지정합니다. OVN-Kubernetes 클러스터 네트워크 공급자를 사용하는 경우 kube-proxy 구성이 적용되지 않습니다.

defaultNetwork 오브젝트 구성
defaultNetwork 오브젝트의 값은 다음 표에 정의되어 있습니다.

표 5.12. defaultNetwork 오브젝트

필드	유형	설명
type	**string**	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, 클러스터 네트워크 공급자가 설치 중에 선택됩니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다. **참고** OpenShift Container Platform은 기본적으로 OpenShift SDN CNI(Container Network Interface) 클러스터 네트워크 공급자를 사용합니다.
openshiftSDNConfig	**object**	이 오브젝트는 OpenShift SDN 클러스터 네트워크 공급자에만 유효합니다.
ovnKubernetesConfig	**object**	이 오브젝트는 OVN-Kubernetes 클러스터 네트워크 공급자에만 유효합니다.

OpenShift SDN CNI 네트워크 공급자에 대한 구성

다음 표에서는 OpenShift SDN Container Network Interface (CNI) 클러스터 네트워크 공급자의 구성 필드를 설명합니다.

표 5.13. openshiftSDNConfig 오브젝트

필드	유형	설명
mode	**string**	OpenShift SDN의 네트워크 격리 모드를 구성합니다. 기본값은 **NetworkPolicy**입니다. **Multitenant** 및 **Subnet** 값은 OpenShift Container Platform 3.x 버전과의 역호환할 수 있지만 권장되지는 않습니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
mtu	**integer**	VXLAN 오버레이 네트워크의 최대 전송 단위(MTU)입니다. 이는 기본 네트워크 인터페이스의 MTU를 기준으로 자동 탐지됩니다. 일반적으로 감지된 MTU를 재정의할 필요는 없습니다. 자동 감지 값이 예상 밖인 경우 노드의 기본 네트워크 인터페이스의 MTU가 올바른지 확인합니다. 이 옵션을 사용하여 노드의 기본 네트워크 인터페이스의 MTU 값을 변경할 수 없습니다. 클러스터에 다른 노드에 대한 다른 MTU 값이 필요한 경우 이 값을 클러스터의 가장 낮은 MTU 값보다 **50** 미만으로 설정해야 합니다. 예를 들어 클러스터의 일부 노드에는 **9001**의 MTU가 있고 일부에는 **1500**의 MTU가 있는 경우 이 값을 **1450**으로 설정해야 합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
vxlanPort	**integer**	모든 VXLAN 패킷에 사용할 포트입니다. 기본값은 **4789**입니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다. 기존 노드가 다른 VXLAN 네트워크에 속하는 가상 환경에서 실행 중인 경우에는 기본값을 변경해야 할 수도 있습니다. 예를 들어 VMware NSX-T 위에서 OpenShift SDN 오버레이를 실행할 때 두 SDN이 동일한 기본 VXLAN 포트 번호를 사용하므로 VXLAN의 대체 포트를 선택해야 합니다. AWS(Amazon Web Services)에서는 포트 **9000**과 포트 **9999** 사이에서 VXLAN의 대체 포트를 선택할 수 있습니다.

OpenShift SDN 구성 예

```
defaultNetwork:
  type: OpenShiftSDN
  openshiftSDNConfig:
    mode: NetworkPolicy
    mtu: 1450
    vxlanPort: 4789
```

OVN-Kubernetes CNI 클러스터 네트워크 공급자에 대한 구성

다음 표에서는 OVN-Kubernetes CNI 클러스터 네트워크 공급자의 구성 필드를 설명합니다.

표 5.14. ovnKubernetesConfig object

필드	유형	설명
mtu	**integer**	Geneve(Generic Network Virtualization Encapsulation) 오버레이 네트워크의 MTU(최대 전송 단위)입니다. 이는 기본 네트워크 인터페이스의 MTU를 기준으로 자동 탐지됩니다. 일반적으로 감지된 MTU를 재정의할 필요는 없습니다. 자동 감지 값이 예상 밖인 경우 노드의 기본 네트워크 인터페이스의 MTU가 올바른지 확인합니다. 이 옵션을 사용하여 노드의 기본 네트워크 인터페이스의 MTU 값을 변경할 수 없습니다. 클러스터에 다른 노드에 대한 다른 MTU 값이 필요한 경우, 이 값을 클러스터의 가장 낮은 MTU 값보다 **100** 미만으로 설정해야 합니다. 예를 들어, 클러스터의 일부 노드에 **9001**의 MTU가 있고 일부에는 **1500**의 MTU가 있는 경우 이 값을 **1400**으로 설정해야 합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
genevePort	**integer**	모든 Geneve 패킷에 사용할 포트입니다. 기본값은 **6081**입니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
ipsecConfig	**object**	IPsec 암호화를 활성화하려면 빈 오브젝트를 지정합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
policyAuditConfig	**object**	네트워크 정책 감사 로깅을 사용자 정의할 구성 오브젝트를 지정합니다. 설정되지 않으면 기본값 감사 로그 설정이 사용됩니다.

표 5.15. **policyAuditConfig** object

필드	유형	설명
rateLimit	integer	노드당 1초마다 생성할 최대 메시지 수입니다. 기본값은 초당 **20** 개의 메시지입니다.
maxFileSize	integer	감사 로그의 최대 크기(바이트)입니다. 기본값은 **50000000 또는 50** MB입니다.
대상	string	다음 추가 감사 로그 대상 중 하나입니다. **libc** 호스트에서 journald 프로세스의 libc **syslog()** 함수입니다. **udp:<host>:<port>** syslog 서버입니다. **<host>:<port>**를 syslog 서버의 호스트 및 포트로 바꿉니다. **unix:<file>** **<file>**로 지정된 Unix Domain Socket 파일입니다. **null** 감사 로그를 추가 대상으로 보내지 마십시오.

필드	유형	설명
syslogFacility	string	RFC5424에 정의된 **kern**과 같은 syslog 기능입니다. 기본값은 **local0**입니다.

OVN-Kubernetes 구성 예

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
    ipsecConfig: {}
```

kubeProxyConfig 오브젝트 구성
kubeProxyConfig 오브젝트의 값은 다음 표에 정의되어 있습니다.

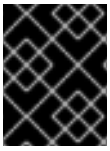
표 5.16. **kubeProxyConfig** object

필드	유형	설명
iptablesSyncPeriod	**string**	**iptables** 규칙의 새로 고침 간격입니다. 기본값은 **30s**입니다. 유효 접미사로 **s**, **m**, **h**가 있으며, 자세한 설명은 Go **time** 패키지 문서를 참조하십시오. **참고** OpenShift Container Platform 4.3 이상에서는 성능이 개선되어 더 이상 **iptablesSyncPeriod** 매개변수를 조정할 필요가 없습니다.

proxyArguments.iptables-min-sync-period	**array**	**iptables** 규칙을 새로 고치기 전 최소 기간입니다. 이 필드를 통해 새로 고침 간격이 너무 짧지 않도록 조정할 수 있습니다. 유효 접미사로 **s**, **m**, **h**가 있으며, 자세한 설명은 Go **time** 패키지를 참조하십시오. 기본값은 다음과 같습니다.
		
		`kubeProxyConfig:`
		`  proxyArguments:`
		`  iptables-min-sync-period:`
		`  - 0s`

5.6.9. OVN-Kubernetes로 하이브리드 네트워킹 구성

OVN-Kubernetes에서 하이브리드 네트워킹을 사용하도록 클러스터를 구성할 수 있습니다. 이를 통해 다양한 노드 네트워킹 구성을 지원하는 하이브리드 클러스터를 사용할 수 있습니다. 예를 들어 클러스터에서 Linux 및 Windows 노드를 모두 실행하려면 이 작업이 필요합니다.



중요

클러스터를 설치하는 동안 OVN-Kubernetes를 사용하여 하이브리드 네트워킹을 구성해야 합니다. 설치 프로세스 후에는 하이브리드 네트워킹으로 전환할 수 없습니다.

사전 요구 사항

- **install-config.yaml** 파일에 **networking.networkType**매개변수의 **OVNKubernetes**가 정의되어 있어야 합니다. 자세한 내용은 선택한 클라우드 공급자에서 OpenShift Container Platform 네트워크 사용자 정의 설정에 필요한 설치 문서를 참조하십시오.

프로세스

1. 설치 프로그램이 포함된 디렉터리로 변경하고 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory>
```

   다음과 같습니다.

   <installation_directory>

   클러스터의 **install-config.yaml** 파일이 포함된 디렉토리의 이름을 지정합니다.

2. **<installation_directory>/ manifests/** 디렉토리에 **cluster-network-03-config.yml**이라는 stub 매니페스트 파일을 만듭니다.

```
$ cat <<EOF > <installation_directory>/manifests/cluster-network-03-config.yml
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
EOF
```

   다음과 같습니다.

   <installation_directory>

   클러스터의 **manifests/** 디렉터리가 포함된 디렉터리 이름을 지정합니다.

3. 편집기에서 **cluster-network-03-config.yml** 파일을 열고 다음 예와 같이 하이브리드 네트워킹을 사용하여 OVN-Kubernetes를 구성합니다.

   하이브리드 네트워킹 구성 지정

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    ovnKubernetesConfig:
      hybridOverlayConfig:
        hybridClusterNetwork: 1
```

```
    - cidr: 10.132.0.0/14
      hostPrefix: 23
    hybridOverlayVXLANPort: 9898 (2)
```

(1) 추가 오버레이 네트워크의 노드에 사용되는 CIDR 구성을 지정합니다. **hybridClusterNetwork** CIDR은 **clusterNetwork** CIDR과 중복될 수 없습니다.

(2) 추가 오버레이 네트워크에 대한 사용자 정의 VXLAN 포트를 지정합니다. 이는 vSphere에 설치된 클러스터에서 Windows 노드를 실행해야 하며 다른 클라우드 공급자에 대해 구성해서는 안 됩니다. 사용자 정의 포트는 기본 **4789** 포트를 제외한 모든 오픈 포트일 수 있습니다. 이 요구 사항에 대한 자세한 내용은 호스트 간의 포드 투 포트 연결 중단 에 대한 Microsoft 문서를 참조하십시오.

참고

Windows Server LTSC(Long-Term Servicing Channel): Windows Server 2019는 사용자 지정 **hybridOverlayVXLANPort** 값이 있는 클러스터에서 지원되지 않습니다. 이 Windows 서버 버전은 사용자 지정 VXLAN 포트를 선택하는 것을 지원하지 않기 때문입니다.

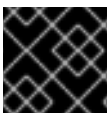
4. **cluster-network-03-config.yml** 파일을 저장하고 텍스트 편집기를 종료합니다.오.
5. 선택사항: **manifests/cluster-network-03-config.yml** 파일을 백업합니다. 설치 프로그램은 클러스터를 생성할 때 **manifests/** 디렉터리를 삭제합니다.

참고

동일한 클러스터에서 Linux 및 Windows 노드를 사용하는 방법에 대한 자세한 내용은 Windows 컨테이너 워크로드 이해 를 참조하십시오.

5.6.10. 클러스터 배포

호환되는 클라우드 플랫폼에 OpenShift Container Platform을 설치할 수 있습니다.



중요

최초 설치 과정에서 설치 프로그램의 **create cluster** 명령을 한 번만 실행할 수 있습니다.

사전 요구 사항

- 클러스터를 호스팅하는 클라우드 플랫폼으로 계정을 구성합니다.
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

프로세스

1. 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터 배포를 초기화합니다.

```
$ ./openshift-install create cluster --dir <installation_directory> \ (1)
    --log-level=info (2)
```

1. **<installation_directory>** 값으로 사용자 지정한 **./install-config.yaml** 파일의 위치를 지정합니다.
2. 다른 설치 세부 사항을 보려면 **info** 대신 **warn**, **debug** 또는 **error**를 지정합니다.

참고

호스트에 구성된 클라우드 공급자 계정에 클러스터를 배포하기에 충분한 권한이 없는 경우, 설치 프로세스가 중단되고 누락된 권한을 알리는 메시지가 표시됩니다.

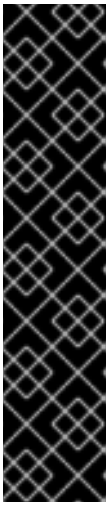
클러스터 배포가 완료되면 웹 콘솔로 연결되는 링크와 **kubeadmin** 사용자의 인증 정보가 포함된 클러스터 액세스 지침이 사용자 터미널에 표시됩니다.

출력 예

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```

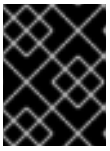
참고

설치에 성공하면 클러스터 액세스 및 인증 정보도 **<installation_directory>/.openshift_install.log**로 출력됩니다.



중요

- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.



중요

설치 프로그램에서 생성되는 파일이나 설치 프로그램을 삭제해서는 안 됩니다. 클러스터를 삭제하려면 두 가지가 모두 필요합니다.

5.6.11. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift CLI(**oc**)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 **oc**를 설치할 수 있습니다.

중요

이전 버전의 **oc**를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 **oc**를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 Linux에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Linux Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. **oc** 바이너리를 **PATH**에 있는 디렉터리에 배치합니다.
PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치

다음 절차에 따라 Windows에 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Windows Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. ZIP 프로그램으로 아카이브의 압축을 풉니다.
5. **oc** 바이너리를 **PATH**에 있는 디렉터리로 이동합니다.
PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치
다음 절차에 따라 macOS에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 MacOSX Client** 항목 옆에 있는 **지금 다운로드를** 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 해제하고 압축을 풉니다.
5. **oc** 바이너리 PATH의 디렉터리로 이동합니다.
   PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

5.6.12. CLI를 사용하여 클러스터에 로그인

클러스터 **kubeconfig** 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. **kubeconfig** 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- **oc** CLI를 설치했습니다.

프로세스

1. **kubeadmin** 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 **oc** 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

추가 리소스

- OpenShift Container Platform 웹 콘솔 액세스 및 이해에 대한 자세한 내용은 웹 콘솔에 액세스를 참조하십시오.

5.6.13. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

5.6.14. 다음 단계

- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.

5.7. AZURE의 클러스터를 기존 VNET에 설치

OpenShift Container Platform 버전 4.9에서는 Microsoft Azure의 기존 Azure Virtual Network(VNet)에 클러스터를 설치할 수 있습니다. 설치 프로그램이 나머지 필수 인프라를 프로비저닝하며, 이후에 추가로 사용자 지정할 수 있습니다. 설치를 사용자 지정하려면 클러스터를 설치하기 전에 **install-config.yaml** 파일에서 매개변수를 수정합니다.

5.7.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비 에 대한 문서를 읽습니다.
- 클러스터를 호스팅하고, 클러스터를 배포할 테스트 및 검증된 리전을 결정하도록 Azure 계정을 구성합니다.
- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성 해야 합니다.
- 사용자의 환경에서 클라우드 ID 및 액세스 관리(IAM) API에 액세스할 수 없거나 **kube-system** 네임스페이스에 관리자 수준의 인증 정보 시크릿을 저장하지 않으려면 IAM 인증 정보 수동 생성 및 유지 관리를 참조하십시오.

5.7.2. OpenShift Container Platform 클러스터에서 VNet의 재사용 정보

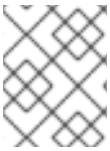
OpenShift Container Platform 4.9에서는 Microsoft Azure의 기존 Azure Virtual Network(VNet)에 클러스터를 배포할 수 있습니다. 이때 VNet 및 라우팅 규칙 내의 기존 서브넷도 사용해야 합니다.

기존 Azure VNet에 OpenShift Container Platform을 배포하면 새 계정의 서비스 한도 제한을 회피하거나 회사의 지침에 따른 운영 제한을 따르기 수월해집니다. 이 방법은 VNet을 직접 생성하는 데 필요한 인프라 생성 권한을 받을 수 없을 때 사용하기 좋은 대안입니다.

5.7.2.1. VNet 사용 요구사항

기존 VNet을 사용하여 클러스터를 배포하는 경우 클러스터를 설치하기 전에 추가 네트워크 구성을 수행해야 합니다. 설치 관리자 프로비저닝 인프라 클러스터에서, 설치 관리자는 일반적으로 다음 구성 요소를 생성하지만 기존 VNet에 설치할 때는 생성하지 않습니다.

- 서브넷
- 라우팅 테이블
- VNet
- 네트워크 보안 그룹



참고

설치 프로그램을 사용하려면 클라우드 제공 DNS 서버를 사용해야 합니다. 사용자 지정 DNS 서버 사용은 지원되지 않으며 이로 인해 설치에 실패합니다.

사용자 지정 VNet을 사용하는 경우, 사용할 클러스터와 설치 프로그램에 맞게 VNet과 해당 서브넷을 구성해야 합니다. 설치 프로그램에서 사용할 클러스터의 네트워크 범위를 세분하거나 서브넷에 대한 라우팅 테이블을 설정하거나 DHCP와 같은 VNet 옵션을 설정할 수 없으므로 클러스터를 설치하기 전에 직접 해당 작업을 수행해야 합니다.

클러스터는 기존 VNet과 서브넷이 포함된 리소스 그룹에 액세스할 수 있어야 합니다. 클러스터가 생성하는 모든 리소스는 생성된 별도의 리소스 그룹에 배치되지만 일부 네트워크 리소스는 별도의 그룹으로부터 사용됩니다. 일부 클러스터 Operator는 두 리소스 그룹 모두의 리소스에 액세스할 수 있어야 합니다. 예를 들어 Machine API 컨트롤러는 생성하는 가상 머신에 대한 NICS를 네트워킹 리소스 그룹의 서브넷에 연결합니다.

VNet은 다음 특성을 충족해야 합니다.

- VNet의 CIDR 블록에는 클러스터 시스템의 IP 주소 풀인 **Networking.MachineCIDR** 범위가 포함되어야 합니다.
- VNet과 해당 서브넷은 동일한 리소스 그룹에 속해야 하며 서브넷은 고정 IP 주소 대신 Azure 할당 DHCP IP 주소를 사용하도록 구성해야 합니다.

VNet 내에 두 서브넷을 제공해야 합니다. 하나는 컨트롤 플레인 시스템용이고 다른 하나는 컴퓨팅 시스템용입니다. Azure는 지정한 리전 내의 다른 가용성 영역에 시스템을 배포하므로 클러스터는 기본적으로 고가용성을 갖습니다.

제공한 서브넷이 적합한지 확인하기 위해 설치 프로그램이 다음 데이터를 확인합니다.

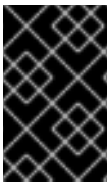
- 지정된 서브넷이 모두 존재합니다.
- 두 개의 사설 서브넷(컨트롤 플레인 시스템용 및 컴퓨팅 시스템용)이 있습니다.
- 서브넷 CIDR이 사용자가 지정한 시스템 CIDR에 속합니다. 프라이빗 서브넷을 제공하지 않는 가용성 영역에서는 시스템이 프로비저닝되지 않습니다. 필요한 경우, 설치 프로그램은 컨트롤 플레인과 작업자 노드를 관리하는 공용 로드 밸런서를 만들고 Azure는 여기에 공용 IP 주소를 할당합니다.

참고

기존 VNet을 사용하는 클러스터를 제거해도 VNet은 삭제되지 않습니다.

5.7.2.1.1. 네트워크 보안 그룹 요구사항

컴퓨팅과 컨트롤 플레인 시스템을 호스팅하는 서브넷에 대한 네트워크 보안 그룹은 클러스터 통신이 올바른지 확인하기 위해 특정 액세스 권한이 필요합니다. 필요한 클러스터 통신 포트에 대한 액세스를 허용하는 규칙을 만들어야 합니다.

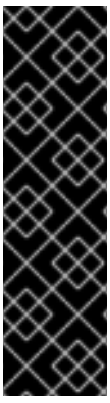


중요

클러스터를 설치하기 전에 네트워크 보안 그룹 규칙을 마련해야 합니다. 필요한 액세스 권한 없이 클러스터를 설치하려고 하면 설치 프로그램이 Azure API에 연결할 수 없으며 설치가 실패합니다.

표 5.17. 필수 포트

포트	설명	컨트롤 플레인	컴퓨팅
80	HTTP 트래픽 허용		x
443	HTTPS 트래픽 허용		x
6443	컨트롤 플레인 시스템과의 통신을 허용합니다.	x	
22623	머신을 프로비저닝하기 위해 머신 구성 서버와의 내부 통신 허용	x	



중요

현재는 머신 구성 서버 끝점을 차단하거나 제한할 수 있는 방법이 없습니다. 기존 구성 또는 상태가 없는 새로 프로비저닝된 머신이 구성을 가져올 수 있도록 머신 구성 서버를 네트워크에 노출해야 합니다. 이 모델에서 신뢰의 루트는 CSR(인증서 서명 요청) 끝점으로, kubelet이 클러스터에 가입하기 위해 승인하기 위해 인증서 서명 요청을 보내는 위치입니다. 이로 인해 시크릿 및 인증서와 같은 중요한 정보를 배포하는 데 머신 구성을 사용해서는 안 됩니다.

머신 구성 서버 끝점, 포트 22623 및 22624가 베어 메탈 시나리오에서 보호되도록 하려면 고객이 적절한 네트워크 정책을 구성해야 합니다.

클러스터 구성 요소는 Kubernetes 컨트롤러가 업데이트하는 사용자 제공 네트워크 보안 그룹을 변경하지 않으므로 나머지 환경에 영향을 주지 않고 Kubernetes 컨트롤러에 대한 의사 네트워크 보안 그룹이 생성됩니다.

추가 리소스

- OpenShift SDN 네트워크 플러그인 정보

5.7.2.2. 권한 분할

OpenShift Container Platform 4.3부터 클러스터를 배포하는 데 설치 프로그램에서 프로비저닝한 인프라

클러스터에 필요한 권한 중 일부가 필요하지 않게 되었습니다. 이러한 변경은 회사에서 보유할 수 있는 권한 분할을 모방합니다. 즉, 몇몇 사람이 다른 사람들과 다른 리소스를 클라우드에 생성할 수 있습니다. 예를 들어 인스턴스, 스토리지, 로드 밸런서와 같은 애플리케이션 관련 항목을 생성할 수는 있지만 VNet, 서브넷 또는 인그레스 규칙과 같은 네트워킹 관련 구성 요소는 생성할 수 없습니다.

클러스터를 생성할 때 사용하는 Azure 자격 증명에는 서브넷, 라우팅 테이블, 인터넷 게이트웨이, NAT, VPN과 같은 VNet 내 핵심 네트워킹 구성 요소와 VNet을 생성하는 데 필요한 네트워킹 권한이 필요하지 않습니다. 하지만 로드 밸런서, 보안 그룹, 스토리지 계정 및 노드와 같이 클러스터 내 시스템에 필요한 애플리케이션 리소스를 생성하려면 여전히 권한이 필요합니다.

5.7.2.3. 클러스터 간 격리

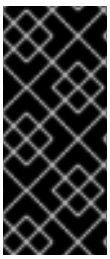
클러스터는 기존 서브넷의 네트워크 보안 그룹을 수정할 수 없기 때문에 VNet에서 클러스터를 서로 격리할 방법이 없습니다.

5.7.3. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.



중요

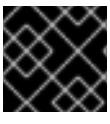
클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

5.7.4. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 **core** 사용자의 **~/.ssh/authorized_keys** 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 **core**로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. **./openshift-install gather** 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.



중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

참고

AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: **~/.ssh/id_ed25519**)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 **'~/.ssh** 디렉터리에 있는지 확인하십시오.

참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 **ed25519** 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 **rsa** 또는 **ecdsa** 알고리즘을 사용하는 키를 생성합니다.

2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 **~/.ssh/id_ed25519.pub** 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 **./openshift-install gather** 명령을 사용하려는 경우 필요합니다.

참고

일부 배포에서는 **~/.ssh/id_rsa** 및 **~/.ssh/id_dsa**와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. **ssh-agent** 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. **ssh-agent**에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: **~/.ssh/id_ed25519**).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

5.7.5. 설치 프로그램 받기

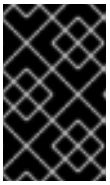
OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

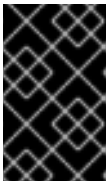
프로세스

1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.
2. 인프라 공급자를 선택합니다.
3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.



중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.



중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 OpenShift Container Platform 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

5.7.6. 설치 구성 파일 만들기

Microsoft Azure에 설치하는 OpenShift Container Platform 클러스터를 사용자 지정할 수 있습니다.

사전 요구 사항

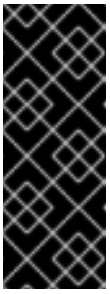
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.
- 서브스크립션 수준에서 서비스 주체 권한을 획득합니다.

절차

1. **install-config.yaml** 파일을 생성합니다.
   a. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

   1 **<installation_directory>**는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.



중요

비어 있는 디렉터리를 지정합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

   b. 화면에 나타나는 지시에 따라 클라우드에 대한 구성 세부 사항을 입력합니다.
      i. 선택사항: 클러스터 시스템에 액세스하는 데 사용할 SSH 키를 선택합니다.

참고

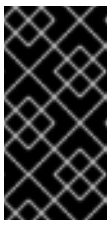
설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

      ii. 대상 플랫폼으로 **azure**를 선택합니다.
      iii. 컴퓨터에 Microsoft Azure 프로필이 저장되어 있지 않은 경우 서비스크립션 및 서비스 주체에 대해 다음 Azure 매개변수 값을 지정합니다.
         - **azure subscription id** 클러스터에 사용할 서브스크립션 ID입니다. 계정 출력의 **id** 값을 지정하십시오.
         - **azure tenant id** 테넌트 ID. 계정 출력의 **tenantId** 값을 지정하십시오.
         - **azure service principal client id** 서비스 주체의 **appId** 매개변수 값.
         - **azure service principal client secret** 서비스 주체의 **password** 매개변수 값.

iv. 클러스터를 배포할 리전을 선택합니다.

v. 클러스터를 배포할 기본 도메인을 선택합니다. 기본 도메인은 클러스터용으로 생성한 Azure DNS 영역에 해당합니다.

vi. 클러스터를 설명할 수 있는 이름을 입력합니다.



중요

공용 끝점을 통해 사용할 수 있는 모든 Azure 리소스에는 리소스 이름 제한 사항이 적용되며, 특정 용어를 사용하는 리소스를 생성할 수 없습니다. Azure가 제한하는 용어 목록은 Azure 문서의 예약된 리소스 이름 오류 해결을 참조하십시오.

vii. Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 붙여넣습니다.

2. **install-config.yaml** 파일을 수정합니다. 사용 가능한 매개변수에 대한 자세한 정보는 "설치 구성 매개변수" 섹션에서 확인할 수 있습니다.

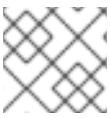
3. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.

중요

install-config.yaml 파일은 설치 과정에서 사용됩니다. 이 파일을 재사용하려면 지금 백업해야 합니다.

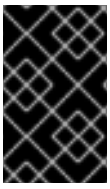
5.7.6.1. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 매개변수 값을 제공하여 클러스터를 호스팅할 클라우드 플랫폼에서 사용자 계정을 설명하고 선택사항으로 클러스터의 플랫폼을 사용자 지정합니다. **install-config.yaml** 설치 구성 파일을 생성할 때 명령줄을 통해 필요한 매개변수 값을 제공합니다. 클러스터를 사용자 지정하면 **install-config.yaml** 파일을 수정하여 플랫폼에 대한 세부 정보를 제공할 수 있습니다.



참고

설치한 후에는 **install-config.yaml** 파일에서 이러한 매개변수를 수정할 수 없습니다.



중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

5.7.6.1.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 5.18. 필수 매개 변수

매개변수	설명	값

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	소문자, 하이픈(**-**), 마침표(**.**)로 구성되는 문자열(예: **dev**)입니다.
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

매개변수	설명	값

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

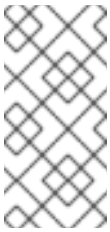
5.7.6.1.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 5.19. 네트워크 매개변수

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체 **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.

매개변수	설명	값
networking.clusterNetwork	Pod의 IP 주소 블록입니다. 기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된 **cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16**  **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

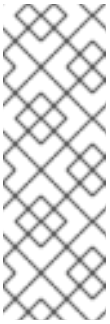
5.7.6.1.3. 선택적 구성 매개변수

선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 5.20. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은**3**입니다.

매개변수	설명	값
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 인터넷에서 액세스할 수 없는 사설 클러스터를 배포하려면 **publish**를 **Internal**로 설정합니다. 기본값은 **External**입니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

5.7.6.1.4. 추가 Azure 구성 매개 변수

추가 Azure 구성 매개변수는 다음 표에 설명되어 있습니다.

표 5.21. 추가 Azure 매개변수

매개변수	설명	값
compute.platform.azure.osDisk.diskSizeGB	VM의 Azure 디스크 크기입니다.	디스크 크기(GB)를 나타내는 정수입니다. 기본값은 **128** 입니다.
compute.platform.azure.osDisk.diskType	디스크 유형을 정의합니다.	**standard_LRS**, **premium_LRS**, or **standardSSD_LRS**. 기본값은 **premium_LRS** 입니다.
controlPlane.platform.azure.osDisk.diskSizeGB	VM의 Azure 디스크 크기입니다.	디스크 크기(GB)를 나타내는 정수입니다. 기본값은 **1024** 입니다.
controlPlane.platform.azure.osDisk.diskType	디스크 유형을 정의합니다.	**Premium_LRS** 또는 **표준SSD_LRS**. 기본값은 **premium_LRS** 입니다.
platform.azure.baseDomainResourceGroupName	기본 도메인의 DNS 영역을 포함하는 리소스 그룹의 이름입니다.	문자열(예: **production_cluster**).

매개변수	설명	값
platform.azure.resourceGroupName	클러스터를 설치할 기존 리소스 그룹의 이름입니다. 이 리소스 그룹은 비어 있어야 하며 특정 클러스터에만 사용해야 합니다. 클러스터 구성 요소는 리소스 그룹의 모든 리소스에 대한 소유권을 가정합니다. 설치 프로그램의 서비스 주체 범위를 이 리소스 그룹으로 제한하는 경우 해당 환경에서 설치 프로그램에서 사용하는 기타 모든 리소스에 퍼블릭 DNS 영역 및 가상 네트워크와 같은 필수 권한이 있는지 확인해야 합니다. 설치 프로그램을 사용하여 클러스터를 삭제하면 이 리소스 그룹이 삭제됩니다.	문자열(예: **existing_resource_group**)
platform.azure.outboundType	클러스터를 인터넷에 연결하는 데 사용되는 아웃 바운드 라우팅 전략입니다. 사용자 정의 라우팅을 사용하는 경우 클러스터를 설치하기 전에 아웃 바운드 라우팅이 이미 구성된 경우 기존 네트워킹을 사용할 수 있어야 합니다. 설치 프로그램은 사용자 정의 라우팅 설정을 하지 않습니다.	**LoadBalancer** 또는 **UserDefinedRouting**. 기본값은 **LoadBalancer** 입니다.
platform.azure.region	클러스터를 호스팅하는 Azure 리전의 이름입니다.	유효한 리전 이름(예: **centralus**).
platform.azure.zone	시스템을 배치하려는 가용성 영역 목록. 고가용성을 위해 최소한 두 개의 영역을 지정하십시오.	영역 목록(예: **["1", "2", "3"]**)
platform.azure.networkResourceGroupName	클러스터를 배포하려는 기존 VNet이 포함된 리소스 그룹의 이름입니다. 이 이름은 **platform.azure.baseDomainResourceGroupName**과 같을 수 없습니다.	문자열.
platform.azure.virtualNetwork	클러스터를 배포할 기존 VNet의 이름입니다.	문자열.
platform.azure.controlPlaneSubnet	컨트롤 플레인 시스템을 배포할 VNet의 기존 서브넷 이름입니다.	유효한 CIDR(예: **10.0.0.0/16**)
platform.azure.computeSubnet	컴퓨팅 시스템을 배포할 VNet의 기존 서브넷 이름입니다.	유효한 CIDR(예: **10.0.0.0/16**)
platform.azure.cloudName	적절한 Azure API 엔드포인트에서 Azure SDK를 구성하는데 사용되는 Azure 클라우드 환경의 이름입니다. 비어있는 경우 기본값 **AzurePublicCloud**가 사용됩니다.	**AzurePublicCloud** 또는 **AzureUSGovernmentCloud** 와 같은 유효한 클라우드 환경.

참고

Azure 가용성 영역을 사용자 지정하거나 Azure 클러스터와 함께 태그를 사용하여 Azure 리소스를 구성할 수 없습니다.

5.7.6.2. 클러스터 설치를 위한 최소 리소스 요구 사항

각 클러스터 시스템이 다음과 같은 최소 요구사항을 충족해야 합니다.

표 5.22. 최소 리소스 요구사항

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
부트스트랩	RHCOS	4	16GB	100GB	300
컨트롤 플레인	RHCOS	4	16GB	100GB	300
컴퓨팅	RHCOS, RHEL 7.9 또는 RHEL 8.4 [3]	2	8GB	100GB	300

1. SMT(동시 멀티스레딩) 또는 하이퍼 스레딩이 활성화되지 않은 경우 하나의 vCPU는 하나의 물리적 코어와 동일합니다. 활성화하면 다음과 같은 공식을 사용하여 해당 비율을 계산합니다. (코어당 스레드 수 × 코어 수) × 소켓 수 = vCPU 수

2. OpenShift Container Platform 및 Kubernetes는 디스크 성능에 민감하며 특히 10ms p99 fsync 기간이 필요한 컨트롤 플레인 노드의 etcd에 더 빠른 스토리지가 권장됩니다. 많은 클라우드 플랫폼에서 스토리지 크기와 IOPS를 함께 확장되므로 충분한 성능을 얻으려면 스토리지 볼륨을 과도하게 할당해야 할 수 있습니다.

3. 사용자가 프로비저닝한 모든 설치와 마찬가지로 클러스터에서 RHEL 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. RHEL 7 컴퓨팅 머신 사용은 더 이상 사용되지 않으며 향후 OpenShift Container Platform 4 릴리스에서 제거될 예정입니다.

중요

Premium **IO** 가 **true** 로 설정된 Azure 가상 머신을 사용해야 합니다. 머신에는 **하이퍼 VGeneration** 속성에 **V1** 이 포함되어 있어야 합니다.

5.7.6.3. Azure용 샘플 사용자 지정 install-config.yaml 파일

install-config.yaml 파일을 사용자 지정하여 OpenShift Container Platform 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.

중요

이 샘플 YAML 파일은 참조용으로만 제공됩니다. 설치 프로그램을 사용하여 **install-config.yaml** 파일을 받아서 수정해야 합니다.

```
apiVersion: v1
baseDomain: example.com (1)
```

```
controlPlane: 2
  hyperthreading: Enabled 3 4
  name: master
  platform:
    azure:
      osDisk:
        diskSizeGB: 1024 5
        diskType: Premium_LRS
      type: Standard_D8s_v3
  replicas: 3
compute: 6
- hyperthreading: Enabled 7
  name: worker
  platform:
    azure:
      type: Standard_D2s_v3
      osDisk:
        diskSizeGB: 512 8
        diskType: Standard_LRS
      zones: 9
      - "1"
      - "2"
      - "3"
  replicas: 5
metadata:
  name: test-cluster 10
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  azure:
    baseDomainResourceGroupName: resource_group 11
    region: centralus 12
    resourceGroupName: existing_resource_group 13
    networkResourceGroupName: vnet_resource_group 14
    virtualNetwork: vnet 15
    controlPlaneSubnet: control_plane_subnet 16
    computeSubnet: compute_subnet 17
    outboundType: Loadbalancer
    cloudName: AzurePublicCloud
pullSecret: '{"auths": ...}' 18
fips: false 19
sshKey: ssh-ed25519 AAAA... 20
```

1 10 12 18 필수 항목입니다. 설치 프로그램에서 이 값을 입력하라는 메시지를 표시합니다.

2 6 이러한 매개변수와 값을 지정하지 않으면 설치 프로그램은 기본값을 적용합니다.

3 7 **controlPlane** 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며

4 동시 멀티스레딩 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. 일부 클러스터 시스템에서 동시 멀티스레딩을 비활성화할 경우에는 해당 멀티스레딩을 모든 클러스터 시스템에서 비활성화해야 합니다.

중요

동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다. 동시 멀티스레딩을 비활성화하는 경우, 사용자 시스템에 더 큰 가상 머신 유형(예: **Standard_D8s_v3**)을 사용하십시오.

5 8 사용할 디스크 크기는 GB 단위로 지정할 수 있습니다. 컨트롤 플레인 노드의 최소 권장 크기는 1024GB입니다.

9 시스템을 배포할 영역 목록을 지정합니다. 고가용성을 위해 최소한 두 개의 영역을 지정하십시오.

11 기본 도메인의 DNS 영역을 포함하는 리소스 그룹의 이름을 지정합니다.

13 클러스터를 설치할 기존 리소스 그룹의 이름을 지정합니다. 정의되지 않은 경우 클러스터에 새 리소스 그룹이 생성됩니다.

14 기존 VNet을 사용하는 경우 이를 포함하는 리소스 그룹의 이름을 지정합니다.

15 기존 VNet을 사용하는 경우 해당 이름을 지정합니다.

16 기존 VNet을 사용하는 경우 컨트롤 플레인 시스템을 호스팅할 서브넷의 이름을 지정합니다.

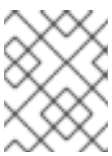
17 기존 VNet을 사용하는 경우 컴퓨팅 시스템을 호스팅할 서브넷의 이름을 지정합니다.

19 FIPS 모드 활성화 또는 비활성화 여부입니다. 기본적으로 FIPS 모드는 비활성화됩니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.

중요

FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.

20 선택사항으로, 클러스터의 시스템에 액세스하는 데 사용할 **sshKey** 값을 제공할 수도 있습니다.



참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

5.7.6.4. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 HTTP 또는 HTTPS 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 OpenShift Container Platform 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 API에 대한 호출을 포함하여 모든 클러스터 발신(Egress) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.

참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 Red Hat OpenStack Platform (RHOSP)에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. 클러스터 외부에서 HTTP 연결을 구축하는 데 사용할 프록시 URL입니다. URL 스키마는 **http**여야 합니다.
2. 클러스터 외부에서 HTTPS 연결을 구축하는 데 사용할 프록시 URL입니다.
3. 대상 도메인 이름, IP 주소 또는 프록시에서 제외할 기타 네트워크 CIDR로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. *를 사용하여 모든 대상에 대해 프록시를 바이패스합니다.
4. 이 값을 제공하면 설치 프로그램에서 추가 CA 인증서를 보유할 **openshift -config 네임스페이스에 user-ca- bundle** 이라는 구성 맵을 생성합니다. **additionalTrustBundle** 및 하나 이상의 프록시 설정을 제공하는 경우 **프록시** 오브젝트는 **trustedCA** 필드의 **user-ca-bundle** 구성 맵을 참조하도록 구성됩니다. 그러면 Cluster Network Operator에서 **trustedCA** 매개변수에 대해 지정된 콘텐츠를 RHCOS 신뢰 번들과 병합하는 **trusted-ca-bundle** 구성 맵을

생성합니다. 프록시의 ID 인증서를 RHCOS 트러스트 번들에 있는 기관에서 서명하지 않은 경우 **additionalTrustBundle** 필드가 있어야 합니다.

참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 OpenShift Container Platform을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 nil이 됩니다.

참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

5.7.7. 클러스터 배포

호환되는 클라우드 플랫폼에 OpenShift Container Platform을 설치할 수 있습니다.

중요

최초 설치 과정에서 설치 프로그램의 **create cluster** 명령을 한 번만 실행할 수 있습니다.

사전 요구 사항

- 클러스터를 호스팅하는 클라우드 플랫폼으로 계정을 구성합니다.
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

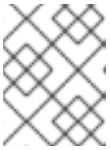
프로세스

1. 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터 배포를 초기화합니다.

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** 값으로 사용자 지정한 **./install-config.yaml** 파일의 위치를 지정합니다.

2 다른 설치 세부 사항을 보려면 **info** 대신 **warn**, **debug** 또는 **error**를 지정합니다.



참고

호스트에 구성된 클라우드 공급자 계정에 클러스터를 배포하기에 충분한 권한이 없는 경우, 설치 프로세스가 중단되고 누락된 권한을 알리는 메시지가 표시됩니다.

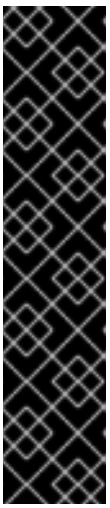
클러스터 배포가 완료되면 웹 콘솔로 연결되는 링크와 **kubeadmin** 사용자의 인증 정보가 포함된 클러스터 액세스 지침이 사용자 터미널에 표시됩니다.

출력 예

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```

참고

설치에 성공하면 클러스터 액세스 및 인증 정보도 **<installation_directory>/.openshift_install.log**로 출력됩니다.



중요

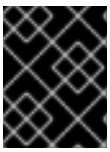
- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

중요

설치 프로그램에서 생성되는 파일이나 설치 프로그램을 삭제해서는 안 됩니다. 클러스터를 삭제하려면 두 가지가 모두 필요합니다.

5.7.8. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift CLI(**oc**)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 **oc**를 설치할 수 있습니다.



중요

이전 버전의 **oc**를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 **oc**를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 Linux에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.

2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Linux Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. **oc** 바이너리를 **PATH**에 있는 디렉터리에 배치합니다.
PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치
다음 절차에 따라 Windows에 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Windows Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. ZIP 프로그램으로 아카이브의 압축을 풉니다.
5. **oc** 바이너리를 **PATH**에 있는 디렉터리로 이동합니다.
PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치
다음 절차에 따라 macOS에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 MacOSX Client** 항목 옆에 있는 **지금 다운로드를** 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 해제하고 압축을 풉니다.
5. **oc** 바이너리 PATH의 디렉터리로 이동합니다.
PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

5.7.9. CLI를 사용하여 클러스터에 로그인

클러스터 **kubeconfig** 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. **kubeconfig** 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- **oc** CLI를 설치했습니다.

프로세스

1. **kubeadmin** 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 **oc** 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

추가 리소스

- OpenShift Container Platform 웹 콘솔 액세스 및 이해에 대한 자세한 내용은 웹 콘솔에 액세스를 참조하십시오.

5.7.10. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

5.7.11. 다음 단계

- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.

5.8. AZURE에 프라이빗 클러스터 설치

OpenShift Container Platform 버전 4.9에서는 Microsoft Azure의 기존 Azure Virtual Network(VNet)에 프라이빗 클러스터를 설치할 수 있습니다. 설치 프로그램이 나머지 필수 인프라를 프로비저닝하며, 이후에 추가로 사용자 지정할 수 있습니다. 설치를 사용자 지정하려면 클러스터를 설치하기 전에 **install-config.yaml** 파일에서 매개변수를 수정합니다.

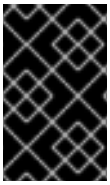
5.8.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비 에 대한 문서를 읽습니다.
- 클러스터를 호스팅하고, 클러스터를 배포할 테스트 및 검증된 리전을 결정하도록 Azure 계정을 구성합니다.
- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성 해야 합니다.
- 사용자의 환경에서 클라우드 ID 및 액세스 관리(IAM) API에 액세스할 수 없거나 **kube-system** 네임스페이스에 관리자 수준의 인증 정보 시크릿을 저장하지 않으려면 IAM 인증 정보 수동 생성 및 유지 관리를 참조하십시오.

5.8.2. 프라이빗 클러스터

외부 엔드 포인트를 노출하지 않는 비공개 OpenShift Container Platform 클러스터를 배포할 수 있습니다. 프라이빗 클러스터는 내부 네트워크에서만 액세스할 수 있으며 인터넷에 표시되지 않습니다.

기본적으로 OpenShift Container Platform은 공개적으로 액세스 가능한 DNS 및 끝점을 사용하여 프로비저닝됩니다. 따라서 개인 클러스터를 배포할 때 클러스터에서 DNS, Ingress Controller 및 API 서버를 비공개로 설정할 수 있습니다. 즉 클러스터 리소스는 내부 네트워크에서만 액세스할 수 있고 인터넷에는 노출되지 않습니다.



중요

클러스터에 퍼블릭 서브넷이 있는 경우 관리자가 생성한 로드 밸런서 서비스에 공개적으로 액세스할 수 있습니다. 클러스터 보안을 보장하기 위해 이러한 서비스에 명시적으로 주석이 지정되었는지 확인합니다.

프라이빗 클러스터를 배포하려면 다음을 수행해야 합니다.

- 요구 사항을 충족하는 기존 네트워킹을 사용합니다. 네트워크의 다른 클러스터 사이에 클러스터 리소스를 공유할 수 있습니다.

- 다음에 액세스할 수 있는 머신에서 배포합니다.
  - 프로비저닝하는 클라우드용 API 서비스
  - 프로비저닝하는 네트워크의 호스트
  - 설치 미디어를 가져올 인터넷

이러한 액세스 요구사항을 충족하고 회사의 지침을 따르는 모든 시스템을 사용할 수 있습니다. 클라우드 네트워크의 배스천 호스트 또는 VPN을 통해 네트워크에 액세스할 수 있는 시스템 등을 예로 들 수 있습니다.

5.8.2.1. Azure의 프라이빗 클러스터

Microsoft Azure에서 개인 클러스터를 만들려면 클러스터를 호스팅할 기존 개인 VNet 및 서브넷을 제공해야 합니다. 또한 설치 프로그램에서 클러스터에 필요한 DNS 레코드를 확인할 수 있어야 합니다. 설치 프로그램은 내부 트래픽용 Ingress Operator 및 API 서버를 구성합니다.

네트워크가 개인 VNET에 연결하는 방법에 따라 클러스터의 프라이빗 DNS 레코드를 확인하기 위해 DNS 전달자를 사용해야 할 수도 있습니다. 클러스터의 시스템은 DNS 확인을 위해 내부적으로 **168.63.129.16**을 사용합니다. 자세한 내용은 Azure 문서의 Azure 프라이빗 DNS란 무엇인가와 IP 주소 168.63.129.16은 무엇인가?를 참조하십시오.

클러스터가 Azure API에 액세스하려면 여전히 인터넷 접속이 필요합니다.

다음은 프라이빗 클러스터를 설치할 때 필요하지 않거나 생성되지 않는 항목들입니다.

- **BaseDomainResourceGroup**(클러스터가 공개 레코드를 생성하지 않으므로)
- 공용 IP 주소
- 공용 DNS 레코드
- 공용 끝점

> The cluster is configured so that the Operators do not create public records for the cluster and all cluster machines are placed in the private subnets that you specify.

5.8.2.1.1. 제한

Azure의 프라이빗 클러스터에는 기존 VNet 사용과 관련된 제한 사항만 적용됩니다.

5.8.2.2. 사용자 정의 아웃 바운드 라우팅

OpenShift Container Platform에서 실행되는 클러스터의 아웃 바운드 라우팅을 선택하고 인터넷에 연결할 수 있습니다. 이를 통해 공용 IP 주소 및 공용 로드 밸런서의 생성을 생략할 수 있습니다.

클러스터를 설치하기 전에 **install-config.yaml** 파일에서 매개 변수를 수정하여 사용자 정의 라우팅을 구성할 수 있습니다. 클러스터를 설치할 때 아웃 바운드 라우팅을 사용하려면 기존 VNet이 필요합니다. 설치 프로그램은 이를 설정하지 않습니다.

사용자 정의 라우팅을 사용하도록 클러스터를 구성할 때 설치 프로그램은 다음 리소스를 생성하지 않습니다.

- 인터넷 액세스를 위한 아웃 바운드 규칙

- 공용 로드 밸런서의 공용 IP
- 아웃 바운드 요청을 위해 공용로드 밸런서에 클러스터 머신을 추가하기위한 Kubernetes 서비스 객체

사용자 정의 라우팅을 설정하기 전에 다음 항목을 사용할 수 있는지 확인해야합니다.

- 내부 레지스트리 미러를 사용하지 않는 경우 인터넷으로의 Egress는 컨테이너 이미지를 가져올 수 있습니다.
- 클러스터는 Azure API에 액세스할 수 있습니다.
- 다양한 허용 목록 엔드 포인트가 설정됩니다. *방화벽 설정* 섹션에서 이러한 엔드 포인트를 참조할 수 있습니다.

사용자 정의 라우팅을 사용하여 인터넷 액세스에 지원되는 기존의 몇 가지 네트워킹 설정이 있습니다.

네트워크 주소 변환을 사용하는 개인 클러스터
Azure VNET NAT (네트워크 주소 변환) 를 사용하여 클러스터의 서브넷에 대한 아웃 바운드 인터넷 액세스를 제공할 수 있습니다. 설정 지침은 Azure 설명서의 Create a NAT gateway using Azure CLI 에서 참조하십시오.

Azure NAT 및 사용자 정의 라우팅이 구성된 VNet 설정을 사용하는 경우 공용 엔드포인트없이 개인 클러스터를 만들 수 있습니다.

Azure 방화벽이 있는 개인 클러스터
Azure 방화벽을 사용하여 클러스터를 설치하는 데 사용되는 VNet의 아웃 바운드 라우팅을 제공할 수 있습니다. Azure 설명서에서 Azure Firewall을 사용하여 사용자 정의 라우팅을 제공하는 방법 에 대해 자세히 알아볼 수 있습니다.

Azure 방화벽 및 사용자 정의 라우팅이 구성된 VNet 설정을 사용하는 경우 공용 엔드포인트없이 개인 클러스터를 만들 수 있습니다.

프록시 설정이 있는 개인 클러스터
사용자 정의 라우팅과 함께 프록시를 사용하여 인터넷으로의 송신 (Egress)을 허용할 수 있습니다. 클러스터 Operator가 프록시를 사용하여 Azure API에 액세스하지 않도록해야합니다. Operator는 프록시 외부에서 Azure API에 액세스할 수 있어야 합니다.

0.0.0.0/0이 Azure에 의해 자동으로 설정된 상태에서 서브넷의 기본 라우팅 테이블을 사용하는 경우 IP 주소가 공용인 경우에도 모든 Azure API 요청이 Azure의 내부 네트워크를 통해 라우팅됩니다. 네트워크 보안 그룹 규칙이 Azure API 엔드포인트으로의 송신을 허용하는 한 사용자 정의 라우팅이 구성된 프록시를 사용하면 공용 엔드포인트없이 개인 클러스터를 만들 수 있습니다.

인터넷 액세스가 없는 개인 클러스터
Azure API를 제외한 인터넷에 대한 모든 액세스를 제한하는 개인 네트워크를 설치할 수 있습니다. 이는 릴리스 이미지 레지스트리를 로컬로 미러링하여 수행됩니다. 클러스터는 다음에 액세스할 수 있어야 합니다.

- 컨테이너 이미지를 가져올 수있는 내부 레지스트리 미러
- Azure API에 액세스

이러한 요구 사항을 사용할 수 있으면 사용자 정의 라우팅을 사용하여 공용 엔드 포인트가없는 개인 클러스터를 만들 수 있습니다.

5.8.3. OpenShift Container Platform 클러스터에서 VNet의 재사용 정보

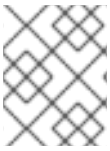
OpenShift Container Platform 4.9에서는 Microsoft Azure의 기존 Azure Virtual Network(VNet)에 클러스터를 배포할 수 있습니다. 이때 VNet 및 라우팅 규칙 내의 기존 서브넷도 사용해야 합니다.

기존 Azure VNet에 OpenShift Container Platform을 배포하면 새 계정의 서비스 한도 제한을 회피하거나 회사의 지침에 따른 운영 제한을 따르기 수월해집니다. 이 방법은 VNet을 직접 생성하는 데 필요한 인프라 생성 권한을 받을 수 없을 때 사용하기 좋은 대안입니다.

5.8.3.1. VNet 사용 요구사항

기존 VNet을 사용하여 클러스터를 배포하는 경우 클러스터를 설치하기 전에 추가 네트워크 구성을 수행해야 합니다. 설치 관리자 프로비저닝 인프라 클러스터에서, 설치 관리자는 일반적으로 다음 구성 요소를 생성하지만 기존 VNet에 설치할 때는 생성하지 않습니다.

- 서브넷
- 라우팅 테이블
- VNet
- 네트워크 보안 그룹



참고

설치 프로그램을 사용하려면 클라우드 제공 DNS 서버를 사용해야 합니다. 사용자 지정 DNS 서버 사용은 지원되지 않으며 이로 인해 설치에 실패합니다.

사용자 지정 VNet을 사용하는 경우, 사용할 클러스터와 설치 프로그램에 맞게 VNet과 해당 서브넷을 구성해야 합니다. 설치 프로그램에서 사용할 클러스터의 네트워크 범위를 세분하거나 서브넷에 대한 라우팅 테이블을 설정하거나 DHCP와 같은 VNet 옵션을 설정할 수 없으므로 클러스터를 설치하기 전에 직접 해당 작업을 수행해야 합니다.

클러스터는 기존 VNet과 서브넷이 포함된 리소스 그룹에 액세스할 수 있어야 합니다. 클러스터가 생성하는 모든 리소스는 생성된 별도의 리소스 그룹에 배치되지만 일부 네트워크 리소스는 별도의 그룹으로부터 사용됩니다. 일부 클러스터 Operator는 두 리소스 그룹 모두의 리소스에 액세스할 수 있어야 합니다. 예를 들어 Machine API 컨트롤러는 생성하는 가상 머신에 대한 NICS를 네트워킹 리소스 그룹의 서브넷에 연결합니다.

VNet은 다음 특성을 충족해야 합니다.

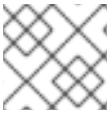
- VNet의 CIDR 블록에는 클러스터 시스템의 IP 주소 풀인 **Networking.MachineCIDR** 범위가 포함되어야 합니다.
- VNet과 해당 서브넷은 동일한 리소스 그룹에 속해야 하며 서브넷은 고정 IP 주소 대신 Azure 할당 DHCP IP 주소를 사용하도록 구성해야 합니다.

VNet 내에 두 서브넷을 제공해야 합니다. 하나는 컨트롤 플레인 시스템용이고 다른 하나는 컴퓨팅 시스템용입니다. Azure는 지정한 리전 내의 다른 가용성 영역에 시스템을 배포하므로 클러스터는 기본적으로 고가용성을 갖습니다.

제공한 서브넷이 적합한지 확인하기 위해 설치 프로그램이 다음 데이터를 확인합니다.

- 지정된 서브넷이 모두 존재합니다.
- 두 개의 사설 서브넷(컨트롤 플레인 시스템용 및 컴퓨팅 시스템용)이 있습니다.

- 서브넷 CIDR이 사용자가 지정한 시스템 CIDR에 속합니다. 프라이빗 서브넷을 제공하지 않는 가용성 영역에서는 시스템이 프로비저닝되지 않습니다.

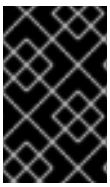


참고

기존 VNet을 사용하는 클러스터를 제거해도 VNet은 삭제되지 않습니다.

5.8.3.1.1. 네트워크 보안 그룹 요구사항

컴퓨팅과 컨트롤 플레인 시스템을 호스팅하는 서브넷에 대한 네트워크 보안 그룹은 클러스터 통신이 올바른지 확인하기 위해 특정 액세스 권한이 필요합니다. 필요한 클러스터 통신 포트에 대한 액세스를 허용하는 규칙을 만들어야 합니다.

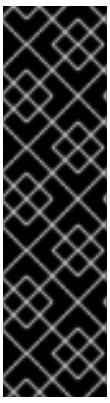


중요

클러스터를 설치하기 전에 네트워크 보안 그룹 규칙을 마련해야 합니다. 필요한 액세스 권한 없이 클러스터를 설치하려고 하면 설치 프로그램이 Azure API에 연결할 수 없으며 설치가 실패합니다.

표 5.23. 필수 포트

포트	설명	컨트롤 플레인	컴퓨팅
80	HTTP 트래픽 허용		x
443	HTTPS 트래픽 허용		x
6443	컨트롤 플레인 시스템과의 통신을 허용합니다.	x	
22623	머신을 프로비저닝하기 위해 머신 구성 서버와의 내부 통신 허용	x	



중요

현재는 머신 구성 서버 끝점을 차단하거나 제한할 수 있는 방법이 없습니다. 기존 구성 또는 상태가 없는 새로 프로비저닝된 머신이 구성을 가져올 수 있도록 머신 구성 서버를 네트워크에 노출해야 합니다. 이 모델에서 신뢰의 루트는 CSR(인증서 서명 요청) 끝점으로, kubelet이 클러스터에 가입하기 위해 승인하기 위해 인증서 서명 요청을 보내는 위치입니다. 이로 인해 시크릿 및 인증서와 같은 중요한 정보를 배포하는 데 머신 구성을 사용해서는 안 됩니다.

머신 구성 서버 끝점, 포트 22623 및 22624가 베어 메탈 시나리오에서 보호되도록 하려면 고객이 적절한 네트워크 정책을 구성해야 합니다.

클러스터 구성 요소는 Kubernetes 컨트롤러가 업데이트하는 사용자 제공 네트워크 보안 그룹을 변경하지 않으므로 나머지 환경에 영향을 주지 않고 Kubernetes 컨트롤러에 대한 의사 네트워크 보안 그룹이 생성됩니다.

추가 리소스

- OpenShift SDN 네트워크 플러그인 정보

5.8.3.2. 권한 분할

OpenShift Container Platform 4.3부터 클러스터를 배포하는 데 설치 프로그램에서 프로비저닝한 인프라 클러스터에 필요한 권한 중 일부가 필요하지 않게 되었습니다. 이러한 변경은 회사에서 보유할 수 있는 권한 분할을 모방합니다. 즉, 몇몇 사람이 다른 사람들과 다른 리소스를 클라우드에 생성할 수 있습니다. 예를 들어 인스턴스, 스토리지, 로드 밸런서와 같은 애플리케이션 관련 항목을 생성할 수는 있지만 VNet, 서브넷 또는 인그레스 규칙과 같은 네트워킹 관련 구성 요소는 생성할 수 없습니다.

클러스터를 생성할 때 사용하는 Azure 자격 증명에는 서브넷, 라우팅 테이블, 인터넷 게이트웨이, NAT, VPN과 같은 VNet 내 핵심 네트워킹 구성 요소와 VNet을 생성하는 데 필요한 네트워킹 권한이 필요하지 않습니다. 하지만 로드 밸런서, 보안 그룹, 스토리지 계정 및 노드와 같이 클러스터 내 시스템에 필요한 애플리케이션 리소스를 생성하려면 여전히 권한이 필요합니다.

5.8.3.3. 클러스터 간 격리

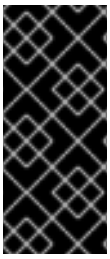
클러스터는 기존 서브넷의 네트워크 보안 그룹을 수정할 수 없기 때문에 VNet에서 클러스터를 서로 격리할 방법이 없습니다.

5.8.4. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.



중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

5.8.5. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 **core** 사용자의 **~/.ssh/authorized_keys** 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 **core**로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. **./openshift-install gather** 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

참고

AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: **~/.ssh/id_ed25519**)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 **'~/.ssh** 디렉터리에 있는지 확인하십시오.

참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 **ed25519** 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 **rsa** 또는 **ecdsa** 알고리즘을 사용하는 키를 생성합니다.

2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 **~/.ssh/id_ed25519.pub** 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 **./openshift-install gather** 명령을 사용하려는 경우 필요합니다.

참고

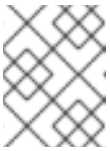
일부 배포에서는 **~/.ssh/id_rsa** 및 **~/.ssh/id_dsa**와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. **ssh-agent** 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```



참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. **ssh-agent**에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: **~/.ssh/id_ed25519**).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

5.8.6. 설치 프로그램 받기

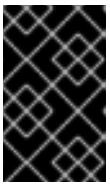
OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

프로세스

1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.
2. 인프라 공급자를 선택합니다.
3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.



중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.

중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 OpenShift Container Platform 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

5.8.7. 수동으로 설치 구성 파일 만들기

내부 네트워크에서만 액세스할 수 있고 인터넷에 표시되지 않는 프라이빗 OpenShift Container Platform 클러스터 설치의 경우 설치 구성 파일을 수동으로 생성해야 합니다.

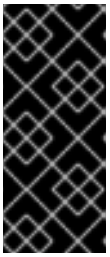
사전 요구 사항

- 로컬 시스템에 설치 프로그램에 제공할 SSH 공개 키가 있습니다. 키는 디버깅 및 재해 복구를 위해 클러스터 노드에 대한 SSH 인증에 사용됩니다.
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿이 있습니다.

프로세스

1. 필요한 설치 자산을 저장할 설치 디렉터리를 만듭니다.

```
$ mkdir <installation_directory>
```



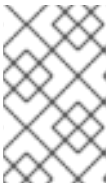
중요

디렉터리를 만들어야 합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

2. 샘플 **install-config.yaml** 파일 템플릿을 사용자 지정하여 **<installation_directory>**에 저장합니다.

참고

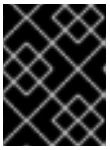
이 설정 파일의 이름을 **install-config.yaml**로 지정해야 합니다.



참고

일부 플랫폼 유형의 경우 대체로 **./openshift-install create install-config --dir <installation_directory>** 를 실행하여 **install-config.yaml** 파일을 생성할 수 있습니다. 프롬프트에서 클러스터 구성에 대한 세부 정보를 제공할 수 있습니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.



중요

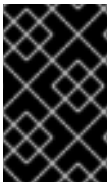
install-config.yaml 파일은 설치 과정의 다음 단계에서 사용됩니다. 이 시점에서 이를 백업해야 합니다.

5.8.7.1. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 매개변수 값을 제공하여 클러스터를 호스팅할 클라우드 플랫폼에서 사용자 계정을 설명하고 선택사항으로 클러스터의 플랫폼을 사용자 지정합니다. **install-config.yaml** 설치 구성 파일을 생성할 때 명령줄을 통해 필요한 매개변수 값을 제공합니다. 클러스터를 사용자 지정하면 **install-config.yaml** 파일을 수정하여 플랫폼에 대한 세부 정보를 제공할 수 있습니다.

참고

설치한 후에는 **install-config.yaml** 파일에서 이러한 매개변수를 수정할 수 없습니다.



중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

5.8.7.1.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 5.24. 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	소문자, 하이픈(**-**), 마침표(**.**)로 구성되는 문자열(예: **dev**)입니다.

매개변수	설명	값
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

5.8.7.1.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 5.25. 네트워크 매개변수

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체 **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.

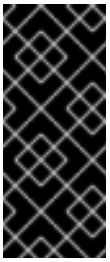
매개변수	설명	값
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.
networking.clusterNetwork	Pod의 IP 주소 블록입니다. 기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된**cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16

매개변수	설명	값
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16**  **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

5.8.7.1.3. 선택적 구성 매개변수

선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 5.26. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**

매개변수	설명	값
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은**3**입니다.
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 인터넷에서 액세스할 수 없는 사설 클러스터를 배포하려면 **publish**를 **Internal**로 설정합니다. 기본값은 **External**입니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

5.8.7.1.4. 추가 Azure 구성 매개 변수

추가 Azure 구성 매개변수는 다음 표에 설명되어 있습니다.

표 5.27. 추가 Azure 매개변수

매개변수	설명	값
compute.platform.azure.osDisk.diskSizeGB	VM의 Azure 디스크 크기입니다.	디스크 크기(GB)를 나타내는 정수입니다. 기본값은 **128** 입니다.
compute.platform.azure.osDisk.diskType	디스크 유형을 정의합니다.	**standard_LRS**, **premium_LRS**, or **standardSSD_LRS**. 기본값은 **premium_LRS** 입니다.
controlPlane.platform.azure.osDisk.diskSizeGB	VM의 Azure 디스크 크기입니다.	디스크 크기(GB)를 나타내는 정수입니다. 기본값은 **1024** 입니다.
controlPlane.platform.azure.osDisk.diskType	디스크 유형을 정의합니다.	**Premium_LRS** 또는 **표준SSD_LRS**. 기본값은 **premium_LRS** 입니다.
platform.azure.baseDomainResourceGroupName	기본 도메인의 DNS 영역을 포함하는 리소스 그룹의 이름입니다.	문자열(예: **production_cluster**).

매개변수	설명	값
platform.azure.resourceGroupName	클러스터를 설치할 기존 리소스 그룹의 이름입니다. 이 리소스 그룹은 비어 있어야 하며 특정 클러스터에만 사용해야 합니다. 클러스터 구성 요소는 리소스 그룹의 모든 리소스에 대한 소유권을 가정합니다. 설치 프로그램의 서비스 주체 범위를 이 리소스 그룹으로 제한하는 경우 해당 환경에서 설치 프로그램에서 사용하는 기타 모든 리소스에 퍼블릭 DNS 영역 및 가상 네트워크와 같은 필수 권한이 있는지 확인해야 합니다. 설치 프로그램을 사용하여 클러스터를 삭제하면 이 리소스 그룹이 삭제됩니다.	문자열(예: **existing_resource_group**)
platform.azure.outboundType	클러스터를 인터넷에 연결하는 데 사용되는 아웃 바운드 라우팅 전략입니다. 사용자 정의 라우팅을 사용하는 경우 클러스터를 설치하기 전에 아웃 바운드 라우팅이 이미 구성된 경우 기존 네트워킹을 사용할 수 있어야 합니다. 설치 프로그램은 사용자 정의 라우팅 설정을 하지 않습니다.	**LoadBalancer** 또는 **UserDefinedRouting**. 기본값은 **LoadBalancer** 입니다.
platform.azure.region	클러스터를 호스팅하는 Azure 리전의 이름입니다.	유효한 리전 이름(예: **centralus**).
platform.azure.zone	시스템을 배치하려는 가용성 영역 목록. 고가용성을 위해 최소한 두 개의 영역을 지정하십시오.	영역 목록(예: **["1", "2", "3"]**)
platform.azure.networkResourceGroupName	클러스터를 배포하려는 기존 VNet이 포함된 리소스 그룹의 이름입니다. 이 이름은 **platform.azure.baseDomainResourceGroupName**과 같을 수 없습니다.	문자열.
platform.azure.virtualNetwork	클러스터를 배포할 기존 VNet의 이름입니다.	문자열.
platform.azure.controlPlaneSubnet	컨트롤 플레인 시스템을 배포할 VNet의 기존 서브넷 이름입니다.	유효한 CIDR(예: **10.0.0.0/16**)
platform.azure.computeSubnet	컴퓨팅 시스템을 배포할 VNet의 기존 서브넷 이름입니다.	유효한 CIDR(예: **10.0.0.0/16**)

매개변수	설명	값
platform.azure.cloudName	적절한 Azure API 엔드포인트에서 Azure SDK를 구성하는데 사용되는 Azure 클라우드 환경의 이름입니다. 비어있는 경우 기본값 **AzurePublicCloud**가 사용됩니다.	**AzurePublicCloud** 또는 **AzureUSGovernmentCloud** 와 같은 유효한 클라우드 환경.

참고

Azure 가용성 영역을 사용자 지정하거나 Azure 클러스터와 함께 태그를 사용하여 Azure 리소스를 구성할 수 없습니다.

5.8.7.2. 클러스터 설치를 위한 최소 리소스 요구 사항

각 클러스터 시스템이 다음과 같은 최소 요구사항을 충족해야 합니다.

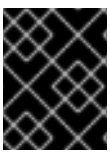
표 5.28. 최소 리소스 요구사항

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
부트스트랩	RHCOS	4	16GB	100GB	300
컨트롤 플레인	RHCOS	4	16GB	100GB	300
컴퓨팅	RHCOS, RHEL 7.9 또는 RHEL 8.4 [3]	2	8GB	100GB	300

1. SMT(동시 멀티스레딩) 또는 하이퍼 스레딩이 활성화되지 않은 경우 하나의 vCPU는 하나의 물리적 코어와 동일합니다. 활성화하면 다음과 같은 공식을 사용하여 해당 비율을 계산합니다. (코어당 스레드 수 × 코어 수) × 소켓 수 = vCPU 수

2. OpenShift Container Platform 및 Kubernetes는 디스크 성능에 민감하며 특히 10ms p99 fsync 기간이 필요한 컨트롤 플레인 노드의 etcd에 더 빠른 스토리지가 권장됩니다. 많은 클라우드 플랫폼에서 스토리지 크기와 IOPS를 함께 확장되므로 충분한 성능을 얻으려면 스토리지 볼륨을 과도하게 할당해야 할 수 있습니다.

3. 사용자가 프로비저닝한 모든 설치와 마찬가지로 클러스터에서 RHEL 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. RHEL 7 컴퓨팅 머신 사용은 더 이상 사용되지 않으며 향후 OpenShift Container Platform 4 릴리스에서 제거될 예정입니다.



중요

Premium **IO** 가 **true** 로 설정된 Azure 가상 머신을 사용해야 합니다. 머신에는 **하이퍼 VGeneration** 속성에 **V1** 이 포함되어 있어야 합니다.

5.8.7.3. Azure용 샘플 사용자 지정 install-config.yaml 파일

install-config.yaml 파일을 사용자 지정하여 OpenShift Container Platform 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.

중요

이 샘플 YAML 파일은 참조용으로만 제공됩니다. 설치 프로그램을 사용하여 **install-config.yaml** 파일을 받아서 수정해야 합니다.

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2)
  hyperthreading: Enabled (3) (4)
  name: master
  platform:
    azure:
      osDisk:
        diskSizeGB: 1024 (5)
        diskType: Premium_LRS
      type: Standard_D8s_v3
  replicas: 3
compute: (6)
- hyperthreading: Enabled (7)
  name: worker
  platform:
    azure:
      type: Standard_D2s_v3
      osDisk:
        diskSizeGB: 512 (8)
        diskType: Standard_LRS
      zones: (9)
      - "1"
      - "2"
      - "3"
  replicas: 5
metadata:
  name: test-cluster (10)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  azure:
    baseDomainResourceGroupName: resource_group (11)
    region: centralus (12)
    resourceGroupName: existing_resource_group (13)
    networkResourceGroupName: vnet_resource_group (14)
    virtualNetwork: vnet (15)
    controlPlaneSubnet: control_plane_subnet (16)
```

```
    computeSubnet: compute_subnet (17)
    outboundType: UserDefinedRouting (18)
    cloudName: AzurePublicCloud
pullSecret: '{"auths": ...}' (19)
fips: false (20)
sshKey: ssh-ed25519 AAAA... (21)
publish: Internal (22)
```

(1) (10) (12) (19) 필수 항목입니다. 설치 프로그램에서 이 값을 입력하라는 메시지를 표시합니다.

(2) (6) 이러한 매개변수와 값을 지정하지 않으면 설치 프로그램은 기본값을 적용합니다.

(3) (7) **controlPlane** 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

(4) 동시 멀티스레딩 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. 일부 클러스터 시스템에서 동시 멀티스레딩을 비활성화할 경우에는 해당 멀티스레딩을 모든 클러스터 시스템에서 비활성화해야 합니다.

중요

동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다. 동시 멀티스레딩을 비활성화하는 경우, 사용자 시스템에 더 큰 가상 머신 유형(예: **Standard_D8s_v3**)을 사용하십시오.

(5) (8) 사용할 디스크 크기는 GB 단위로 지정할 수 있습니다. 컨트롤 플레인 노드의 최소 권장 크기는 1024GB입니다.

(9) 시스템을 배포할 영역 목록을 지정합니다. 고가용성을 위해 최소한 두 개의 영역을 지정하십시오.

(11) 기본 도메인의 DNS 영역을 포함하는 리소스 그룹의 이름을 지정합니다.

(13) 클러스터를 설치할 기존 리소스 그룹의 이름을 지정합니다. 정의되지 않은 경우 클러스터에 새 리소스 그룹이 생성됩니다.

(14) 기존 VNet을 사용하는 경우 이를 포함하는 리소스 그룹의 이름을 지정합니다.

(15) 기존 VNet을 사용하는 경우 해당 이름을 지정합니다.

(16) 기존 VNet을 사용하는 경우 컨트롤 플레인 시스템을 호스팅할 서브넷의 이름을 지정합니다.

(17) 기존 VNet을 사용하는 경우 컴퓨팅 시스템을 호스팅할 서브넷의 이름을 지정합니다.

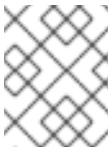
(18) 사용자의 아웃 바운드 라우팅을 사용자 지정할 수 있습니다. 사용자 정의 라우팅을 구성하면 클러스터에서 외부 엔드 포인트가 노출되지 않습니다. 송신에 대한 사용자 정의 라우팅을 사용하려면 기존 VNet에 클러스터를 배포해야 합니다.

(20) FIPS 모드 활성화 또는 비활성화 여부입니다. 기본적으로 FIPS 모드는 비활성화됩니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.

중요

FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.

21 선택사항으로, 클러스터의 시스템에 액세스하는 데 사용할 **sshKey** 값을 제공할 수도 있습니다.



참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

22 클러스터의 사용자 끝점을 게시하는 방법. 인터넷에서 액세스할 수 없는 프라이빗 클러스터를 배포하려면 **publish**를 **Internal**로 설정합니다. 기본값은 **External**입니다.

5.8.7.4. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 HTTP 또는 HTTPS 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 OpenShift Container Platform 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 API에 대한 호출을 포함하여 모든 클러스터 발신(Egress) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.

참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 Red Hat OpenStack Platform (RHOSP)에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
```

```
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. 클러스터 외부에서 HTTP 연결을 구축하는 데 사용할 프록시 URL입니다. URL 스키마는 **http**여야 합니다.
2. 클러스터 외부에서 HTTPS 연결을 구축하는 데 사용할 프록시 URL입니다.
3. 대상 도메인 이름, IP 주소 또는 프록시에서 제외할 기타 네트워크 CIDR로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. *를 사용하여 모든 대상에 대해 프록시를 바이패스합니다.
4. 이 값을 제공하면 설치 프로그램에서 추가 CA 인증서를 보유할 **openshift -config 네임스페이스에 user-ca- bundle** 이라는 구성 맵을 생성합니다. **additionalTrustBundle** 및 하나 이상의 프록시 설정을 제공하는 경우 **프록시** 오브젝트는 **trustedCA** 필드의 **user-ca-bundle** 구성 맵을 참조하도록 구성됩니다. 그러면 Cluster Network Operator에서 **trustedCA** 매개변수에 대해 지정된 콘텐츠를 RHCOS 신뢰 번들과 병합하는 **trusted-ca-bundle** 구성 맵을 생성합니다. 프록시의 ID 인증서를 RHCOS 트러스트 번들에 있는 기관에서 서명하지 않은 경우 **additionalTrustBundle** 필드가 있어야 합니다.

참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 OpenShift Container Platform을 설치할 때 참조하십시오.

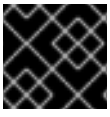
제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 nil이 됩니다.

참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

5.8.8. 클러스터 배포

호환되는 클라우드 플랫폼에 OpenShift Container Platform을 설치할 수 있습니다.



중요

최초 설치 과정에서 설치 프로그램의 **create cluster** 명령을 한 번만 실행할 수 있습니다.

사전 요구 사항

- 클러스터를 호스팅하는 클라우드 플랫폼으로 계정을 구성합니다.
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

절차

1. 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터 배포를 초기화합니다.

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>**는 다음을 지정합니다.

2 다른 설치 세부 사항을 보려면 **info** 대신 **warn**, **debug** 또는 **error**를 지정합니다.

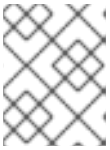
참고

호스트에 구성된 클라우드 공급자 계정에 클러스터를 배포하기에 충분한 권한이 없는 경우, 설치 프로세스가 중단되고 누락된 권한을 알리는 메시지가 표시됩니다.

클러스터 배포가 완료되면 웹 콘솔로 연결되는 링크와 **kubeadmin** 사용자의 인증 정보가 포함된 클러스터 액세스 지침이 사용자 터미널에 표시됩니다.

출력 예

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```



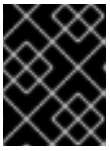
참고

설치에 성공하면 클러스터 액세스 및 인증 정보도 **<installation_directory>/.openshift_install.log**로 출력됩니다.

중요

- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.

- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.



중요

설치 프로그램에서 생성되는 파일이나 설치 프로그램을 삭제해서는 안 됩니다. 클러스터를 삭제하려면 두 가지가 모두 필요합니다.

5.8.9. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift CLI(**oc**)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 **oc**를 설치할 수 있습니다.

중요

이전 버전의 **oc**를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 **oc**를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치
다음 절차를 사용하여 Linux에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Linux Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. **oc** 바이너리를 **PATH**에 있는 디렉터리에 배치합니다.
PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치
다음 절차에 따라 Windows에 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Windows Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. ZIP 프로그램으로 아카이브의 압축을 풉니다.
5. **oc** 바이너리를 **PATH**에 있는 디렉터리로 이동합니다.
PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치
다음 절차에 따라 macOS에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 MacOSX Client** 항목 옆에 있는 **지금 다운로드를** 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 해제하고 압축을 풉니다.
5. **oc** 바이너리 PATH의 디렉터리로 이동합니다.
   PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

5.8.10. CLI를 사용하여 클러스터에 로그인

클러스터 **kubeconfig** 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. **kubeconfig** 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- **oc** CLI를 설치했습니다.

프로세스

1. **kubeadmin** 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 **oc** 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

-

출력 예

```
system:admin
```

추가 리소스

- OpenShift Container Platform 웹 콘솔 액세스 및 이해에 대한 자세한 내용은 웹 콘솔에 액세스를 참조하십시오.

5.8.11. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

5.8.12. 다음 단계

- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.

5.9. 자세한 내용은 정부 리전에 AZURE의 클러스터 설치를 참조하십시오.

OpenShift Container Platform 버전 4.9에서는 사용자가 제공하는 인프라를 사용하여 Microsoft Azure에 클러스터를 설치할 수 있습니다. 정부 리전을 설정하려면 클러스터를 설치하기 전에 **install-config.yaml** 파일에서 매개변수를 수정합니다.

5.9.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비 에 대한 문서를 읽습니다.
- 클러스터를 호스팅하고, 클러스터를 배포할 테스트 및 검증된 정부 리전을 결정하도록 Azure 계정을 구성합니다.
- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성 해야 합니다.
- 사용자의 환경에서 클라우드 ID 및 액세스 관리(IAM) API에 액세스할 수 없거나 **kube-system** 네임스페이스에 관리자 수준의 인증 정보 시크릿을 저장하지 않으려면 IAM 인증 정보 수동 생성 및 유지 관리를 참조하십시오.

5.9.2. 지원되는 Azure 정부 리전

OpenShift Container Platform은 Microsoft Azure Government (MAG) 지역에 클러스터 배포를 지원합니다. MAG는 연방, 주, 지역 수준의 미국 정부 기관은 물론 Azure에서 중요한 워크로드를 실행해야 하는 미국 정부 기관, 계약자, 교육 기관 및 기타 미국 고객을 위해 설계되었습니다. MAG는 정부 전용 데이터 센터 리전으로 구성되며 모두 영향 레벨 5 임시 승인 이 부여됩니다.

MAG 리전에 설치하려면 **install-config.yaml** 파일에서 Azure Government 전용 클라우드 인스턴스 및 리전을 수동으로 구성해야 합니다. 또한 적절한 정부 환경을 참조하도록 서비스 주체를 업데이트해야 합니다.

참고

설치 프로그램의 터미널 프롬프트를 사용하여 Azure 정부 리전을 선택할 수 없습니다. **install-config.yaml** 파일에서 리전을 수동으로 정의해야 합니다. 지정된 리전에 따라 **AzureUSGovernmentCloud** 와 같은 전용 클라우드 인스턴스도 설정해야 합니다.

5.9.3. 프라이빗 클러스터

외부 엔드 포인트를 노출하지 않는 비공개 OpenShift Container Platform 클러스터를 배포할 수 있습니다. 프라이빗 클러스터는 내부 네트워크에서만 액세스할 수 있으며 인터넷에 표시되지 않습니다.

기본적으로 OpenShift Container Platform은 공개적으로 액세스 가능한 DNS 및 끝점을 사용하여 프로비저닝됩니다. 따라서 개인 클러스터를 배포할 때 클러스터에서 DNS, Ingress Controller 및 API 서버를 비공개로 설정할 수 있습니다. 즉 클러스터 리소스는 내부 네트워크에서만 액세스할 수 있고 인터넷에는 노출되지 않습니다.

중요

클러스터에 퍼블릭 서브넷이 있는 경우 관리자가 생성한 로드 밸런서 서비스에 공개적으로 액세스할 수 있습니다. 클러스터 보안을 보장하기 위해 이러한 서비스에 명시적으로 주석이 지정되었는지 확인합니다.

프라이빗 클러스터를 배포하려면 다음을 수행해야 합니다.

- 요구 사항을 충족하는 기존 네트워킹을 사용합니다. 네트워크의 다른 클러스터 사이에 클러스터 리소스를 공유할 수 있습니다.
- 다음에 액세스할 수 있는 머신에서 배포합니다.
  - 프로비저닝하는 클라우드용 API 서비스
  - 프로비저닝하는 네트워크의 호스트
  - 설치 미디어를 가져올 인터넷

이러한 액세스 요구사항을 충족하고 회사의 지침을 따르는 모든 시스템을 사용할 수 있습니다. 클라우드 네트워크의 배스천 호스트 또는 VPN을 통해 네트워크에 액세스할 수 있는 시스템 등을 예로 들 수 있습니다.

5.9.3.1. Azure의 프라이빗 클러스터

Microsoft Azure에서 개인 클러스터를 만들려면 클러스터를 호스팅할 기존 개인 VNet 및 서브넷을 제공해야 합니다. 또한 설치 프로그램에서 클러스터에 필요한 DNS 레코드를 확인할 수 있어야 합니다. 설치 프로그램은 내부 트래픽용 Ingress Operator 및 API 서버를 구성합니다.

네트워크가 개인 VNET에 연결하는 방법에 따라 클러스터의 프라이빗 DNS 레코드를 확인하기 위해 DNS 전달자를 사용해야 할 수도 있습니다. 클러스터의 시스템은 DNS 확인을 위해 내부적으로 **168.63.129.16**을 사용합니다. 자세한 내용은 Azure 문서의 Azure 프라이빗 DNS란 무엇인가와 IP 주소 168.63.129.16은 무엇인가?를 참조하십시오.

클러스터가 Azure API에 액세스하려면 여전히 인터넷 접속이 필요합니다.

다음은 프라이빗 클러스터를 설치할 때 필요하지 않거나 생성되지 않는 항목들입니다.

- **BaseDomainResourceGroup**(클러스터가 공개 레코드를 생성하지 않으므로)
- 공용 IP 주소
- 공용 DNS 레코드
- 공용 끝점

> The cluster is configured so that the Operators do not create public records for the cluster and all cluster machines are placed in the private subnets that you specify.

5.9.3.1.1. 제한

Azure의 프라이빗 클러스터에는 기존 VNet 사용과 관련된 제한 사항만 적용됩니다.

5.9.3.2. 사용자 정의 아웃 바운드 라우팅

OpenShift Container Platform에서 실행되는 클러스터의 아웃 바운드 라우팅을 선택하고 인터넷에 연결할 수 있습니다. 이를 통해 공용 IP 주소 및 공용 로드 밸런서의 생성을 생략할 수 있습니다.

클러스터를 설치하기 전에 **install-config.yaml** 파일에서 매개 변수를 수정하여 사용자 정의 라우팅을 구성할 수 있습니다. 클러스터를 설치할 때 아웃 바운드 라우팅을 사용하려면 기존 VNet이 필요합니다. 설치 프로그램은 이를 설정하지 않습니다.

사용자 정의 라우팅을 사용하도록 클러스터를 구성할 때 설치 프로그램은 다음 리소스를 생성하지 않습니다.

- 인터넷 액세스를 위한 아웃 바운드 규칙
- 공용 로드 밸런서의 공용 IP
- 아웃 바운드 요청을 위해 공용로드 밸런서에 클러스터 머신을 추가하기위한 Kubernetes 서비스 객체

사용자 정의 라우팅을 설정하기 전에 다음 항목을 사용할 수 있는지 확인해야합니다.

- 내부 레지스트리 미러를 사용하지 않는 경우 인터넷으로의 Egress는 컨테이너 이미지를 가져올 수 있습니다.
- 클러스터는 Azure API에 액세스할 수 있습니다.
- 다양한 허용 목록 엔드 포인트가 설정됩니다. *방화벽 설정* 섹션에서 이러한 엔드 포인트를 참조할 수 있습니다.

사용자 정의 라우팅을 사용하여 인터넷 액세스에 지원되는 기존의 몇 가지 네트워킹 설정이 있습니다.

네트워크 주소 변환을 사용하는 개인 클러스터

Azure VNET NAT (네트워크 주소 변환) 를 사용하여 클러스터의 서브넷에 대한 아웃 바운드 인터넷 액세스를 제공할 수 있습니다. 설정 지침은 Azure 설명서의 Create a NAT gateway using Azure CLI 에서 참조하십시오.

Azure NAT 및 사용자 정의 라우팅이 구성된 VNet 설정을 사용하는 경우 공용 엔드포인트없이 개인 클러스터를 만들 수 있습니다.

Azure 방화벽이 있는 개인 클러스터
Azure 방화벽을 사용하여 클러스터를 설치하는 데 사용되는 VNet의 아웃 바운드 라우팅을 제공할 수 있습니다. Azure 설명서에서 Azure Firewall을 사용하여 사용자 정의 라우팅을 제공하는 방법 에 대해 자세히 알아볼 수 있습니다.

Azure 방화벽 및 사용자 정의 라우팅이 구성된 VNet 설정을 사용하는 경우 공용 엔드포인트없이 개인 클러스터를 만들 수 있습니다.

프록시 설정이 있는 개인 클러스터
사용자 정의 라우팅과 함께 프록시를 사용하여 인터넷으로의 송신 (Egress)을 허용할 수 있습니다. 클러스터 Operator가 프록시를 사용하여 Azure API에 액세스하지 않도록해야합니다. Operator는 프록시 외부에서 Azure API에 액세스할 수 있어야 합니다.

0.0.0.0/0이 Azure에 의해 자동으로 설정된 상태에서 서브넷의 기본 라우팅 테이블을 사용하는 경우 IP 주소가 공용인 경우에도 모든 Azure API 요청이 Azure의 내부 네트워크를 통해 라우팅됩니다. 네트워크 보안 그룹 규칙이 Azure API 엔드포인트으로의 송신을 허용하는 한 사용자 정의 라우팅이 구성된 프록시를 사용하면 공용 엔드포인트없이 개인 클러스터를 만들 수 있습니다.

인터넷 액세스가 없는 개인 클러스터
Azure API를 제외한 인터넷에 대한 모든 액세스를 제한하는 개인 네트워크를 설치할 수 있습니다. 이는 릴리스 이미지 레지스트리를 로컬로 미러링하여 수행됩니다. 클러스터는 다음에 액세스할 수 있어야 합니다.

- 컨테이너 이미지를 가져올 수있는 내부 레지스트리 미러
- Azure API에 액세스

이러한 요구 사항을 사용할 수 있으면 사용자 정의 라우팅을 사용하여 공용 엔드 포인트가없는 개인 클러스터를 만들 수 있습니다.

5.9.4. OpenShift Container Platform 클러스터에서 VNet의 재사용 정보

OpenShift Container Platform 4.9에서는 Microsoft Azure의 기존 Azure Virtual Network(VNet)에 클러스터를 배포할 수 있습니다. 이때 VNet 및 라우팅 규칙 내의 기존 서브넷도 사용해야 합니다.

기존 Azure VNet에 OpenShift Container Platform을 배포하면 새 계정의 서비스 한도 제한을 회피하거나 회사의 지침에 따른 운영 제한을 따르기 수월해집니다. 이 방법은 VNet을 직접 생성하는 데 필요한 인프라 생성 권한을 받을 수 없을 때 사용하기 좋은 대안입니다.

5.9.4.1. VNet 사용 요구사항

기존 VNet을 사용하여 클러스터를 배포하는 경우 클러스터를 설치하기 전에 추가 네트워크 구성을 수행해야 합니다. 설치 관리자 프로비저닝 인프라 클러스터에서, 설치 관리자는 일반적으로 다음 구성 요소를 생성하지만 기존 VNet에 설치할 때는 생성하지 않습니다.

- 서브넷
- 라우팅 테이블
- VNet

- 네트워크 보안 그룹

참고

설치 프로그램을 사용하려면 클라우드 제공 DNS 서버를 사용해야 합니다. 사용자 지정 DNS 서버 사용은 지원되지 않으며 이로 인해 설치에 실패합니다.

사용자 지정 VNet을 사용하는 경우, 사용할 클러스터와 설치 프로그램에 맞게 VNet과 해당 서브넷을 구성해야 합니다. 설치 프로그램에서 사용할 클러스터의 네트워크 범위를 세분하거나 서브넷에 대한 라우팅 테이블을 설정하거나 DHCP와 같은 VNet 옵션을 설정할 수 없으므로 클러스터를 설치하기 전에 직접 해당 작업을 수행해야 합니다.

클러스터는 기존 VNet과 서브넷이 포함된 리소스 그룹에 액세스할 수 있어야 합니다. 클러스터가 생성하는 모든 리소스는 생성된 별도의 리소스 그룹에 배치되지만 일부 네트워크 리소스는 별도의 그룹으로부터 사용됩니다. 일부 클러스터 Operator는 두 리소스 그룹 모두의 리소스에 액세스할 수 있어야 합니다. 예를 들어 Machine API 컨트롤러는 생성하는 가상 머신에 대한 NICS를 네트워킹 리소스 그룹의 서브넷에 연결합니다.

VNet은 다음 특성을 충족해야 합니다.

- VNet의 CIDR 블록에는 클러스터 시스템의 IP 주소 풀인 **Networking.MachineCIDR** 범위가 포함되어야 합니다.

- VNet과 해당 서브넷은 동일한 리소스 그룹에 속해야 하며 서브넷은 고정 IP 주소 대신 Azure 할당 DHCP IP 주소를 사용하도록 구성해야 합니다.

VNet 내에 두 서브넷을 제공해야 합니다. 하나는 컨트롤 플레인 시스템용이고 다른 하나는 컴퓨팅 시스템용입니다. Azure는 지정한 리전 내의 다른 가용성 영역에 시스템을 배포하므로 클러스터는 기본적으로 고가용성을 갖습니다.

제공한 서브넷이 적합한지 확인하기 위해 설치 프로그램이 다음 데이터를 확인합니다.

- 지정된 서브넷이 모두 존재합니다.

- 두 개의 사설 서브넷(컨트롤 플레인 시스템용 및 컴퓨팅 시스템용)이 있습니다.

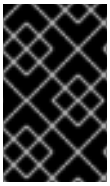
- 서브넷 CIDR이 사용자가 지정한 시스템 CIDR에 속합니다. 프라이빗 서브넷을 제공하지 않는 가용성 영역에서는 시스템이 프로비저닝되지 않습니다. 필요한 경우, 설치 프로그램은 컨트롤 플레인과 작업자 노드를 관리하는 공용 로드 밸런서를 만들고 Azure는 여기에 공용 IP 주소를 할당합니다.

참고

기존 VNet을 사용하는 클러스터를 제거해도 VNet은 삭제되지 않습니다.

5.9.4.1.1. 네트워크 보안 그룹 요구사항

컴퓨팅과 컨트롤 플레인 시스템을 호스팅하는 서브넷에 대한 네트워크 보안 그룹은 클러스터 통신이 올바른지 확인하기 위해 특정 액세스 권한이 필요합니다. 필요한 클러스터 통신 포트에 대한 액세스를 허용하는 규칙을 만들어야 합니다.



중요

클러스터를 설치하기 전에 네트워크 보안 그룹 규칙을 마련해야 합니다. 필요한 액세스 권한 없이 클러스터를 설치하려고 하면 설치 프로그램이 Azure API에 연결할 수 없으며 설치가 실패합니다.

표 5.29. 필수 포트

포트	설명	컨트롤 플레인	컴퓨팅
80	HTTP 트래픽 허용		x
443	HTTPS 트래픽 허용		x
6443	컨트롤 플레인 시스템과의 통신을 허용합니다.	x	
22623	머신을 프로비저닝하기 위해 머신 구성 서버와의 내부 통신 허용	x	

중요

현재는 머신 구성 서버 끝점을 차단하거나 제한할 수 있는 방법이 없습니다. 기존 구성 또는 상태가 없는 새로 프로비저닝된 머신이 구성을 가져올 수 있도록 머신 구성 서버를 네트워크에 노출해야 합니다. 이 모델에서 신뢰의 루트는 CSR(인증서 서명 요청) 끝점으로, kubelet이 클러스터에 가입하기 위해 승인하기 위해 인증서 서명 요청을 보내는 위치입니다. 이로 인해 시크릿 및 인증서와 같은 중요한 정보를 배포하는 데 머신 구성을 사용해서는 안 됩니다.

머신 구성 서버 끝점, 포트 22623 및 22624가 베어 메탈 시나리오에서 보호되도록 하려면 고객이 적절한 네트워크 정책을 구성해야 합니다.

클러스터 구성 요소는 Kubernetes 컨트롤러가 업데이트하는 사용자 제공 네트워크 보안 그룹을 변경하지 않으므로 나머지 환경에 영향을 주지 않고 Kubernetes 컨트롤러에 대한 의사 네트워크 보안 그룹이 생성됩니다.

추가 리소스

- OpenShift SDN 네트워크 플러그인 정보

5.9.4.2. 권한 분할

OpenShift Container Platform 4.3부터 클러스터를 배포하는 데 설치 프로그램에서 프로비저닝한 인프라 클러스터에 필요한 권한 중 일부가 필요하지 않게 되었습니다. 이러한 변경은 회사에서 보유할 수 있는 권한 분할을 모방합니다. 즉, 몇몇 사람이 다른 사람들과 다른 리소스를 클라우드에 생성할 수 있습니다. 예를 들어 인스턴스, 스토리지, 로드 밸런서와 같은 애플리케이션 관련 항목을 생성할 수는 있지만 VNet, 서브넷 또는 인그레스 규칙과 같은 네트워킹 관련 구성 요소는 생성할 수 없습니다.

클러스터를 생성할 때 사용하는 Azure 자격 증명에는 서브넷, 라우팅 테이블, 인터넷 게이트웨이, NAT, VPN과 같은 VNet 내 핵심 네트워킹 구성 요소와 VNet을 생성하는 데 필요한 네트워킹 권한이 필요하지 않습니다. 하지만 로드 밸런서, 보안 그룹, 스토리지 계정 및 노드와 같이 클러스터 내 시스템에 필요한 애플리케이션 리소스를 생성하려면 여전히 권한이 필요합니다.

5.9.4.3. 클러스터 간 격리

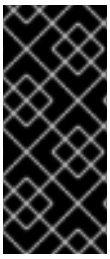
클러스터는 기존 서브넷의 네트워크 보안 그룹을 수정할 수 없기 때문에 VNet에서 클러스터를 서로 격리할 방법이 없습니다.

5.9.5. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.



중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

5.9.6. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 **core** 사용자의 **~/.ssh/authorized_keys** 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 **core**로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. **./openshift-install gather** 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

참고

AWS 키 쌍 과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: **~/.ssh/id_ed25519**)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 **'~/.ssh** 디렉터리에 있는지 확인하십시오.

참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 **ed25519** 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 **rsa** 또는 **ecdsa** 알고리즘을 사용하는 키를 생성합니다.

2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 **~/.ssh/id_ed25519.pub** 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 **./openshift-install gather** 명령을 사용하려는 경우 필요합니다.

참고

일부 배포에서는 **~/.ssh/id_rsa** 및 **~/.ssh/id_dsa**와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. **ssh-agent** 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. **ssh-agent**에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: **~/.ssh/id_ed25519**).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

5.9.7. 설치 프로그램 받기

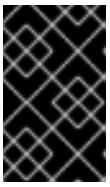
OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

프로세스

1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.
2. 인프라 공급자를 선택합니다.
3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.



중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.

중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 OpenShift Container Platform 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

5.9.8. 수동으로 설치 구성 파일 만들기

Microsoft Azure의 OpenShift Container Platform을 정부 지역에 설치할 때 설치 구성 파일을 수동으로 생성해야 합니다.

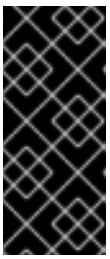
사전 요구 사항

- 로컬 시스템에 설치 프로그램에 제공할 SSH 공개 키가 있습니다. 키는 디버깅 및 재해 복구를 위해 클러스터 노드에 대한 SSH 인증에 사용됩니다.
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿이 있습니다.

프로세스

1. 필요한 설치 자산을 저장할 설치 디렉터리를 만듭니다.

```
$ mkdir <installation_directory>
```



중요

디렉터리를 만들어야 합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

2. 샘플 **install-config.yaml** 파일 템플릿을 사용자 지정하여 **<installation_directory>**에 저장합니다.

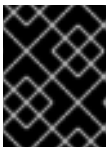
참고

이 설정 파일의 이름을 **install-config.yaml**로 지정해야 합니다.

참고

일부 플랫폼 유형의 경우 대체로 **./openshift-install create install-config --dir <installation_directory>** 를 실행하여 **install-config.yaml** 파일을 생성할 수 있습니다. 프롬프트에서 클러스터 구성에 대한 세부 정보를 제공할 수 있습니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.



중요

install-config.yaml 파일은 설치 과정의 다음 단계에서 사용됩니다. 이 시점에서 이를 백업해야 합니다.

5.9.8.1. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 매개변수 값을 제공하여 클러스터를 호스팅할 클라우드 플랫폼에서 사용자 계정을 설명하고 선택사항으로 클러스터의 플랫폼을 사용자 지정합니다. **install-config.yaml** 설치 구성 파일을 생성할 때 명령줄을 통해 필요한 매개변수 값을 제공합니다. 클러스터를 사용자 지정하면 **install-config.yaml** 파일을 수정하여 플랫폼에 대한 세부 정보를 제공할 수 있습니다.

참고

설치한 후에는 **install-config.yaml** 파일에서 이러한 매개변수를 수정할 수 없습니다.

중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

5.9.8.1.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 5.30. 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	소문자, 하이픈(**-**), 마침표(**.**)로 구성되는 문자열(예: **dev**)입니다.
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

매개변수	설명	값

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

5.9.8.1.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 5.31. 네트워크 매개변수

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체 **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.

매개변수	설명	값
networking.clusterNetwork	Pod의 IP 주소 블록입니다. 기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된 **cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16**  **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

5.9.8.1.3. 선택적 구성 매개변수

선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 5.32. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은 **3**입니다.

매개변수	설명	값
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 인터넷에서 액세스할 수 없는 사설 클러스터를 배포하려면 **publish**를 **Internal**로 설정합니다. 기본값은 **External**입니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

5.9.8.1.4. 추가 Azure 구성 매개 변수

추가 Azure 구성 매개변수는 다음 표에 설명되어 있습니다.

표 5.33. 추가 Azure 매개변수

매개변수	설명	값
compute.platform.azure.osDisk.diskSizeGB	VM의 Azure 디스크 크기입니다.	디스크 크기(GB)를 나타내는 정수입니다. 기본값은 **128** 입니다.
compute.platform.azure.osDisk.diskType	디스크 유형을 정의합니다.	**standard_LRS**, **premium_LRS**, or **standardSSD_LRS**. 기본값은 **premium_LRS** 입니다.
controlPlane.platform.azure.osDisk.diskSizeGB	VM의 Azure 디스크 크기입니다.	디스크 크기(GB)를 나타내는 정수입니다. 기본값은 **1024** 입니다.
controlPlane.platform.azure.osDisk.diskType	디스크 유형을 정의합니다.	**Premium_LRS** 또는 **표준SSD_LRS**. 기본값은 **premium_LRS** 입니다.
platform.azure.baseDomainResourceGroupName	기본 도메인의 DNS 영역을 포함하는 리소스 그룹의 이름입니다.	문자열(예: **production_cluster**).

매개변수	설명	값
platform.azure.resourceGroupName	클러스터를 설치할 기존 리소스 그룹의 이름입니다. 이 리소스 그룹은 비어 있어야 하며 특정 클러스터에만 사용해야 합니다. 클러스터 구성 요소는 리소스 그룹의 모든 리소스에 대한 소유권을 가정합니다. 설치 프로그램의 서비스 주체 범위를 이 리소스 그룹으로 제한하는 경우 해당 환경에서 설치 프로그램에서 사용하는 기타 모든 리소스에 퍼블릭 DNS 영역 및 가상 네트워크와 같은 필수 권한이 있는지 확인해야 합니다. 설치 프로그램을 사용하여 클러스터를 삭제하면 이 리소스 그룹이 삭제됩니다.	문자열(예: **existing_resource_group**)
platform.azure.outboundType	클러스터를 인터넷에 연결하는 데 사용되는 아웃 바운드 라우팅 전략입니다. 사용자 정의 라우팅을 사용하는 경우 클러스터를 설치하기 전에 아웃 바운드 라우팅이 이미 구성된 경우 기존 네트워킹을 사용할 수 있어야 합니다. 설치 프로그램은 사용자 정의 라우팅 설정을 하지 않습니다.	**LoadBalancer** 또는 **UserDefinedRouting**. 기본값은 **LoadBalancer** 입니다.
platform.azure.region	클러스터를 호스팅하는 Azure 리전의 이름입니다.	유효한 리전 이름(예: **centralus**).
platform.azure.zone	시스템을 배치하려는 가용성 영역 목록. 고가용성을 위해 최소한 두 개의 영역을 지정하십시오.	영역 목록(예: **["1", "2", "3"]**)
platform.azure.networkResourceGroupName	클러스터를 배포하려는 기존 VNet이 포함된 리소스 그룹의 이름입니다. 이 이름은 **platform.azure.baseDomainResourceGroupName**과 같을 수 없습니다.	문자열.
platform.azure.virtualNetwork	클러스터를 배포할 기존 VNet의 이름입니다.	문자열.
platform.azure.controlPlaneSubnet	컨트롤 플레인 시스템을 배포할 VNet의 기존 서브넷 이름입니다.	유효한 CIDR(예: **10.0.0.0/16**)
platform.azure.computeSubnet	컴퓨팅 시스템을 배포할 VNet의 기존 서브넷 이름입니다.	유효한 CIDR(예: **10.0.0.0/16**)
platform.azure.cloudName	적절한 Azure API 엔드포인트에서 Azure SDK를 구성하는데 사용되는 Azure 클라우드 환경의 이름입니다. 비어있는 경우 기본값 **AzurePublicCloud**가 사용됩니다.	**AzurePublicCloud** 또는 **AzureUSGovernmentCloud** 와 같은 유효한 클라우드 환경.

참고

Azure 가용성 영역을 사용자 지정하거나 Azure 클러스터와 함께 태그를 사용하여 Azure 리소스를 구성할 수 없습니다.

5.9.8.2. 클러스터 설치를 위한 최소 리소스 요구 사항

각 클러스터 시스템이 다음과 같은 최소 요구사항을 충족해야 합니다.

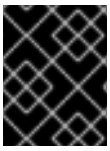
표 5.34. 최소 리소스 요구사항

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
부트스트랩	RHCOS	4	16GB	100GB	300
컨트롤 플레인	RHCOS	4	16GB	100GB	300
컴퓨팅	RHCOS, RHEL 7.9 또는 RHEL 8.4 [3]	2	8GB	100GB	300

1. SMT(동시 멀티스레딩) 또는 하이퍼 스레딩이 활성화되지 않은 경우 하나의 vCPU는 하나의 물리적 코어와 동일합니다. 활성화하면 다음과 같은 공식을 사용하여 해당 비율을 계산합니다. (코어당 스레드 수 × 코어 수) × 소켓 수 = vCPU 수

2. OpenShift Container Platform 및 Kubernetes는 디스크 성능에 민감하며 특히 10ms p99 fsync 기간이 필요한 컨트롤 플레인 노드의 etcd에 더 빠른 스토리지가 권장됩니다. 많은 클라우드 플랫폼에서 스토리지 크기와 IOPS를 함께 확장되므로 충분한 성능을 얻으려면 스토리지 볼륨을 과도하게 할당해야 할 수 있습니다.

3. 사용자가 프로비저닝한 모든 설치와 마찬가지로 클러스터에서 RHEL 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. RHEL 7 컴퓨팅 머신 사용은 더 이상 사용되지 않으며 향후 OpenShift Container Platform 4 릴리스에서 제거될 예정입니다.

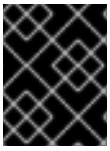


중요

Premium **IO** 가 **true** 로 설정된 Azure 가상 머신을 사용해야 합니다. 머신에는 **하이퍼 VGeneration** 속성에 **V1** 이 포함되어 있어야 합니다.

5.9.8.3. Azure용 샘플 사용자 지정 install-config.yaml 파일

install-config.yaml 파일을 사용자 지정하여 OpenShift Container Platform 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.



중요

이 샘플 YAML 파일은 참조용으로만 제공됩니다. 설치 프로그램을 사용하여 **install-config.yaml** 파일을 받아서 수정해야 합니다.

```
apiVersion: v1
baseDomain: example.com (1)
```

```
controlPlane: (2)
  hyperthreading: Enabled (3) (4)
  name: master
  platform:
    azure:
      osDisk:
        diskSizeGB: 1024 (5)
        diskType: Premium_LRS
      type: Standard_D8s_v3
  replicas: 3
compute: (6)
- hyperthreading: Enabled (7)
  name: worker
  platform:
    azure:
      type: Standard_D2s_v3
      osDisk:
        diskSizeGB: 512 (8)
        diskType: Standard_LRS
      zones: (9)
      - "1"
      - "2"
      - "3"
  replicas: 5
metadata:
  name: test-cluster (10)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  azure:
    baseDomainResourceGroupName: resource_group (11)
    region: usgovvirginia
    resourceGroupName: existing_resource_group (12)
    networkResourceGroupName: vnet_resource_group (13)
    virtualNetwork: vnet (14)
    controlPlaneSubnet: control_plane_subnet (15)
    computeSubnet: compute_subnet (16)
    outboundType: UserDefinedRouting (17)
    cloudName: AzureUSGovernmentCloud (18)
pullSecret: '{"auths": ...}' (19)
fips: false (20)
sshKey: ssh-ed25519 AAAA... (21)
publish: Internal (22)
```

(1) (10) (19) 필수 항목입니다.

2 6 이러한 매개변수와 값을 지정하지 않으면 설치 프로그램은 기본값을 적용합니다.

3 7 **controlPlane** 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

4 동시 멀티스레딩 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. 일부 클러스터 시스템에서 동시 멀티스레딩을 비활성화할 경우에는 해당 멀티스레딩을 모든 클러스터 시스템에서 비활성화해야 합니다.

중요

동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다. 동시 멀티스레딩을 비활성화하는 경우, 사용자 시스템에 더 큰 가상 머신 유형(예: **Standard_D8s_v3**)을 사용하십시오.

5 8 사용할 디스크 크기는 GB 단위로 지정할 수 있습니다. 컨트롤 플레인 노드의 최소 권장 크기는 1024GB입니다.

9 시스템을 배포할 영역 목록을 지정합니다. 고가용성을 위해 최소한 두 개의 영역을 지정하십시오.

11 기본 도메인의 DNS 영역을 포함하는 리소스 그룹의 이름을 지정합니다.

12 클러스터를 설치할 기존 리소스 그룹의 이름을 지정합니다. 정의되지 않은 경우 클러스터에 새 리소스 그룹이 생성됩니다.

13 기존 VNet을 사용하는 경우 이를 포함하는 리소스 그룹의 이름을 지정합니다.

14 기존 VNet을 사용하는 경우 해당 이름을 지정합니다.

15 기존 VNet을 사용하는 경우 컨트롤 플레인 시스템을 호스팅할 서브넷의 이름을 지정합니다.

16 기존 VNet을 사용하는 경우 컴퓨팅 시스템을 호스팅할 서브넷의 이름을 지정합니다.

17 사용자의 아웃 바운드 라우팅을 사용자 지정할 수 있습니다. 사용자 정의 라우팅을 구성하면 클러스터에서 외부 엔드 포인트가 노출되지 않습니다. 송신에 대한 사용자 정의 라우팅을 사용하려면 기존 VNet에 클러스터를 배포해야 합니다.

18 클러스터를 배포할 Azure 클라우드 환경의 이름을 지정합니다. MAG (Microsoft Azure Government) 리전에 배포하도록 **AzureUSGovernmentCloud**를 설정합니다. 기본값은 **AzurePublicCloud** 입니다.

20 FIPS 모드 활성화 또는 비활성화 여부입니다. 기본적으로 FIPS 모드는 비활성화됩니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.

중요

FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.

21 선택사항으로, 클러스터의 시스템에 액세스하는 데 사용할 **sshKey** 값을 제공할 수도 있습니다.

참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

22 클러스터의 사용자 끝점을 게시하는 방법. 인터넷에서 액세스할 수 없는 프라이빗 클러스터를 배포하려면 **publish**를 **Internal**로 설정합니다. 기본값은 **External**입니다.

5.9.8.4. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 HTTP 또는 HTTPS 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 OpenShift Container Platform 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 API에 대한 호출을 포함하여 모든 클러스터 발신(Egress) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.

참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 Red Hat OpenStack Platform (RHOSP)에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1 클러스터 외부에서 HTTP 연결을 구축하는 데 사용할 프록시 URL입니다. URL 스키마는 **http**여야 합니다.

2 클러스터 외부에서 HTTPS 연결을 구축하는 데 사용할 프록시 URL입니다.

3 대상 도메인 이름, IP 주소 또는 프록시에서 제외할 기타 네트워크 CIDR로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. ***를 사용하여 모든 대상에 대해 프록시를 바이패스합니다.

4 이 값을 제공하면 설치 프로그램에서 추가 CA 인증서를 보유할 **openshift -config 네임스페이스에 user-ca- bundle** 이라는 구성 맵을 생성합니다. **additionalTrustBundle** 및 하나 이상의 프록시 설정을 제공하는 경우 **프록시** 오브젝트는 **trustedCA** 필드의 **user-ca-bundle** 구성 맵을 참조하도록 구성됩니다. 그러면 Cluster Network Operator에서 **trustedCA** 매개변수에 대해 지정된 콘텐츠를 RHCOS 신뢰 번들과 병합하는 **trusted-ca-bundle** 구성 맵을 생성합니다. 프록시의 ID 인증서를 RHCOS 트러스트 번들에 있는 기관에서 서명하지 않은 경우 **additionalTrustBundle** 필드가 있어야 합니다.

참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 OpenShift Container Platform을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 nil이 됩니다.

참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

5.9.9. 클러스터 배포

호환되는 클라우드 플랫폼에 OpenShift Container Platform을 설치할 수 있습니다.

중요

최초 설치 과정에서 설치 프로그램의 **create cluster** 명령을 한 번만 실행할 수 있습니다.

사전 요구 사항

- 클러스터를 호스팅하는 클라우드 플랫폼으로 계정을 구성합니다.
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

절차

1. 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터 배포를 초기화합니다.

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** 값으로 사용자 지정한 **./install-config.yaml** 파일의 위치를 지정합니다.

2 다른 설치 세부 사항을 보려면 **info** 대신 **warn**, **debug** 또는 **error**를 지정합니다.

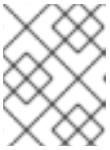
참고

호스트에 구성된 클라우드 공급자 계정에 클러스터를 배포하기에 충분한 권한이 없는 경우, 설치 프로세스가 중단되고 누락된 권한을 알리는 메시지가 표시됩니다.

클러스터 배포가 완료되면 웹 콘솔로 연결되는 링크와 **kubeadmin** 사용자의 인증 정보가 포함된 클러스터 액세스 지침이 사용자 터미널에 표시됩니다.

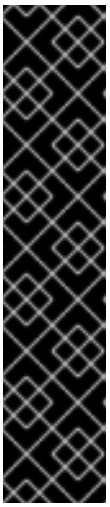
출력 예

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```



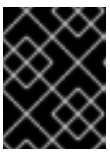
참고

설치에 성공하면 클러스터 액세스 및 인증 정보도 **<installation_directory>/.openshift_install.log**로 출력됩니다.



중요

- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

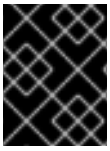


중요

설치 프로그램에서 생성되는 파일이나 설치 프로그램을 삭제해서는 안 됩니다. 클러스터를 삭제하려면 두 가지가 모두 필요합니다.

5.9.10. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift CLI(**oc**)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 **oc**를 설치할 수 있습니다.



중요

이전 버전의 **oc**를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 **oc**를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치
다음 절차를 사용하여 Linux에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Linux Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. **oc** 바이너리를 **PATH**에 있는 디렉터리에 배치합니다.
PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치
다음 절차에 따라 Windows에 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Windows Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. ZIP 프로그램으로 아카이브의 압축을 풉니다.
5. **oc** 바이너리를 **PATH**에 있는 디렉터리로 이동합니다.
PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치
다음 절차에 따라 macOS에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 MacOSX Client** 항목 옆에 있는 **지금 다운로드를** 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 해제하고 압축을 풉니다.
5. **oc** 바이너리 PATH의 디렉터리로 이동합니다.
   PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

5.9.11. CLI를 사용하여 클러스터에 로그인

클러스터 **kubeconfig** 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. **kubeconfig** 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- **oc** CLI를 설치했습니다.

프로세스

1. **kubeadmin** 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 **oc** 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

추가 리소스

- OpenShift Container Platform 웹 콘솔 액세스 및 이해에 대한 자세한 내용은 웹 콘솔에 액세스를 참조하십시오.

5.9.12. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보 를 참조하십시오.

5.9.13. 다음 단계

- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃 을 수행할 수 있습니다.

5.10. ARM 템플릿을 사용하여 AZURE에 클러스터 설치

OpenShift Container Platform 버전 4.9에서는 사용자가 제공하는 인프라를 사용하여 Microsoft Azure에 클러스터를 설치할 수 있습니다.

이러한 단계를 완료할 수 있도록 지원하거나 자체 모델링에 도움이 되는 여러 가지 ARM(Azure Resource Manager) 템플릿이 제공됩니다.

중요

사용자가 프로비저닝한 인프라 설치를 수행하는 단계는 예시용으로만 제공됩니다. 사용자가 제공하는 인프라를 사용하여 클러스터를 설치하려면 클라우드 공급자 및 OpenShift Container Platform 설치 프로세스에 대한 정보가 필요합니다. 안내된 단계를 수행하거나 자체 모델링에 유용한 몇 가지 ARM 템플릿이 제공됩니다. 여러 다른 방법을 통해 필요한 리소스를 자유롭게 생성할 수도 있습니다. 템플릿은 예시일 뿐입니다.

5.10.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비 에 대한 문서를 읽습니다.
- 클러스터를 호스팅할 Azure 계정을 구성했습니다.
- Azure CLI를 다운로드하여 컴퓨터에 설치했습니다. Azure 문서의 Azure CLI 설치를 참조하십시오. 아래 문서는 Azure CLI 버전 **2.38.0** 을 사용하여 마지막으로 테스트되었습니다. Azure CLI 명령은 사용하는 버전에 따라 다르게 수행될 수 있습니다.
- 방화벽을 사용하며 Telemetry 서비스를 사용할 예정인 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성해야 합니다.
- 사용자의 환경에서 클라우드 ID 및 액세스 관리(IAM) API에 액세스할 수 없거나 **kube-system** 네임스페이스에 관리자 수준의 인증 정보 시크릿을 저장하지 않으려면 IAM 인증 정보 수동 생성 및 유지 관리를 참조하십시오.

참고

프록시를 구성하는 경우에도 해당 사이트 목록을 검토하십시오.

5.10.2. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.

중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

5.10.3. Azure 프로젝트 구성

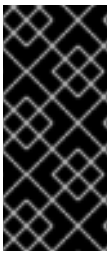
OpenShift Container Platform을 설치하려면 먼저 호스팅할 Azure 프로젝트를 구성해야 합니다.

중요

공용 끝점을 통해 사용할 수 있는 모든 Azure 리소스에는 리소스 이름 제한 사항이 적용되며, 특정 용어를 사용하는 리소스를 생성할 수 없습니다. Azure가 제한하는 용어 목록은 Azure 문서의 예약된 리소스 이름 오류 해결 을 참조하십시오.

5.10.3.1. Azure 계정 제한

OpenShift Container Platform 클러스터는 수많은 Microsoft Azure 구성 요소를 사용하며, 기본 Azure 서브스크립션 및 서비스 제한, 할당량 및 제약 조건은 OpenShift Container Platform 클러스터 설치에 영향을 미칩니다.



중요

기본 제한값은 무료 평가판 및 종량 과금제와 같은 상품 카테고리 유형과 Dv2, F, G 등 시리즈에 따라 다릅니다. 예를 들어 기업 계약 서브스크립션의 기본값은 350개 코어입니다.

서브스크립션 유형에 대한 제한값을 확인하고 필요한 경우 Azure에 기본 클러스터를 설치하기 전에 계정에 대한 할당량 제한값을 늘리십시오.

다음 표에 OpenShift Container Platform 클러스터를 설치하고 실행하는 데 영향을 미칠 수 있는 Azure 구성 요소 제한값이 요약되어 있습니다.

구성 요소	기본적으로 필요한 구성 요소 수	기본 Azure 제한 값	설명
vCPU	40	리전당 20개	기본 클러스터의 경우 40개의 vCPU가 필요하므로 계정 제한을 늘려야 합니다. 기본적으로 각 클러스터는 다음 인스턴스를 생성합니다. ● 설치 후 제거되는 하나의 부트스트랩 시스템 ● 컨트롤 플레인 시스템 세 개 ● 컴퓨팅 시스템 세 개 부트스트랩 시스템은 4개의 vCPU를 사용하는 **Standard_D4s_v3** 시스템을 사용하고, 컨트롤 플레인 시스템은 8개의 vCPU를 사용하는 **Standard_D8s_v3** 가상 머신을 사용하고, 작업자 시스템은 4개의 vCPU를 사용하는 **Standard_D4s_v3** 가상 머신을 사용하므로, 기본 클러스터에는 40개의 vCPU가 필요합니다. 4개의 vCPU를 사용하는 부트스트랩 노드 VM은 설치 중에만 사용됩니다. 더 많은 작업자 노드를 배포하거나, 자동 크기 조정을 활성화하거나, 대규모 워크로드를 배포하거나, 다른 인스턴스 유형을 사용하려면 필요한 시스템을 클러스터가 배포할 수 있도록 계정의 vCPU 제한값을 더 늘려야 합니다. 기본적으로 설치 프로그램은 컨트롤 플레인과 컴퓨팅 시스템을 한 리전 내의 모든 가용성 영역에 분배합니다. 클러스터의 고가용성을 보장하기 위해서는 가용성 영역이 세 개 이상인 리전을 선택하십시오. 각 리전에 가용성 영역이 세 개 미만인 경우에는 설치 프로그램이 사용 가능한 영역에 둘 이상의 컨트롤 플레인 시스템을 배치합니다.
OS 디스크	7		VM OS 디스크는 컨트롤 플레인 시스템에 대해 테스트되고 권장되는 최소 처리량 5000 IOPS/200MBps를 유지할 수 있어야 합니다. 이 처리량은 최소 1 TiB Premium SSD (P30)를 보유하여 제공할 수 있습니다. Azure에서는 디스크 성능은 SSD 디스크 크기에 직접적으로 의존하므로 **Standard_D8s_v3** 또는 기타 유사한 시스템 유형에서 지원되는 처리량과 5000 IOPS 목표를 달성하려면 최소 P30 디스크가 필요합니다. 읽기 대기 시간이 짧고 읽기 IOPS 및 처리량을 높이려면 호스트 캐싱을 **ReadOnly**로 설정해야 합니다. VM 메모리 또는 로컬 SSD 디스크에 있는 캐시에서 수행된 읽기는 Blob 스토리지에 있는 데이터 디스크에서 읽기보다 훨씬 빠릅니다.
VNet	1	리전당 1000개	각 기본 클러스터에는 두 개의 서브넷이 포함된 하나의 가상 네트워크(VNet)가 필요합니다.

구성 요소	기본적으로 필요한 구성 요소 수	기본 Azure 제한 값	설명
네트워크 인터페이스	7	리전당 65,536개	각 기본 클러스터에는 7개의 네트워크 인터페이스가 필요합니다. 더 많은 시스템을 생성하거나 배포된 워크로드가 로드 밸런서를 생성하는 경우 클러스터는 더 많은 네트워크 인터페이스를 사용합니다.
네트워크 보안 그룹	2	5000	각 클러스터는 VNet의 각 서브넷에 대한 네트워크 보안 그룹을 생성합니다. 기본 클러스터는 컨트롤 플레인과 컴퓨팅 노드 서브넷에 대한 네트워크 보안 그룹을 생성합니다: **controlplane**: 어디에서나 포트 6443에서 컨트롤 플레인 시스템에 도달할 수 있습니다. **node**: 포트 80 및 443을 통해 인터넷에서 작업자 노드에 도달할 수 있습니다.
네트워크 로드 밸런서	3	리전당 1000개	각 클러스터는 다음 로드 밸런서를 생성합니다. **default**: 작업자 시스템에서 포트 80과 443에 대한 요청을 로드 밸런싱하는 공용 IP 주소 **internal**: 컨트롤 플레인 시스템에서 포트 6443과 22623에 대한 요청을 로드 밸런싱하는 개인 IP 주소 **external**: 컨트롤 플레인 시스템에서 포트 6443에 대한 요청을 로드 밸런싱하는 공용 IP 주소 애플리케이션이 더 많은 Kubernetes **LoadBalancer** Service 개체를 생성하면 클러스터가 더 많은 로드 밸런서를 사용합니다.
공용 IP 주소	3		두 개의 공용 로드 밸런서 각각이 공용 IP 주소를 사용합니다. 또한 부트스트랩 시스템은 공용 IP 주소를 사용하므로 설치 중 문제를 해결하기 위해 시스템으로 SSH를 실행할 수 있습니다. 부트스트랩 노드의 IP 주소는 설치 중에만 사용됩니다.
개인 IP 주소	7		내부 로드 밸런서, 3개의 컨트롤 플레인 시스템 및 3개의 각 작업자 시스템은 각각 개인 IP 주소를 사용합니다.

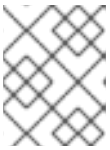
구성 요소	기본적으로 필요한 구성 요소 수	기본 Azure 제한 값	설명
spot VM vCPU (선택 사항)	0 스팟 VM을 구성하는 경우 클러스터에 모든 컴퓨팅 노드에 대해 두 개의 스팟 VM vCPU가 있어야 합니다.	리전당 20개	선택적 구성 요소입니다. Spot 가상 머신을 사용하려면 Azure 기본 제한을 클러스터의 컴퓨팅 노드 수보다 두 배 이상 늘려야 합니다.  **참고** 컨트롤 플레인 노드에 스팟 VM을 사용하는 것은 권장되지 않습니다.

5.10.3.2. Azure에서 퍼블릭 DNS 영역 구성

OpenShift Container Platform을 설치하려면 사용하는 Microsoft Azure 계정에 전용의 퍼블릭 호스팅 DNS 영역이 있어야 합니다. 도메인에 대한 권한도 이 영역에 있어야 합니다. 이 서비스는 클러스터와 외부 연결에 필요한 클러스터 DNS 확인 및 이름 조회 기능을 제공합니다.

프로세스

1. 도메인 또는 하위 도메인과 등록 기관을 식별합니다. 기존 도메인 및 등록 기관을 이전하거나 Azure 또는 다른 소스를 통해 새 도메인과 등록 기관을 받을 수 있습니다.

   참고

   Azure를 통한 도메인 구매에 대한 자세한 내용은 Azure 문서에서 Azure App Service의 사용자 지정 도메인 이름 구매를 참조하십시오.

2. 기존 도메인과 등록 기관을 사용하는 경우 해당 DNS를 Azure로 마이그레이션합니다. Azure 문서의 활성 DNS 이름을 Azure App Service로 마이그레이션 을 참조하십시오.

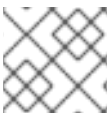
3. 도메인에 맞게 DNS를 구성합니다. Azure 문서의 자습서: Azure DNS에서 도메인 호스팅 에 나온 단계에 따라 도메인 또는 하위 도메인에 대한 퍼블릭 호스팅 영역을 생성하고, 권한 있는 새 이름 서버를 추출하고, 도메인이 사용하는 이름 서버의 등록 기관 레코드를 업데이트합니다.
   적절한 루트 도메인(예: **openshiftcorp.com**) 또는 하위 도메인(예: **clusters.openshiftcorp.com**)을 사용합니다.

4. 하위 도메인을 사용하는 경우, 회사의 프로시저에 따라 상위 도메인에 위임 레코드를 추가합니다.

DNS 영역을 만드는 이 예제 를 방문하여 Azure의 DNS 솔루션을 볼 수 있습니다.

5.10.3.3. Azure 계정 제한 늘리기

계정 제한을 늘리려면 Azure 포털에서 지원 요청을 제출하십시오.



참고

지원 요청당 한 가지 유형의 할당량만 늘릴 수 있습니다.

프로세스

1. Azure 포털의 왼쪽 하단 모서리에서 **Help + support**를 클릭합니다.
2. **New support request**를 클릭한 후 필요한 값을 선택합니다.
   a. **Issue type** 목록에서 **Service and subscription limits (quotas)**을 선택합니다.
   b. **Subscription** 목록에서 수정할 서브스크립션을 선택합니다.
   c. **Quota type** 목록에서 증가시킬 할당량을 선택합니다. 예를 들어 클러스터를 설치하는 데 필요해서 vCPU 수를 늘리려면 **Compute-VM (cores-vCPUs) subscription limit increases**를 선택합니다.
   d. **Next: Solutions**를 클릭합니다.
3. **Problem Details** 페이지에서 할당량 증가에 필요한 정보를 제공합니다.
   a. **Provide details**를 클릭하고 **Quota details** 창에서 필요한 세부 사항을 제공합니다.
   b. SUPPORT METHOD 및 CONTACT INFO 섹션에서 문제 심각도와 연락처 정보를 제공합니다.
4. **Next: Review + create**을 클릭한 후 **Create**을 클릭합니다.

5.10.3.4. 인증서 서명 요청 관리

사용자가 프로비저닝하는 인프라를 사용하는 경우 자동 시스템 관리 기능으로 인해 클러스터의 액세스가 제한되므로 설치한 후 클러스터 인증서 서명 요청(CSR)을 승인하는 메커니즘을 제공해야 합니다. **kube-controller-manager**는 kubelet 클라이언트 CSR만 승인합니다. **machine-approver**는 올바른 시스템에서 발행한 요청인지 확인할 수 없기 때문에 kubelet 자격 증명을 사용하여 요청하는 서비스 인증서의 유효성을 보장할 수 없습니다. kubelet 서빙 인증서 요청의 유효성을 확인하고 요청을 승인하는 방법을 결정하여 구현해야 합니다.

5.10.3.5. 필수 Azure 역할

OpenShift Container Platform에는 Microsoft Azure 리소스를 관리할 수 있도록 서비스 주체가 필요합니다. 서비스 주체를 생성하려면 Azure 계정 서브스크립션에 다음 역할이 있어야 합니다.

- **User Access Administrator**
- **기여자**

Azure 포털에서 역할을 설정하려면 Azure 문서의 RBAC 및 Azure 포털을 사용하여 Azure 리소스에 대한 액세스 관리를 참조하십시오.

5.10.3.6. 서비스 주체 생성

OpenShift Container Platform 및 해당 설치 프로그램은 Azure Resource Manager를 사용하여 Microsoft Azure 리소스를 생성하므로 이를 나타내는 서비스 주체를 생성해야 합니다.

사전 요구 사항

- Azure CLI를 설치 또는 업데이트합니다.
- Azure 계정은 사용하는 서브스크립션에 대한 필요한 역할을 갖습니다.

프로세스

1. Azure CLI에 로그인합니다.

```
$ az login
```

2. Azure 계정이 서브스크립션을 사용하는 경우 올바른 서브스크립션을 사용하고 있는지 확인합니다.

   a. 사용 가능한 계정 목록을 보고 클러스터에 사용하려는 서브스크립션의 **tenantId** 값을 기록합니다.

```
$ az account list --refresh
```

   출력 예

```
[
  {
    "cloudName": "AzureCloud",
    "id": "9bab1460-96d5-40b3-a78e-17b15e978a80",
    "isDefault": true,
    "name": "Subscription Name",
    "state": "Enabled",
    "tenantId": "6057c7e9-b3ae-489d-a54e-de3f6bf6a8ee",
    "user": {
      "name": "you@example.com",
      "type": "user"
    }
  }
]
```

   b. 활성 계정 세부 사항을 보고 **tenantId** 값이 사용하려는 서브스크립션과 일치하는지 확인합니다.

```
$ az account show
```

   출력 예

```
{
  "environmentName": "AzureCloud",
  "id": "9bab1460-96d5-40b3-a78e-17b15e978a80",
  "isDefault": true,
  "name": "Subscription Name",
  "state": "Enabled",
  "tenantId": "6057c7e9-b3ae-489d-a54e-de3f6bf6a8ee", (1)
  "user": {
    "name": "you@example.com",
    "type": "user"
  }
}
```

(1) **tenantId** 매개변수 값이 올바른 서브스크립션 ID인지 확인합니다.

c. 올바른 서브스크립션을 사용하지 않는 경우, 활성 서브스크립션을 변경합니다.

```
$ az account set -s <subscription_id> 1
```

1 서브스크립션 ID를 지정합니다.

d. 서브스크립션 ID 업데이트를 확인합니다.

```
$ az account show
```

출력 예

```
{
  "environmentName": "AzureCloud",
  "id": "33212d16-bdf6-45cb-b038-f6565b61edda",
  "isDefault": true,
  "name": "Subscription Name",
  "state": "Enabled",
  "tenantId": "8049c7e9-c3de-762d-a54e-dc3f6be6a7ee",
  "user": {
    "name": "you@example.com",
    "type": "user"
  }
}
```

3. 출력의 **tenantId** 및 **id** 매개변수 값을 기록합니다. OpenShift Container Platform 설치 중에 이러한 값이 필요합니다.

4. 계정에 대한 서비스 주체를 생성합니다.

```
$ az ad sp create-for-rbac --role Contributor --name <service_principal> \ 1
  --scopes /subscriptions/<subscription_id> 2
  --years <years> 3
```

1 서비스 주체 이름을 지정합니다.

2 서브스크립션 ID를 지정합니다.

3 년 수를 지정합니다. 기본적으로 서비스 주체는 1년 후에 만료됩니다. **--years** 옵션을 사용하면 서비스 주체의 유효성을 확장할 수 있습니다.

출력 예

```
Creating 'Contributor' role assignment under scope '/subscriptions/<subscription_id>'
The output includes credentials that you must protect. Be sure that you do not
include these credentials in your code or check the credentials into your source
control. For more information, see https://aka.ms/azadsp-cli
{
  "appId": "ac461d78-bf4b-4387-ad16-7e32e328aec6",
  "displayName": <service_principal>",
```

```
  "password": "00000000-0000-0000-0000-000000000000",
  "tenantId": "8049c7e9-c3de-762d-a54e-dc3f6be6a7ee"
}
```

5. 이전 출력의 **appId** 및 **password** 매개변수 값을 기록합니다. OpenShift Container Platform 설치 중에 이러한 값이 필요합니다.

6. 다음 명령을 실행하여 **User Access Administrator** 역할을 할당합니다.

```
$ az role assignment create --role "User Access Administrator" \
  --assignee-object-id $(az ad sp show --id <appId> --query id -o tsv) 1
```

1 서비스 주체의 **appId** 매개변수 값을 지정합니다.

추가 리소스

- CCO 모드에 대한 자세한 내용은 Cloud Credential Operator 정보를 참조하십시오.

5.10.3.7. 지원되는 Azure 리전

설치 프로그램은 서브스크립션을 기반으로 사용 가능한 Microsoft Azure 리전 목록을 동적으로 생성합니다.

지원되는 Azure 리전

- **australiacentral** (호주 중부)
- **australiaeast** (호주 동부)
- **australiasoutheast** (호주 남동부)
- **brazilsouth** (브라질 남부)
- **canadacentral** (캐나다 중부)
- **canadaeast** (캐나다 동부)
- **centralindia** (인도 중부)
- **centralus** (미국 중부)
- **eastasia** (동아시아)
- **eastus** (미국 동부)
- **eastus2** (미국 동부 2)
- **francecentral** (프랑스 중부)
- **germanywestcentral** (독일 중서부)
- **japaneast** (일본 동부)
- **japanwest** (일본 서부)

- **koreacentral** (한국 중부)
- **koreasouth** (한국 남부)
- **northcentralus** (미국 중북부)
- **northeurope** (북유럽)
- **norwayeast** (노르웨이 동부)
- **CloudEventtarcentral** (Qatar Central)
- **southafricanorth** (남아프리카 북부)
- **southcentralus** (미국 중남부)
- **southeastasia** (동남아시아)
- **southindia** (인도 남부)
- **switzerlandnorth** (스위스 북부)
- **uaenorth** (UAE 북부)
- **uksouth** (영국 남부)
- **ukwest** (영국 서부)
- **westcentralus** (미국 중서부)
- **westeurope** (서유럽)
- **westindia** (인도 서부)
- **westus** (미국 서부)
- **westus2** (미국 서부 2)

지원되는 Azure Government 리전
OpenShift Container Platform 버전 4.6에는 다음과 같은 MAG (Microsoft Azure Government) 리전에 대한 지원이 추가되어 있습니다.

- **usgovtexas** (US Gov Texas)
- **usgovvirginia** (US Gov Virginia)

사용 가능한 모든 MAG 리전은 Azure 설명서에서 확인할 수 있습니다. 제공되는 다른 MAG 리전은 OpenShift Container Platform에서 작동할 것으로 예상되지만 아직 테스트되지 않았습니다.

5.10.4. 사용자 프로비저닝 인프라를 포함한 클러스터의 시스템 요구사항

사용자 프로비저닝 인프라가 포함된 클러스터의 경우, 필요한 모든 시스템을 배포해야 합니다.

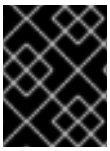
이 섹션에서는 사용자 프로비저닝 인프라에 OpenShift Container Platform을 배포해야 하는 요구 사항에 대해 설명합니다.

5.10.4.1. 클러스터 설치에 필요한 시스템

최소 OpenShift Container Platform 클러스터에 다음과 같은 호스트가 필요합니다.

표 5.35. 최소 필수 호스트

호스트	설명
임시 부트스트랩 시스템 한 개	컨트롤 플레인 시스템 세 개에 OpenShift Container Platform 클러스터를 배포하기 위한 부트스트랩 시스템이 클러스터에 필요합니다. 클러스터를 설치한 후 부트스트랩 시스템을 제거할 수 있습니다.
컨트롤 플레인 시스템 세 개	컨트롤 플레인 시스템은 컨트롤 플레인을 구성하는 Kubernetes 및 OpenShift Container Platform 서비스를 실행합니다.
두 개 이상의 컴퓨팅 시스템(작업자 시스템이라고도 함).	OpenShift Container Platform 사용자가 요청한 워크로드는 컴퓨팅 머신에서 실행됩니다.



중요

클러스터의 고가용성을 유지하려면 이러한 클러스터 시스템에 대해 별도의 물리적 호스트를 사용하십시오.

부트스트랩, 컨트롤 플레인 시스템은 운영 체제로 RHCOS (Red Hat Enterprise Linux CoreOS)를 사용해야 합니다. 그러나 컴퓨팅 머신은 RHCOS(Red Hat Enterprise Linux CoreOS), RHEL (Red Hat Enterprise Linux) 7.9 또는 RHEL 8.4 중에서 선택할 수 있습니다.

RHCOS는 RHEL 8(Red Hat Enterprise Linux)을 기반으로하며 모든 하드웨어 인증 및 요구사항을 모두 이어받습니다. Red Hat Enterprise Linux 기술 기능 및 제한 을 참조하십시오.

5.10.4.2. 클러스터 설치를 위한 최소 리소스 요구 사항

각 클러스터 시스템이 다음과 같은 최소 요구사항을 충족해야 합니다.

표 5.36. 최소 리소스 요구사항

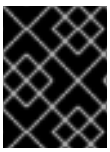
시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
부트스트랩	RHCOS	4	16GB	100GB	300
컨트롤 플레인	RHCOS	4	16GB	100GB	300
컴퓨팅	RHCOS, RHEL 7.9 또는 RHEL 8.4 [3]	2	8GB	100GB	300

1. SMT(동시 멀티스레딩) 또는 하이퍼 스레딩이 활성화되지 않은 경우 하나의 vCPU는 하나의 물리적 코어와 동일합니다. 활성화하면 다음과 같은 공식을 사용하여 해당 비율을 계산합니다. (코어당 스레드 수 × 코어 수) × 소켓 수 = vCPU 수

2. OpenShift Container Platform 및 Kubernetes는 디스크 성능에 민감하며 특히 10ms p99 fsync 기

간이 필요한 컨트롤 플레인 노드의 etcd에 더 빠른 스토리지가 권장됩니다. 많은 클라우드 플랫폼에서 스토리지 크기와 IOPS를 함께 확장되므로 충분한 성능을 얻으려면 스토리지 볼륨을 과도하게 할당해야 할 수 있습니다.

3. 사용자가 프로비저닝한 모든 설치와 마찬가지로 클러스터에서 RHEL 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. RHEL 7 컴퓨팅 머신 사용은 더 이상 사용되지 않으며 향후 OpenShift Container Platform 4 릴리스에서 제거될 예정입니다.



중요

Premium **IO** 가 **true** 로 설정된 Azure 가상 머신을 사용해야 합니다. 머신에는 **하이퍼 VGeneration** 속성에 **V1** 이 포함되어 있어야 합니다.

5.10.5. Azure Marketplace 이미지 선택

Azure Marketplace 오퍼링을 사용하여 OpenShift Container Platform 클러스터를 배포하는 경우 먼저 Azure Marketplace 이미지를 가져와야 합니다. 설치 프로그램은 이 이미지를 사용하여 작업자 노드를 배포합니다. 이미지를 가져올 때 다음 사항을 고려하십시오.

- 이미지가 동일하지만 Azure Marketplace 게시자는 지역에 따라 다릅니다. 북미에 있는 경우 게시자로 **redhat** 을 지정합니다. EMEA에 있는 경우 게시자로 **redhat-limited** 를 지정합니다.
- 이 프로모션에는 **rh-ocp-worker** SKU 및 **rh-ocp-worker-gen1** SKU가 포함되어 있습니다. **rh-ocp-worker** SKU는 Hyper-V generation 버전 2 VM 이미지를 나타냅니다. OpenShift Container Platform에서 사용되는 기본 인스턴스 유형은 버전 2와 호환됩니다. 버전 1 호환 가능한 인스턴스 유형을 사용하려는 경우 **rh-ocp-worker-gen1** SKU와 연결된 이미지를 사용합니다. **rh-ocp-worker-gen1** SKU는 Hyper-V 버전 1 VM 이미지를 나타냅니다.

사전 요구 사항

- Azure CLI 클라이언트 **(az)** 를 설치했습니다.
- Azure 계정에는 제공 권한이 있으며 Azure CLI 클라이언트를 사용하여 이 계정에 로그인했습니다.

프로세스

1. 다음 명령 중 하나를 실행하여 사용 가능한 모든 OpenShift Container Platform 이미지를 표시합니다.
   - 북미:

```
$ az vm image list --all --offer rh-ocp-worker --publisher redhat -o table
```

출력 예

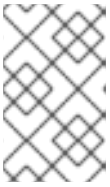
```
Offer          Publisher      Sku                 Urn                                                                 Version
-------------  -------------  ------------------  ------------------------------------------------------------------  -------------
rh-ocp-worker  RedHat         rh-ocp-worker       RedHat:rh-ocp-worker:rh-ocpworker:4.8.2021122100                    4.8.2021122100
rh-ocp-worker  RedHat         rh-ocp-worker-gen1  RedHat:rh-ocp-worker:rh-ocp-worker-gen1:4.8.2021122100              4.8.2021122100
```

- EMEA:

```
$  az vm image list --all --offer rh-ocp-worker --publisher redhat-limited -o table
```

출력 예

```
Offer          Publisher       Sku                 Urn                                                                  Version
-------------  --------------  ------------------  -------------------------------------------------------------------  ---
----  --------------
rh-ocp-worker  redhat-limited  rh-ocp-worker       redhat-limited:rh-ocp-worker:rh-ocp-
worker:4.8.2021122100       4.8.2021122100
rh-ocp-worker  redhat-limited  rh-ocp-worker-gen1  redhat-limited:rh-ocp-worker:rh-ocp-
worker-gen1:4.8.2021122100  4.8.2021122100
```



참고

설치하는 OpenShift Container Platform 버전에 관계없이 사용할 올바른 버전의 Azure Marketplace 이미지는 4.8.x입니다. 필요한 경우 설치 프로세스의 일부로 VM이 자동으로 업그레이드됩니다.

2. 다음 명령 중 하나를 실행하여 제안의 이미지를 검사합니다.

- 북미:

```
$ az vm image show --urn redhat:rh-ocp-worker:rh-ocp-worker:<version>
```

- EMEA:

```
$ az vm image show --urn redhat-limited:rh-ocp-worker:rh-ocp-worker:<version>
```

3. 다음 명령 중 하나를 실행하여 제안 조건을 검토합니다.

- 북미:

```
$ az vm image terms show --urn redhat:rh-ocp-worker:rh-ocp-worker:<version>
```

- EMEA:

```
$ az vm image terms show --urn redhat-limited:rh-ocp-worker:rh-ocp-worker:<version>
```

4. 다음 명령 중 하나를 실행하여 제품 약관에 동의합니다.

- 북미:

```
$ az vm image terms accept --urn redhat:rh-ocp-worker:rh-ocp-worker:<version>
```

- EMEA:

```
$ az vm image terms accept --urn redhat-limited:rh-ocp-worker:rh-ocp-worker:<version>
```

5. 제안의 이미지 세부 정보를 기록하고 이를 사용하여 **06_workers.json** Azure Resource Manager(ARM) 템플릿을 업데이트합니다.

6. **id** 매개변수를 삭제하고 제안의 값을 사용하여 **offer**,**publisher**,**sku** 및 **version** 매개변수를 추가하여 **storageProfile.imageReference** 필드를 업데이트합니다. "Azure에서 추가 작업자 머신 생성" 섹션에서 샘플 템플릿을 찾을 수 있습니다.

5.10.6. 설치 프로그램 받기

OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

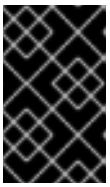
- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

프로세스

1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.
2. 인프라 공급자를 선택합니다.
3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.

중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.



중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 OpenShift Container Platform 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

5.10.7. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 **core** 사용자의 **~/.ssh/authorized_keys** 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 **core**로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다. SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. **./openshift-install gather** 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

참고

AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: **~/.ssh/id_ed25519**)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 **'~/.ssh** 디렉터리에 있는지 확인하십시오.

참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 **ed25519** 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 **rsa** 또는 **ecdsa** 알고리즘을 사용하는 키를 생성합니다.

2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 **~/.ssh/id_ed25519.pub** 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 **./openshift-install gather** 명령을 사용하려는 경우 필요합니다.

참고

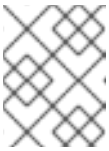
일부 배포에서는 **~/.ssh/id_rsa** 및 **~/.ssh/id_dsa**와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. **ssh-agent** 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```



참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. **ssh-agent**에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: **~/.ssh/id_ed25519**).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다. 프로비저닝하는 인프라에 클러스터를 설치하는 경우 설치 프로그램에 키를 제공해야 합니다.

5.10.8. Azure용 설치 파일 생성

사용자 프로비저닝 인프라를 사용하는 OpenShift Container Platform을 Microsoft Azure에 설치하려면 설치 프로그램에서 클러스터를 배포하는 데 필요한 파일을 생성하고 클러스터가 사용할 시스템만을 생성하도록 파일을 수정해야 합니다. **install-config.yaml** 파일, Kubernetes 매니페스트 및 Ignition 구성 파일을 생성하고 사용자 지정합니다. 또한 설치 준비 단계에서 별도의 **var** 파티션을 먼저 설정할 수 있는 옵션이 있습니다.

5.10.8.1. 선택 사항: 별도의 /var 파티션 만들기

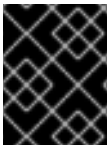
OpenShift Container Platform의 디스크 파티션 설정은 설치 프로그램에 맡기는 것이 좋습니다. 그러나 확장하려는 파일 시스템의 일부에 별도의 파티션을 생성해야 하는 경우가 있습니다.

OpenShift 컨테이너 플랫폼은 **/var** 파티션 또는 **/var**의 하위 디렉터리 중 하나에 스토리지를 연결하는 단일 파티션의 추가를 지원합니다. 예를 들면 다음과 같습니다.

- **/var/lib/containers**: 시스템에 더 많은 이미지와 컨테이너가 추가됨에 따라 확장될 수 있는 컨테이너 관련 콘텐츠를 보관합니다.
- **/var/lib/etcd**: etcd 스토리지의 성능 최적화와 같은 목적으로 별도로 보관할 데이터를 보관합니다.
- **/var**: 감사 등의 목적에 맞게 별도로 분리하여 보관해야 하는 데이터를 보관합니다.

/var 디렉터리의 콘텐츠를 별도로 저장하면 필요에 따라 해당 영역에 대한 스토리지 확장을 보다 용이하게 하고 나중에 OpenShift Container Platform을 다시 설치하여 해당 데이터를 그대로 보존할 수 있습니다. 이 방법을 사용하면 모든 컨테이너를 다시 가져올 필요가 없으며 시스템을 업데이트할 때 대용량 로그 파일을 복사할 필요도 없습니다.

RHCOS(Red Hat Enterprise Linux CoreOS)를 새로 설치하기 전에 **/var가 있어야 하므로 다음 절차에서는 OpenShift Container Platform 설치의 openshift-install 준비 단계 중에 삽입되는 머신 구성 매니페스트를 생성하여 별도의 /var** 파티션을 설정합니다.



중요

이 절차에서 별도의 **/var** 파티션을 생성하기 위해 단계에 따라 이 섹션의 뒷부분에서 설명한 대로 Kubernetes 매니페스트 및 Ignition 구성 파일을 다시 생성할 필요가 없습니다.

프로세스

1. OpenShift Container Platform 설치 파일을 저장할 디렉터리를 만듭니다.

```
$ mkdir $HOME/clusterconfig
```

2. **openshift-install**을 실행하여 **manifest** 및 **openshift** 하위 디렉터리에 파일 세트를 만듭니다. 프롬프트가 표시되면 시스템 질문에 대답합니다.

```
$ openshift-install create manifests --dir $HOME/clusterconfig
```

출력 예

```
? SSH Public Key ...
INFO Credentials loaded from the "myprofile" profile in file "/home/myuser/.aws/credentials"
INFO Consuming Install Config from target directory
INFO Manifests created in: $HOME/clusterconfig/manifests and
$HOME/clusterconfig/openshift
```

3. 선택 사항: 설치 프로그램이 **clusterconfig/openshift** 디렉터리에 매니페스트를 생성했는지 확인합니다.

```
$ ls $HOME/clusterconfig/openshift/
```

출력 예

```
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

4. 추가 파티션을 구성하는 Butane 구성을 생성합니다. 예를 들어 파일 이름을 **$HOME/clusterconfig/98-var-partition.bu** 로 지정하고, 디스크 장치 이름을 **worker** 시스템의 스토리지 장치 이름으로 변경하고 스토리지 크기를 적절하게 설정합니다. 이 예에서는 **/var** 디렉토리를 별도의 파티션에 배치합니다.

```
variant: openshift
version: 4.9.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
```

```
disks:
- device: /dev/<device_name> 1
  partitions:
  - label: var
    start_mib: <partition_start_offset> 2
    size_mib: <partition_size> 3
filesystems:
  - device: /dev/disk/by-partlabel/var
    path: /var
    format: xfs
    mount_options: [defaults, prjquota] 4
    with_mount_unit: true
```

1. 파티션을 설정해야하는 디스크 저장 장치 이름입니다.
2. 데이터 파티션을 부트 디스크에 추가할 때 최소 25000MiB(메비 바이트)가 권장됩니다. 루트 파일 시스템은 지정된 오프셋까지 사용 가능한 모든 공간을 채우기 위해 자동으로 크기가 조정됩니다. 값이 지정되지 않거나 지정된 값이 권장 최소값보다 작으면 생성되는 루트 파일 시스템의 크기가 너무 작아지고 RHCOS를 나중에 다시 설치할 때 데이터 파티션의 첫 번째 부분을 덮어 쓸 수 있습니다.
3. 데이터 파티션의 크기(MB)입니다.
4. 컨테이너 스토리지에 사용되는 파일 시스템에 **prjquota** 마운트 옵션을 활성화해야 합니다.

참고

별도의 **/var** 파티션을 만들 때 다른 인스턴스 유형에 동일한 장치 이름이 없는 경우 작업자 노드에 다른 인스턴스 유형을 사용할 수 없습니다.

5. Butane 구성에서 매니페스트를 생성하여 **clusterconfig/openshift** 디렉터리에 저장합니다. 예를 들어 다음 명령을 실행합니다.

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

6. **openshift-install**을 다시 실행하여 **manifest** 및 **openshift** 하위 디렉터리의 파일 세트에서 Ignition 구성을 만듭니다.

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

이제 Ignition 구성 파일을 설치 절차에 대한 입력으로 사용하여 RHCOS (Red Hat Enterprise Linux CoreOS) 시스템을 설치할 수 있습니다.

5.10.8.2. 설치 구성 파일 만들기

Microsoft Azure에 설치하는 OpenShift Container Platform 클러스터를 사용자 지정할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

- 서브스크립션 수준에서 서비스 주체 권한을 획득합니다.

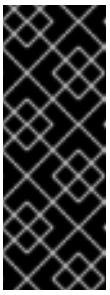
프로세스

1. **install-config.yaml** 파일을 생성합니다.

   a. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

   ```
   $ ./openshift-install create install-config --dir <installation_directory> 1
   ```

   1 **<installation_directory>**는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.

   중요

   비어 있는 디렉터리를 지정합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

   b. 화면에 나타나는 지시에 따라 클라우드에 대한 구성 세부 사항을 입력합니다.

      i. 선택사항: 클러스터 시스템에 액세스하는 데 사용할 SSH 키를 선택합니다.

      참고

      설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

      ii. 대상 플랫폼으로 **azure**를 선택합니다.

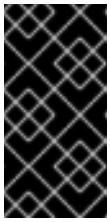
      iii. 컴퓨터에 Microsoft Azure 프로필이 저장되어 있지 않은 경우 서비스크립션 및 서비스 주체에 대해 다음 Azure 매개변수 값을 지정합니다.

      - **azure subscription id** 클러스터에 사용할 서브스크립션 ID입니다. 계정 출력의 **id** 값을 지정하십시오.
      - **azure tenant id** 테넌트 ID. 계정 출력의 **tenantId** 값을 지정하십시오.
      - **azure service principal client id** 서비스 주체의 **appId** 매개변수 값.
      - **azure service principal client secret** 서비스 주체의 **password** 매개변수 값.

      iv. 클러스터를 배포할 리전을 선택합니다.

      v. 클러스터를 배포할 기본 도메인을 선택합니다. 기본 도메인은 클러스터용으로 생성한 Azure DNS 영역에 해당합니다.

      vi. 클러스터를 설명할 수 있는 이름을 입력합니다.



중요

공용 끝점을 통해 사용할 수 있는 모든 Azure 리소스에는 리소스 이름 제한 사항이 적용되며, 특정 용어를 사용하는 리소스를 생성할 수 없습니다. Azure가 제한하는 용어 목록은 Azure 문서의 예약된 리소스 이름 오류 해결을 참조하십시오.

vii. Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 붙여넣습니다.

c. 선택사항: 클러스터가 컴퓨팅 시스템을 프로비저닝하지 않도록 하려면 **compute** 풀에 대해 **replicas**를 **0**으로 설정하도록 결과 **install-config.yaml** 파일을 편집하여 컴퓨팅 풀을 비우십시오.

```
compute:
- hyperthreading: Enabled
  name: worker
  platform: {}
  replicas: 0 1
```

1 **0**으로 설정합니다.

2. **install-config.yaml** 파일을 수정합니다. 사용 가능한 매개변수에 대한 자세한 정보는 "설치 구성 매개변수" 섹션에서 확인할 수 있습니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.

중요

install-config.yaml 파일은 설치 과정에서 사용됩니다. 이 파일을 재사용하려면 지금 백업해야 합니다.

5.10.8.3. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 HTTP 또는 HTTPS 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 OpenShift Container Platform 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 API에 대한 호출을 포함하여 모든 클러스터 발신(Egress) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.

참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 Red Hat OpenStack Platform (RHOSP)에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. 클러스터 외부에서 HTTP 연결을 구축하는 데 사용할 프록시 URL입니다. URL 스키마는 **http**여야 합니다.
2. 클러스터 외부에서 HTTPS 연결을 구축하는 데 사용할 프록시 URL입니다.
3. 대상 도메인 이름, IP 주소 또는 프록시에서 제외할 기타 네트워크 CIDR로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. ***를 사용하여 모든 대상에 대해 프록시를 바이패스합니다.
4. 이 값을 제공하면 설치 프로그램에서 추가 CA 인증서를 보유할 **openshift -config 네임스페이스에 user-ca- bundle** 이라는 구성 맵을 생성합니다. **additionalTrustBundle** 및 하나 이상의 프록시 설정을 제공하는 경우 **프록시** 오브젝트는 **trustedCA** 필드의 **user-ca-bundle** 구성 맵을 참조하도록 구성됩니다. 그러면 Cluster Network Operator에서 **trustedCA** 매개변수에 대해 지정된 콘텐츠를 RHCOS 신뢰 번들과 병합하는 **trusted-ca-bundle** 구성 맵을 생성합니다. 프록시의 ID 인증서를 RHCOS 트러스트 번들에 있는 기관에서 서명하지 않은 경우 **additionalTrustBundle** 필드가 있어야 합니다.

참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 OpenShift Container Platform을 설치할 때 참조하십시오.

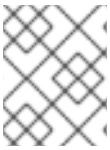
제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 nil이 됩니다.

참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

5.10.8.4. ARM 템플릿의 공통 변수 내보내기

Microsoft Azure에서 사용자 제공 인프라 설치를 지원하기 위해 제공된 ARM(Azure Resource Manager) 템플릿과 함께 사용되는 공통 변수 세트를 내보내야 합니다.



참고

특정 ARM 템플릿에는 추가적으로 내보낸 변수도 필요할 수 있습니다. 관련 프로시저에서 자세히 설명합니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

절차

1. 제공된 ARM 템플릿이 사용할 **install-config.yaml**에 있는 공통 변수를 내보냅니다.

```
$ export CLUSTER_NAME=<cluster_name> 1
$ export AZURE_REGION=<azure_region> 2
$ export SSH_KEY=<ssh_key> 3
$ export BASE_DOMAIN=<base_domain> 4
$ export BASE_DOMAIN_RESOURCE_GROUP=<base_domain_resource_group> 5
```

1. **install-config.yaml** 파일의 **.metadata.name** 속성 값입니다.
2. 를 배치할 리전(예: **centralus**). **install-config.yaml** 파일의 **.platform.azure.region** 속성 값입니다.
3. 문자열 형태의 SSH RSA 공개 키 파일입니다. 공백이 포함되어 있으므로 SSH 키를 따옴표로 묶어야 합니다. **install-config.yaml** 파일의 **.sshKey** 속성 값입니다.
4. 클러스터를 배포할 기본 도메인. 기본 도메인은 클러스터용으로 생성한 퍼블릭 DNS 영역에 해당합니다. **install-config.yaml** 파일의 **.baseDomain** 속성 값입니다.
5. encapsulated 퍼블릭 DNS 영역이 존재하는 리소스 그룹입니다. **install-config.yaml** 파일의 **.platform.azure.baseDomainResourceGroupName** 속성 값입니다.

예를 들면 다음과 같습니다.

```
$ export CLUSTER_NAME=test-cluster
$ export AZURE_REGION=centralus
$ export SSH_KEY="ssh-rsa xxx/xxx/xxx= user@email.com"
$ export BASE_DOMAIN=example.com
$ export BASE_DOMAIN_RESOURCE_GROUP=ocp-cluster
```

2. kubeadmin 인증 정보를 내보냅니다.

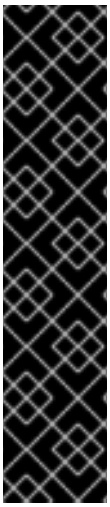
```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

5.10.8.5. Kubernetes 매니페스트 및 Ignition 설정 파일 생성

일부 클러스터 정의 파일을 수정하고 클러스터 시스템을 수동으로 시작해야 하므로 클러스터가 시스템을 구성하는 데 필요한 Kubernetes 매니페스트 및 Ignition 구성 파일을 사용자가 생성해야 합니다.

설치 구성 파일은 Kubernetes 매니페스트로 변환됩니다. 매니페스트는 나중에 클러스터 머신을 구성하는 데 사용되는 Ignition 구성 파일로 래핑됩니다.



중요

- OpenShift Container Platform 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *만료된 컨트롤 플레인 인증서에서 복구* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램을 가져오셨습니다.
- **install-config.yaml** 설치 구성 파일을 생성하셨습니다.

프로세스

1. OpenShift Container Platform 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터에 대한 Kubernetes 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>**는 사용자가 만든 **install-config.yaml** 파일이 포함된 설치 디렉터리를 지정합니다.

2. 컨트롤 플레인 시스템을 정의하는 Kubernetes 매니페스트 파일을 제거합니다.

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

이러한 파일을 제거하면 클러스터가 컨트롤 플레인 시스템을 자동으로 생성하지 못합니다.

3. 작업자 시스템을 정의하는 Kubernetes 매니페스트 파일을 제거합니다.

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

작업자 시스템은 사용자가 직접 생성하고 관리하기 때문에 초기화할 필요가 없습니다.

4. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes 매니페스트 파일의 **mastersSchedulable** 매개변수가 **false**로 설정되어 있는지 확인합니다. 이 설정으로 인해 컨트롤 플레인 머신에서 포드가 예약되지 않습니다.

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** 파일을 엽니다.

   b. **mastersSchedulable** 매개변수를 찾아서 값을 **False**로 설정되어 있는지 확인합니다.

   c. 파일을 저장하고 종료합니다.

5. 선택사항: Ingress Operator가 사용자 대신 DNS 레코드를 생성하지 못하도록 하려면 **<installation_directory>/manifests/cluster-dns-02-config.yml** DNS 구성 파일에서 **privateZone** 및 **publicZone** 섹션을 제거합니다.

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

   1 2 이 섹션을 완전히 제거합니다.

   제거한 경우 나중에 인그레스 DNS 레코드를 수동으로 추가해야 합니다.

6. 사용자 프로비저닝 인프라에서 Azure를 구성할 때, ARM(Azure Resource Manager) 템플릿에서 나중에 사용할 수 있도록 매니페스트 파일에 정의된 일부 공통 변수를 내보내야 합니다.

   a. 다음 명령을 사용하여 인프라 ID를 내보냅니다.

```
$ export INFRA_ID=<infra_id> 1
```

   1 OpenShift Container Platform 클러스터에 **<cluster_name>-<random_string>** 형식으로 식별자(**INFRA_ID**)가 할당되었습니다. 제공된 ARM 템플릿을 사용해서 생성된 대부분의 리소스에 대해 기본 이름으로 사용됩니다. **manifests/cluster-infrastructure-02-config.yml** 파일의 **.status.infrastructureName** 속성 값입니다.

   b. 다음 명령을 사용하여 리소스 그룹을 내보냅니다:

```
$ export RESOURCE_GROUP=<resource_group> 1
```

   1 이 Azure 배포에서 생성된 모든 리소스는 리소스 그룹의 일부로 존재합니다. 또한 리소스 그룹 이름은 **<cluster_name>-<random_string>-rg** 형식의 **INFRA_ID**를 기반으로 합니다. **manifests/cluster-infrastructure-02-config.yml** 파일의 **.status.platformStatus.azure.resourceGroupName** 속성 값입니다.

7. Ignition 구성 파일을 생성하려면 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행합니다.

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>**는 동일한 설치 디렉터리를 지정합니다.

설치 디렉터리의 부트스트랩, 컨트롤 플레인 및 컴퓨팅 노드에 대한 Ignition 구성 파일이 생성됩니다. **kubeadmin-password** 및 **kubeconfig** 파일은 **./<installation_directory>/auth** 디렉터리에 생성됩니다.

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

5.10.9. Azure 리소스 그룹 생성

Microsoft Azure 리소스 그룹과 해당 리소스 그룹의 ID를 생성해야 합니다. 두 가지 모두 Azure에 OpenShift Container Platform 클러스터를 설치하는 동안 사용됩니다.

사전 요구 사항

- Azure 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.

프로세스

1. 지원되는 Azure 리전에서 리소스 그룹을 생성합니다.

```
$ az group create --name ${RESOURCE_GROUP} --location ${AZURE_REGION}
```

2. 리소스 그룹에 대한 Azure ID를 생성합니다.

```
$ az identity create -g ${RESOURCE_GROUP} -n ${INFRA_ID}-identity
```

클러스터의 Operators에게 필요한 액세스 권한을 부여하는 데 사용됩니다. 예를 들어 Ingress Operator는 공용 IP와 로드 밸런서를 생성할 수 있습니다. Azure ID를 역할에 할당해야 합니다.

3. Azure ID에 Contributor 역할을 부여합니다.

   a. Azure 역할 할당에 필요한 다음 변수를 내보냅니다.

```
$ export PRINCIPAL_ID=`az identity show -g ${RESOURCE_GROUP} -n ${INFRA_ID}-identity --query principalId --out tsv`
```

```
$ export RESOURCE_GROUP_ID=`az group show -g ${RESOURCE_GROUP} --query id --out tsv`
```

   b. Contributor 역할을 ID에 할당합니다.

```
$ az role assignment create --assignee "${PRINCIPAL_ID}" --role 'Contributor' --scope
"${RESOURCE_GROUP_ID}"
```

5.10.10. RHCOS 클러스터 이미지 및 부트스트랩 Ignition 구성 파일 업로드

Azure 클라이언트는 로컬로 존재하는 파일을 기반으로 한 배포를 지원하지 않습니다. 따라서 RHCOS VHD(가상 하드 디스크) 클러스터 이미지와 부트스트랩 Ignition 구성 파일을 스토리지 컨테이너에 복사하여 저장해야 배포 중에 액세스할 수 있습니다.

사전 요구 사항

- Azure 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.

프로세스

1. VHD 클러스터 이미지를 저장할 Azure 스토리지 계정을 생성합니다.

```
$ az storage account create -g ${RESOURCE_GROUP} --location ${AZURE_REGION} --
name ${CLUSTER_NAME}sa --kind Storage --sku Standard_LRS
```

주의

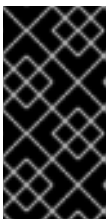
Azure 스토리지 계정 이름은 3자에서 24자 사이여야 하며 숫자와 소문자만 사용해야 합니다. **CLUSTER_NAME** 변수가 이러한 제한 사항을 따르지 않으면 Azure 스토리지 계정 이름을 수동으로 정의해야 합니다. Azure 스토리지 계정 이름 제한 사항에 대한 자세한 내용은 Azure 문서의 스토리지 계정 이름 오류 해결을 참조하십시오.

2. 스토리지 계정 키를 환경 변수 형태로 내보냅니다.

```
$ export ACCOUNT_KEY=`az storage account keys list -g ${RESOURCE_GROUP} --
account-name ${CLUSTER_NAME}sa --query "[0].value" -o tsv`
```

3. 사용할 RHCOS 버전을 선택하고 VHD의 URL을 환경 변수로 내보냅니다.

```
$ export VHD_URL=`curl -s https://raw.githubusercontent.com/openshift/installer/release-
4.9/data/data/rhcos.json | jq -r .azure.url`
```



중요

RHCOS 이미지는 OpenShift Container Platform 릴리스에 따라 변경되지 않을 수 있습니다. 설치하는 OpenShift Container Platform 버전 이하에서 가장 높은 버전의 이미지를 지정해야 합니다. 지원되는 경우 OpenShift Container Platform 버전과 일치하는 이미지 버전을 사용합니다.

4. VHD용 스토리지 컨테이너를 생성합니다.

```
$ az storage container create --name vhd --account-name ${CLUSTER_NAME}sa --account-key ${ACCOUNT_KEY}
```

5. 선택한 VHD를 Blob에 복사합니다.

```
$ az storage blob copy start --account-name ${CLUSTER_NAME}sa --account-key ${ACCOUNT_KEY} --destination-blob "rhcos.vhd" --destination-container vhd --source-uri "${VHD_URL}"
```

6. Blob 스토리지 컨테이너를 만들고 생성된 **bootstrap.ign** 파일을 업로드합니다.

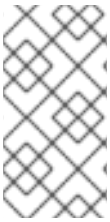
```
$ az storage container create --name files --account-name ${CLUSTER_NAME}sa --account-key ${ACCOUNT_KEY} --public-access blob
```

```
$ az storage blob upload --account-name ${CLUSTER_NAME}sa --account-key ${ACCOUNT_KEY} -c "files" -f "<installation_directory>/bootstrap.ign" -n "bootstrap.ign"
```

5.10.11. DNS 영역 생성 예

사용자 프로비저닝 인프라를 사용하는 클러스터에는 DNS 레코드가 필요합니다. 시나리오에 맞는 DNS 전략을 선택해야 합니다.

이 예에서는 Azure의 DNS 솔루션이 사용되므로 외부 (인터넷) 가시성을 위한 새로운 퍼블릭 DNS 영역과 내부 클러스터 확인을 위한 프라이빗 DNS 영역을 만듭니다.



참고

퍼블릭 DNS 영역은 클러스터 배포와 동일한 리소스 그룹에 존재할 필요는 없으며, 원하는 기본 도메인에 대해 조직에 이미 존재할 수도 있습니다. 이 경우 퍼블릭 DNS 영역 생성을 건너뛸 수 있습니다. 이전에 생성한 설치 구성이 해당 시나리오를 반영하는지 확인하십시오.

사전 요구 사항

- Azure 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.

프로세스

1. **BASE_DOMAIN_RESOURCE_GROUP** 환경 변수로 내보낸 리소스 그룹에 새 퍼블릭 DNS 영역을 생성합니다.

```
$ az network dns zone create -g ${BASE_DOMAIN_RESOURCE_GROUP} -n ${CLUSTER_NAME}.${BASE_DOMAIN}
```

   이미 존재하는 퍼블릭 DNS 영역을 사용 중인 경우에는 이 단계를 건너뛸 수 있습니다.

2. 이 배포의 나머지 부분과 동일한 리소스 그룹에 프라이빗 DNS 영역을 생성합니다.

```
$ az network private-dns zone create -g ${RESOURCE_GROUP} -n
${CLUSTER_NAME}.${BASE_DOMAIN}
```

해당 섹션을 방문하여 Azure에서 퍼블릭 DNS 영역 구성 에 대해 자세히 알아볼 수 있습니다.

5.10.12. Azure에서 VNet 생성

Microsoft Azure에 OpenShift Container Platform 클러스터가 사용할 가상 네트워크(VNet)를 만들어야 합니다. 요구사항에 맞춰 VNet을 사용자 지정할 수 있습니다. VNet을 생성하는 한 가지 방법은 제공된 ARM(Azure Resource Manager) 템플릿을 수정하는 것입니다.

참고

Azure 인프라를 생성하는 데 제공된 ARM 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- Azure 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.

프로세스

1. 이 항목의 **VNet에 대한 ARM 템플릿** 섹션에서 템플릿을 복사하여 클러스터의 설치 디렉터리에 **01_vnet.json**으로 저장합니다. 이 템플릿은 클러스터에 필요한 VNet을 설명합니다.
2. **az** CLI를 사용하여 배포를 생성합니다.

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/01_vnet.json" \
  --parameters baseName="${INFRA_ID}" 1
```

1 리소스 이름에 사용될 기본 이름; 일반적으로 클러스터의 인프라 ID입니다.

3. VNet 템플릿을 프라이빗 DNS 영역에 연결합니다.

```
$ az network private-dns link vnet create -g ${RESOURCE_GROUP} -z
${CLUSTER_NAME}.${BASE_DOMAIN} -n ${INFRA_ID}-network-link -v "${INFRA_ID}-vnet"
-e false
```

5.10.12.1. VNet용 ARM 템플릿

다음 ARM(Azure Resource Manager) 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 VNet을 배포할 수 있습니다.

예 5.1. 01_vnet.json ARM 템플릿

```
{
  "$schema" : "https://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
```

```
  "contentVersion" : "1.0.0.0",
  "parameters" : {
    "baseName" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Base name to be used in resource names (usually the cluster's Infra ID)"
      }
    }
  },
  "variables" : {
    "location" : "[resourceGroup().location]",
    "virtualNetworkName" : "[concat(parameters('baseName'), '-vnet')]",
    "addressPrefix" : "10.0.0.0/16",
    "masterSubnetName" : "[concat(parameters('baseName'), '-master-subnet')]",
    "masterSubnetPrefix" : "10.0.0.0/24",
    "nodeSubnetName" : "[concat(parameters('baseName'), '-worker-subnet')]",
    "nodeSubnetPrefix" : "10.0.1.0/24",
    "clusterNsgName" : "[concat(parameters('baseName'), '-nsg')]"
  },
  "resources" : [
    {
      "apiVersion" : "2018-12-01",
      "type" : "Microsoft.Network/virtualNetworks",
      "name" : "[variables('virtualNetworkName')]",
      "location" : "[variables('location')]",
      "dependsOn" : [
        "[concat('Microsoft.Network/networkSecurityGroups/', variables('clusterNsgName'))]"
      ],
      "properties" : {
        "addressSpace" : {
          "addressPrefixes" : [
            "[variables('addressPrefix')]"
          ]
        },
        "subnets" : [
          {
            "name" : "[variables('masterSubnetName')]",
            "properties" : {
              "addressPrefix" : "[variables('masterSubnetPrefix')]",
              "serviceEndpoints": [],
              "networkSecurityGroup" : {
                "id" : "[resourceId('Microsoft.Network/networkSecurityGroups',
variables('clusterNsgName'))]"
              }
            }
          },
          {
            "name" : "[variables('nodeSubnetName')]",
            "properties" : {
              "addressPrefix" : "[variables('nodeSubnetPrefix')]",
              "serviceEndpoints": [],
              "networkSecurityGroup" : {
                "id" : "[resourceId('Microsoft.Network/networkSecurityGroups',
variables('clusterNsgName'))]"
              }
```

```
          }
        }
      ]
    }
  },
  {
    "type" : "Microsoft.Network/networkSecurityGroups",
    "name" : "[variables('clusterNsgName')]",
    "apiVersion" : "2018-10-01",
    "location" : "[variables('location')]",
    "properties" : {
      "securityRules" : [
        {
          "name" : "apiserver_in",
          "properties" : {
            "protocol" : "Tcp",
            "sourcePortRange" : "*",
            "destinationPortRange" : "6443",
            "sourceAddressPrefix" : "*",
            "destinationAddressPrefix" : "*",
            "access" : "Allow",
            "priority" : 101,
            "direction" : "Inbound"
          }
        }
      ]
    }
  }
 ]
}
```

5.10.13. Azure 인프라에 대한 RHCOS 클러스터 이미지 배포

OpenShift Container Platform 노드에 대해서는 Microsoft Azure에 유효한 RHCOS(Red Hat Enterprise Linux CoreOS) 이미지를 사용해야 합니다.

사전 요구 사항

- Azure 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.
- RHCOS VHD(가상 하드 디스크) 클러스터 이미지를 Azure 스토리지 컨테이너에 저장하십시오.
- 부트스트랩 ignition 구성 파일을 Azure 스토리지 컨테이너에 저장하십시오.

프로세스

1. 이 항목의 **이미지 스토리지에 대한 ARM 템플릿** 섹션에서 템플릿을 복사하여 클러스터의 설치 디렉터리에 **02_storage.json**으로 저장합니다. 이 템플릿은 클러스터에 필요한 이미지 스토리지를 설명합니다.
2. RHCOS VHD BLOB URL을 변수 형태로 내보냅니다.

```
$ export VHD_BLOB_URL=`az storage blob url --account-name ${CLUSTER_NAME}sa --
account-key ${ACCOUNT_KEY} -c vhd -n "rhcos.vhd" -o tsv`
```

3. 클러스터 이미지를 배포합니다.

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/02_storage.json" \
  --parameters vhdBlobURL="${VHD_BLOB_URL}" \ 1
  --parameters baseName="${INFRA_ID}" 2
```

1 마스터 및 작업자 시스템을 생성하는 데 사용되는 RHCOS VHD의 Blob URL.

2 리소스 이름에 사용될 기본 이름; 일반적으로 클러스터의 인프라 ID입니다.

5.10.13.1. 이미지 스토리지를 위한 ARM 템플릿

다음 ARM(Azure Resource Manager) 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 저장된 RHCOS(Red Hat Enterprise Linux CoreOS) 이미지를 배포할 수 있습니다.

예 5.2. 02_storage.json ARM 템플릿

```
{
  "$schema" : "https://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
  "contentVersion" : "1.0.0.0",
  "parameters" : {
    "baseName" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Base name to be used in resource names (usually the cluster's Infra ID)"
      }
    },
    "vhdBlobURL" : {
      "type" : "string",
      "metadata" : {
        "description" : "URL pointing to the blob where the VHD to be used to create master and
worker machines is located"
      }
    }
  },
  "variables" : {
    "location" : "[resourceGroup().location]",
    "imageName" : "[concat(parameters('baseName'), '-image')]"
  },
  "resources" : [
    {
      "apiVersion" : "2018-06-01",
      "type": "Microsoft.Compute/images",
      "name": "[variables('imageName')]",
      "location" : "[variables('location')]",
      "properties": {
        "storageProfile": {
```

```
            "osDisk": {
              "osType": "Linux",
              "osState": "Generalized",
              "blobUri": "[parameters('vhdBlobURL')]",
              "storageAccountType": "Standard_LRS"
            }
          }
        }
      }
    ]
  }
```

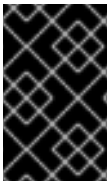
5.10.14. 사용자 프로비저닝 인프라에 대한 네트워킹 요구사항

모든 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템이 부팅 중에 Ignition 구성 파일을 가져오려면 **initramfs**에 네트워킹을 구성해야 합니다.

5.10.14.1. 네트워크 연결 요구사항

OpenShift Container Platform 클러스터 구성 요소가 통신할 수 있도록 시스템 간 네트워크 연결을 구성해야 합니다. 각 시스템에서 클러스터에 있는 다른 모든 시스템의 호스트 이름을 확인할 수 있어야 합니다.

이 섹션에서는 필요한 포트에 대해 자세히 설명합니다.



중요

연결된 OpenShift Container Platform 환경에서 모든 노드는 플랫폼 컨테이너의 이미지를 가져오고 Red Hat에 원격 측정 데이터를 제공하기 위해 인터넷에 액세스할 수 있어야 합니다.

표 5.37. 모든 시스템 간 통신에 사용되는 포트

프로토콜	포트	설명
ICMP	해당 없음	네트워크 연결성 테스트
TCP	**1936**	메트릭
	9000-**9999**	**9100**-**9101** 포트의 노드 내보내기 및 **9099** 포트의 Cluster Version Operator를 포함한 호스트 수준 서비스.
	10250-**10259**	Kubernetes에서 예약하는 기본 포트
	10256	openshift-sdn
UDP	**4789**	VXLAN 및 Geneve
	6081	VXLAN 및 Geneve

프로토콜	포트	설명
	9000-9999	**9100-9101** 포트의 노드 내보내기를 포함한 호스트 수준 서비스.
	500	IPsec IKE 패킷
	4500	IPsec NAT-T 패킷
TCP/UDP	**30000-32767**	Kubernetes 노드 포트
ESP	해당 없음	IPsec Encapsulating Security Payload (ESP)

표 5.38. 모든 시스템과 컨트롤 플레인 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**6443**	Kubernetes API

표 5.39. 컨트롤 플레인 머신 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**2379-2380**	etcd 서버 및 피어 포트

5.10.15. Azure에서 네트워킹 및 로드 밸런싱 구성 요소 생성

Microsoft Azure에서 OpenShift Container Platform 클러스터가 사용할 네트워킹 및 로드 밸런싱을 구성해야 합니다. 이러한 구성 요소를 생성하는 한 가지 방법은 제공된 ARM(Azure Resource Manager) 템플릿을 수정하는 것입니다.

참고

Azure 인프라를 생성하는 데 제공된 ARM 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- Azure 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.
- Azure에서 VNet 및 관련 서브넷을 생성하고 구성하십시오.

프로세스

1. 이 항목의 **네트워크 및 로드 밸런서에 대한 ARM 템플릿** 섹션에서 템플릿을 복사하여 클러스터의 설치 디렉터리에 **03_infra.json**으로 저장합니다. 이 템플릿은 클러스터에 필요한 네트워킹 및 로드 밸런싱 개체를 설명합니다.

2. **az** CLI를 사용하여 배포를 생성합니다.

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/03_infra.json" \
  --parameters privateDNSZoneName="${CLUSTER_NAME}.${BASE_DOMAIN}" \ 1
  --parameters baseName="${INFRA_ID}" 2
```

1 프라이빗 DNS 영역의 이름입니다.

2 리소스 이름에 사용될 기본 이름; 일반적으로 클러스터의 인프라 ID입니다.

3. API 공용 로드 밸런서의 퍼블릭 영역에 **API** DNS 레코드를 만듭니다. **${BASE_DOMAIN_RESOURCE_GROUP}** 변수는 퍼블릭 DNS 영역이 존재하는 리소스 그룹을 가리켜야 합니다.

   a. 다음 변수를 내보냅니다.

```
$ export PUBLIC_IP=`az network public-ip list -g ${RESOURCE_GROUP} --query "[?
name=='${INFRA_ID}-master-pip'] | [0].ipAddress" -o tsv`
```

   b. 새 퍼블릭 영역에 **api** DNS 레코드를 생성합니다.

```
$ az network dns record-set a add-record -g ${BASE_DOMAIN_RESOURCE_GROUP} -
z ${CLUSTER_NAME}.${BASE_DOMAIN} -n api -a ${PUBLIC_IP} --ttl 60
```

   기존 퍼블릭 영역에 클러스터를 추가하는 경우 대신 **api** DNS 레코드를 생성할 수 있습니다.

```
$ az network dns record-set a add-record -g ${BASE_DOMAIN_RESOURCE_GROUP} -
z ${BASE_DOMAIN} -n api.${CLUSTER_NAME} -a ${PUBLIC_IP} --ttl 60
```

5.10.15.1. 네트워크 및 로드 밸런서에 대한 ARM 템플릿

다음 ARM(Azure Resource Manager) 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 네트워킹 개체 및 로드 밸런서를 배포할 수 있습니다.

예 5.3. 03_infra.json ARM 템플릿

```
{
  "$schema" : "https://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
  "contentVersion" : "1.0.0.0",
  "parameters" : {
    "baseName" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Base name to be used in resource names (usually the cluster's Infra ID)"
      }
    },
```

```
    "privateDNSZoneName" : {
      "type" : "string",
      "metadata" : {
        "description" : "Name of the private DNS zone"
      }
    }
  },
  "variables" : {
    "location" : "[resourceGroup().location]",
    "virtualNetworkName" : "[concat(parameters('baseName'), '-vnet')]",
    "virtualNetworkID" : "[resourceId('Microsoft.Network/virtualNetworks',
variables('virtualNetworkName'))]",
    "masterSubnetName" : "[concat(parameters('baseName'), '-master-subnet')]",
    "masterSubnetRef" : "[concat(variables('virtualNetworkID'), '/subnets/',
variables('masterSubnetName'))]",
    "masterPublicIpAddressName" : "[concat(parameters('baseName'), '-master-pip')]",
    "masterPublicIpAddressID" : "[resourceId('Microsoft.Network/publicIPAddresses',
variables('masterPublicIpAddressName'))]",
    "masterLoadBalancerName" : "[concat(parameters('baseName'), '-public-lb')]",
    "masterLoadBalancerID" : "[resourceId('Microsoft.Network/loadBalancers',
variables('masterLoadBalancerName'))]",
    "internalLoadBalancerName" : "[concat(parameters('baseName'), '-internal-lb')]",
    "internalLoadBalancerID" : "[resourceId('Microsoft.Network/loadBalancers',
variables('internalLoadBalancerName'))]",
    "skuName": "Standard"
  },
  "resources" : [
    {
      "apiVersion" : "2018-12-01",
      "type" : "Microsoft.Network/publicIPAddresses",
      "name" : "[variables('masterPublicIpAddressName')]",
      "location" : "[variables('location')]",
      "sku": {
        "name": "[variables('skuName')]"
      },
      "properties" : {
        "publicIPAllocationMethod" : "Static",
        "dnsSettings" : {
          "domainNameLabel" : "[variables('masterPublicIpAddressName')]"
        }
      }
    },
    {
      "apiVersion" : "2018-12-01",
      "type" : "Microsoft.Network/loadBalancers",
      "name" : "[variables('masterLoadBalancerName')]",
      "location" : "[variables('location')]",
      "sku": {
        "name": "[variables('skuName')]"
      },
      "dependsOn" : [
        "[concat('Microsoft.Network/publicIPAddresses/', variables('masterPublicIpAddressName'))]"
      ],
      "properties" : {
        "frontendIPConfigurations" : [
          {
```

```
          "name" : "public-lb-ip",
          "properties" : {
            "publicIPAddress" : {
              "id" : "[variables('masterPublicIpAddressID')]"
            }
          }
        }
      ],
      "backendAddressPools" : [
        {
          "name" : "public-lb-backend"
        }
      ],
      "loadBalancingRules" : [
        {
          "name" : "api-internal",
          "properties" : {
            "frontendIPConfiguration" : {
              "id" :"[concat(variables('masterLoadBalancerID'), '/frontendIPConfigurations/public-lb-
ip')]"
            },
            "backendAddressPool" : {
              "id" : "[concat(variables('masterLoadBalancerID'), '/backendAddressPools/public-lb-
backend')]"
            },
            "protocol" : "Tcp",
            "loadDistribution" : "Default",
            "idleTimeoutInMinutes" : 30,
            "frontendPort" : 6443,
            "backendPort" : 6443,
            "probe" : {
              "id" : "[concat(variables('masterLoadBalancerID'), '/probes/api-internal-probe')]"
            }
          }
        }
      ],
      "probes" : [
        {
          "name" : "api-internal-probe",
          "properties" : {
            "protocol" : "Https",
            "port" : 6443,
            "requestPath": "/readyz",
            "intervalInSeconds" : 10,
            "numberOfProbes" : 3
          }
        }
      ]
    }
  },
  {
    "apiVersion" : "2018-12-01",
    "type" : "Microsoft.Network/loadBalancers",
    "name" : "[variables('internalLoadBalancerName')]",
    "location" : "[variables('location')]",
    "sku": {
```

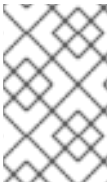
```
      "name": "[variables('skuName')]"
    },
    "properties" : {
      "frontendIPConfigurations" : [
        {
          "name" : "internal-lb-ip",
          "properties" : {
            "privateIPAllocationMethod" : "Dynamic",
            "subnet" : {
              "id" : "[variables('masterSubnetRef')]"
            },
            "privateIPAddressVersion" : "IPv4"
          }
        }
      ],
      "backendAddressPools" : [
        {
          "name" : "internal-lb-backend"
        }
      ],
      "loadBalancingRules" : [
        {
          "name" : "api-internal",
          "properties" : {
            "frontendIPConfiguration" : {
              "id" : "[concat(variables('internalLoadBalancerID'), '/frontendIPConfigurations/internal-lb-
ip')]"
            },
            "frontendPort" : 6443,
            "backendPort" : 6443,
            "enableFloatingIP" : false,
            "idleTimeoutInMinutes" : 30,
            "protocol" : "Tcp",
            "enableTcpReset" : false,
            "loadDistribution" : "Default",
            "backendAddressPool" : {
              "id" : "[concat(variables('internalLoadBalancerID'), '/backendAddressPools/internal-lb-
backend')]"
            },
            "probe" : {
              "id" : "[concat(variables('internalLoadBalancerID'), '/probes/api-internal-probe')]"
            }
          }
        },
        {
          "name" : "sint",
          "properties" : {
            "frontendIPConfiguration" : {
              "id" : "[concat(variables('internalLoadBalancerID'), '/frontendIPConfigurations/internal-lb-
ip')]"
            },
            "frontendPort" : 22623,
            "backendPort" : 22623,
            "enableFloatingIP" : false,
            "idleTimeoutInMinutes" : 30,
            "protocol" : "Tcp",
```

```
            "enableTcpReset" : false,
            "loadDistribution" : "Default",
            "backendAddressPool" : {
              "id" : "[concat(variables('internalLoadBalancerID'), '/backendAddressPools/internal-lb-backend')]"
            },
            "probe" : {
              "id" : "[concat(variables('internalLoadBalancerID'), '/probes/sint-probe')]"
            }
          }
        }
      ],
      "probes" : [
        {
          "name" : "api-internal-probe",
          "properties" : {
            "protocol" : "Https",
            "port" : 6443,
            "requestPath": "/readyz",
            "intervalInSeconds" : 10,
            "numberOfProbes" : 3
          }
        },
        {
          "name" : "sint-probe",
          "properties" : {
            "protocol" : "Https",
            "port" : 22623,
            "requestPath": "/healthz",
            "intervalInSeconds" : 10,
            "numberOfProbes" : 3
          }
        }
      ]
    }
  },
  {
    "apiVersion": "2018-09-01",
    "type": "Microsoft.Network/privateDnsZones/A",
    "name": "[concat(parameters('privateDNSZoneName'), '/api')]",
    "location" : "[variables('location')]",
    "dependsOn" : [
      "[concat('Microsoft.Network/loadBalancers/', variables('internalLoadBalancerName'))]"
    ],
    "properties": {
      "ttl": 60,
      "aRecords": [
        {
          "ipv4Address": "[reference(variables('internalLoadBalancerName')).frontendIPConfigurations[0].properties.privateIPAddress]"
        }
      ]
    }
  },
  {
```

```
      "apiVersion": "2018-09-01",
      "type": "Microsoft.Network/privateDnsZones/A",
      "name": "[concat(parameters('privateDNSZoneName'), '/api-int')]",
      "location" : "[variables('location')]",
      "dependsOn" : [
        "[concat('Microsoft.Network/loadBalancers/', variables('internalLoadBalancerName'))]"
      ],
      "properties": {
        "ttl": 60,
        "aRecords": [
          {
            "ipv4Address": "
[reference(variables('internalLoadBalancerName')).frontendIPConfigurations[0].properties.privateIP
Address]"
          }
        ]
      }
    }
  ]
}
```

5.10.16. Azure에서 부트스트랩 시스템 생성

OpenShift Container Platform 클러스터 초기화 과정에서 사용할 부트스트랩 시스템을 Microsoft Azure에 생성해야 합니다. 이 시스템을 생성하는 한 가지 방법은 제공된 ARM(Azure Resource Manager) 템플릿을 수정하는 것입니다.



참고

부트스트랩 시스템을 생성하는 데 제공된 ARM 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- Azure 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.
- Azure에서 VNet 및 관련 서브넷을 생성하고 구성하십시오.
- Azure에서 네트워킹 및 로드 밸런서를 생성하고 구성하십시오.
- 컨트롤 플레인 및 컴퓨팅 역할을 생성하십시오.

프로세스

1. 이 항목의 **부트스트랩 시스템에 대한 ARM 템플릿** 섹션에서 템플릿을 복사하여 클러스터의 설치 디렉터리에 **04_bootstrap.json**으로 저장합니다. 이 템플릿은 클러스터에 필요한 부트스트랩 시스템을 설명합니다.
2. 부트스트랩 URL 변수를 내보냅니다.

```
$ export BOOTSTRAP_URL=`az storage blob url --account-name ${CLUSTER_NAME}sa --
account-key ${ACCOUNT_KEY} -c "files" -n "bootstrap.ign" -o tsv`
```

3. 부트스트랩 ignition 변수를 내보냅니다.

```
$ export BOOTSTRAP_IGNITION=`jq -rcnM --arg v "3.2.0" --arg url ${BOOTSTRAP_URL}
'{ignition:{version:$v,config:{replace:{source:$url}}}}' | base64 | tr -d '\n'`
```

4. **az** CLI를 사용하여 배포를 생성합니다.

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/04_bootstrap.json" \
  --parameters bootstrapIgnition="${BOOTSTRAP_IGNITION}" \ 1
  --parameters sshKeyData="${SSH_KEY}" \ 2
  --parameters baseName="${INFRA_ID}" 3
```

1. 부트스트랩 클러스터의 부트스트랩 ignition 내용입니다.
2. 문자열 형태의 SSH RSA 공개 키 파일입니다.
3. 리소스 이름에 사용될 기본 이름; 일반적으로 클러스터의 인프라 ID입니다.

5.10.16.1. 부트스트랩 시스템용 ARM 템플릿

다음 ARM(Azure Resource Manager) 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 부트스트랩 시스템을 배포할 수 있습니다.

예 5.4. 04_bootstrap.json ARM 템플릿

```
{
  "$schema" : "https://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
  "contentVersion" : "1.0.0.0",
  "parameters" : {
    "baseName" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Base name to be used in resource names (usually the cluster's Infra ID)"
      }
    },
    "bootstrapIgnition" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Bootstrap ignition content for the bootstrap cluster"
      }
    },
    "sshKeyData" : {
      "type" : "securestring",
      "metadata" : {
        "description" : "SSH RSA public key file as a string."
      }
```

```
    },
    "bootstrapVMSize" : {
      "type" : "string",
      "defaultValue" : "Standard_D4s_v3",
      "allowedValues" : [
        "Standard_A2",
        "Standard_A3",
        "Standard_A4",
        "Standard_A5",
        "Standard_A6",
        "Standard_A7",
        "Standard_A8",
        "Standard_A9",
        "Standard_A10",
        "Standard_A11",
        "Standard_D2",
        "Standard_D3",
        "Standard_D4",
        "Standard_D11",
        "Standard_D12",
        "Standard_D13",
        "Standard_D14",
        "Standard_D2_v2",
        "Standard_D3_v2",
        "Standard_D4_v2",
        "Standard_D5_v2",
        "Standard_D8_v3",
        "Standard_D11_v2",
        "Standard_D12_v2",
        "Standard_D13_v2",
        "Standard_D14_v2",
        "Standard_E2_v3",
        "Standard_E4_v3",
        "Standard_E8_v3",
        "Standard_E16_v3",
        "Standard_E32_v3",
        "Standard_E64_v3",
        "Standard_E2s_v3",
        "Standard_E4s_v3",
        "Standard_E8s_v3",
        "Standard_E16s_v3",
        "Standard_E32s_v3",
        "Standard_E64s_v3",
        "Standard_G1",
        "Standard_G2",
        "Standard_G3",
        "Standard_G4",
        "Standard_G5",
        "Standard_DS2",
        "Standard_DS3",
        "Standard_DS4",
        "Standard_DS11",
        "Standard_DS12",
        "Standard_DS13",
        "Standard_DS14",
        "Standard_DS2_v2",
```

```
        "Standard_DS3_v2",
        "Standard_DS4_v2",
        "Standard_DS5_v2",
        "Standard_DS11_v2",
        "Standard_DS12_v2",
        "Standard_DS13_v2",
        "Standard_DS14_v2",
        "Standard_GS1",
        "Standard_GS2",
        "Standard_GS3",
        "Standard_GS4",
        "Standard_GS5",
        "Standard_D2s_v3",
        "Standard_D4s_v3",
        "Standard_D8s_v3"
      ],
      "metadata" : {
        "description" : "The size of the Bootstrap Virtual Machine"
      }
    }
  },
  "variables" : {
    "location" : "[resourceGroup().location]",
    "virtualNetworkName" : "[concat(parameters('baseName'), '-vnet')]",
    "virtualNetworkID" : "[resourceId('Microsoft.Network/virtualNetworks',
variables('virtualNetworkName'))]",
    "masterSubnetName" : "[concat(parameters('baseName'), '-master-subnet')]",
    "masterSubnetRef" : "[concat(variables('virtualNetworkID'), '/subnets/',
variables('masterSubnetName'))]",
    "masterLoadBalancerName" : "[concat(parameters('baseName'), '-public-lb')]",
    "internalLoadBalancerName" : "[concat(parameters('baseName'), '-internal-lb')]",
    "sshKeyPath" : "/home/core/.ssh/authorized_keys",
    "identityName" : "[concat(parameters('baseName'), '-identity')]",
    "vmName" : "[concat(parameters('baseName'), '-bootstrap')]",
    "nicName" : "[concat(variables('vmName'), '-nic')]",
    "imageName" : "[concat(parameters('baseName'), '-image')]",
    "clusterNsgName" : "[concat(parameters('baseName'), '-nsg')]",
    "sshPublicIpAddressName" : "[concat(variables('vmName'), '-ssh-pip')]"
  },
  "resources" : [
    {
      "apiVersion" : "2018-12-01",
      "type" : "Microsoft.Network/publicIPAddresses",
      "name" : "[variables('sshPublicIpAddressName')]",
      "location" : "[variables('location')]",
      "sku": {
        "name": "Standard"
      },
      "properties" : {
        "publicIPAllocationMethod" : "Static",
        "dnsSettings" : {
          "domainNameLabel" : "[variables('sshPublicIpAddressName')]"
        }
      }
    },
    {
```

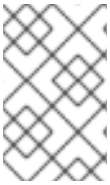
```
    "apiVersion" : "2018-06-01",
    "type" : "Microsoft.Network/networkInterfaces",
    "name" : "[variables('nicName')]",
    "location" : "[variables('location')]",
    "dependsOn" : [
     "[resourceId('Microsoft.Network/publicIPAddresses', variables('sshPublicIpAddressName'))]"
    ],
    "properties" : {
     "ipConfigurations" : [
      {
       "name" : "pipConfig",
       "properties" : {
        "privateIPAllocationMethod" : "Dynamic",
        "publicIPAddress": {
         "id": "[resourceId('Microsoft.Network/publicIPAddresses',
variables('sshPublicIpAddressName'))]"
        },
        "subnet" : {
         "id" : "[variables('masterSubnetRef')]"
        },
        "loadBalancerBackendAddressPools" : [
         {
          "id" : "[concat('/subscriptions/', subscription().subscriptionId, '/resourceGroups/',
resourceGroup().name, '/providers/Microsoft.Network/loadBalancers/',
variables('masterLoadBalancerName'), '/backendAddressPools/public-lb-backend')]"
         },
         {
          "id" : "[concat('/subscriptions/', subscription().subscriptionId, '/resourceGroups/',
resourceGroup().name, '/providers/Microsoft.Network/loadBalancers/',
variables('internalLoadBalancerName'), '/backendAddressPools/internal-lb-backend')]"
         }
        ]
       }
      }
     ]
    }
   },
   {
    "apiVersion" : "2018-06-01",
    "type" : "Microsoft.Compute/virtualMachines",
    "name" : "[variables('vmName')]",
    "location" : "[variables('location')]",
    "identity" : {
     "type" : "userAssigned",
     "userAssignedIdentities" : {
      "[resourceID('Microsoft.ManagedIdentity/userAssignedIdentities/',
variables('identityName'))]" : {}
     }
    },
    "dependsOn" : [
     "[concat('Microsoft.Network/networkInterfaces/', variables('nicName'))]"
    ],
    "properties" : {
     "hardwareProfile" : {
      "vmSize" : "[parameters('bootstrapVMSize')]"
     },
```

```
      "osProfile" : {
        "computerName" : "[variables('vmName')]",
        "adminUsername" : "core",
        "customData" : "[parameters('bootstrapIgnition')]",
        "linuxConfiguration" : {
          "disablePasswordAuthentication" : true,
          "ssh" : {
            "publicKeys" : [
              {
                "path" : "[variables('sshKeyPath')]",
                "keyData" : "[parameters('sshKeyData')]"
              }
            ]
          }
        }
      },
      "storageProfile" : {
        "imageReference": {
          "id": "[resourceId('Microsoft.Compute/images', variables('imageName'))]"
        },
        "osDisk" : {
          "name": "[concat(variables('vmName'),'_OSDisk')]",
          "osType" : "Linux",
          "createOption" : "FromImage",
          "managedDisk": {
            "storageAccountType": "Premium_LRS"
          },
          "diskSizeGB" : 100
        }
      },
      "networkProfile" : {
        "networkInterfaces" : [
          {
            "id" : "[resourceId('Microsoft.Network/networkInterfaces', variables('nicName'))]"
          }
        ]
      }
    },
    {
      "apiVersion" : "2018-06-01",
      "type": "Microsoft.Network/networkSecurityGroups/securityRules",
      "name" : "[concat(variables('clusterNsgName'), '/bootstrap_ssh_in')]",
      "location" : "[variables('location')]",
      "dependsOn" : [
        "[resourceId('Microsoft.Compute/virtualMachines', variables('vmName'))]"
      ],
      "properties": {
        "protocol" : "Tcp",
        "sourcePortRange" : "*",
        "destinationPortRange" : "22",
        "sourceAddressPrefix" : "*",
        "destinationAddressPrefix" : "*",
        "access" : "Allow",
        "priority" : 100,
        "direction" : "Inbound"
```

```
      }
    }
  ]
}
```

5.10.17. Azure에 컨트롤 플레인 시스템 생성

클러스터가 사용할 Microsoft Azure에서 컨트롤 플레인 시스템을 생성해야 합니다. 이러한 시스템을 생성하는 한 가지 방법은 제공된 ARM(Azure Resource Manager) 템플릿을 수정하는 것입니다.



참고

컨트롤 플레인 시스템을 생성하는 데 제공된 ARM 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- Azure 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.
- Azure에서 VNet 및 관련 서브넷을 생성하고 구성하십시오.
- Azure에서 네트워킹 및 로드 밸런서를 생성하고 구성하십시오.
- 컨트롤 플레인 및 컴퓨팅 역할을 생성하십시오.
- 부트스트랩 시스템을 생성합니다.

프로세스

1. 이 항목의 **컨트롤 플레인 시스템에 대한 ARM 템플릿** 섹션에서 템플릿을 복사하여 클러스터의 설치 디렉터리에 **05_masters.json**으로 저장합니다. 이 템플릿은 클러스터에 필요한 컨트롤 플레인 시스템을 설명합니다.
2. 컨트롤 플레인 시스템 배포에 필요한 다음 변수를 내보냅니다.

```
$ export MASTER_IGNITION=`cat <installation_directory>/master.ign | base64 | tr -d '\n'`
```

3. **az** CLI를 사용하여 배포를 생성합니다.

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/05_masters.json" \
  --parameters masterIgnition="${MASTER_IGNITION}" \ 1
  --parameters sshKeyData="${SSH_KEY}" \ 2
  --parameters privateDNSZoneName="${CLUSTER_NAME}.${BASE_DOMAIN}" \ 3
  --parameters baseName="${INFRA_ID}" 4
```

1. 컨트롤 플레인 노드의 Ignition 내용입니다.
2. 문자열 형태의 SSH RSA 공개 키 파일입니다.

3 컨트롤 플레인 노드가 연결된 프라이빗 DNS 영역의 이름입니다.

4 리소스 이름에 사용될 기본 이름; 일반적으로 클러스터의 인프라 ID입니다.

5.10.17.1. 컨트롤 플레인 시스템에 대한 ARM 템플릿

다음 ARM(Azure Resource Manager) 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 컨트롤 플레인 시스템을 배포할 수 있습니다.

예 5.5. 05_masters.json ARM 템플릿

```
{
  "$schema" : "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion" : "1.0.0.0",
  "parameters" : {
    "baseName" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Base name to be used in resource names (usually the cluster's Infra ID)"
      }
    },
    "masterIgnition" : {
      "type" : "string",
      "metadata" : {
        "description" : "Ignition content for the master nodes"
      }
    },
    "numberOfMasters" : {
      "type" : "int",
      "defaultValue" : 3,
      "minValue" : 2,
      "maxValue" : 30,
      "metadata" : {
        "description" : "Number of OpenShift masters to deploy"
      }
    },
    "sshKeyData" : {
      "type" : "securestring",
      "metadata" : {
        "description" : "SSH RSA public key file as a string"
      }
    },
    "privateDNSZoneName" : {
      "type" : "string",
      "metadata" : {
        "description" : "Name of the private DNS zone the master nodes are going to be attached to"
      }
    },
    "masterVMSize" : {
      "type" : "string",
      "defaultValue" : "Standard_D8s_v3",
      "allowedValues" : [
        "Standard_A2",
```

```
        "Standard_A3",
        "Standard_A4",
        "Standard_A5",
        "Standard_A6",
        "Standard_A7",
        "Standard_A8",
        "Standard_A9",
        "Standard_A10",
        "Standard_A11",
        "Standard_D2",
        "Standard_D3",
        "Standard_D4",
        "Standard_D11",
        "Standard_D12",
        "Standard_D13",
        "Standard_D14",
        "Standard_D2_v2",
        "Standard_D3_v2",
        "Standard_D4_v2",
        "Standard_D5_v2",
        "Standard_D8_v3",
        "Standard_D11_v2",
        "Standard_D12_v2",
        "Standard_D13_v2",
        "Standard_D14_v2",
        "Standard_E2_v3",
        "Standard_E4_v3",
        "Standard_E8_v3",
        "Standard_E16_v3",
        "Standard_E32_v3",
        "Standard_E64_v3",
        "Standard_E2s_v3",
        "Standard_E4s_v3",
        "Standard_E8s_v3",
        "Standard_E16s_v3",
        "Standard_E32s_v3",
        "Standard_E64s_v3",
        "Standard_G1",
        "Standard_G2",
        "Standard_G3",
        "Standard_G4",
        "Standard_G5",
        "Standard_DS2",
        "Standard_DS3",
        "Standard_DS4",
        "Standard_DS11",
        "Standard_DS12",
        "Standard_DS13",
        "Standard_DS14",
        "Standard_DS2_v2",
        "Standard_DS3_v2",
        "Standard_DS4_v2",
        "Standard_DS5_v2",
        "Standard_DS11_v2",
        "Standard_DS12_v2",
        "Standard_DS13_v2",
```

```
        "Standard_DS14_v2",
        "Standard_GS1",
        "Standard_GS2",
        "Standard_GS3",
        "Standard_GS4",
        "Standard_GS5",
        "Standard_D2s_v3",
        "Standard_D4s_v3",
        "Standard_D8s_v3"
      ],
      "metadata" : {
        "description" : "The size of the Master Virtual Machines"
      }
    },
    "diskSizeGB" : {
      "type" : "int",
      "defaultValue" : 1024,
      "metadata" : {
        "description" : "Size of the Master VM OS disk, in GB"
      }
    }
  },
  "variables" : {
    "location" : "[resourceGroup().location]",
    "virtualNetworkName" : "[concat(parameters('baseName'), '-vnet')]",
    "virtualNetworkID" : "[resourceId('Microsoft.Network/virtualNetworks',
variables('virtualNetworkName'))]",
    "masterSubnetName" : "[concat(parameters('baseName'), '-master-subnet')]",
    "masterSubnetRef" : "[concat(variables('virtualNetworkID'), '/subnets/',
variables('masterSubnetName'))]",
    "masterLoadBalancerName" : "[concat(parameters('baseName'), '-public-lb')]",
    "internalLoadBalancerName" : "[concat(parameters('baseName'), '-internal-lb')]",
    "sshKeyPath" : "/home/core/.ssh/authorized_keys",
    "identityName" : "[concat(parameters('baseName'), '-identity')]",
    "imageName" : "[concat(parameters('baseName'), '-image')]",
    "copy" : [
      {
        "name" : "vmNames",
        "count" :  "[parameters('numberOfMasters')]",
        "input" : "[concat(parameters('baseName'), '-master-', copyIndex('vmNames'))]"
      }
    ]
  },
  "resources" : [
    {
      "apiVersion" : "2018-06-01",
      "type" : "Microsoft.Network/networkInterfaces",
      "copy" : {
        "name" : "nicCopy",
        "count" : "[length(variables('vmNames'))]"
      },
      "name" : "[concat(variables('vmNames')[copyIndex()], '-nic')]",
      "location" : "[variables('location')]",
      "properties" : {
        "ipConfigurations" : [
          {
```

```
            "name" : "pipConfig",
            "properties" : {
              "privateIPAllocationMethod" : "Dynamic",
              "subnet" : {
                "id" : "[variables('masterSubnetRef')]"
              },
              "loadBalancerBackendAddressPools" : [
                {
                  "id" : "[concat('/subscriptions/', subscription().subscriptionId, '/resourceGroups/',
resourceGroup().name, '/providers/Microsoft.Network/loadBalancers/',
variables('masterLoadBalancerName'), '/backendAddressPools/public-lb-backend')]"
                },
                {
                  "id" : "[concat('/subscriptions/', subscription().subscriptionId, '/resourceGroups/',
resourceGroup().name, '/providers/Microsoft.Network/loadBalancers/',
variables('internalLoadBalancerName'), '/backendAddressPools/internal-lb-backend')]"
                }
              ]
            }
          }
        ]
      }
    },
    {
      "apiVersion": "2018-09-01",
      "type": "Microsoft.Network/privateDnsZones/SRV",
      "name": "[concat(parameters('privateDNSZoneName'), '/_etcd-server-ssl._tcp')]",
      "location" : "[variables('location')]",
      "properties": {
        "ttl": 60,
        "copy": [{
          "name": "srvRecords",
          "count": "[length(variables('vmNames'))]",
          "input": {
            "priority": 0,
            "weight" : 10,
            "port" : 2380,
            "target" : "[concat('etcd-', copyIndex('srvRecords'), '.',
parameters('privateDNSZoneName'))]"
          }
        }]
      }
    },
    {
      "apiVersion": "2018-09-01",
      "type": "Microsoft.Network/privateDnsZones/A",
      "copy" : {
        "name" : "dnsCopy",
        "count" : "[length(variables('vmNames'))]"
      },
      "name": "[concat(parameters('privateDNSZoneName'), '/etcd-', copyIndex())]",
      "location" : "[variables('location')]",
      "dependsOn" : [
        "[concat('Microsoft.Network/networkInterfaces/', concat(variables('vmNames')[copyIndex()], '-
nic'))]"
      ],
```

```
      "properties": {
        "ttl": 60,
        "aRecords": [
          {
            "ipv4Address": "[reference(concat(variables('vmNames')[copyIndex()], '-
nic')).ipConfigurations[0].properties.privateIPAddress]"
          }
        ]
      }
    },
    {
      "apiVersion" : "2018-06-01",
      "type" : "Microsoft.Compute/virtualMachines",
      "copy" : {
        "name" : "vmCopy",
        "count" : "[length(variables('vmNames'))]"
      },
      "name" : "[variables('vmNames')[copyIndex()]]",
      "location" : "[variables('location')]",
      "identity" : {
        "type" : "userAssigned",
        "userAssignedIdentities" : {
          "[resourceID('Microsoft.ManagedIdentity/userAssignedIdentities/',
variables('identityName'))]" : {}
        }
      },
      "dependsOn" : [
        "[concat('Microsoft.Network/networkInterfaces/', concat(variables('vmNames')[copyIndex()], '-
nic'))]",
        "[concat('Microsoft.Network/privateDnsZones/', parameters('privateDNSZoneName'),
'/A/etcd-', copyIndex())]",
        "[concat('Microsoft.Network/privateDnsZones/', parameters('privateDNSZoneName'),
'/SRV/_etcd-server-ssl._tcp')]"
      ],
      "properties" : {
        "hardwareProfile" : {
          "vmSize" : "[parameters('masterVMSize')]"
        },
        "osProfile" : {
          "computerName" : "[variables('vmNames')[copyIndex()]]",
          "adminUsername" : "core",
          "customData" : "[parameters('masterIgnition')]",
          "linuxConfiguration" : {
            "disablePasswordAuthentication" : true,
            "ssh" : {
              "publicKeys" : [
                {
                  "path" : "[variables('sshKeyPath')]",
                  "keyData" : "[parameters('sshKeyData')]"
                }
              ]
            }
          }
        },
        "storageProfile" : {
          "imageReference": {
```

```
            "id": "[resourceId('Microsoft.Compute/images', variables('imageName'))]"
          },
          "osDisk" : {
            "name": "[concat(variables('vmNames')[copyIndex()], '_OSDisk')]",
            "osType" : "Linux",
            "createOption" : "FromImage",
            "caching": "ReadOnly",
            "writeAcceleratorEnabled": false,
            "managedDisk": {
              "storageAccountType": "Premium_LRS"
            },
            "diskSizeGB" : "[parameters('diskSizeGB')]"
          }
        },
        "networkProfile" : {
          "networkInterfaces" : [
            {
              "id" : "[resourceId('Microsoft.Network/networkInterfaces', concat(variables('vmNames')
[copyIndex()], '-nic'))]",
              "properties": {
                "primary": false
              }
            }
          ]
        }
      }
    }
  ]
}
```

5.10.18. 부트스트랩이 완료되길 기다렸다가 Azure에서 부트스트랩 리소스 제거

Microsoft Azure에 필요한 인프라를 모두 생성한 후 설치 프로그램에 의해 생성된 Ignition 구성 파일을 사용하여 프로비저닝한 시스템에서 부트스트랩 프로세스가 완료될 때까지 기다립니다.

사전 요구 사항

- Azure 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.
- Azure에서 VNet 및 관련 서브넷을 생성하고 구성하십시오.
- Azure에서 네트워킹 및 로드 밸런서를 생성하고 구성하십시오.
- 컨트롤 플레인 및 컴퓨팅 역할을 생성하십시오.
- 부트스트랩 시스템을 생성합니다.
- 컨트롤 플레인 시스템을 생성합니다.

프로세스

1. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

```
$ ./openshift-install wait-for bootstrap-complete --dir <installation_directory> \ 1
    --log-level info 2
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2 다른 설치 세부 사항을 보려면 **info** 대신 **warn**, **debug** 또는 **error**를 지정합니다.

FATAL 경고 없이 명령이 종료되면 프로덕션 컨트롤 플레인이 초기화된 것입니다.

2. 부트스트랩 리소스를 삭제합니다.

```
$ az network nsg rule delete -g ${RESOURCE_GROUP} --nsg-name ${INFRA_ID}-nsg --
name bootstrap_ssh_in
$ az vm stop -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap
$ az vm deallocate -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap
$ az vm delete -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap --yes
$ az disk delete -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap_OSDisk --no-
wait --yes
$ az network nic delete -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap-nic --no-
wait
$ az storage blob delete --account-key ${ACCOUNT_KEY} --account-name
${CLUSTER_NAME}sa --container-name files --name bootstrap.ign
$ az network public-ip delete -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap-
ssh-pip
```

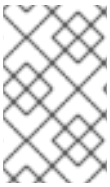
참고

부트스트랩 서버를 삭제하지 않으면 API 트래픽이 부트스트랩 서버로 라우팅되므로 설치가 성공하지 못할 수 있습니다.

5.10.19. Azure에 추가 작업자 시스템 생성

개별 인스턴스를 따로 시작하거나 자동 확장 그룹과 같은 클러스터 외부의 자동화된 프로세스를 통해 클러스터에서 사용할 작업자 시스템을 Microsoft Azure에 생성할 수 있습니다. OpenShift Container Platform의 기본 제공 클러스터 확장 메커니즘과 시스템 API를 활용할 수도 있습니다.

예에서는 ARM(Azure Resource Manager) 템플릿을 사용하여 인스턴스 하나를 수동으로 시작합니다. 파일에 **06_worker.json** 유형의 추가 리소스를 포함시켜 추가 인스턴스를 시작할 수 있습니다.



참고

작업자 시스템을 생성하는 데 제공된 ARM 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- Azure 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.
- Azure에서 VNet 및 관련 서브넷을 생성하고 구성하십시오.

- Azure에서 네트워킹 및 로드 밸런서를 생성하고 구성하십시오.
- 컨트롤 플레인 및 컴퓨팅 역할을 생성하십시오.
- 부트스트랩 시스템을 생성합니다.
- 컨트롤 플레인 시스템을 생성합니다.

프로세스

1. 이 항목의 **작업자 시스템에 대한 ARM 템플릿** 섹션에서 템플릿을 복사하여 클러스터의 설치 디렉터리에 **06_workers.json**으로 저장합니다. 이 템플릿에서 클러스터에 필요한 작업자 시스템을 설명합니다.
2. 작업자 시스템 배포에 필요한 다음 변수를 내보냅니다.

```
$ export WORKER_IGNITION=`cat <installation_directory>/worker.ign | base64 | tr -d '\n'`
```

3. **az** CLI를 사용하여 배포를 생성합니다.

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/06_workers.json" \
  --parameters workerIgnition="${WORKER_IGNITION}" \ 1
  --parameters sshKeyData="${SSH_KEY}" \ 2
  --parameters baseName="${INFRA_ID}" 3
```

1. 작업자 노드의 ignition 내용입니다.
2. 문자열 형태의 SSH RSA 공개 키 파일입니다.
3. 리소스 이름에 사용될 기본 이름; 일반적으로 클러스터의 인프라 ID입니다.

5.10.19.1. 작업자 시스템에 대한 ARM 템플릿

다음 ARM(Azure Resource Manager) 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 작업자 시스템을 배포할 수 있습니다.

예 5.6. 06_workers.json ARM 템플릿

```
{
  "$schema" : "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion" : "1.0.0.0",
  "parameters" : {
    "baseName" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Base name to be used in resource names (usually the cluster's Infra ID)"
      }
    },
    "workerIgnition" : {
      "type" : "string",
      "metadata" : {
```

```
      "description" : "Ignition content for the worker nodes"
    }
  },
  "numberOfNodes" : {
    "type" : "int",
    "defaultValue" : 3,
    "minValue" : 2,
    "maxValue" : 30,
    "metadata" : {
      "description" : "Number of OpenShift compute nodes to deploy"
    }
  },
  "sshKeyData" : {
    "type" : "securestring",
    "metadata" : {
      "description" : "SSH RSA public key file as a string"
    }
  },
  "nodeVMSize" : {
    "type" : "string",
    "defaultValue" : "Standard_D4s_v3",
    "allowedValues" : [
      "Standard_A2",
      "Standard_A3",
      "Standard_A4",
      "Standard_A5",
      "Standard_A6",
      "Standard_A7",
      "Standard_A8",
      "Standard_A9",
      "Standard_A10",
      "Standard_A11",
      "Standard_D2",
      "Standard_D3",
      "Standard_D4",
      "Standard_D11",
      "Standard_D12",
      "Standard_D13",
      "Standard_D14",
      "Standard_D2_v2",
      "Standard_D3_v2",
      "Standard_D4_v2",
      "Standard_D5_v2",
      "Standard_D8_v3",
      "Standard_D11_v2",
      "Standard_D12_v2",
      "Standard_D13_v2",
      "Standard_D14_v2",
      "Standard_E2_v3",
      "Standard_E4_v3",
      "Standard_E8_v3",
      "Standard_E16_v3",
      "Standard_E32_v3",
      "Standard_E64_v3",
      "Standard_E2s_v3",
      "Standard_E4s_v3",
```

```
        "Standard_E8s_v3",
        "Standard_E16s_v3",
        "Standard_E32s_v3",
        "Standard_E64s_v3",
        "Standard_G1",
        "Standard_G2",
        "Standard_G3",
        "Standard_G4",
        "Standard_G5",
        "Standard_DS2",
        "Standard_DS3",
        "Standard_DS4",
        "Standard_DS11",
        "Standard_DS12",
        "Standard_DS13",
        "Standard_DS14",
        "Standard_DS2_v2",
        "Standard_DS3_v2",
        "Standard_DS4_v2",
        "Standard_DS5_v2",
        "Standard_DS11_v2",
        "Standard_DS12_v2",
        "Standard_DS13_v2",
        "Standard_DS14_v2",
        "Standard_GS1",
        "Standard_GS2",
        "Standard_GS3",
        "Standard_GS4",
        "Standard_GS5",
        "Standard_D2s_v3",
        "Standard_D4s_v3",
        "Standard_D8s_v3"
      ],
      "metadata" : {
        "description" : "The size of the each Node Virtual Machine"
      }
    }
  },
  "variables" : {
    "location" : "[resourceGroup().location]",
    "virtualNetworkName" : "[concat(parameters('baseName'), '-vnet')]",
    "virtualNetworkID" : "[resourceId('Microsoft.Network/virtualNetworks',
variables('virtualNetworkName'))]",
    "nodeSubnetName" : "[concat(parameters('baseName'), '-worker-subnet')]",
    "nodeSubnetRef" : "[concat(variables('virtualNetworkID'), '/subnets/',
variables('nodeSubnetName'))]",
    "infraLoadBalancerName" : "[parameters('baseName')]",
    "sshKeyPath" : "/home/capi/.ssh/authorized_keys",
    "identityName" : "[concat(parameters('baseName'), '-identity')]",
    "imageName" : "[concat(parameters('baseName'), '-image')]",
    "copy" : [
      {
        "name" : "vmNames",
        "count" :  "[parameters('numberOfNodes')]",
        "input" : "[concat(parameters('baseName'), '-worker-', variables('location'), '-',
copyIndex('vmNames', 1))]"
```

```
      }
    ]
  },
  "resources" : [
    {
      "apiVersion" : "2019-05-01",
      "name" : "[concat('node', copyIndex())]",
      "type" : "Microsoft.Resources/deployments",
      "copy" : {
        "name" : "nodeCopy",
        "count" : "[length(variables('vmNames'))]"
      },
      "properties" : {
        "mode" : "Incremental",
        "template" : {
          "$schema" : "http://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
          "contentVersion" : "1.0.0.0",
          "resources" : [
            {
              "apiVersion" : "2018-06-01",
              "type" : "Microsoft.Network/networkInterfaces",
              "name" : "[concat(variables('vmNames')[copyIndex()], '-nic')]",
              "location" : "[variables('location')]",
              "properties" : {
                "ipConfigurations" : [
                  {
                    "name" : "pipConfig",
                    "properties" : {
                      "privateIPAllocationMethod" : "Dynamic",
                      "subnet" : {
                        "id" : "[variables('nodeSubnetRef')]"
                      }
                    }
                  }
                ]
              }
            },
            {
              "apiVersion" : "2018-06-01",
              "type" : "Microsoft.Compute/virtualMachines",
              "name" : "[variables('vmNames')[copyIndex()]]",
              "location" : "[variables('location')]",
              "tags" : {
                "kubernetes.io-cluster-ffranzupi": "owned"
              },
              "identity" : {
                "type" : "userAssigned",
                "userAssignedIdentities" : {
                  "[resourceID('Microsoft.ManagedIdentity/userAssignedIdentities/',
variables('identityName'))]" : {}
                }
              },
              "dependsOn" : [
                "[concat('Microsoft.Network/networkInterfaces/', concat(variables('vmNames')
[copyIndex()], '-nic'))]"
```

```
            ],
            "properties" : {
              "hardwareProfile" : {
                "vmSize" : "[parameters('nodeVMSize')]"
              },
              "osProfile" : {
                "computerName" : "[variables('vmNames')[copyIndex()]]",
                "adminUsername" : "capi",
                "customData" : "[parameters('workerIgnition')]",
                "linuxConfiguration" : {
                  "disablePasswordAuthentication" : true,
                  "ssh" : {
                    "publicKeys" : [
                      {
                        "path" : "[variables('sshKeyPath')]",
                        "keyData" : "[parameters('sshKeyData')]"
                      }
                    ]
                  }
                }
              },
              "storageProfile" : {
                "imageReference": {
                  "id": "[resourceId('Microsoft.Compute/images', variables('imageName'))]"
                },
                "osDisk" : {
                  "name": "[concat(variables('vmNames')[copyIndex()],'_OSDisk')]",
                  "osType" : "Linux",
                  "createOption" : "FromImage",
                  "managedDisk": {
                    "storageAccountType": "Premium_LRS"
                  },
                  "diskSizeGB": 128
                }
              },
              "networkProfile" : {
                "networkInterfaces" : [
                  {
                    "id" : "[resourceId('Microsoft.Network/networkInterfaces',
concat(variables('vmNames')[copyIndex()], '-nic'))]",
                    "properties": {
                      "primary": true
                    }
                  }
                ]
              }
            }
          }
        ]
      }
    }
  ]
}
```

5.10.20. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift CLI(**oc**)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 **oc**를 설치할 수 있습니다.

중요

이전 버전의 **oc**를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 **oc**를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치
다음 절차를 사용하여 Linux에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Linux Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. **oc** 바이너리를 **PATH**에 있는 디렉터리에 배치합니다.
PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치
다음 절차에 따라 Windows에 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Windows Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. ZIP 프로그램으로 아카이브의 압축을 풉니다.
5. **oc** 바이너리를 **PATH**에 있는 디렉터리로 이동합니다.
PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치
다음 절차에 따라 macOS에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 MacOSX Client** 항목 옆에 있는 **지금 다운로드를** 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 해제하고 압축을 풉니다.
5. **oc** 바이너리 PATH의 디렉터리로 이동합니다.
   PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

   ```
   $ echo $PATH
   ```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

5.10.21. CLI를 사용하여 클러스터에 로그인

클러스터 **kubeconfig** 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. **kubeconfig** 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- **oc** CLI를 설치했습니다.

프로세스

1. **kubeadmin** 인증 정보를 내보냅니다.

   ```
   $ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
   ```

   1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 **oc** 명령을 성공적으로 실행할 수 있는지 확인합니다.

   ```
   $ oc whoami
   ```

   출력 예

   ```
   system:admin
   ```

-

5.10.22. 머신의 인증서 서명 요청 승인

클러스터에 시스템을 추가하면 추가한 시스템별로 보류 중인 인증서 서명 요청(CSR)이 두 개씩 생성됩니다. 이러한 CSR이 승인되었는지 확인해야 하며, 필요한 경우 이를 직접 승인해야 합니다. 클라이언트 요청을 먼저 승인한 다음 서버 요청을 승인해야 합니다.

사전 요구 사항

- 클러스터에 시스템을 추가했습니다.

프로세스

1. 클러스터가 시스템을 인식하는지 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.22.1
master-1  Ready     master  63m  v1.22.1
master-2  Ready     master  64m  v1.22.1
```

출력에 생성된 모든 시스템이 나열됩니다.

참고

이전 출력에는 일부 CSR이 승인될 때까지 컴퓨팅 노드(작업자 노드라고도 함)가 포함되지 않을 수 있습니다.

2. 보류 중인 CSR을 검토하고 클러스터에 추가한 각 시스템에 대해 **Pending** 또는 **Approved** 상태의 클라이언트 및 서버 요청이 표시되는지 확인합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
...
```

예에서는 두 시스템이 클러스터에 참여하고 있습니다. 목록에는 승인된 CSR이 더 많이 나타날 수도 있습니다.

3. CSR이 승인되지 않은 경우, 추가된 시스템에 대한 모든 보류 중인 CSR이 **Pending** 상태로 전환된 후 클러스터 시스템의 CSR을 승인합니다.

참고

CSR은 교체 주기가 자동으로 만료되므로 클러스터에 시스템을 추가한 후 1시간 이내에 CSR을 승인하십시오. 한 시간 내에 승인하지 않으면 인증서가 교체되고 각 노드에 대해 두 개 이상의 인증서가 표시됩니다. 이러한 인증서를 모두 승인해야 합니다. 클라이언트 CSR이 승인되면 Kubelet은 인증서에 대한 보조 CSR을 생성하므로 수동 승인이 필요합니다. 그러면 Kubelet에서 동일한 매개변수를 사용하여 새 인증서를 요청하는 경우 인증서 갱신 요청은 **machine-approver**에 의해 자동으로 승인됩니다.

참고

베어 메탈 및 기타 사용자 프로비저닝 인프라와 같이 머신 API를 사용하도록 활성화되지 않는 플랫폼에서 실행되는 클러스터의 경우 CSR(Kubelet service Certificate Request)을 자동으로 승인하는 방법을 구현해야 합니다. 요청이 승인되지 않으면 API 서버가 kubelet에 연결될 때 서비스 인증서가 필요하므로 **oc exec**, **oc rsh**, **oc logs** 명령을 성공적으로 수행할 수 없습니다. Kubelet 엔드 포인트에 연결하는 모든 작업을 수행하려면 이 인증서 승인이 필요합니다. 이 방법은 새 CSR을 감시하고 CSR이 **system:node** 또는 **system:admin** 그룹의 **node-bootstrapper** 서비스 계정에 의해 제출되었는지 확인하고 노드의 ID를 확인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>**은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

참고

일부 Operator는 일부 CSR이 승인될 때까지 사용할 수 없습니다.

4. 이제 클라이언트 요청이 승인되었으므로 클러스터에 추가한 각 머신의 서버 요청을 검토해야 합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE    REQUESTOR                                                   CONDITION
csr-bfd72   5m26s  system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s  system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 나머지 CSR이 승인되지 않고 **Pending** 상태인 경우 클러스터 머신의 CSR을 승인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>**은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. 모든 클라이언트 및 서버 CSR이 승인된 후 머신은 **Ready** 상태가 됩니다. 다음 명령을 실행하여 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0 Ready    master  73m  v1.22.1
master-1 Ready    master  73m  v1.22.1
master-2 Ready    master  74m  v1.22.1
worker-0 Ready    worker  11m  v1.22.1
worker-1 Ready    worker  11m  v1.22.1
```

참고

머신이 **Ready** 상태로 전환하는 데 서버 CSR의 승인 후 몇 분이 걸릴 수 있습니다.

추가 정보

- CSR에 대한 자세한 내용은 인증서 서명 요청을 참조하십시오.

5.10.23. 인그레스 DNS 레코드 추가

Kubernetes 매니페스트를 생성하고 Ignition 구성을 생성할 때 DNS 영역 구성을 제거한 경우, 인그레스 로드 밸런서를 가리키는 DNS 레코드를 수동으로 생성해야 합니다. 와일드카드 ***.apps.{baseDomain}.*** 또는 특정 레코드를 생성할 수 있습니다. 사용자 요구사항에 따라 A, CNAME 및 기타 레코드를 사용할 수 있습니다.

사전 요구 사항

- 사용자 프로비저닝 인프라를 사용하여 Microsoft Azure에 OpenShift Container Platform 클러스터를 배포했습니다.
- OpenShift CLI(**oc**)를 설치합니다.
- Azure CLI를 설치 또는 업데이트합니다.

프로세스

1. 인그레스 라우터가 로드 밸런서를 생성하고 **EXTERNAL-IP** 필드를 채웠는지 확인합니다.

```
$ oc -n openshift-ingress get service router-default
```

출력 예

```
NAME            TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)                      AGE
router-default  LoadBalancer   172.30.20.10   35.130.120.110
80:32288/TCP,443:31215/TCP   20
```

2. 인그레스 라우터 IP를 변수 형태로 내보냅니다.

```
$ export PUBLIC_IP_ROUTER=`oc -n openshift-ingress get service router-default --no-
headers | awk '{print $4}'`
```

3. 퍼블릭 DNS 영역에 ***.apps** 레코드를 추가합니다.

   a. 이 클러스터를 새로운 퍼블릭 영역에 추가하는 경우 다음을 실행하십시오.

   ```
   $ az network dns record-set a add-record -g ${BASE_DOMAIN_RESOURCE_GROUP} -
   z ${CLUSTER_NAME}.${BASE_DOMAIN} -n *.apps -a ${PUBLIC_IP_ROUTER} --ttl 300
   ```

   b. 이 클러스터를 이미 존재하는 기존 퍼블릭 영역에 추가하는 다음을 실행하십시오.

   ```
   $ az network dns record-set a add-record -g ${BASE_DOMAIN_RESOURCE_GROUP} -
   z ${BASE_DOMAIN} -n *.apps.${CLUSTER_NAME} -a ${PUBLIC_IP_ROUTER} --ttl 300
   ```

4. 프라이빗 DNS 영역에 ***.apps** 레코드를 추가합니다.

   a. 다음 명령을 사용하여 ***.apps** 레코드를 생성합니다.

   ```
   $ az network private-dns record-set a create -g ${RESOURCE_GROUP} -z
   ${CLUSTER_NAME}.${BASE_DOMAIN} -n *.apps --ttl 300
   ```

   b. 다음 명령을 사용하여 ***.apps** 레코드를 프라이빗 DNS 영역에 추가합니다.

   ```
   $ az network private-dns record-set a add-record -g ${RESOURCE_GROUP} -z
   ${CLUSTER_NAME}.${BASE_DOMAIN} -n *.apps -a ${PUBLIC_IP_ROUTER}
   ```

와일드카드를 사용하지 않고 명시적 도메인을 추가하는 방식을 선호하면 클러스터의 현재 경로 각각에 대한 항목을 생성할 수 있습니다.

```
$ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}{"\n"}{end}
{end}' routes
```

출력 예

```
oauth-openshift.apps.cluster.basedomain.com
console-openshift-console.apps.cluster.basedomain.com
downloads-openshift-console.apps.cluster.basedomain.com
alertmanager-main-openshift-monitoring.apps.cluster.basedomain.com
grafana-openshift-monitoring.apps.cluster.basedomain.com
prometheus-k8s-openshift-monitoring.apps.cluster.basedomain.com
```

5.10.24. 사용자 프로비저닝 인프라에서 Azure 설치 완료

Microsoft Azure 사용자 프로비저닝 인프라에서 OpenShift Container Platform 설치를 시작한 후 클러스터가 준비를 마칠 때까지 클러스터 이벤트를 모니터링할 수 있습니다.

사전 요구 사항

- 사용자 프로비저닝 Azure 인프라에 OpenShift Container Platform 클러스터용 부트스트랩 시스템을 배포하십시오.
- **oc** CLI를 설치하고 로그인하십시오.

프로세스

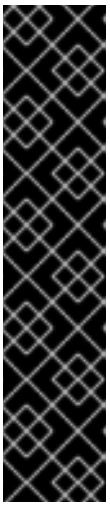
- 클러스터 설치를 완료합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

출력 예

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.



중요

- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

5.10.25. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

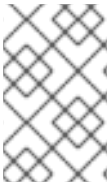
- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

5.11. AZURE의 클러스터 설치 제거

Microsoft Azure에 배포한 클러스터를 제거할 수 있습니다.

5.11.1. 설치 관리자가 프로비저닝한 인프라를 사용하는 클러스터 제거

클라우드에서 설치 관리자 프로비저닝 인프라를 사용하는 클러스터를 제거할 수 있습니다.



참고

설치 제거 후 특히 UPI(User Provisioned Infrastructure) 클러스터에서 제거되지 않은 리소스에 대해 클라우드 공급자를 확인합니다. 설치 관리자가 생성하지 않았거나 설치 프로그램이 액세스할 수 없는 리소스가 있을 수 있습니다.

사전 요구 사항

- 클러스터를 배포하는 데 사용한 설치 프로그램의 사본을 준비합니다.
- 클러스터를 만들 때 설치 프로그램에 의해 생성된 파일을 준비합니다.

OpenShift Container Platform 버전 4.13 이상을 사용하여 배포하는 데 사용된 설치 프로그램의 사본을 사용하여 클러스터를 제거할 수 있습니다.

서비스 주체를 제거하는 것은 Microsoft Azure AD Graph API에 따라 다릅니다. 설치 프로그램의 버전 4.13 이상을 사용하면 Microsoft가 Azure AD Graph API를 축소 할 때 수동 개입 없이도 서비스 주체가 제거됩니다.

프로세스

1. 클러스터를 설치하는 데 사용한 컴퓨터에 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행합니다.

```
$ ./openshift-install destroy cluster \
--dir <installation_directory> --log-level info 1 2
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2 다른 세부 사항을 보려면 **info** 대신 **warn**, **debug** 또는 **error**를 지정합니다.

참고

클러스터의 클러스터 정의 파일이 포함되어 있는 디렉터리를 지정해야 합니다. 설치 프로그램이 클러스터를 삭제하려면 이 디렉터리에 있는 **metadata.json** 파일이 필요합니다.

2. 선택사항: **<installation_directory>** 디렉터리와 OpenShift Container Platform 설치 프로그램을 삭제합니다.

6장. AZURE STACK HUB에 설치

6.1. AZURE STACK HUB에 설치 준비

6.1.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비 에 대한 문서를 읽습니다.
- Azure Stack Hub 버전 2008 이상이 설치되어 있습니다.

6.1.2. Azure Stack Hub에 OpenShift Container Platform을 설치하기 위한 요구사항

Microsoft Azure Stack Hub에 OpenShift Container Platform을 설치하기 전에 Azure 계정을 구성해야 합니다. 계정 구성, 계정 제한, DNS 영역 구성, 필수 역할 및 서비스 주체 생성에 대한 자세한 내용은 Azure Stack Hub 계정 구성을 참조하십시오.

클러스터를 Azure Stack Hub에 설치할 때 클라우드 인증 정보를 수동으로 관리해야 합니다. 클러스터를 설치하기 전에 수동 모드로 CCO(Cloud Credential Operator)를 구성하여 이 작업을 수행합니다. 자세한 내용은 Azure용 IAM 수동 생성을 참조하십시오.

6.1.3. Azure Stack Hub에 OpenShift Container Platform을 설치할 방법 선택

사용자 프로비저닝 인프라를 사용하여 Azure Stack Hub에 OpenShift Container Platform을 설치할 수 있습니다. 즉 클러스터 리소스를 직접 관리하고 유지 관리해야 합니다. 현재 클러스터 인프라를 자동으로 프로비저닝하는 설치 프로그램을 사용하여 Azure Stack Hub에 OpenShift Container Platform을 설치하는 것은 지원되지 않습니다.

설치 관리자 프로비저닝 및 사용자 프로비저닝 설치 프로세스에 대한 자세한 내용은 설치 프로세스를 참조하십시오.

6.1.3.1. 사용자 프로비저닝 인프라를 사용하여 클러스터 설치

다음 방법을 사용하여 프로비저닝하는 Azure Stack Hub 인프라에 클러스터를 설치할 수 있습니다.

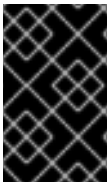
- **ARM 템플릿을 사용하여 Azure Stack Hub에 클러스터 설치**: 사용자가 제공하는 인프라를 사용하여 Azure Stack Hub에 OpenShift Container Platform을 설치할 수 있습니다. 제공된 ARM(Azure Resource Manager) 템플릿을 사용하여 설치를 지원할 수 있습니다.

6.1.4. 다음 단계

- Azure Stack Hub 계정 구성

6.2. AZURE STACK HUB 계정 구성

OpenShift Container Platform을 설치하려면 먼저 Microsoft Azure 계정을 구성해야 합니다.



중요

공용 끝점을 통해 사용할 수 있는 모든 Azure 리소스에는 리소스 이름 제한 사항이 적용되며, 특정 용어를 사용하는 리소스를 생성할 수 없습니다. Azure가 제한하는 용어 목록은 Azure 문서의 예약된 리소스 이름 오류 해결 을 참조하십시오.

6.2.1. Azure Stack Hub 계정 제한

OpenShift Container Platform 클러스터는 여러 Microsoft Azure Stack Hub 구성 요소를 사용하며 Azure Stack Hub의 기본 할당량 유형은 OpenShift Container Platform 클러스터를 설치하는 기능에 영향을 줍니다.

다음 표에는 OpenShift Container Platform 클러스터를 설치 및 실행하는 기능에 영향을 줄 수 있는 제한이 있는 Azure Stack Hub 구성 요소가 요약되어 있습니다.

구성 요소	기본적으로 필요한 구성 요소 수	설명
vCPU	56	기본 클러스터에는 56개의 vCPU가 필요하므로 계정 제한을 늘려야 합니다. 기본적으로 각 클러스터는 다음 인스턴스를 생성합니다. ● 설치 후 제거되는 하나의 부트스트랩 시스템 ● 컨트롤 플레인 시스템 세 개 ● 컴퓨팅 시스템 세 개 부트스트랩, 컨트롤 플레인 및 작업자 시스템은 8개의 vCPU를 사용하는 **Standard_DS4_v2** 가상 시스템을 사용하기 때문에 기본 클러스터에는 56개의 vCPU가 필요합니다. 부트스트랩 노드 VM은 설치 중에만 사용됩니다. 더 많은 작업자 노드를 배포하거나, 자동 크기 조정을 활성화하거나, 대규모 워크로드를 배포하거나, 다른 인스턴스 유형을 사용하려면 필요한 시스템을 클러스터가 배포할 수 있도록 계정의 vCPU 제한값을 더 늘려야 합니다.
OS 디스크	7	VM OS 디스크는 컨트롤 플레인 시스템에 대해 테스트되고 권장되는 최소 처리량 5000 IOPS/200MBps를 유지할 수 있어야 합니다. 이 처리량은 최소 1 TiB Premium SSD (P30)를 보유하여 제공할 수 있습니다. Azure Stack Hub에서 디스크 성능은 SSD 디스크 크기에 직접 의존하므로 **Standard_D8s_v3** 또는 기타 유사한 머신 유형 및 5000 IOPS 대상에서 지원하는 처리량을 달성해야 합니다. 최소 P30 디스크가 필요합니다. 읽기 대기 시간이 짧고 읽기 IOPS 및 처리량을 높이려면 호스트 캐싱을 **ReadOnly**로 설정해야 합니다. VM 메모리 또는 로컬 SSD 디스크에 있는 캐시에서 수행된 읽기는 Blob 스토리지에 있는 데이터 디스크에서 읽기보다 훨씬 빠릅니다.
VNet	1	각 기본 클러스터에는 두 개의 서브넷이 포함된 하나의 가상 네트워크(VNet)가 필요합니다.

구성 요소	기본적으로 필요한 구성 요소 수	설명
네트워크 인터페이스	7	각 기본 클러스터에는 7개의 네트워크 인터페이스가 필요합니다. 더 많은 시스템을 생성하거나 배포된 워크로드가 로드 밸런서를 생성하는 경우 클러스터는 더 많은 네트워크 인터페이스를 사용합니다.
네트워크 보안 그룹	2	각 클러스터는 VNet의 각 서브넷에 대한 네트워크 보안 그룹을 생성합니다. 기본 클러스터는 컨트롤 플레인과 컴퓨팅 노드 서브넷에 대한 네트워크 보안 그룹을 생성합니다: **controlplane**: 어디에서나 포트 6443에서 컨트롤 플레인 시스템에 도달할 수 있습니다. **node**: 포트 80 및 443을 통해 인터넷에서 작업자 노드에 도달할 수 있습니다.
네트워크 로드 밸런서	3	각 클러스터는 다음 로드 밸런서를 생성합니다. **default**: 작업자 시스템에서 포트 80과 443에 대한 요청을 로드 밸런싱하는 공용 IP 주소 **internal**: 컨트롤 플레인 시스템에서 포트 6443과 22623에 대한 요청을 로드 밸런싱하는 개인 IP 주소 **external**: 컨트롤 플레인 시스템에서 포트 6443에 대한 요청을 로드 밸런싱하는 공용 IP 주소 애플리케이션이 더 많은 Kubernetes **LoadBalancer** Service 개체를 생성하면 클러스터가 더 많은 로드 밸런서를 사용합니다.
공용 IP 주소	2	공용 로드 밸런서는 공용 IP 주소를 사용합니다. 또한 부트스트랩 시스템은 공용 IP 주소를 사용하므로 설치 중 문제를 해결하기 위해 시스템으로 SSH를 실행할 수 있습니다. 부트스트랩 노드의 IP 주소는 설치 중에만 사용됩니다.
개인 IP 주소	7	내부 로드 밸런서, 3개의 컨트롤 플레인 시스템 및 3개의 각 작업자 시스템은 각각 개인 IP 주소를 사용합니다.

6.2.2. Azure Stack Hub에서 DNS 영역 구성

Azure Stack Hub에 OpenShift Container Platform을 성공적으로 설치하려면 Azure Stack Hub DNS 영역에서 DNS 레코드를 생성해야 합니다. DNS 영역은 도메인에 대한 권한이 있어야 합니다. 등록 프로그램의 DNS 영역을 Azure Stack Hub에 위임하려면 Azure Stack Hub 데이터센터 DNS 통합 에 대한 Microsoft 설명서를 참조하십시오.

6.2.3. 필수 Azure Stack Hub 역할

Microsoft Azure Stack Hub 계정에는 사용하는 서브스크립션에 대한 다음 역할이 있어야 합니다.

- **소유자**

Azure 포털에서 역할을 설정하려면 Microsoft 문서의 역할 기반 액세스 제어를 사용하여 Azure Stack Hub의 리소스에 대한 액세스 관리를 참조하십시오.

6.2.4. 서비스 주체 생성

OpenShift Container Platform 및 해당 설치 프로그램은 Azure Resource Manager를 사용하여 Microsoft Azure 리소스를 생성하므로 이를 나타내는 서비스 주체를 생성해야 합니다.

사전 요구 사항

- Azure CLI를 설치 또는 업데이트합니다.
- Azure 계정은 사용하는 서브스크립션에 대한 필요한 역할을 갖습니다.

절차

1. 환경을 등록합니다.

```
$ az cloud register -n AzureStackCloud --endpoint-resource-manager <endpoint> 1
```

1 Azure Resource Manager 끝점을 'https://management.<region>.<fqdn>/'로 지정합니다.

   자세한 내용은 Microsoft 설명서를 참조하십시오.

2. 활성 환경을 설정합니다.

```
$ az cloud set -n AzureStackCloud
```

3. Azure Stack Hub의 특정 API 버전을 사용하도록 환경 구성을 업데이트합니다.

```
$ az cloud update --profile 2019-03-01-hybrid
```

4. Azure CLI에 로그인합니다.

```
$ az login
```

   다중 테넌트 환경에 있는 경우 테넌트 ID도 제공해야 합니다.

5. Azure 계정이 서브스크립션을 사용하는 경우 올바른 서브스크립션을 사용하고 있는지 확인합니다.

   a. 사용 가능한 계정 목록을 보고 클러스터에 사용하려는 서브스크립션의 **tenantId** 값을 기록합니다.

```
$ az account list --refresh
```

   출력 예

```
[
  {
    "cloudName": AzureStackCloud",
    "id": "9bab1460-96d5-40b3-a78e-17b15e978a80",
    "isDefault": true,
    "name": "Subscription Name",
    "state": "Enabled",
    "tenantId": "6057c7e9-b3ae-489d-a54e-de3f6bf6a8ee",
    "user": {
      "name": "you@example.com",
      "type": "user"
    }
  }
]
```

b. 활성 계정 세부 사항을 보고 **tenantId** 값이 사용하려는 서브스크립션과 일치하는지 확인합니다.

```
$ az account show
```

출력 예

```
{
  "environmentName": AzureStackCloud",
  "id": "9bab1460-96d5-40b3-a78e-17b15e978a80",
  "isDefault": true,
  "name": "Subscription Name",
  "state": "Enabled",
  "tenantId": "6057c7e9-b3ae-489d-a54e-de3f6bf6a8ee", (1)
  "user": {
    "name": "you@example.com",
    "type": "user"
  }
}
```

(1) **tenantId** 매개변수 값이 올바른 서브스크립션 ID인지 확인합니다.

c. 올바른 서브스크립션을 사용하지 않는 경우, 활성 서브스크립션을 변경합니다.

```
$ az account set -s <subscription_id> (1)
```

(1) 서브스크립션 ID를 지정합니다.

d. 서브스크립션 ID 업데이트를 확인합니다.

```
$ az account show
```

출력 예

```
{
  "environmentName": AzureStackCloud",
```

```
  "id": "33212d16-bdf6-45cb-b038-f6565b61edda",
  "isDefault": true,
  "name": "Subscription Name",
  "state": "Enabled",
  "tenantId": "8049c7e9-c3de-762d-a54e-dc3f6be6a7ee",
  "user": {
    "name": "you@example.com",
    "type": "user"
  }
}
```


6. 출력의 **tenantId** 및 **id** 매개변수 값을 기록합니다. OpenShift Container Platform 설치 중에 이러한 값이 필요합니다.

7. 계정에 대한 서비스 주체를 생성합니다.

```
$ az ad sp create-for-rbac --role Contributor --name <service_principal> \ 1
  --scopes /subscriptions/<subscription_id> 2
  --years <years> 3
```

1. 서비스 주체 이름을 지정합니다.
2. 서브스크립션 ID를 지정합니다.
3. 년 수를 지정합니다. 기본적으로 서비스 주체는 1년 후에 만료됩니다. **--years** 옵션을 사용하면 서비스 주체의 유효성을 확장할 수 있습니다.

출력 예

```
Creating 'Contributor' role assignment under scope '/subscriptions/<subscription_id>'
The output includes credentials that you must protect. Be sure that you do not
include these credentials in your code or check the credentials into your source
control. For more information, see https://aka.ms/azadsp-cli
{
  "appId": "ac461d78-bf4b-4387-ad16-7e32e328aec6",
  "displayName": <service_principal>",
  "password": "00000000-0000-0000-0000-000000000000",
  "tenantId": "8049c7e9-c3de-762d-a54e-dc3f6be6a7ee"
}
```

8. 이전 출력의 **appId** 및 **password** 매개변수 값을 기록합니다. OpenShift Container Platform 설치 중에 이러한 값이 필요합니다.

추가 리소스

- CCO 모드에 대한 자세한 내용은 Cloud Credential Operator 정보를 참조하십시오.

6.2.5. 다음 단계

- Azure Stack Hub에 대한 IAM 수동 생성 에 따라 Azure Stack Hub 인증 정보를 구성합니다.
- ARM 템플릿을 사용하여 Azure Stack Hub에 클러스터 설치 에 따라 사용자 프로비저닝 인프라가 있는 Azure Stack Hub에 OpenShift Container Platform 클러스터를 설치합니다.

6.3. AZURE STACK HUB의 IAM 수동 생성

클라우드 ID 및 액세스 관리(IAM) API에 연결할 수 없는 환경에서는 클러스터를 설치하기 전에 CCO(Cloud Credential Operator)를 수동 모드로 설정해야 합니다.

6.3.1. kube-system 프로젝트에 관리자 수준 시크릿을 저장하는 대안

CCO(Cloud Credential Operator)는 클라우드 공급자 인증 정보를 Kubernetes CRD(사용자 지정 리소스 정의)로 관리합니다. **install-config.yaml** 파일에서 **credentialsMode** 매개변수의 다른 값을 설정하여 조직의 보안 요구 사항에 맞게 CCO를 구성할 수 있습니다.

추가 리소스

- 사용 가능한 모든 CCO 인증 정보 모드 및 지원되는 플랫폼에 대한 자세한 내용은 Cloud Credential Operator 정보를 참조하십시오.

6.3.2. 수동으로 IAM 생성

CCO(Cloud Credential Operator)는 수동 모드에서만 클라우드 공급자를 지원합니다. 따라서 클라우드 공급자에 대한 IAM(ID 및 액세스 관리) 보안을 지정해야 합니다.

절차

1. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행하여 **install-config.yaml** 파일을 생성합니다.

```
$ openshift-install create install-config --dir <installation_directory>
```

   여기서 **<installation_directory>** 는 설치 프로그램이 파일을 생성하는 디렉터리입니다.

2. **install-config.yaml** 구성 파일을 편집하여 **credentialsMode** 매개 변수가 **Manual**로 설정되도록 합니다.

   install-config.yaml 설정 파일 예

```
apiVersion: v1
baseDomain: cluster1.example.com
credentialsMode: Manual 1
compute:
- architecture: amd64
  hyperthreading: Enabled
...
```

   1 이 행은 **credentialsMode** 매개변수를 **Manual**로 설정하기 위해 추가됩니다.

3. 매니페스트를 생성하려면 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행합니다.

```
$ openshift-install create manifests --dir <installation_directory>
```

   여기서 **<installation_directory>** 는 설치 프로그램이 파일을 생성하는 디렉터리입니다.

4. 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행하여 **openshift-install** 바이너리가 사용하도록 빌드된 OpenShift Container Platform 릴리스 이미지에 대한 세부 정보를 가져옵니다.

```
$ openshift-install version
```

출력 예

```
release image quay.io/openshift-release-dev/ocp-release:4.y.z-x86_64
```

5. 다음 명령을 실행하여 배포 중인 클라우드를 대상으로 하는 이 릴리스 이미지에서 모든 **CredentialsRequest** 오브젝트를 찾습니다.

```
$ oc adm release extract quay.io/openshift-release-dev/ocp-release:4.y.z-x86_64 \
  --credentials-requests \
  --cloud=azure
```

이 명령을 수행하면 각 **CredentialsRequest** 오브젝트에 대해 YAML 파일이 생성됩니다.

샘플 CredentialsRequest 개체

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component-credentials-request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AzureProviderSpec
    roleBindings:
    - role: Contributor
  ...
```

6. 이전에 생성한 **openshift-install** 매니페스트 디렉터리에 시크릿 YAML 파일을 만듭니다. 시크릿은 각 **CredentialsRequest** 오브젝트의 **spec.secretRef**에 정의된 네임 스페이스 및 시크릿 이름을 사용하여 저장해야 합니다.

보안이 포함된 샘플 CredentialsRequest 오브젝트

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component-credentials-request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AzureProviderSpec
    roleBindings:
    - role: Contributor
      ...
  secretRef:
    name: <component-secret>
    namespace: <component-namespace>
  ...
```

-

샘플 Secret 오브젝트

```
apiVersion: v1
kind: Secret
metadata:
  name: <component-secret>
  namespace: <component-namespace>
data:
  azure_subscription_id: <base64_encoded_azure_subscription_id>
  azure_client_id: <base64_encoded_azure_client_id>
  azure_client_secret: <base64_encoded_azure_client_secret>
  azure_tenant_id: <base64_encoded_azure_tenant_id>
  azure_resource_prefix: <base64_encoded_azure_resource_prefix>
  azure_resourcegroup: <base64_encoded_azure_resourcegroup>
  azure_region: <base64_encoded_azure_region>
```

중요

수동으로 유지 관리되는 인증 정보를 사용하는 클러스터를 업그레이드하기 전에 CCO가 업그레이드 가능한 상태인지 확인해야 합니다. 자세한 내용은 클라우드 공급자에 대한 설치 콘텐츠의 "수동으로 유지 관리되는 인증 정보를 사용하여 클러스터 업그레이드" 섹션을 참조하십시오.

6.3.3. 수동으로 유지 관리되는 인증 정보로 클러스터 업그레이드

CCO(Cloud Credential Operator) 수동으로 유지 관리되는 인증 정보가 있는 클러스터의 **Upgradable** 상태는 기본적으로 **False** 입니다.

- 마이너 릴리스(예: 4.8에서 4.9로)의 경우 이 상태는 업데이트된 권한을 처리하고 **CloudCredential** 리소스에 주석을 달아 권한이 다음 버전에 필요에 따라 업데이트되었음을 나타낼 때까지 업그레이드되지 않도록 합니다. 이 주석은 **Upgradable** 상태를 **True**로 변경합니다.

- 예를 들어 4.9.0에서 4.9.1으로 z-stream 릴리스의 경우 권한이 추가되거나 변경되지 않으므로 업그레이드가 차단되지 않습니다.

수동으로 유지 관리되는 인증 정보로 클러스터를 업그레이드하기 전에 업그레이드할 릴리스 이미지에 대한 새 인증 정보를 생성해야 합니다. 또한 기존 인증 정보에 필요한 권한을 검토하고 해당 구성 요소에 대한 새 릴리스에 새 권한 요구 사항을 수용해야 합니다.

프로세스

1. 새 릴리스에 대한 **CredentialsRequest** 사용자 지정 리소스를 추출하고 검사합니다.
   클라우드 공급자용 설치 콘텐츠의 "수동으로 IAM 생성" 섹션에서는 클라우드에 필요한 인증 정보를 획득하고 사용하는 방법을 설명합니다.

2. 클러스터에서 수동으로 유지 관리되는 인증 정보를 업데이트합니다.

   - 새 릴리스 이미지에서 추가한 **CredentialsRequest** 사용자 정의 리소스에 대한 새 시크릿을 생성합니다.

   - 시크릿에 저장된 기존 인증 정보에 대한 **CredentialsRequest** 사용자 정의 리소스가 권한 요구 사항이 변경된 경우 필요에 따라 권한을 업데이트합니다.

3. 모든 보안이 새 릴리스에 대해 올바른 경우 클러스터를 업그레이드할 준비가 되었음을 나타냅니다.

   a. **cluster-admin** 역할의 사용자로 OpenShift Container Platform CLI에 로그인합니다.

   b. **CloudCredential** 리소스를 편집하여 **metadata** 필드 내에 **upgradeable-to** 주석을 추가합니다.

   ```
   $ oc edit cloudcredential cluster
   ```

   추가할 텍스트

   ```
   ...
     metadata:
       annotations:
         cloudcredential.openshift.io/upgradeable-to: <version_number>
   ...
   ```

   여기서 **<version_number>**는 **x.y.z** 형식으로 업그레이드할 버전입니다. 예를 들어 OpenShift Container Platform 4.8.2 의 경우 **4.8.2**입니다.

   주석을 추가한 후 업그레이드 가능 상태가 변경되는 데 몇 분이 소요될 수 있습니다.

4. CCO를 업그레이드할 수 있는지 확인합니다.

   a. 웹 콘솔의 **관리자** 화면에서 **관리자 → 클러스터 설정**으로 이동합니다.

   b. CCO 상태 세부 정보를 보려면 **Cluster Operators** 목록에서 **cloud-credential**을 클릭합니다.

   c. **Conditions** 섹션의 **Upgradeable** 상태가 **False**인 경우 **upgradeable-to 주석**에 오타 오류가 없는지 확인합니다.

Conditions 섹션의 **Upgradeable** 상태가 **True** 이면 OpenShift Container Platform 업그레이드를 시작할 수 있습니다.

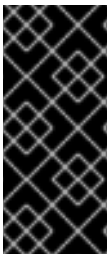
6.3.4. 다음 단계

- ARM 템플릿을 사용하여 Azure Stack Hub에 클러스터 설치 에 따라 사용자 프로비저닝 인프라가 있는 Azure Stack Hub에 OpenShift Container Platform 클러스터를 설치합니다.

6.4. ARM 템플릿을 사용하여 AZURE STACK HUB에 클러스터 설치

OpenShift Container Platform 버전 4.9에서는 사용자가 제공하는 인프라를 사용하여 Microsoft Azure Stack Hub에 클러스터를 설치할 수 있습니다.

이러한 단계를 완료할 수 있도록 지원하거나 자체 모델링에 도움이 되는 여러 가지 ARM(Azure Resource Manager) 템플릿이 제공됩니다.



중요

사용자가 프로비저닝한 인프라 설치를 수행하는 단계는 예시용으로만 제공됩니다. 사용자가 제공하는 인프라를 사용하여 클러스터를 설치하려면 클라우드 공급자 및 OpenShift Container Platform 설치 프로세스에 대한 정보가 필요합니다. 안내된 단계를 수행하거나 자체 모델링에 유용한 몇 가지 ARM 템플릿이 제공됩니다. 여러 다른 방법을 통해 필요한 리소스를 자유롭게 생성할 수도 있습니다. 템플릿은 예시일 뿐입니다.

6.4.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비 에 대한 문서를 읽습니다.
- 클러스터를 호스팅할 Azure Stack Hub 계정을 구성 했습니다.
- Azure CLI를 다운로드하여 컴퓨터에 설치했습니다. Azure 문서의 Azure CLI 설치를 참조하십시오. 아래 문서는 Azure CLI 버전 **2.28.0**을 사용하여 테스트되었습니다. Azure CLI 명령은 사용하는 버전에 따라 다르게 수행될 수 있습니다.
- 방화벽을 사용하며 Telemetry 서비스를 사용할 예정인 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성해야 합니다.

참고

프록시를 구성하는 경우에도 해당 사이트 목록을 검토하십시오.

6.4.2. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.

중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

6.4.3. Azure Stack Hub 프로젝트 구성

OpenShift Container Platform을 설치하려면 먼저 호스팅할 Azure 프로젝트를 구성해야 합니다.

중요

공용 엔드포인트를 통해 사용할 수 있는 모든 Azure Stack Hub 리소스에는 리소스 이름 제한이 적용되며 특정 용어를 사용하는 리소스를 생성할 수 없습니다. Azure Stack Hub가 제한하는 용어 목록은 Azure 문서의 예약된 리소스 이름 오류 해결 을 참조하십시오.

6.4.3.1. Azure Stack Hub 계정 제한

OpenShift Container Platform 클러스터는 여러 Microsoft Azure Stack Hub 구성 요소를 사용하며 Azure Stack Hub의 기본 할당량 유형은 OpenShift Container Platform 클러스터를 설치하는 기능에 영향을 줍니다.

다음 표에는 OpenShift Container Platform 클러스터를 설치 및 실행하는 기능에 영향을 줄 수 있는 제한이 있는 Azure Stack Hub 구성 요소가 요약되어 있습니다.

구성 요소	기본적으로 필요한 구성 요소 수	설명
vCPU	56	기본 클러스터에는 56개의 vCPU가 필요하므로 계정 제한을 늘려야 합니다. 기본적으로 각 클러스터는 다음 인스턴스를 생성합니다. ● 설치 후 제거되는 하나의 부트스트랩 시스템 ● 컨트롤 플레인 시스템 세 개 ● 컴퓨팅 시스템 세 개 부트스트랩, 컨트롤 플레인 및 작업자 시스템은 8개의 vCPU를 사용하는 **Standard_DS4_v2** 가상 시스템을 사용하기 때문에 기본 클러스터에는 56개의 vCPU가 필요합니다. 부트스트랩 노드 VM은 설치 중에만 사용됩니다. 더 많은 작업자 노드를 배포하거나, 자동 크기 조정을 활성화하거나, 대규모 워크로드를 배포하거나, 다른 인스턴스 유형을 사용하려면 필요한 시스템을 클러스터가 배포할 수 있도록 계정의 vCPU 제한값을 더 늘려야 합니다.
OS 디스크	7	VM OS 디스크는 컨트롤 플레인 시스템에 대해 테스트되고 권장되는 최소 처리량 5000 IOPS/200MBps를 유지할 수 있어야 합니다. 이 처리량은 최소 1 TiB Premium SSD (P30)를 보유하여 제공할 수 있습니다. Azure Stack Hub에서 디스크 성능은 SSD 디스크 크기에 직접 의존하므로 **Standard_D8s_v3** 또는 기타 유사한 머신 유형 및 5000 IOPS 대상에서 지원하는 처리량을 달성해야 합니다. 최소 P30 디스크가 필요합니다. 읽기 대기 시간이 짧고 읽기 IOPS 및 처리량을 높이려면 호스트 캐싱을 **ReadOnly**로 설정해야 합니다. VM 메모리 또는 로컬 SSD 디스크에 있는 캐시에서 수행된 읽기는 Blob 스토리지에 있는 데이터 디스크에서 읽기보다 훨씬 빠릅니다.
VNet	1	각 기본 클러스터에는 두 개의 서브넷이 포함된 하나의 가상 네트워크(VNet)가 필요합니다.
네트워크 인터페이스	7	각 기본 클러스터에는 7개의 네트워크 인터페이스가 필요합니다. 더 많은 시스템을 생성하거나 배포된 워크로드가 로드 밸런서를 생성하는 경우 클러스터는 더 많은 네트워크 인터페이스를 사용합니다.

구성 요소	기본적으로 필요한 구성 요소 수	설명
네트워크 보안 그룹	2	각 클러스터는 VNet의 각 서브넷에 대한 네트워크 보안 그룹을 생성합니다. 기본 클러스터는 컨트롤 플레인과 컴퓨팅 노드 서브넷에 대한 네트워크 보안 그룹을 생성합니다: **controlplane**: 어디에서나 포트 6443에서 컨트롤 플레인 시스템에 도달할 수 있습니다. **node**: 포트 80 및 443을 통해 인터넷에서 작업자 노드에 도달할 수 있습니다.
네트워크 로드 밸런서	3	각 클러스터는 다음 로드 밸런서를 생성합니다. **default**: 작업자 시스템에서 포트 80과 443에 대한 요청을 로드 밸런싱하는 공용 IP 주소 **internal**: 컨트롤 플레인 시스템에서 포트 6443과 22623에 대한 요청을 로드 밸런싱하는 개인 IP 주소 **external**: 컨트롤 플레인 시스템에서 포트 6443에 대한 요청을 로드 밸런싱하는 공용 IP 주소 애플리케이션이 더 많은 Kubernetes **LoadBalancer** Service 개체를 생성하면 클러스터가 더 많은 로드 밸런서를 사용합니다.
공용 IP 주소	2	공용 로드 밸런서는 공용 IP 주소를 사용합니다. 또한 부트스트랩 시스템은 공용 IP 주소를 사용하므로 설치 중 문제를 해결하기 위해 시스템으로 SSH를 실행할 수 있습니다. 부트스트랩 노드의 IP 주소는 설치 중에만 사용됩니다.
개인 IP 주소	7	내부 로드 밸런서, 3개의 컨트롤 플레인 시스템 및 3개의 각 작업자 시스템은 각각 개인 IP 주소를 사용합니다.

6.4.3.2. Azure Stack Hub에서 DNS 영역 구성

Azure Stack Hub에 OpenShift Container Platform을 성공적으로 설치하려면 Azure Stack Hub DNS 영역에서 DNS 레코드를 생성해야 합니다. DNS 영역은 도메인에 대한 권한이 있어야 합니다. 등록 프로그램의 DNS 영역을 Azure Stack Hub에 위임하려면 Azure Stack Hub 데이터센터 DNS 통합 에 대한 Microsoft 설명서를 참조하십시오.

DNS 영역을 만드는 예제 에서 Azure의 DNS 솔루션을 확인할 수 있습니다.

6.4.3.3. 인증서 서명 요청 관리

사용자가 프로비저닝하는 인프라를 사용하는 경우 자동 시스템 관리 기능으로 인해 클러스터의 액세스가 제한되므로 설치한 후 클러스터 인증서 서명 요청(CSR)을 승인하는 메커니즘을 제공해야 합니다. **kube-**

controller-manager는 kubelet 클라이언트 CSR만 승인합니다. **machine-approver**는 올바른 시스템에서 발행한 요청인지 확인할 수 없기 때문에 kubelet 자격 증명을 사용하여 요청하는 서비스 인증서의 유효성을 보장할 수 없습니다. kubelet 서빙 인증서 요청의 유효성을 확인하고 요청을 승인하는 방법을 결정하여 구현해야 합니다.

6.4.3.4. 필수 Azure Stack Hub 역할

Microsoft Azure Stack Hub 계정에는 사용하는 서브스크립션에 대한 다음 역할이 있어야 합니다.

- **소유자**

Azure 포털에서 역할을 설정하려면 Microsoft 문서의 역할 기반 액세스 제어를 사용하여 Azure Stack Hub의 리소스에 대한 액세스 관리를 참조하십시오.

6.4.3.5. 서비스 주체 생성

OpenShift Container Platform 및 해당 설치 프로그램은 Azure Resource Manager를 사용하여 Microsoft Azure 리소스를 생성하므로 이를 나타내는 서비스 주체를 생성해야 합니다.

사전 요구 사항

- Azure CLI를 설치 또는 업데이트합니다.
- Azure 계정은 사용하는 서브스크립션에 대한 필요한 역할을 갖습니다.

절차

1. 환경을 등록합니다.

```
$ az cloud register -n AzureStackCloud --endpoint-resource-manager <endpoint> 1
```

1 Azure Resource Manager 끝점을 'https://management.<region>.<fqdn>/'로 지정합니다.

   자세한 내용은 Microsoft 설명서를 참조하십시오.

2. 활성 환경을 설정합니다.

```
$ az cloud set -n AzureStackCloud
```

3. Azure Stack Hub의 특정 API 버전을 사용하도록 환경 구성을 업데이트합니다.

```
$ az cloud update --profile 2019-03-01-hybrid
```

4. Azure CLI에 로그인합니다.

```
$ az login
```

   다중 테넌트 환경에 있는 경우 테넌트 ID도 제공해야 합니다.

5. Azure 계정이 서브스크립션을 사용하는 경우 올바른 서브스크립션을 사용하고 있는지 확인합니다.

a. 사용 가능한 계정 목록을 보고 클러스터에 사용하려는 서브스크립션의 **tenantId** 값을 기록합니다.

```
$ az account list --refresh
```

출력 예

```
[
  {
    "cloudName": AzureStackCloud",
    "id": "9bab1460-96d5-40b3-a78e-17b15e978a80",
    "isDefault": true,
    "name": "Subscription Name",
    "state": "Enabled",
    "tenantId": "6057c7e9-b3ae-489d-a54e-de3f6bf6a8ee",
    "user": {
      "name": "you@example.com",
      "type": "user"
    }
  }
]
```

b. 활성 계정 세부 사항을 보고 **tenantId** 값이 사용하려는 서브스크립션과 일치하는지 확인합니다.

```
$ az account show
```

출력 예

```
{
  "environmentName": AzureStackCloud",
  "id": "9bab1460-96d5-40b3-a78e-17b15e978a80",
  "isDefault": true,
  "name": "Subscription Name",
  "state": "Enabled",
  "tenantId": "6057c7e9-b3ae-489d-a54e-de3f6bf6a8ee", (1)
  "user": {
    "name": "you@example.com",
    "type": "user"
  }
}
```

(1) **tenantId** 매개변수 값이 올바른 서브스크립션 ID인지 확인합니다.

c. 올바른 서브스크립션을 사용하지 않는 경우, 활성 서브스크립션을 변경합니다.

```
$ az account set -s <subscription_id> (1)
```

(1) 서브스크립션 ID를 지정합니다.

d. 서브스크립션 ID 업데이트를 확인합니다.

```
$ az account show
```

출력 예

```
{
  "environmentName": AzureStackCloud",
  "id": "33212d16-bdf6-45cb-b038-f6565b61edda",
  "isDefault": true,
  "name": "Subscription Name",
  "state": "Enabled",
  "tenantId": "8049c7e9-c3de-762d-a54e-dc3f6be6a7ee",
  "user": {
    "name": "you@example.com",
    "type": "user"
  }
}
```

6. 출력의 **tenantId** 및 **id** 매개변수 값을 기록합니다. OpenShift Container Platform 설치 중에 이러한 값이 필요합니다.

7. 계정에 대한 서비스 주체를 생성합니다.

```
$ az ad sp create-for-rbac --role Contributor --name <service_principal> \ 1
  --scopes /subscriptions/<subscription_id> 2
  --years <years> 3
```

1 서비스 주체 이름을 지정합니다.

2 서브스크립션 ID를 지정합니다.

3 년 수를 지정합니다. 기본적으로 서비스 주체는 1년 후에 만료됩니다. **--years** 옵션을 사용하면 서비스 주체의 유효성을 확장할 수 있습니다.

출력 예

```
Creating 'Contributor' role assignment under scope '/subscriptions/<subscription_id>'
The output includes credentials that you must protect. Be sure that you do not
include these credentials in your code or check the credentials into your source
control. For more information, see https://aka.ms/azadsp-cli
{
  "appId": "ac461d78-bf4b-4387-ad16-7e32e328aec6",
  "displayName": <service_principal>",
  "password": "00000000-0000-0000-0000-000000000000",
  "tenantId": "8049c7e9-c3de-762d-a54e-dc3f6be6a7ee"
}
```

8. 이전 출력의 **appId** 및 **password** 매개변수 값을 기록합니다. OpenShift Container Platform 설치 중에 이러한 값이 필요합니다.

추가 리소스

- CCO 모드에 대한 자세한 내용은 Cloud Credential Operator 정보를 참조하십시오.

6.4.4. 설치 프로그램 받기

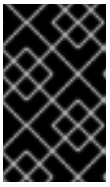
OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

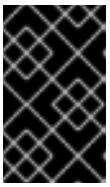
절차

1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.
2. Azure Stack Hub에 클러스터를 설치하는 경우 클라우드 공급자로 **Azure**를 선택합니다.
3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.



중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.



중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 OpenShift Container Platform 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

6.4.5. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 **core** 사용자의 **~/.ssh/authorized_keys** 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 **core**로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다. SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. **./openshift-install gather** 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

참고

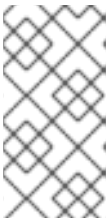
AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: **~/.ssh/id_ed25519**)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 **'~/.ssh** 디렉터리에 있는지 확인하십시오.



참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 **ed25519** 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 **rsa** 또는 **ecdsa** 알고리즘을 사용하는 키를 생성합니다.

2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 **~/.ssh/id_ed25519.pub** 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 **./openshift-install gather** 명령을 사용하려는 경우 필요합니다.

참고

일부 배포에서는 **~/.ssh/id_rsa** 및 **~/.ssh/id_dsa**와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. **ssh-agent** 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. **ssh-agent**에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: **~/.ssh/id_ed25519**).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

6.4.6. Azure Stack Hub에 대한 설치 파일 생성

사용자 프로비저닝 인프라를 사용하여 Microsoft Azure Stack Hub에 OpenShift Container Platform을 설치하려면 설치 프로그램에서 클러스터를 배포하는 데 필요한 파일을 생성하고 클러스터가 사용할 시스템만 생성하도록 파일을 수정해야 합니다. **install-config.yaml** 파일을 수동으로 생성한 다음 Kubernetes 매니페스트 및 Ignition 구성 파일을 생성하고 사용자 지정합니다. 또한 설치 준비 단계에서 별도의 **var** 파티션을 먼저 설정할 수 있는 옵션이 있습니다.

6.4.6.1. 수동으로 설치 구성 파일 만들기

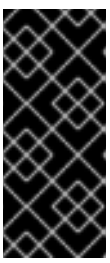
사전 요구 사항

- 로컬 시스템에 설치 프로그램에 제공할 SSH 공개 키가 있습니다. 키는 디버깅 및 재해 복구를 위해 클러스터 노드에 대한 SSH 인증에 사용됩니다.
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿이 있습니다.

프로세스

1. 필요한 설치 자산을 저장할 설치 디렉터리를 만듭니다.

```
$ mkdir <installation_directory>
```



중요

디렉터리를 만들어야 합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

2. 샘플 **install-config.yaml** 파일 템플릿을 사용자 지정하여 **<installation_directory>**에 저장합니다.

참고

이 설정 파일의 이름을 **install-config.yaml**로 지정해야 합니다.

Azure Stack Hub에 대해 다음과 같이 수정합니다.

a. **compute** 풀에 대해 **replicas** 매개변수를 **0**으로 설정합니다.

```
compute:
- hyperthreading: Enabled
  name: worker
  platform: {}
  replicas: 0 1
```

1 **0**으로 설정합니다.

나중에 컴퓨팅 시스템이 수동으로 프로비저닝됩니다.

b. **install-config.yaml** 파일의 **platform.azure** 섹션을 업데이트하여 Azure Stack Hub 설정을 구성합니다.

```
platform:
  azure:
    armEndpoint: <azurestack_arm_endpoint> 1
    baseDomainResourceGroupName: <resource_group> 2
    cloudName: AzureStackCloud 3
    region: <azurestack_region> 4
```

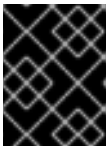
1 **https://management.local.azurestack.external** 과 같이 Azure Stack Hub 환경의 Azure Resource Manager 끝점을 지정합니다.

2 기본 도메인의 DNS 영역을 포함하는 리소스 그룹의 이름을 지정합니다.

3 적절한 Azure API 엔드포인트로 Azure SDK를 구성하는 데 사용되는 Azure Stack Hub 환경을 지정합니다.

4 Azure Stack Hub 리전의 이름을 지정합니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.



중요

install-config.yaml 파일은 설치 과정의 다음 단계에서 사용됩니다. 이 시점에서 이를 백업해야 합니다.

6.4.6.2. Azure Stack Hub의 샘플 사용자 지정 install-config.yaml 파일

install-config.yaml 파일을 사용자 지정하여 OpenShift Container Platform 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.

중요

이 샘플 YAML 파일은 참조용으로만 제공됩니다. 수동으로 생성한 설치 구성 파일에 매개 변수 값을 입력하기 위한 리소스로 사용합니다.

```
apiVersion: v1
baseDomain: example.com
controlPlane: 1
  name: master
  replicas: 3
compute: 2
- name: worker
  platform: {}
  replicas: 0
metadata:
  name: test-cluster 3
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  azure:
    armEndpoint: azurestack_arm_endpoint 4
    baseDomainResourceGroupName: resource_group 5
    region: azure_stack_local_region 6
    resourceGroupName: existing_resource_group 7
    outboundType: Loadbalancer
    cloudName: AzureStackCloud 8
pullSecret: '{"auths": ...}' 9
fips: false 10
additionalTrustBundle: | 11
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
sshKey: ssh-ed25519 AAAA... 12
```

1 2 **controlPlane** 섹션은 단일 매핑이지만 compute 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

3 클러스터 이름을 지정합니다.

4 Azure Stack Hub Operator가 제공하는 Azure Resource Manager 엔드포인트를 지정합니다.

5 기본 도메인의 DNS 영역을 포함하는 리소스 그룹의 이름을 지정합니다.

6 Azure Stack Hub 로컬 리전의 이름을 지정합니다.

7 클러스터를 설치할 기존 리소스 그룹의 이름을 지정합니다. 정의되지 않은 경우 클러스터에 새 리소스 그룹이 생성됩니다.

8 대상 플랫폼으로 Azure Stack Hub 환경을 지정합니다.

9 클러스터를 인증하는 데 필요한 풀 시크릿을 지정합니다.

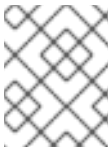
10 FIPS 모드 활성화 또는 비활성화 여부입니다. 기본적으로 FIPS 모드는 비활성화됩니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.

중요

FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.

11 Azure Stack Hub 환경에서 내부 인증 기관(CA)을 사용하는 경우 필요한 인증서 번들을 **.pem** 형식으로 추가합니다.

12 선택사항으로, 클러스터의 시스템에 액세스하는 데 사용할 **sshKey** 값을 제공할 수도 있습니다.



참고

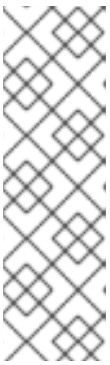
설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

6.4.6.3. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 HTTP 또는 HTTPS 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 OpenShift Container Platform 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 API에 대한 호출을 포함하여 모든 클러스터 발신(Egress) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.



참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

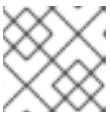
Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 Red Hat OpenStack Platform (RHOSP)에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. 클러스터 외부에서 HTTP 연결을 구축하는 데 사용할 프록시 URL입니다. URL 스키마는 **http**여야 합니다.
2. 클러스터 외부에서 HTTPS 연결을 구축하는 데 사용할 프록시 URL입니다.
3. 대상 도메인 이름, IP 주소 또는 프록시에서 제외할 기타 네트워크 CIDR로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. ***를 사용하여 모든 대상에 대해 프록시를 바이패스합니다.
4. 이 값을 제공하면 설치 프로그램에서 추가 CA 인증서를 보유할 **openshift -config 네임스페이스에 user-ca- bundle** 이라는 구성 맵을 생성합니다. **additionalTrustBundle** 및 하나 이상의 프록시 설정을 제공하는 경우 **프록시** 오브젝트는 **trustedCA** 필드의 **user-ca-bundle** 구성 맵을 참조하도록 구성됩니다. 그러면 Cluster Network Operator에서 **trustedCA** 매개변수에 대해 지정된 콘텐츠를 RHCOS 신뢰 번들과 병합하는 **trusted-ca-bundle** 구성 맵을 생성합니다. 프록시의 ID 인증서를 RHCOS 트러스트 번들에 있는 기관에서 서명하지 않은 경우 **additionalTrustBundle** 필드가 있어야 합니다.



참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 OpenShift Container Platform을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 nil이 됩니다.

참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

6.4.6.4. ARM 템플릿의 공통 변수 내보내기

Microsoft Azure Stack Hub에서 사용자 제공 인프라 설치를 지원하기 위해 제공된 ARM(Azure Resource Manager) 템플릿과 함께 사용되는 공통 변수 세트를 내보내야 합니다.

참고

특정 ARM 템플릿에는 추가적으로 내보낸 변수도 필요할 수 있습니다. 관련 프로시저에서 자세히 설명합니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

절차

1. 제공된 ARM 템플릿이 사용할 **install-config.yaml**에 있는 공통 변수를 내보냅니다.

```
$ export CLUSTER_NAME=<cluster_name> 1
$ export AZURE_REGION=<azure_region> 2
$ export SSH_KEY=<ssh_key> 3
$ export BASE_DOMAIN=<base_domain> 4
$ export BASE_DOMAIN_RESOURCE_GROUP=<base_domain_resource_group> 5
```

1. **install-config.yaml** 파일의 **.metadata.name** 속성 값입니다.
2. 클러스터를 배포할 리전입니다. **install-config.yaml** 파일의 **.platform.azure.region** 속성 값입니다.
3. 문자열 형태의 SSH RSA 공개 키 파일입니다. 공백이 포함되어 있으므로 SSH 키를 따옴표로 묶어야 합니다. **install-config.yaml** 파일의 **.sshKey** 속성 값입니다.
4. 클러스터를 배포할 기본 도메인. 기본 도메인은 클러스터용으로 생성한 DNS 영역에 해당합니다. **install-config.yaml** 파일의 **.baseDomain** 속성 값입니다.
5. DNS 영역이 존재하는 리소스 그룹입니다. **install-config.yaml** 파일의 **.platform.azure.baseDomainResourceGroupName** 속성 값입니다.

예를 들면 다음과 같습니다.

```
$ export CLUSTER_NAME=test-cluster
$ export AZURE_REGION=centralus
$ export SSH_KEY="ssh-rsa xxx/xxx/xxx= user@email.com"
$ export BASE_DOMAIN=example.com
$ export BASE_DOMAIN_RESOURCE_GROUP=ocp-cluster
```

2. kubeadmin 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1. **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

6.4.6.5. Kubernetes 매니페스트 및 Ignition 설정 파일 생성

일부 클러스터 정의 파일을 수정하고 클러스터 시스템을 수동으로 시작해야 하므로 클러스터가 시스템을 구성하는 데 필요한 Kubernetes 매니페스트 및 Ignition 구성 파일을 사용자가 생성해야 합니다.

설치 구성 파일은 Kubernetes 매니페스트로 변환됩니다. 매니페스트는 나중에 클러스터 머신을 구성하는 데 사용되는 Ignition 구성 파일로 래핑됩니다.

중요

- OpenShift Container Platform 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *만료된 컨트롤 플레인 인증서에서 복구* 문서를 참조하십시오.

- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램을 가져오셨습니다.
- **install-config.yaml** 설치 구성 파일을 생성하셨습니다.

프로세스

1. OpenShift Container Platform 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터에 대한 Kubernetes 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>**는 사용자가 만든 **install-config.yaml** 파일이 포함된 설치 디렉터리를 지정합니다.

2. 컨트롤 플레인 시스템을 정의하는 Kubernetes 매니페스트 파일을 제거합니다.

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

이러한 파일을 제거하면 클러스터가 컨트롤 플레인 시스템을 자동으로 생성하지 못합니다.

3. 작업자 시스템을 정의하는 Kubernetes 매니페스트 파일을 제거합니다.

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

작업자 시스템은 사용자가 직접 생성하고 관리하기 때문에 초기화할 필요가 없습니다.

4. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes 매니페스트 파일의 **mastersSchedulable** 매개변수가 **false**로 설정되어 있는지 확인합니다. 이 설정으로 인해 컨트롤 플레인 머신에서 포드가 예약되지 않습니다.

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** 파일을 엽니다.

   b. **mastersSchedulable** 매개변수를 찾아서 값을 **False**로 설정되어 있는지 확인합니다.

c. 파일을 저장하고 종료합니다.

5. 선택사항: Ingress Operator가 사용자 대신 DNS 레코드를 생성하지 못하도록 하려면 **<installation_directory>/manifests/cluster-dns-02-config.yml** DNS 구성 파일에서 **privateZone** 및 **publicZone** 섹션을 제거합니다.

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

1 2 이 섹션을 완전히 제거합니다.

제거한 경우 나중에 인그레스 DNS 레코드를 수동으로 추가해야 합니다.

6. 선택 사항: Azure Stack Hub 환경에서 내부 인증 기관(CA)을 사용하는 경우 **<installation_directory>/manifests/cluster-proxy-01-config.yaml** 파일에서 **.spec.trustedCA.name** 필드를 업데이트하여 **user-ca-bundle** 을 사용합니다.

```
...
spec:
  trustedCA:
    name: user-ca-bundle
...
```

나중에 CA를 포함하도록 부트스트랩 ignition을 업데이트해야 합니다.

7. 사용자 프로비저닝 인프라에서 Azure를 구성할 때, ARM(Azure Resource Manager) 템플릿에서 나중에 사용할 수 있도록 매니페스트 파일에 정의된 일부 공통 변수를 내보내야 합니다.

a. 다음 명령을 사용하여 인프라 ID를 내보냅니다.

```
$ export INFRA_ID=<infra_id> 1
```

1 OpenShift Container Platform 클러스터에 **<cluster_name>-<random_string>** 형식으로 식별자(**INFRA_ID**)가 할당되었습니다. 제공된 ARM 템플릿을 사용해서 생성된 대부분의 리소스에 대해 기본 이름으로 사용됩니다. **manifests/cluster-infrastructure-02-config.yml** 파일의 **.status.infrastructureName** 속성 값입니다.

b. 다음 명령을 사용하여 리소스 그룹을 내보냅니다:

```
$ export RESOURCE_GROUP=<resource_group> 1
```

1 이 Azure 배포에서 생성된 모든 리소스는 리소스 그룹의 일부로 존재합니다. 또한 리소스 그룹 이름은 **<cluster_name>-<random_string>-rg** 형식의 **INFRA_ID**를 기반으로 합니다. **manifests/cluster-infrastructure-02-config.yml** 파일의 **.status.platformStatus.azure.resourceGroupName** 속성 값입니다.

8. 클라우드 인증 정보를 수동으로 생성합니다.

   a. 설치 프로그램이 포함된 디렉터리에서 **openshift-install** 바이너리가 다음을 사용하도록 빌드된 OpenShift Container Platform 릴리스 이미지에 대한 세부 정보를 가져옵니다.

   ```
   $ openshift-install version
   ```

   출력 예

   ```
   release image quay.io/openshift-release-dev/ocp-release:4.y.z-x86_64
   ```

   b. 배포중인 클라우드를 대상으로하는이 릴리스 이미지에서 모든 **CredentialsRequest** 개체를 찾습니다.

   ```
   $ oc adm release extract quay.io/openshift-release-dev/ocp-release:4.y.z-x86_64 --
   credentials-requests --cloud=azure
   ```

   이 명령을 수행하면 각 **CredentialsRequest** 오브젝트에 대해 YAML 파일이 생성됩니다.

   샘플 CredentialsRequest 개체

   ```
   apiVersion: cloudcredential.openshift.io/v1
   kind: CredentialsRequest
   metadata:
     labels:
       controller-tools.k8s.io: "1.0"
     name: openshift-image-registry-azure
     namespace: openshift-cloud-credential-operator
   spec:
     secretRef:
       name: installer-cloud-credentials
       namespace: openshift-image-registry
     providerSpec:
       apiVersion: cloudcredential.openshift.io/v1
       kind: AzureProviderSpec
       roleBindings:
       - role: Contributor
   ```

   c. 이전에 생성한 **openshift-install** 매니페스트 디렉터리에 시크릿 YAML 파일을 만듭니다. 시크릿은 각 **CredentialsRequest** 오브젝트의 **spec.secretRef**에 정의된 네임 스페이스 및 시크릿 이름을 사용하여 저장해야 합니다. 시크릿 데이터의 형식은 클라우드 공급자마다 다릅니다.

   d. CCO(Cloud Config Operator)가 비활성화된 매니페스트 디렉터리에 **cco-configmap.yaml** 파일을 생성합니다.

   샘플 ConfigMap 오브젝트

```
apiVersion: v1
kind: ConfigMap
metadata:
name: cloud-credential-operator-config
namespace: openshift-cloud-credential-operator
  annotations:
    release.openshift.io/create-only: "true"
data:
  disabled: "true"
```

9. Ignition 구성 파일을 생성하려면 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행합니다.

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>**는 동일한 설치 디렉터리를 지정합니다.

설치 디렉터리의 부트스트랩, 컨트롤 플레인 및 컴퓨팅 노드에 대한 Ignition 구성 파일이 생성됩니다. **kubeadmin-password** 및 **kubeconfig** 파일은 **./<installation_directory>/auth** 디렉터리에 생성됩니다.

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

6.4.6.6. 선택 사항: 별도의 /var 파티션 만들기

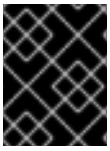
OpenShift Container Platform의 디스크 파티션 설정은 설치 프로그램에 맡기는 것이 좋습니다. 그러나 확장하려는 파일 시스템의 일부에 별도의 파티션을 생성해야 하는 경우가 있습니다.

OpenShift 컨테이너 플랫폼은 **/var** 파티션 또는 **/var**의 하위 디렉터리 중 하나에 스토리지를 연결하는 단일 파티션의 추가를 지원합니다. 예를 들면 다음과 같습니다.

- **/var/lib/containers**: 시스템에 더 많은 이미지와 컨테이너가 추가됨에 따라 확장될 수 있는 컨테이너 관련 콘텐츠를 보관합니다.
- **/var/lib/etcd**: etcd 스토리지의 성능 최적화와 같은 목적으로 별도로 보관할 데이터를 보관합니다.
- **/var**: 감사 등의 목적에 맞게 별도로 분리하여 보관해야 하는 데이터를 보관합니다.

/var 디렉터리의 콘텐츠를 별도로 저장하면 필요에 따라 해당 영역에 대한 스토리지 확장을 보다 용이하게 하고 나중에 OpenShift Container Platform을 다시 설치하여 해당 데이터를 그대로 보존할 수 있습니다. 이 방법을 사용하면 모든 컨테이너를 다시 가져올 필요가 없으며 시스템을 업데이트할 때 대용량 로그 파일을 복사할 필요도 없습니다.

RHCOS(Red Hat Enterprise Linux CoreOS)를 새로 설치하기 전에 **/var가 있어야 하므로 다음 절차에서는 OpenShift Container Platform 설치의 openshift-install 준비 단계 중에 삽입되는 머신 구성 매니페스트를 생성하여 별도의 /var** 파티션을 설정합니다.



중요

이 절차에서 별도의 **/var** 파티션을 생성하기 위해 단계에 따라 이 섹션의 뒷부분에서 설명한 대로 Kubernetes 매니페스트 및 Ignition 구성 파일을 다시 생성할 필요가 없습니다.

프로세스

1. OpenShift Container Platform 설치 파일을 저장할 디렉터리를 만듭니다.

```
$ mkdir $HOME/clusterconfig
```

2. **openshift-install**을 실행하여 **manifest** 및 **openshift** 하위 디렉터리에 파일 세트를 만듭니다. 프롬프트가 표시되면 시스템 질문에 대답합니다.

```
$ openshift-install create manifests --dir $HOME/clusterconfig
```

출력 예

```
? SSH Public Key ...
INFO Credentials loaded from the "myprofile" profile in file "/home/myuser/.aws/credentials"
INFO Consuming Install Config from target directory
INFO Manifests created in: $HOME/clusterconfig/manifests and
$HOME/clusterconfig/openshift
```

3. 선택 사항: 설치 프로그램이 **clusterconfig/openshift** 디렉터리에 매니페스트를 생성했는지 확인합니다.

```
$ ls $HOME/clusterconfig/openshift/
```

출력 예

```
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

4. 추가 파티션을 구성하는 Butane 구성을 생성합니다. 예를 들어 파일 이름을 **$HOME/clusterconfig/98-var-partition.bu** 로 지정하고, 디스크 장치 이름을 **worker** 시스템의 스토리지 장치 이름으로 변경하고 스토리지 크기를 적절하게 설정합니다. 이 예에서는 **/var** 디렉토리를 별도의 파티션에 배치합니다.

```
variant: openshift
version: 4.9.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/<device_name> 1
    partitions:
    - label: var
```

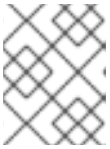
```
    start_mib: <partition_start_offset> 2
    size_mib: <partition_size> 3
filesystems:
  - device: /dev/disk/by-partlabel/var
    path: /var
    format: xfs
    mount_options: [defaults, prjquota] 4
    with_mount_unit: true
```

1 파티션을 설정해야하는 디스크 저장 장치 이름입니다.

2 데이터 파티션을 부트 디스크에 추가할 때 최소 25000MiB(메비 바이트)가 권장됩니다. 루트 파일 시스템은 지정된 오프셋까지 사용 가능한 모든 공간을 채우기 위해 자동으로 크기가 조정됩니다. 값이 지정되지 않거나 지정된 값이 권장 최소값보다 작으면 생성되는 루트 파일 시스템의 크기가 너무 작아지고 RHCOS를 나중에 다시 설치할 때 데이터 파티션의 첫 번째 부분을 덮어 쓸 수 있습니다.

3 데이터 파티션의 크기(MB)입니다.

4 컨테이너 스토리지에 사용되는 파일 시스템에 **prjquota** 마운트 옵션을 활성화해야 합니다.



참고

별도의 **/var** 파티션을 만들 때 다른 인스턴스 유형에 동일한 장치 이름이 없는 경우 작업자 노드에 다른 인스턴스 유형을 사용할 수 없습니다.

5. Butane 구성에서 매니페스트를 생성하여 **clusterconfig/openshift** 디렉터리에 저장합니다. 예를 들어 다음 명령을 실행합니다.

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

6. **openshift-install**을 다시 실행하여 **manifest** 및 **openshift** 하위 디렉터리의 파일 세트에서 Ignition 구성을 만듭니다.

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

이제 Ignition 구성 파일을 설치 절차에 대한 입력으로 사용하여 RHCOS (Red Hat Enterprise Linux CoreOS) 시스템을 설치할 수 있습니다.

6.4.7. Azure 리소스 그룹 생성

Microsoft Azure 리소스 그룹을 생성해야 합니다. 이는 Azure Stack Hub에 OpenShift Container Platform 클러스터를 설치하는 동안 사용됩니다.

사전 요구 사항

- Azure 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.

프로세스

- 지원되는 Azure 리전에서 리소스 그룹을 생성합니다.

```
$ az group create --name ${RESOURCE_GROUP} --location ${AZURE_REGION}
```

6.4.8. RHCOS 클러스터 이미지 및 부트스트랩 Ignition 구성 파일 업로드

Azure 클라이언트는 로컬로 존재하는 파일을 기반으로 한 배포를 지원하지 않습니다. 따라서 RHCOS VHD(가상 하드 디스크) 클러스터 이미지와 부트스트랩 Ignition 구성 파일을 스토리지 컨테이너에 복사하여 저장해야 배포 중에 액세스할 수 있습니다.

사전 요구 사항

- Azure 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.

프로세스

1. VHD 클러스터 이미지를 저장할 Azure 스토리지 계정을 생성합니다.

```
$ az storage account create -g ${RESOURCE_GROUP} --location ${AZURE_REGION} --
name ${CLUSTER_NAME}sa --kind Storage --sku Standard_LRS
```

주의

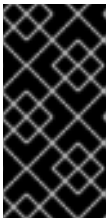
Azure 스토리지 계정 이름은 3자에서 24자 사이여야 하며 숫자와 소문자만 사용해야 합니다. **CLUSTER_NAME** 변수가 이러한 제한 사항을 따르지 않으면 Azure 스토리지 계정 이름을 수동으로 정의해야 합니다. Azure 스토리지 계정 이름 제한 사항에 대한 자세한 내용은 Azure 문서의 스토리지 계정 이름 오류 해결을 참조하십시오.

2. 스토리지 계정 키를 환경 변수 형태로 내보냅니다.

```
$ export ACCOUNT_KEY=`az storage account keys list -g ${RESOURCE_GROUP} --
account-name ${CLUSTER_NAME}sa --query "[0].value" -o tsv`
```

3. 사용할 RHCOS 버전을 선택하고 VHD의 URL을 환경 변수로 내보냅니다.

```
$ export COMPRESSED_VHD_URL=`curl -s
https://raw.githubusercontent.com/openshift/installer/release-4.9/data/data/rhcos-amd64.json
| jq -r '(.baseURI + .images.azurestack.path)'`
```



중요

RHCOS 이미지는 OpenShift Container Platform 릴리스에 따라 변경되지 않을 수 있습니다. 설치하는 OpenShift Container Platform 버전 이하에서 가장 높은 버전의 이미지를 지정해야 합니다. 지원되는 경우 OpenShift Container Platform 버전과 일치하는 이미지 버전을 사용합니다.

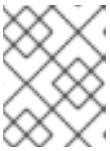
4. VHD용 스토리지 컨테이너를 생성합니다.

```
$ az storage container create --name vhd --account-name ${CLUSTER_NAME}sa --account-key ${ACCOUNT_KEY}
```

5. 압축된 RHCOS VHD 파일을 로컬에서 다운로드합니다.

```
$ curl -O -L ${COMPRESSED_VHD_URL}
```

6. VHD 파일의 압축을 풉니다.



참고

압축 해제된 VHD 파일은 약 16GB이므로 호스트 시스템에 사용 가능한 공간이 16GB인지 확인합니다. 업로드한 후에는 VHD 파일을 삭제할 수 있습니다.

7. 선택한 VHD를 Blob에 복사합니다.

```
$ az storage blob upload --account-name ${CLUSTER_NAME}sa --account-key ${ACCOUNT_KEY} -c vhd -n "rhcos.vhd" -f rhcos-<rhcos_version>-azurestack.x86_64.vhd
```

8. Blob 스토리지 컨테이너를 만들고 생성된 **bootstrap.ign** 파일을 업로드합니다.

```
$ az storage container create --name files --account-name ${CLUSTER_NAME}sa --account-key ${ACCOUNT_KEY} --public-access blob
```

```
$ az storage blob upload --account-name ${CLUSTER_NAME}sa --account-key ${ACCOUNT_KEY} -c "files" -f "<installation_directory>/bootstrap.ign" -n "bootstrap.ign"
```

6.4.9. DNS 영역 생성 예

사용자 프로비저닝 인프라를 사용하는 클러스터에는 DNS 레코드가 필요합니다. 시나리오에 맞는 DNS 전략을 선택해야 합니다.

이 예에서는 Azure Stack Hub의 데이터센터 DNS 통합이 사용되므로 DNS 영역을 생성합니다.

참고

DNS 영역은 클러스터 배포와 동일한 리소스 그룹에 존재할 필요는 없으며 원하는 기본 도메인의 조직에 이미 있을 수 있습니다. 이 경우 DNS 영역 생성을 건너뛸 수 있습니다. 이전에 생성한 설치 구성이 해당 시나리오를 반영하는지 확인하십시오.

사전 요구 사항

- Azure 계정을 구성하십시오.

- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.

프로세스

- **BASE_DOMAIN_RESOURCE_GROUP** 환경 변수에서 내보낸 리소스 그룹에 새 DNS 영역을 생성합니다.

```
$ az network dns zone create -g ${BASE_DOMAIN_RESOURCE_GROUP} -n
${CLUSTER_NAME}.${BASE_DOMAIN}
```

  이미 존재하는 DNS 영역을 사용하는 경우 이 단계를 건너뛸 수 있습니다.

해당 섹션을 방문하여 Azure Stack Hub에서 DNS 영역을 구성하는 방법 에 대해 자세히 알아볼 수 있습니다.

6.4.10. Azure Stack Hub에서 VNet 생성

Microsoft Azure Stack Hub에서 OpenShift Container Platform 클러스터가 사용할 가상 네트워크(VNet)를 생성해야 합니다. 요구사항에 맞춰 VNet을 사용자 지정할 수 있습니다. VNet을 생성하는 한 가지 방법은 제공된 ARM(Azure Resource Manager) 템플릿을 수정하는 것입니다.

참고

Azure Stack Hub 인프라를 생성하는 데 제공된 ARM 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- Azure 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.

프로세스

1. 이 항목의 **VNet에 대한 ARM 템플릿** 섹션에서 템플릿을 복사하여 클러스터의 설치 디렉터리에 **01_vnet.json**으로 저장합니다. 이 템플릿은 클러스터에 필요한 VNet을 설명합니다.
2. **az** CLI를 사용하여 배포를 생성합니다.

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/01_vnet.json" \
  --parameters baseName="${INFRA_ID}" 1
```

1 리소스 이름에 사용될 기본 이름; 일반적으로 클러스터의 인프라 ID입니다.

6.4.10.1. VNet용 ARM 템플릿

다음 ARM(Azure Resource Manager) 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 VNet을 배포할 수 있습니다.

예 6.1. 01_vnet.json ARM 템플릿

link:https://raw.githubusercontent.com/openshift/installer/release-4.9/upi/azurestack/01_vnet.json[]

6.4.11. Azure Stack Hub 인프라에 대한 RHCOS 클러스터 이미지 배포

OpenShift Container Platform 노드에 대해서는 Microsoft Azure Stack Hub에 유효한 RHCOS(Red Hat Enterprise Linux CoreOS) 이미지를 사용해야 합니다.

사전 요구 사항

- Azure 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.
- RHCOS VHD(가상 하드 디스크) 클러스터 이미지를 Azure 스토리지 컨테이너에 저장하십시오.
- 부트스트랩 ignition 구성 파일을 Azure 스토리지 컨테이너에 저장하십시오.

절차

1. 이 항목의 **이미지 스토리지에 대한 ARM 템플릿** 섹션에서 템플릿을 복사하여 클러스터의 설치 디렉터리에 **02_storage.json**으로 저장합니다. 이 템플릿은 클러스터에 필요한 이미지 스토리지를 설명합니다.
2. RHCOS VHD BLOB URL을 변수 형태로 내보냅니다.

```
$ export VHD_BLOB_URL=`az storage blob url --account-name ${CLUSTER_NAME}sa --
account-key ${ACCOUNT_KEY} -c vhd -n "rhcos.vhd" -o tsv`
```

3. 클러스터 이미지를 배포합니다.

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/02_storage.json" \
  --parameters vhdBlobURL="${VHD_BLOB_URL}" \ 1
  --parameters baseName="${INFRA_ID}" 2
```

1. 마스터 및 작업자 시스템을 생성하는 데 사용되는 RHCOS VHD의 Blob URL.
2. 리소스 이름에 사용될 기본 이름; 일반적으로 클러스터의 인프라 ID입니다.

6.4.11.1. 이미지 스토리지를 위한 ARM 템플릿

다음 ARM(Azure Resource Manager) 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 저장된 RHCOS(Red Hat Enterprise Linux CoreOS) 이미지를 배포할 수 있습니다.

예 6.2. 02_storage.json ARM 템플릿

link:https://raw.githubusercontent.com/openshift/installer/release-4.9/upi/azurestack/02_storage.json[]

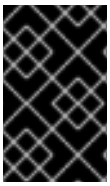
6.4.12. 사용자 프로비저닝 인프라에 대한 네트워킹 요구사항

모든 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템이 부팅 중에 Ignition 구성 파일을 가져오려면 **initramfs**에 네트워킹을 구성해야 합니다.

6.4.12.1. 네트워크 연결 요구사항

OpenShift Container Platform 클러스터 구성 요소가 통신할 수 있도록 시스템 간 네트워크 연결을 구성해야 합니다. 각 시스템에서 클러스터에 있는 다른 모든 시스템의 호스트 이름을 확인할 수 있어야 합니다.

이 섹션에서는 필요한 포트에 대해 자세히 설명합니다.



중요

연결된 OpenShift Container Platform 환경에서 모든 노드는 플랫폼 컨테이너의 이미지를 가져오고 Red Hat에 원격 측정 데이터를 제공하기 위해 인터넷에 액세스할 수 있어야 합니다.

표 6.1. 모든 시스템 간 통신에 사용되는 포트

프로토콜	포트	설명
ICMP	해당 없음	네트워크 연결성 테스트
TCP	**1936**	메트릭
	9000-9999	**9100-9101** 포트의 노드 내보내기 및 **9099** 포트의 Cluster Version Operator를 포함한 호스트 수준 서비스.
	10250-10259	Kubernetes에서 예약하는 기본 포트
	10256	openshift-sdn
UDP	**4789**	VXLAN 및 Geneve
	6081	VXLAN 및 Geneve
	9000-9999	**9100-9101** 포트의 노드 내보내기를 포함한 호스트 수준 서비스.
	500	IPsec IKE 패킷
	4500	IPsec NAT-T 패킷
TCP/UDP	**30000-32767**	Kubernetes 노드 포트
ESP	해당 없음	IPsec Encapsulating Security Payload (ESP)

표 6.2. 모든 시스템과 컨트롤 플레인 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**6443**	Kubernetes API

표 6.3. 컨트롤 플레인 머신 간 통신에 사용되는 포트

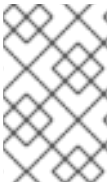
프로토콜	포트	설명
TCP	**2379-2380**	etcd 서버 및 피어 포트

6.4.13. Azure Stack Hub에서 네트워킹 및 로드 밸런싱 구성 요소 생성

Microsoft Azure Stack Hub에서 OpenShift Container Platform 클러스터가 사용할 네트워킹 및 로드 밸런싱을 구성해야 합니다. 이러한 구성 요소를 생성하는 한 가지 방법은 제공된 ARM(Azure Resource Manager) 템플릿을 수정하는 것입니다.

로드 밸런싱에는 다음과 같은 DNS 레코드가 필요합니다.

- DNS **영역** 의 API 공용 로드 밸런서에 대한 API DNS 레코드입니다.
- DNS 영역의 API 내부 로드 밸런서에 대한 **api-int** DNS 레코드입니다.



참고

Azure Stack Hub 인프라를 생성하는 데 제공된 ARM 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- Azure 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.
- Azure Stack Hub에서 VNet 및 관련 서브넷을 생성하고 구성합니다.

절차

1. 이 항목의 **네트워크 및 로드 밸런서에 대한 ARM 템플릿** 섹션에서 템플릿을 복사하여 클러스터의 설치 디렉터리에 **03_infra.json**으로 저장합니다. 이 템플릿은 클러스터에 필요한 네트워킹 및 로드 밸런싱 개체를 설명합니다.
2. **az** CLI를 사용하여 배포를 생성합니다.

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/03_infra.json" \
  --parameters baseName="${INFRA_ID}" 1
```

1 리소스 이름에 사용될 기본 이름; 일반적으로 클러스터의 인프라 ID입니다.

3. **api** DNS 레코드 및 **api-int** DNS 레코드를 만듭니다. API DNS 레코드를 생성할 때 **${BASE_DOMAIN_RESOURCE_GROUP}** 변수는 DNS 영역이 존재하는 리소스 그룹을 가리켜야 합니다.

   a. 다음 변수를 내보냅니다.

   ```
   $ export PUBLIC_IP=`az network public-ip list -g ${RESOURCE_GROUP} --query "[?name=='${INFRA_ID}-master-pip'] | [0].ipAddress" -o tsv`
   ```

   b. 다음 변수를 내보냅니다.

   ```
   $ export PRIVATE_IP=`az network lb frontend-ip show -g "$RESOURCE_GROUP" --lb-name "${INFRA_ID}-internal" -n internal-lb-ip --query "privateIpAddress" -o tsv`
   ```

   c. 새 **DNS 영역에 API** DNS 레코드를 생성합니다.

   ```
   $ az network dns record-set a add-record -g ${BASE_DOMAIN_RESOURCE_GROUP} -z ${CLUSTER_NAME}.${BASE_DOMAIN} -n api -a ${PUBLIC_IP} --ttl 60
   ```

   기존 DNS 영역에 클러스터를 추가하는 경우 대신 **api** DNS 레코드를 생성할 수 있습니다.

   ```
   $ az network dns record-set a add-record -g ${BASE_DOMAIN_RESOURCE_GROUP} -z ${BASE_DOMAIN} -n api.${CLUSTER_NAME} -a ${PUBLIC_IP} --ttl 60
   ```

   d. 새 DNS 영역에 **api-int** DNS 레코드를 생성합니다.

   ```
   $ az network dns record-set a add-record -g ${BASE_DOMAIN_RESOURCE_GROUP} -z "${CLUSTER_NAME}.${BASE_DOMAIN}" -n api-int -a ${PRIVATE_IP} --ttl 60
   ```

   기존 DNS 영역에 클러스터를 추가하는 경우 대신 **api-int** DNS 레코드를 생성할 수 있습니다.

   ```
   $ az network dns record-set a add-record -g ${BASE_DOMAIN_RESOURCE_GROUP} -z ${BASE_DOMAIN} -n api-int.${CLUSTER_NAME} -a ${PRIVATE_IP} --ttl 60
   ```

6.4.13.1. 네트워크 및 로드 밸런서에 대한 ARM 템플릿

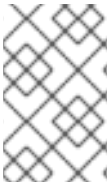
다음 ARM(Azure Resource Manager) 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 네트워킹 개체 및 로드 밸런서를 배포할 수 있습니다.

예 6.3. 03_infra.json ARM 템플릿

```
link:https://raw.githubusercontent.com/openshift/installer/release-4.9/upi/azurestack/03_infra.json[]
```

6.4.14. Azure Stack Hub에서 부트스트랩 시스템 생성

OpenShift Container Platform 클러스터 초기화 중에 사용할 부트스트랩 시스템을 Microsoft Azure Stack Hub에 생성해야 합니다. 이 시스템을 생성하는 한 가지 방법은 제공된 ARM(Azure Resource Manager) 템플릿을 수정하는 것입니다.



참고

부트스트랩 시스템을 생성하는 데 제공된 ARM 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- Azure 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.
- Azure Stack Hub에서 VNet 및 관련 서브넷을 생성하고 구성합니다.
- Azure Stack Hub에서 네트워킹 및 로드 밸런서를 생성하고 구성합니다.
- 컨트롤 플레인 및 컴퓨팅 역할을 생성합니다.

절차

1. 이 항목의 **부트스트랩 시스템에 대한 ARM 템플릿** 섹션에서 템플릿을 복사하여 클러스터의 설치 디렉터리에 **04_bootstrap.json**으로 저장합니다. 이 템플릿은 클러스터에 필요한 부트스트랩 시스템을 설명합니다.
2. 부트스트랩 URL 변수를 내보냅니다.

```
$ export BOOTSTRAP_URL=`az storage blob url --account-name ${CLUSTER_NAME}sa --account-key ${ACCOUNT_KEY} -c "files" -n "bootstrap.ign" -o tsv`
```

3. 부트스트랩 ignition 변수를 내보냅니다.
   a. 환경에서 공용 CA(인증 기관)를 사용하는 경우 다음 명령을 실행합니다.

```
$ export BOOTSTRAP_IGNITION=`jq -rcnM --arg v "3.2.0" --arg url ${BOOTSTRAP_URL} '{ignition:{version:$v,config:{replace:{source:$url}}}}' | base64 | tr -d '\n'`
```

   b. 환경에서 내부 CA를 사용하는 경우 부트스트랩 가상 머신이 스토리지 계정에서 부트스트랩 ignition을 가져올 수 있도록 PEM 인코딩 번들을 부트스트랩 ignition에 추가해야 합니다. 다음 명령을 실행하여 CA가 **CA라는 파일에 있다고 가정합니다.**

```
$ export CA="data:text/plain;charset=utf-8;base64,$(cat CA.pem |base64 |tr -d '\n')"
```

```
$ export BOOTSTRAP_IGNITION=`jq -rcnM --arg v "3.2.0" --arg url "$BOOTSTRAP_URL" --arg cert "$CA" '{ignition:{version:$v,security:{tls:{certificateAuthorities:[{source:$cert}]}},config:{replace:{source:$url}}}}' | base64 | tr -d '\n'`
```

4. **az** CLI를 사용하여 배포를 생성합니다.

```
$ az deployment group create --verbose -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/04_bootstrap.json" \
  --parameters bootstrapIgnition="${BOOTSTRAP_IGNITION}" \ 1
```

```
--parameters sshKeyData="${SSH_KEY}" \ 2
--parameters baseName="${INFRA_ID}" \ 3
--parameters diagnosticsStorageAccountName="${CLUSTER_NAME}sa" 4
```

1. 부트스트랩 클러스터의 부트스트랩 ignition 내용입니다.
2. 문자열 형태의 SSH RSA 공개 키 파일입니다.
3. 리소스 이름에 사용될 기본 이름; 일반적으로 클러스터의 인프라 ID입니다.
4. 클러스터의 스토리지 계정 이름입니다.

6.4.14.1. 부트스트랩 시스템용 ARM 템플릿

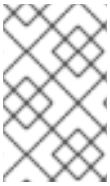
다음 ARM(Azure Resource Manager) 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 부트스트랩 시스템을 배포할 수 있습니다.

예 6.4. 04_bootstrap.json ARM 템플릿

link:https://raw.githubusercontent.com/openshift/installer/release-4.9/upi/azurestack/04_bootstrap.json[]

6.4.15. Azure Stack Hub에서 컨트롤 플레인 시스템 생성

클러스터가 사용할 Microsoft Azure Stack Hub에서 컨트롤 플레인 시스템을 생성해야 합니다. 이러한 시스템을 생성하는 한 가지 방법은 제공된 ARM(Azure Resource Manager) 템플릿을 수정하는 것입니다.



참고

컨트롤 플레인 시스템을 생성하는 데 제공된 ARM 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- Azure 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.
- Azure Stack Hub에서 VNet 및 관련 서브넷을 생성하고 구성합니다.
- Azure Stack Hub에서 네트워킹 및 로드 밸런서를 생성하고 구성합니다.
- 컨트롤 플레인 및 컴퓨팅 역할을 생성합니다.
- 부트스트랩 시스템을 생성합니다.

프로세스

1. 이 항목의 **컨트롤 플레인 시스템에 대한 ARM 템플릿** 섹션에서 템플릿을 복사하여 클러스터의 설치 디렉터리에 **05_masters.json**으로 저장합니다. 이 템플릿은 클러스터에 필요한 컨트롤 플레인 시스템을 설명합니다.

2. 컨트롤 플레인 시스템 배포에 필요한 다음 변수를 내보냅니다.

```
$ export MASTER_IGNITION=`cat <installation_directory>/master.ign | base64 | tr -d '\n'`
```

3. **az** CLI를 사용하여 배포를 생성합니다.

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/05_masters.json" \
  --parameters masterIgnition="${MASTER_IGNITION}" \ 1
  --parameters sshKeyData="${SSH_KEY}" \ 2
  --parameters baseName="${INFRA_ID}" \ 3
  --parameters diagnosticsStorageAccountName="${CLUSTER_NAME}sa" 4
```

1. 컨트롤 플레인 노드(마스터 노드라고도 함)의 Ignition 내용입니다.
2. 문자열 형태의 SSH RSA 공개 키 파일입니다.
3. 리소스 이름에 사용될 기본 이름; 일반적으로 클러스터의 인프라 ID입니다.
4. 클러스터의 스토리지 계정 이름입니다.

6.4.15.1. 컨트롤 플레인 시스템에 대한 ARM 템플릿

다음 ARM(Azure Resource Manager) 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 컨트롤 플레인 시스템을 배포할 수 있습니다.

예 6.5. 05_masters.json ARM 템플릿

link:https://raw.githubusercontent.com/openshift/installer/release-4.9/upi/azurestack/05_masters.json[]

6.4.16. 부트스트랩이 완료될 때까지 기다렸다가 Azure Stack Hub에서 부트스트랩 리소스 제거

Microsoft Azure Stack Hub에 필요한 인프라를 모두 생성한 후 설치 프로그램으로 생성한 Ignition 구성 파일을 사용하여 프로비저닝한 시스템에서 부트스트랩 프로세스가 완료될 때까지 기다립니다.

사전 요구 사항

- Azure 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.
- Azure Stack Hub에서 VNet 및 관련 서브넷을 생성하고 구성합니다.
- Azure Stack Hub에서 네트워킹 및 로드 밸런서를 생성하고 구성합니다.
- 컨트롤 플레인 및 컴퓨팅 역할을 생성합니다.
- 부트스트랩 시스템을 생성합니다.

- 컨트롤 플레인 시스템을 생성합니다.

프로세스

1. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

```
$ ./openshift-install wait-for bootstrap-complete --dir <installation_directory> \ 1
    --log-level info 2
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2 다른 설치 세부 사항을 보려면 **info** 대신 **warn**, **debug** 또는 **error**를 지정합니다.

FATAL 경고 없이 명령이 종료되면 프로덕션 컨트롤 플레인이 초기화된 것입니다.

2. 부트스트랩 리소스를 삭제합니다.

```
$ az network nsg rule delete -g ${RESOURCE_GROUP} --nsg-name ${INFRA_ID}-nsg --
name bootstrap_ssh_in
$ az vm stop -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap
$ az vm deallocate -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap
$ az vm delete -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap --yes
$ az disk delete -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap_OSDisk --no-
wait --yes
$ az network nic delete -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap-nic --no-
wait
$ az storage blob delete --account-key ${ACCOUNT_KEY} --account-name
${CLUSTER_NAME}sa --container-name files --name bootstrap.ign
$ az network public-ip delete -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap-
ssh-pip
```

참고

부트스트랩 서버를 삭제하지 않으면 API 트래픽이 부트스트랩 서버로 라우팅되므로 설치가 성공하지 못할 수 있습니다.

6.4.17. Azure Stack Hub에서 추가 작업자 시스템 생성

개별 인스턴스를 따로 시작하거나 자동 확장 그룹과 같은 클러스터 외부의 자동화된 프로세스를 통해 클러스터에서 사용할 작업자 시스템을 Microsoft Azure Stack Hub에 생성할 수 있습니다. OpenShift Container Platform의 기본 제공 클러스터 확장 메커니즘과 시스템 API를 활용할 수도 있습니다.

예에서는 ARM(Azure Resource Manager) 템플릿을 사용하여 인스턴스 하나를 수동으로 시작합니다. 파일에 **06_worker.json** 유형의 추가 리소스를 포함시켜 추가 인스턴스를 시작할 수 있습니다.

참고

작업자 시스템을 생성하는 데 제공된 ARM 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- Azure 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.
- Azure Stack Hub에서 VNet 및 관련 서브넷을 생성하고 구성합니다.
- Azure Stack Hub에서 네트워킹 및 로드 밸런서를 생성하고 구성합니다.
- 컨트롤 플레인 및 컴퓨팅 역할을 생성합니다.
- 부트스트랩 시스템을 생성합니다.
- 컨트롤 플레인 시스템을 생성합니다.

프로세스

1. 이 항목의 **작업자 시스템에 대한 ARM 템플릿** 섹션에서 템플릿을 복사하여 클러스터의 설치 디렉터리에 **06_workers.json**으로 저장합니다. 이 템플릿에서 클러스터에 필요한 작업자 시스템을 설명합니다.
2. 작업자 시스템 배포에 필요한 다음 변수를 내보냅니다.

```
$ export WORKER_IGNITION=`cat <installation_directory>/worker.ign | base64 | tr -d '\n'`
```

3. **az** CLI를 사용하여 배포를 생성합니다.

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/06_workers.json" \
  --parameters workerIgnition="${WORKER_IGNITION}" \ 1
  --parameters sshKeyData="${SSH_KEY}" \ 2
  --parameters baseName="${INFRA_ID}" 3
  --parameters diagnosticsStorageAccountName="${CLUSTER_NAME}sa" 4
```

1. 작업자 노드의 ignition 내용입니다.
2. 문자열 형태의 SSH RSA 공개 키 파일입니다.
3. 리소스 이름에 사용될 기본 이름; 일반적으로 클러스터의 인프라 ID입니다.
4. 클러스터의 스토리지 계정 이름입니다.

6.4.17.1. 작업자 시스템에 대한 ARM 템플릿

다음 ARM(Azure Resource Manager) 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 작업자 시스템을 배포할 수 있습니다.

예 6.6. 06_workers.json ARM 템플릿

link:https://raw.githubusercontent.com/openshift/installer/release-4.9/upi/azurestack/06_workers.json[]

6.4.18. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift CLI(**oc**)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 **oc**를 설치할 수 있습니다.

중요

이전 버전의 **oc**를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 **oc**를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치
다음 절차를 사용하여 Linux에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Linux Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. **oc** 바이너리를 **PATH**에 있는 디렉터리에 배치합니다.
PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치
다음 절차에 따라 Windows에 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Windows Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. ZIP 프로그램으로 아카이브의 압축을 풉니다.
5. **oc** 바이너리를 **PATH**에 있는 디렉터리로 이동합니다.
PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치
다음 절차에 따라 macOS에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 MacOSX Client** 항목 옆에 있는 **지금 다운로드를** 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 해제하고 압축을 풉니다.
5. **oc** 바이너리 PATH의 디렉터리로 이동합니다.
   PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

   ```
   $ echo $PATH
   ```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

6.4.19. CLI를 사용하여 클러스터에 로그인

클러스터 **kubeconfig** 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. **kubeconfig** 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- **oc** CLI를 설치했습니다.

프로세스

1. **kubeadmin** 인증 정보를 내보냅니다.

   ```
   $ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
   ```

   1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 **oc** 명령을 성공적으로 실행할 수 있는지 확인합니다.

   ```
   $ oc whoami
   ```

   출력 예

   ```
   system:admin
   ```

-

6.4.20. 머신의 인증서 서명 요청 승인

클러스터에 시스템을 추가하면 추가한 시스템별로 보류 중인 인증서 서명 요청(CSR)이 두 개씩 생성됩니다. 이러한 CSR이 승인되었는지 확인해야 하며, 필요한 경우 이를 직접 승인해야 합니다. 클라이언트 요청을 먼저 승인한 다음 서버 요청을 승인해야 합니다.

사전 요구 사항

- 클러스터에 시스템을 추가했습니다.

프로세스

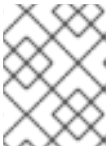
1. 클러스터가 시스템을 인식하는지 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.22.1
master-1  Ready     master  63m  v1.22.1
master-2  Ready     master  64m  v1.22.1
```

출력에 생성된 모든 시스템이 나열됩니다.



참고

이전 출력에는 일부 CSR이 승인될 때까지 컴퓨팅 노드(작업자 노드라고도 함)가 포함되지 않을 수 있습니다.

2. 보류 중인 CSR을 검토하고 클러스터에 추가한 각 시스템에 대해 **Pending** 또는 **Approved** 상태의 클라이언트 및 서버 요청이 표시되는지 확인합니다.

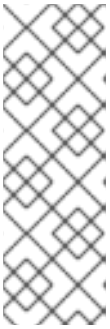
```
$ oc get csr
```

출력 예

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
...
```

예에서는 두 시스템이 클러스터에 참여하고 있습니다. 목록에는 승인된 CSR이 더 많이 나타날 수도 있습니다.

3. CSR이 승인되지 않은 경우, 추가된 시스템에 대한 모든 보류 중인 CSR이 **Pending** 상태로 전환된 후 클러스터 시스템의 CSR을 승인합니다.



참고

CSR은 교체 주기가 자동으로 만료되므로 클러스터에 시스템을 추가한 후 1시간 이내에 CSR을 승인하십시오. 한 시간 내에 승인하지 않으면 인증서가 교체되고 각 노드에 대해 두 개 이상의 인증서가 표시됩니다. 이러한 인증서를 모두 승인해야 합니다. 클라이언트 CSR이 승인되면 Kubelet은 인증서에 대한 보조 CSR을 생성하므로 수동 승인이 필요합니다. 그러면 Kubelet에서 동일한 매개변수를 사용하여 새 인증서를 요청하는 경우 인증서 갱신 요청은 **machine-approver**에 의해 자동으로 승인됩니다.

참고

베어 메탈 및 기타 사용자 프로비저닝 인프라와 같이 머신 API를 사용하도록 활성화되지 않는 플랫폼에서 실행되는 클러스터의 경우 CSR(Kubelet service Certificate Request)을 자동으로 승인하는 방법을 구현해야 합니다. 요청이 승인되지 않으면 API 서버가 kubelet에 연결될 때 서비스 인증서가 필요하므로 **oc exec**, **oc rsh**, **oc logs** 명령을 성공적으로 수행할 수 없습니다. Kubelet 엔드 포인트에 연결하는 모든 작업을 수행하려면 이 인증서 승인이 필요합니다. 이 방법은 새 CSR을 감시하고 CSR이 **system:node** 또는 **system:admin** 그룹의 **node-bootstrapper** 서비스 계정에 의해 제출되었는지 확인하고 노드의 ID를 확인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>**은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

참고

일부 Operator는 일부 CSR이 승인될 때까지 사용할 수 없습니다.

4. 이제 클라이언트 요청이 승인되었으므로 클러스터에 추가한 각 머신의 서버 요청을 검토해야 합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE    REQUESTOR                                                   CONDITION
csr-bfd72   5m26s  system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s  system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 나머지 CSR이 승인되지 않고 **Pending** 상태인 경우 클러스터 머신의 CSR을 승인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>**은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. 모든 클라이언트 및 서버 CSR이 승인된 후 머신은 **Ready** 상태가 됩니다. 다음 명령을 실행하여 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0 Ready    master  73m  v1.22.1
master-1 Ready    master  73m  v1.22.1
master-2 Ready    master  74m  v1.22.1
worker-0 Ready    worker  11m  v1.22.1
worker-1 Ready    worker  11m  v1.22.1
```

참고

머신이 **Ready** 상태로 전환하는 데 서버 CSR의 승인 후 몇 분이 걸릴 수 있습니다.

추가 정보

- CSR에 대한 자세한 내용은 인증서 서명 요청을 참조하십시오.

6.4.21. 인그레스 DNS 레코드 추가

Kubernetes 매니페스트를 생성하고 Ignition 구성을 생성할 때 DNS 영역 구성을 제거한 경우, 인그레스 로드 밸런서를 가리키는 DNS 레코드를 수동으로 생성해야 합니다. 와일드카드 ***.apps.{baseDomain}.** 또는 특정 레코드를 생성할 수 있습니다. 사용자 요구사항에 따라 A, CNAME 및 기타 레코드를 사용할 수 있습니다.

사전 요구 사항

- 프로비저닝한 인프라를 사용하여 Microsoft Azure Stack Hub에 OpenShift Container Platform 클러스터를 배포했습니다.
- OpenShift CLI(**oc**)를 설치합니다.
- Azure CLI를 설치 또는 업데이트합니다.

프로세스

1. 인그레스 라우터가 로드 밸런서를 생성하고 **EXTERNAL-IP** 필드를 채웠는지 확인합니다.

```
$ oc -n openshift-ingress get service router-default
```

출력 예

```
NAME            TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)                      AGE
router-default  LoadBalancer   172.30.20.10    35.130.120.110
80:32288/TCP,443:31215/TCP   20
```

2. 인그레스 라우터 IP를 변수 형태로 내보냅니다.

```
$ export PUBLIC_IP_ROUTER=`oc -n openshift-ingress get service router-default --no-
headers | awk '{print $4}'`
```

3. DNS 영역에 ***.apps** 레코드를 추가합니다.

   a. 이 클러스터를 새 DNS 영역에 추가하는 경우 다음을 실행합니다.

```
$ az network dns record-set a add-record -g ${BASE_DOMAIN_RESOURCE_GROUP} -
z ${CLUSTER_NAME}.${BASE_DOMAIN} -n *.apps -a ${PUBLIC_IP_ROUTER} --ttl 300
```

   b. 이 클러스터를 기존 DNS 영역에 추가하는 경우 다음을 실행합니다.

```
$ az network dns record-set a add-record -g ${BASE_DOMAIN_RESOURCE_GROUP} -
z ${BASE_DOMAIN} -n *.apps.${CLUSTER_NAME} -a ${PUBLIC_IP_ROUTER} --ttl 300
```

와일드카드를 사용하지 않고 명시적 도메인을 추가하는 방식을 선호하면 클러스터의 현재 경로 각각에 대한 항목을 생성할 수 있습니다.

```
$ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}{"\n"}{end}
{end}' routes
```

출력 예

```
oauth-openshift.apps.cluster.basedomain.com
console-openshift-console.apps.cluster.basedomain.com
downloads-openshift-console.apps.cluster.basedomain.com
alertmanager-main-openshift-monitoring.apps.cluster.basedomain.com
grafana-openshift-monitoring.apps.cluster.basedomain.com
prometheus-k8s-openshift-monitoring.apps.cluster.basedomain.com
```

6.4.22. 사용자 프로비저닝 인프라에서 Azure Stack Hub 설치 완료

Microsoft Azure Stack Hub 사용자 프로비저닝 인프라에서 OpenShift Container Platform 설치를 시작한 후 클러스터가 준비될 때까지 클러스터 이벤트를 모니터링할 수 있습니다.

사전 요구 사항

- 사용자 프로비저닝 Azure Stack Hub 인프라에 OpenShift Container Platform 클러스터의 부트스트랩 시스템을 배포합니다.
- **oc** CLI를 설치하고 로그인하십시오.

프로세스

- 클러스터 설치를 완료합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

출력 예

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

중요

- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

7장. GCP에 설치

7.1. GCP에 설치할 준비

7.1.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비 에 대한 문서를 읽습니다.

7.1.2. GCP에 OpenShift Container Platform을 설치하기 위한 요구사항

GCP(Google Cloud Platform)에 OpenShift Container Platform을 설치하기 전에 서비스 계정을 생성하고 GCP 프로젝트를 구성해야 합니다. 프로젝트 생성, API 서비스 활성화, DNS, GCP 계정 제한 및 지원되는 GCP 리전 구성에 대한 자세한 내용은 GCP 프로젝트 구성을 참조하십시오.

사용자의 환경에서 클라우드 ID 및 액세스 관리(IAM) API에 액세스할 수 없거나 **kube-system** 네임스페이스에 관리자 수준의 인증 정보 시크릿을 저장하지 않으려면 다른 옵션에 대해 GCP용 IAM 수동 생성을 참조하십시오.

7.1.3. GCP에 OpenShift Container Platform을 설치할 방법 선택

설치 관리자 프로비저닝 또는 사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치할 수 있습니다. 기본 설치 유형은 설치 프로그램이 클러스터의 기본 인프라를 프로비저닝하는 설치 관리자 프로비저닝 인프라를 사용합니다. 제공하는 인프라에 OpenShift Container Platform도 설치할 수 있습니다. 설치 프로그램에서 프로비저닝한 인프라를 사용하지 않는 경우 클러스터 리소스를 직접 관리하고 유지보수해야 합니다.

설치 관리자 프로비저닝 및 사용자 프로비저닝 설치 프로세스에 대한 자세한 내용은 설치 프로세스를 참조하십시오.

7.1.3.1. 설치 관리자 프로비저닝 인프라를 사용하여 클러스터 설치

다음 방법 중 하나를 사용하여 OpenShift Container Platform 설치 프로그램에서 프로비저닝한 GCP 인프라에 컨테이너를 설치할 수 있습니다.

- **GCP에 클러스터 빠른 설치**: OpenShift Container Platform 설치 프로그램에서 프로비저닝한 GCP 인프라에 OpenShift Container Platform을 설치할 수 있습니다. 기본 구성 옵션을 사용하여 빠르게 클러스터를 설치할 수 있습니다.
- **GCP에 사용자 지정 클러스터 설치**: 설치 프로그램이 프로비저닝하는 GCP 인프라에 사용자 지정 클러스터를 설치할 수 있습니다. 설치 프로그램을 통해 설치 단계에서 일부 사용자 지정을 적용할 수 있습니다. 다른 많은 사용자 정의 옵션은 설치 후사용할 수 있습니다.
- **네트워크 사용자 지정으로 GCP에 클러스터 설치**: 설치 중에 OpenShift Container Platform 네트워크 구성을 사용자 지정할 수 있으므로 클러스터가 기존 IP 주소 할당과 공존하고 네트워크 요구사항을 준수할 수 있습니다.
- **제한된 네트워크의 GCP에 클러스터 설치**: 설치 릴리스 콘텐츠의 내부 미러를 사용하여 설치 프로그램 프로비저닝 인프라에 있는 GCP에 OpenShift Container Platform을 설치할 수 있습니다. 이 방법을 사용하여 소프트웨어 구성 요소를 받기 위해 활성 인터넷 연결이 필요하지 않은 클러스터를 설치할 수 있습니다. 미러링된 콘텐츠를 사용하여 OpenShift Container Platform 클러스터를 설치할 수는 있지만 GCP API를 사용하려면 여전히 클러스터에 인터넷 액세스가 필요합니다.

- **기존 가상 프라이빗 클라우드에 클러스터 설치**: 기존 GCP VPC(Virtual Private Cloud)에 OpenShift Container Platform을 설치할 수 있습니다. 새 계정 또는 인프라 생성 시 제한 사항과 같이 회사의 지침에 따라 설정되는 제약 조건이 있는 경우 이 설치 방법을 사용할 수 있습니다.

- **기존 VPC에 프라이빗 클러스터 설치**: 개인 GCP VPC에 프라이빗 클러스터를 설치할 수 있습니다. 이 방법을 사용하여 인터넷에 표시되지 않는 내부 네트워크에 OpenShift Container Platform을 배포할 수 있습니다.

7.1.3.2. 사용자 프로비저닝 인프라를 사용하여 클러스터 설치

다음 방법 중 하나를 사용하여 프로비저닝하는 GCP 인프라에 클러스터를 설치할 수 있습니다.

- **사용자 프로비저닝 인프라로 GCP에 클러스터 설치**: 사용자가 제공하는 GCP 인프라에 OpenShift Container Platform을 설치할 수 있습니다. 제공된 Deployment Manager 템플릿을 사용하여 설치를 지원할 수 있습니다.

- **GCP의 사용자 프로비저닝 인프라에 공유 VPC로 클러스터 설치**: 제공된 Deployment Manager 템플릿을 사용하여 공유 VPC 인프라에 GCP 리소스를 생성할 수 있습니다.

- **사용자 프로비저닝 인프라가 있는 제한된 네트워크의 GCP에 클러스터 설치**: 사용자 프로비저닝 인프라가 있는 제한된 네트워크의 GCP에 OpenShift Container Platform을 설치할 수 있습니다. 설치 릴리스 컨텐츠의 내부 미러를 만들어 소프트웨어 구성 요소를 가져오는 데 활성 인터넷 연결이 필요하지 않은 클러스터를 설치할 수 있습니다. 이 설치 방법을 사용하여 클러스터가 외부 콘텐츠에 대해 조직의 제어 조건을 충족하는 컨테이너 이미지만 사용하도록 할 수 있습니다.

7.1.4. 다음 단계

- GCP 프로젝트 구성

7.2. GCP 프로젝트 구성

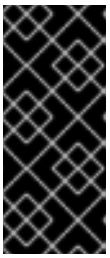
OpenShift Container Platform을 설치하려면 먼저 호스팅할 GCP(Google Cloud Platform) 프로젝트를 구성해야 합니다.

7.2.1. GCP 프로젝트 생성

OpenShift Container Platform을 설치하려면 클러스터를 호스팅할 GCP(Google Cloud Platform) 계정에 프로젝트를 생성해야 합니다.

프로세스

- OpenShift Container Platform 클러스터를 호스팅할 프로젝트를 생성합니다. GCP 문서의 프로젝트 생성 및 관리 단원을 참조하십시오.



중요

설치 관리자 프로비저닝 인프라를 사용하는 경우 GCP 프로젝트는 Premium Network Service Tier를 사용해야 합니다. 설치 프로그램을 사용하여 설치된 클러스터의 Standard Network Service Tier는 지원되지 않습니다. 설치 프로그램은 **api-int.<cluster_name>.<base_domain>** URL에 대한 내부 로드 밸런싱을 구성합니다.

7.2.2. GCP에서 API 서비스 활성화

GCP(Google Cloud Platform) 프로젝트에서 OpenShift Container Platform 설치를 완료하려면 여러 API 서비스에 액세스해야 합니다.

사전 요구 사항

- 클러스터를 호스팅할 프로젝트 생성을 완료했습니다.

프로세스

- 클러스터를 호스팅하는 프로젝트에서 다음과 같은 필수 API 서비스를 활성화합니다. GCP 문서의 서비스 활성화 단원을 참조하십시오.

표 7.1. 필수 API 서비스

API 서비스	콘솔 서비스 이름
컴퓨팅 엔진 API	**compute.googleapis.com**
Google 클라우드 API	**cloudapis.googleapis.com**
클라우드 리소스 관리자 API	**cloudresourcemanager.googleapis.com**
Google DNS API	**dns.googleapis.com**
IAM 서비스 계정 자격 증명 API	**iamcredentials.googleapis.com**
IAM(ID 및 액세스 관리) API	**iam.googleapis.com**
서비스 관리 API	**servicemanagement.googleapis.com**
서비스 사용량 API	**serviceusage.googleapis.com**
Google 클라우드 스토리지 JSON API	**storage-api.googleapis.com**
클라우드 스토리지	**storage-component.googleapis.com**

7.2.3. GCP용 DNS 구성

OpenShift Container Platform을 설치하려면 사용하는 GCP(Google Cloud Platform) 계정에 OpenShift Container Platform 클러스터를 호스팅하는 프로젝트와 동일한 프로젝트에 전용 퍼블릭 호스팅 영역이 있어야 합니다. 도메인에 대한 권한도 이 영역에 있어야 합니다. DNS 서비스는 클러스터와 외부 연결에 필요한 클러스터 DNS 확인 및 이름 조회 기능을 제공합니다.

프로세스

1. 도메인 또는 하위 도메인과 등록 기관을 식별합니다. 기존 도메인 및 등록 기관을 이전하거나 GCP 또는 다른 소스를 통해 새 도메인과 등록 기관을 구할 수 있습니다.

참고

새 도메인을 구입하는 경우, 관련 DNS 변경사항이 전파되는 데 시간이 걸릴 수 있습니다. Google을 통한 도메인 구매에 대한 자세한 내용은 Google 도메인 을 참조하십시오.

2. GCP 프로젝트에서 도메인 또는 하위 도메인의 퍼블릭 호스팅 영역을 생성합니다. GCP 문서의 퍼블릭 영역 생성 단원을 참조하십시오.
적절한 루트 도메인(예: **openshiftcorp.com**) 또는 하위 도메인(예: **clusters.openshiftcorp.com**)을 사용합니다.

3. 호스팅 영역 레코드에서 권한이 있는 새 이름 서버를 추출합니다. GCP 문서의 클라우드 DNS 이름 서버 조회 단원을 참조하십시오.
일반적으로 네 가지 이름 서버가 있습니다.

4. 도메인에서 사용하는 이름 서버의 등록 기관 레코드를 업데이트합니다. 예를 들어, 사용자 도메인을 Google 도메인에 등록한 경우 Google 도메인 도움말의 사용자 지정 이름 서버로 전환하는 방법 항목을 참조하십시오.

5. 루트 도메인을 Google Cloud DNS로 마이그레이션했으면 DNS 레코드를 마이그레이션합니다. GCP 문서의 클라우드 DNS로 마이그레이션 을 참조하십시오.

6. 하위 도메인을 사용하는 경우, 회사의 프로시저에 따라 상위 도메인에 위임 레코드를 추가합니다. 이 과정에 회사의 IT 부서 또는 회사의 루트 도메인 및 DNS 서비스를 제어하는 부서에 요청하는 일도 포함될 수 있습니다.

7.2.4. GCP 계정 제한

OpenShift Container Platform 클러스터는 여러 GCP(Google Cloud Platform) 구성 요소를 사용하지만 기본 할당량이 기본 OpenShift Container Platform 클러스터 설치가 가능할지 여부에 영향을 미치지 않습니다.

컴퓨팅 및 컨트롤 플레인 시스템 세 개가 포함된 기본 클러스터는 다음과 같은 리소스를 사용합니다. 일부 리소스는 부트스트랩 프로세스 중에만 필요하며 클러스터 배포 후 제거됩니다.

표 7.2. 기본 클러스터에서 사용되는 GCP 리소스

서비스	구성 요소	위치	필요한 총 리소스	부트스트랩 후 제거된 리소스
서비스 계정	IAM	글로벌	5	0
방화벽 규칙	컴퓨팅	글로벌	11	1
전송 규칙	컴퓨팅	글로벌	2	0
사용 중인 글로벌 IP 주소	컴퓨팅	글로벌	4	1
상태 검사	컴퓨팅	글로벌	3	0
이미지	컴퓨팅	글로벌	1	0

서비스	구성 요소	위치	필요한 총 리소스	부트스트랩 후 제거된 리소스
네트워크	컴퓨팅	글로벌	2	0
고정 IP 주소	컴퓨팅	리전	4	1
라우터	컴퓨팅	글로벌	1	0
라우트	컴퓨팅	글로벌	2	0
서브네트워크	컴퓨팅	글로벌	2	0
대상 풀	컴퓨팅	글로벌	3	0
CPU	컴퓨팅	리전	28	4
영구 디스크 SSD(GB)	컴퓨팅	리전	896	128

참고

설치하는 동안 할당량이 충분하지 않으면 설치 프로그램에서 초과된 할당량과 리전을 모두 안내하는 오류 메시지를 표시합니다.

실제 클러스터 크기, 예상 클러스터 증가, 계정과 연결된 다른 클러스터의 사용량을 모두 고려해야 합니다. CPU, 고정 IP 주소, 영구 디스크 SSD(스토리지) 할당량이 가장 부족하기 쉬운 할당량입니다.

다음 리전 중 하나에서 클러스터를 배포하려는 경우, 최대 스토리지 할당량을 초과할 것이며, CPU 할당량 제한을 초과할 가능성도 있습니다.

- **asia-east2**
- **asia-northeast2**
- **asia-south1**
- **australia-southeast1**
- **europe-north1**
- **europe-west2**
- **europe-west3**
- **europe-west6**
- **northamerica-northeast1**
- **southamerica-east1**
- **us-west2**

GCP 콘솔에서 리소스 할당량을 늘릴 수는 있지만 지원 티켓을 제출해야 할 수도 있습니다. OpenShift Container Platform 클러스터를 설치하기 전에 지원 티켓을 해결할 시간이 충분하도록 조기에 클러스터 크기를 계획해야 합니다.

7.2.5. GCP에서 서비스 계정 생성

OpenShift Container Platform에는 Google API의 데이터에 액세스하기 위한 인증 및 승인을 제공하는 GCP(Google Cloud Platform) 서비스 계정이 필요합니다. 프로젝트에 필요한 역할이 포함된 기존 IAM 서비스 계정이 없으면 새로 생성해야 합니다.

사전 요구 사항

- 클러스터를 호스팅할 프로젝트 생성을 완료했습니다.

프로세스

1. OpenShift Container Platform 클러스터를 호스팅하는 데 사용하는 프로젝트에 서비스 계정을 생성합니다. GCP 문서의 서비스 계정 생성 단원을 참조하십시오.
2. 서비스 계정에 적절한 권한을 부여합니다. 뒤따르는 개별 권한을 부여하거나 **Owner** 역할을 할당할 수 있습니다. 서비스 계정에 특정 리소스에 대한 역할 부여 를 참조하십시오.

참고

서비스 계정을 프로젝트 소유자로 지정하는 것은 가장 쉽게 필요한 권한을 얻는 방법이며, 서비스 계정으로 프로젝트를 완전히 제어할 수 있음을 의미합니다. 해당 권한을 제공하는 데 따른 위험이 수용 가능한 수준인지 확인해봐야 합니다.

3. JSON 형식으로 서비스 계정 키를 생성합니다. GCP 문서의 서비스 계정 키 생성 단원을 참조하십시오.
   클러스터를 생성하기 위해서는 서비스 계정 키가 필요합니다.

7.2.5.1. 필요한 GCP 권한

생성하는 서비스 계정에 **Owner** 역할을 연결하면 OpenShift Container Platform 설치에 필요한 권한을 포함하여 모든 권한이 해당 서비스 계정에 부여됩니다. OpenShift Container Platform 클러스터를 배포하려면 서비스 계정에 다음과 같은 권한이 필요합니다. 기존 VPC에 클러스터를 배포할 때는 다음 목록에 제시된 특정 네트워킹 권한이 서비스 계정에 필요하지 않습니다.

설치 프로그램에 필요한 역할

- 컴퓨팅 관리자
- 보안 관리자
- 서비스 계정 관리자
- 서비스 계정 사용자
- 스토리지 관리자

설치 과정에서 네트워크 리소스를 생성하는 데 필요한 역할

- DNS 관리자

선택적 역할

운영자를 위한 제한적 자격 증명을 새로 생성하는 클러스터의 경우 다음 역할을 추가합니다.

- 서비스 계정 키 관리자

컨트롤 플레인 및 컴퓨팅 시스템이 사용하는 서비스 계정에 적용되는 역할입니다.

표 7.3. GCP 서비스 계정 권한

계정	역할
컨트롤 플레인	**roles/compute.instanceAdmin**
	roles/compute.networkAdmin
	roles/compute.securityAdmin
	roles/storage.admin
	roles/iam.serviceAccountUser
컴퓨팅	**roles/compute.viewer**
	roles/storage.admin

7.2.6. 지원되는 GCP 리전

다음과 같은 GCP(Google Cloud Platform) 리전에 OpenShift Container Platform 클러스터를 배포할 수 있습니다.

- **asia-east1** (대만 장화현)
- **asia-east2** (홍콩)
- **asia-northeast1** (일본 도쿄)
- **asia-northeast2** (일본 오사카)
- **asia-northeast3** (한국 서울)
- **asia-south1** (인도 뭄바이)
- **asia-southeast1** (싱가포르 주롱 웨스트)
- **asia-southeast2** (인도네시아 자카르타)
- **australia-southeast1** (호주 시드니)
- **europe-central2** (폴란드 바르샤바)
- **europe-north1** (핀란드 하미나)
- **europe-west1** (벨기에 생기슬랭)

- **europe-west2** (영국 런던)
- **europe-west3** (독일 프랑크푸르트)
- **europe-west4** (네덜란드 엠스하벤)
- **europe-west6** (스위스 취리히)
- **northamerica-northeast1** (캐나다 퀘벡 주 몬트리올)
- **southamerica-east1** (브라질 상파울루)
- **us-central1** (미국 아이오와 주 카운실 블러프스)
- **us-east1** (미국 사우스 캐롤라이나 주 몽크스 코너)
- **us-east4** (미국 노던 버지니아 주 애쉬번)
- **us-west1** (미국 오레곤 주 댈러스)
- **us-west2** (미국 캘리포니아 주 로스앤젤레스)
- **us-west3** (미국 유타 주 솔트레이크시티)
- **us-west4** (미국 네바다 라스베거스)

7.2.7. 다음 단계

- OpenShift Container Platform 클러스터를 GCP에 설치합니다. 사용자 지정 클러스터를 설치 하거나 기본 옵션을 적용하여 클러스터를 빠르게 설치 할 수 있습니다.

7.3. 수동으로 GCP 용 IAM 생성

클라우드 ID 및 액세스 관리(IAM) API에 연결할 수 없는 환경에서 또는 관리자가 클러스터 **kube-system** 네임 스페이스에 관리자 수준의 인증 정보 시크릿을 저장하지 않는 것을 선호하는 경우 클러스터를 설치하기 전에 CCO(Cloud Credential Operator)를 수동 모드로 설정할 수 있습니다.

7.3.1. kube-system 프로젝트에 관리자 수준 시크릿을 저장하는 대안

CCO(Cloud Credential Operator)는 클라우드 공급자 인증 정보를 Kubernetes CRD(사용자 지정 리소스 정의)로 관리합니다. **install-config.yaml** 파일에서 **credentialsMode** 매개변수의 다른 값을 설정하여 조직의 보안 요구 사항에 맞게 CCO를 구성할 수 있습니다.

클러스터 **kube-system** 프로젝트에 관리자 수준 인증 정보 시크릿을 저장하지 않으려면 OpenShift Container Platform을 설치할 때 다음 옵션 중 하나를 선택할 수 있습니다.

- **클라우드 인증 정보를 수동으로 관리**:
  CCO의 **credentialsMode** 매개변수를 수동으로 클라우드 인증 정보를 관리하도록 **Manual**로 설정할 수 있습니다. 수동 모드를 사용하면 각 클러스터 구성 요소에는 클러스터에 관리자 수준 인증 정보를 저장하지 않고 필요한 권한만 보유할 수 있습니다. 환경이 클라우드 공급자 공용 IAM 끝점에 연결되지 않은 경우 이 모드를 사용할 수도 있습니다. 그러나 업그레이드할 때마다 새 릴리스 이미지로 권한을 수동으로 조정해야 합니다. 또한 요청하는 모든 구성 요소에 대한 인증 정보를 수동으로 제공해야 합니다.

- **mint 모드를 사용하여 OpenShift Container Platform을 설치한 후 관리자 수준 인증 정보 시크릿 제거**:

credentialsMode 매개변수가 **Mint**로 설정된 CCO를 사용하는 경우 OpenShift Container Platform을 설치한 후 관리자 수준 인증 정보를 제거하거나 순환할 수 있습니다. Mint 모드는 CCO의 기본 구성입니다. 이 옵션을 사용하려면 설치 중에 관리자 수준 인증 정보가 있어야 합니다. 관리자 수준 인증 정보는 설치 중에 일부 권한이 부여된 다른 인증 정보를 Mint하는 데 사용됩니다. 원래 인증 정보 시크릿은 클러스터에 영구적으로 저장되지 않습니다.

참고

z-stream 외 업그레이드 이전에는 관리자 수준 인증 정보를 사용하여 인증 정보 시크릿을 복원해야 합니다. 인증 정보가 없으면 업그레이드가 차단될 수 있습니다.

추가 리소스

- 클라우드 공급자 인증 정보 교체 또는 제거 .

사용 가능한 모든 CCO 인증 정보 모드 및 지원되는 플랫폼에 대한 자세한 내용은 Cloud Credential Operator 정보를 참조하십시오.

7.3.2. 수동으로 IAM 생성

Cloud Credential Operator (CCO)는 클라우드 아이덴티티 및 액세스 관리 (IAM) API에 연결할 수 없는 환경에서 설치하기 전에 수동 모드로 전환할 수 있습니다. 또는 관리자가 클러스터 **kube-system** 네임 스페이스에 관리자 수준의 인증 정보 시크릿을 저장하지 않도록 합니다.

프로세스

1. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행하여 **install-config.yaml** 파일을 생성합니다.

```
$ openshift-install create install-config --dir <installation_directory>
```

여기서 **<installation_directory>** 는 설치 프로그램이 파일을 생성하는 디렉터리입니다.

2. **install-config.yaml** 구성 파일을 편집하여 **credentialsMode** 매개 변수가 **Manual**로 설정되도록 합니다.

install-config.yaml 설정 파일 예

```
apiVersion: v1
baseDomain: cluster1.example.com
credentialsMode: Manual (1)
compute:
- architecture: amd64
  hyperthreading: Enabled
...
```

(1) 이 행은 **credentialsMode** 매개변수를 **Manual**로 설정하기 위해 추가됩니다.

3. 매니페스트를 생성하려면 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행합니다.

```
$ openshift-install create manifests --dir <installation_directory>
```

여기서 **<installation_directory>** 는 설치 프로그램이 파일을 생성하는 디렉터리입니다.

4. 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행하여 **openshift-install** 바이너리가 사용하도록 빌드된 OpenShift Container Platform 릴리스 이미지에 대한 세부 정보를 가져옵니다.

```
$ openshift-install version
```

출력 예

```
release image quay.io/openshift-release-dev/ocp-release:4.y.z-x86_64
```

5. 다음 명령을 실행하여 배포 중인 클라우드를 대상으로 하는 이 릴리스 이미지에서 모든 **CredentialsRequest** 오브젝트를 찾습니다.

```
$ oc adm release extract quay.io/openshift-release-dev/ocp-release:4.y.z-x86_64 \
  --credentials-requests \
  --cloud=gcp
```

이 명령을 수행하면 각 **CredentialsRequest** 오브젝트에 대해 YAML 파일이 생성됩니다.

샘플 CredentialsRequest 개체

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component-credentials-request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: GCPProviderSpec
    predefinedRoles:
    - roles/storage.admin
    - roles/iam.serviceAccountUser
    skipServiceCheck: true
    ...
```

6. 이전에 생성한 **openshift-install** 매니페스트 디렉터리에 시크릿 YAML 파일을 만듭니다. 시크릿은 각 **CredentialsRequest** 오브젝트의 **spec.secretRef**에 정의된 네임 스페이스 및 시크릿 이름을 사용하여 저장해야 합니다.

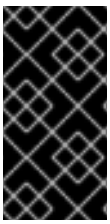
보안이 포함된 샘플 CredentialsRequest 오브젝트

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component-credentials-request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
      ...
  secretRef:
```

```
    name: <component-secret>
    namespace: <component-namespace>
  ...
```

샘플 Secret 오브젝트

```
apiVersion: v1
kind: Secret
metadata:
  name: <component-secret>
  namespace: <component-namespace>
data:
  service_account.json: <base64_encoded_gcp_service_account_file>
```



중요

수동으로 유지 관리되는 인증 정보를 사용하는 클러스터를 업그레이드하기 전에 CCO가 업그레이드 가능한 상태인지 확인해야 합니다. 자세한 내용은 클라우드 공급자에 대한 설치 콘텐츠의 "수동으로 유지 관리되는 인증 정보를 사용하여 클러스터 업그레이드" 섹션을 참조하십시오.

7.3.3. 수동으로 유지 관리되는 인증 정보로 클러스터 업그레이드

CCO(Cloud Credential Operator) 수동으로 유지 관리되는 인증 정보가 있는 클러스터의 **Upgradable** 상태는 기본적으로 **False** 입니다.

- 마이너 릴리스(예: 4.8에서 4.9로)의 경우 이 상태는 업데이트된 권한을 처리하고 **CloudCredential** 리소스에 주석을 달아 권한이 다음 버전에 필요에 따라 업데이트되었음을 나타낼 때까지 업그레이드되지 않도록 합니다. 이 주석은 **Upgradable** 상태를 **True**로 변경합니다.
- 예를 들어 4.9.0에서 4.9.1으로 z-stream 릴리스의 경우 권한이 추가되거나 변경되지 않으므로 업그레이드가 차단되지 않습니다.

수동으로 유지 관리되는 인증 정보로 클러스터를 업그레이드하기 전에 업그레이드할 릴리스 이미지에 대한 새 인증 정보를 생성해야 합니다. 또한 기존 인증 정보에 필요한 권한을 검토하고 해당 구성 요소에 대한 새 릴리스에 새 권한 요구 사항을 수용해야 합니다.

프로세스

1. 새 릴리스에 대한 **CredentialsRequest** 사용자 지정 리소스를 추출하고 검사합니다.
   클라우드 공급자용 설치 콘텐츠의 "수동으로 IAM 생성" 섹션에서는 클라우드에 필요한 인증 정보를 획득하고 사용하는 방법을 설명합니다.
2. 클러스터에서 수동으로 유지 관리되는 인증 정보를 업데이트합니다.
   - 새 릴리스 이미지에서 추가한 **CredentialsRequest** 사용자 정의 리소스에 대한 새 시크릿을 생성합니다.
   - 시크릿에 저장된 기존 인증 정보에 대한 **CredentialsRequest** 사용자 정의 리소스가 권한 요구 사항이 변경된 경우 필요에 따라 권한을 업데이트합니다.
3. 모든 보안이 새 릴리스에 대해 올바른 경우 클러스터를 업그레이드할 준비가 되었음을 나타냅니다.
   a. **cluster-admin** 역할의 사용자로 OpenShift Container Platform CLI에 로그인합니다.

b. **CloudCredential** 리소스를 편집하여 **metadata** 필드 내에 **upgradeable-to** 주석을 추가합니다.

```
$ oc edit cloudcredential cluster
```

추가할 텍스트

```
...
  metadata:
    annotations:
      cloudcredential.openshift.io/upgradeable-to: <version_number>
...
```

여기서 **<version_number>**는 **x.y.z** 형식으로 업그레이드할 버전입니다. 예를 들어 OpenShift Container Platform 4.8.2 의 경우 **4.8.2**입니다.

주석을 추가한 후 업그레이드 가능 상태가 변경되는 데 몇 분이 소요될 수 있습니다.

4. CCO를 업그레이드할 수 있는지 확인합니다.

a. 웹 콘솔의 **관리자** 화면에서 **관리자 → 클러스터 설정**으로 이동합니다.

b. CCO 상태 세부 정보를 보려면 **Cluster Operators** 목록에서 **cloud-credential**을 클릭합니다.

c. **Conditions** 섹션의 **Upgradeable** 상태가 **False**인 경우 **upgradeable-to 주석**에 오타 오류가 없는지 확인합니다.

Conditions 섹션의 **Upgradeable** 상태가 **True** 이면 OpenShift Container Platform 업그레이드를 시작할 수 있습니다.

```
:_content-type: CONCEPT
```

7.3.4. Mint 모드

Mint 모드는 OpenShift Container Platform의 기본 CCO(Cloud Credential Operator) 인증 정보 모드입니다. 이 모드에서 CCO는 제공된 관리자 수준 클라우드 인증 정보를 사용하여 클러스터를 실행합니다. Mint 모드는 AWS 및 GCP에서 지원됩니다.

mint 모드에서 **admin** 인증 정보가 **kube-system** 네임 스페이스에 저장된 다음 CCO에서 클러스터의 **CredentialsRequest** 오브젝트를 처리하고 각각에 대해 특정 권한이 있는 사용자를 만드는 데 사용됩니다.

mint 모드의 장점은 다음과 같습니다.

- 각 클러스터 구성 요소에는 필요한 권한만 있습니다.
- 추가 인증 정보 또는 권한을 포함한 클라우드 인증 정보에 대해 지속적인 자동 조정이 이루어집니다. 이는 업그레이드에 필요할 수 있습니다.

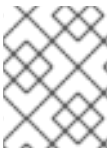
한 가지 단점은 mint 모드에 클러스터 **kube-system** 시크릿에 대한 **admin** 인증 정보 저장소가 필요하다는 것입니다.

7.3.5. 관리자 수준 인증 정보를 제거하거나 교체하는 Mint 모드

현재 이 모드는 AWS 및 GCP에서만 지원됩니다.

이 모드에서 사용자는 일반 Mint 모드와 마찬가지로 관리자 수준 인증 정보를 사용하여 OpenShift Container Platform을 설치합니다. 그러나 이 프로세스에서는 설치 후 클러스터에서 관리자 수준 인증 정보 시크릿을 제거합니다.

관리자는 모든 **CredentialsRequest** 오브젝트에 필요한 사용 권한이 있고 따라서 내용을 변경해야 하는 경우가 아니면 관리자 수준의 자격 증명이 필요 없음을 확인하기 위해 Cloud Credential Operator가 읽기 전용 자격 증명을 스스로 요청하도록 할 수 있습니다. 연결된 인증 정보를 제거한 후 원하는 경우 기본 클라우드에서 삭제하거나 비활성화할 수 있습니다.



참고

z-stream 외 업그레이드 이전에는 관리자 수준 인증 정보를 사용하여 인증 정보 시크릿을 복원해야 합니다. 인증 정보가 없으면 업그레이드가 차단될 수 있습니다.

관리자 수준 인증 정보는 클러스터에 영구적으로 저장되지 않습니다.

이러한 단계를 따르려면 짧은 기간 동안 클러스터의 관리자 수준 인증 정보가 필요합니다. 또한 각 업그레이드에 대해 관리자 수준 인증 정보로 시크릿을 수동으로 복원해야 합니다.

7.3.6. 다음 단계

- OpenShift Container Platform 클러스터 설치:
  - 설치 관리자가 프로비저닝한 인프라에 기본 옵션으로 GCP에 클러스터 빠른 설치
  - 설치 관리자가 프로비저닝한 인프라에 클라우드 사용자 지정으로 클러스터 설치
  - 설치 관리자가 프로비저닝한 인프라에 네트워크 사용자 지정으로 클러스터 설치

7.4. GCP에 클러스터 빠른 설치

OpenShift Container Platform 4.9 버전에서는 기본 구성 옵션을 사용하는 클러스터를 GCP(Google Cloud Platform)에 설치할 수 있습니다.

7.4.1. 사전 요구 사항

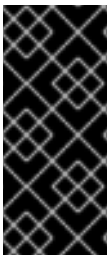
- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비 에 대한 문서를 읽습니다.
- 클러스터를 호스팅할 GCP 프로젝트를 구성했습니다.
- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성해야 합니다.
- 사용자의 환경에서 클라우드 ID 및 액세스 관리(IAM) API에 액세스할 수 없거나 **kube-system** 네임스페이스에 관리자 수준의 인증 정보 시크릿을 저장하지 않으려면 IAM 인증 정보 수동 생성 및 유지 관리를 참조하십시오.

7.4.2. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.



중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

7.4.3. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 **core** 사용자의 **~/.ssh/authorized_keys** 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 **core**로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. **./openshift-install gather** 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

참고

AWS 키 쌍 과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: **~/.ssh/id_ed25519**)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 **'~/.ssh** 디렉터리에 있는지 확인하십시오.

참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 **ed25519** 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 **rsa** 또는 **ecdsa** 알고리즘을 사용하는 키를 생성합니다.

2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 **~/.ssh/id_ed25519.pub** 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 **./openshift-install gather** 명령을 사용하려는 경우 필요합니다.

참고

일부 배포에서는 **~/.ssh/id_rsa** 및 **~/.ssh/id_dsa**와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. **ssh-agent** 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. **ssh-agent**에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: **~/.ssh/id_ed25519**).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

5. **GOOGLE_APPLICATION_CREDENTIALS** 환경 변수를 서비스 계정 개인 키 파일의 전체 경로로 설정합니다.

-

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

6. 인증 정보가 적용되었는지 확인합니다.

```
$ gcloud auth list
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

7.4.4. 설치 프로그램 받기

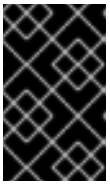
OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

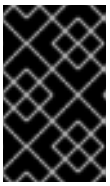
프로세스

1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.
2. 인프라 공급자를 선택합니다.
3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.



중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.



중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 OpenShift Container Platform 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

7.4.5. 클러스터 배포

호환되는 클라우드 플랫폼에 OpenShift Container Platform을 설치할 수 있습니다.

중요

최초 설치 과정에서 설치 프로그램의 **create cluster** 명령을 한 번만 실행할 수 있습니다.

사전 요구 사항

- 클러스터를 호스팅하는 클라우드 플랫폼으로 계정을 구성합니다.
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

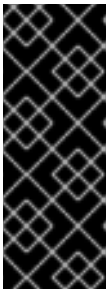
프로세스

1. 클러스터에 맞춰 구성하여 다음 위치에 저장한 GCP 계정의 서비스 계정 키를 사용하지 않는 기존 GCP 사용자 자격 증명을 삭제합니다.
   - **GOOGLE_CREDENTIALS**, **GOOGLE_CLOUD_KEYFILE_JSON** 또는 **GCLOUD_KEYFILE_JSON** 환경 변수
   - **~/.gcp/osServiceAccount.json** 파일
   - **gcloud cli** 기본 자격 증명
2. 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터 배포를 초기화합니다.

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>**는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.

2 다른 설치 세부 사항을 보려면 **info** 대신 **warn**, **debug** 또는 **error**를 지정합니다.

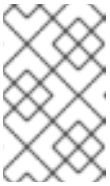


중요

비어 있는 디렉터리를 지정합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

화면에 나타나는 지시에 따라 필요한 값을 입력합니다.

a. 선택사항: 클러스터 시스템에 액세스하는 데 사용할 SSH 키를 선택합니다.



참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

b. 대상 플랫폼으로 **gcp**를 선택합니다.

c. 사용자 컴퓨터에 GCP 계정의 서비스 계정 키가 구성되어 있지 않은 경우, GCP에서 해당 키를 가져와 파일의 내용을 붙여 넣거나 파일의 전체 경로를 입력해야 합니다.

d. 클러스터를 프로비저닝할 프로젝트 ID를 선택합니다. 구성한 서비스 계정에 의해 기본값이 지정됩니다.

e. 클러스터를 배포할 리전을 선택합니다.

f. 클러스터를 배포할 기본 도메인을 선택합니다. 기본 도메인은 클러스터용으로 생성한 퍼블릭 DNS 영역에 해당합니다.

g. 클러스터를 설명할 수 있는 이름을 입력합니다. 이름이 6자를 초과하면 클러스터 이름에서 생성되는 인프라 ID에 이름의 처음 6자만 사용됩니다.

h. Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 붙여넣습니다.

참고

호스트에 구성된 클라우드 공급자 계정에 클러스터를 배포하기에 충분한 권한이 없는 경우, 설치 프로세스가 중단되고 누락된 권한을 알리는 메시지가 표시됩니다.

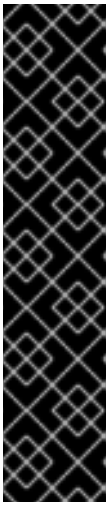
클러스터 배포가 완료되면 웹 콘솔로 연결되는 링크와 **kubeadmin** 사용자의 인증 정보가 포함된 클러스터 액세스 지침이 사용자 터미널에 표시됩니다.

출력 예

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```

참고

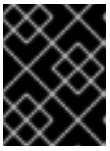
설치에 성공하면 클러스터 액세스 및 인증 정보도 **<installation_directory>/.openshift_install.log**로 출력됩니다.



중요

- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.

- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.



중요

설치 프로그램에서 생성되는 파일이나 설치 프로그램을 삭제해서는 안 됩니다. 클러스터를 삭제하려면 두 가지가 모두 필요합니다.

3. 선택사항: 클러스터를 설치하는 데 사용한 서비스 계정에 대한 권한 수를 줄일 수 있습니다.
   - 서비스 계정에 **Owner** 역할을 할당한 경우, 해당 역할을 제거하고 **Viewer** 역할로 바꿀 수 있습니다.
   - **Service Account Key Admin** 역할이 포함되어 있으면 제거할 수도 있습니다.

7.4.6. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift CLI(**oc**)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 **oc**를 설치할 수 있습니다.

중요

이전 버전의 **oc**를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 **oc**를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 Linux에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Linux Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. **oc** 바이너리를 **PATH**에 있는 디렉터리에 배치합니다.
   PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치
다음 절차에 따라 Windows에 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Windows Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. ZIP 프로그램으로 아카이브의 압축을 풉니다.
5. **oc** 바이너리를 **PATH**에 있는 디렉터리로 이동합니다.
   PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치
다음 절차에 따라 macOS에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 MacOSX Client** 항목 옆에 있는 **지금 다운로드를** 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 해제하고 압축을 풉니다.
5. **oc** 바이너리 PATH의 디렉터리로 이동합니다.
   PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

7.4.7. CLI를 사용하여 클러스터에 로그인

클러스터 **kubeconfig** 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. **kubeconfig** 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- **oc** CLI를 설치했습니다.

프로세스

1. **kubeadmin** 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 **oc** 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

추가 리소스

- OpenShift Container Platform 웹 콘솔 액세스 및 이해에 대한 자세한 내용은 웹 콘솔에 액세스를 참조하십시오.

7.4.8. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

7.4.9. 다음 단계

- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.

7.5. 사용자 지정 설정의 GCP에 클러스터 설치

OpenShift Container Platform 4.9 버전에서는 설치 프로그램이 GCP(Google Cloud Platform)에 프로비저닝하는 인프라에 사용자 지정 클러스터를 설치할 수 있습니다. 설치를 사용자 지정하려면 클러스터를 설치하기 전에 **install-config.yaml** 파일에서 매개변수를 수정합니다.

7.5.1. 사전 요구 사항

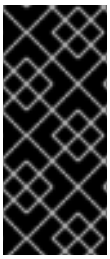
- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비 에 대한 문서를 읽습니다.
- 클러스터를 호스팅할 GCP 프로젝트를 구성했습니다.
- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성해야 합니다.
- 사용자의 환경에서 클라우드 ID 및 액세스 관리(IAM) API에 액세스할 수 없거나 **kube-system** 네임스페이스에 관리자 수준의 인증 정보 시크릿을 저장하지 않으려면 IAM 인증 정보 수동 생성 및 유지 관리를 참조하십시오.

7.5.2. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.



중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

7.5.3. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 **core** 사용자의 **~/.ssh/authorized_keys** 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 **core**로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. **./openshift-install gather** 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

참고

AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: **~/.ssh/id_ed25519**)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 **'~/.ssh** 디렉터리에 있는지 확인하십시오.

참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 **ed25519** 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 **rsa** 또는 **ecdsa** 알고리즘을 사용하는 키를 생성합니다.

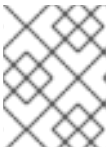
2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 **~/.ssh/id_ed25519.pub** 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 **./openshift-install gather** 명령을 사용하려는 경우 필요합니다.



참고

일부 배포에서는 **~/.ssh/id_rsa** 및 **~/.ssh/id_dsa**와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. **ssh-agent** 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. **ssh-agent**에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: **~/.ssh/id_ed25519**).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

5. **GOOGLE_APPLICATION_CREDENTIALS** 환경 변수를 서비스 계정 개인 키 파일의 전체 경로로 설정합니다.

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

6. 인증 정보가 적용되었는지 확인합니다.

```
$ gcloud auth list
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

7.5.4. 설치 프로그램 받기

OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

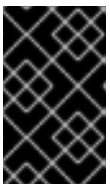
- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

프로세스

1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.
2. 인프라 공급자를 선택합니다.
3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.

중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.



중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 OpenShift Container Platform 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

7.5.5. 설치 구성 파일 만들기

GCP(Google Cloud Platform)에 설치하는 OpenShift Container Platform 클러스터를 사용자 지정할 수 있습니다.

사전 요구 사항

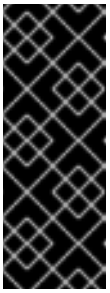
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.
- 서브스크립션 수준에서 서비스 주체 권한을 획득합니다.

프로세스

1. **install-config.yaml** 파일을 생성합니다.
   a. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 **<installation_directory>**는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.



중요

비어 있는 디렉터리를 지정합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

b. 화면에 나타나는 지시에 따라 클라우드에 대한 구성 세부 사항을 입력합니다.

i. 선택사항: 클러스터 시스템에 액세스하는 데 사용할 SSH 키를 선택합니다.

참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

ii. 대상 플랫폼으로 **gcp**를 선택합니다.

iii. 사용자 컴퓨터에 GCP 계정의 서비스 계정 키가 구성되어 있지 않은 경우, GCP에서 해당 키를 가져와 파일의 내용을 붙여 넣거나 파일의 전체 경로를 입력해야 합니다.

iv. 클러스터를 프로비저닝할 프로젝트 ID를 선택합니다. 구성한 서비스 계정에 의해 기본값이 지정됩니다.

v. 클러스터를 배포할 리전을 선택합니다.

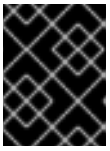
vi. 클러스터를 배포할 기본 도메인을 선택합니다. 기본 도메인은 클러스터용으로 생성한 퍼블릭 DNS 영역에 해당합니다.

vii. 클러스터를 설명할 수 있는 이름을 입력합니다.

viii. Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 붙여넣습니다.

2. **install-config.yaml** 파일을 수정합니다. 사용 가능한 매개변수에 대한 자세한 정보는 "설치 구성 매개변수" 섹션에서 확인할 수 있습니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.



중요

install-config.yaml 파일은 설치 과정에서 사용됩니다. 이 파일을 재사용하려면 지금 백업해야 합니다.

7.5.5.1. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 매개변수 값을 제공하여 클러스터를 호스팅할 클라우드 플랫폼에서 사용자 계정을 설명하고 선택사항으로 클러스터의 플랫폼을 사용자 지정합니다. **install-config.yaml** 설치 구성 파일을 생성할 때 명령줄을 통해 필요한 매개변수 값을 제공합니다. 클러스터를 사용자 지정하면 **install-config.yaml** 파일을 수정하여 플랫폼에 대한 세부 정보를 제공할 수 있습니다.

참고

설치한 후에는 **install-config.yaml** 파일에서 이러한 매개변수를 수정할 수 없습니다.

중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

7.5.5.1.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 7.4. 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	소문자, 하이픈(**-**), 마침표(**.**)로 구성되는 문자열(예: **dev**)입니다.
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

매개변수	설명	값

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

7.5.5.1.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 7.5. 네트워크 매개변수

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체 **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.

매개변수	설명	값
networking.clusterNetwork	Pod의 IP 주소 블록입니다. 기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된 **cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16**  **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

7.5.5.1.3. 선택적 구성 매개변수

선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 7.6. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은**3**입니다.

매개변수	설명	값
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다. GCP에 공유 VPC(가상 프라이빗 클라우드)를 설치하는 경우 **credentialsMode** 를 **Passthrough** 로 설정해야 합니다. **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오. **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 인터넷에서 액세스할 수 없는 사설 클러스터를 배포하려면 **publish**를 **Internal**로 설정합니다. 기본값은 **External**입니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

7.5.5.1.4. 추가 GCP(Google Cloud Platform) 구성 매개변수

추가 GCP 구성 매개변수는 다음 표에 설명되어 있습니다.

표 7.7. 추가 GCP 매개변수

매개변수	설명	값
platform.gcp.network	클러스터를 배포할 기존 VPC의 이름입니다.	문자열.
platform.gcp.region	클러스터를 호스팅하는 GCP 리전의 이름입니다.	유효한 리전 이름 (예: **us-central1**)
platform.gcp.type	GCP 시스템 유형	GCP 시스템 유형입니다.
platform.gcp.zones	설치 프로그램에서 지정된 MachinePool에 대한 시스템을 생성하는 가용 영역입니다.	**us-central1-a**와 같이 유효한 GCP 가용 영역 목록에 YAML 순서로 나열됩니다.
platform.gcp.controlPlaneSubnet	컨트롤 플레인 시스템을 배포할 VPC의 기존 서브넷 이름	서브넷 이름입니다.

매개변수	설명	값
platform.gcp.computeSubnet	컴퓨팅 시스템을 배포할 VPC의 기존 서브넷 이름	서브넷 이름입니다.
platform.gcp.licenses	컴퓨팅 이미지에 적용해야 하는 라이센스 URL 목록입니다. **중요** **licenses** 매개변수는 더 이상 사용되지 않는 필드이며 중첩된 가상화는 기본적으로 활성화되어 있습니다. 이 필드를 사용하지 않는 것이 좋습니다.	중첩된 가상화 를 활성화하는 라이센스와 같은 라이센스 API 와 함께 사용할 수 있는 모든 라이센스입니다. 사전 빌드된 이미지를 생성하는 메커니즘과 함께 이 매개변수를 사용할 수 없습니다. 라이센스 URL을 사용하면 설치 프로그램이 소스 이미지를 사용하기 전에 복사해야 합니다.
platform.gcp.osDisk.diskSizeGB	디스크 크기(GB)입니다.	16GB와 65536GB 사이의 모든 크기입니다.
platform.gcp.osDisk.diskType	디스크 유형입니다.	기본 **pd-ssd** 또는 **pd-standard** 디스크 유형입니다. 컨트롤 플레인 노드는 **pd-ssd** 디스크 유형이어야 합니다. 작업자 노드는 유형 중 하나일 수 있습니다.
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.name	컨트롤 플레인 머신 디스크 암호화에 사용되는 고객 관리 암호화 키의 이름입니다.	암호화 키 이름입니다.

매개변수	설명	값
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.keyRing	컨트롤 플레인 머신의 경우 KMS 키가 속한 KMS 키 링의 이름입니다.	KMS 키 링의 이름입니다.
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.location	컨트롤 플레인 머신의 경우 키 링이 있는 GCP 위치입니다. KMS 위치에 대한 자세한 내용은 Cloud KMS 위치에 대한 Google의 문서를 참조하십시오.	키 링의 GCP 위치입니다.
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.projectID	컨트롤 플레인 머신의 경우 KMS 키 링이 있는 프로젝트의 ID입니다. 이 값은 설정되지 않은 경우 VM 프로젝트 ID가 기본값입니다.	GCP 프로젝트 ID입니다.
compute.platform.gcp.osDisk.encryptionKey.kmsKey.name	컴퓨팅 머신 디스크 암호화에 사용되는 고객 관리 암호화 키의 이름입니다.	암호화 키 이름입니다.

매개변수	설명	값
compute.platform.gcp.osDisk.encryptionKey.kmsKey.keyRing	컴퓨팅 머신의 경우 KMS 키가 속한 KMS 키 링의 이름입니다.	KMS 키 링의 이름입니다.
compute.platform.gcp.osDisk.encryptionKey.kmsKey.location	컴퓨팅 머신의 경우 키 링이 있는 GCP 위치입니다. KMS 위치에 대한 자세한 내용은 Cloud KMS 위치에 대한 Google의 문서를 참조하십시오.	키 링의 GCP 위치입니다.
compute.platform.gcp.osDisk.encryptionKey.kmsKey.projectID	컴퓨팅 머신의 경우 KMS 키 링이 있는 프로젝트의 ID입니다. 이 값은 설정되지 않은 경우 VM 프로젝트 ID가 기본값입니다.	GCP 프로젝트 ID입니다.

7.5.5.2. 클러스터 설치를 위한 최소 리소스 요구 사항

각 클러스터 시스템이 다음과 같은 최소 요구사항을 충족해야 합니다.

표 7.8. 최소 리소스 요구사항

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
부트스트랩	RHCOS	4	16GB	100GB	300
컨트롤 플레인	RHCOS	4	16GB	100GB	300

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
컴퓨팅	RHCOS, RHEL 7.9 또는 RHEL 8.4 [3]	2	8GB	100GB	300

1. SMT(동시 멀티스레딩) 또는 하이퍼 스레딩이 활성화되지 않은 경우 하나의 vCPU는 하나의 물리적 코어와 동일합니다. 활성화하면 다음과 같은 공식을 사용하여 해당 비율을 계산합니다. (코어당 스레드 수 × 코어 수) × 소켓 수 = vCPU 수

2. OpenShift Container Platform 및 Kubernetes는 디스크 성능에 민감하며 특히 10ms p99 fsync 기간이 필요한 컨트롤 플레인 노드의 etcd에 더 빠른 스토리지가 권장됩니다. 많은 클라우드 플랫폼에서 스토리지 크기와 IOPS를 함께 확장되므로 충분한 성능을 얻으려면 스토리지 볼륨을 과도하게 할당해야 할 수 있습니다.

3. 사용자가 프로비저닝한 모든 설치와 마찬가지로 클러스터에서 RHEL 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. RHEL 7 컴퓨팅 머신 사용은 더 이상 사용되지 않으며 향후 OpenShift Container Platform 4 릴리스에서 제거될 예정입니다.

7.5.5.3. 사용자 정의 머신 유형 사용

OpenShift Container Platform 클러스터를 설치하기 위해 사용자 정의 머신 유형을 사용할 수 있습니다.

사용자 지정 머신 유형을 사용할 때는 다음을 고려하십시오.

- 사전 정의된 인스턴스 유형과 유사하게 사용자 정의 머신 유형은 컨트롤 플레인 및 컴퓨팅 시스템의 최소 리소스 요구 사항을 충족해야 합니다. 자세한 내용은 "클러스터 설치를 위한 최소 리소스 요구 사항"을 참조하십시오.

- 사용자 정의 머신 유형의 이름은 다음 구문을 준수해야 합니다.
  custom-<number_of_cpus>-<amount_of_memory_in_mb>

  예를 들면 **custom-6-20480** 입니다.

설치 프로세스의 일부로 **install-config.yaml** 파일에 사용자 지정 머신 유형을 지정합니다.

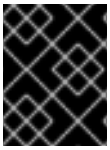
사용자 지정 머신 유형이 있는 샘플 install-config.yaml 파일

```
compute:
- architecture: amd64
  hyperthreading: Enabled
  name: worker
  platform:
    gcp:
      type: custom-6-20480
  replicas: 2
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  platform:
```

```
    gcp:
      type: custom-6-20480
  replicas: 3
```

7.5.5.4. GCP용 샘플 사용자 지정 install-config.yaml 파일

install-config.yaml 파일을 사용자 지정하여 OpenShift Container Platform 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.



중요

이 샘플 YAML 파일은 참조용으로만 제공됩니다. 설치 프로그램을 사용하여 **install-config.yaml** 파일을 받아서 수정해야 합니다.

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2) (3)
  hyperthreading: Enabled (4)
  name: master
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-ssd
        diskSizeGB: 1024
        encryptionKey: (5)
          kmsKey:
            name: worker-key
            keyRing: test-machine-keys
            location: global
            projectID: project-id
  replicas: 3
compute: (6) (7)
- hyperthreading: Enabled (8)
  name: worker
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-standard
        diskSizeGB: 128
        encryptionKey: (9)
          kmsKey:
            name: worker-key
            keyRing: test-machine-keys
            location: global
            projectID: project-id
```

```
  replicas: 3
metadata:
  name: test-cluster 10
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  gcp:
    projectID: openshift-production 11
    region: us-central1 12
pullSecret: '{"auths": ...}' 13
fips: false 14
sshKey: ssh-ed25519 AAAA... 15
```

1 10 11 12 13 필수 항목입니다. 설치 프로그램에서 이 값을 입력하라는 메시지를 표시합니다.

2 6 이러한 매개변수와 값을 지정하지 않으면 설치 프로그램은 기본값을 적용합니다.

3 7 **controlPlane** 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

4 8 동시 멀티스레딩 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. 일부 클러스터 시스템에서 동시 멀티스레딩을 비활성화할 경우에는 해당 멀티스레딩을 모든 클러스터 시스템에서 비활성화해야 합니다.

중요

동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다. 동시 멀티스레딩을 비활성화하는 경우, 사용자 시스템에 더 큰 시스템 유형(예: **n1-standard-8**)을 사용하십시오.

5 9 선택 사항: 사용자 정의 암호화 키 섹션으로 가상 머신과 영구 볼륨을 모두 암호화합니다. 기본 컴퓨팅 서비스 계정에는 KMS 키를 사용하도록 부여된 권한이 있어야 하며 올바른 IAM 역할을 할당해야 합니다. 기본 서비스 계정 이름은 **service-<project_number>@compute-system.iam.gserviceaccount.com** 패턴을 따릅니다. 서비스 계정에 대한 올바른 권한을 부여하는 방법에 대한 자세한 내용은 "머신 관리" → " 머신 세트 생성" → “GCP에서 머신 세트 생성”을 참조하십시오.

14 FIPS 모드 활성화 또는 비활성화 여부입니다. 기본적으로 FIPS 모드는 비활성화됩니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.

중요

FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.

15 선택사항으로, 클러스터의 시스템에 액세스하는 데 사용할 **sshKey** 값을 제공할 수도 있습니다.

참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

추가 리소스

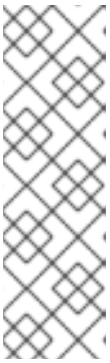
- 머신 세트의 고객 관리 암호화 키 활성화

7.5.5.5. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 HTTP 또는 HTTPS 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 OpenShift Container Platform 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 API에 대한 호출을 포함하여 모든 클러스터 발신(Egress) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.



참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 Red Hat OpenStack Platform (RHOSP)에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

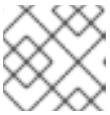
절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
```

```
additionalTrustBundle: | 4
   -----BEGIN CERTIFICATE-----
   <MY_TRUSTED_CA_CERT>
   -----END CERTIFICATE-----
...
```

1. 클러스터 외부에서 HTTP 연결을 구축하는 데 사용할 프록시 URL입니다. URL 스키마는 **http**여야 합니다.
2. 클러스터 외부에서 HTTPS 연결을 구축하는 데 사용할 프록시 URL입니다.
3. 대상 도메인 이름, IP 주소 또는 프록시에서 제외할 기타 네트워크 CIDR로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. *를 사용하여 모든 대상에 대해 프록시를 바이패스합니다.
4. 이 값을 제공하면 설치 프로그램에서 추가 CA 인증서를 보유할 **openshift -config 네임스페이스에 user-ca- bundle** 이라는 구성 맵을 생성합니다. **additionalTrustBundle** 및 하나 이상의 프록시 설정을 제공하는 경우 **프록시** 오브젝트는 **trustedCA** 필드의 **user-ca-bundle** 구성 맵을 참조하도록 구성됩니다. 그러면 Cluster Network Operator에서 **trustedCA** 매개변수에 대해 지정된 콘텐츠를 RHCOS 신뢰 번들과 병합하는 **trusted-ca-bundle** 구성 맵을 생성합니다. 프록시의 ID 인증서를 RHCOS 트러스트 번들에 있는 기관에서 서명하지 않은 경우 **additionalTrustBundle** 필드가 있어야 합니다.



참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 OpenShift Container Platform을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 nil이 됩니다.

참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

7.5.6. GCP Marketplace 이미지 사용

GCP Marketplace 이미지를 사용하여 OpenShift Container Platform 클러스터를 배포하려면 매니페스트를 생성하고 컴퓨팅 머신 세트 정의를 편집하여 GCP Marketplace 이미지를 지정해야 합니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿이 있습니다.

절차

1. 다음 명령을 실행하여 설치 매니페스트를 생성합니다.

```
$ openshift-install create manifests --dir <installation_dir>
```

2. 다음 파일을 찾습니다.
   - **<installation_dir>/openshift/99_openshift-cluster-api_worker-machineset-0.yaml**
   - **<installation_dir>/openshift/99_openshift-cluster-api_worker-machineset-1.yaml**
   - **<installation_dir>/openshift/99_openshift-cluster-api_worker-machineset-2.yaml**
3. 각 파일에서 사용할 제안을 참조하도록 **.spec.template.spec.providerSpec.value.disks[0].image** 속성을 편집합니다.

   OpenShift Container Platform

   projects/redhat-marketplace-public/global/images/redhat-coreos-ocp-48-x86-64-202210040145

   OpenShift Platform Plus

   projects/redhat-marketplace-public/global/images/redhat-coreos-opp-48-x86-64-202206140145

   OpenShift Kubernetes Engine

   projects/redhat-marketplace-public/global/images/redhat-coreos-oke-48-x86-64-202206140145

GCP Marketplace 이미지가 있는 컴퓨팅 머신 세트의 예

```
deletionProtection: false
disks:
- autoDelete: true
  boot: true
  image: projects/redhat-marketplace-public/global/images/redhat-coreos-ocp-48-x86-64-
202210040145
  labels: null
  sizeGb: 128
  type: pd-ssd
kind: GCPMachineProviderSpec
machineType: n2-standard-4
```

7.5.7. 클러스터 배포

호환되는 클라우드 플랫폼에 OpenShift Container Platform을 설치할 수 있습니다.

중요

최초 설치 과정에서 설치 프로그램의 **create cluster** 명령을 한 번만 실행할 수 있습니다.

사전 요구 사항

- 클러스터를 호스팅하는 클라우드 플랫폼으로 계정을 구성합니다.
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

절차

1. 클러스터에 맞춰 구성하여 다음 위치에 저장한 GCP 계정의 서비스 계정 키를 사용하지 않는 기존 GCP 사용자 자격 증명을 삭제합니다.

- **GOOGLE_CREDENTIALS**, **GOOGLE_CLOUD_KEYFILE_JSON** 또는 **GCLOUD_KEYFILE_JSON** 환경 변수
- **~/.gcp/osServiceAccount.json** 파일
- **gcloud cli** 기본 자격 증명

2. 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터 배포를 초기화합니다.

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** 값으로 사용자 지정한 **./install-config.yaml** 파일의 위치를 지정합니다.

2 다른 설치 세부 사항을 보려면 **info** 대신 **warn**, **debug** 또는 **error**를 지정합니다.

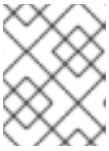
참고

호스트에 구성된 클라우드 공급자 계정에 클러스터를 배포하기에 충분한 권한이 없는 경우, 설치 프로세스가 중단되고 누락된 권한을 알리는 메시지가 표시됩니다.

클러스터 배포가 완료되면 웹 콘솔로 연결되는 링크와 **kubeadmin** 사용자의 인증 정보가 포함된 클러스터 액세스 지침이 사용자 터미널에 표시됩니다.

출력 예

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```



참고

설치에 성공하면 클러스터 액세스 및 인증 정보도 **<installation_directory>/.openshift_install.log**로 출력됩니다.

중요

- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.

- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

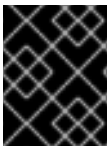
중요

설치 프로그램에서 생성되는 파일이나 설치 프로그램을 삭제해서는 안 됩니다. 클러스터를 삭제하려면 두 가지가 모두 필요합니다.

3. 선택사항: 클러스터를 설치하는 데 사용한 서비스 계정에 대한 권한 수를 줄일 수 있습니다.
   - 서비스 계정에 **Owner** 역할을 할당한 경우, 해당 역할을 제거하고 **Viewer** 역할로 바꿀 수 있습니다.
   - **Service Account Key Admin** 역할이 포함되어 있으면 제거할 수도 있습니다.

7.5.8. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift CLI(**oc**)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 **oc**를 설치할 수 있습니다.



중요

이전 버전의 **oc**를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 **oc**를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 Linux에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Linux Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. **oc** 바이너리를 **PATH**에 있는 디렉터리에 배치합니다.
   PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치
다음 절차에 따라 Windows에 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Windows Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. ZIP 프로그램으로 아카이브의 압축을 풉니다.
5. **oc** 바이너리를 **PATH**에 있는 디렉터리로 이동합니다.
   PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치
다음 절차에 따라 macOS에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 MacOSX Client** 항목 옆에 있는 **지금 다운로드를** 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 해제하고 압축을 풉니다.
5. **oc** 바이너리 PATH의 디렉터리로 이동합니다.
   PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

7.5.9. CLI를 사용하여 클러스터에 로그인

클러스터 **kubeconfig** 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. **kubeconfig** 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- **oc** CLI를 설치했습니다.

프로세스

1. **kubeadmin** 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 **oc** 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

추가 리소스

- OpenShift Container Platform 웹 콘솔 액세스 및 이해에 대한 자세한 내용은 웹 콘솔에 액세스를 참조하십시오.

7.5.10. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

7.5.11. 다음 단계

- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.

7.6. 네트워크 사용자 지정 설정으로 GCP에 클러스터 설치

OpenShift Container Platform 4.9 버전에서는 설치 프로그램에서 GCP(Google Cloud Platform)에 프로비저닝하는 인프라에 사용자 지정 네트워크 구성으로 클러스터를 설치할 수 있습니다. 네트워크 구성을 사용자 지정할 경우, 클러스터가 사용자 환경의 기존 IP 주소 할당과 공존하고 기존 MTU 및 VXLAN 구성과 통합될 수 있습니다. 설치를 사용자 지정하려면 클러스터를 설치하기 전에 **install-config.yaml** 파일에서 매개변수를 수정합니다.

설치 과정에서 네트워크 구성 매개변수를 대부분 설정해야 하며, 실행 중인 클러스터에서는 **kubeProxy** 구성 매개변수만 수정할 수 있습니다.

7.6.1. 사전 요구 사항

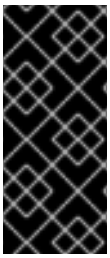
- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비 에 대한 문서를 읽습니다.
- 클러스터를 호스팅할 GCP 프로젝트를 구성했습니다.
- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성해야 합니다.
- 사용자의 환경에서 클라우드 ID 및 액세스 관리(IAM) API에 액세스할 수 없거나 **kube-system** 네임스페이스에 관리자 수준의 인증 정보 시크릿을 저장하지 않으려면 IAM 인증 정보 수동 생성 및 유지 관리를 참조하십시오.

7.6.2. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.



중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

7.6.3. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 **core** 사용자의 **~/.ssh/authorized_keys** 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 **core**로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다. SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. **./openshift-install gather** 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

참고

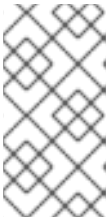
AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: **~/.ssh/id_ed25519**)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 **'~/.ssh** 디렉터리에 있는지 확인하십시오.



참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 **ed25519** 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 **rsa** 또는 **ecdsa** 알고리즘을 사용하는 키를 생성합니다.

2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 **~/.ssh/id_ed25519.pub** 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 **./openshift-install gather** 명령을 사용하려는 경우 필요합니다.

참고

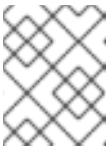
일부 배포에서는 **~/.ssh/id_rsa** 및 **~/.ssh/id_dsa**와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. **ssh-agent** 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```



참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. **ssh-agent**에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: **~/.ssh/id_ed25519**).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

5. **GOOGLE_APPLICATION_CREDENTIALS** 환경 변수를 서비스 계정 개인 키 파일의 전체 경로로 설정합니다.

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

6. 인증 정보가 적용되었는지 확인합니다.

```
$ gcloud auth list
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

7.6.4. 설치 프로그램 받기

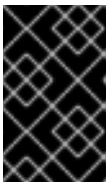
OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

프로세스

1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.
2. 인프라 공급자를 선택합니다.
3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.



중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.

중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 OpenShift Container Platform 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

7.6.5. 설치 구성 파일 만들기

GCP(Google Cloud Platform)에 설치하는 OpenShift Container Platform 클러스터를 사용자 지정할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.
- 서브스크립션 수준에서 서비스 주체 권한을 획득합니다.

프로세스

1. **install-config.yaml** 파일을 생성합니다.
   a. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 **<installation_directory>**는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.

중요

비어 있는 디렉터리를 지정합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

b. 화면에 나타나는 지시에 따라 클라우드에 대한 구성 세부 사항을 입력합니다.

i. 선택사항: 클러스터 시스템에 액세스하는 데 사용할 SSH 키를 선택합니다.

참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

ii. 대상 플랫폼으로 **gcp**를 선택합니다.

iii. 사용자 컴퓨터에 GCP 계정의 서비스 계정 키가 구성되어 있지 않은 경우, GCP에서 해당 키를 가져와 파일의 내용을 붙여 넣거나 파일의 전체 경로를 입력해야 합니다.

iv. 클러스터를 프로비저닝할 프로젝트 ID를 선택합니다. 구성한 서비스 계정에 의해 기본값이 지정됩니다.

v. 클러스터를 배포할 리전을 선택합니다.

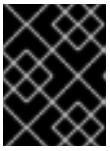
vi. 클러스터를 배포할 기본 도메인을 선택합니다. 기본 도메인은 클러스터용으로 생성한 퍼블릭 DNS 영역에 해당합니다.

vii. 클러스터를 설명할 수 있는 이름을 입력합니다.

viii. Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 붙여넣습니다.

2. **install-config.yaml** 파일을 수정합니다. 사용 가능한 매개변수에 대한 자세한 정보는 "설치 구성 매개변수" 섹션에서 확인할 수 있습니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.



중요

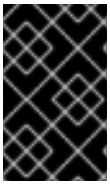
install-config.yaml 파일은 설치 과정에서 사용됩니다. 이 파일을 재사용하려면 지금 백업해야 합니다.

7.6.5.1. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 매개변수 값을 제공하여 클러스터를 호스팅할 클라우드 플랫폼에서 사용자 계정을 설명하고 선택사항으로 클러스터의 플랫폼을 사용자 지정합니다. **install-config.yaml** 설치 구성 파일을 생성할 때 명령줄을 통해 필요한 매개변수 값을 제공합니다. 클러스터를 사용자 지정하면 **install-config.yaml** 파일을 수정하여 플랫폼에 대한 세부 정보를 제공할 수 있습니다.

참고

설치한 후에는 **install-config.yaml** 파일에서 이러한 매개변수를 수정할 수 없습니다.



중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

7.6.5.1.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 7.9. 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	소문자, 하이픈(**-**), 마침표(**.**)로 구성되는 문자열(예: **dev**)입니다.
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

매개변수	설명	값

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

7.6.5.1.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 7.10. 네트워크 매개변수

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체 **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.

매개변수	설명	값
networking.clusterNetwork	Pod의 IP 주소 블록입니다. 기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된 **cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16**  **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

7.6.5.1.3. 선택적 구성 매개변수

선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 7.11. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은**3**입니다.

매개변수	설명	값
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다. GCP에 공유 VPC(가상 프라이빗 클라우드)를 설치하는 경우 **credentialsMode** 를 **Passthrough** 로 설정해야 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 인터넷에서 액세스할 수 없는 사설 클러스터를 배포하려면 **publish**를 **Internal**로 설정합니다. 기본값은 **External**입니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

7.6.5.1.4. 추가 GCP(Google Cloud Platform) 구성 매개변수

추가 GCP 구성 매개변수는 다음 표에 설명되어 있습니다.

표 7.12. 추가 GCP 매개변수

매개변수	설명	값
platform.gcp.network	클러스터를 배포할 기존 VPC의 이름입니다.	문자열.
platform.gcp.region	클러스터를 호스팅하는 GCP 리전의 이름입니다.	유효한 리전 이름 (예: **us-central1**)
platform.gcp.type	GCP 시스템 유형	GCP 시스템 유형입니다.
platform.gcp.zones	설치 프로그램에서 지정된 MachinePool에 대한 시스템을 생성하는 가용 영역입니다.	**us-central1-a**와 같이 유효한 GCP 가용 영역 목록에 YAML 순서로 나열됩니다.
platform.gcp.controlPlaneSubnet	컨트롤 플레인 시스템을 배포할 VPC의 기존 서브넷 이름	서브넷 이름입니다.

매개변수	설명	값
platform.gcp.computeSubnet	컴퓨팅 시스템을 배포할 VPC의 기존 서브넷 이름	서브넷 이름입니다.
platform.gcp.licenses	컴퓨팅 이미지에 적용해야 하는 라이센스 URL 목록입니다. **중요** **licenses** 매개변수는 더 이상 사용되지 않는 필드이며 중첩된 가상화는 기본적으로 활성화되어 있습니다. 이 필드를 사용하지 않는 것이 좋습니다.	중첩된 가상화 를 활성화하는 라이센스와 같은 라이센스 API 와 함께 사용할 수 있는 모든 라이센스입니다. 사전 빌드된 이미지를 생성하는 메커니즘과 함께 이 매개변수를 사용할 수 없습니다. 라이센스 URL을 사용하면 설치 프로그램이 소스 이미지를 사용하기 전에 복사해야 합니다.
platform.gcp.osDisk.diskSizeGB	디스크 크기(GB)입니다.	16GB와 65536GB 사이의 모든 크기입니다.
platform.gcp.osDisk.diskType	디스크 유형입니다.	기본 **pd-ssd** 또는 **pd-standard** 디스크 유형입니다. 컨트롤 플레인 노드는 **pd-ssd** 디스크 유형이어야 합니다. 작업자 노드는 유형 중 하나일 수 있습니다.
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.name	컨트롤 플레인 머신 디스크 암호화에 사용되는 고객 관리 암호화 키의 이름입니다.	암호화 키 이름입니다.

매개변수	설명	값
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.keyRing	컨트롤 플레인 머신의 경우 KMS 키가 속한 KMS 키 링의 이름입니다.	KMS 키 링의 이름입니다.
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.location	컨트롤 플레인 머신의 경우 키 링이 있는 GCP 위치입니다. KMS 위치에 대한 자세한 내용은 Cloud KMS 위치에 대한 Google의 문서를 참조하십시오.	키 링의 GCP 위치입니다.
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.projectID	컨트롤 플레인 머신의 경우 KMS 키 링이 있는 프로젝트의 ID입니다. 이 값은 설정되지 않은 경우 VM 프로젝트 ID가 기본값입니다.	GCP 프로젝트 ID입니다.
compute.platform.gcp.osDisk.encryptionKey.kmsKey.name	컴퓨팅 머신 디스크 암호화에 사용되는 고객 관리 암호화 키의 이름입니다.	암호화 키 이름입니다.

매개변수	설명	값
compute.platform.gcp.osDisk.encryptionKey.kmsKey.keyRing	컴퓨팅 머신의 경우 KMS 키가 속한 KMS 키 링의 이름입니다.	KMS 키 링의 이름입니다.
compute.platform.gcp.osDisk.encryptionKey.kmsKey.location	컴퓨팅 머신의 경우 키 링이 있는 GCP 위치입니다. KMS 위치에 대한 자세한 내용은 Cloud KMS 위치에 대한 Google의 문서를 참조하십시오.	키 링의 GCP 위치입니다.
compute.platform.gcp.osDisk.encryptionKey.kmsKey.projectID	컴퓨팅 머신의 경우 KMS 키 링이 있는 프로젝트의 ID입니다. 이 값은 설정되지 않은 경우 VM 프로젝트 ID가 기본값입니다.	GCP 프로젝트 ID입니다.

7.6.5.2. 클러스터 설치를 위한 최소 리소스 요구 사항

각 클러스터 시스템이 다음과 같은 최소 요구사항을 충족해야 합니다.

표 7.13. 최소 리소스 요구사항

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
부트스트랩	RHCOS	4	16GB	100GB	300
컨트롤 플레인	RHCOS	4	16GB	100GB	300

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
컴퓨팅	RHCOS, RHEL 7.9 또는 RHEL 8.4 [3]	2	8GB	100GB	300

1. SMT(동시 멀티스레딩) 또는 하이퍼 스레딩이 활성화되지 않은 경우 하나의 vCPU는 하나의 물리적 코어와 동일합니다. 활성화하면 다음과 같은 공식을 사용하여 해당 비율을 계산합니다. (코어당 스레드 수 × 코어 수) × 소켓 수 = vCPU 수

2. OpenShift Container Platform 및 Kubernetes는 디스크 성능에 민감하며 특히 10ms p99 fsync 기간이 필요한 컨트롤 플레인 노드의 etcd에 더 빠른 스토리지가 권장됩니다. 많은 클라우드 플랫폼에서 스토리지 크기와 IOPS를 함께 확장되므로 충분한 성능을 얻으려면 스토리지 볼륨을 과도하게 할당해야 할 수 있습니다.

3. 사용자가 프로비저닝한 모든 설치와 마찬가지로 클러스터에서 RHEL 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. RHEL 7 컴퓨팅 머신 사용은 더 이상 사용되지 않으며 향후 OpenShift Container Platform 4 릴리스에서 제거될 예정입니다.

7.6.5.3. 사용자 정의 머신 유형 사용

OpenShift Container Platform 클러스터를 설치하기 위해 사용자 정의 머신 유형을 사용할 수 있습니다.

사용자 지정 머신 유형을 사용할 때는 다음을 고려하십시오.

- 사전 정의된 인스턴스 유형과 유사하게 사용자 정의 머신 유형은 컨트롤 플레인 및 컴퓨팅 시스템의 최소 리소스 요구 사항을 충족해야 합니다. 자세한 내용은 "클러스터 설치를 위한 최소 리소스 요구 사항"을 참조하십시오.

- 사용자 정의 머신 유형의 이름은 다음 구문을 준수해야 합니다.
  custom-<number_of_cpus>-<amount_of_memory_in_mb>

  예를 들면 **custom-6-20480** 입니다.

설치 프로세스의 일부로 **install-config.yaml** 파일에 사용자 지정 머신 유형을 지정합니다.

사용자 지정 머신 유형이 있는 샘플 install-config.yaml 파일

```
compute:
- architecture: amd64
  hyperthreading: Enabled
  name: worker
  platform:
    gcp:
      type: custom-6-20480
  replicas: 2
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  platform:
```

```
    gcp:
      type: custom-6-20480
  replicas: 3
```

7.6.5.4. GCP용 샘플 사용자 지정 install-config.yaml 파일

install-config.yaml 파일을 사용자 지정하여 OpenShift Container Platform 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.

중요

이 샘플 YAML 파일은 참조용으로만 제공됩니다. 설치 프로그램을 사용하여 **install-config.yaml** 파일을 받아서 수정해야 합니다.

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2) (3)
  hyperthreading: Enabled (4)
  name: master
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-ssd
        diskSizeGB: 1024
        encryptionKey: (5)
          kmsKey:
            name: worker-key
            keyRing: test-machine-keys
            location: global
            projectID: project-id
  replicas: 3
compute: (6) (7)
- hyperthreading: Enabled (8)
  name: worker
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-standard
        diskSizeGB: 128
        encryptionKey: (9)
          kmsKey:
            name: worker-key
            keyRing: test-machine-keys
            location: global
            projectID: project-id
```

```
  replicas: 3
metadata:
  name: test-cluster 10
networking: 11
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  gcp:
    projectID: openshift-production 12
    region: us-central1 13
pullSecret: '{"auths": ...}' 14
fips: false 15
sshKey: ssh-ed25519 AAAA... 16
```

1 10 12 13 14 필수 항목입니다. 설치 프로그램에서 이 값을 입력하라는 메시지를 표시합니다.

2 6 11 이러한 매개변수와 값을 지정하지 않으면 설치 프로그램은 기본값을 적용합니다.

3 7 **controlPlane** 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

4 8 동시 멀티스레딩 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. 일부 클러스터 시스템에서 동시 멀티스레딩을 비활성화할 경우에는 해당 멀티스레딩을 모든 클러스터 시스템에서 비활성화해야 합니다.

중요

동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다. 동시 멀티스레딩을 비활성화하는 경우, 사용자 시스템에 더 큰 시스템 유형(예: **n1-standard-8**)을 사용하십시오.

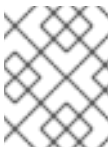
5 9 선택 사항: 사용자 정의 암호화 키 섹션으로 가상 머신과 영구 볼륨을 모두 암호화합니다. 기본 컴퓨팅 서비스 계정에는 KMS 키를 사용하도록 부여된 권한이 있어야 하며 올바른 IAM 역할을 할당해야 합니다. 기본 서비스 계정 이름은 **service-<project_number>@compute-system.iam.gserviceaccount.com** 패턴을 따릅니다. 서비스 계정에 대한 올바른 권한을 부여하는 방법에 대한 자세한 내용은 "머신 관리" → " 머신 세트 생성" → "GCP에서 머신 세트 생성"을 참조하십시오.

15 FIPS 모드 활성화 또는 비활성화 여부입니다. 기본적으로 FIPS 모드는 비활성화됩니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.

중요

FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.

16 선택사항으로, 클러스터의 시스템에 액세스하는 데 사용할 **sshKey** 값을 제공할 수도 있습니다.



참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

7.6.6. 추가 리소스

- 머신 세트의 고객 관리 암호화 키 활성화

7.6.6.1. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 HTTP 또는 HTTPS 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 OpenShift Container Platform 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 API에 대한 호출을 포함하여 모든 클러스터 발신(Egress) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.

참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 Red Hat OpenStack Platform (RHOSP)에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
```

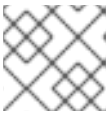
```
additionalTrustBundle: | 4
  -----BEGIN CERTIFICATE-----
  <MY_TRUSTED_CA_CERT>
  -----END CERTIFICATE-----
...
```

1 클러스터 외부에서 HTTP 연결을 구축하는 데 사용할 프록시 URL입니다. URL 스키마는 **http**여야 합니다.

2 클러스터 외부에서 HTTPS 연결을 구축하는 데 사용할 프록시 URL입니다.

3 대상 도메인 이름, IP 주소 또는 프록시에서 제외할 기타 네트워크 CIDR로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. *를 사용하여 모든 대상에 대해 프록시를 바이패스합니다.

4 이 값을 제공하면 설치 프로그램에서 추가 CA 인증서를 보유할 **openshift -config 네임스페이스에 user-ca- bundle** 이라는 구성 맵을 생성합니다. **additionalTrustBundle** 및 하나 이상의 프록시 설정을 제공하는 경우 **프록시** 오브젝트는 **trustedCA** 필드의 **user-ca-bundle** 구성 맵을 참조하도록 구성됩니다. 그러면 Cluster Network Operator에서 **trustedCA** 매개 변수에 대해 지정된 콘텐츠를 RHCOS 신뢰 번들과 병합하는 **trusted-ca-bundle** 구성 맵을 생성합니다. 프록시의 ID 인증서를 RHCOS 트러스트 번들에 있는 기관에서 서명하지 않은 경우 **additionalTrustBundle** 필드가 있어야 합니다.



참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 OpenShift Container Platform을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 nil이 됩니다.

참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

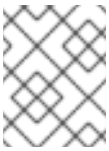
7.6.7. 네트워크 구성 단계

OpenShift Container Platform을 설치하기 전에 네트워크 구성을 사용자 지정할 수 있습니다.

1단계

매니페스트 파일을 생성하기 전에 **install-config.yaml** 파일에서 다음 네트워크 관련 필드를 사용자 지정할 수 있습니다.

- **networking.networkType**
- **networking.clusterNetwork**
- **networking.serviceNetwork**
- **networking.machineNetwork**
  이러한 필드에 대한 자세한 내용은 *설치 구성 매개 변수*에서 참조하십시오.



참고

기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

2 단계

openshift-install create manifests 를 실행하여 매니페스트 파일을 생성한 후 수정할 필드로 사용자 지정된 Cluster Network Operator 매니페스트를 정의할 수 있습니다. 매니페스트를 사용하여 고급 네트워크 구성을 지정할 수 있습니다.

install-config.yaml 파일에서 1 단계에 지정된 값을 2 단계에서 재정의할 수 없습니다. 그러나 2 단계에서 클러스터 네트워크 공급자를 추가로 사용자 지정할 수 있습니다.

7.6.8. 고급 네트워크 구성 지정

클러스터 네트워크 제공자의 고급 네트워크 구성을 사용하여 클러스터를 기존 네트워크 환경에 통합할 수 있습니다. 클러스터를 설치하기 전에만 고급 네트워크 구성을 지정할 수 있습니다.

중요

설치 프로그램에서 생성한 OpenShift Container Platform 매니페스트 파일을 수정하여 네트워크 구성을 사용자 정의하는 것은 지원되지 않습니다. 다음 절차에서와 같이 생성한 매니페스트 파일을 적용할 수 있습니다.

사전 요구 사항

- **install-config.yaml** 파일을 생성하고 수정 작업을 완료했습니다.

절차

1. 설치 프로그램이 포함된 디렉터리로 변경하고 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>**는 클러스터의 **install-config.yaml** 파일이 포함된 디렉터리의 이름을 지정합니다.

2. **<installation_directory>/ manifests/** 디렉토리에 **cluster-network-03-config.yml**이라는 stub 매니페스트 파일을 만듭니다.

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
```

3. 다음과 같이 **cluster-network-03-config.yml** 파일에서 클러스터의 고급 네트워크 구성을 지정합니다.

OpenShift SDN 네트워크 공급자에 대한 다른 VXLAN 포트 지정

```
apiVersion: operator.openshift.io/v1
```

```
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    openshiftSDNConfig:
      vxlanPort: 4800
```

OVN-Kubernetes 네트워크 공급자의 IPsec 활성화

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    ovnKubernetesConfig:
      ipsecConfig: {}
```

4. 선택사항: **manifests/cluster-network-03-config.yml** 파일을 백업합니다. 설치 프로그램은 Ignition 구성 파일을 생성할 때 **manifests/** 디렉터리를 사용합니다.

7.6.9. CNO(Cluster Network Operator) 구성

클러스터 네트워크의 구성은 CNO(Cluster Network Operator) 구성의 일부로 지정되며 **cluster**라는 이름의 CR(사용자 정의 리소스) 오브젝트에 저장됩니다. CR은 **operator.openshift.io** API 그룹에서 **Network** API의 필드를 지정합니다.

CNO 구성은 **Network.config.openshift.io** API 그룹의 **Network** API에서 클러스터 설치 중에 다음 필드를 상속하며 이러한 필드는 변경할 수 없습니다.

clusterNetwork

Pod IP 주소가 할당되는 IP 주소 풀입니다.

serviceNetwork

서비스를 위한 IP 주소 풀입니다.

defaultNetwork.type

OpenShift SDN 또는 OVN-Kubernetes와 같은 클러스터 네트워크 공급자입니다.

cluster라는 CNO 오브젝트에서 **defaultNetwork** 오브젝트의 필드를 설정하여 클러스터의 클러스터 네트워크 공급자 구성을 지정할 수 있습니다.

7.6.9.1. CNO(Cluster Network Operator) 구성 오브젝트

CNO(Cluster Network Operator)의 필드는 다음 표에 설명되어 있습니다.

표 7.14. CNO(Cluster Network Operator) 구성 오브젝트

필드	유형	설명
metadata.name	**string**	CNO 개체 이름입니다. 이 이름은 항상 **cluster**입니다.

필드	유형	설명
spec.clusterNetwork	**array**	Pod IP 주소가 할당되는 IP 주소 블록과 클러스터의 각 개별 노드에 할당된 서브넷 접두사 길이를 지정하는 목록입니다. 예를 들면 다음과 같습니다. spec: clusterNetwork: - cidr: 10.128.0.0/19 hostPrefix: 23 - cidr: 10.128.32.0/19 hostPrefix: 23 매니페스트를 생성하기 전에 **install-config.yaml** 파일에서만 이 필드를 사용자 지정할 수 있습니다. 값은 매니페스트 파일에서 읽기 전용입니다.
spec.serviceNetwork	**array**	서비스의 IP 주소 블록입니다. OpenShift SDN 및 OVN-Kubernetes CNI(Container Network Interface) 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다. 예를 들면 다음과 같습니다. spec: serviceNetwork: - 172.30.0.0/14 매니페스트를 생성하기 전에 **install-config.yaml** 파일에서만 이 필드를 사용자 지정할 수 있습니다. 값은 매니페스트 파일에서 읽기 전용입니다.
spec.defaultNetwork	**object**	클러스터 네트워크의 CNI(Container Network Interface) 클러스터 네트워크 공급자를 구성합니다.
spec.kubeProxyConfig	**object**	이 개체의 필드는 kube-proxy 구성을 지정합니다. OVN-Kubernetes 클러스터 네트워크 공급자를 사용하는 경우 kube-proxy 구성이 적용되지 않습니다.

defaultNetwork 오브젝트 구성

defaultNetwork 오브젝트의 값은 다음 표에 정의되어 있습니다.

표 7.15. **defaultNetwork** 오브젝트

필드	유형	설명

필드	유형	설명
type	**string**	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, 클러스터 네트워크 공급자가 설치 중에 선택됩니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다. **참고** OpenShift Container Platform은 기본적으로 OpenShift SDN CNI(Container Network Interface) 클러스터 네트워크 공급자를 사용합니다.
openshiftSDNConfig	**object**	이 오브젝트는 OpenShift SDN 클러스터 네트워크 공급자에만 유효합니다.
ovnKubernetesConfig	**object**	이 오브젝트는 OVN-Kubernetes 클러스터 네트워크 공급자에만 유효합니다.

OpenShift SDN CNI 네트워크 공급자에 대한 구성
다음 표에서는 OpenShift SDN Container Network Interface (CNI) 클러스터 네트워크 공급자의 구성 필드를 설명합니다.

표 7.16. openshiftSDNConfig 오브젝트

필드	유형	설명
mode	**string**	OpenShift SDN의 네트워크 격리 모드를 구성합니다. 기본값은 **NetworkPolicy**입니다. **Multitenant** 및 **Subnet** 값은 OpenShift Container Platform 3.x 버전과의 역호환할 수 있지만 권장되지는 않습니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
mtu	**integer**	VXLAN 오버레이 네트워크의 최대 전송 단위(MTU)입니다. 이는 기본 네트워크 인터페이스의 MTU를 기준으로 자동 탐지됩니다. 일반적으로 감지된 MTU를 재정의할 필요는 없습니다. 자동 감지 값이 예상 밖인 경우 노드의 기본 네트워크 인터페이스의 MTU가 올바른지 확인합니다. 이 옵션을 사용하여 노드의 기본 네트워크 인터페이스의 MTU 값을 변경할 수 없습니다. 클러스터에 다른 노드에 대한 다른 MTU 값이 필요한 경우 이 값을 클러스터의 가장 낮은 MTU 값보다 **50** 미만으로 설정해야 합니다. 예를 들어 클러스터의 일부 노드에는 **9001**의 MTU가 있고 일부에는 **1500**의 MTU가 있는 경우 이 값을 **1450**으로 설정해야 합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.

필드	유형	설명
vxlanPort	**integer**	모든 VXLAN 패킷에 사용할 포트입니다. 기본값은 **4789**입니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다. 기존 노드가 다른 VXLAN 네트워크에 속하는 가상 환경에서 실행 중인 경우에는 기본값을 변경해야 할 수도 있습니다. 예를 들어 VMware NSX-T 위에서 OpenShift SDN 오버레이를 실행할 때 두 SDN이 동일한 기본 VXLAN 포트 번호를 사용하므로 VXLAN의 대체 포트를 선택해야 합니다. AWS(Amazon Web Services)에서는 포트 **9000**과 포트 **9999** 사이에서 VXLAN의 대체 포트를 선택할 수 있습니다.

OpenShift SDN 구성 예

```
defaultNetwork:
  type: OpenShiftSDN
  openshiftSDNConfig:
    mode: NetworkPolicy
    mtu: 1450
    vxlanPort: 4789
```

OVN-Kubernetes CNI 클러스터 네트워크 공급자에 대한 구성

다음 표에서는 OVN-Kubernetes CNI 클러스터 네트워크 공급자의 구성 필드를 설명합니다.

표 7.17. ovnKubernetesConfig object

필드	유형	설명
mtu	**integer**	Geneve(Generic Network Virtualization Encapsulation) 오버레이 네트워크의 MTU(최대 전송 단위)입니다. 이는 기본 네트워크 인터페이스의 MTU를 기준으로 자동 탐지됩니다. 일반적으로 감지된 MTU를 재정의할 필요는 없습니다. 자동 감지 값이 예상 밖인 경우 노드의 기본 네트워크 인터페이스의 MTU가 올바른지 확인합니다. 이 옵션을 사용하여 노드의 기본 네트워크 인터페이스의 MTU 값을 변경할 수 없습니다. 클러스터에 다른 노드에 대한 다른 MTU 값이 필요한 경우, 이 값을 클러스터의 가장 낮은 MTU 값보다 **100** 미만으로 설정해야 합니다. 예를 들어, 클러스터의 일부 노드에 **9001**의 MTU가 있고 일부에는 **1500**의 MTU가 있는 경우 이 값을**1400**으로 설정해야 합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
genevePort	**integer**	모든 Geneve 패킷에 사용할 포트입니다. 기본값은 **6081**입니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.

필드	유형	설명
ipsecConfig	**object**	IPsec 암호화를 활성화하려면 빈 오브젝트를 지정합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
policyAuditConfig	**object**	네트워크 정책 감사 로깅을 사용자 정의할 구성 오브젝트를 지정합니다. 설정되지 않으면 기본값 감사 로그 설정이 사용됩니다.

표 7.18. **policyAuditConfig** object

필드	유형	설명
rateLimit	integer	노드당 1초마다 생성할 최대 메시지 수입니다. 기본값은 초당 **20** 개의 메시지입니다.
maxFileSize	integer	감사 로그의 최대 크기(바이트)입니다. 기본값은 **50000000** 또는 **50** MB입니다.
대상	string	다음 추가 감사 로그 대상 중 하나입니다. **libc** 호스트에서 journald 프로세스의 libc **syslog()** 함수입니다. **udp:<host>:<port>** syslog 서버입니다. **<host>:<port>**를 syslog 서버의 호스트 및 포트로 바꿉니다. **unix:<file>** **<file>**로 지정된 Unix Domain Socket 파일입니다. **null** 감사 로그를 추가 대상으로 보내지 마십시오.
syslogFacility	string	RFC5424에 정의된 **kern**과 같은 syslog 기능입니다. 기본값은 **local0**입니다.

OVN-Kubernetes 구성 예

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
    ipsecConfig: {}
```

kubeProxyConfig 오브젝트 구성

kubeProxyConfig 오브젝트의 값은 다음 표에 정의되어 있습니다.

표 7.19. **kubeProxyConfig** object

필드	유형	설명
iptablesSyncPeriod	**string**	**iptables** 규칙의 새로 고침 간격입니다. 기본값은 **30s**입니다. 유효 접미사로 **s**, **m**, **h**가 있으며, 자세한 설명은 Go **time** 패키지 문서를 참조하십시오.  **참고** OpenShift Container Platform 4.3 이상에서는 성능이 개선되어 더 이상 **iptablesSyncPeriod** 매개변수를 조정할 필요가 없습니다.

proxyArguments.iptables-min-sync-period	**array**	**iptables** 규칙을 새로 고치기 전 최소 기간입니다. 이 필드를 통해 새로 고침 간격이 너무 짧지 않도록 조정할 수 있습니다. 유효 접미사로 **s**, **m**, **h**가 있으며, 자세한 설명은 Go **time** 패키지를 참조하십시오. 기본값은 다음과 같습니다.
		
		`kubeProxyConfig:`
		`  proxyArguments:`
		`  iptables-min-sync-period:`
		`  - 0s`

7.6.10. 클러스터 배포

호환되는 클라우드 플랫폼에 OpenShift Container Platform을 설치할 수 있습니다.

중요

최초 설치 과정에서 설치 프로그램의 **create cluster** 명령을 한 번만 실행할 수 있습니다.

사전 요구 사항

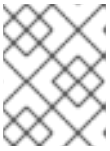
- 클러스터를 호스팅하는 클라우드 플랫폼으로 계정을 구성합니다.
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

프로세스

1. 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터 배포를 초기화합니다.

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1. **<installation_directory>**는 다음을 지정합니다.
2. 다른 설치 세부 사항을 보려면 **info** 대신 **warn**, **debug** 또는 **error**를 지정합니다.



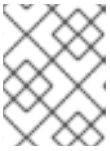
참고

호스트에 구성된 클라우드 공급자 계정에 클러스터를 배포하기에 충분한 권한이 없는 경우, 설치 프로세스가 중단되고 누락된 권한을 알리는 메시지가 표시됩니다.

클러스터 배포가 완료되면 웹 콘솔로 연결되는 링크와 **kubeadmin** 사용자의 인증 정보가 포함된 클러스터 액세스 지침이 사용자 터미널에 표시됩니다.

출력 예

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```

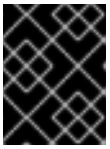


참고

설치에 성공하면 클러스터 액세스 및 인증 정보도 **<installation_directory>/.openshift_install.log**로 출력됩니다.

중요

- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.



중요

설치 프로그램에서 생성되는 파일이나 설치 프로그램을 삭제해서는 안 됩니다. 클러스터를 삭제하려면 두 가지가 모두 필요합니다.

7.6.11. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift CLI(**oc**)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 **oc**를 설치할 수 있습니다.

중요

이전 버전의 **oc**를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 **oc**를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치
다음 절차를 사용하여 Linux에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Linux Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. **oc** 바이너리를 **PATH**에 있는 디렉터리에 배치합니다.
PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치
다음 절차에 따라 Windows에 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Windows Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. ZIP 프로그램으로 아카이브의 압축을 풉니다.
5. **oc** 바이너리를 **PATH**에 있는 디렉터리로 이동합니다.
PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치
다음 절차에 따라 macOS에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 MacOSX Client** 항목 옆에 있는 **지금 다운로드를** 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 해제하고 압축을 풉니다.
5. **oc** 바이너리 PATH의 디렉터리로 이동합니다.
   PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

7.6.12. CLI를 사용하여 클러스터에 로그인

클러스터 **kubeconfig** 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. **kubeconfig** 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- **oc** CLI를 설치했습니다.

프로세스

1. **kubeadmin** 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 **oc** 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

추가 리소스

- OpenShift Container Platform 웹 콘솔 액세스 및 이해에 대한 자세한 내용은 웹 콘솔에 액세스를 참조하십시오.

7.6.13. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

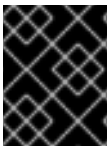
- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보 를 참조하십시오.

7.6.14. 다음 단계

- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃 을 수행할 수 있습니다.

7.7. 제한된 네트워크에서 GCP에 클러스터 설치

OpenShift Container Platform 4.9에서는 기존 Google VPC(Virtual Private Cloud)에 설치 릴리스 콘텐츠의 내부 미러를 생성하여 제한된 네트워크의 Red Hat OpenStack Platform (RHOSP)에 클러스터를 설치할 수 있습니다.



중요

미러링된 설치 릴리스 콘텐츠를 사용하여 OpenShift Container Platform 클러스터를 설치할 수는 있지만 GCP API를 사용하려면 클러스터에 인터넷 액세스가 필요합니다.

7.7.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비 에 대한 문서를 읽습니다.
- 레지스트리에 연결이 끊긴 설치 이미지를 미러링 하고 사용 중인 OpenShift Container Platform 버전의 **imageContentSources** 데이터를 가져옵니다.

  중요

  미러 호스트에 설치 미디어가 있으므로 해당 컴퓨터를 사용하여 모든 설치 단계를 완료하십시오.

- GCP에 기존 VPC가 있습니다. 설치 관리자 프로비저닝 인프라를 사용하여 제한된 네트워크에 클러스터를 설치할 때 설치 관리자 프로비저닝 VPC를 사용할 수 없습니다. 다음 요구사항 중 하나를 충족하는 사용자 프로비저닝 VPC를 사용해야 합니다.
  - 미러 레지스트리 정보가 있습니다.
  - 방화벽 규칙 또는 피어링 연결이 다른 위치에서 호스팅되는 미러 레지스트리에 액세스할 수 있습니다.

- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성 해야 합니다. 액세스 권한을 부여해야 할 사이트가 더 있을지도 모르지만 ***.googleapis.com**과 **accounts.google.com**에 액세스 권한은 반드시 부여해야 합니다.

- 사용자의 환경에서 클라우드 ID 및 액세스 관리(IAM) API에 액세스할 수 없거나 **kube-system** 네임스페이스에 관리자 수준의 인증 정보 시크릿을 저장하지 않으려면 IAM 인증 정보 수동 생성 및 유지 관리를 참조하십시오.

7.7.2. 네트워크가 제한된 환경에서의 설치 정보

OpenShift Container Platform 4.9에서는 소프트웨어 구성 요소를 받기 위한 인터넷 접속이 필요하지 않은 설치를 수행할 수 있습니다. 제한된 네트워크 설치는 클러스터를 설치하는 클라우드 플랫폼에 따라 설치 관리자 프로비저닝 인프라 또는 사용자 프로비저닝 인프라를 사용하여 완료할 수 있습니다.

클라우드 플랫폼에 제한된 네트워크 설치를 수행하는 방법을 선택해도 클라우드 API에 액세스는 가능해야 합니다. Amazon Web Service의 Route 53 DNS 및 IAM 서비스와 같은 일부 클라우드 기능에는 인터넷 액세스가 필요합니다. 사용 중인 네트워크에 따라 베어메탈 하드웨어 또는 VMware vSphere에 설치하기 위해 필요한 인터넷 액세스가 줄어들 수 있습니다.

제한된 네트워크 설치를 완료하려면 OpenShift Container Platform 레지스트리의 내용을 미러링하고 설치 미디어를 포함할 레지스트리를 생성해야 합니다. 인터넷과 폐쇄 네트워크에 모두 액세스하거나 제한 사항을 따르는 다른 방법을 통해 미러 호스트에 레지스트리를 생성할 수 있습니다.

7.7.2.1. 추가 제한

제한된 네트워크의 클러스터에는 다음과 같은 추가 제한이 있습니다.

- **ClusterVersion** 상태에 **사용 가능한 업데이트를 검색할 수 없음** 오류가 포함되어 있습니다.

- 기본적으로 필요한 이미지 스트림 태그에 액세스할 수 없기 때문에 개발자 카탈로그의 내용을 사용할 수 없습니다.

7.7.3. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하기 위해 필요한 이미지를 받으려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.

- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.

- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.

중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

7.7.4. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

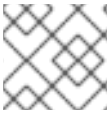
OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 **core** 사용자의 **~/.ssh/authorized_keys** 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 **core**로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다. SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. **./openshift-install gather** 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.



참고

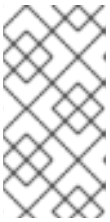
AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: **~/.ssh/id_ed25519**)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 **'~/.ssh** 디렉터리에 있는지 확인하십시오.



참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 **ed25519** 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 **rsa** 또는 **ecdsa** 알고리즘을 사용하는 키를 생성합니다.

2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 **~/.ssh/id_ed25519.pub** 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 **./openshift-install gather** 명령을 사용하려는 경우 필요합니다.

참고

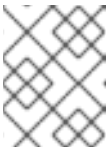
일부 배포에서는 **~/.ssh/id_rsa** 및 **~/.ssh/id_dsa**와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. **ssh-agent** 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```



참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. **ssh-agent**에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: **~/.ssh/id_ed25519**).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

5. **GOOGLE_APPLICATION_CREDENTIALS** 환경 변수를 서비스 계정 개인 키 파일의 전체 경로로 설정합니다.

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

6. 인증 정보가 적용되었는지 확인합니다.

```
$ gcloud auth list
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

7.7.5. 설치 구성 파일 만들기

GCP(Google Cloud Platform)에 설치하는 OpenShift Container Platform 클러스터를 사용자 지정할 수 있습니다.

사전 요구 사항

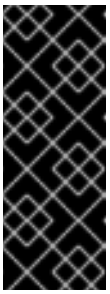
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다. 제한된 네트워크 설치의 경우, 해당 파일은 미러 호스트에 있습니다.
- 미러 레지스트리 작성 중에 생성된 **imageContentSources** 값이 있어야 합니다.
- 미러 레지스트리에 대한 인증서의 내용을 가져옵니다.
- 서브스크립션 수준에서 서비스 주체 권한을 획득합니다.

프로세스

1. **install-config.yaml** 파일을 생성합니다.
   a. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

   1 **<installation_directory>**는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.



중요

비어 있는 디렉터리를 지정합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

   b. 화면에 나타나는 지시에 따라 클라우드에 대한 구성 세부 사항을 입력합니다.
      i. 선택사항: 클러스터 시스템에 액세스하는 데 사용할 SSH 키를 선택합니다.

참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

      ii. 대상 플랫폼으로 **gcp**를 선택합니다.
      iii. 사용자 컴퓨터에 GCP 계정의 서비스 계정 키가 구성되어 있지 않은 경우, GCP에서 해당 키를 가져와 파일의 내용을 붙여 넣거나 파일의 전체 경로를 입력해야 합니다.
      iv. 클러스터를 프로비저닝할 프로젝트 ID를 선택합니다. 구성한 서비스 계정에 의해 기본값이 지정됩니다.
      v. 클러스터를 배포할 리전을 선택합니다.
      vi. 클러스터를 배포할 기본 도메인을 선택합니다. 기본 도메인은 클러스터용으로 생성한 퍼블릭 DNS 영역에 해당합니다.
      vii. 클러스터를 설명할 수 있는 이름을 입력합니다.

viii. Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 붙여넣습니다.

2. 제한된 네트워크에서의 설치에 필요한 추가 정보를 제공하려면 **install-config.yaml** 파일을 편집합니다.

   a. 레지스트리의 인증 정보를 포함하도록 **pullSecret** 값을 업데이트합니다.

   ```
   pullSecret: '{"auths":{"<mirror_host_name>:5000": {"auth": "<credentials>","email":
   "you@example.com"}}}'
   ```

   <mirror_host_name>의 경우 미러 레지스트리의 인증서에 지정한 레지스트리 도메인 이름을 지정하고 **<credentials>**의 경우 미러 레지스트리에 base64로 인코딩된 사용자 이름 및 암호를 지정합니다.

   b. **additionalTrustBundle** 매개변수와 값을 추가합니다.

   ```
   additionalTrustBundle: |
     -----BEGIN CERTIFICATE-----

   ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
     -----END CERTIFICATE-----
   ```

   값은 미러 레지스트리에 사용한 인증서 파일의 내용이어야 하며, 신뢰할 수 있는 기존 인증 기관 또는 미러 레지스트리에 대해 생성한 자체 서명 인증서일 수 있습니다.

   c. 상위 **platform.gcp** 필드 아래에 클러스터를 설치할 VPC의 네트워크 및 서브넷을 정의합니다.

   ```
   network: <existing_vpc>
   controlPlaneSubnet: <control_plane_subnet>
   computeSubnet: <compute_subnet>
   ```

   platform.gcp.network의 경우 기존 Google VPC의 이름을 지정합니다.
   platform.gcp.controlPlaneSubnet 및 **platform.gcp.computeSubnet**의 경우 각각 컨트롤 플레인 시스템 및 컴퓨팅 시스템을 배포할 기존 서브넷을 지정합니다.

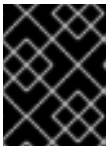
   d. 다음과 같은 이미지 콘텐츠 리소스를 추가합니다.

   ```
   imageContentSources:
   - mirrors:
     - <mirror_host_name>:5000/<repo_name>/release
     source: quay.io/openshift-release-dev/ocp-release
   - mirrors:
     - <mirror_host_name>:5000/<repo_name>/release
     source: registry.redhat.io/ocp/release
   ```

   이러한 값을 완료하려면 미러 레지스트리 생성 중에 기록한 **imageContentSources**를 사용하십시오.

3. 필요한 **install-config.yaml** 파일을 수정합니다. 사용 가능한 매개변수에 대한 자세한 정보는 **Installation configuration parameters** 섹션에서 확인할 수 있습니다.

4. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.



중요

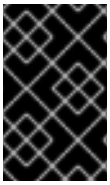
install-config.yaml 파일은 설치 과정에서 사용됩니다. 이 파일을 재사용하려면 지금 백업해야 합니다.

7.7.5.1. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 매개변수 값을 제공하여 클러스터를 호스팅할 클라우드 플랫폼에서 사용자 계정을 설명하고 선택사항으로 클러스터의 플랫폼을 사용자 지정합니다. **install-config.yaml** 설치 구성 파일을 생성할 때 명령줄을 통해 필요한 매개변수 값을 제공합니다. 클러스터를 사용자 지정하면 **install-config.yaml** 파일을 수정하여 플랫폼에 대한 세부 정보를 제공할 수 있습니다.

참고

설치한 후에는 **install-config.yaml** 파일에서 이러한 매개변수를 수정할 수 없습니다.



중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

7.7.5.1.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 7.20. 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체

매개변수	설명	값
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	소문자, 하이픈(**-**), 마침표(**.**)로 구성되는 문자열(예: **dev**)입니다.
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

7.7.5.1.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 7.21. 네트워크 매개변수

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체 **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.

매개변수	설명	값
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.
networking.clusterNetwork	Pod의 IP 주소 블록입니다. 기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된**cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16

매개변수	설명	값
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16**  **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

7.7.5.1.3. 선택적 구성 매개변수

선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

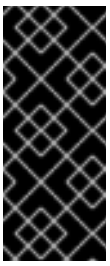
표 7.22. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열

매개변수	설명	값
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은 **3**입니다.
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열

매개변수	설명	값
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다. GCP에 공유 VPC(가상 프라이빗 클라우드)를 설치하는 경우 **credentialsMode** 를 **Passthrough** 로 설정해야 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 인터넷에서 액세스할 수 없는 사설 클러스터를 배포하려면 **publish**를 **Internal**로 설정합니다. 기본값은 **External**입니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

7.7.5.1.4. 추가 GCP(Google Cloud Platform) 구성 매개변수

추가 GCP 구성 매개변수는 다음 표에 설명되어 있습니다.

표 7.23. 추가 GCP 매개변수

매개변수	설명	값
platform.gcp.network	클러스터를 배포할 기존 VPC의 이름입니다.	문자열.
platform.gcp.region	클러스터를 호스팅하는 GCP 리전의 이름입니다.	유효한 리전 이름 (예: **us-central1**)
platform.gcp.type	GCP 시스템 유형	GCP 시스템 유형입니다.
platform.gcp.zones	설치 프로그램에서 지정된 MachinePool에 대한 시스템을 생성하는 가용 영역입니다.	**us-central1-a**와 같이 유효한 GCP 가용 영역 목록에 YAML 순서로 나열됩니다.
platform.gcp.controlPlaneSubnet	컨트롤 플레인 시스템을 배포할 VPC의 기존 서브넷 이름	서브넷 이름입니다.

매개변수	설명	값
platform.gcp.computeSubnet	컴퓨팅 시스템을 배포할 VPC의 기존 서브넷 이름	서브넷 이름입니다.
platform.gcp.licenses	컴퓨팅 이미지에 적용해야 하는 라이센스 URL 목록입니다. **중요** **licenses** 매개변수는 더 이상 사용되지 않는 필드이며 중첩된 가상화는 기본적으로 활성화되어 있습니다. 이 필드를 사용하지 않는 것이 좋습니다.	중첩된 가상화 를 활성화하는 라이센스와 같은 라이센스 API 와 함께 사용할 수 있는 모든 라이센스입니다. 사전 빌드된 이미지를 생성하는 메커니즘과 함께 이 매개변수를 사용할 수 없습니다. 라이센스 URL을 사용하면 설치 프로그램이 소스 이미지를 사용하기 전에 복사해야 합니다.
platform.gcp.osDisk.diskSizeGB	디스크 크기(GB)입니다.	16GB와 65536GB 사이의 모든 크기입니다.
platform.gcp.osDisk.diskType	디스크 유형입니다.	기본 **pd-ssd** 또는 **pd-standard** 디스크 유형입니다. 컨트롤 플레인 노드는 **pd-ssd** 디스크 유형이어야 합니다. 작업자 노드는 유형 중 하나일 수 있습니다.
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.name	컨트롤 플레인 머신 디스크 암호화에 사용되는 고객 관리 암호화 키의 이름입니다.	암호화 키 이름입니다.

매개변수	설명	값
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.keyRing	컨트롤 플레인 머신의 경우 KMS 키가 속한 KMS 키 링의 이름입니다.	KMS 키 링의 이름입니다.
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.location	컨트롤 플레인 머신의 경우 키 링이 있는 GCP 위치입니다. KMS 위치에 대한 자세한 내용은 Cloud KMS 위치에 대한 Google의 문서를 참조하십시오.	키 링의 GCP 위치입니다.
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.projectID	컨트롤 플레인 머신의 경우 KMS 키 링이 있는 프로젝트의 ID입니다. 이 값은 설정되지 않은 경우 VM 프로젝트 ID가 기본값입니다.	GCP 프로젝트 ID입니다.
compute.platform.gcp.osDisk.encryptionKey.kmsKey.name	컴퓨팅 머신 디스크 암호화에 사용되는 고객 관리 암호화 키의 이름입니다.	암호화 키 이름입니다.

매개변수	설명	값
compute.platform.gcp.osDisk.encryptionKey.kmsKey.keyRing	컴퓨팅 머신의 경우 KMS 키가 속한 KMS 키 링의 이름입니다.	KMS 키 링의 이름입니다.
compute.platform.gcp.osDisk.encryptionKey.kmsKey.location	컴퓨팅 머신의 경우 키 링이 있는 GCP 위치입니다. KMS 위치에 대한 자세한 내용은 Cloud KMS 위치에 대한 Google의 문서를 참조하십시오.	키 링의 GCP 위치입니다.
compute.platform.gcp.osDisk.encryptionKey.kmsKey.projectID	컴퓨팅 머신의 경우 KMS 키 링이 있는 프로젝트의 ID입니다. 이 값은 설정되지 않은 경우 VM 프로젝트 ID가 기본값입니다.	GCP 프로젝트 ID입니다.

7.7.5.2. 클러스터 설치를 위한 최소 리소스 요구 사항

각 클러스터 시스템이 다음과 같은 최소 요구사항을 충족해야 합니다.

표 7.24. 최소 리소스 요구사항

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
부트스트랩	RHCOS	4	16GB	100GB	300
컨트롤 플레인	RHCOS	4	16GB	100GB	300

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
컴퓨팅	RHCOS, RHEL 7.9 또는 RHEL 8.4 [3]	2	8GB	100GB	300

1. SMT(동시 멀티스레딩) 또는 하이퍼 스레딩이 활성화되지 않은 경우 하나의 vCPU는 하나의 물리적 코어와 동일합니다. 활성화하면 다음과 같은 공식을 사용하여 해당 비율을 계산합니다. (코어당 스레드 수 × 코어 수) × 소켓 수 = vCPU 수

2. OpenShift Container Platform 및 Kubernetes는 디스크 성능에 민감하며 특히 10ms p99 fsync 기간이 필요한 컨트롤 플레인 노드의 etcd에 더 빠른 스토리지가 권장됩니다. 많은 클라우드 플랫폼에서 스토리지 크기와 IOPS를 함께 확장되므로 충분한 성능을 얻으려면 스토리지 볼륨을 과도하게 할당해야 할 수 있습니다.

3. 사용자가 프로비저닝한 모든 설치와 마찬가지로 클러스터에서 RHEL 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. RHEL 7 컴퓨팅 머신 사용은 더 이상 사용되지 않으며 향후 OpenShift Container Platform 4 릴리스에서 제거될 예정입니다.

7.7.5.3. 사용자 정의 머신 유형 사용

OpenShift Container Platform 클러스터를 설치하기 위해 사용자 정의 머신 유형을 사용할 수 있습니다.

사용자 지정 머신 유형을 사용할 때는 다음을 고려하십시오.

- 사전 정의된 인스턴스 유형과 유사하게 사용자 정의 머신 유형은 컨트롤 플레인 및 컴퓨팅 시스템의 최소 리소스 요구 사항을 충족해야 합니다. 자세한 내용은 "클러스터 설치를 위한 최소 리소스 요구 사항"을 참조하십시오.

- 사용자 정의 머신 유형의 이름은 다음 구문을 준수해야 합니다.
  custom-<number_of_cpus>-<amount_of_memory_in_mb>

  예를 들면 **custom-6-20480** 입니다.

설치 프로세스의 일부로 **install-config.yaml** 파일에 사용자 지정 머신 유형을 지정합니다.

사용자 지정 머신 유형이 있는 샘플 install-config.yaml 파일

```
compute:
- architecture: amd64
  hyperthreading: Enabled
  name: worker
  platform:
    gcp:
      type: custom-6-20480
  replicas: 2
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  platform:
```

```
    gcp:
      type: custom-6-20480
  replicas: 3
```

7.7.5.4. GCP용 샘플 사용자 지정 install-config.yaml 파일

install-config.yaml 파일을 사용자 지정하여 OpenShift Container Platform 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.

중요

이 샘플 YAML 파일은 참조용으로만 제공됩니다. 설치 프로그램을 사용하여 **install-config.yaml** 파일을 받아서 수정해야 합니다.

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2) (3)
  hyperthreading: Enabled (4)
  name: master
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-ssd
        diskSizeGB: 1024
        encryptionKey: (5)
          kmsKey:
            name: worker-key
            keyRing: test-machine-keys
            location: global
            projectID: project-id
  replicas: 3
compute: (6) (7)
- hyperthreading: Enabled (8)
  name: worker
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-standard
        diskSizeGB: 128
        encryptionKey: (9)
          kmsKey:
            name: worker-key
            keyRing: test-machine-keys
            location: global
            projectID: project-id
```

```
  replicas: 3
metadata:
  name: test-cluster (10)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  gcp:
    projectID: openshift-production (11)
    region: us-central1 (12)
    network: existing_vpc (13)
    controlPlaneSubnet: control_plane_subnet (14)
    computeSubnet: compute_subnet (15)
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}}' (16)
fips: false (17)
sshKey: ssh-ed25519 AAAA... (18)
additionalTrustBundle: | (19)
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
imageContentSources: (20)
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

(1) (10) (11) (12) 필수 항목입니다. 설치 프로그램에서 이 값을 입력하라는 메시지를 표시합니다.

(2) (6) 이러한 매개변수와 값을 지정하지 않으면 설치 프로그램은 기본값을 적용합니다.

(3) (7) **controlPlane** 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

(4) (8) 동시 멀티스레딩 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. 일부 클러스터 시스템에서 동시 멀티스레딩을 비활성화할 경우에는 해당 멀티스레딩을 모든 클러스터 시스템에서 비활성화해야 합니다.

중요

동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다. 동시 멀티스레딩을 비활성화하는 경우, 사용자 시스템에 더 큰 시스템 유형(예: **n1-standard-8**)을 사용하십시오.

5 9 선택 사항: 사용자 정의 암호화 키 섹션으로 가상 머신과 영구 볼륨을 모두 암호화합니다. 기본 컴퓨팅 서비스 계정에는 KMS 키를 사용하도록 부여된 권한이 있어야 하며 올바른 IAM 역할을 할당해야

13 기존 VPC의 이름을 지정합니다.

14 컨트롤 플레인 시스템을 배포할 기존 서브넷의 이름을 지정합니다. 지정한 VPC에 속하는 서브넷이어야 합니다.

15 컴퓨팅 시스템을 배포할 기존 서브넷의 이름을 지정합니다. 지정한 VPC에 속하는 서브넷이어야 합니다.

16 **<local_registry>**는 미러 레지스트리가 해당 내용을 제공하는 데 사용하는 레지스트리 도메인 이름과 포트(선택사항)를 지정합니다. 예: **registry.example.com** 또는 **registry.example.com:5000** **<credentials>**는 미러 레지스트리의 base64 인코딩 사용자 이름과 암호를 지정합니다.

17 FIPS 모드 활성화 또는 비활성화 여부입니다. 기본적으로 FIPS 모드는 비활성화됩니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.

중요

FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.

18 선택사항으로, 클러스터의 시스템에 액세스하는 데 사용할 **sshKey** 값을 제공할 수도 있습니다.

참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

19 미러 레지스트리에 사용한 인증서 파일의 내용을 제공하십시오.

20 명령 출력에서 **imageContentSources** 섹션을 제공하여 리포지토리를 미러링하십시오.

7.7.5.5. GCP에서 글로벌 액세스 권한이 있는 Ingress 컨트롤러 생성

GCP(Google Cloud Platform) 클러스터에 글로벌 액세스할 수 있는 Ingress 컨트롤러를 생성할 수 있습니다. 글로벌 액세스는 내부 로드 밸런서를 사용하는 Ingress 컨트롤러에서만 사용할 수 있습니다.

사전 요구 사항

- **install-config.yaml**을 생성하고 수정 완료했습니다.

절차

새 GCP 클러스터에서 글로벌 액세스 권한이 있는 Ingress 컨트롤러를 생성합니다.

1. 설치 프로그램이 포함된 디렉터리로 변경하고 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>**는 클러스터의 **install-config.yaml** 파일이 포함된 디렉터리의 이름을 지정합니다.

2. **<installation_directory>/manifests/** 디렉터리에 **cluster-ingress-default-ingresscontroller.yaml**이라는 이름으로 파일을 만듭니다.

```
$ touch <installation_directory>/manifests/cluster-ingress-default-ingresscontroller.yaml 1
```

1 **<installation_directory>**는 클러스터의 **manifests** / 디렉터리가 포함된 디렉터리 이름을 지정합니다.

파일이 생성되면 다음과 같이 여러 네트워크 구성 파일이 **manifests/** 디렉토리에 나타납니다.

```
$ ls <installation_directory>/manifests/cluster-ingress-default-ingresscontroller.yaml
```

출력 예

```
cluster-ingress-default-ingresscontroller.yaml
```

3. 편집기에서 **cluster-ingress-default-ingresscontroller.yaml** 파일을 열고 원하는 운영자 구성을 설명하는 CR(사용자 정의 리소스)을 입력합니다.

Global에 대한 clientAccess 구성 샘플

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: default
  namespace: openshift-ingress-operator
spec:
  endpointPublishingStrategy:
    loadBalancer:
      providerParameters:
        gcp:
          clientAccess: Global 1
        type: GCP
      scope: Internal 2
    type: LoadBalancerService
```

1 **gcp.clientAccess**를 **Global**로 설정합니다.

2 글로벌 액세스는 내부 로드 밸런서를 사용하는 Ingress 컨트롤러에서만 사용할 수 있습니다.

7.7.5.6. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 HTTP 또는 HTTPS 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 OpenShift Container Platform 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 API에 대한 호출을 포함하여 모든 클러스터 발신(Egress) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.

참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

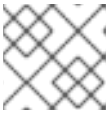
Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 Red Hat OpenStack Platform (RHOSP)에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. 클러스터 외부에서 HTTP 연결을 구축하는 데 사용할 프록시 URL입니다. URL 스키마는 **http**여야 합니다.
2. 클러스터 외부에서 HTTPS 연결을 구축하는 데 사용할 프록시 URL입니다.
3. 대상 도메인 이름, IP 주소 또는 프록시에서 제외할 기타 네트워크 CIDR로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. ***를 사용하여 모든 대상에 대해 프록시를 바이패스합니다.
4. 이 값을 제공하면 설치 프로그램에서 추가 CA 인증서를 보유할 **openshift -config 네임스페이스에 user-ca- bundle** 이라는 구성 맵을 생성합니다. **additionalTrustBundle** 및 하나 이상의 프록시 설정을 제공하는 경우 **프록시** 오브젝트는 **trustedCA** 필드의 **user-ca-bundle** 구성 맵을 참조하도록 구성됩니다. 그러면 Cluster Network Operator에서 **trustedCA** 매개변수에 대해 지정된 콘텐츠를 RHCOS 신뢰 번들과 병합하는 **trusted-ca-bundle** 구성 맵을 생성합니다. 프록시의 ID 인증서를 RHCOS 트러스트 번들에 있는 기관에서 서명하지 않은 경우 **additionalTrustBundle** 필드가 있어야 합니다.



참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 OpenShift Container Platform을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 nil이 됩니다.

참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

7.7.6. 클러스터 배포

호환되는 클라우드 플랫폼에 OpenShift Container Platform을 설치할 수 있습니다.

중요

최초 설치 과정에서 설치 프로그램의 **create cluster** 명령을 한 번만 실행할 수 있습니다.

사전 요구 사항

- 클러스터를 호스팅하는 클라우드 플랫폼으로 계정을 구성합니다.
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

절차

1. 클러스터에 맞춰 구성하여 다음 위치에 저장한 GCP 계정의 서비스 계정 키를 사용하지 않는 기존 GCP 사용자 자격 증명을 삭제합니다.
   - **GOOGLE_CREDENTIALS**, **GOOGLE_CLOUD_KEYFILE_JSON** 또는 **GCLOUD_KEYFILE_JSON** 환경 변수
   - **~/.gcp/osServiceAccount.json** 파일
   - **gcloud cli** 기본 자격 증명
2. 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터 배포를 초기화합니다.

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** 값으로 사용자 지정한 **./install-config.yaml** 파일의 위치를 지정합니다.

2 다른 설치 세부 사항을 보려면 **info** 대신 **warn**, **debug** 또는 **error**를 지정합니다.

참고

호스트에 구성된 클라우드 공급자 계정에 클러스터를 배포하기에 충분한 권한이 없는 경우, 설치 프로세스가 중단되고 누락된 권한을 알리는 메시지가 표시됩니다.

클러스터 배포가 완료되면 웹 콘솔로 연결되는 링크와 **kubeadmin** 사용자의 인증 정보가 포함된 클러스터 액세스 지침이 사용자 터미널에 표시됩니다.

출력 예

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```

참고

설치에 성공하면 클러스터 액세스 및 인증 정보도 **<installation_directory>/.openshift_install.log**로 출력됩니다.

중요

- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

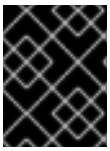
중요

설치 프로그램에서 생성되는 파일이나 설치 프로그램을 삭제해서는 안 됩니다. 클러스터를 삭제하려면 두 가지가 모두 필요합니다.

3. 선택사항: 클러스터를 설치하는 데 사용한 서비스 계정에 대한 권한 수를 줄일 수 있습니다.
   - 서비스 계정에 **Owner** 역할을 할당한 경우, 해당 역할을 제거하고 **Viewer** 역할로 바꿀 수 있습니다.
   - **Service Account Key Admin** 역할이 포함되어 있으면 제거할 수도 있습니다.

7.7.7. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift CLI(**oc**)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 **oc**를 설치할 수 있습니다.



중요

이전 버전의 **oc**를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 **oc**를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 Linux에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Linux Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. **oc** 바이너리를 **PATH**에 있는 디렉터리에 배치합니다.
PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치

다음 절차에 따라 Windows에 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Windows Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. ZIP 프로그램으로 아카이브의 압축을 풉니다.
5. **oc** 바이너리를 **PATH**에 있는 디렉터리로 이동합니다.
PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치
다음 절차에 따라 macOS에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 MacOSX Client** 항목 옆에 있는 **지금 다운로드를** 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 해제하고 압축을 풉니다.
5. **oc** 바이너리 PATH의 디렉터리로 이동합니다.
   PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

7.7.8. CLI를 사용하여 클러스터에 로그인

클러스터 **kubeconfig** 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. **kubeconfig** 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- **oc** CLI를 설치했습니다.

프로세스

1. **kubeadmin** 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 **oc** 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

7.7.9. 기본 OperatorHub 소스 비활성화

Red Hat 및 커뮤니티 프로젝트에서 제공하는 콘텐츠를 소싱하는 Operator 카탈로그는 OpenShift Container Platform을 설치하는 동안 기본적으로 OperatorHub용으로 구성됩니다. 제한된 네트워크 환경에서는 클러스터 관리자로서 기본 카탈로그를 비활성화해야 합니다.

프로세스

- **OperatorHub** 오브젝트에 **disableAllDefaultSources: true**를 추가하여 기본 카탈로그의 소스를 비활성화합니다.

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

작은 정보

또는 웹 콘솔을 사용하여 카탈로그 소스를 관리할 수 있습니다. **관리 → 클러스터 설정 → 구성 → OperatorHub** 페이지에서 개별 소스를 생성, 삭제, 비활성화 및 활성화할 수 있는 소스 **탭을** 클릭합니다.

7.7.10. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

7.7.11. 다음 단계

- 설치를 확인합니다.
- 클러스터를 사용자 지정합니다.
- Cluster Samples Operator 및 **must-gather** 툴의 이미지 스트림을 구성합니다.
- 제한된 네트워크에서 Operator Lifecycle Manager (OLM) 사용 방법에 대해 살펴봅니다.
- 클러스터를 설치하는 데 사용한 미러 레지스트리에 신뢰할 수 있는 CA가 있는 경우 추가 신뢰 저장소를 구성하여 클러스터에 추가합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.

7.8. GCP의 클러스터를 기존 VPC에 설치

OpenShift Container Platform 4.9 버전에서는 GCP(Google Cloud Platform)의 기존 VPC(Virtual Private Cloud)에 클러스터를 설치할 수 있습니다. 설치 프로그램이 나머지 필수 인프라를 프로비저닝하며, 이후에 추가로 사용자 지정할 수 있습니다. 설치를 사용자 지정하려면 클러스터를 설치하기 전에 **install-**

config.yaml 파일에서 매개변수를 수정합니다.

7.8.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비 에 대한 문서를 읽습니다.
- 클러스터를 호스팅할 GCP 프로젝트를 구성 했습니다.
- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성 해야 합니다.
- 사용자의 환경에서 클라우드 ID 및 액세스 관리(IAM) API에 액세스할 수 없거나 **kube-system** 네임스페이스에 관리자 수준의 인증 정보 시크릿을 저장하지 않으려면 IAM 인증 정보 수동 생성 및 유지 관리를 참조하십시오.

7.8.2. 사용자 지정 VPC 사용 정보

OpenShift Container Platform 4.9에서는 GCP(Google Cloud Platform)의 기존 VPC(Virtual Private Cloud)에 있는 기존 서브넷에 클러스터를 배포할 수 있습니다. 기존 GCP VPC에 OpenShift Container Platform을 배포하면 새 계정의 한도 제한을 회피하거나 회사의 지침에 따른 운영 제한을 따르기 수월해집니다. VPC를 직접 생성하는 데 필요한 인프라 생성 권한을 받을 수 없는 경우 이 설치 옵션을 사용합니다. 서브넷에 대한 네트워킹을 구성해야 합니다.

7.8.2.1. VPC 사용 요구사항

VPC CIDR 블록과 머신 네트워크 CIDR의 통합은 비어 있지 않아야 합니다. 서브넷은 시스템 네트워크 내에 있어야 합니다.

설치 프로그램에서 다음 구성 요소를 생성하지 않습니다.

- NAT 게이트웨이
- 서브넷
- 라우팅 테이블
- VPC 네트워크

참고

설치 프로그램을 사용하려면 클라우드 제공 DNS 서버를 사용해야 합니다. 사용자 지정 DNS 서버 사용은 지원되지 않으며 이로 인해 설치에 실패합니다.

7.8.2.2. VPC 검증

제공한 서브넷이 적합한지 확인하기 위해 설치 프로그램이 다음 데이터를 확인합니다.

- 사용자가 지정하는 모든 서브넷이 있는지 여부.
- 컨트롤 플레인 시스템용 서브넷과 컴퓨팅 시스템용 서브넷 1개를 제공합니다.
- 서브넷의 CIDR은 사용자가 지정한 시스템 CIDR에 속합니다.

7.8.2.3. 권한 분할

일부 개인은 클라우드에서 다른 리소스와 다른 리소스를 만들 수 있습니다. 예를 들어 인스턴스, 버킷, 로드 밸런서와 같은 애플리케이션 관련 항목은 생성할 수 있지만 VPC 서브넷 또는 인그레스 규칙과 같은 네트워킹 관련 구성 요소는 생성하지 못할 수 있습니다.

7.8.2.4. 클러스터 간 격리

OpenShift Container Platform을 기존 네트워크에 배포하면 클러스터 서비스 격리가 다음과 같은 방식으로 감소합니다.

- 동일한 VPC에 여러 OpenShift Container Platform 클러스터를 설치할 수 있습니다.
- ICMP 인그레스가 전체 네트워크에 허용됩니다.
- TCP 22 인그레스(SSH)가 전체 네트워크에 허용됩니다.
- 컨트롤 플레인 TCP 6443 인그레스(Kubernetes API)가 전체 네트워크에 허용됩니다.
- 컨트롤 플레인 TCP 22623 인그레스(MCS)가 전체 네트워크에 허용됩니다.

7.8.3. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.

중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

7.8.4. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 **core** 사용자의 **~/.ssh/authorized_keys** 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 **core**로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. **./openshift-install gather** 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

참고

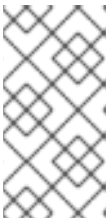
AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: **~/.ssh/id_ed25519**)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 **'~/.ssh** 디렉터리에 있는지 확인하십시오.



참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 **ed25519** 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 **rsa** 또는 **ecdsa** 알고리즘을 사용하는 키를 생성합니다.

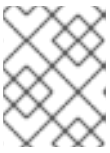
2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 **~/.ssh/id_ed25519.pub** 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 **./openshift-install gather** 명령을 사용하려는 경우 필요합니다.



참고

일부 배포에서는 **~/.ssh/id_rsa** 및 **~/.ssh/id_dsa**와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. **ssh-agent** 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. **ssh-agent**에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: **~/.ssh/id_ed25519**).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

5. **GOOGLE_APPLICATION_CREDENTIALS** 환경 변수를 서비스 계정 개인 키 파일의 전체 경로로 설정합니다.

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

6. 인증 정보가 적용되었는지 확인합니다.

```
$ gcloud auth list
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

7.8.5. 설치 프로그램 받기

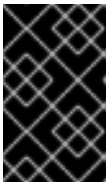
OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

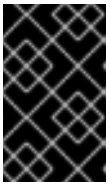
프로세스

1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.
2. 인프라 공급자를 선택합니다.
3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.



중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.



중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 OpenShift Container Platform 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

7.8.6. 설치 구성 파일 만들기

GCP(Google Cloud Platform)에 설치하는 OpenShift Container Platform 클러스터를 사용자 지정할 수 있습니다.

사전 요구 사항

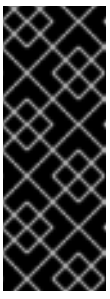
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.
- 서브스크립션 수준에서 서비스 주체 권한을 획득합니다.

프로세스

1. **install-config.yaml** 파일을 생성합니다.
   a. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

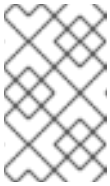
1 **<installation_directory>**는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.



중요

비어 있는 디렉터리를 지정합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

   b. 화면에 나타나는 지시에 따라 클라우드에 대한 구성 세부 사항을 입력합니다.
      i. 선택사항: 클러스터 시스템에 액세스하는 데 사용할 SSH 키를 선택합니다.



참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

ii. 대상 플랫폼으로 **gcp**를 선택합니다.

iii. 사용자 컴퓨터에 GCP 계정의 서비스 계정 키가 구성되어 있지 않은 경우, GCP에서 해당 키를 가져와 파일의 내용을 붙여 넣거나 파일의 전체 경로를 입력해야 합니다.

iv. 클러스터를 프로비저닝할 프로젝트 ID를 선택합니다. 구성한 서비스 계정에 의해 기본값이 지정됩니다.

v. 클러스터를 배포할 리전을 선택합니다.

vi. 클러스터를 배포할 기본 도메인을 선택합니다. 기본 도메인은 클러스터용으로 생성한 퍼블릭 DNS 영역에 해당합니다.

vii. 클러스터를 설명할 수 있는 이름을 입력합니다.

viii. Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 붙여넣습니다.

2. **install-config.yaml** 파일을 수정합니다. 사용 가능한 매개변수에 대한 자세한 정보는 "설치 구성 매개변수" 섹션에서 확인할 수 있습니다.

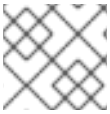
3. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.

중요

install-config.yaml 파일은 설치 과정에서 사용됩니다. 이 파일을 재사용하려면 지금 백업해야 합니다.

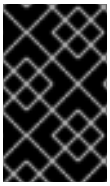
7.8.6.1. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 매개변수 값을 제공하여 클러스터를 호스팅할 클라우드 플랫폼에서 사용자 계정을 설명하고 선택사항으로 클러스터의 플랫폼을 사용자 지정합니다. **install-config.yaml** 설치 구성 파일을 생성할 때 명령줄을 통해 필요한 매개변수 값을 제공합니다. 클러스터를 사용자 지정하면 **install-config.yaml** 파일을 수정하여 플랫폼에 대한 세부 정보를 제공할 수 있습니다.



참고

설치한 후에는 **install-config.yaml** 파일에서 이러한 매개변수를 수정할 수 없습니다.



중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

7.8.6.1.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 7.25. 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	소문자, 하이픈(**-**), 마침표(**.**)로 구성되는 문자열(예: **dev**)입니다.
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

매개변수	설명	값

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

7.8.6.1.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 7.26. 네트워크 매개변수

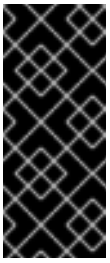
매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체 **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.

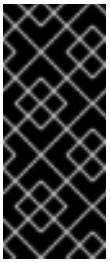
매개변수	설명	값
networking.clusterNetwork	Pod의 IP 주소 블록입니다. 기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된 **cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16**  **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

7.8.6.1.3. 선택적 구성 매개변수

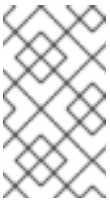
선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 7.27. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은**3**입니다.

매개변수	설명	값
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다. GCP에 공유 VPC(가상 프라이빗 클라우드)를 설치하는 경우 **credentialsMode** 를 **Passthrough** 로 설정해야 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 인터넷에서 액세스할 수 없는 사설 클러스터를 배포하려면 **publish**를 **Internal**로 설정합니다. 기본값은 **External**입니다.

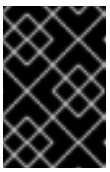
매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

7.8.6.1.4. 추가 GCP(Google Cloud Platform) 구성 매개변수

추가 GCP 구성 매개변수는 다음 표에 설명되어 있습니다.

표 7.28. 추가 GCP 매개변수

매개변수	설명	값
platform.gcp.network	클러스터를 배포할 기존 VPC의 이름입니다.	문자열.
platform.gcp.region	클러스터를 호스팅하는 GCP 리전의 이름입니다.	유효한 리전 이름 (예: **us-central1**)
platform.gcp.type	GCP 시스템 유형	GCP 시스템 유형입니다.
platform.gcp.zones	설치 프로그램에서 지정된 MachinePool에 대한 시스템을 생성하는 가용 영역입니다.	**us-central1-a**와 같이 유효한 GCP 가용 영역 목록에 YAML 순서로 나열됩니다.
platform.gcp.controlPlaneSubnet	컨트롤 플레인 시스템을 배포할 VPC의 기존 서브넷 이름	서브넷 이름입니다.

매개변수	설명	값
platform.gcp.computeSubnet	컴퓨팅 시스템을 배포할 VPC의 기존 서브넷 이름	서브넷 이름입니다.
platform.gcp.licenses	컴퓨팅 이미지에 적용해야 하는 라이센스 URL 목록입니다.  **중요** **licenses** 매개변수는 더 이상 사용되지 않는 필드이며 중첩된 가상화는 기본적으로 활성화되어 있습니다. 이 필드를 사용하지 않는 것이 좋습니다.	중첩된 가상화 를 활성화하는 라이센스와 같은 라이센스 API 와 함께 사용할 수 있는 모든 라이센스입니다. 사전 빌드된 이미지를 생성하는 메커니즘과 함께 이 매개변수를 사용할 수 없습니다. 라이센스 URL을 사용하면 설치 프로그램이 소스 이미지를 사용하기 전에 복사해야 합니다.
platform.gcp.osDisk.diskSizeGB	디스크 크기(GB)입니다.	16GB와 65536GB 사이의 모든 크기입니다.
platform.gcp.osDisk.diskType	디스크 유형입니다.	기본 **pd-ssd** 또는 **pd-standard** 디스크 유형입니다. 컨트롤 플레인 노드는 **pd-ssd** 디스크 유형이어야 합니다. 작업자 노드는 유형 중 하나일 수 있습니다.
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.name	컨트롤 플레인 머신 디스크 암호화에 사용되는 고객 관리 암호화 키의 이름입니다.	암호화 키 이름입니다.

매개변수	설명	값
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.keyRing	컨트롤 플레인 머신의 경우 KMS 키가 속한 KMS 키 링의 이름입니다.	KMS 키 링의 이름입니다.
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.location	컨트롤 플레인 머신의 경우 키 링이 있는 GCP 위치입니다. KMS 위치에 대한 자세한 내용은 Cloud KMS 위치에 대한 Google의 문서를 참조하십시오.	키 링의 GCP 위치입니다.
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.projectID	컨트롤 플레인 머신의 경우 KMS 키 링이 있는 프로젝트의 ID입니다. 이 값은 설정되지 않은 경우 VM 프로젝트 ID가 기본값입니다.	GCP 프로젝트 ID입니다.
compute.platform.gcp.osDisk.encryptionKey.kmsKey.name	컴퓨팅 머신 디스크 암호화에 사용되는 고객 관리 암호화 키의 이름입니다.	암호화 키 이름입니다.

매개변수	설명	값
compute.platform.gcp.osDisk.encryptionKey.kmsKey.keyRing	컴퓨팅 머신의 경우 KMS 키가 속한 KMS 키 링의 이름입니다.	KMS 키 링의 이름입니다.
compute.platform.gcp.osDisk.encryptionKey.kmsKey.location	컴퓨팅 머신의 경우 키 링이 있는 GCP 위치입니다. KMS 위치에 대한 자세한 내용은 Cloud KMS 위치에 대한 Google의 문서를 참조하십시오.	키 링의 GCP 위치입니다.
compute.platform.gcp.osDisk.encryptionKey.kmsKey.projectID	컴퓨팅 머신의 경우 KMS 키 링이 있는 프로젝트의 ID입니다. 이 값은 설정되지 않은 경우 VM 프로젝트 ID가 기본값입니다.	GCP 프로젝트 ID입니다.

7.8.6.2. 클러스터 설치를 위한 최소 리소스 요구 사항

각 클러스터 시스템이 다음과 같은 최소 요구사항을 충족해야 합니다.

표 7.29. 최소 리소스 요구사항

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
부트스트랩	RHCOS	4	16GB	100GB	300
컨트롤 플레인	RHCOS	4	16GB	100GB	300

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
컴퓨팅	RHCOS, RHEL 7.9 또는 RHEL 8.4 [3]	2	8GB	100GB	300

1. SMT(동시 멀티스레딩) 또는 하이퍼 스레딩이 활성화되지 않은 경우 하나의 vCPU는 하나의 물리적 코어와 동일합니다. 활성화하면 다음과 같은 공식을 사용하여 해당 비율을 계산합니다. (코어당 스레드 수 × 코어 수) × 소켓 수 = vCPU 수

2. OpenShift Container Platform 및 Kubernetes는 디스크 성능에 민감하며 특히 10ms p99 fsync 기간이 필요한 컨트롤 플레인 노드의 etcd에 더 빠른 스토리지가 권장됩니다. 많은 클라우드 플랫폼에서 스토리지 크기와 IOPS를 함께 확장되므로 충분한 성능을 얻으려면 스토리지 볼륨을 과도하게 할당해야 할 수 있습니다.

3. 사용자가 프로비저닝한 모든 설치와 마찬가지로 클러스터에서 RHEL 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. RHEL 7 컴퓨팅 머신 사용은 더 이상 사용되지 않으며 향후 OpenShift Container Platform 4 릴리스에서 제거될 예정입니다.

7.8.6.3. 사용자 정의 머신 유형 사용

OpenShift Container Platform 클러스터를 설치하기 위해 사용자 정의 머신 유형을 사용할 수 있습니다.

사용자 지정 머신 유형을 사용할 때는 다음을 고려하십시오.

- 사전 정의된 인스턴스 유형과 유사하게 사용자 정의 머신 유형은 컨트롤 플레인 및 컴퓨팅 시스템의 최소 리소스 요구 사항을 충족해야 합니다. 자세한 내용은 "클러스터 설치를 위한 최소 리소스 요구 사항"을 참조하십시오.

- 사용자 정의 머신 유형의 이름은 다음 구문을 준수해야 합니다.
  custom-<number_of_cpus>-<amount_of_memory_in_mb>

  예를 들면 **custom-6-20480** 입니다.

설치 프로세스의 일부로 **install-config.yaml** 파일에 사용자 지정 머신 유형을 지정합니다.

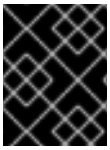
사용자 지정 머신 유형이 있는 샘플 install-config.yaml 파일

```
compute:
- architecture: amd64
  hyperthreading: Enabled
  name: worker
  platform:
    gcp:
      type: custom-6-20480
  replicas: 2
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  platform:
```

```
    gcp:
      type: custom-6-20480
  replicas: 3
```

7.8.6.4. GCP용 샘플 사용자 지정 install-config.yaml 파일

install-config.yaml 파일을 사용자 지정하여 OpenShift Container Platform 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.



중요

이 샘플 YAML 파일은 참조용으로만 제공됩니다. 설치 프로그램을 사용하여 **install-config.yaml** 파일을 받아서 수정해야 합니다.

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2) (3)
  hyperthreading: Enabled (4)
  name: master
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-ssd
        diskSizeGB: 1024
        encryptionKey: (5)
          kmsKey:
            name: worker-key
            keyRing: test-machine-keys
            location: global
            projectID: project-id
  replicas: 3
compute: (6) (7)
- hyperthreading: Enabled (8)
  name: worker
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-standard
        diskSizeGB: 128
        encryptionKey: (9)
          kmsKey:
            name: worker-key
            keyRing: test-machine-keys
            location: global
            projectID: project-id
```

```
  replicas: 3
metadata:
  name: test-cluster 10
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  gcp:
    projectID: openshift-production 11
    region: us-central1 12
    network: existing_vpc 13
    controlPlaneSubnet: control_plane_subnet 14
    computeSubnet: compute_subnet 15
pullSecret: '{"auths": ...}' 16
fips: false 17
sshKey: ssh-ed25519 AAAA... 18
```

1 10 11 12 16 필수 항목입니다. 설치 프로그램에서 이 값을 입력하라는 메시지를 표시합니다.

2 6 이러한 매개변수와 값을 지정하지 않으면 설치 프로그램은 기본값을 적용합니다.

3 7 **controlPlane** 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

4 8 동시 멀티스레딩 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. 일부 클러스터 시스템에서 동시 멀티스레딩을 비활성화할 경우에는 해당 멀티스레딩을 모든 클러스터 시스템에서 비활성화해야 합니다.

중요

동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다. 동시 멀티스레딩을 비활성화하는 경우, 사용자 시스템에 더 큰 시스템 유형(예: **n1-standard-8**)을 사용하십시오.

5 9 선택 사항: 사용자 정의 암호화 키 섹션으로 가상 머신과 영구 볼륨을 모두 암호화합니다. 기본 컴퓨팅 서비스 계정에는 KMS 키를 사용하도록 부여된 권한이 있어야 하며 올바른 IAM 역할을 할당해야 합니다. 기본 서비스 계정 이름은 **service-<project_number>@compute-system.iam.gserviceaccount.com** 패턴을 따릅니다. 서비스 계정에 대한 올바른 권한을 부여하는 방법에 대한 자세한 내용은 "머신 관리" → " 머신 세트 생성" → “GCP에서 머신 세트 생성”을 참조하십시오.

13 기존 VPC의 이름을 지정합니다.

14 컨트롤 플레인 시스템을 배포할 기존 서브넷의 이름을 지정합니다. 지정한 VPC에 속하는 서브넷이어야 합니다.

15 컴퓨팅 시스템을 배포할 기존 서브넷의 이름을 지정합니다. 지정한 VPC에 속하는 서브넷이어야 합니다.

17 FIPS 모드 활성화 또는 비활성화 여부입니다. 기본적으로 FIPS 모드는 비활성화됩니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.

중요

FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.

18 선택사항으로, 클러스터의 시스템에 액세스하는 데 사용할 **sshKey** 값을 제공할 수도 있습니다.

참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

7.8.6.5. GCP에서 글로벌 액세스 권한이 있는 Ingress 컨트롤러 생성

GCP(Google Cloud Platform) 클러스터에 글로벌 액세스할 수 있는 Ingress 컨트롤러를 생성할 수 있습니다. 글로벌 액세스는 내부 로드 밸런서를 사용하는 Ingress 컨트롤러에서만 사용할 수 있습니다.

사전 요구 사항

- **install-config.yaml**을 생성하고 수정 완료했습니다.

절차

새 GCP 클러스터에서 글로벌 액세스 권한이 있는 Ingress 컨트롤러를 생성합니다.

1. 설치 프로그램이 포함된 디렉터리로 변경하고 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>**는 클러스터의 **install-config.yaml** 파일이 포함된 디렉터리의 이름을 지정합니다.

2. **<installation_directory>/manifests/** 디렉터리에 **cluster-ingress-default-ingresscontroller.yaml**이라는 이름으로 파일을 만듭니다.

```
$ touch <installation_directory>/manifests/cluster-ingress-default-ingresscontroller.yaml 1
```

1 **<installation_directory>**는 클러스터의 **manifests** / 디렉터리가 포함된 디렉터리 이름을 지정합니다.

파일이 생성되면 다음과 같이 여러 네트워크 구성 파일이 **manifests/** 디렉토리에 나타납니다.

```
$ ls <installation_directory>/manifests/cluster-ingress-default-ingresscontroller.yaml
```

출력 예

```
cluster-ingress-default-ingresscontroller.yaml
```

3. 편집기에서 **cluster-ingress-default-ingresscontroller.yaml** 파일을 열고 원하는 운영자 구성을 설명하는 CR(사용자 정의 리소스)을 입력합니다.

Global에 대한 clientAccess 구성 샘플

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: default
  namespace: openshift-ingress-operator
spec:
  endpointPublishingStrategy:
    loadBalancer:
      providerParameters:
        gcp:
          clientAccess: Global 1
        type: GCP
      scope: Internal 2
    type: LoadBalancerService
```

1 **gcp.clientAccess**를 **Global**로 설정합니다.

2 글로벌 액세스는 내부 로드 밸런서를 사용하는 Ingress 컨트롤러에서만 사용할 수 있습니다.

7.8.7. 추가 리소스

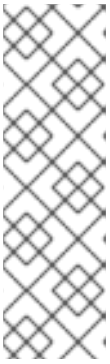
- 머신 세트의 고객 관리 암호화 키 활성화

7.8.7.1. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 HTTP 또는 HTTPS 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 OpenShift Container Platform 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 API에 대한 호출을 포함하여 모든 클러스터 발신(Egress) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.



참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 Red Hat OpenStack Platform (RHOSP)에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. 클러스터 외부에서 HTTP 연결을 구축하는 데 사용할 프록시 URL입니다. URL 스키마는 **http**여야 합니다.
2. 클러스터 외부에서 HTTPS 연결을 구축하는 데 사용할 프록시 URL입니다.
3. 대상 도메인 이름, IP 주소 또는 프록시에서 제외할 기타 네트워크 CIDR로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. ***를 사용하여 모든 대상에 대해 프록시를 바이패스합니다.
4. 이 값을 제공하면 설치 프로그램에서 추가 CA 인증서를 보유할 **openshift -config 네임스페이스에 user-ca- bundle** 이라는 구성 맵을 생성합니다. **additionalTrustBundle** 및 하나 이상의 프록시 설정을 제공하는 경우 **프록시** 오브젝트는 **trustedCA** 필드의 **user-ca-bundle** 구성 맵을 참조하도록 구성됩니다. 그러면 Cluster Network Operator에서 **trustedCA** 매개변수에 대해 지정된 콘텐츠를 RHCOS 신뢰 번들과 병합하는 **trusted-ca-bundle** 구성 맵을 생성합니다. 프록시의 ID 인증서를 RHCOS 트러스트 번들에 있는 기관에서 서명하지 않은 경우 **additionalTrustBundle** 필드가 있어야 합니다.

참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 OpenShift Container Platform을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 nil이 됩니다.

참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

7.8.8. 클러스터 배포

호환되는 클라우드 플랫폼에 OpenShift Container Platform을 설치할 수 있습니다.

중요

최초 설치 과정에서 설치 프로그램의 **create cluster** 명령을 한 번만 실행할 수 있습니다.

사전 요구 사항

- 클러스터를 호스팅하는 클라우드 플랫폼으로 계정을 구성합니다.
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

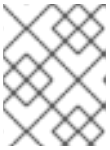
절차

1. 클러스터에 맞춰 구성하여 다음 위치에 저장한 GCP 계정의 서비스 계정 키를 사용하지 않는 기존 GCP 사용자 자격 증명을 삭제합니다.
   - **GOOGLE_CREDENTIALS**, **GOOGLE_CLOUD_KEYFILE_JSON** 또는 **GCLOUD_KEYFILE_JSON** 환경 변수
   - **~/.gcp/osServiceAccount.json** 파일
   - **gcloud cli** 기본 자격 증명
2. 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터 배포를 초기화합니다.

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** 값으로 사용자 지정한 **./install-config.yaml** 파일의 위치를 지정합니다.

2 다른 설치 세부 사항을 보려면 **info** 대신 **warn**, **debug** 또는 **error**를 지정합니다.



참고

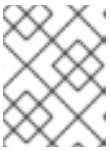
호스트에 구성된 클라우드 공급자 계정에 클러스터를 배포하기에 충분한 권한이 없는 경우, 설치 프로세스가 중단되고 누락된 권한을 알리는 메시지가 표시됩니다.

클러스터 배포가 완료되면 웹 콘솔로 연결되는 링크와 **kubeadmin** 사용자의 인증 정보가 포함된 클러스터 액세스 지침이 사용자 터미널에 표시됩니다.

출력 예

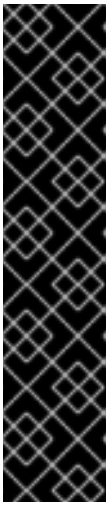
```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
```

```
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```



참고

설치에 성공하면 클러스터 액세스 및 인증 정보도 **<installation_directory>/.openshift_install.log**로 출력됩니다.



중요

- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

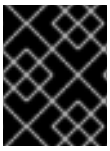
중요

설치 프로그램에서 생성되는 파일이나 설치 프로그램을 삭제해서는 안 됩니다. 클러스터를 삭제하려면 두 가지가 모두 필요합니다.

3. 선택사항: 클러스터를 설치하는 데 사용한 서비스 계정에 대한 권한 수를 줄일 수 있습니다.
   - 서비스 계정에 **Owner** 역할을 할당한 경우, 해당 역할을 제거하고 **Viewer** 역할로 바꿀 수 있습니다.
   - **Service Account Key Admin** 역할이 포함되어 있으면 제거할 수도 있습니다.

7.8.9. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift CLI(**oc**)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 **oc**를 설치할 수 있습니다.



중요

이전 버전의 **oc**를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 **oc**를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 Linux에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

절차

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Linux Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. **oc** 바이너리를 **PATH**에 있는 디렉터리에 배치합니다.
PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치
다음 절차에 따라 Windows에 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

절차

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Windows Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. ZIP 프로그램으로 아카이브의 압축을 풉니다.
5. **oc** 바이너리를 **PATH**에 있는 디렉터리로 이동합니다.
PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치
다음 절차에 따라 macOS에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

절차

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 MacOSX Client** 항목 옆에 있는 **지금 다운로드를** 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 해제하고 압축을 풉니다.

5. **oc** 바이너리 PATH의 디렉터리로 이동합니다.
PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

7.8.10. CLI를 사용하여 클러스터에 로그인

클러스터 **kubeconfig** 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. **kubeconfig** 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- **oc** CLI를 설치했습니다.

프로세스

1. **kubeadmin** 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 **oc** 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

추가 리소스

- OpenShift Container Platform 웹 콘솔 액세스 및 이해에 대한 자세한 내용은 웹 콘솔에 액세스를 참조하십시오.

7.8.11. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

7.8.12. 다음 단계

- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.

7.9. GCP에 개인 클러스터 설치

OpenShift Container Platform 4.9 버전에서는 GCP(Google Cloud Platform)의 기존 VPC에 프라이빗 클러스터를 설치할 수 있습니다. 설치 프로그램이 나머지 필수 인프라를 프로비저닝하며, 이후에 추가로 사용자 지정할 수 있습니다. 설치를 사용자 지정하려면 클러스터를 설치하기 전에 **install-config.yaml** 파일에서 매개변수를 수정합니다.

7.9.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비 에 대한 문서를 읽습니다.
- 클러스터를 호스팅할 GCP 프로젝트를 구성했습니다.
- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성해야 합니다.
- 사용자의 환경에서 클라우드 ID 및 액세스 관리(IAM) API에 액세스할 수 없거나 **kube-system** 네임스페이스에 관리자 수준의 인증 정보 시크릿을 저장하지 않으려면 IAM 인증 정보 수동 생성 및 유지 관리를 참조하십시오.

7.9.2. 프라이빗 클러스터

외부 엔드 포인트를 노출하지 않는 비공개 OpenShift Container Platform 클러스터를 배포할 수 있습니다. 프라이빗 클러스터는 내부 네트워크에서만 액세스할 수 있으며 인터넷에 표시되지 않습니다.

기본적으로 OpenShift Container Platform은 공개적으로 액세스 가능한 DNS 및 끝점을 사용하여 프로비저닝됩니다. 따라서 개인 클러스터를 배포할 때 클러스터에서 DNS, Ingress Controller 및 API 서버를 비공개로 설정할 수 있습니다. 즉 클러스터 리소스는 내부 네트워크에서만 액세스할 수 있고 인터넷에는 노출되지 않습니다.

중요

클러스터에 퍼블릭 서브넷이 있는 경우 관리자가 생성한 로드 밸런서 서비스에 공개적으로 액세스할 수 있습니다. 클러스터 보안을 보장하기 위해 이러한 서비스에 명시적으로 주석이 지정되었는지 확인합니다.

프라이빗 클러스터를 배포하려면 다음을 수행해야 합니다.

- 요구 사항을 충족하는 기존 네트워킹을 사용합니다. 네트워크의 다른 클러스터 사이에 클러스터 리소스를 공유할 수 있습니다.
- 다음에 액세스할 수 있는 머신에서 배포합니다.
  - 프로비저닝하는 클라우드용 API 서비스
  - 프로비저닝하는 네트워크의 호스트
  - 설치 미디어를 가져올 인터넷

이러한 액세스 요구사항을 충족하고 회사의 지침을 따르는 모든 시스템을 사용할 수 있습니다. 클라우드 네트워크의 배스천 호스트 또는 VPN을 통해 네트워크에 액세스할 수 있는 시스템 등을 예로 들 수 있습니다.

7.9.2.1. GCP의 프라이빗 클러스터

GCP(Google Cloud Platform)에 프라이빗 클러스터를 생성하려면 클러스터를 호스팅할 기존 프라이빗 VPC와 서브넷을 제공해야 합니다. 또한 설치 프로그램에서 클러스터에 필요한 DNS 레코드를 확인할 수 있어야 합니다. 설치 프로그램은 내부 트래픽용 Ingress Operator 및 API 서버를 구성합니다.

클러스터가 GCP API에 액세스하려면 여전히 인터넷 접속이 필요합니다.

다음은 프라이빗 클러스터를 설치할 때 필요하지 않거나 생성되지 않는 항목들입니다.

- 퍼블릭 서브넷
- 퍼블릭 인그레스를 지원하는 퍼블릭 네트워크 로드 밸런서
- 클러스터의 **baseDomain**과 일치하는 퍼블릭 DNS 영역

사용자가 지정하는 **baseDomain**을 사용하여 설치 프로그램에서 프라이빗 DNS 영역과 클러스터에 필요한 레코드를 생성합니다. Operator가 클러스터에 대한 공용 레코드를 생성하지 않고 사용자가 지정하는 프라이빗 서브넷에 모든 클러스터 시스템이 배치되도록 클러스터가 구성됩니다.

소스 태그를 기반으로 외부 로드 밸런서에 액세스를 제한할 수 없기 때문에 프라이빗 클러스터에서는 내부 로드 밸런서만을 사용하여 내부 인스턴스에 액세스를 허용합니다.

내부 로드 밸런서는 네트워크 로드 밸런서가 사용하는 대상 풀이 아닌 인스턴스 그룹에 의존합니다. 해당 그룹에 인스턴스가 없는 경우에도 각 영역에 대한 인스턴스 그룹이 설치 프로그램에 의해 생성됩니다.

- 클러스터 IP 주소는 내부 전용입니다.
- 한 가지 전달 규칙이 Kubernetes API 및 시스템 구성 서버 포트를 모두 관리합니다.
- 백엔드 서비스는 각 영역의 인스턴스 그룹과 부트스트랩 인스턴스 그룹(존재하는 동안)으로 구성됩니다.
- 방화벽에는 내부 소스 범위만을 기반으로 하는 단일 규칙이 사용됩니다.

7.9.2.1.1. 제한

로드 밸런서 기능의 차이로 인해 시스템 구성 서버의 상태 검사 **/healthz**는 실행되지 않습니다. 내부 로드 밸런서 두 개가 단일 IP 주소를 공유할 수는 없지만 네트워크 로드 밸런서 두 개는 하나의 외부 IP 주소를 공유할 수 있습니다. 대신 전적으로 포트 6443의 **/readyz** 검사에 의해 인스턴스 상태가 결정됩니다.

7.9.3. 사용자 지정 VPC 사용 정보

OpenShift Container Platform 4.9에서는 GCP(Google Cloud Platform)의 기존 VPC에 클러스터를 배포할 수 있습니다. 이때 VPC 및 라우팅 규칙 내의 기존 서브넷도 사용해야 합니다.

기존 GCP VPC에 OpenShift Container Platform을 배포하면 새 계정의 한도 제한을 회피하거나 회사의 지침에 따른 운영 제한을 따르기 수월해집니다. 이 방법은 VPC를 직접 생성하는 데 필요한 인프라 생성 권한을 받을 수 없을 때 사용하기 좋은 대안입니다.

7.9.3.1. VPC 사용 요구사항

설치 프로그램에서 다음 구성 요소를 더 이상 생성하지 않습니다:

- VPC
- 서브넷
- 클라우드 라우터
- 클라우드 NAT
- NAT IP 주소

사용자 지정 VPC를 사용하는 경우, 사용할 클러스터와 설치 프로그램에 맞게 VPC와 해당 서브넷을 구성해야 합니다. 설치 프로그램에서 사용할 클러스터의 네트워크 범위를 세분하거나 서브넷에 대한 라우팅 테이블을 설정하거나 DHCP와 같은 VPC 옵션을 설정할 수 없으므로 클러스터를 설치하기 전에 직접 해당 작업을 수행해야 합니다.

사용자 VPC와 서브넷이 다음과 같은 특성을 충족해야 합니다.

- OpenShift Container Platform 클러스터를 배포하는 것과 동일한 GCP 프로젝트에 VPC가 있어야 합니다.
- 컨트롤 플레인 및 컴퓨팅 시스템에서 인터넷에 액세스할 수 있도록 하려면 서브넷에서 클라우드 NAT를 구성하여 이그레스를 허용해야 합니다. 이러한 시스템에는 공용 주소가 없습니다. 인터넷에 액세스할 필요가 없는 경우에도 VPC 네트워크로 이그레스를 허용해야만 설치 프로그램과 이미지를 받을 수 있습니다. 공유 서브넷에 여러 Cloud NAT를 구성할 수 없기 때문에 설치 프로그램에서 구성할 수 없습니다.

제공한 서브넷이 적합한지 확인하기 위해 설치 프로그램이 다음 데이터를 확인합니다.

- 사용자가 지정하는 모든 서브넷이 존재하며 지정한 VPC에 속합니다.
- 서브넷 CIDR이 시스템 CIDR에 속합니다.
- 클러스터 컨트롤 플레인 및 컴퓨팅 시스템을 배포할 서브넷을 제공해야 합니다. 두 시스템 유형 모두에 동일한 서브넷을 사용할 수 있습니다.

기존 VPC를 사용하는 클러스터를 제거해도 VPC는 삭제되지 않습니다.

7.9.3.2. 권한 분할

OpenShift Container Platform 4.3부터 클러스터를 배포하는 데 설치 프로그램에서 프로비저닝한 인프라 클러스터에 필요한 권한 중 일부가 필요하지 않게 되었습니다. 이러한 변경은 회사에서 보유할 수 있는 권한 분할을 모방합니다. 즉, 몇몇 사람이 다른 사람들과 다른 리소스를 클라우드에 생성할 수 있습니다. 예를 들어 인스턴스, 버킷, 로드 밸런서와 같은 애플리케이션 관련 항목을 생성할 수는 있지만 VPC, 서브넷 또는 인그레스 규칙과 같은 네트워킹 관련 구성 요소는 생성할 수 없습니다.

클러스터를 생성할 때 사용하는 GCP 자격 증명에는 서브넷, 라우팅 테이블, 인터넷 게이트웨이, NAT,

VPN과 같은 VPC 내 핵심 네트워킹 구성 요소와 VPC를 생성하는 데 필요한 네트워킹 권한이 필요하지 않습니다. 하지만 로드 밸런서, 보안 그룹, 스토리지 및 노드와 같이 클러스터 내 시스템에 필요한 애플리케이션 리소스를 생성하려면 여전히 권한이 필요합니다.

7.9.3.3. 클러스터 간 격리

기존 네트워크에 OpenShift Container Platform을 배포하는 경우, 클러스터의 인프라 ID로 클러스터의 시스템을 참조하는 방화벽 규칙에 따라 클러스터 서비스의 격리가 유지됩니다. 클러스터 내 트래픽만이 허용됩니다.

동일한 VPC에 여러 클러스터를 배포하는 경우, 다음 구성 요소들이 클러스터 간에 액세스를 공유할 수 있습니다.

- 외부 게시 전략을 통해 전역에서 사용 가능한 또는 내부 게시 전략을 통해 네트워크 범위에서 사용 가능한 API
- SSH 및 ICMP 액세스를 위해 머신 CIDRr에 열려있는 VM 인스턴스의 포트와 같은 디버깅 도구

7.9.4. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.

중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

7.9.5. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 **core** 사용자의 **~/.ssh/authorized_keys** 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 **core**로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. **./openshift-install gather** 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

참고

AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: **~/.ssh/id_ed25519**)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 **'~/.ssh** 디렉터리에 있는지 확인하십시오.

참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 **ed25519** 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 **rsa** 또는 **ecdsa** 알고리즘을 사용하는 키를 생성합니다.

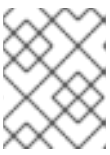
2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 **~/.ssh/id_ed25519.pub** 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 **./openshift-install gather** 명령을 사용하려는 경우 필요합니다.



참고

일부 배포에서는 **~/.ssh/id_rsa** 및 **~/.ssh/id_dsa**와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. **ssh-agent** 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. **ssh-agent**에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: **~/.ssh/id_ed25519**).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

5. **GOOGLE_APPLICATION_CREDENTIALS** 환경 변수를 서비스 계정 개인 키 파일의 전체 경로로 설정합니다.

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

6. 인증 정보가 적용되었는지 확인합니다.

```
$ gcloud auth list
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

7.9.6. 설치 프로그램 받기

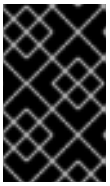
OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

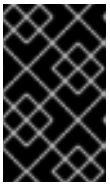
프로세스

1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.
2. 인프라 공급자를 선택합니다.
3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.



중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.



중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 OpenShift Container Platform 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

7.9.7. 수동으로 설치 구성 파일 만들기

내부 네트워크에서만 액세스할 수 있고 인터넷에 표시되지 않는 프라이빗 OpenShift Container Platform 클러스터 설치의 경우 설치 구성 파일을 수동으로 생성해야 합니다.

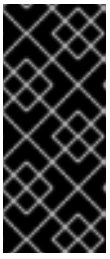
사전 요구 사항

- 로컬 시스템에 설치 프로그램에 제공할 SSH 공개 키가 있습니다. 키는 디버깅 및 재해 복구를 위해 클러스터 노드에 대한 SSH 인증에 사용됩니다.
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿이 있습니다.

프로세스

1. 필요한 설치 자산을 저장할 설치 디렉터리를 만듭니다.

```
$ mkdir <installation_directory>
```



중요

디렉터리를 만들어야 합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

2. 샘플 **install-config.yaml** 파일 템플릿을 사용자 지정하여 **<installation_directory>**에 저장합니다.

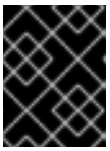
참고

이 설정 파일의 이름을 **install-config.yaml**로 지정해야 합니다.

참고

일부 플랫폼 유형의 경우 대체로 **./openshift-install create install-config --dir <installation_directory>** 를 실행하여 **install-config.yaml** 파일을 생성할 수 있습니다. 프롬프트에서 클러스터 구성에 대한 세부 정보를 제공할 수 있습니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.

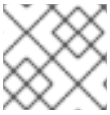


중요

install-config.yaml 파일은 설치 과정의 다음 단계에서 사용됩니다. 이 시점에서 이를 백업해야 합니다.

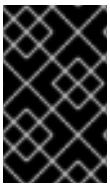
7.9.7.1. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 매개변수 값을 제공하여 클러스터를 호스팅할 클라우드 플랫폼에서 사용자 계정을 설명하고 선택사항으로 클러스터의 플랫폼을 사용자 지정합니다. **install-config.yaml** 설치 구성 파일을 생성할 때 명령줄을 통해 필요한 매개변수 값을 제공합니다. 클러스터를 사용자 지정하면 **install-config.yaml** 파일을 수정하여 플랫폼에 대한 세부 정보를 제공할 수 있습니다.



참고

설치한 후에는 **install-config.yaml** 파일에서 이러한 매개변수를 수정할 수 없습니다.



중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

7.9.7.1.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 7.30. 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).

매개변수	설명	값
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	소문자, 하이픈(**-**), 마침표(**.**)로 구성되는 문자열(예: **dev**)입니다.
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

7.9.7.1.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 7.31. 네트워크 매개변수

매개변수	설명	값

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체 **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.
networking.clusterNetwork	Pod의 IP 주소 블록입니다. 기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된 **cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16

매개변수	설명	값
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16** **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

7.9.7.1.3. 선택적 구성 매개변수

선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 7.32. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열

매개변수	설명	값
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은 **3**입니다.
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열

매개변수	설명	값
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다. GCP에 공유 VPC(가상 프라이빗 클라우드)를 설치하는 경우 **credentialsMode** 를 **Passthrough** 로 설정해야 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 인터넷에서 액세스할 수 없는 사설 클러스터를 배포하려면 **publish**를 **Internal**로 설정합니다. 기본값은 **External**입니다.

매개변수	설명	값

sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다.
	
	참고
	
	설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.
		하나 이상의 키입니다. 예를 들면 다음과 같습니다.
		
		`sshKey:`
		`  <key1>`
		`  <key2>`
		`  <key3>`

7.9.7.1.4. 추가 GCP(Google Cloud Platform) 구성 매개변수

추가 GCP 구성 매개변수는 다음 표에 설명되어 있습니다.

표 7.33. 추가 GCP 매개변수

매개변수	설명	값
platform.gcp.network	클러스터를 배포할 기존 VPC의 이름입니다.	문자열.
platform.gcp.region	클러스터를 호스팅하는 GCP 리전의 이름입니다.	유효한 리전 이름 (예: **us-central1**)
platform.gcp.type	GCP 시스템 유형	GCP 시스템 유형입니다.
platform.gcp.zones	설치 프로그램에서 지정된 MachinePool에 대한 시스템을 생성하는 가용 영역입니다.	**us-central1-a**와 같이 유효한 GCP 가용 영역 목록에 YAML 순서로 나열됩니다.
platform.gcp.controlPlaneSubnet	컨트롤 플레인 시스템을 배포할 VPC의 기존 서브넷 이름	서브넷 이름입니다.

매개변수	설명	값
platform.gcp.computeSubnet	컴퓨팅 시스템을 배포할 VPC의 기존 서브넷 이름	서브넷 이름입니다.
platform.gcp.licenses	컴퓨팅 이미지에 적용해야 하는 라이센스 URL 목록입니다. **중요** **licenses** 매개변수는 더 이상 사용되지 않는 필드이며 중첩된 가상화는 기본적으로 활성화되어 있습니다. 이 필드를 사용하지 않는 것이 좋습니다.	중첩된 가상화 를 활성화하는 라이센스와 같은 라이센스 API 와 함께 사용할 수 있는 모든 라이센스입니다. 사전 빌드된 이미지를 생성하는 메커니즘과 함께 이 매개변수를 사용할 수 없습니다. 라이센스 URL을 사용하면 설치 프로그램이 소스 이미지를 사용하기 전에 복사해야 합니다.
platform.gcp.osDisk.diskSizeGB	디스크 크기(GB)입니다.	16GB와 65536GB 사이의 모든 크기입니다.
platform.gcp.osDisk.diskType	디스크 유형입니다.	기본 **pd-ssd** 또는 **pd-standard** 디스크 유형입니다. 컨트롤 플레인 노드는 **pd-ssd** 디스크 유형이어야 합니다. 작업자 노드는 유형 중 하나일 수 있습니다.
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.name	컨트롤 플레인 머신 디스크 암호화에 사용되는 고객 관리 암호화 키의 이름입니다.	암호화 키 이름입니다.

매개변수	설명	값
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.keyRing	컨트롤 플레인 머신의 경우 KMS 키가 속한 KMS 키 링의 이름입니다.	KMS 키 링의 이름입니다.
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.location	컨트롤 플레인 머신의 경우 키 링이 있는 GCP 위치입니다. KMS 위치에 대한 자세한 내용은 Cloud KMS 위치에 대한 Google의 문서를 참조하십시오.	키 링의 GCP 위치입니다.
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.projectID	컨트롤 플레인 머신의 경우 KMS 키 링이 있는 프로젝트의 ID입니다. 이 값은 설정되지 않은 경우 VM 프로젝트 ID가 기본값입니다.	GCP 프로젝트 ID입니다.
compute.platform.gcp.osDisk.encryptionKey.kmsKey.name	컴퓨팅 머신 디스크 암호화에 사용되는 고객 관리 암호화 키의 이름입니다.	암호화 키 이름입니다.

매개변수	설명	값
compute.platform.gcp.osDisk.encryptionKey.kmsKey.keyRing	컴퓨팅 머신의 경우 KMS 키가 속한 KMS 키 링의 이름입니다.	KMS 키 링의 이름입니다.
compute.platform.gcp.osDisk.encryptionKey.kmsKey.location	컴퓨팅 머신의 경우 키 링이 있는 GCP 위치입니다. KMS 위치에 대한 자세한 내용은 Cloud KMS 위치에 대한 Google의 문서를 참조하십시오.	키 링의 GCP 위치입니다.
compute.platform.gcp.osDisk.encryptionKey.kmsKey.projectID	컴퓨팅 머신의 경우 KMS 키 링이 있는 프로젝트의 ID입니다. 이 값은 설정되지 않은 경우 VM 프로젝트 ID가 기본값입니다.	GCP 프로젝트 ID입니다.

7.9.7.2. 클러스터 설치를 위한 최소 리소스 요구 사항

각 클러스터 시스템이 다음과 같은 최소 요구사항을 충족해야 합니다.

표 7.34. 최소 리소스 요구사항

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
부트스트랩	RHCOS	4	16GB	100GB	300
컨트롤 플레인	RHCOS	4	16GB	100GB	300

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
컴퓨팅	RHCOS, RHEL 7.9 또는 RHEL 8.4 [3]	2	8GB	100GB	300

1. SMT(동시 멀티스레딩) 또는 하이퍼 스레딩이 활성화되지 않은 경우 하나의 vCPU는 하나의 물리적 코어와 동일합니다. 활성화하면 다음과 같은 공식을 사용하여 해당 비율을 계산합니다. (코어당 스레드 수 × 코어 수) × 소켓 수 = vCPU 수

2. OpenShift Container Platform 및 Kubernetes는 디스크 성능에 민감하며 특히 10ms p99 fsync 기간이 필요한 컨트롤 플레인 노드의 etcd에 더 빠른 스토리지가 권장됩니다. 많은 클라우드 플랫폼에서 스토리지 크기와 IOPS를 함께 확장되므로 충분한 성능을 얻으려면 스토리지 볼륨을 과도하게 할당해야 할 수 있습니다.

3. 사용자가 프로비저닝한 모든 설치와 마찬가지로 클러스터에서 RHEL 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. RHEL 7 컴퓨팅 머신 사용은 더 이상 사용되지 않으며 향후 OpenShift Container Platform 4 릴리스에서 제거될 예정입니다.

7.9.7.3. 사용자 정의 머신 유형 사용

OpenShift Container Platform 클러스터를 설치하기 위해 사용자 정의 머신 유형을 사용할 수 있습니다.

사용자 지정 머신 유형을 사용할 때는 다음을 고려하십시오.

- 사전 정의된 인스턴스 유형과 유사하게 사용자 정의 머신 유형은 컨트롤 플레인 및 컴퓨팅 시스템의 최소 리소스 요구 사항을 충족해야 합니다. 자세한 내용은 "클러스터 설치를 위한 최소 리소스 요구 사항"을 참조하십시오.

- 사용자 정의 머신 유형의 이름은 다음 구문을 준수해야 합니다.
  custom-<number_of_cpus>-<amount_of_memory_in_mb>

  예를 들면 **custom-6-20480** 입니다.

설치 프로세스의 일부로 **install-config.yaml** 파일에 사용자 지정 머신 유형을 지정합니다.

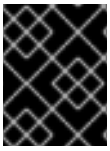
사용자 지정 머신 유형이 있는 샘플 install-config.yaml 파일

```
compute:
- architecture: amd64
  hyperthreading: Enabled
  name: worker
  platform:
    gcp:
      type: custom-6-20480
  replicas: 2
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  platform:
```

```
    gcp:
      type: custom-6-20480
  replicas: 3
```

7.9.7.4. GCP용 샘플 사용자 지정 install-config.yaml 파일

install-config.yaml 파일을 사용자 지정하여 OpenShift Container Platform 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.



중요

이 샘플 YAML 파일은 참조용으로만 제공됩니다. 설치 프로그램을 사용하여 **install-config.yaml** 파일을 받아서 수정해야 합니다.

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2) (3)
  hyperthreading: Enabled (4)
  name: master
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-ssd
        diskSizeGB: 1024
        encryptionKey: (5)
          kmsKey:
            name: worker-key
            keyRing: test-machine-keys
            location: global
            projectID: project-id
  replicas: 3
compute: (6) (7)
- hyperthreading: Enabled (8)
  name: worker
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-standard
        diskSizeGB: 128
        encryptionKey: (9)
          kmsKey:
            name: worker-key
            keyRing: test-machine-keys
            location: global
            projectID: project-id
```

```
  replicas: 3
metadata:
  name: test-cluster 10
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  gcp:
    projectID: openshift-production 11
    region: us-central1 12
    network: existing_vpc 13
    controlPlaneSubnet: control_plane_subnet 14
    computeSubnet: compute_subnet 15
pullSecret: '{"auths": ...}' 16
fips: false 17
sshKey: ssh-ed25519 AAAA... 18
publish: Internal 19
```

1 10 11 12 16 필수 항목입니다. 설치 프로그램에서 이 값을 입력하라는 메시지를 표시합니다.

2 6 이러한 매개변수와 값을 지정하지 않으면 설치 프로그램은 기본값을 적용합니다.

3 7 **controlPlane** 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

4 8 동시 멀티스레딩 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. 일부 클러스터 시스템에서 동시 멀티스레딩을 비활성화할 경우에는 해당 멀티스레딩을 모든 클러스터 시스템에서 비활성화해야 합니다.

중요

동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다. 동시 멀티스레딩을 비활성화하는 경우, 사용자 시스템에 더 큰 시스템 유형(예: **n1-standard-8**)을 사용하십시오.

5 9 선택 사항: 사용자 정의 암호화 키 섹션으로 가상 머신과 영구 볼륨을 모두 암호화합니다. 기본 컴퓨팅 서비스 계정에는 KMS 키를 사용하도록 부여된 권한이 있어야 하며 올바른 IAM 역할을 할당해야 합니다. 기본 서비스 계정 이름은 **service-<project_number>@compute-system.iam.gserviceaccount.com** 패턴을 따릅니다. 서비스 계정에 대한 올바른 권한을 부여하는 방법에 대한 자세한 내용은 "머신 관리" → " 머신 세트 생성" → "GCP에서 머신 세트 생성"을 참조하십시오.

13 기존 VPC의 이름을 지정합니다.

14 컨트롤 플레인 시스템을 배포할 기존 서브넷의 이름을 지정합니다. 지정한 VPC에 속하는 서브넷이어야 합니다.

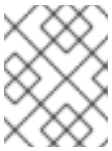
15 컴퓨팅 시스템을 배포할 기존 서브넷의 이름을 지정합니다. 지정한 VPC에 속하는 서브넷이어야 합니다.

17 FIPS 모드 활성화 또는 비활성화 여부입니다. 기본적으로 FIPS 모드는 비활성화됩니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.

중요

FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.

18 선택사항으로, 클러스터의 시스템에 액세스하는 데 사용할 **sshKey** 값을 제공할 수도 있습니다.



참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

19 클러스터의 사용자 끝점을 게시하는 방법. 인터넷에서 액세스할 수 없는 프라이빗 클러스터를 배포하려면 **publish**를 **Internal**로 설정합니다. 기본값은 **External**입니다.

7.9.7.5. GCP에서 글로벌 액세스 권한이 있는 Ingress 컨트롤러 생성

GCP(Google Cloud Platform) 클러스터에 글로벌 액세스할 수 있는 Ingress 컨트롤러를 생성할 수 있습니다. 글로벌 액세스는 내부 로드 밸런서를 사용하는 Ingress 컨트롤러에서만 사용할 수 있습니다.

사전 요구 사항

- **install-config.yaml**을 생성하고 수정 완료했습니다.

절차

새 GCP 클러스터에서 글로벌 액세스 권한이 있는 Ingress 컨트롤러를 생성합니다.

1. 설치 프로그램이 포함된 디렉터리로 변경하고 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>**는 클러스터의 **install-config.yaml** 파일이 포함된 디렉터리의 이름을 지정합니다.

2. **<installation_directory>/manifests/** 디렉터리에 **cluster-ingress-default-ingresscontroller.yaml**이라는 이름으로 파일을 만듭니다.

```
$ touch <installation_directory>/manifests/cluster-ingress-default-ingresscontroller.yaml 1
```

1 **<installation_directory>**는 클러스터의 **manifests** / 디렉터리가 포함된 디렉터리 이름을 지정합니다.

파일이 생성되면 다음과 같이 여러 네트워크 구성 파일이 **manifests/** 디렉토리에 나타납니다.

```
$ ls <installation_directory>/manifests/cluster-ingress-default-ingresscontroller.yaml
```

출력 예

```
cluster-ingress-default-ingresscontroller.yaml
```

3. 편집기에서 **cluster-ingress-default-ingresscontroller.yaml** 파일을 열고 원하는 운영자 구성을 설명하는 CR(사용자 정의 리소스)을 입력합니다.

Global에 대한 clientAccess 구성 샘플

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: default
  namespace: openshift-ingress-operator
spec:
  endpointPublishingStrategy:
    loadBalancer:
      providerParameters:
        gcp:
          clientAccess: Global 1
        type: GCP
      scope: Internal 2
    type: LoadBalancerService
```

1. **gcp.clientAccess**를 **Global**로 설정합니다.
2. 글로벌 액세스는 내부 로드 밸런서를 사용하는 Ingress 컨트롤러에서만 사용할 수 있습니다.

7.9.8. 추가 리소스

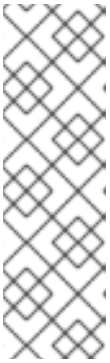
- 머신 세트의 고객 관리 암호화 키 활성화

7.9.8.1. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 HTTP 또는 HTTPS 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 OpenShift Container Platform 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 API에 대한 호출을 포함하여 모든 클러스터 발신(Egress) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.



참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

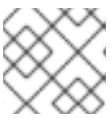
Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 Red Hat OpenStack Platform (RHOSP)에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. 클러스터 외부에서 HTTP 연결을 구축하는 데 사용할 프록시 URL입니다. URL 스키마는 **http**여야 합니다.
2. 클러스터 외부에서 HTTPS 연결을 구축하는 데 사용할 프록시 URL입니다.
3. 대상 도메인 이름, IP 주소 또는 프록시에서 제외할 기타 네트워크 CIDR로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. ***를 사용하여 모든 대상에 대해 프록시를 바이패스합니다.
4. 이 값을 제공하면 설치 프로그램에서 추가 CA 인증서를 보유할 **openshift -config 네임스페이스에 user-ca- bundle** 이라는 구성 맵을 생성합니다. **additionalTrustBundle** 및 하나 이상의 프록시 설정을 제공하는 경우 **프록시** 오브젝트는 **trustedCA** 필드의 **user-ca-bundle** 구성 맵을 참조하도록 구성됩니다. 그러면 Cluster Network Operator에서 **trustedCA** 매개변수에 대해 지정된 콘텐츠를 RHCOS 신뢰 번들과 병합하는 **trusted-ca-bundle** 구성 맵을 생성합니다. 프록시의 ID 인증서를 RHCOS 트러스트 번들에 있는 기관에서 서명하지 않은 경우 **additionalTrustBundle** 필드가 있어야 합니다.

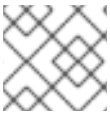


참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 OpenShift Container Platform을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 nil이 됩니다.



참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

7.9.9. 클러스터 배포

호환되는 클라우드 플랫폼에 OpenShift Container Platform을 설치할 수 있습니다.

중요

최초 설치 과정에서 설치 프로그램의 **create cluster** 명령을 한 번만 실행할 수 있습니다.

사전 요구 사항

- 클러스터를 호스팅하는 클라우드 플랫폼으로 계정을 구성합니다.
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

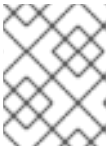
절차

1. 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터 배포를 초기화합니다.

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>**는 다음을 지정합니다.

2 다른 설치 세부 사항을 보려면 **info** 대신 **warn**, **debug** 또는 **error**를 지정합니다.



참고

호스트에 구성된 클라우드 공급자 계정에 클러스터를 배포하기에 충분한 권한이 없는 경우, 설치 프로세스가 중단되고 누락된 권한을 알리는 메시지가 표시됩니다.

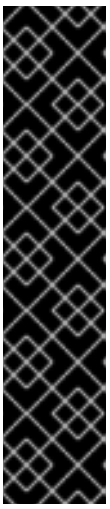
클러스터 배포가 완료되면 웹 콘솔로 연결되는 링크와 **kubeadmin** 사용자의 인증 정보가 포함된 클러스터 액세스 지침이 사용자 터미널에 표시됩니다.

출력 예

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```

참고

설치에 성공하면 클러스터 액세스 및 인증 정보도 **<installation_directory>/.openshift_install.log**로 출력됩니다.



중요

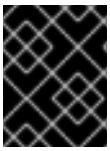
- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

중요

설치 프로그램에서 생성되는 파일이나 설치 프로그램을 삭제해서는 안 됩니다. 클러스터를 삭제하려면 두 가지가 모두 필요합니다.

7.9.10. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift CLI(**oc**)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 **oc**를 설치할 수 있습니다.



중요

이전 버전의 **oc**를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 **oc**를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 Linux에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

절차

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Linux Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. **oc** 바이너리를 **PATH**에 있는 디렉터리에 배치합니다.
PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

-

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치
다음 절차에 따라 Windows에 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

절차

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Windows Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. ZIP 프로그램으로 아카이브의 압축을 풉니다.
5. **oc** 바이너리를 **PATH**에 있는 디렉터리로 이동합니다.
   PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

   ```
   C:\> path
   ```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치
다음 절차에 따라 macOS에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

절차

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 MacOSX Client** 항목 옆에 있는 **지금 다운로드를** 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 해제하고 압축을 풉니다.
5. **oc** 바이너리 PATH의 디렉터리로 이동합니다.
   PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

   ```
   $ echo $PATH
   ```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

7.9.11. CLI를 사용하여 클러스터에 로그인

클러스터 **kubeconfig** 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다.
kubeconfig 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러

스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- **oc** CLI를 설치했습니다.

프로세스

1. **kubeadmin** 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 **oc** 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

추가 리소스

- OpenShift Container Platform 웹 콘솔 액세스 및 이해에 대한 자세한 내용은 웹 콘솔에 액세스를 참조하십시오.

7.9.12. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

7.9.13. 다음 단계

- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.

7.10. DEPLOYMENT MANAGER 템플릿을 사용하여 GCP의 사용자 프로비저닝 인프라에 클러스터 설치

OpenShift Container Platform 4.9 버전에서는 사용자가 제공하는 인프라를 사용하는 클러스터를 GCP(Google Cloud Platform)에 설치할 수 있습니다.

사용자 제공 인프라 설치 단계가 여기에 요약되어 있습니다. 안내된 단계를 수행하거나 자체 모델링에 유용한 몇 가지 Deployment Manager 템플릿이 제공됩니다. 여러 다른 방법을 통해 필요한 리소스를 자유롭게 생성할 수도 있습니다.

중요

사용자가 프로비저닝한 인프라 설치를 수행하는 단계는 예시용으로만 제공됩니다. 사용자가 제공하는 인프라를 사용하여 클러스터를 설치하려면 클라우드 공급자 및 OpenShift Container Platform 설치 프로세스에 대한 정보가 필요합니다. 안내된 단계를 수행하거나 자체 모델링에 유용한 몇 가지 Deployment Manager 템플릿이 제공됩니다. 여러 다른 방법을 통해 필요한 리소스를 자유롭게 생성할 수도 있습니다. 템플릿은 예시일 뿐입니다.

7.10.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비 에 대한 문서를 읽습니다.
- 방화벽을 사용하며 Telemetry 서비스를 사용할 예정인 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성해야 합니다.
- 사용자의 환경에서 클라우드 ID 및 액세스 관리(IAM) API에 액세스할 수 없거나 **kube-system** 네임스페이스에 관리자 수준의 인증 정보 시크릿을 저장하지 않으려면 IAM 인증 정보 수동 생성 및 유지 관리를 참조하십시오.

참고

프록시를 구성하는 경우에도 해당 사이트 목록을 검토하십시오.

7.10.2. 인증서 서명 요청 관리

사용자가 프로비저닝하는 인프라를 사용하는 경우 자동 시스템 관리 기능으로 인해 클러스터의 액세스가 제한되므로 설치한 후 클러스터 인증서 서명 요청(CSR)을 승인하는 메커니즘을 제공해야 합니다. **kube-controller-manager**는 kubelet 클라이언트 CSR만 승인합니다. **machine-approver**는 올바른 시스템에서 발행한 요청인지 확인할 수 없기 때문에 kubelet 자격 증명을 사용하여 요청하는 서비스 인증서의 유효성을 보장할 수 없습니다. kubelet 서빙 인증서 요청의 유효성을 확인하고 요청을 승인하는 방법을 결정하여 구현해야 합니다.

7.10.3. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.

- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.

중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

7.10.4. GCP 프로젝트 구성

OpenShift Container Platform을 설치하려면 먼저 호스팅할 GCP(Google Cloud Platform) 프로젝트를 구성해야 합니다.

7.10.4.1. GCP 프로젝트 생성

OpenShift Container Platform을 설치하려면 클러스터를 호스팅할 GCP(Google Cloud Platform) 계정에 프로젝트를 생성해야 합니다.

프로세스

- OpenShift Container Platform 클러스터를 호스팅할 프로젝트를 생성합니다. GCP 문서의 프로젝트 생성 및 관리 단원을 참조하십시오.

중요

설치 관리자 프로비저닝 인프라를 사용하는 경우 GCP 프로젝트는 Premium Network Service Tier를 사용해야 합니다. 설치 프로그램을 사용하여 설치된 클러스터의 Standard Network Service Tier는 지원되지 않습니다. 설치 프로그램은 **api-int.<cluster_name>.<base_domain>** URL에 대한 내부 로드 밸런싱을 구성합니다.

7.10.4.2. GCP에서 API 서비스 활성화

GCP(Google Cloud Platform) 프로젝트에서 OpenShift Container Platform 설치를 완료하려면 여러 API 서비스에 액세스해야 합니다.

사전 요구 사항

- 클러스터를 호스팅할 프로젝트 생성을 완료했습니다.

프로세스

- 클러스터를 호스팅하는 프로젝트에서 다음과 같은 필수 API 서비스를 활성화합니다. GCP 문서의 서비스 활성화 단원을 참조하십시오.

표 7.35. 필수 API 서비스

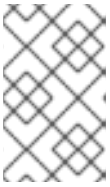
API 서비스	콘솔 서비스 이름
Cloud Deployment Manager V2 API	**deploymentmanager.googleapis.com**
컴퓨팅 엔진 API	**compute.googleapis.com**
Google 클라우드 API	**cloudapis.googleapis.com**
클라우드 리소스 관리자 API	**cloudresourcemanager.googleapis.com**
Google DNS API	**dns.googleapis.com**
IAM 서비스 계정 자격 증명 API	**iamcredentials.googleapis.com**
IAM(ID 및 액세스 관리) API	**iam.googleapis.com**
서비스 관리 API	**servicemanagement.googleapis.com**
서비스 사용량 API	**serviceusage.googleapis.com**
Google 클라우드 스토리지 JSON API	**storage-api.googleapis.com**
클라우드 스토리지	**storage-component.googleapis.com**

7.10.4.3. GCP용 DNS 구성

OpenShift Container Platform을 설치하려면 사용하는 GCP(Google Cloud Platform) 계정에 OpenShift Container Platform 클러스터를 호스팅하는 프로젝트와 동일한 프로젝트에 전용 퍼블릭 호스팅 영역이 있어야 합니다. 도메인에 대한 권한도 이 영역에 있어야 합니다. DNS 서비스는 클러스터와 외부 연결에 필요한 클러스터 DNS 확인 및 이름 조회 기능을 제공합니다.

프로세스

1. 도메인 또는 하위 도메인과 등록 기관을 식별합니다. 기존 도메인 및 등록 기관을 이전하거나 GCP 또는 다른 소스를 통해 새 도메인과 등록 기관을 구할 수 있습니다.



참고

새 도메인을 구입하는 경우, 관련 DNS 변경사항이 전파되는 데 시간이 걸릴 수 있습니다. Google을 통한 도메인 구매에 대한 자세한 내용은 Google 도메인 을 참조하십시오.

2. GCP 프로젝트에서 도메인 또는 하위 도메인의 퍼블릭 호스팅 영역을 생성합니다. GCP 문서의 퍼블릭 영역 생성 단원을 참조하십시오.
적절한 루트 도메인(예: **openshiftcorp.com**) 또는 하위 도메인(예: **clusters.openshiftcorp.com**)을 사용합니다.

3. 호스팅 영역 레코드에서 권한이 있는 새 이름 서버를 추출합니다. GCP 문서의 클라우드 DNS 이름 서버 조회 단원을 참조하십시오.

일반적으로 네 가지 이름 서버가 있습니다.

4. 도메인에서 사용하는 이름 서버의 등록 기관 레코드를 업데이트합니다. 예를 들어, 사용자 도메인을 Google 도메인에 등록한 경우 Google 도메인 도움말의 사용자 지정 이름 서버로 전환하는 방법 항목을 참조하십시오.

5. 루트 도메인을 Google Cloud DNS로 마이그레이션했으면 DNS 레코드를 마이그레이션합니다. GCP 문서의 클라우드 DNS로 마이그레이션을 참조하십시오.

6. 하위 도메인을 사용하는 경우, 회사의 프로시저에 따라 상위 도메인에 위임 레코드를 추가합니다. 이 과정에 회사의 IT 부서 또는 회사의 루트 도메인 및 DNS 서비스를 제어하는 부서에 요청하는 일도 포함될 수 있습니다.

7.10.4.4. GCP 계정 제한

OpenShift Container Platform 클러스터는 여러 GCP(Google Cloud Platform) 구성 요소를 사용하지만 기본 할당량이 기본 OpenShift Container Platform 클러스터 설치가 가능할지 여부에 영향을 미치지 않습니다.

컴퓨팅 및 컨트롤 플레인 시스템 세 개가 포함된 기본 클러스터는 다음과 같은 리소스를 사용합니다. 일부 리소스는 부트스트랩 프로세스 중에만 필요하며 클러스터 배포 후 제거됩니다.

표 7.36. 기본 클러스터에서 사용되는 GCP 리소스

서비스	구성 요소	위치	필요한 총 리소스	부트스트랩 후 제거된 리소스
서비스 계정	IAM	글로벌	5	0
방화벽 규칙	네트워킹	글로벌	11	1
전송 규칙	컴퓨팅	글로벌	2	0
상태 검사	컴퓨팅	글로벌	2	0
이미지	컴퓨팅	글로벌	1	0
네트워크	네트워킹	글로벌	1	0
라우터	네트워킹	글로벌	1	0
라우트	네트워킹	글로벌	2	0
서브네트워크	컴퓨팅	글로벌	2	0
대상 풀	네트워킹	글로벌	2	0

참고

설치하는 동안 할당량이 충분하지 않으면 설치 프로그램에서 초과된 할당량과 리전을 모두 안내하는 오류 메시지를 표시합니다.

실제 클러스터 크기, 예상 클러스터 증가, 계정과 연결된 다른 클러스터의 사용량을 모두 고려해야 합니다. CPU, 고정 IP 주소, 영구 디스크 SSD(스토리지) 할당량이 가장 부족하기 쉬운 할당량입니다.

다음 리전 중 하나에서 클러스터를 배포하려는 경우, 최대 스토리지 할당량을 초과할 것이며, CPU 할당량 제한을 초과할 가능성도 있습니다.

- **asia-east2**
- **asia-northeast2**
- **asia-south1**
- **australia-southeast1**
- **europe-north1**
- **europe-west2**
- **europe-west3**
- **europe-west6**
- **northamerica-northeast1**
- **southamerica-east1**
- **us-west2**

GCP 콘솔에서 리소스 할당량을 늘릴 수는 있지만 지원 티켓을 제출해야 할 수도 있습니다. OpenShift Container Platform 클러스터를 설치하기 전에 지원 티켓을 해결할 시간이 충분하도록 조기에 클러스터 크기를 계획해야 합니다.

7.10.4.5. GCP에서 서비스 계정 생성

OpenShift Container Platform에는 Google API의 데이터에 액세스하기 위한 인증 및 승인을 제공하는 GCP(Google Cloud Platform) 서비스 계정이 필요합니다. 프로젝트에 필요한 역할이 포함된 기존 IAM 서비스 계정이 없으면 새로 생성해야 합니다.

사전 요구 사항

- 클러스터를 호스팅할 프로젝트 생성을 완료했습니다.

프로세스

1. OpenShift Container Platform 클러스터를 호스팅하는 데 사용하는 프로젝트에 서비스 계정을 생성합니다. GCP 문서의 서비스 계정 생성 단원을 참조하십시오.
2. 서비스 계정에 적절한 권한을 부여합니다. 뒤따르는 개별 권한을 부여하거나 **Owner** 역할을 할당할 수 있습니다. 서비스 계정에 특정 리소스에 대한 역할 부여 를 참조하십시오.

참고

서비스 계정을 프로젝트 소유자로 지정하는 것은 가장 쉽게 필요한 권한을 얻는 방법이며, 서비스 계정으로 프로젝트를 완전히 제어할 수 있음을 의미합니다. 해당 권한을 제공하는 데 따른 위험이 수용 가능한 수준인지 확인해봐야 합니다.

3. JSON 형식으로 서비스 계정 키를 생성합니다. GCP 문서의 서비스 계정 키 생성 단원을 참조하십시오.
   클러스터를 생성하기 위해서는 서비스 계정 키가 필요합니다.

7.10.4.5.1. 필요한 GCP 권한

생성하는 서비스 계정에 **Owner** 역할을 연결하면 OpenShift Container Platform 설치에 필요한 권한을 포함하여 모든 권한이 해당 서비스 계정에 부여됩니다. OpenShift Container Platform 클러스터를 배포하려면 서비스 계정에 다음과 같은 권한이 필요합니다. 기존 VPC에 클러스터를 배포할 때는 다음 목록에 제시된 특정 네트워킹 권한이 서비스 계정에 필요하지 않습니다.

설치 프로그램에 필요한 역할

- 컴퓨팅 관리자
- 보안 관리자
- 서비스 계정 관리자
- 서비스 계정 사용자
- 스토리지 관리자

설치 과정에서 네트워크 리소스를 생성하는 데 필요한 역할

- DNS 관리자

사용자 프로비저닝 GCP 인프라에 필요한 역할

- 배포 관리자 편집자
- 서비스 계정 키 관리자

선택적 역할

운영자를 위한 제한적 자격 증명을 새로 생성하는 클러스터의 경우 다음 역할을 추가합니다.

- 서비스 계정 키 관리자

컨트롤 플레인 및 컴퓨팅 시스템이 사용하는 서비스 계정에 적용되는 역할입니다.

표 7.37. GCP 서비스 계정 권한

계정	역할
컨트롤 플레인	**roles/compute.instanceAdmin**
	roles/compute.networkAdmin
	roles/compute.securityAdmin
	roles/storage.admin
	roles/iam.serviceAccountUser

계정	역할
컴퓨팅	**roles/compute.viewer**
	roles/storage.admin

7.10.4.6. 지원되는 GCP 리전

다음과 같은 GCP(Google Cloud Platform) 리전에 OpenShift Container Platform 클러스터를 배포할 수 있습니다.

- **asia-east1** (대만 장화현)
- **asia-east2** (홍콩)
- **asia-northeast1** (일본 도쿄)
- **asia-northeast2** (일본 오사카)
- **asia-northeast3** (한국 서울)
- **asia-south1** (인도 뭄바이)
- **asia-southeast1** (싱가포르 주롱 웨스트)
- **asia-southeast2** (인도네시아 자카르타)
- **australia-southeast1** (호주 시드니)
- **europe-central2** (폴란드 바르샤바)
- **europe-north1** (핀란드 하미나)
- **europe-west1** (벨기에 생기슬랭)
- **europe-west2** (영국 런던)
- **europe-west3** (독일 프랑크푸르트)
- **europe-west4** (네덜란드 엠스하벤)
- **europe-west6** (스위스 취리히)
- **northamerica-northeast1** (캐나다 퀘벡 주 몬트리올)
- **southamerica-east1** (브라질 상파울루)
- **us-central1** (미국 아이오와 주 카운실 블러프스)
- **us-east1** (미국 사우스 캐롤라이나 주 몽크스 코너)
- **us-east4** (미국 노던 버지니아 주 애쉬번)
- **us-west1** (미국 오레곤 주 댈러스)

- **us-west2** (미국 캘리포니아 주 로스앤젤레스)
- **us-west3** (미국 유타 주 솔트레이크시티)
- **us-west4** (미국 네바다 라스베거스)

7.10.4.7. GCP용 CLI 도구 설치 및 구성

사용자 프로비저닝 인프라를 사용하는 OpenShift Container Platform을 GCP(Google Cloud Platform)에 설치하려면 GCP용 CLI 도구를 설치하고 구성해야 합니다.

사전 요구 사항

- 클러스터를 호스팅할 프로젝트 생성을 완료했습니다.
- 서비스 계정 생성과 필요한 권한 부여 작업을 마쳤습니다.

프로세스

1. **$PATH에** 다음 바이너리를 설치합니다.
   - **gcloud**
   - **gsutil**

   GCP 문서의 최신 클라우드 SDK 버전 설치 단원을 참조하십시오.
2. 구성된 서비스 계정과 **gcloud** 도구를 사용하여 인증합니다.
   GCP 문서의 서비스 계정 인증을 참조하십시오.

7.10.5. 사용자 프로비저닝 인프라를 포함한 클러스터의 시스템 요구사항

사용자 프로비저닝 인프라가 포함된 클러스터의 경우, 필요한 모든 시스템을 배포해야 합니다.

이 섹션에서는 사용자 프로비저닝 인프라에 OpenShift Container Platform을 배포해야 하는 요구 사항에 대해 설명합니다.

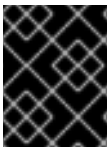
7.10.5.1. 클러스터 설치에 필요한 시스템

최소 OpenShift Container Platform 클러스터에 다음과 같은 호스트가 필요합니다.

표 7.38. 최소 필수 호스트

호스트	설명
임시 부트스트랩 시스템 한 개	컨트롤 플레인 시스템 세 개에 OpenShift Container Platform 클러스터를 배포하기 위한 부트스트랩 시스템이 클러스터에 필요합니다. 클러스터를 설치한 후 부트스트랩 시스템을 제거할 수 있습니다.
컨트롤 플레인 시스템 세 개	컨트롤 플레인 시스템은 컨트롤 플레인을 구성하는 Kubernetes 및 OpenShift Container Platform 서비스를 실행합니다.

호스트	설명
두 개 이상의 컴퓨팅 시스템(작업자 시스템이라고도 함).	OpenShift Container Platform 사용자가 요청한 워크로드는 컴퓨팅 머신에서 실행됩니다.



중요

클러스터의 고가용성을 유지하려면 이러한 클러스터 시스템에 대해 별도의 물리적 호스트를 사용하십시오.

부트스트랩, 컨트롤 플레인 시스템은 운영 체제로 RHCOS (Red Hat Enterprise Linux CoreOS)를 사용해야 합니다. 그러나 컴퓨팅 머신은 RHCOS(Red Hat Enterprise Linux CoreOS), RHEL (Red Hat Enterprise Linux) 7.9 또는 RHEL 8.4 중에서 선택할 수 있습니다.

RHCOS는 RHEL 8(Red Hat Enterprise Linux)을 기반으로하며 모든 하드웨어 인증 및 요구사항을 모두 이어받습니다. Red Hat Enterprise Linux 기술 기능 및 제한 을 참조하십시오.

7.10.5.2. 클러스터 설치를 위한 최소 리소스 요구 사항

각 클러스터 시스템이 다음과 같은 최소 요구사항을 충족해야 합니다.

표 7.39. 최소 리소스 요구사항

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
부트스트랩	RHCOS	4	16GB	100GB	300
컨트롤 플레인	RHCOS	4	16GB	100GB	300
컴퓨팅	RHCOS, RHEL 7.9 또는 RHEL 8.4 [3]	2	8GB	100GB	300

1. SMT(동시 멀티스레딩) 또는 하이퍼 스레딩이 활성화되지 않은 경우 하나의 vCPU는 하나의 물리적 코어와 동일합니다. 활성화하면 다음과 같은 공식을 사용하여 해당 비율을 계산합니다. (코어당 스레드 수 × 코어 수) × 소켓 수 = vCPU 수

2. OpenShift Container Platform 및 Kubernetes는 디스크 성능에 민감하며 특히 10ms p99 fsync 기간이 필요한 컨트롤 플레인 노드의 etcd에 더 빠른 스토리지가 권장됩니다. 많은 클라우드 플랫폼에서 스토리지 크기와 IOPS를 함께 확장되므로 충분한 성능을 얻으려면 스토리지 볼륨을 과도하게 할당해야 할 수 있습니다.

3. 사용자가 프로비저닝한 모든 설치와 마찬가지로 클러스터에서 RHEL 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. RHEL 7 컴퓨팅 머신 사용은 더 이상 사용되지 않으며 향후 OpenShift Container Platform 4 릴리스에서 제거될 예정입니다.

7.10.5.3. 사용자 정의 머신 유형 사용

OpenShift Container Platform 클러스터를 설치하기 위해 사용자 정의 머신 유형을 사용할 수 있습니다.

사용자 지정 머신 유형을 사용할 때는 다음을 고려하십시오.

- 사전 정의된 인스턴스 유형과 유사하게 사용자 정의 머신 유형은 컨트롤 플레인 및 컴퓨팅 시스템의 최소 리소스 요구 사항을 충족해야 합니다. 자세한 내용은 "클러스터 설치를 위한 최소 리소스 요구 사항"을 참조하십시오.
- 사용자 정의 머신 유형의 이름은 다음 구문을 준수해야 합니다.
  custom-<number_of_cpus>-<amount_of_memory_in_mb>

  예를 들면 **custom-6-20480** 입니다.

7.10.6. GCP용 설치 파일 생성

사용자 프로비저닝 인프라를 사용하는 OpenShift Container Platform을 GCP(Google Cloud Platform)에 설치하려면 설치 프로그램에서 클러스터를 배포하는 데 필요한 파일을 생성하고 클러스터가 사용할 시스템만을 생성하도록 파일을 수정해야 합니다. **install-config.yaml** 파일, Kubernetes 매니페스트 및 Ignition 구성 파일을 생성하고 사용자 지정합니다. 또한 설치 준비 단계에서 별도의 **var** 파티션을 먼저 설정할 수 있는 옵션이 있습니다.

7.10.6.1. 선택 사항: 별도의 /var 파티션 만들기

OpenShift Container Platform의 디스크 파티션 설정은 설치 프로그램에 맡기는 것이 좋습니다. 그러나 확장하려는 파일 시스템의 일부에 별도의 파티션을 생성해야 하는 경우가 있습니다.

OpenShift 컨테이너 플랫폼은 **/var** 파티션 또는 **/var**의 하위 디렉터리 중 하나에 스토리지를 연결하는 단일 파티션의 추가를 지원합니다. 예를 들면 다음과 같습니다.

- **/var/lib/containers**: 시스템에 더 많은 이미지와 컨테이너가 추가됨에 따라 확장될 수 있는 컨테이너 관련 콘텐츠를 보관합니다.
- **/var/lib/etcd**: etcd 스토리지의 성능 최적화와 같은 목적으로 별도로 보관할 데이터를 보관합니다.
- **/var**: 감사 등의 목적에 맞게 별도로 분리하여 보관해야 하는 데이터를 보관합니다.

/var 디렉터리의 콘텐츠를 별도로 저장하면 필요에 따라 해당 영역에 대한 스토리지 확장을 보다 용이하게 하고 나중에 OpenShift Container Platform을 다시 설치하여 해당 데이터를 그대로 보존할 수 있습니다. 이 방법을 사용하면 모든 컨테이너를 다시 가져올 필요가 없으며 시스템을 업데이트할 때 대용량 로그 파일을 복사할 필요도 없습니다.

RHCOS(Red Hat Enterprise Linux CoreOS)를 새로 설치하기 전에 **/var가 있어야 하므로 다음 절차에서는 OpenShift Container Platform 설치의 openshift-install 준비 단계 중에 삽입되는 머신 구성 매니페스트를 생성하여 별도의 /var** 파티션을 설정합니다.

중요

이 절차에서 별도의 **/var** 파티션을 생성하기 위해 단계에 따라 이 섹션의 뒷부분에서 설명한 대로 Kubernetes 매니페스트 및 Ignition 구성 파일을 다시 생성할 필요가 없습니다.

프로세스

1. OpenShift Container Platform 설치 파일을 저장할 디렉터리를 만듭니다.

```
$ mkdir $HOME/clusterconfig
```

2. **openshift-install**을 실행하여 **manifest** 및 **openshift** 하위 디렉터리에 파일 세트를 만듭니다. 프롬프트가 표시되면 시스템 질문에 대답합니다.

```
$ openshift-install create manifests --dir $HOME/clusterconfig
```

출력 예

```
? SSH Public Key ...
INFO Credentials loaded from the "myprofile" profile in file "/home/myuser/.aws/credentials"
INFO Consuming Install Config from target directory
INFO Manifests created in: $HOME/clusterconfig/manifests and
$HOME/clusterconfig/openshift
```

3. 선택 사항: 설치 프로그램이 **clusterconfig/openshift** 디렉터리에 매니페스트를 생성했는지 확인합니다.

```
$ ls $HOME/clusterconfig/openshift/
```

출력 예

```
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

4. 추가 파티션을 구성하는 Butane 구성을 생성합니다. 예를 들어 파일 이름을 **$HOME/clusterconfig/98-var-partition.bu** 로 지정하고, 디스크 장치 이름을 **worker** 시스템의 스토리지 장치 이름으로 변경하고 스토리지 크기를 적절하게 설정합니다. 이 예에서는 **/var** 디렉토리를 별도의 파티션에 배치합니다.

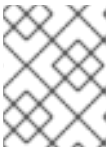
```
variant: openshift
version: 4.9.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1 파티션을 설정해야하는 디스크 저장 장치 이름입니다.

2 데이터 파티션을 부트 디스크에 추가할 때 최소 25000MiB(메비 바이트)가 권장됩니다. 루트 파일 시스템은 지정된 오프셋까지 사용 가능한 모든 공간을 채우기 위해 자동으로 크기가

3 데이터 파티션의 크기(MB)입니다.

4 컨테이너 스토리지에 사용되는 파일 시스템에 **prjquota** 마운트 옵션을 활성화해야 합니다.



참고

별도의 **/var** 파티션을 만들 때 다른 인스턴스 유형에 동일한 장치 이름이 없는 경우 작업자 노드에 다른 인스턴스 유형을 사용할 수 없습니다.

5. Butane 구성에서 매니페스트를 생성하여 **clusterconfig/openshift** 디렉터리에 저장합니다. 예를 들어 다음 명령을 실행합니다.

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

6. **openshift-install**을 다시 실행하여 **manifest** 및 **openshift** 하위 디렉터리의 파일 세트에서 Ignition 구성을 만듭니다.

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

이제 Ignition 구성 파일을 설치 절차에 대한 입력으로 사용하여 RHCOS (Red Hat Enterprise Linux CoreOS) 시스템을 설치할 수 있습니다.

7.10.6.2. 설치 구성 파일 만들기

GCP(Google Cloud Platform)에 설치하는 OpenShift Container Platform 클러스터를 사용자 지정할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.
- 서브스크립션 수준에서 서비스 주체 권한을 획득합니다.

프로세스

1. **install-config.yaml** 파일을 생성합니다.
   a. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 **<installation_directory>**는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.

중요

비어 있는 디렉터리를 지정합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

b. 화면에 나타나는 지시에 따라 클라우드에 대한 구성 세부 사항을 입력합니다.

i. 선택사항: 클러스터 시스템에 액세스하는 데 사용할 SSH 키를 선택합니다.

참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

ii. 대상 플랫폼으로 **gcp**를 선택합니다.

iii. 사용자 컴퓨터에 GCP 계정의 서비스 계정 키가 구성되어 있지 않은 경우, GCP에서 해당 키를 가져와 파일의 내용을 붙여 넣거나 파일의 전체 경로를 입력해야 합니다.

iv. 클러스터를 프로비저닝할 프로젝트 ID를 선택합니다. 구성한 서비스 계정에 의해 기본값이 지정됩니다.

v. 클러스터를 배포할 리전을 선택합니다.

vi. 클러스터를 배포할 기본 도메인을 선택합니다. 기본 도메인은 클러스터용으로 생성한 퍼블릭 DNS 영역에 해당합니다.

vii. 클러스터를 설명할 수 있는 이름을 입력합니다.

viii. Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 붙여넣습니다.

c. 선택사항: 클러스터가 컴퓨팅 시스템을 프로비저닝하지 않도록 하려면 **compute** 풀에 대해 **replicas**를 **0**으로 설정하도록 결과 **install-config.yaml** 파일을 편집하여 컴퓨팅 풀을 비우십시오.

```
compute:
- hyperthreading: Enabled
  name: worker
  platform: {}
  replicas: 0 (1)
```

(1) **0**으로 설정합니다.

2. **install-config.yaml** 파일을 수정합니다. 사용 가능한 매개변수에 대한 자세한 정보는 "설치 구성 매개변수" 섹션에서 확인할 수 있습니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.

중요

install-config.yaml 파일은 설치 과정에서 사용됩니다. 이 파일을 재사용하려면 지금 백업해야 합니다.

7.10.6.3. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 HTTP 또는 HTTPS 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 OpenShift Container Platform 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 API에 대한 호출을 포함하여 모든 클러스터 발신(Egress) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.

참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 Red Hat OpenStack Platform (RHOSP)에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. 클러스터 외부에서 HTTP 연결을 구축하는 데 사용할 프록시 URL입니다. URL 스키마는 **http**여야 합니다.
2. 클러스터 외부에서 HTTPS 연결을 구축하는 데 사용할 프록시 URL입니다.
3. 대상 도메인 이름, IP 주소 또는 프록시에서 제외할 기타 네트워크 CIDR로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. *를 사용하여 모든 대상에 대

해 프록시를 바이패스합니다.

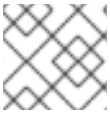
4 이 값을 제공하면 설치 프로그램에서 추가 CA 인증서를 보유할 **openshift -config 네임스페이스에 user-ca- bundle** 이라는 구성 맵을 생성합니다. **additionalTrustBundle** 및 하나 이상의 프록시 설정을 제공하는 경우 **프록시** 오브젝트는 **trustedCA** 필드의 **user-ca-bundle** 구성 맵을 참조하도록 구성됩니다. 그러면 Cluster Network Operator에서 **trustedCA** 매개변수에 대해 지정된 콘텐츠를 RHCOS 신뢰 번들과 병합하는 **trusted-ca-bundle** 구성 맵을 생성합니다. 프록시의 ID 인증서를 RHCOS 트러스트 번들에 있는 기관에서 서명하지 않은 경우 **additionalTrustBundle** 필드가 있어야 합니다.

참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 OpenShift Container Platform을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 nil이 됩니다.



참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

7.10.6.4. Kubernetes 매니페스트 및 Ignition 설정 파일 생성

일부 클러스터 정의 파일을 수정하고 클러스터 시스템을 수동으로 시작해야 하므로 클러스터가 시스템을 구성하는 데 필요한 Kubernetes 매니페스트 및 Ignition 구성 파일을 사용자가 생성해야 합니다.

설치 구성 파일은 Kubernetes 매니페스트로 변환됩니다. 매니페스트는 나중에 클러스터 머신을 구성하는 데 사용되는 Ignition 구성 파일로 래핑됩니다.

중요

- OpenShift Container Platform 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램을 가져오셨습니다.
- **install-config.yaml** 설치 구성 파일을 생성하셨습니다.

절차

절차

1. OpenShift Container Platform 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터에 대한 Kubernetes 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>**는 사용자가 만든 **install-config.yaml** 파일이 포함된 설치 디렉터리를 지정합니다.

2. 컨트롤 플레인 시스템을 정의하는 Kubernetes 매니페스트 파일을 제거합니다.

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

이러한 파일을 제거하면 클러스터가 컨트롤 플레인 시스템을 자동으로 생성하지 못합니다.

3. 선택사항: 클러스터가 컴퓨팅 시스템을 프로비저닝하지 않도록 하려면 작업자 시스템을 정의하는 Kubernetes 매니페스트 파일을 제거하십시오.

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

작업자 시스템은 사용자가 직접 생성하고 관리하기 때문에 초기화할 필요가 없습니다.

4. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes 매니페스트 파일의 **mastersSchedulable** 매개변수가 **false**로 설정되어 있는지 확인합니다. 이 설정으로 인해 컨트롤 플레인 머신에서 포드가 예약되지 않습니다.

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** 파일을 엽니다.

   b. **mastersSchedulable** 매개변수를 찾아서 값을 **False**로 설정되어 있는지 확인합니다.

   c. 파일을 저장하고 종료합니다.

5. 선택사항: Ingress Operator가 사용자 대신 DNS 레코드를 생성하지 못하도록 하려면 **<installation_directory>/manifests/cluster-dns-02-config.yml** DNS 구성 파일에서 **privateZone** 및 **publicZone** 섹션을 제거합니다.

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

1 2 이 섹션을 완전히 제거합니다.

제거한 경우 나중에 인그레스 DNS 레코드를 수동으로 추가해야 합니다.

6. Ignition 구성 파일을 생성하려면 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행합니다.

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>**는 동일한 설치 디렉터리를 지정합니다.

설치 디렉터리의 부트스트랩, 컨트롤 플레인 및 컴퓨팅 노드에 대한 Ignition 구성 파일이 생성됩니다. **kubeadmin-password** 및 **kubeconfig** 파일은 **./<installation_directory>/auth** 디렉터리에 생성됩니다.

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

추가 리소스

- 선택사항 : 인그레스 DNS 레코드 추가

7.10.7. 공통 변수 내보내기

7.10.7.1. 인프라 이름 추출

Ignition 구성 파일에는 GCP(Google Cloud Platform)에서 클러스터를 고유하게 식별하는 데 사용할 수 있는 고유한 클러스터 ID가 포함되어 있습니다. 인프라 이름은 OpenShift Container Platform 설치 중에 적절한 GCP 리소스를 찾는 데도 사용됩니다. 제공된 Deployment Manager 템플릿에 이 인프라 이름에 대한 참조가 포함되어 있으므로 이름을 추출해야 합니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받으셨습니다.
- 클러스터에 대한 Ignition 구성 파일을 생성하셨습니다.
- **jq** CLI를 설치하셨습니다.

절차

- Ignition 구성 파일 메타데이터에서 인프라 이름을 추출하여 확인하려면 다음 명령을 실행하십시오.

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

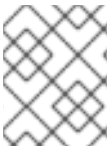
출력 예

```
openshift-vw9j6 1
```

1 이 명령의 출력은 클러스터 이름과 임의의 문자열입니다.

7.10.7.2. Deployment Manager 템플릿의 공통 변수 내보내기

GCP(Google Cloud Platform)에서 사용자 제공 인프라 설치를 지원하기 위해 제공된 Deployment Manager 템플릿과 함께 사용되는 공통 변수 세트를 내보내야 합니다.



참고

특정 Deployment Manager 템플릿에는 추가적으로 내보낸 변수도 필요할 수 있습니다. 관련 프로시저에서 자세히 설명합니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.
- **jq** 패키지를 설치합니다.

프로세스

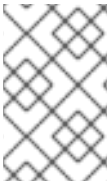
1. 제공된 Deployment Manager 템플릿에서 사용할 다음 공통 변수를 내보냅니다.

```
$ export BASE_DOMAIN='<base_domain>'
$ export BASE_DOMAIN_ZONE_NAME='<base_domain_zone_name>'
$ export NETWORK_CIDR='10.0.0.0/16'
$ export MASTER_SUBNET_CIDR='10.0.0.0/17'
$ export WORKER_SUBNET_CIDR='10.0.128.0/17'

$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
$ export CLUSTER_NAME=`jq -r .clusterName <installation_directory>/metadata.json`
$ export INFRA_ID=`jq -r .infraID <installation_directory>/metadata.json`
$ export PROJECT_NAME=`jq -r .gcp.projectID <installation_directory>/metadata.json`
$ export REGION=`jq -r .gcp.region <installation_directory>/metadata.json`
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

7.10.8. GCP에서 VPC 생성

OpenShift Container Platform 클러스터에서 사용할 VPC를 GCP(Google Cloud Platform)에 생성해야 합니다. 요구사항에 맞춰 VPC를 사용자 지정할 수 있습니다. VPC를 생성하는 한 가지 방법은 제공된 Deployment Manager 템플릿을 수정하는 것입니다.



참고

GCP 인프라를 생성하는 데 제공된 Deployment Manager 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- GCP 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.

프로세스

1. 이 항목의 **VPC에 대한 Deployment Manager 템플릿** 섹션에서 템플릿을 복사하여 사용자 컴퓨터에 **01_vpc.py**로 저장합니다. 이 템플릿은 클러스터에 필요한 VPC를 설명합니다.
2. **01_vpc.yaml** 리소스 정의 파일을 생성합니다.

```
$ cat <<EOF >01_vpc.yaml
imports:
- path: 01_vpc.py

resources:
- name: cluster-vpc
  type: 01_vpc.py
  properties:
    infra_id: '${INFRA_ID}' 1
    region: '${REGION}' 2
    master_subnet_cidr: '${MASTER_SUBNET_CIDR}' 3
    worker_subnet_cidr: '${WORKER_SUBNET_CIDR}' 4
EOF
```

1. **infra_id**는 추출 단계에서 가져온 **INFRA_ID** 인프라 이름입니다.
2. **region**은 클러스터를 배포할 영역입니다(예: **us-central1**).
3. **master_subnet_cidr**은 마스터 서브넷의 CIDR입니다. (예: **10.0.0.0/17**)
4. **worker_subnet_cidr**은 작업자 서브넷의 CIDR입니다. (예: **10.0.128.0/17**)

3. **gcloud** CLI를 사용하여 배포를 생성합니다.

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-vpc --config 01_vpc.yaml
```

7.10.8.1. VPC용 Deployment Manager 템플릿

다음 Deployment Manager 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 VPC를 배포할 수 있습니다.

예 7.1. 01_vpc.py Deployment Manager 템플릿

```
def GenerateConfig(context):

  resources = [{
    'name': context.properties['infra_id'] + '-network',
    'type': 'compute.v1.network',
    'properties': {
        'region': context.properties['region'],
        'autoCreateSubnetworks': False
    }
```

```
    }, {
        'name': context.properties['infra_id'] + '-master-subnet',
        'type': 'compute.v1.subnetwork',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'ipCidrRange': context.properties['master_subnet_cidr']
        }
    }, {
        'name': context.properties['infra_id'] + '-worker-subnet',
        'type': 'compute.v1.subnetwork',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'ipCidrRange': context.properties['worker_subnet_cidr']
        }
    }, {
        'name': context.properties['infra_id'] + '-router',
        'type': 'compute.v1.router',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'nats': [{
                'name': context.properties['infra_id'] + '-nat-master',
                'natIpAllocateOption': 'AUTO_ONLY',
                'minPortsPerVm': 7168,
                'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
                'subnetworks': [{
                    'name': '$(ref.' + context.properties['infra_id'] + '-master-subnet.selfLink)',
                    'sourceIpRangesToNat': ['ALL_IP_RANGES']
                }]
            }, {
                'name': context.properties['infra_id'] + '-nat-worker',
                'natIpAllocateOption': 'AUTO_ONLY',
                'minPortsPerVm': 512,
                'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
                'subnetworks': [{
                    'name': '$(ref.' + context.properties['infra_id'] + '-worker-subnet.selfLink)',
                    'sourceIpRangesToNat': ['ALL_IP_RANGES']
                }]
            }]
        }
    }]

    return {'resources': resources}
```

7.10.9. 사용자 프로비저닝 인프라에 대한 네트워킹 요구사항

모든 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템이 부팅 중에 Ignition 구성 파일을 가져오려면 **initramfs**에 네트워킹을 구성해야 합니다.

7.10.9.1. DHCP를 통해 클러스터 노드의 호스트 이름 설정

RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 호스트 이름은 NetworkManager를 통해 설정됩

니다. 기본적으로 시스템은 DHCP를 통해 호스트 이름을 가져옵니다. DHCP에서 호스트 이름을 제공하지 않거나, 커널 인수를 통해 정적으로 설정하거나 다른 방법을 통해 설정하면 역방향 DNS 조회를 통해 얻을 수 있습니다. 역방향 DNS 조회는 노드에서 네트워크를 초기화한 후 수행되며 확인하는 데 시간이 걸릴 수 있습니다. 다른 시스템 서비스는 이 보다 먼저 시작하여 호스트 이름을 **localhost** 등으로 감지할 수 있습니다. DHCP를 사용하여 각 클러스터 노드의 호스트 이름을 제공하여 이 문제를 방지할 수 있습니다.

또한 DHCP를 통해 호스트 이름을 설정하면 DNS 분할 수평 구현 환경에서 수동으로 DNS 레코드 이름 구성 오류를 무시할 수 있습니다.

7.10.9.2. 네트워크 연결 요구사항

OpenShift Container Platform 클러스터 구성 요소가 통신할 수 있도록 시스템 간 네트워크 연결을 구성해야 합니다. 각 시스템에서 클러스터에 있는 다른 모든 시스템의 호스트 이름을 확인할 수 있어야 합니다.

이 섹션에서는 필요한 포트에 대해 자세히 설명합니다.

중요

연결된 OpenShift Container Platform 환경에서 모든 노드는 플랫폼 컨테이너의 이미지를 가져오고 Red Hat에 원격 측정 데이터를 제공하기 위해 인터넷에 액세스할 수 있어야 합니다.

표 7.40. 모든 시스템 간 통신에 사용되는 포트

프로토콜	포트	설명
ICMP	해당 없음	네트워크 연결성 테스트
TCP	**1936**	메트릭
	9000-9999	**9100-9101** 포트의 노드 내보내기 및 **9099** 포트의 Cluster Version Operator를 포함한 호스트 수준 서비스.
	10250-10259	Kubernetes에서 예약하는 기본 포트
	10256	openshift-sdn
UDP	**4789**	VXLAN 및 Geneve
	6081	VXLAN 및 Geneve
	9000-9999	**9100-9101** 포트의 노드 내보내기를 포함한 호스트 수준 서비스.
	500	IPsec IKE 패킷
	4500	IPsec NAT-T 패킷
TCP/UDP	**30000-32767**	Kubernetes 노드 포트

프로토콜	포트	설명
ESP	해당 없음	IPsec Encapsulating Security Payload (ESP)

표 7.41. 모든 시스템과 컨트롤 플레인 간 통신에 사용되는 포트

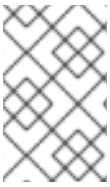
프로토콜	포트	설명
TCP	**6443**	Kubernetes API

표 7.42. 컨트롤 플레인 머신 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**2379-2380**	etcd 서버 및 피어 포트

7.10.10. GCP에서 로드 밸런서 생성

OpenShift Container Platform 클러스터가 사용할 로드 밸런서를 GCP(Google Cloud Platform)에 구성해야 합니다. 이러한 구성 요소를 생성하는 한 가지 방법은 제공된 Deployment Manager 템플릿을 수정하는 것입니다.



참고

GCP 인프라를 생성하는 데 제공된 Deployment Manager 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- GCP 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.
- GCP에서 VPC 및 관련 서브넷을 생성하고 구성하십시오.

프로세스

1. 이 항목의 **내부 로드 밸런서에 대한 Deployment Manager 템플릿** 섹션에서 템플릿을 복사하여 사용자 컴퓨터에 **02_lb_int.py**로 저장합니다. 이 템플릿에서 클러스터에 필요한 내부 로드 밸런싱 개체를 설명합니다.
2. 외부 클러스터에 대해서도, 이 항목의 **외부 로드 밸런서에 대한 Deployment Manager 템플릿** 섹션에서 템플릿을 복사하여 사용자 컴퓨터에 **02_lb_ext.py**로 저장합니다. 이 템플릿에서 클러스터에 필요한 외부 로드 밸런싱 개체를 설명합니다.
3. 배포 템플릿이 사용하는 변수를 내보냅니다.
   a. 클러스터 네트워크 위치를 내보냅니다.

```
$ export CLUSTER_NETWORK=(`gcloud compute networks describe ${INFRA_ID}-
network --format json | jq -r .selfLink`)
```

b. 컨트롤 플레인 서브넷 위치를 내보냅니다.

```
$ export CONTROL_SUBNET=(`gcloud compute networks subnets describe
${INFRA_ID}-master-subnet --region=${REGION} --format json | jq -r .selfLink`)
```

c. 클러스터가 사용하는 세 영역을 내보냅니다.

```
$ export ZONE_0=(`gcloud compute regions describe ${REGION} --format=json | jq -r
.zones[0] | cut -d "/" -f9`)
```

```
$ export ZONE_1=(`gcloud compute regions describe ${REGION} --format=json | jq -r
.zones[1] | cut -d "/" -f9`)
```

```
$ export ZONE_2=(`gcloud compute regions describe ${REGION} --format=json | jq -r
.zones[2] | cut -d "/" -f9`)
```

4. **02_infra.yaml** 리소스 정의 파일을 생성합니다.

```
$ cat <<EOF >02_infra.yaml
imports:
- path: 02_lb_ext.py
- path: 02_lb_int.py (1)
resources:
- name: cluster-lb-ext (2)
  type: 02_lb_ext.py
  properties:
    infra_id: '${INFRA_ID}' (3)
    region: '${REGION}' (4)
- name: cluster-lb-int
  type: 02_lb_int.py
  properties:
    cluster_network: '${CLUSTER_NETWORK}'
    control_subnet: '${CONTROL_SUBNET}' (5)
    infra_id: '${INFRA_ID}'
    region: '${REGION}'
    zones: (6)
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'
EOF
```

(1) (2) 외부 클러스터를 배포하는 경우에만 필요합니다.

(3) **infra_id**는 추출 단계에서 가져온 **INFRA_ID** 인프라 이름입니다.

(4) **region**은 클러스터를 배포할 영역입니다(예: **us-central1**).

(5) **control_subnet**은 컨트롤 서브넷에 대한 URI입니다.

(6) **zones**는 **us-east1-b**, **us-east1-c** 및 **us-east1-d**와 같이 컨트롤 플레인 인스턴스를 배포하는

5. **gcloud** CLI를 사용하여 배포를 생성합니다.

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-infra --config 02_infra.yaml
```

6. 클러스터 IP 주소를 내보냅니다.

```
$ export CLUSTER_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-ip --
region=${REGION} --format json | jq -r .address`)
```

7. 외부 클러스터의 경우 클러스터 공용 IP 주소도 내보냅니다.

```
$ export CLUSTER_PUBLIC_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-
public-ip --region=${REGION} --format json | jq -r .address`)
```

7.10.10.1. 외부 로드 밸런서에 대한 Deployment Manager 템플릿

다음 Deployment Manager 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 외부 로드 밸런서를 배포할 수 있습니다.

예 7.2. 02_lb_ext.py Deployment Manager 템플릿

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-cluster-public-ip',
        'type': 'compute.v1.address',
        'properties': {
            'region': context.properties['region']
        }
    }, {
        # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check
probe for kube-apiserver
        'name': context.properties['infra_id'] + '-api-http-health-check',
        'type': 'compute.v1.httpHealthCheck',
        'properties': {
            'port': 6080,
            'requestPath': '/readyz'
        }
    }, {
        'name': context.properties['infra_id'] + '-api-target-pool',
        'type': 'compute.v1.targetPool',
        'properties': {
            'region': context.properties['region'],
            'healthChecks': ['$(ref.' + context.properties['infra_id'] + '-api-http-health-check.selfLink)'],
            'instances': []
        }
    }, {
        'name': context.properties['infra_id'] + '-api-forwarding-rule',
        'type': 'compute.v1.forwardingRule',
        'properties': {
            'region': context.properties['region'],
            'IPAddress': '$(ref.' + context.properties['infra_id'] + '-cluster-public-ip.selfLink)',
            'target': '$(ref.' + context.properties['infra_id'] + '-api-target-pool.selfLink)',
            'portRange': '6443'
```

```
        }
    }]

    return {'resources': resources}
```

7.10.10.2. 내부 로드 밸런서에 대한 Deployment Manager 템플릿

다음 Deployment Manager 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 내부 로드 밸런서를 배포할 수 있습니다.

예 7.3.02_lb_int.py Deployment Manager 템플릿

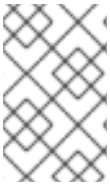
```
def GenerateConfig(context):

    backends = []
    for zone in context.properties['zones']:
        backends.append({
            'group': '$(ref.' + context.properties['infra_id'] + '-master-' + zone + '-instance-group' +
'.selfLink)'
        })

    resources = [{
        'name': context.properties['infra_id'] + '-cluster-ip',
        'type': 'compute.v1.address',
        'properties': {
            'addressType': 'INTERNAL',
            'region': context.properties['region'],
            'subnetwork': context.properties['control_subnet']
        }
    }, {
        # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check
probe for kube-apiserver
        'name': context.properties['infra_id'] + '-api-internal-health-check',
        'type': 'compute.v1.healthCheck',
        'properties': {
            'httpsHealthCheck': {
                'port': 6443,
                'requestPath': '/readyz'
            },
            'type': "HTTPS"
        }
    }, {
        'name': context.properties['infra_id'] + '-api-internal-backend-service',
        'type': 'compute.v1.regionBackendService',
        'properties': {
            'backends': backends,
            'healthChecks': ['$(ref.' + context.properties['infra_id'] + '-api-internal-health-
check.selfLink)'],
            'loadBalancingScheme': 'INTERNAL',
            'region': context.properties['region'],
            'protocol': 'TCP',
            'timeoutSec': 120
        }
    }, {
```

```
        'name': context.properties['infra_id'] + '-api-internal-forwarding-rule',
        'type': 'compute.v1.forwardingRule',
        'properties': {
            'backendService': '$(ref.' + context.properties['infra_id'] + '-api-internal-backend-
service.selfLink)',
            'IPAddress': '$(ref.' + context.properties['infra_id'] + '-cluster-ip.selfLink)',
            'loadBalancingScheme': 'INTERNAL',
            'ports': ['6443','22623'],
            'region': context.properties['region'],
            'subnetwork': context.properties['control_subnet']
        }
    }]

    for zone in context.properties['zones']:
        resources.append({
            'name': context.properties['infra_id'] + '-master-' + zone + '-instance-group',
            'type': 'compute.v1.instanceGroup',
            'properties': {
                'namedPorts': [
                    {
                        'name': 'ignition',
                        'port': 22623
                    }, {
                        'name': 'https',
                        'port': 6443
                    }
                ],
                'network': context.properties['cluster_network'],
                'zone': zone
            }
        })

    return {'resources': resources}
```

외부 클러스터를 생성할 때 **02_lb_ext.py** 템플릿 외에도 이 템플릿이 필요합니다.

7.10.11. GCP에서 프라이빗 DNS 영역 생성

OpenShift Container Platform 클러스터가 사용할 프라이빗 DNS 영역을 GCP(Google Cloud Platform)에 구성해야 합니다. 이 구성 요소를 생성하는 한 가지 방법은 제공된 Deployment Manager 템플릿을 수정하는 것입니다.



참고

GCP 인프라를 생성하는 데 제공된 Deployment Manager 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- GCP 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.

- GCP에서 VPC 및 관련 서브넷을 생성하고 구성하십시오.

프로세스

1. 이 항목의 **프라이빗 DNS에 대한 Deployment Manager 템플릿** 섹션에서 템플릿을 복사하여 사용자 컴퓨터에 **02_dns.py**로 저장합니다. 이 템플릿에서 클러스터에 필요한 프라이빗 DNS 개체를 설명합니다.

2. **02_dns.yaml** 리소스 정의 파일을 생성합니다.

```
$ cat <<EOF >02_dns.yaml
imports:
- path: 02_dns.py

resources:
- name: cluster-dns
  type: 02_dns.py
  properties:
    infra_id: '${INFRA_ID}' (1)
    cluster_domain: '${CLUSTER_NAME}.${BASE_DOMAIN}' (2)
    cluster_network: '${CLUSTER_NETWORK}' (3)
EOF
```

(1) **infra_id**는 추출 단계에서 가져온 **INFRA_ID** 인프라 이름입니다.

(2) **cluster_domain**은 클러스터의 도메인입니다(예: **openshift.example.com**).

(3) **cluster_network**는 클러스터 네트워크에 대한 **selfLink** URL입니다.

3. **gcloud** CLI를 사용하여 배포를 생성합니다.

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-dns --config 02_dns.yaml
```

4. Deployment Manager의 제한으로 인해 템플릿을 통해 DNS 항목이 생성되지 않으므로 수동으로 생성해야 합니다.

   a. 내부 DNS 항목을 추가합니다.

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction add ${CLUSTER_IP} --name
api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-
private-zone
$ gcloud dns record-sets transaction add ${CLUSTER_IP} --name api-
int.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-
private-zone
$ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone
```

   b. 외부 클러스터의 경우 외부 DNS 항목도 추가합니다.

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction add ${CLUSTER_PUBLIC_IP} --name
```

```
api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone
${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction execute --zone ${BASE_DOMAIN_ZONE_NAME}
```

7.10.11.1. 프라이빗 DNS에 대한 Deployment Manager 템플릿

다음 Deployment Manager 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 프라이빗 DNS를 배포할 수 있습니다.

예 7.4. 02_dns.py Deployment Manager 템플릿

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-private-zone',
        'type': 'dns.v1.managedZone',
        'properties': {
            'description': '',
            'dnsName': context.properties['cluster_domain'] + '.',
            'visibility': 'private',
            'privateVisibilityConfig': {
                'networks': [{
                    'networkUrl': context.properties['cluster_network']
                }]
            }
        }
    }]

    return {'resources': resources}
```

7.10.12. GCP에서 방화벽 규칙 생성

OpenShift Container Platform 클러스터에서 사용할 방화벽 규칙을 GCP(Google Cloud Platform)에서 생성해야 합니다. 이러한 구성 요소를 생성하는 한 가지 방법은 제공된 Deployment Manager 템플릿을 수정하는 것입니다.

참고

GCP 인프라를 생성하는 데 제공된 Deployment Manager 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- GCP 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.
- GCP에서 VPC 및 관련 서브넷을 생성하고 구성하십시오.

프로세스

1. 이 항목의 **방화벽 규칙에 대한 Deployment Manager 템플릿** 섹션에서 템플릿을 복사하여 사용자 컴퓨터에 **03_firewall.py**로 저장합니다. 이 템플릿에서 클러스터에 필요한 보안 그룹을 설명합니다.

2. **03_firewall.yaml** 리소스 정의 파일을 생성합니다.

```
$ cat <<EOF >03_firewall.yaml
imports:
- path: 03_firewall.py

resources:
- name: cluster-firewall
  type: 03_firewall.py
  properties:
    allowed_external_cidr: '0.0.0.0/0' (1)
    infra_id: '${INFRA_ID}' (2)
    cluster_network: '${CLUSTER_NETWORK}' (3)
    network_cidr: '${NETWORK_CIDR}' (4)
EOF
```

(1) **allowed_external_cidr**은 부트스트랩 호스트에 클러스터 API 및 SSH 액세스가 가능한 CIDR 범위입니다. 내부 클러스터의 경우 이 값을 **${NETWORK_CIDR}**로 설정합니다.

(2) **infra_id**는 추출 단계에서 가져온 **INFRA_ID** 인프라 이름입니다.

(3) **cluster_network**는 클러스터 네트워크에 대한 **selfLink** URL입니다.

(4) **network_cidr**은 VPC 네트워크의 CIDR입니다(예: **10.0.0.0/16**).

3. **gcloud** CLI를 사용하여 배포를 생성합니다.

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-firewall --config
03_firewall.yaml
```

7.10.12.1. 방화벽 규칙에 대한 Deployment Manager 템플릿

다음 Deployment Manager 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 방화벽 규칙을 배포할 수 있습니다.

예 7.5. 03_firewall.py Deployment Manager 템플릿

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-bootstrap-in-ssh',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['22']
            }],
            'sourceRanges': [context.properties['allowed_external_cidr']],
            'targetTags': [context.properties['infra_id'] + '-bootstrap']
```

```
        }
    }, {
        'name': context.properties['infra_id'] + '-api',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['6443']
            }],
            'sourceRanges': [context.properties['allowed_external_cidr']],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-health-checks',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['6080', '6443', '22624']
            }],
            'sourceRanges': ['35.191.0.0/16', '130.211.0.0/22', '209.85.152.0/22', '209.85.204.0/22'],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-etcd',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['2379-2380']
            }],
            'sourceTags': [context.properties['infra_id'] + '-master'],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-control-plane',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['10257']
            },{
                'IPProtocol': 'tcp',
                'ports': ['10259']
            },{
                'IPProtocol': 'tcp',
                'ports': ['22623']
            }],
            'sourceTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ],
```

```
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-internal-network',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'icmp'
            },{
                'IPProtocol': 'tcp',
                'ports': ['22']
            }],
            'sourceRanges': [context.properties['network_cidr']],
            'targetTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ]
        }
    }, {
        'name': context.properties['infra_id'] + '-internal-cluster',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'udp',
                'ports': ['4789', '6081']
            },{
                'IPProtocol': 'udp',
                'ports': ['500', '4500']
            },{
                'IPProtocol': 'esp',
            },{
                'IPProtocol': 'tcp',
                'ports': ['9000-9999']
            },{
                'IPProtocol': 'udp',
                'ports': ['9000-9999']
            },{
                'IPProtocol': 'tcp',
                'ports': ['10250']
            },{
                'IPProtocol': 'tcp',
                'ports': ['30000-32767']
            },{
                'IPProtocol': 'udp',
                'ports': ['30000-32767']
            }],
            'sourceTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ],
            'targetTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ]
```

```
    }
}]

return {'resources': resources}
```

7.10.13. GCP에서 IAM 역할 생성

OpenShift Container Platform 클러스터에서 사용할 IAM 역할을 GCP(Google Cloud Platform)에 생성해야 합니다. 이러한 구성 요소를 생성하는 한 가지 방법은 제공된 Deployment Manager 템플릿을 수정하는 것입니다.

참고

GCP 인프라를 생성하는 데 제공된 Deployment Manager 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- GCP 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.
- GCP에서 VPC 및 관련 서브넷을 생성하고 구성하십시오.

프로세스

1. 이 항목의 **IAM 역할에 대한 Deployment Manager 템플릿**섹션에서 템플릿을 복사하여 사용자 컴퓨터에 **03_iam.py**로 저장합니다. 이 템플릿에서 클러스터에 필요한 IAM 역할을 설명합니다.
2. **03_iam.yaml** 리소스 정의 파일을 생성합니다.

```
$ cat <<EOF >03_iam.yaml
imports:
- path: 03_iam.py
resources:
- name: cluster-iam
  type: 03_iam.py
  properties:
    infra_id: '${INFRA_ID}' (1)
EOF
```

(1) **infra_id**는 추출 단계에서 가져온 **INFRA_ID** 인프라 이름입니다.

3. **gcloud** CLI를 사용하여 배포를 생성합니다.

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-iam --config 03_iam.yaml
```

4. 마스터 서비스 계정에 대한 변수를 내보냅니다.

```
$ export MASTER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-m@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

-

5. 작업자 서비스 계정에 대한 변수를 내보냅니다.

```
$ export WORKER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

6. 컴퓨팅 머신을 호스팅하는 서브넷의 변수를 내보냅니다.

```
$ export COMPUTE_SUBNET=(`gcloud compute networks subnets describe ${INFRA_ID}-
worker-subnet --region=${REGION} --format json | jq -r .selfLink`)
```

7. Deployment Manager의 제한으로 인해 템플릿을 통해 정책 바인딩이 생성되지 않으므로 수동으로 생성해야 합니다.

```
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.instanceAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.networkAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.securityAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/iam.serviceAccountUser"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/storage.admin"

$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/compute.viewer"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/storage.admin"
```

8. 서비스 계정 키를 생성하여 나중에 사용할 수 있도록 로컬로 저장합니다.

```
$ gcloud iam service-accounts keys create service-account-key.json --iam-
account=${MASTER_SERVICE_ACCOUNT}
```

7.10.13.1. IAM 역할에 대한 Deployment Manager 템플릿

다음 Deployment Manager 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 IAM 역할을 배포할 수 있습니다.

예 7.6. 03_iam.py Deployment Manager 템플릿

```
def GenerateConfig(context):

  resources = [{
    'name': context.properties['infra_id'] + '-master-node-sa',
    'type': 'iam.v1.serviceAccount',
    'properties': {
        'accountId': context.properties['infra_id'] + '-m',
        'displayName': context.properties['infra_id'] + '-master-node'
    }
  }, {
    'name': context.properties['infra_id'] + '-worker-node-sa',
    'type': 'iam.v1.serviceAccount',
```

```
        'properties': {
            'accountId': context.properties['infra_id'] + '-w',
            'displayName': context.properties['infra_id'] + '-worker-node'
        }
    }]

    return {'resources': resources}
```

7.10.14. GCP 인프라용 RHCOS 클러스터 이미지 생성

OpenShift Container Platform 노드에 유효한 GCP(Google Cloud Platform)용 RHCOS(Red Hat Enterprise Linux CoreOS) 이미지를 사용해야 합니다.

프로세스

1. RHCOS 이미지 미러 페이지에서 RHCOS 이미지를 가져옵니다.

중요

RHCOS 이미지는 OpenShift Container Platform 릴리스에 따라 변경되지 않을 수 있습니다. 설치하는 OpenShift Container Platform 버전과 같거나 그 이하의 버전 중 가장 최신 버전의 이미지를 다운로드해야 합니다. 지원되는 경우 OpenShift Container Platform 버전과 일치하는 이미지 버전을 사용합니다.

rhcos-<version>-<arch>-gcp.<arch>.tar.gz 형식의 OpenShift Container Platform 버전 번호가 파일 이름에 포함되어 있습니다.

2. Google 스토리지 버킷을 생성합니다.

```
$ gsutil mb gs://<bucket_name>
```

3. RHCOS 이미지를 Google 스토리지 버킷에 업로드합니다.

```
$ gsutil cp <downloaded_image_file_path>/rhcos-<version>-x86_64-gcp.x86_64.tar.gz
gs://<bucket_name>
```

4. 업로드된 RHCOS 이미지 위치를 변수로 내보냅니다.

```
$ export IMAGE_SOURCE=gs://<bucket_name>/rhcos-<version>-x86_64-gcp.x86_64.tar.gz
```

5. 클러스터 이미지를 생성합니다.

```
$ gcloud compute images create "${INFRA_ID}-rhcos-image" \
    --source-uri="${IMAGE_SOURCE}"
```

7.10.15. GCP에서 부트스트랩 시스템 생성

OpenShift Container Platform 클러스터 초기화 과정에서 사용할 부트스트랩 시스템을 GCP(Google Cloud Platform)에 생성해야 합니다. 이 시스템을 생성하는 한 가지 방법은 제공된 Deployment Manager 템플릿을 수정하는 것입니다.

참고

부트스트랩 시스템을 생성하는 데 제공된 Deployment Manager 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- GCP 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.
- GCP에서 VPC 및 관련 서브넷을 생성하고 구성하십시오.
- GCP에서 네트워킹 및 로드 밸런서를 생성하고 구성하십시오.
- 컨트롤 플레인 및 컴퓨팅 역할을 생성합니다.
- pyOpenSSL이 설치되어 있는지 확인하십시오.

프로세스

1. 이 항목의 **부트스트랩 시스템에 대한 Deployment Manager 템플릿** 섹션에서 템플릿을 복사하여 사용자 컴퓨터에 **04_bootstrap.py**로 저장합니다. 이 템플릿은 클러스터에 필요한 부트스트랩 시스템을 설명합니다.
2. 설치 프로그램에 필요한 RHCOS(Red Hat Enterprise Linux CoreOS) 이미지의 위치를 내보냅니다.

```
$ export CLUSTER_IMAGE=(`gcloud compute images describe ${INFRA_ID}-rhcos-image --format json | jq -r .selfLink`)
```

3. 버킷을 생성하고 **bootstrap.ign** 파일을 업로드합니다.

```
$ gsutil mb gs://${INFRA_ID}-bootstrap-ignition
$ gsutil cp <installation_directory>/bootstrap.ign gs://${INFRA_ID}-bootstrap-ignition/
```

4. Ignition 구성 파일에 액세스하는 데 사용할 부트스트랩 인스턴스에 대한 서명된 URL을 생성합니다. 출력에서 URL을 변수로 내보냅니다.

```
$ export BOOTSTRAP_IGN=`gsutil signurl -d 1h service-account-key.json gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign | grep "^gs:" | awk '{print $5}'`
```

5. **04_bootstrap.yaml** 리소스 정의 파일을 생성합니다.

```
$ cat <<EOF >04_bootstrap.yaml
imports:
- path: 04_bootstrap.py

resources:
- name: cluster-bootstrap
  type: 04_bootstrap.py
  properties:
    infra_id: '${INFRA_ID}' 1
```

```
  region: '${REGION}' (2)
  zone: '${ZONE_0}' (3)

  cluster_network: '${CLUSTER_NETWORK}' (4)
  control_subnet: '${CONTROL_SUBNET}' (5)
  image: '${CLUSTER_IMAGE}' (6)
  machine_type: 'n1-standard-4' (7)
  root_volume_size: '128' (8)

  bootstrap_ign: '${BOOTSTRAP_IGN}' (9)
EOF
```

(1) **infra_id**는 추출 단계에서 가져온 **INFRA_ID** 인프라 이름입니다.

(2) **region**은 클러스터를 배포할 영역입니다(예: **us-central1**).

(3) **zone**은 부트스트랩 인스턴스를 배포할 영역입니다(예: **us-central1-b**).

(4) **cluster_network**는 클러스터 네트워크에 대한 **selfLink** URL입니다.

(5) **control_subnet**은 컨트롤 서브넷에 대한 **selfLink** URL입니다.

(6) **image**는 RHCOS 이미지에 대한 **selfLink** URL입니다.

(7) **machine_type**은 인스턴스의 시스템 유형입니다(예: **n1-standard-4**).

(8) **root_volume_size**는 부트스트랩 시스템의 부팅 디스크 크기입니다.

(9) **bootstrap_ign**은 서명된 URL을 생성할 때 출력되는 URL입니다.

6. **gcloud** CLI를 사용하여 배포를 생성합니다.

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-bootstrap --config
04_bootstrap.yaml
```

7. Deployment Manager의 제한으로 인해 템플릿을 통해 로드 밸런서 멤버십이 관리되지 않으므로 수동으로 부트스트랩 시스템을 추가해야 합니다.

   a. 내부 로드 밸런서 인스턴스 그룹에 부트스트랩 인스턴스를 추가합니다.

```
$ gcloud compute instance-groups unmanaged add-instances \
  ${INFRA_ID}-bootstrap-instance-group --zone=${ZONE_0} --instances=${INFRA_ID}-
bootstrap
```

   b. 내부 로드 밸런서 백엔드 서비스에 부트스트랩 인스턴스 그룹을 추가합니다.

```
$ gcloud compute backend-services add-backend \
  ${INFRA_ID}-api-internal-backend-service --region=${REGION} --instance-
group=${INFRA_ID}-bootstrap-instance-group --instance-group-zone=${ZONE_0}
```

7.10.15.1. 부트스트랩 시스템에 대한 Deployment Manager 템플릿

다음 Deployment Mananger 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 부트스트랩 시스템을 배포할 수 있습니다.

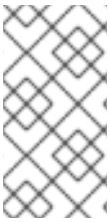
예 7.7.04_bootstrap.py Deployment Manager 템플릿

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-bootstrap-public-ip',
        'type': 'compute.v1.address',
        'properties': {
            'region': context.properties['region']
        }
    }, {
        'name': context.properties['infra_id'] + '-bootstrap',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': '{"ignition":{"config":{"replace":{"source":"' + context.properties['bootstrap_ign']
+ '"}},"version":"3.1.0"}}',
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet'],
                'accessConfigs': [{
                    'natIP': '$(ref.' + context.properties['infra_id'] + '-bootstrap-public-ip.address)'
                }]
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                    context.properties['infra_id'] + '-bootstrap'
                ]
            },
            'zone': context.properties['zone']
        }
    }, {
        'name': context.properties['infra_id'] + '-bootstrap-instance-group',
        'type': 'compute.v1.instanceGroup',
        'properties': {
            'namedPorts': [
                {
                    'name': 'ignition',
                    'port': 22623
```

```
            }, {
                'name': 'https',
                'port': 6443
            }
        ],
        'network': context.properties['cluster_network'],
        'zone': context.properties['zone']
    }
}]

return {'resources': resources}
```

7.10.16. GCP에 컨트롤 플레인 시스템 생성

클러스터에서 사용할 컨트롤 플레인 시스템을 GCP(Google Cloud Platform)에 생성해야 합니다. 이러한 시스템을 생성하는 한 가지 방법은 제공된 Deployment Manager 템플릿을 수정하는 것입니다.



참고

컨트롤 플레인 시스템을 생성하는 데 제공된 Deployment Manager 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- GCP 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.
- GCP에서 VPC 및 관련 서브넷을 생성하고 구성하십시오.
- GCP에서 네트워킹 및 로드 밸런서를 생성하고 구성하십시오.
- 컨트롤 플레인 및 컴퓨팅 역할을 생성합니다.
- 부트스트랩 시스템을 생성합니다.

프로세스

1. 이 항목의 **컨트롤 플레인 시스템에 대한 Deployment Manager 템플릿** 섹션에서 템플릿을 복사하여 사용자 컴퓨터에 **05_control_plane.py**로 저장합니다. 이 템플릿은 클러스터에 필요한 컨트롤 플레인 시스템을 설명합니다.
2. 리소스 정의에 필요한 다음 변수를 내보냅니다.

```
$ export MASTER_IGNITION=`cat <installation_directory>/master.ign`
```

3. **05_control_plane.yaml** 리소스 정의 파일을 생성합니다.

```
$ cat <<EOF >05_control_plane.yaml
imports:
- path: 05_control_plane.py
```

```
resources:
- name: cluster-control-plane
  type: 05_control_plane.py
  properties:
    infra_id: '${INFRA_ID}' (1)
    zones: (2)
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'

    control_subnet: '${CONTROL_SUBNET}' (3)
    image: '${CLUSTER_IMAGE}' (4)
    machine_type: 'n1-standard-4' (5)
    root_volume_size: '128'
    service_account_email: '${MASTER_SERVICE_ACCOUNT}' (6)

    ignition: '${MASTER_IGNITION}' (7)
EOF
```

1. **infra_id**는 추출 단계에서 가져온 **INFRA_ID** 인프라 이름입니다.
2. **zones**은 컨트롤 플레인 인스턴스를 배포할 영역입니다(예: **us-central1-a**, **us-central1-b**, **us-central1-c**).
3. **control_subnet**은 컨트롤 서브넷에 대한 **selfLink** URL입니다.
4. **image**는 RHCOS 이미지에 대한 **selfLink** URL입니다.
5. **machine_type**은 인스턴스의 시스템 유형입니다(예: **n1-standard-4**).
6. **service_account_email**은 사용자가 생성한 마스터 서비스 계정의 이메일 주소입니다.
7. **ignition**은 **master.ign** 파일의 내용입니다.

4. **gcloud** CLI를 사용하여 배포를 생성합니다.

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-control-plane --config
05_control_plane.yaml
```

5. Deployment Manager의 제한으로 인해 템플릿을 통해 로드 밸런서 멤버십이 관리되지 않으므로 수동으로 컨트롤 플레인 시스템을 추가해야 합니다.

   - 다음 명령을 실행하여 적절한 인스턴스 그룹에 컨트롤 플레인 머신을 추가합니다.

     ```
     $ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
     ${ZONE_0}-instance-group --zone=${ZONE_0} --instances=${INFRA_ID}-master-0
     $ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
     ${ZONE_1}-instance-group --zone=${ZONE_1} --instances=${INFRA_ID}-master-1
     $ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
     ${ZONE_2}-instance-group --zone=${ZONE_2} --instances=${INFRA_ID}-master-2
     ```

   - 외부 클러스터의 경우 다음 명령을 실행하여 대상 풀에 컨트롤 플레인 머신을 추가해야 합니다.

```
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_0}" --instances=${INFRA_ID}-master-0
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_1}" --instances=${INFRA_ID}-master-1
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_2}" --instances=${INFRA_ID}-master-2
```

7.10.16.1. 컨트롤 플레인 시스템에 대한 Deployment Manager 템플릿

다음 Deployment Mananger 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 컨트롤 플레인 시스템을 배포할 수 있습니다.

예 7.8. 05_control_plane.py Deployment Manager 템플릿

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-master-0',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'diskType': 'zones/' + context.properties['zones'][0] + '/diskTypes/pd-ssd',
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zones'][0] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                ]
            },
            'zone': context.properties['zones'][0]
        }
    }, {
        'name': context.properties['infra_id'] + '-master-1',
        'type': 'compute.v1.instance',
        'properties': {
```

```
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'diskType': 'zones/' + context.properties['zones'][1] + '/diskTypes/pd-ssd',
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zones'][1] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                ]
            },
            'zone': context.properties['zones'][1]
        }
    }, {
        'name': context.properties['infra_id'] + '-master-2',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'diskType': 'zones/' + context.properties['zones'][2] + '/diskTypes/pd-ssd',
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zones'][2] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet']
            }],
            'serviceAccounts': [{
```

```
            'email': context.properties['service_account_email'],
            'scopes': ['https://www.googleapis.com/auth/cloud-platform']
        }],
        'tags': {
            'items': [
                context.properties['infra_id'] + '-master',
            ]
        },
        'zone': context.properties['zones'][2]
    }
}]

return {'resources': resources}
```

7.10.17. 부트스트랩이 완료되길 기다렸다가 GCP에서 부트스트랩 리소스를 제거합니다.

GCP(Google Cloud Platform)에 필요한 인프라를 모두 생성한 후 설치 프로그램에 의해 생성된 Ignition 구성 파일을 사용하여 프로비저닝한 시스템에서 부트스트랩 프로세스가 완료될 때까지 기다립니다.

사전 요구 사항

- GCP 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.
- GCP에서 VPC 및 관련 서브넷을 생성하고 구성하십시오.
- GCP에서 네트워킹 및 로드 밸런서를 생성하고 구성하십시오.
- 컨트롤 플레인 및 컴퓨팅 역할을 생성합니다.
- 부트스트랩 시스템을 생성합니다.
- 컨트롤 플레인 시스템을 생성합니다.

프로세스

1. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

```
$ ./openshift-install wait-for bootstrap-complete --dir <installation_directory> \ 1
    --log-level info 2
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2 다른 설치 세부 사항을 보려면 **info** 대신 **warn**, **debug** 또는 **error**를 지정합니다.

FATAL 경고 없이 명령이 종료되면 프로덕션 컨트롤 플레인이 초기화된 것입니다.

2. 부트스트랩 리소스를 삭제합니다.

```
$ gcloud compute backend-services remove-backend ${INFRA_ID}-api-internal-backend-
service --region=${REGION} --instance-group=${INFRA_ID}-bootstrap-instance-group --
instance-group-zone=${ZONE_0}
```

```
$ gsutil rm gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign
$ gsutil rb gs://${INFRA_ID}-bootstrap-ignition
$ gcloud deployment-manager deployments delete ${INFRA_ID}-bootstrap
```

7.10.18. GCP에 추가 작업자 시스템 생성

개별 인스턴스를 따로 시작하거나 자동 확장 그룹과 같은 클러스터 외부의 자동화된 프로세스를 통해 클러스터에서 사용할 작업자 시스템을 GCP(Google Cloud Platform)에 생성할 수 있습니다. OpenShift Container Platform의 기본 제공 클러스터 확장 메커니즘과 시스템 API를 활용할 수도 있습니다.

예에서는 Deployment Manager 템플릿을 사용하여 인스턴스 하나를 수동으로 시작합니다. 파일에 **06_worker.py** 유형의 추가 리소스를 포함시켜 추가 인스턴스를 시작할 수 있습니다.

참고

작업자 시스템을 생성하는 데 제공된 Deployment Manager 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- GCP 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.
- GCP에서 VPC 및 관련 서브넷을 생성하고 구성하십시오.
- GCP에서 네트워킹 및 로드 밸런서를 생성하고 구성하십시오.
- 컨트롤 플레인 및 컴퓨팅 역할을 생성합니다.
- 부트스트랩 시스템을 생성합니다.
- 컨트롤 플레인 시스템을 생성합니다.

프로세스

1. 이 항목의 **작업자 시스템에 대한 Deployment Manager 템플릿** 섹션에서 템플릿을 복사하여 사용자 컴퓨터에 **06_worker.py**로 저장합니다. 이 템플릿에서 클러스터에 필요한 작업자 시스템을 설명합니다.
2. 리소스 정의가 사용하는 변수를 내보냅니다.
   a. 컴퓨팅 시스템을 호스팅하는 서브넷을 내보냅니다.

   ```
   $ export COMPUTE_SUBNET=(`gcloud compute networks subnets describe
   ${INFRA_ID}-worker-subnet --region=${REGION} --format json | jq -r .selfLink`)
   ```

   b. 서비스 계정의 이메일 주소를 내보냅니다.

   ```
   $ export WORKER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
   "email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
   ```

   c. 컴퓨팅 시스템 Ignition 구성 파일의 위치를 내보냅니다.

```
$ export WORKER_IGNITION=`cat <installation_directory>/worker.ign`
```

3. **06_worker.yaml** 리소스 정의 파일을 생성합니다.

```
$ cat <<EOF >06_worker.yaml
imports:
- path: 06_worker.py

resources:
- name: 'worker-0' (1)
  type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}' (2)
    zone: '${ZONE_0}' (3)
    compute_subnet: '${COMPUTE_SUBNET}' (4)
    image: '${CLUSTER_IMAGE}' (5)
    machine_type: 'n1-standard-4' (6)
    root_volume_size: '128'
    service_account_email: '${WORKER_SERVICE_ACCOUNT}' (7)
    ignition: '${WORKER_IGNITION}' (8)
- name: 'worker-1'
  type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}' (9)
    zone: '${ZONE_1}' (10)
    compute_subnet: '${COMPUTE_SUBNET}' (11)
    image: '${CLUSTER_IMAGE}' (12)
    machine_type: 'n1-standard-4' (13)
    root_volume_size: '128'
    service_account_email: '${WORKER_SERVICE_ACCOUNT}' (14)
    ignition: '${WORKER_IGNITION}' (15)
EOF
```

(1) **name**은 작업자 머신의 이름입니다(예: **worker-0**).

(2) (9) **infra_id**는 추출 단계에서 가져온 **INFRA_ID** 인프라 이름입니다.

(3) (10) **zone**은 작업자 시스템을 배포할 영역입니다(예: **us-central1-a**).

(4) (11) **compute_subnet**은 컴퓨팅 서브넷에 대한 **selfLink** URL입니다.

(5) (12) **image**는 RHCOS 이미지에 대한 **selfLink** URL입니다. [1]

(6) (13) **machine_type**은 인스턴스의 시스템 유형입니다(예: **n1-standard-4**).

(7) (14) **service_account_email**은 사용자가 생성한 작업자 서비스 계정의 이메일 주소입니다.

(8) (15) **ignition**은 **worker.ign** 파일의 내용입니다.

4. 선택사항: 추가 인스턴스를 시작하려면 **06_worker.yaml** 리소스 정의 파일에 **06_worker.py** 형식의 추가 리소스를 포함시키십시오.

5. **gcloud** CLI를 사용하여 배포를 생성합니다.

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-worker --config
06_worker.yaml
```

1. GCP Marketplace 이미지를 사용하려면 사용할 오퍼를 지정합니다.

   - OpenShift Container Platform:
     https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-ocp-48-x86-64-202210040145
   - OpenShift Platform Plus: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-opp-48-x86-64-202206140145**
   - OpenShift Kubernetes Engine:
     https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-oke-48-x86-64-202206140145

7.10.18.1. 작업자 시스템에 대한 Deployment Manager 템플릿

다음 Deloyment Manager 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 작업자 시스템을 배포할 수 있습니다.

예 7.9. 06_worker.py Deployment Manager 템플릿

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-' + context.env['name'],
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['compute_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-worker',
```

```
            ]
        },
        'zone': context.properties['zone']
    }
}]

return {'resources': resources}
```

7.10.19. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift CLI(**oc**)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 **oc**를 설치할 수 있습니다.

중요

이전 버전의 **oc**를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 **oc**를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 Linux에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Linux Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. **oc** 바이너리를 **PATH**에 있는 디렉터리에 배치합니다.
PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치

다음 절차에 따라 Windows에 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Windows Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.

4. ZIP 프로그램으로 아카이브의 압축을 풉니다.
5. **oc** 바이너리를 **PATH**에 있는 디렉터리로 이동합니다.
   PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치
다음 절차에 따라 macOS에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 MacOSX Client** 항목 옆에 있는 **지금 다운로드를** 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 해제하고 압축을 풉니다.
5. **oc** 바이너리 PATH의 디렉터리로 이동합니다.
   PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

7.10.20. CLI를 사용하여 클러스터에 로그인

클러스터 **kubeconfig** 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. **kubeconfig** 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- **oc** CLI를 설치했습니다.

프로세스

1. **kubeadmin** 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 **oc** 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

7.10.21. 머신의 인증서 서명 요청 승인

클러스터에 시스템을 추가하면 추가한 시스템별로 보류 중인 인증서 서명 요청(CSR)이 두 개씩 생성됩니다. 이러한 CSR이 승인되었는지 확인해야 하며, 필요한 경우 이를 직접 승인해야 합니다. 클라이언트 요청을 먼저 승인한 다음 서버 요청을 승인해야 합니다.

사전 요구 사항

- 클러스터에 시스템을 추가했습니다.

프로세스

1. 클러스터가 시스템을 인식하는지 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0 Ready    master  63m  v1.22.1
master-1 Ready    master  63m  v1.22.1
master-2 Ready    master  64m  v1.22.1
```

출력에 생성된 모든 시스템이 나열됩니다.

참고

이전 출력에는 일부 CSR이 승인될 때까지 컴퓨팅 노드(작업자 노드라고도 함)가 포함되지 않을 수 있습니다.

2. 보류 중인 CSR을 검토하고 클러스터에 추가한 각 시스템에 대해 **Pending** 또는 **Approved** 상태의 클라이언트 및 서버 요청이 표시되는지 확인합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE    REQUESTOR                                                   CONDITION
csr-8b2br   15m    system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
csr-8vnps   15m    system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
...
```

예에서는 두 시스템이 클러스터에 참여하고 있습니다. 목록에는 승인된 CSR이 더 많이 나타날 수도 있습니다.

3. CSR이 승인되지 않은 경우, 추가된 시스템에 대한 모든 보류 중인 CSR이 **Pending** 상태로 전환된 후 클러스터 시스템의 CSR을 승인합니다.

참고

CSR은 교체 주기가 자동으로 만료되므로 클러스터에 시스템을 추가한 후 1시간 이내에 CSR을 승인하십시오. 한 시간 내에 승인하지 않으면 인증서가 교체되고 각 노드에 대해 두 개 이상의 인증서가 표시됩니다. 이러한 인증서를 모두 승인해야 합니다. 클라이언트 CSR이 승인되면 Kubelet은 인증서에 대한 보조 CSR을 생성하므로 수동 승인이 필요합니다. 그러면 Kubelet에서 동일한 매개변수를 사용하여 새 인증서를 요청하는 경우 인증서 갱신 요청은 **machine-approver**에 의해 자동으로 승인됩니다.

참고

베어 메탈 및 기타 사용자 프로비저닝 인프라와 같이 머신 API를 사용하도록 활성화되지 않는 플랫폼에서 실행되는 클러스터의 경우 CSR(Kubelet service Certificate Request)을 자동으로 승인하는 방법을 구현해야 합니다. 요청이 승인되지 않으면 API 서버가 kubelet에 연결될 때 서비스 인증서가 필요하므로 **oc exec**, **oc rsh**, **oc logs** 명령을 성공적으로 수행할 수 없습니다. Kubelet 엔드 포인트에 연결하는 모든 작업을 수행하려면 이 인증서 승인이 필요합니다. 이 방법은 새 CSR을 감시하고 CSR이 **system:node** 또는 **system:admin** 그룹의 **node-bootstrapper** 서비스 계정에 의해 제출되었는지 확인하고 노드의 ID를 확인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>**은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

참고

일부 Operator는 일부 CSR이 승인될 때까지 사용할 수 없습니다.

4. 이제 클라이언트 요청이 승인되었으므로 클러스터에 추가한 각 머신의 서버 요청을 검토해야 합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE    REQUESTOR                                                   CONDITION
csr-bfd72   5m26s  system:node:ip-10-0-50-126.us-east-2.compute.internal
```

```
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 나머지 CSR이 승인되지 않고 **Pending** 상태인 경우 클러스터 머신의 CSR을 승인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>**은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. 모든 클라이언트 및 서버 CSR이 승인된 후 머신은 **Ready** 상태가 됩니다. 다음 명령을 실행하여 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME      STATUS   ROLES   AGE  VERSION
master-0  Ready    master  73m  v1.22.1
master-1  Ready    master  73m  v1.22.1
master-2  Ready    master  74m  v1.22.1
worker-0  Ready    worker  11m  v1.22.1
worker-1  Ready    worker  11m  v1.22.1
```

참고

머신이 **Ready** 상태로 전환하는 데 서버 CSR의 승인 후 몇 분이 걸릴 수 있습니다.

추가 정보

- CSR에 대한 자세한 내용은 인증서 서명 요청을 참조하십시오.

7.10.22. 선택사항 : 인그레스 DNS 레코드 추가

Kubernetes 매니페스트를 생성하고 Ignition 구성을 생성할 때 DNS 영역 구성을 제거한 경우, 인그레스 로드 밸런서를 가리키는 DNS 레코드를 수동으로 생성해야 합니다. 와일드카드 ***.apps.{baseDomain}.** 또는 특정 레코드를 생성할 수 있습니다. 사용자 요구사항에 따라 A, CNAME 및 기타 레코드를 사용할 수 있습니다.

사전 요구 사항

- GCP 계정을 구성하십시오.
- Kubernetes 매니페스트를 생성하고 Ignition 구성을 생성할 때 DNS 영역 구성을 제거하십시오.

- GCP에서 VPC 및 관련 서브넷을 생성하고 구성하십시오.
- GCP에서 네트워킹 및 로드 밸런서를 생성하고 구성하십시오.
- 컨트롤 플레인 및 컴퓨팅 역할을 생성합니다.
- 부트스트랩 시스템을 생성합니다.
- 컨트롤 플레인 시스템을 생성합니다.
- 작업자 시스템을 생성합니다.

프로세스

1. 인그레스 라우터가 로드 밸런서를 생성하고 **EXTERNAL-IP** 필드를 채울 때까지 기다립니다.

```
$ oc -n openshift-ingress get service router-default
```

출력 예

```
NAME            TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)                      AGE
router-default  LoadBalancer   172.30.18.154  35.233.157.184
80:32288/TCP,443:31215/TCP   98
```

2. 영역에 A 레코드를 추가합니다.
   - A 레코드를 사용하려면,
     i. 라우터 IP 주소에 대한 변수를 내보냅니다.

```
$ export ROUTER_IP=`oc -n openshift-ingress get service router-default --no-
headers | awk '{print $4}'`
```

     ii. 프라이빗 영역에 A 레코드를 추가합니다.

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction add ${ROUTER_IP} --name
\*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone
${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone
```

     iii. 외부 클러스터의 경우 퍼블릭 영역에 A 레코드를 추가합니다.

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction add ${ROUTER_IP} --name
\*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone
${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction execute --zone
${BASE_DOMAIN_ZONE_NAME}
```

   - 와일드카드를 사용하지 않고 명시적 도메인을 추가하려면 클러스터의 현재 경로에 대한 항목을 생성합니다.

```
$ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}
{"\n"}{end}{end}' routes
```

출력 예

```
oauth-openshift.apps.your.cluster.domain.example.com
console-openshift-console.apps.your.cluster.domain.example.com
downloads-openshift-console.apps.your.cluster.domain.example.com
alertmanager-main-openshift-monitoring.apps.your.cluster.domain.example.com
grafana-openshift-monitoring.apps.your.cluster.domain.example.com
prometheus-k8s-openshift-monitoring.apps.your.cluster.domain.example.com
```

7.10.23. 사용자 프로비저닝 인프라에서 GCP 설치 완료

GCP(Google Cloud Platform) 사용자 프로비저닝 인프라에서 OpenShift Container Platform 설치를 시작한 후 클러스터가 준비를 마칠 때까지 클러스터 이벤트를 모니터링할 수 있습니다.

사전 요구 사항

- 사용자 프로비저닝 GCP 인프라에 OpenShift Container Platform 클러스터용 부트스트랩 시스템을 배포하십시오.
- **oc** CLI를 설치하고 로그인하십시오.

프로세스

1. 클러스터 설치를 완료합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

출력 예

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

중요

- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

2. 클러스터의 실행 상태를 관찰합니다.

   a. 다음 명령을 실행하여 현재 클러스터 버전과 상태를 확인합니다.

   ```
   $ oc get clusterversion
   ```

   출력 예

   ```
   NAME     VERSION   AVAILABLE   PROGRESSING   SINCE   STATUS
   version            False       True          24m     Working towards 4.5.4: 99% complete
   ```

   b. 다음 명령을 실행하여 컨트롤 플레인에서 관리하되는 운영자를 CVO(Cluster Version Operator)별로 확인합니다.

   ```
   $ oc get clusteroperators
   ```

   출력 예

   ```
   NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
   SINCE
   authentication                             4.5.4     True        False         False      7m56s
   cloud-credential                           4.5.4     True        False         False      31m
   cluster-autoscaler                         4.5.4     True        False         False      16m
   console                                    4.5.4     True        False         False      10m
   csi-snapshot-controller                    4.5.4     True        False         False      16m
   dns                                        4.5.4     True        False         False      22m
   etcd                                       4.5.4     False       False         False      25s
   image-registry                             4.5.4     True        False         False      16m
   ingress                                    4.5.4     True        False         False      16m
   insights                                   4.5.4     True        False         False      17m
   kube-apiserver                             4.5.4     True        False         False      19m
   kube-controller-manager                    4.5.4     True        False         False      20m
   kube-scheduler                             4.5.4     True        False         False      20m
   kube-storage-version-migrator              4.5.4     True        False         False      16m
   machine-api                                4.5.4     True        False         False      22m
   machine-config                             4.5.4     True        False         False      22m
   marketplace                                4.5.4     True        False         False      16m
   monitoring                                 4.5.4     True        False         False      10m
   network                                    4.5.4     True        False         False      23m
   node-tuning                                4.5.4     True        False         False      23m
   openshift-apiserver                        4.5.4     True        False         False      17m
   openshift-controller-manager               4.5.4     True        False         False      15m
   openshift-samples                          4.5.4     True        False         False      16m
   operator-lifecycle-manager                 4.5.4     True        False         False      22m
   operator-lifecycle-manager-catalog         4.5.4     True        False         False      22m
   operator-lifecycle-manager-packageserver   4.5.4     True        False         False      18m
   service-ca                                 4.5.4     True        False         False      23m
   service-catalog-apiserver                  4.5.4     True        False         False      23m
   service-catalog-controller-manager         4.5.4     True        False         False      23m
   storage                                    4.5.4     True        False         False      17m
   ```

   c. 다음 명령어를 실행하여 클러스터 Pod를 확인합니다.

   ```
   $ oc get pods --all-namespaces
   ```

출력 예

```
NAMESPACE                                          NAME
READY   STATUS    RESTARTS   AGE
kube-system                                        etcd-member-ip-10-0-3-111.us-east-
2.compute.internal             1/1     Running   0          35m
kube-system                                        etcd-member-ip-10-0-3-239.us-east-
2.compute.internal             1/1     Running   0          37m
kube-system                                        etcd-member-ip-10-0-3-24.us-east-
2.compute.internal             1/1     Running   0          35m
openshift-apiserver-operator                       openshift-apiserver-operator-6d6674f4f4-
h7t2t                          1/1     Running   1          37m
openshift-apiserver                                apiserver-fm48r
1/1     Running   0          30m
openshift-apiserver                                apiserver-fxkvv
1/1     Running   0          29m
openshift-apiserver                                apiserver-q85nm
1/1     Running   0          29m
...
openshift-service-ca-operator                      openshift-service-ca-operator-66ff6dc6cd-
9r257                          1/1     Running   0          37m
openshift-service-ca                               apiservice-cabundle-injector-695b6bcbc-cl5hm
1/1     Running   0          35m
openshift-service-ca                               configmap-cabundle-injector-8498544d7-
25qn6                          1/1     Running   0          35m
openshift-service-ca                               service-serving-cert-signer-6445fc9c6-wqdqn
1/1     Running   0          35m
openshift-service-catalog-apiserver-operator       openshift-service-catalog-apiserver-
operator-549f44668b-b5q2w      1/1     Running   0          32m
openshift-service-catalog-controller-manager-operator   openshift-service-catalog-
controller-manager-operator-b78cr2lnm   1/1     Running   0          31m
```

현재 클러스터 버전이 **AVAILABLE**이면 설치가 완료된 것입니다.

7.10.24. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

7.10.25. 다음 단계

- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.

- GCP에서 Ingress 컨트롤러에 대한 글로벌 액세스를 구성합니다 .

7.11. DEPLOYMENT MANAGER 템플릿을 사용하여 GCP의 공유 VPC에 클러스터 설치

OpenShift Container Platform 4.9 버전에서는 사용자가 제공하는 인프라를 사용하는 클러스터를 GCP(Google Cloud Platform)의 공유 VPC(Virtual Private Cloud)에 설치할 수 있습니다. 이 컨텍스트에서 공유 VPC에 설치된 클러스터는 클러스터가 배포되는 위치와 다른 프로젝트의 VPC를 사용하도록 구성된 클러스터입니다.

공유 VPC를 통해 조직은 여러 프로젝트의 리소스를 공통 VPC 네트워크로 연결할 수 있습니다. 해당 네트워크의 내부 IP를 사용하여 조직 내에서 안전하고 효율적으로 통신할 수 있습니다. 공유 VPC에 대한 자세한 내용은 GCP 문서의 공유 VPC 개요를 참조하십시오.

공유 VPC에 사용자 제공 인프라 설치를 수행하는 단계가 여기에 안내되어 있습니다. 안내된 단계를 수행하거나 자체 모델링에 유용한 몇 가지 Deployment Manager 템플릿이 제공됩니다. 여러 다른 방법을 통해 필요한 리소스를 자유롭게 생성할 수도 있습니다.

중요

사용자가 프로비저닝한 인프라 설치를 수행하는 단계는 예시용으로만 제공됩니다. 사용자가 제공하는 인프라를 사용하여 클러스터를 설치하려면 클라우드 공급자 및 OpenShift Container Platform 설치 프로세스에 대한 정보가 필요합니다. 안내된 단계를 수행하거나 자체 모델링에 유용한 몇 가지 Deployment Manager 템플릿이 제공됩니다. 여러 다른 방법을 통해 필요한 리소스를 자유롭게 생성할 수도 있습니다. 템플릿은 예시일 뿐입니다.

7.11.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비 에 대한 문서를 읽습니다.
- 방화벽을 사용하며 Telemetry 서비스를 사용할 예정인 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성해야 합니다.
- 사용자의 환경에서 클라우드 ID 및 액세스 관리(IAM) API에 액세스할 수 없거나 **kube-system** 네임스페이스에 관리자 수준의 인증 정보 시크릿을 저장하지 않으려면 IAM 인증 정보 수동 생성 및 유지 관리를 참조하십시오.

참고

프록시를 구성하는 경우에도 해당 사이트 목록을 검토하십시오.

7.11.2. 인증서 서명 요청 관리

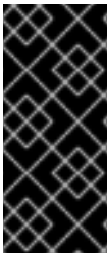
사용자가 프로비저닝하는 인프라를 사용하는 경우 자동 시스템 관리 기능으로 인해 클러스터의 액세스가 제한되므로 설치한 후 클러스터 인증서 서명 요청(CSR)을 승인하는 메커니즘을 제공해야 합니다. **kube-controller-manager**는 kubelet 클라이언트 CSR만 승인합니다. **machine-approver**는 올바른 시스템에서 발행한 요청인지 확인할 수 없기 때문에 kubelet 자격 증명을 사용하여 요청하는 서비스 인증서의 유효성을 보장할 수 없습니다. kubelet 서빙 인증서 요청의 유효성을 확인하고 요청을 승인하는 방법을 결정하여 구현해야 합니다.

7.11.3. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.



중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

7.11.4. 클러스터를 호스팅하는 GCP 프로젝트 구성

OpenShift Container Platform을 설치하려면 먼저 호스팅할 GCP(Google Cloud Platform) 프로젝트를 구성해야 합니다.

7.11.4.1. GCP 프로젝트 생성

OpenShift Container Platform을 설치하려면 클러스터를 호스팅할 GCP(Google Cloud Platform) 계정에 프로젝트를 생성해야 합니다.

프로세스

- OpenShift Container Platform 클러스터를 호스팅할 프로젝트를 생성합니다. GCP 문서의 프로젝트 생성 및 관리 단원을 참조하십시오.

중요

설치 관리자 프로비저닝 인프라를 사용하는 경우 GCP 프로젝트는 Premium Network Service Tier를 사용해야 합니다. 설치 프로그램을 사용하여 설치된 클러스터의 Standard Network Service Tier는 지원되지 않습니다. 설치 프로그램은 **api-int.<cluster_name>.<base_domain>** URL에 대한 내부 로드 밸런싱을 구성합니다.

7.11.4.2. GCP에서 API 서비스 활성화

GCP(Google Cloud Platform) 프로젝트에서 OpenShift Container Platform 설치를 완료하려면 여러 API 서비스에 액세스해야 합니다.

사전 요구 사항

- 클러스터를 호스팅할 프로젝트 생성을 완료했습니다.

프로세스

- 클러스터를 호스팅하는 프로젝트에서 다음과 같은 필수 API 서비스를 활성화합니다. GCP 문서의 서비스 활성화 단원을 참조하십시오.

표 7.43. 필수 API 서비스

API 서비스	콘솔 서비스 이름
Cloud Deployment Manager V2 API	**deploymentmanager.googleapis.com**
컴퓨팅 엔진 API	**compute.googleapis.com**
Google 클라우드 API	**cloudapis.googleapis.com**
클라우드 리소스 관리자 API	**cloudresourcemanager.googleapis.com**
Google DNS API	**dns.googleapis.com**
IAM 서비스 계정 자격 증명 API	**iamcredentials.googleapis.com**
IAM(ID 및 액세스 관리) API	**iam.googleapis.com**
서비스 관리 API	**servicemanagement.googleapis.com**
서비스 사용량 API	**serviceusage.googleapis.com**
Google 클라우드 스토리지 JSON API	**storage-api.googleapis.com**
클라우드 스토리지	**storage-component.googleapis.com**

7.11.4.3. GCP 계정 제한

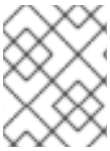
OpenShift Container Platform 클러스터는 여러 GCP(Google Cloud Platform) 구성 요소를 사용하지만 기본 할당량이 기본 OpenShift Container Platform 클러스터 설치가 가능할지 여부에 영향을 미치지 않습니다.

컴퓨팅 및 컨트롤 플레인 시스템 세 개가 포함된 기본 클러스터는 다음과 같은 리소스를 사용합니다. 일부 리소스는 부트스트랩 프로세스 중에만 필요하며 클러스터 배포 후 제거됩니다.

표 7.44. 기본 클러스터에서 사용되는 GCP 리소스

서비스	구성 요소	위치	필요한 총 리소스	부트스트랩 후 제거된 리소스
서비스 계정	IAM	글로벌	5	0
방화벽 규칙	네트워킹	글로벌	11	1
전송 규칙	컴퓨팅	글로벌	2	0

서비스	구성 요소	위치	필요한 총 리소스	부트스트랩 후 제거된 리소스
상태 검사	컴퓨팅	글로벌	2	0
이미지	컴퓨팅	글로벌	1	0
네트워크	네트워킹	글로벌	1	0
라우터	네트워킹	글로벌	1	0
라우트	네트워킹	글로벌	2	0
서브네트워크	컴퓨팅	글로벌	2	0
대상 풀	네트워킹	글로벌	2	0



참고

설치하는 동안 할당량이 충분하지 않으면 설치 프로그램에서 초과된 할당량과 리전을 모두 안내하는 오류 메시지를 표시합니다.

실제 클러스터 크기, 예상 클러스터 증가, 계정과 연결된 다른 클러스터의 사용량을 모두 고려해야 합니다. CPU, 고정 IP 주소, 영구 디스크 SSD(스토리지) 할당량이 가장 부족하기 쉬운 할당량입니다.

다음 리전 중 하나에서 클러스터를 배포하려는 경우, 최대 스토리지 할당량을 초과할 것이며, CPU 할당량 제한을 초과할 가능성도 있습니다.

- **asia-east2**
- **asia-northeast2**
- **asia-south1**
- **australia-southeast1**
- **europe-north1**
- **europe-west2**
- **europe-west3**
- **europe-west6**
- **northamerica-northeast1**
- **southamerica-east1**
- **us-west2**

GCP 콘솔에서 리소스 할당량을 늘릴 수는 있지만 지원 티켓을 제출해야 할 수도 있습니다. OpenShift Container Platform 클러스터를 설치하기 전에 지원 티켓을 해결할 시간이 충분하도록 조기에 클러스터 크기를 계획해야 합니다.

7.11.4.4. GCP에서 서비스 계정 생성

OpenShift Container Platform에는 Google API의 데이터에 액세스하기 위한 인증 및 승인을 제공하는 GCP(Google Cloud Platform) 서비스 계정이 필요합니다. 프로젝트에 필요한 역할이 포함된 기존 IAM 서비스 계정이 없으면 새로 생성해야 합니다.

사전 요구 사항

- 클러스터를 호스팅할 프로젝트 생성을 완료했습니다.

프로세스

1. OpenShift Container Platform 클러스터를 호스팅하는 데 사용하는 프로젝트에 서비스 계정을 생성합니다. GCP 문서의 서비스 계정 생성 단원을 참조하십시오.
2. 서비스 계정에 적절한 권한을 부여합니다. 뒤따르는 개별 권한을 부여하거나 **Owner** 역할을 할당할 수 있습니다. 서비스 계정에 특정 리소스에 대한 역할 부여 를 참조하십시오.

참고

서비스 계정을 프로젝트 소유자로 지정하는 것은 가장 쉽게 필요한 권한을 얻는 방법이며, 서비스 계정으로 프로젝트를 완전히 제어할 수 있음을 의미합니다. 해당 권한을 제공하는 데 따른 위험이 수용 가능한 수준인지 확인해봐야 합니다.

3. JSON 형식으로 서비스 계정 키를 생성합니다. GCP 문서의 서비스 계정 키 생성 단원을 참조하십시오.
   클러스터를 생성하기 위해서는 서비스 계정 키가 필요합니다.

7.11.4.4.1. 필요한 GCP 권한

생성하는 서비스 계정에 **Owner** 역할을 연결하면 OpenShift Container Platform 설치에 필요한 권한을 포함하여 모든 권한이 해당 서비스 계정에 부여됩니다. OpenShift Container Platform 클러스터를 배포하려면 서비스 계정에 다음과 같은 권한이 필요합니다. 기존 VPC에 클러스터를 배포할 때는 다음 목록에 제시된 특정 네트워킹 권한이 서비스 계정에 필요하지 않습니다.

설치 프로그램에 필요한 역할

- 컴퓨팅 관리자
- 보안 관리자
- 서비스 계정 관리자
- 서비스 계정 사용자
- 스토리지 관리자

설치 과정에서 네트워크 리소스를 생성하는 데 필요한 역할

- DNS 관리자

사용자 프로비저닝 GCP 인프라에 필요한 역할

- 배포 관리자 편집자
- 서비스 계정 키 관리자

선택적 역할

운영자를 위한 제한적 자격 증명을 새로 생성하는 클러스터의 경우 다음 역할을 추가합니다.

- 서비스 계정 키 관리자

컨트롤 플레인 및 컴퓨팅 시스템이 사용하는 서비스 계정에 적용되는 역할입니다.

표 7.45. GCP 서비스 계정 권한

계정	역할
컨트롤 플레인	**roles/compute.instanceAdmin**
	roles/compute.networkAdmin
	roles/compute.securityAdmin
	roles/storage.admin
	roles/iam.serviceAccountUser
컴퓨팅	**roles/compute.viewer**
	roles/storage.admin

7.11.4.5. 지원되는 GCP 리전

다음과 같은 GCP(Google Cloud Platform) 리전에 OpenShift Container Platform 클러스터를 배포할 수 있습니다.

- **asia-east1** (대만 장화현)
- **asia-east2** (홍콩)
- **asia-northeast1** (일본 도쿄)
- **asia-northeast2** (일본 오사카)
- **asia-northeast3** (한국 서울)
- **asia-south1** (인도 뭄바이)
- **asia-southeast1** (싱가포르 주롱 웨스트)
- **asia-southeast2** (인도네시아 자카르타)
- **australia-southeast1** (호주 시드니)

- **europe-central2** (폴란드 바르샤바)
- **europe-north1** (핀란드 하미나)
- **europe-west1** (벨기에 생기슬랭)
- **europe-west2** (영국 런던)
- **europe-west3** (독일 프랑크푸르트)
- **europe-west4** (네덜란드 엠스하벤)
- **europe-west6** (스위스 취리히)
- **northamerica-northeast1** (캐나다 퀘벡 주 몬트리올)
- **southamerica-east1** (브라질 상파울루)
- **us-central1** (미국 아이오와 주 카운실 블러프스)
- **us-east1** (미국 사우스 캐롤라이나 주 몽크스 코너)
- **us-east4** (미국 노던 버지니아 주 애쉬번)
- **us-west1** (미국 오레곤 주 댈러스)
- **us-west2** (미국 캘리포니아 주 로스앤젤레스)
- **us-west3** (미국 유타 주 솔트레이크시티)
- **us-west4** (미국 네바다 라스베거스)

7.11.4.6. GCP용 CLI 도구 설치 및 구성

사용자 프로비저닝 인프라를 사용하는 OpenShift Container Platform을 GCP(Google Cloud Platform)에 설치하려면 GCP용 CLI 도구를 설치하고 구성해야 합니다.

사전 요구 사항

- 클러스터를 호스팅할 프로젝트 생성을 완료했습니다.
- 서비스 계정 생성과 필요한 권한 부여 작업을 마쳤습니다.

프로세스

1. **$PATH에** 다음 바이너리를 설치합니다.
   - **gcloud**
   - **gsutil**

   GCP 문서의 최신 클라우드 SDK 버전 설치 단원을 참조하십시오.
2. 구성된 서비스 계정과 **gcloud** 도구를 사용하여 인증합니다.
   GCP 문서의 서비스 계정 인증을 참조하십시오.

7.11.5. 사용자 프로비저닝 인프라를 포함한 클러스터의 시스템 요구사항

사용자 프로비저닝 인프라가 포함된 클러스터의 경우, 필요한 모든 시스템을 배포해야 합니다.

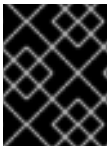
이 섹션에서는 사용자 프로비저닝 인프라에 OpenShift Container Platform을 배포해야 하는 요구 사항에 대해 설명합니다.

7.11.5.1. 클러스터 설치에 필요한 시스템

최소 OpenShift Container Platform 클러스터에 다음과 같은 호스트가 필요합니다.

표 7.46. 최소 필수 호스트

호스트	설명
임시 부트스트랩 시스템 한 개	컨트롤 플레인 시스템 세 개에 OpenShift Container Platform 클러스터를 배포하기 위한 부트스트랩 시스템이 클러스터에 필요합니다. 클러스터를 설치한 후 부트스트랩 시스템을 제거할 수 있습니다.
컨트롤 플레인 시스템 세 개	컨트롤 플레인 시스템은 컨트롤 플레인을 구성하는 Kubernetes 및 OpenShift Container Platform 서비스를 실행합니다.
두 개 이상의 컴퓨팅 시스템(작업자 시스템이라고도 함).	OpenShift Container Platform 사용자가 요청한 워크로드는 컴퓨팅 머신에서 실행됩니다.



중요

클러스터의 고가용성을 유지하려면 이러한 클러스터 시스템에 대해 별도의 물리적 호스트를 사용하십시오.

부트스트랩, 컨트롤 플레인 시스템은 운영 체제로 RHCOS (Red Hat Enterprise Linux CoreOS)를 사용해야 합니다. 그러나 컴퓨팅 머신은 RHCOS(Red Hat Enterprise Linux CoreOS), RHEL (Red Hat Enterprise Linux) 7.9 또는 RHEL 8.4 중에서 선택할 수 있습니다.

RHCOS는 RHEL 8(Red Hat Enterprise Linux)을 기반으로하며 모든 하드웨어 인증 및 요구사항을 모두 이어받습니다. Red Hat Enterprise Linux 기술 기능 및 제한 을 참조하십시오.

7.11.5.2. 클러스터 설치를 위한 최소 리소스 요구 사항

각 클러스터 시스템이 다음과 같은 최소 요구사항을 충족해야 합니다.

표 7.47. 최소 리소스 요구사항

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
부트스트랩	RHCOS	4	16GB	100GB	300
컨트롤 플레인	RHCOS	4	16GB	100GB	300

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
컴퓨팅	RHCOS, RHEL 7.9 또는 RHEL 8.4 [3]	2	8GB	100GB	300

1. SMT(동시 멀티스레딩) 또는 하이퍼 스레딩이 활성화되지 않은 경우 하나의 vCPU는 하나의 물리적 코어와 동일합니다. 활성화하면 다음과 같은 공식을 사용하여 해당 비율을 계산합니다. (코어당 스레드 수 × 코어 수) × 소켓 수 = vCPU 수

2. OpenShift Container Platform 및 Kubernetes는 디스크 성능에 민감하며 특히 10ms p99 fsync 기간이 필요한 컨트롤 플레인 노드의 etcd에 더 빠른 스토리지가 권장됩니다. 많은 클라우드 플랫폼에서 스토리지 크기와 IOPS를 함께 확장되므로 충분한 성능을 얻으려면 스토리지 볼륨을 과도하게 할당해야 할 수 있습니다.

3. 사용자가 프로비저닝한 모든 설치와 마찬가지로 클러스터에서 RHEL 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. RHEL 7 컴퓨팅 머신 사용은 더 이상 사용되지 않으며 향후 OpenShift Container Platform 4 릴리스에서 제거될 예정입니다.

7.11.5.3. 사용자 정의 머신 유형 사용

OpenShift Container Platform 클러스터를 설치하기 위해 사용자 정의 머신 유형을 사용할 수 있습니다.

사용자 지정 머신 유형을 사용할 때는 다음을 고려하십시오.

- 사전 정의된 인스턴스 유형과 유사하게 사용자 정의 머신 유형은 컨트롤 플레인 및 컴퓨팅 시스템의 최소 리소스 요구 사항을 충족해야 합니다. 자세한 내용은 "클러스터 설치를 위한 최소 리소스 요구 사항"을 참조하십시오.

- 사용자 정의 머신 유형의 이름은 다음 구문을 준수해야 합니다.
  custom-<number_of_cpus>-<amount_of_memory_in_mb>

  예를 들면 **custom-6-20480** 입니다.

7.11.6. 공유 VPC 네트워크를 호스팅하는 GCP 프로젝트 구성

공유 VPC(Virtual Private Cloud)를 사용하여 GCP(Google Cloud Platform)에서 OpenShift Container Platform 클러스터를 호스팅하는 경우, 호스팅할 프로젝트를 구성해야 합니다.

참고

공유 VPC 네트워크를 호스팅하는 프로젝트가 이미 있으면 이 섹션을 검토하여 프로젝트가 OpenShift Container Platform 클러스터를 설치하기 위한 모든 요구사항을 충족하는지 확인하십시오.

프로세스

1. OpenShift Container Platform 클러스터의 공유 VPC를 호스팅할 프로젝트를 생성합니다. GCP 문서의 프로젝트 생성 및 관리 단원을 참조하십시오.

2. 공유 VPC를 호스팅하는 프로젝트에서 서비스 계정을 생성합니다. GCP 문서의 서비스 계정 생성 단원을 참조하십시오.

3. 서비스 계정에 적절한 권한을 부여합니다. 뒤따르는 개별 권한을 부여하거나 **Owner** 역할을 할당할 수 있습니다. 서비스 계정에 특정 리소스에 대한 역할 부여 를 참조하십시오.

참고

서비스 계정을 프로젝트 소유자로 지정하는 것은 가장 쉽게 필요한 권한을 얻는 방법이며, 서비스 계정으로 프로젝트를 완전히 제어할 수 있음을 의미합니다. 해당 권한을 제공하는 데 따른 위험이 수용 가능한 수준인지 확인해봐야 합니다.

공유 VPC 네트워크를 호스팅하는 프로젝트의 서비스 계정에는 다음과 같은 역할이 필요합니다.

- 컴퓨팅 네트워크 사용자
- 컴퓨팅 보안 관리자
- 배포 관리자 편집자
- DNS 관리자
- 보안 관리자
- 네트워크 관리 관리자

7.11.6.1. GCP용 DNS 구성

OpenShift Container Platform을 설치하려면 사용하는 GCP(Google Cloud Platform) 계정에 클러스터를 설치하는 공유 VPC를 호스팅하는 프로젝트에 전용 퍼블릭 호스팅 영역이 있어야 합니다. 도메인에 대한 권한도 이 영역에 있어야 합니다. DNS 서비스는 클러스터와 외부 연결에 필요한 클러스터 DNS 확인 및 이름 조회 기능을 제공합니다.

프로세스

1. 도메인 또는 하위 도메인과 등록 기관을 식별합니다. 기존 도메인 및 등록 기관을 이전하거나 GCP 또는 다른 소스를 통해 새 도메인과 등록 기관을 구할 수 있습니다.

참고

새 도메인을 구입하는 경우, 관련 DNS 변경사항이 전파되는 데 시간이 걸릴 수 있습니다. Google을 통한 도메인 구매에 대한 자세한 내용은 Google 도메인 을 참조하십시오.

2. GCP 프로젝트에서 도메인 또는 하위 도메인의 퍼블릭 호스팅 영역을 생성합니다. GCP 문서의 퍼블릭 영역 생성 단원을 참조하십시오.
적절한 루트 도메인(예: **openshiftcorp.com**) 또는 하위 도메인(예: **clusters.openshiftcorp.com**)을 사용합니다.

3. 호스팅 영역 레코드에서 권한이 있는 새 이름 서버를 추출합니다. GCP 문서의 클라우드 DNS 이름 서버 조회 단원을 참조하십시오.
일반적으로 네 가지 이름 서버가 있습니다.

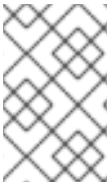
4. 도메인에서 사용하는 이름 서버의 등록 기관 레코드를 업데이트합니다. 예를 들어, 사용자 도메인을 Google 도메인에 등록한 경우 Google 도메인 도움말의 사용자 지정 이름 서버로 전환하는 방법 항목을 참조하십시오.

5. 루트 도메인을 Google Cloud DNS로 마이그레이션했으면 DNS 레코드를 마이그레이션합니다. GCP 문서의 클라우드 DNS로 마이그레이션을 참조하십시오.

6. 하위 도메인을 사용하는 경우, 회사의 프로시저에 따라 상위 도메인에 위임 레코드를 추가합니다. 이 과정에 회사의 IT 부서 또는 회사의 루트 도메인 및 DNS 서비스를 제어하는 부서에 요청하는 일도 포함될 수 있습니다.

7.11.6.2. GCP에서 VPC 생성

OpenShift Container Platform 클러스터에서 사용할 VPC를 GCP(Google Cloud Platform)에 생성해야 합니다. 요구사항에 맞춰 VPC를 사용자 지정할 수 있습니다. VPC를 생성하는 한 가지 방법은 제공된 Deployment Manager 템플릿을 수정하는 것입니다.



참고

GCP 인프라를 생성하는 데 제공된 Deployment Manager 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- GCP 계정을 구성하십시오.

프로세스

1. 이 항목의 **VPC에 대한 Deployment Manager 템플릿** 섹션에서 템플릿을 복사하여 사용자 컴퓨터에 **01_vpc.py**로 저장합니다. 이 템플릿은 클러스터에 필요한 VPC를 설명합니다.

2. 리소스 정의에 필요한 다음 변수들을 내보냅니다.

   a. 컨트롤 플레인 CIDR을 내보냅니다.

   ```
   $ export MASTER_SUBNET_CIDR='10.0.0.0/17'
   ```

   b. 컴퓨팅 CIDR을 내보냅니다.

   ```
   $ export WORKER_SUBNET_CIDR='10.0.128.0/17'
   ```

   c. VPC 네트워크와 클러스터를 배포할 리전을 내보냅니다.

   ```
   $ export REGION='<region>'
   ```

3. 공유 VPC를 호스팅하는 프로젝트의 ID에 해당하는 변수를 내보냅니다.

   ```
   $ export HOST_PROJECT=<host_project>
   ```

4. 호스트 프로젝트에 속하는 서비스 계정의 이메일에 해당하는 변수를 내보냅니다.

   ```
   $ export HOST_PROJECT_ACCOUNT=<host_service_account_email>
   ```

5. **01_vpc.yaml** 리소스 정의 파일을 생성합니다.

   ```
   $ cat <<EOF >01_vpc.yaml
   ```

```
imports:
- path: 01_vpc.py

resources:
- name: cluster-vpc
  type: 01_vpc.py
  properties:
    infra_id: '<prefix>' 1
    region: '${REGION}' 2
    master_subnet_cidr: '${MASTER_SUBNET_CIDR}' 3
    worker_subnet_cidr: '${WORKER_SUBNET_CIDR}' 4
EOF
```

1. **infra_id**는 네트워크 이름의 접두사입니다.
2. **region**은 클러스터를 배포할 영역입니다(예: **us-central1**).
3. **master_subnet_cidr**은 마스터 서브넷의 CIDR입니다. (예: **10.0.0.0/17**)
4. **worker_subnet_cidr**은 작업자 서브넷의 CIDR입니다. (예: **10.0.128.0/17**)

6. **gcloud** CLI를 사용하여 배포를 생성합니다.

```
$ gcloud deployment-manager deployments create <vpc_deployment_name> --config
01_vpc.yaml --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT} 1
```

1. **<vpc_deployment_name>** 값으로 배포할 VPC의 이름을 지정합니다.

7. 다른 구성 요소에 필요한 VPC 변수들을 내보냅니다.
   a. 호스트 프로젝트 네트워크의 이름을 내보냅니다.

   ```
   $ export HOST_PROJECT_NETWORK=<vpc_network>
   ```

   b. 호스트 프로젝트 컨트롤 플레인의 이름을 내보냅니다.

   ```
   $ export HOST_PROJECT_CONTROL_SUBNET=<control_plane_subnet>
   ```

   c. 호스트 프로젝트 컴퓨팅 서브넷의 이름을 내보냅니다.

   ```
   $ export HOST_PROJECT_COMPUTE_SUBNET=<compute_subnet>
   ```

8. 공유 VPC를 설정합니다. GCP 문서의 공유 VPC 설정 단원을 참조하십시오.

7.11.6.2.1. VPC용 Deployment Manager 템플릿

다음 Deployment Manager 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 VPC를 배포할 수 있습니다.

예 7.10. 01_vpc.py Deployment Manager 템플릿

```
def GenerateConfig(context):
```

```
resources = [{
    'name': context.properties['infra_id'] + '-network',
    'type': 'compute.v1.network',
    'properties': {
        'region': context.properties['region'],
        'autoCreateSubnetworks': False
    }
}, {
    'name': context.properties['infra_id'] + '-master-subnet',
    'type': 'compute.v1.subnetwork',
    'properties': {
        'region': context.properties['region'],
        'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
        'ipCidrRange': context.properties['master_subnet_cidr']
    }
}, {
    'name': context.properties['infra_id'] + '-worker-subnet',
    'type': 'compute.v1.subnetwork',
    'properties': {
        'region': context.properties['region'],
        'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
        'ipCidrRange': context.properties['worker_subnet_cidr']
    }
}, {
    'name': context.properties['infra_id'] + '-router',
    'type': 'compute.v1.router',
    'properties': {
        'region': context.properties['region'],
        'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
        'nats': [{
            'name': context.properties['infra_id'] + '-nat-master',
            'natIpAllocateOption': 'AUTO_ONLY',
            'minPortsPerVm': 7168,
            'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
            'subnetworks': [{
                'name': '$(ref.' + context.properties['infra_id'] + '-master-subnet.selfLink)',
                'sourceIpRangesToNat': ['ALL_IP_RANGES']
            }]
        }, {
            'name': context.properties['infra_id'] + '-nat-worker',
            'natIpAllocateOption': 'AUTO_ONLY',
            'minPortsPerVm': 512,
            'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
            'subnetworks': [{
                'name': '$(ref.' + context.properties['infra_id'] + '-worker-subnet.selfLink)',
                'sourceIpRangesToNat': ['ALL_IP_RANGES']
            }]
        }]
    }
}]

return {'resources': resources}
```

7.11.7. GCP용 설치 파일 생성

사용자 프로비저닝 인프라를 사용하는 OpenShift Container Platform을 GCP(Google Cloud Platform)에 설치하려면 설치 프로그램에서 클러스터를 배포하는 데 필요한 파일을 생성하고 클러스터가 사용할 시스템만을 생성하도록 파일을 수정해야 합니다. **install-config.yaml** 파일, Kubernetes 매니페스트 및 Ignition 구성 파일을 생성하고 사용자 지정합니다. 또한 설치 준비 단계에서 별도의 **var** 파티션을 먼저 설정할 수 있는 옵션이 있습니다.

7.11.7.1. 수동으로 설치 구성 파일 만들기

OpenShift Container Platform을 사용자가 프로비저닝한 설치의 경우 설치 구성 파일을 수동으로 생성합니다.

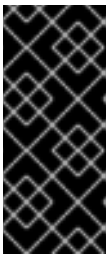
사전 요구 사항

- 로컬 시스템에 설치 프로그램에 제공할 SSH 공개 키가 있습니다. 키는 디버깅 및 재해 복구를 위해 클러스터 노드에 대한 SSH 인증에 사용됩니다.
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿이 있습니다.

프로세스

1. 필요한 설치 자산을 저장할 설치 디렉터리를 만듭니다.

```
$ mkdir <installation_directory>
```



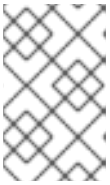
중요

디렉터리를 만들어야 합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

2. 샘플 **install-config.yaml** 파일 템플릿을 사용자 지정하여 **<installation_directory>**에 저장합니다.

참고

이 설정 파일의 이름을 **install-config.yaml**로 지정해야 합니다.



참고

일부 플랫폼 유형의 경우 대체로 **./openshift-install create install-config --dir <installation_directory>** 를 실행하여 **install-config.yaml** 파일을 생성할 수 있습니다. 프롬프트에서 클러스터 구성에 대한 세부 정보를 제공할 수 있습니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.

중요

install-config.yaml 파일은 설치 과정의 다음 단계에서 사용됩니다. 이 시점에서 이를 백업해야 합니다.

7.11.7.2. GCP용 샘플 사용자 지정 install-config.yaml 파일

install-config.yaml 파일을 사용자 지정하여 OpenShift Container Platform 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.

중요

이 샘플 YAML 파일은 참조용으로만 제공됩니다. 설치 프로그램을 사용하여 **install-config.yaml** 파일을 받아서 수정해야 합니다.

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2)
  hyperthreading: Enabled (3) (4)
  name: master
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
  replicas: 3
compute: (5)
- hyperthreading: Enabled (6)
  name: worker
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
  replicas: 0
metadata:
  name: test-cluster
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  gcp:
    projectID: openshift-production (7)
    region: us-central1 (8)
pullSecret: '{"auths": ...}'
fips: false (9)
sshKey: ssh-ed25519 AAAA... (10)
publish: Internal (11)
```

(1) 호스트 프로젝트에서 퍼블릭 DNS를 지정합니다.

2 5 이러한 매개변수와 값을 지정하지 않으면 설치 프로그램은 기본값을 적용합니다.

3 6 **controlPlane** 섹션은 단일 매핑이지만 compute 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 현재 두 섹션이 모두 단일 시스템 풀을 정의하지만 향후 출시되는 OpenShift Container Platform 버전은 설치 과정에서 여러 컴퓨팅 풀 정의를 지원할 수 있습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

4 동시 멀티스레딩 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. 일부 클러스터 시스템에서 동시 멀티스레딩을 비활성화할 경우에는 해당 멀티스레딩을 모든 클러스터 시스템에서 비활성화해야 합니다.

중요

동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다. 동시 멀티스레딩을 비활성화하는 경우, 사용자 시스템에 더 큰 시스템 유형(예: **n1-standard-8**)을 사용하십시오.

7 VM 인스턴스가 있는 기본 프로젝트를 지정합니다.

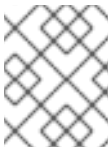
8 VPC 네트워크가 속한 리전을 지정합니다.

9 FIPS 모드 활성화 또는 비활성화 여부입니다. 기본적으로 FIPS 모드는 비활성화됩니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.

중요

FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.

10 선택사항으로, 클러스터의 시스템에 액세스하는 데 사용할 **sshKey** 값을 제공할 수도 있습니다.



참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

11 클러스터의 사용자 끝점을 게시하는 방법. 인터넷에서 액세스할 수 없는 프라이빗 클러스터를 배포하려면 **publish**를 **Internal**로 설정합니다. 기본값은 **External**입니다. 프로비저닝하는 인프라를 사용하는 클러스터에서 공용 VPC를 사용하려면 **publish**를 **Internal**로 설정해야 합니다. 그러면 이제 설치 프로그램에서 호스트 프로젝트의 기본 도메인에 대한 퍼블릭 DNS 영역에 액세스할 수 없습니다.

7.11.7.3. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 HTTP 또는 HTTPS 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 OpenShift Container Platform 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 API에 대한 호출을 포함하여 모든 클러스터 발신(Egress) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.

참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

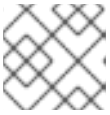
Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 Red Hat OpenStack Platform (RHOSP)에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

프로세스

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. 클러스터 외부에서 HTTP 연결을 구축하는 데 사용할 프록시 URL입니다. URL 스키마는 **http**여야 합니다.
2. 클러스터 외부에서 HTTPS 연결을 구축하는 데 사용할 프록시 URL입니다.
3. 대상 도메인 이름, IP 주소 또는 프록시에서 제외할 기타 네트워크 CIDR로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. ***를 사용하여 모든 대상에 대해 프록시를 바이패스합니다.
4. 이 값을 제공하면 설치 프로그램에서 추가 CA 인증서를 보유할 **openshift -config 네임스페이스에 user-ca- bundle** 이라는 구성 맵을 생성합니다. **additionalTrustBundle** 및 하나 이상의 프록시 설정을 제공하는 경우 **프록시** 오브젝트는 **trustedCA** 필드의 **user-ca-bundle** 구성 맵을 참조하도록 구성됩니다. 그러면 Cluster Network Operator에서 **trustedCA** 매개변수에 대해 지정된 콘텐츠를 RHCOS 신뢰 번들과 병합하는 **trusted-ca-bundle** 구성 맵을 생성합니다. 프록시의 ID 인증서를 RHCOS 트러스트 번들에 있는 기관에서 서명하지 않은 경우 **additionalTrustBundle** 필드가 있어야 합니다.



참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 OpenShift Container Platform을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 nil이 됩니다.

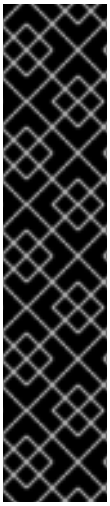
참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

7.11.7.4. Kubernetes 매니페스트 및 Ignition 설정 파일 생성

일부 클러스터 정의 파일을 수정하고 클러스터 시스템을 수동으로 시작해야 하므로 클러스터가 시스템을 구성하는 데 필요한 Kubernetes 매니페스트 및 Ignition 구성 파일을 사용자가 생성해야 합니다.

설치 구성 파일은 Kubernetes 매니페스트로 변환됩니다. 매니페스트는 나중에 클러스터 머신을 구성하는 데 사용되는 Ignition 구성 파일로 래핑됩니다.



중요

- OpenShift Container Platform 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램을 가져오셨습니다.
- **install-config.yaml** 설치 구성 파일을 생성하셨습니다.

절차

1. OpenShift Container Platform 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터에 대한 Kubernetes 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>**는 사용자가 만든 **install-config.yaml** 파일이 포함된 설치 디렉터리를 지정합니다.

2. 컨트롤 플레인 시스템을 정의하는 Kubernetes 매니페스트 파일을 제거합니다.

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

이러한 파일을 제거하면 클러스터가 컨트롤 플레인 시스템을 자동으로 생성하지 못합니다.

3. 작업자 시스템을 정의하는 Kubernetes 매니페스트 파일을 제거합니다.

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

작업자 시스템은 사용자가 직접 생성하고 관리하기 때문에 초기화할 필요가 없습니다.

4. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes 매니페스트 파일의 **mastersSchedulable** 매개변수가 **false**로 설정되어 있는지 확인합니다. 이 설정으로 인해 컨트롤 플레인 머신에서 포드가 예약되지 않습니다.

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** 파일을 엽니다.

   b. **mastersSchedulable** 매개변수를 찾아서 값을 **False**로 설정되어 있는지 확인합니다.

   c. 파일을 저장하고 종료합니다.

5. **<installation_directory>/manifests/cluster-dns-02-config.yml** DNS 구성 파일에서 **privateZone** 섹션을 제거합니다.

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
status: {}
```

1 이 섹션을 완전히 제거합니다.

6. VPC에 대한 클라우드 공급자를 구성합니다.

   a. **<installation_directory>/manifests/cloud-provider-config.yaml** 파일을 엽니다.

   b. **network-project-id** 매개변수를 추가하고 공유 VPC 네트워크를 호스팅하는 프로젝트의 ID를 매개변수 값으로 설정합니다.

   c. **network-name** 매개변수를 추가하고 OpenShift Container Platform 클러스터를 호스팅하는 공유 VPC 네트워크의 이름을 매개변수 값으로 설정합니다.

   d. **subnetwork-name** 매개변수의 값을 컴퓨팅 시스템을 호스팅하는 공유 VPC 서브넷의 값으로 변경합니다.

<installation_directory>/manifests/cloud-provider-config.yaml의 내용은 다음 예와 유사합니다.

```
config: |+
```

```
[global]
project-id     = example-project
regional       = true
multizone      = true
node-tags      = opensh-ptzzx-master
node-tags      = opensh-ptzzx-worker
node-instance-prefix = opensh-ptzzx
external-instance-groups-prefix = opensh-ptzzx
network-project-id = example-shared-vpc
network-name    = example-network
subnetwork-name = example-worker-subnet
```

7. 프라이빗 네트워크에 없는 클러스터를 배포하는 경우 **<installation_directory>/manifests/cluster-ingress-default-ingresscontroller.yaml** 파일을 열고 **scope** 매개변수 값을 **External**로 변경합니다. 파일의 내용은 다음 예와 유사합니다.

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  creationTimestamp: null
  name: default
  namespace: openshift-ingress-operator
spec:
  endpointPublishingStrategy:
    loadBalancer:
      scope: External
    type: LoadBalancerService
status:
  availableReplicas: 0
  domain: ''
  selector: ''
```

8. Ignition 구성 파일을 생성하려면 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행합니다.

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>**는 동일한 설치 디렉터리를 지정합니다.

설치 디렉터리의 부트스트랩, 컨트롤 플레인 및 컴퓨팅 노드에 대한 Ignition 구성 파일이 생성됩니다. **kubeadmin-password** 및 **kubeconfig** 파일은 **./<installation_directory>/auth** 디렉터리에 생성됩니다.

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

7.11.8. 공통 변수 내보내기

7.11.8.1. 인프라 이름 추출

Ignition 구성 파일에는 GCP(Google Cloud Platform)에서 클러스터를 고유하게 식별하는 데 사용할 수 있는 고유한 클러스터 ID가 포함되어 있습니다. 인프라 이름은 OpenShift Container Platform 설치 중에 적절한 GCP 리소스를 찾는 데도 사용됩니다. 제공된 Deployment Manager 템플릿에 이 인프라 이름에 대한 참조가 포함되어 있으므로 이름을 추출해야 합니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받으셨습니다.
- 클러스터에 대한 Ignition 구성 파일을 생성하셨습니다.
- **jq** CLI를 설치하셨습니다.

절차

- Ignition 구성 파일 메타데이터에서 인프라 이름을 추출하여 확인하려면 다음 명령을 실행하십시오.

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

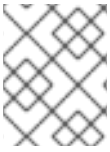
출력 예

```
openshift-vw9j6 1
```

1 이 명령의 출력은 클러스터 이름과 임의의 문자열입니다.

7.11.8.2. Deployment Manager 템플릿의 공통 변수 내보내기

GCP(Google Cloud Platform)에서 사용자 제공 인프라 설치를 지원하기 위해 제공된 Deployment Manager 템플릿과 함께 사용되는 공통 변수 세트를 내보내야 합니다.



참고

특정 Deployment Manager 템플릿에는 추가적으로 내보낸 변수도 필요할 수 있습니다. 관련 프로시저에서 자세히 설명합니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.
- **jq** 패키지를 설치합니다.

프로세스

1. 제공된 Deployment Manager 템플릿에서 사용할 다음 공통 변수를 내보냅니다.

```
$ export BASE_DOMAIN='<base_domain>' 1
$ export BASE_DOMAIN_ZONE_NAME='<base_domain_zone_name>' 2
$ export NETWORK_CIDR='10.0.0.0/16'

$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 3
$ export CLUSTER_NAME=`jq -r .clusterName <installation_directory>/metadata.json`
$ export INFRA_ID=`jq -r .infraID <installation_directory>/metadata.json`
$ export PROJECT_NAME=`jq -r .gcp.projectID <installation_directory>/metadata.json`
```

1 2 호스트 프로젝트의 값을 제공합니다.

3 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

7.11.9. 사용자 프로비저닝 인프라에 대한 네트워킹 요구사항

모든 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템이 부팅 중에 Ignition 구성 파일을 가져오려면 **initramfs**에 네트워킹을 구성해야 합니다.

7.11.9.1. DHCP를 통해 클러스터 노드의 호스트 이름 설정

RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 호스트 이름은 NetworkManager를 통해 설정됩니다. 기본적으로 시스템은 DHCP를 통해 호스트 이름을 가져옵니다. DHCP에서 호스트 이름을 제공하지 않거나, 커널 인수를 통해 정적으로 설정하거나 다른 방법을 통해 설정하면 역방향 DNS 조회를 통해 얻을 수 있습니다. 역방향 DNS 조회는 노드에서 네트워크를 초기화한 후 수행되며 확인하는 데 시간이 걸릴 수 있습니다. 다른 시스템 서비스는 이 보다 먼저 시작하여 호스트 이름을 **localhost** 등으로 감지할 수 있습니다. DHCP를 사용하여 각 클러스터 노드의 호스트 이름을 제공하여 이 문제를 방지할 수 있습니다.

또한 DHCP를 통해 호스트 이름을 설정하면 DNS 분할 수평 구현 환경에서 수동으로 DNS 레코드 이름 구성 오류를 무시할 수 있습니다.

7.11.9.2. 네트워크 연결 요구사항

OpenShift Container Platform 클러스터 구성 요소가 통신할 수 있도록 시스템 간 네트워크 연결을 구성해야 합니다. 각 시스템에서 클러스터에 있는 다른 모든 시스템의 호스트 이름을 확인할 수 있어야 합니다.

이 섹션에서는 필요한 포트에 대해 자세히 설명합니다.

중요

연결된 OpenShift Container Platform 환경에서 모든 노드는 플랫폼 컨테이너의 이미지를 가져오고 Red Hat에 원격 측정 데이터를 제공하기 위해 인터넷에 액세스할 수 있어야 합니다.

표 7.48. 모든 시스템 간 통신에 사용되는 포트

프로토콜	포트	설명
ICMP	해당 없음	네트워크 연결성 테스트
TCP	**1936**	메트릭

프로토콜	포트	설명
	9000-9999	**9100-9101** 포트의 노드 내보내기 및 **9099** 포트의 Cluster Version Operator를 포함한 호스트 수준 서비스.
	10250-10259	Kubernetes에서 예약하는 기본 포트
	10256	openshift-sdn
UDP	**4789**	VXLAN 및 Geneve
	6081	VXLAN 및 Geneve
	9000-9999	**9100-9101** 포트의 노드 내보내기를 포함한 호스트 수준 서비스.
	500	IPsec IKE 패킷
	4500	IPsec NAT-T 패킷
TCP/UDP	**30000-32767**	Kubernetes 노드 포트
ESP	해당 없음	IPsec Encapsulating Security Payload (ESP)

표 7.49. 모든 시스템과 컨트롤 플레인 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**6443**	Kubernetes API

표 7.50. 컨트롤 플레인 머신 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**2379-2380**	etcd 서버 및 피어 포트

7.11.10. GCP에서 로드 밸런서 생성

OpenShift Container Platform 클러스터가 사용할 로드 밸런서를 GCP(Google Cloud Platform)에 구성해야 합니다. 이러한 구성 요소를 생성하는 한 가지 방법은 제공된 Deployment Manager 템플릿을 수정하는 것입니다.

참고

GCP 인프라를 생성하는 데 제공된 Deployment Manager 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- GCP 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.
- GCP에서 VPC 및 관련 서브넷을 생성하고 구성하십시오.

프로세스

1. 이 항목의 **내부 로드 밸런서에 대한 Deployment Manager 템플릿** 섹션에서 템플릿을 복사하여 사용자 컴퓨터에 **02_lb_int.py**로 저장합니다. 이 템플릿에서 클러스터에 필요한 내부 로드 밸런싱 개체를 설명합니다.
2. 외부 클러스터에 대해서도, 이 항목의 **외부 로드 밸런서에 대한 Deployment Manager 템플릿** 섹션에서 템플릿을 복사하여 사용자 컴퓨터에 **02_lb_ext.py**로 저장합니다. 이 템플릿에서 클러스터에 필요한 외부 로드 밸런싱 개체를 설명합니다.
3. 배포 템플릿이 사용하는 변수를 내보냅니다.
   a. 클러스터 네트워크 위치를 내보냅니다.

```
$ export CLUSTER_NETWORK=(`gcloud compute networks describe
${HOST_PROJECT_NETWORK} --project ${HOST_PROJECT} --account
${HOST_PROJECT_ACCOUNT} --format json | jq -r .selfLink`)
```

   b. 컨트롤 플레인 서브넷 위치를 내보냅니다.

```
$ export CONTROL_SUBNET=(`gcloud compute networks subnets describe
${HOST_PROJECT_CONTROL_SUBNET} --region=${REGION} --project
${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT} --format json | jq -r
.selfLink`)
```

   c. 클러스터가 사용하는 세 영역을 내보냅니다.

```
$ export ZONE_0=(`gcloud compute regions describe ${REGION} --format=json | jq -r
.zones[0] | cut -d "/" -f9`)
```

```
$ export ZONE_1=(`gcloud compute regions describe ${REGION} --format=json | jq -r
.zones[1] | cut -d "/" -f9`)
```

```
$ export ZONE_2=(`gcloud compute regions describe ${REGION} --format=json | jq -r
.zones[2] | cut -d "/" -f9`)
```

4. **02_infra.yaml** 리소스 정의 파일을 생성합니다.

```
$ cat <<EOF >02_infra.yaml
imports:
- path: 02_lb_ext.py
- path: 02_lb_int.py 1
resources:
- name: cluster-lb-ext 2
  type: 02_lb_ext.py
  properties:
```

```
    infra_id: '${INFRA_ID}' 3
    region: '${REGION}' 4
- name: cluster-lb-int
  type: 02_lb_int.py
  properties:
    cluster_network: '${CLUSTER_NETWORK}'
    control_subnet: '${CONTROL_SUBNET}' 5
    infra_id: '${INFRA_ID}'
    region: '${REGION}'
    zones: 6
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'
EOF
```

1 2 외부 클러스터를 배포하는 경우에만 필요합니다.

3 **infra_id**는 추출 단계에서 가져온 **INFRA_ID** 인프라 이름입니다.

4 **region**은 클러스터를 배포할 영역입니다(예: **us-central1**).

5 **control_subnet**은 컨트롤 서브넷에 대한 URI입니다.

6 **zones**는 **us-east1-b**, **us-east1-c** 및 **us-east1-d**와 같이 컨트롤 플레인 인스턴스를 배포하는 영역입니다.

5. **gcloud** CLI를 사용하여 배포를 생성합니다.

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-infra --config 02_infra.yaml
```

6. 클러스터 IP 주소를 내보냅니다.

```
$ export CLUSTER_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-ip --region=${REGION} --format json | jq -r .address`)
```

7. 외부 클러스터의 경우 클러스터 공용 IP 주소도 내보냅니다.

```
$ export CLUSTER_PUBLIC_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-public-ip --region=${REGION} --format json | jq -r .address`)
```

7.11.10.1. 외부 로드 밸런서에 대한 Deployment Manager 템플릿

다음 Deployment Manager 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 외부 로드 밸런서를 배포할 수 있습니다.

예 7.11. 02_lb_ext.py Deployment Manager 템플릿

```
def GenerateConfig(context):

  resources = [{
    'name': context.properties['infra_id'] + '-cluster-public-ip',
    'type': 'compute.v1.address',
    'properties': {
```

```
            'region': context.properties['region']
        }
    }, {
        # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check
probe for kube-apiserver
        'name': context.properties['infra_id'] + '-api-http-health-check',
        'type': 'compute.v1.httpHealthCheck',
        'properties': {
            'port': 6080,
            'requestPath': '/readyz'
        }
    }, {
        'name': context.properties['infra_id'] + '-api-target-pool',
        'type': 'compute.v1.targetPool',
        'properties': {
            'region': context.properties['region'],
            'healthChecks': ['$(ref.' + context.properties['infra_id'] + '-api-http-health-check.selfLink)'],
            'instances': []
        }
    }, {
        'name': context.properties['infra_id'] + '-api-forwarding-rule',
        'type': 'compute.v1.forwardingRule',
        'properties': {
            'region': context.properties['region'],
            'IPAddress': '$(ref.' + context.properties['infra_id'] + '-cluster-public-ip.selfLink)',
            'target': '$(ref.' + context.properties['infra_id'] + '-api-target-pool.selfLink)',
            'portRange': '6443'
        }
    }]

    return {'resources': resources}
```

7.11.10.2. 내부 로드 밸런서에 대한 Deployment Manager 템플릿

다음 Deployment Manager 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 내부 로드 밸런서를 배포할 수 있습니다.

예 7.12. 02_lb_int.py Deployment Manager 템플릿

```
def GenerateConfig(context):

    backends = []
    for zone in context.properties['zones']:
        backends.append({
            'group': '$(ref.' + context.properties['infra_id'] + '-master-' + zone + '-instance-group' +
'.selfLink)'
        })

    resources = [{
        'name': context.properties['infra_id'] + '-cluster-ip',
        'type': 'compute.v1.address',
        'properties': {
            'addressType': 'INTERNAL',
            'region': context.properties['region'],
```

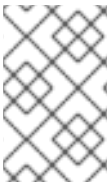
```
            'subnetwork': context.properties['control_subnet']
        }
    }, {
        # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check
probe for kube-apiserver
        'name': context.properties['infra_id'] + '-api-internal-health-check',
        'type': 'compute.v1.healthCheck',
        'properties': {
            'httpsHealthCheck': {
                'port': 6443,
                'requestPath': '/readyz'
            },
            'type': "HTTPS"
        }
    }, {
        'name': context.properties['infra_id'] + '-api-internal-backend-service',
        'type': 'compute.v1.regionBackendService',
        'properties': {
            'backends': backends,
            'healthChecks': ['$(ref.' + context.properties['infra_id'] + '-api-internal-health-
check.selfLink)'],
            'loadBalancingScheme': 'INTERNAL',
            'region': context.properties['region'],
            'protocol': 'TCP',
            'timeoutSec': 120
        }
    }, {
        'name': context.properties['infra_id'] + '-api-internal-forwarding-rule',
        'type': 'compute.v1.forwardingRule',
        'properties': {
            'backendService': '$(ref.' + context.properties['infra_id'] + '-api-internal-backend-
service.selfLink)',
            'IPAddress': '$(ref.' + context.properties['infra_id'] + '-cluster-ip.selfLink)',
            'loadBalancingScheme': 'INTERNAL',
            'ports': ['6443','22623'],
            'region': context.properties['region'],
            'subnetwork': context.properties['control_subnet']
        }
    }]

    for zone in context.properties['zones']:
        resources.append({
            'name': context.properties['infra_id'] + '-master-' + zone + '-instance-group',
            'type': 'compute.v1.instanceGroup',
            'properties': {
                'namedPorts': [
                    {
                        'name': 'ignition',
                        'port': 22623
                    }, {
                        'name': 'https',
                        'port': 6443
                    }
                ],
                'network': context.properties['cluster_network'],
                'zone': zone
```

```
        }
    })

    return {'resources': resources}
```

외부 클러스터를 생성할 때 **02_lb_ext.py** 템플릿 외에도 이 템플릿이 필요합니다.

7.11.11. GCP에서 프라이빗 DNS 영역 생성

OpenShift Container Platform 클러스터가 사용할 프라이빗 DNS 영역을 GCP(Google Cloud Platform)에 구성해야 합니다. 이 구성 요소를 생성하는 한 가지 방법은 제공된 Deployment Manager 템플릿을 수정하는 것입니다.



참고

GCP 인프라를 생성하는 데 제공된 Deployment Manager 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- GCP 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.
- GCP에서 VPC 및 관련 서브넷을 생성하고 구성하십시오.

프로세스

1. 이 항목의 **프라이빗 DNS에 대한 Deployment Manager 템플릿** 섹션에서 템플릿을 복사하여 사용자 컴퓨터에 **02_dns.py**로 저장합니다. 이 템플릿에서 클러스터에 필요한 프라이빗 DNS 개체를 설명합니다.
2. **02_dns.yaml** 리소스 정의 파일을 생성합니다.

```
$ cat <<EOF >02_dns.yaml
imports:
- path: 02_dns.py

resources:
- name: cluster-dns
  type: 02_dns.py
  properties:
    infra_id: '${INFRA_ID}' (1)
    cluster_domain: '${CLUSTER_NAME}.${BASE_DOMAIN}' (2)
    cluster_network: '${CLUSTER_NETWORK}' (3)
EOF
```

(1) **infra_id**는 추출 단계에서 가져온 **INFRA_ID** 인프라 이름입니다.

(2) **cluster_domain**은 클러스터의 도메인입니다(예: **openshift.example.com**).

(3) **cluster_network**는 클러스터 네트워크에 대한 **selfLink** URL입니다.

3. **gcloud** CLI를 사용하여 배포를 생성합니다.

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-dns --config 02_dns.yaml --
project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
```

4. Deployment Manager의 제한으로 인해 템플릿을 통해 DNS 항목이 생성되지 않으므로 수동으로 생성해야 합니다.

   a. 내부 DNS 항목을 추가합니다.

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone --project
${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
$ gcloud dns record-sets transaction add ${CLUSTER_IP} --name
api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-
private-zone --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
$ gcloud dns record-sets transaction add ${CLUSTER_IP} --name api-
int.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-
private-zone --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
$ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone --project
${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
```

   b. 외부 클러스터의 경우 외부 DNS 항목도 추가합니다.

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT} dns
record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME}
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT} dns
record-sets transaction add ${CLUSTER_PUBLIC_IP} --name
api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone
${BASE_DOMAIN_ZONE_NAME}
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT} dns
record-sets transaction execute --zone ${BASE_DOMAIN_ZONE_NAME}
```

7.11.11.1. 프라이빗 DNS에 대한 Deployment Manager 템플릿

다음 Deployment Manager 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 프라이빗 DNS를 배포할 수 있습니다.

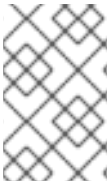
예 7.13. 02_dns.py Deployment Manager 템플릿

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-private-zone',
        'type': 'dns.v1.managedZone',
        'properties': {
            'description': '',
            'dnsName': context.properties['cluster_domain'] + '.',
            'visibility': 'private',
            'privateVisibilityConfig': {
                'networks': [{
                    'networkUrl': context.properties['cluster_network']
                }]
```

```
        }
    }
}]

return {'resources': resources}
```

7.11.12. GCP에서 방화벽 규칙 생성

OpenShift Container Platform 클러스터에서 사용할 방화벽 규칙을 GCP(Google Cloud Platform)에서 생성해야 합니다. 이러한 구성 요소를 생성하는 한 가지 방법은 제공된 Deployment Manager 템플릿을 수정하는 것입니다.



참고

GCP 인프라를 생성하는 데 제공된 Deployment Manager 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- GCP 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.
- GCP에서 VPC 및 관련 서브넷을 생성하고 구성하십시오.

프로세스

1. 이 항목의 **방화벽 규칙에 대한 Deployment Manager 템플릿** 섹션에서 템플릿을 복사하여 사용자 컴퓨터에 **03_firewall.py**로 저장합니다. 이 템플릿에서 클러스터에 필요한 보안 그룹을 설명합니다.
2. **03_firewall.yaml** 리소스 정의 파일을 생성합니다.

```
$ cat <<EOF >03_firewall.yaml
imports:
- path: 03_firewall.py

resources:
- name: cluster-firewall
  type: 03_firewall.py
  properties:
    allowed_external_cidr: '0.0.0.0/0' (1)
    infra_id: '${INFRA_ID}' (2)
    cluster_network: '${CLUSTER_NETWORK}' (3)
    network_cidr: '${NETWORK_CIDR}' (4)
EOF
```

(1) **allowed_external_cidr**은 부트스트랩 호스트에 클러스터 API 및 SSH 액세스가 가능한 CIDR 범위입니다. 내부 클러스터의 경우 이 값을 **${NETWORK_CIDR}**로 설정합니다.

(2) **infra_id**는 추출 단계에서 가져온 **INFRA_ID** 인프라 이름입니다.

3. **cluster_network**는 클러스터 네트워크에 대한 **selfLink** URL입니다.
4. **network_cidr**은 VPC 네트워크의 CIDR입니다(예: **10.0.0.0/16**).

3. **gcloud** CLI를 사용하여 배포를 생성합니다.

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-firewall --config
03_firewall.yaml --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
```

7.11.12.1. 방화벽 규칙에 대한 Deployment Manager 템플릿

다음 Deployment Manager 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 방화벽 규칙을 배포할 수 있습니다.

예 7.14. 03_firewall.py Deployment Manager 템플릿

```
def GenerateConfig(context):

  resources = [{
    'name': context.properties['infra_id'] + '-bootstrap-in-ssh',
    'type': 'compute.v1.firewall',
    'properties': {
      'network': context.properties['cluster_network'],
      'allowed': [{
        'IPProtocol': 'tcp',
        'ports': ['22']
      }],
      'sourceRanges': [context.properties['allowed_external_cidr']],
      'targetTags': [context.properties['infra_id'] + '-bootstrap']
    }
  }, {
    'name': context.properties['infra_id'] + '-api',
    'type': 'compute.v1.firewall',
    'properties': {
      'network': context.properties['cluster_network'],
      'allowed': [{
        'IPProtocol': 'tcp',
        'ports': ['6443']
      }],
      'sourceRanges': [context.properties['allowed_external_cidr']],
      'targetTags': [context.properties['infra_id'] + '-master']
    }
  }, {
    'name': context.properties['infra_id'] + '-health-checks',
    'type': 'compute.v1.firewall',
    'properties': {
      'network': context.properties['cluster_network'],
      'allowed': [{
        'IPProtocol': 'tcp',
        'ports': ['6080', '6443', '22624']
      }],
      'sourceRanges': ['35.191.0.0/16', '130.211.0.0/22', '209.85.152.0/22', '209.85.204.0/22'],
      'targetTags': [context.properties['infra_id'] + '-master']
    }
```

```
    }, {
        'name': context.properties['infra_id'] + '-etcd',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['2379-2380']
            }],
            'sourceTags': [context.properties['infra_id'] + '-master'],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-control-plane',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['10257']
            },{
                'IPProtocol': 'tcp',
                'ports': ['10259']
            },{
                'IPProtocol': 'tcp',
                'ports': ['22623']
            }],
            'sourceTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-internal-network',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'icmp'
            },{
                'IPProtocol': 'tcp',
                'ports': ['22']
            }],
            'sourceRanges': [context.properties['network_cidr']],
            'targetTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ]
        }
    }, {
        'name': context.properties['infra_id'] + '-internal-cluster',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
```

```
            'IPProtocol': 'udp',
            'ports': ['4789', '6081']
        },{
            'IPProtocol': 'udp',
            'ports': ['500', '4500']
        },{
            'IPProtocol': 'esp',
        },{
            'IPProtocol': 'tcp',
            'ports': ['9000-9999']
        },{
            'IPProtocol': 'udp',
            'ports': ['9000-9999']
        },{
            'IPProtocol': 'tcp',
            'ports': ['10250']
        },{
            'IPProtocol': 'tcp',
            'ports': ['30000-32767']
        },{
            'IPProtocol': 'udp',
            'ports': ['30000-32767']
        }],
        'sourceTags': [
            context.properties['infra_id'] + '-master',
            context.properties['infra_id'] + '-worker'
        ],
        'targetTags': [
            context.properties['infra_id'] + '-master',
            context.properties['infra_id'] + '-worker'
        ]
    }
}]

return {'resources': resources}
```

7.11.13. GCP에서 IAM 역할 생성

OpenShift Container Platform 클러스터에서 사용할 IAM 역할을 GCP(Google Cloud Platform)에 생성해야 합니다. 이러한 구성 요소를 생성하는 한 가지 방법은 제공된 Deployment Manager 템플릿을 수정하는 것입니다.

참고

GCP 인프라를 생성하는 데 제공된 Deployment Manager 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- GCP 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.

- GCP에서 VPC 및 관련 서브넷을 생성하고 구성하십시오.

프로세스

1. 이 항목의 **IAM 역할에 대한 Deployment Manager 템플릿** 섹션에서 템플릿을 복사하여 사용자 컴퓨터에 **03_iam.py**로 저장합니다. 이 템플릿에서 클러스터에 필요한 IAM 역할을 설명합니다.

2. **03_iam.yaml** 리소스 정의 파일을 생성합니다.

```
$ cat <<EOF >03_iam.yaml
imports:
- path: 03_iam.py
resources:
- name: cluster-iam
  type: 03_iam.py
  properties:
    infra_id: '${INFRA_ID}' (1)
EOF
```

(1) **infra_id**는 추출 단계에서 가져온 **INFRA_ID** 인프라 이름입니다.

3. **gcloud** CLI를 사용하여 배포를 생성합니다.

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-iam --config 03_iam.yaml
```

4. 마스터 서비스 계정에 대한 변수를 내보냅니다.

```
$ export MASTER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-m@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

5. 작업자 서비스 계정에 대한 변수를 내보냅니다.

```
$ export WORKER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

6. 설치 프로그램에서 컨트롤 플레인 및 컴퓨팅 서브넷을 호스팅하는 서브넷의 서비스 계정에 필요로 하는 권한을 할당합니다.

   a. 공유 VPC를 호스팅하는 프로젝트의 **networkViewer** 역할을 마스터 서비스 계정에 부여합니다.

```
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
projects add-iam-policy-binding ${HOST_PROJECT} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role
"roles/compute.networkViewer"
```

   b. 컨트롤 플레인 서브넷의 마스터 서비스 계정에 **networkUser** 역할을 부여합니다.

```
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
compute networks subnets add-iam-policy-binding
"${HOST_PROJECT_CONTROL_SUBNET}" --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.networkUser"
--region ${REGION}
```

c. 컨트롤 플레인 서브넷의 작업자 서비스 계정에 **networkUser** 역할을 부여합니다.

```
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
compute networks subnets add-iam-policy-binding
"${HOST_PROJECT_CONTROL_SUBNET}" --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role
"roles/compute.networkUser" --region ${REGION}
```

d. 컴퓨팅 서브넷의 마스터 서비스 계정에 **networkUser** 역할을 부여합니다.

```
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
compute networks subnets add-iam-policy-binding
"${HOST_PROJECT_COMPUTE_SUBNET}" --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.networkUser"
--region ${REGION}
```

e. 컴퓨팅 서브넷의 작업자 서비스 계정에 **networkUser** 역할을 부여합니다.

```
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
compute networks subnets add-iam-policy-binding
"${HOST_PROJECT_COMPUTE_SUBNET}" --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role
"roles/compute.networkUser" --region ${REGION}
```

7. Deployment Manager의 제한으로 인해 템플릿을 통해 정책 바인딩이 생성되지 않으므로 수동으로 생성해야 합니다.

```
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.instanceAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.networkAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.securityAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/iam.serviceAccountUser"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/storage.admin"

$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/compute.viewer"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/storage.admin"
```

8. 서비스 계정 키를 생성하여 나중에 사용할 수 있도록 로컬로 저장합니다.

```
$ gcloud iam service-accounts keys create service-account-key.json --iam-
account=${MASTER_SERVICE_ACCOUNT}
```

7.11.13.1. IAM 역할에 대한 Deployment Manager 템플릿

다음 Deployment Manager 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 IAM 역할을 배포할 수 있습니다.

예 7.15. 03_iam.py Deployment Manager 템플릿

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-master-node-sa',
        'type': 'iam.v1.serviceAccount',
        'properties': {
            'accountId': context.properties['infra_id'] + '-m',
            'displayName': context.properties['infra_id'] + '-master-node'
        }
    }, {
        'name': context.properties['infra_id'] + '-worker-node-sa',
        'type': 'iam.v1.serviceAccount',
        'properties': {
            'accountId': context.properties['infra_id'] + '-w',
            'displayName': context.properties['infra_id'] + '-worker-node'
        }
    }]

    return {'resources': resources}
```

7.11.14. GCP 인프라용 RHCOS 클러스터 이미지 생성

OpenShift Container Platform 노드에 유효한 GCP(Google Cloud Platform)용 RHCOS(Red Hat Enterprise Linux CoreOS) 이미지를 사용해야 합니다.

프로세스

1. RHCOS 이미지 미러 페이지에서 RHCOS 이미지를 가져옵니다.

중요

RHCOS 이미지는 OpenShift Container Platform 릴리스에 따라 변경되지 않을 수 있습니다. 설치하는 OpenShift Container Platform 버전과 같거나 그 이하의 버전 중 가장 최신 버전의 이미지를 다운로드해야 합니다. 지원되는 경우 OpenShift Container Platform 버전과 일치하는 이미지 버전을 사용합니다.

rhcos-<version>-<arch>-gcp.<arch>.tar.gz 형식의 OpenShift Container Platform 버전 번호가 파일 이름에 포함되어 있습니다.

2. Google 스토리지 버킷을 생성합니다.

```
$ gsutil mb gs://<bucket_name>
```

3. RHCOS 이미지를 Google 스토리지 버킷에 업로드합니다.

```
$ gsutil cp <downloaded_image_file_path>/rhcos-<version>-x86_64-gcp.x86_64.tar.gz
gs://<bucket_name>
```

4. 업로드된 RHCOS 이미지 위치를 변수로 내보냅니다.

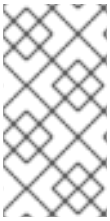
```
$ export IMAGE_SOURCE=gs://<bucket_name>/rhcos-<version>-x86_64-gcp.x86_64.tar.gz
```

5. 클러스터 이미지를 생성합니다.

```
$ gcloud compute images create "${INFRA_ID}-rhcos-image" \
    --source-uri="${IMAGE_SOURCE}"
```

7.11.15. GCP에서 부트스트랩 시스템 생성

OpenShift Container Platform 클러스터 초기화 과정에서 사용할 부트스트랩 시스템을 GCP(Google Cloud Platform)에 생성해야 합니다. 이 시스템을 생성하는 한 가지 방법은 제공된 Deployment Manager 템플릿을 수정하는 것입니다.



참고

부트스트랩 시스템을 생성하는 데 제공된 Deployment Manager 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- GCP 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.
- GCP에서 VPC 및 관련 서브넷을 생성하고 구성하십시오.
- GCP에서 네트워킹 및 로드 밸런서를 생성하고 구성하십시오.
- 컨트롤 플레인 및 컴퓨팅 역할을 생성합니다.
- pyOpenSSL이 설치되어 있는지 확인하십시오.

프로세스

1. 이 항목의 **부트스트랩 시스템에 대한 Deployment Manager 템플릿** 섹션에서 템플릿을 복사하여 사용자 컴퓨터에 **04_bootstrap.py**로 저장합니다. 이 템플릿은 클러스터에 필요한 부트스트랩 시스템을 설명합니다.
2. 설치 프로그램에 필요한 RHCOS(Red Hat Enterprise Linux CoreOS) 이미지의 위치를 내보냅니다.

```
$ export CLUSTER_IMAGE=(`gcloud compute images describe ${INFRA_ID}-rhcos-image --
format json | jq -r .selfLink`)
```

3. 버킷을 생성하고 **bootstrap.ign** 파일을 업로드합니다.

```
$ gsutil mb gs://${INFRA_ID}-bootstrap-ignition
$ gsutil cp <installation_directory>/bootstrap.ign gs://${INFRA_ID}-bootstrap-ignition/
```

4. Ignition 구성 파일에 액세스하는 데 사용할 부트스트랩 인스턴스에 대한 서명된 URL을 생성합니다. 출력에서 URL을 변수로 내보냅니다.

```
$ export BOOTSTRAP_IGN=`gsutil signurl -d 1h service-account-key.json gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign | grep "^gs:" | awk '{print $5}'`
```

5. **04_bootstrap.yaml** 리소스 정의 파일을 생성합니다.

```
$ cat <<EOF >04_bootstrap.yaml
imports:
- path: 04_bootstrap.py

resources:
- name: cluster-bootstrap
  type: 04_bootstrap.py
  properties:
    infra_id: '${INFRA_ID}' (1)
    region: '${REGION}' (2)
    zone: '${ZONE_0}' (3)

    cluster_network: '${CLUSTER_NETWORK}' (4)
    control_subnet: '${CONTROL_SUBNET}' (5)
    image: '${CLUSTER_IMAGE}' (6)
    machine_type: 'n1-standard-4' (7)
    root_volume_size: '128' (8)

    bootstrap_ign: '${BOOTSTRAP_IGN}' (9)
EOF
```

(1) **infra_id**는 추출 단계에서 가져온 **INFRA_ID** 인프라 이름입니다.

(2) **region**은 클러스터를 배포할 영역입니다(예: **us-central1**).

(3) **zone**은 부트스트랩 인스턴스를 배포할 영역입니다(예: **us-central1-b**).

(4) **cluster_network**는 클러스터 네트워크에 대한 **selfLink** URL입니다.

(5) **control_subnet**은 컨트롤 서브넷에 대한 **selfLink** URL입니다.

(6) **image**는 RHCOS 이미지에 대한 **selfLink** URL입니다.

(7) **machine_type**은 인스턴스의 시스템 유형입니다(예: **n1-standard-4**).

(8) **root_volume_size**는 부트스트랩 시스템의 부팅 디스크 크기입니다.

(9) **bootstrap_ign**은 서명된 URL을 생성할 때 출력되는 URL입니다.

6. **gcloud** CLI를 사용하여 배포를 생성합니다.

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-bootstrap --config 04_bootstrap.yaml
```

7. 내부 로드 밸런서 인스턴스 그룹에 부트스트랩 인스턴스를 추가합니다.

```
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-bootstrap-instance-group --zone=${ZONE_0} --instances=${INFRA_ID}-bootstrap
```

8. 내부 로드 밸런서 백엔드 서비스에 부트스트랩 인스턴스 그룹을 추가합니다.

```
$ gcloud compute backend-services add-backend ${INFRA_ID}-api-internal-backend-service
--region=${REGION} --instance-group=${INFRA_ID}-bootstrap-instance-group --instance-
group-zone=${ZONE_0}
```

7.11.15.1. 부트스트랩 시스템에 대한 Deployment Manager 템플릿

다음 Deployment Mananger 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 부트스트랩 시스템을 배포할 수 있습니다.

예 7.16. 04_bootstrap.py Deployment Manager 템플릿

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-bootstrap-public-ip',
        'type': 'compute.v1.address',
        'properties': {
            'region': context.properties['region']
        }
    }, {
        'name': context.properties['infra_id'] + '-bootstrap',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': '{"ignition":{"config":{"replace":{"source":"' + context.properties['bootstrap_ign']
+ '"}},"version":"3.1.0"}}',
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet'],
                'accessConfigs': [{
                    'natIP': '$(ref.' + context.properties['infra_id'] + '-bootstrap-public-ip.address)'
                }]
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                    context.properties['infra_id'] + '-bootstrap'
                ]
            },
            'zone': context.properties['zone']
```

```
        }
    }, {
        'name': context.properties['infra_id'] + '-bootstrap-instance-group',
        'type': 'compute.v1.instanceGroup',
        'properties': {
            'namedPorts': [
                {
                    'name': 'ignition',
                    'port': 22623
                }, {
                    'name': 'https',
                    'port': 6443
                }
            ],
            'network': context.properties['cluster_network'],
            'zone': context.properties['zone']
        }
    }]

    return {'resources': resources}
```

7.11.16. GCP에 컨트롤 플레인 시스템 생성

클러스터에서 사용할 컨트롤 플레인 시스템을 GCP(Google Cloud Platform)에 생성해야 합니다. 이러한 시스템을 생성하는 한 가지 방법은 제공된 Deployment Manager 템플릿을 수정하는 것입니다.

참고

컨트롤 플레인 시스템을 생성하는 데 제공된 Deployment Manager 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- GCP 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.
- GCP에서 VPC 및 관련 서브넷을 생성하고 구성하십시오.
- GCP에서 네트워킹 및 로드 밸런서를 생성하고 구성하십시오.
- 컨트롤 플레인 및 컴퓨팅 역할을 생성합니다.
- 부트스트랩 시스템을 생성합니다.

프로세스

1. 이 항목의 **컨트롤 플레인 시스템에 대한 Deployment Manager 템플릿** 섹션에서 템플릿을 복사하여 사용자 컴퓨터에 **05_control_plane.py**로 저장합니다. 이 템플릿은 클러스터에 필요한 컨트롤 플레인 시스템을 설명합니다.
2. 리소스 정의에 필요한 다음 변수를 내보냅니다.

```
$ export MASTER_IGNITION=`cat <installation_directory>/master.ign`
```

3. **05_control_plane.yaml** 리소스 정의 파일을 생성합니다.

```
$ cat <<EOF >05_control_plane.yaml
imports:
- path: 05_control_plane.py

resources:
- name: cluster-control-plane
  type: 05_control_plane.py
  properties:
    infra_id: '${INFRA_ID}' 1
    zones: 2
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'

    control_subnet: '${CONTROL_SUBNET}' 3
    image: '${CLUSTER_IMAGE}' 4
    machine_type: 'n1-standard-4' 5
    root_volume_size: '128'
    service_account_email: '${MASTER_SERVICE_ACCOUNT}' 6

    ignition: '${MASTER_IGNITION}' 7
EOF
```

1. **infra_id**는 추출 단계에서 가져온 **INFRA_ID** 인프라 이름입니다.
2. **zones**은 컨트롤 플레인 인스턴스를 배포할 영역입니다(예: **us-central1-a**, **us-central1-b**, **us-central1-c**).
3. **control_subnet**은 컨트롤 서브넷에 대한 **selfLink** URL입니다.
4. **image**는 RHCOS 이미지에 대한 **selfLink** URL입니다.
5. **machine_type**은 인스턴스의 시스템 유형입니다(예: **n1-standard-4**).
6. **service_account_email**은 사용자가 생성한 마스터 서비스 계정의 이메일 주소입니다.
7. **ignition**은 **master.ign** 파일의 내용입니다.

4. **gcloud** CLI를 사용하여 배포를 생성합니다.

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-control-plane --config
05_control_plane.yaml
```

5. Deployment Manager의 제한으로 인해 템플릿을 통해 로드 밸런서 멤버십이 관리되지 않으므로 수동으로 컨트롤 플레인 시스템을 추가해야 합니다.

- 다음 명령을 실행하여 적절한 인스턴스 그룹에 컨트롤 플레인 머신을 추가합니다.

```
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
```

```
${ZONE_0}-instance-group --zone=${ZONE_0} --instances=${INFRA_ID}-master-0
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_1}-instance-group --zone=${ZONE_1} --instances=${INFRA_ID}-master-1
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_2}-instance-group --zone=${ZONE_2} --instances=${INFRA_ID}-master-2
```

- 외부 클러스터의 경우 다음 명령을 실행하여 대상 풀에 컨트롤 플레인 머신을 추가해야 합니다.

```
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_0}" --instances=${INFRA_ID}-master-0
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_1}" --instances=${INFRA_ID}-master-1
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_2}" --instances=${INFRA_ID}-master-2
```

7.11.16.1. 컨트롤 플레인 시스템에 대한 Deployment Manager 템플릿

다음 Deployment Mananger 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 컨트롤 플레인 시스템을 배포할 수 있습니다.

예 7.17. 05_control_plane.py Deployment Manager 템플릿

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-master-0',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'diskType': 'zones/' + context.properties['zones'][0] + '/diskTypes/pd-ssd',
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zones'][0] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
```

```
                context.properties['infra_id'] + '-master',
            ]
        },
        'zone': context.properties['zones'][0]
    }
}, {
    'name': context.properties['infra_id'] + '-master-1',
    'type': 'compute.v1.instance',
    'properties': {
        'disks': [{
            'autoDelete': True,
            'boot': True,
            'initializeParams': {
                'diskSizeGb': context.properties['root_volume_size'],
                'diskType': 'zones/' + context.properties['zones'][1] + '/diskTypes/pd-ssd',
                'sourceImage': context.properties['image']
            }
        }],
        'machineType': 'zones/' + context.properties['zones'][1] + '/machineTypes/' +
context.properties['machine_type'],
        'metadata': {
            'items': [{
                'key': 'user-data',
                'value': context.properties['ignition']
            }]
        },
        'networkInterfaces': [{
            'subnetwork': context.properties['control_subnet']
        }],
        'serviceAccounts': [{
            'email': context.properties['service_account_email'],
            'scopes': ['https://www.googleapis.com/auth/cloud-platform']
        }],
        'tags': {
            'items': [
                context.properties['infra_id'] + '-master',
            ]
        },
        'zone': context.properties['zones'][1]
    }
}, {
    'name': context.properties['infra_id'] + '-master-2',
    'type': 'compute.v1.instance',
    'properties': {
        'disks': [{
            'autoDelete': True,
            'boot': True,
            'initializeParams': {
                'diskSizeGb': context.properties['root_volume_size'],
                'diskType': 'zones/' + context.properties['zones'][2] + '/diskTypes/pd-ssd',
                'sourceImage': context.properties['image']
            }
        }],
        'machineType': 'zones/' + context.properties['zones'][2] + '/machineTypes/' +
context.properties['machine_type'],
        'metadata': {
```

```
            'items': [{
                'key': 'user-data',
                'value': context.properties['ignition']
            }]
        },
        'networkInterfaces': [{
            'subnetwork': context.properties['control_subnet']
        }],
        'serviceAccounts': [{
            'email': context.properties['service_account_email'],
            'scopes': ['https://www.googleapis.com/auth/cloud-platform']
        }],
        'tags': {
            'items': [
                context.properties['infra_id'] + '-master',
            ]
        },
        'zone': context.properties['zones'][2]
    }
}]

return {'resources': resources}
```

7.11.17. 부트스트랩이 완료되길 기다렸다가 GCP에서 부트스트랩 리소스를 제거합니다.

GCP(Google Cloud Platform)에 필요한 인프라를 모두 생성한 후 설치 프로그램에 의해 생성된 Ignition 구성 파일을 사용하여 프로비저닝한 시스템에서 부트스트랩 프로세스가 완료될 때까지 기다립니다.

사전 요구 사항

- GCP 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.
- GCP에서 VPC 및 관련 서브넷을 생성하고 구성하십시오.
- GCP에서 네트워킹 및 로드 밸런서를 생성하고 구성하십시오.
- 컨트롤 플레인 및 컴퓨팅 역할을 생성합니다.
- 부트스트랩 시스템을 생성합니다.
- 컨트롤 플레인 시스템을 생성합니다.

프로세스

1. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

```
$ ./openshift-install wait-for bootstrap-complete --dir <installation_directory> \ 1
    --log-level info 2
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2 다른 설치 세부 사항을 보려면 **info** 대신 **warn**, **debug** 또는 **error**를 지정합니다.

FATAL 경고 없이 명령이 종료되면 프로덕션 컨트롤 플레인이 초기화된 것입니다.

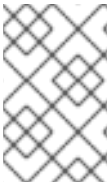
2. 부트스트랩 리소스를 삭제합니다.

```
$ gcloud compute backend-services remove-backend ${INFRA_ID}-api-internal-backend-
service --region=${REGION} --instance-group=${INFRA_ID}-bootstrap-instance-group --
instance-group-zone=${ZONE_0}
$ gsutil rm gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign
$ gsutil rb gs://${INFRA_ID}-bootstrap-ignition
$ gcloud deployment-manager deployments delete ${INFRA_ID}-bootstrap
```

7.11.18. GCP에 추가 작업자 시스템 생성

개별 인스턴스를 따로 시작하거나 자동 확장 그룹과 같은 클러스터 외부의 자동화된 프로세스를 통해 클러스터에서 사용할 작업자 시스템을 GCP(Google Cloud Platform)에 생성할 수 있습니다. OpenShift Container Platform의 기본 제공 클러스터 확장 메커니즘과 시스템 API를 활용할 수도 있습니다.

예에서는 Deployment Manager 템플릿을 사용하여 인스턴스 하나를 수동으로 시작합니다. 파일에 **06_worker.py** 유형의 추가 리소스를 포함시켜 추가 인스턴스를 시작할 수 있습니다.



참고

작업자 시스템을 생성하는 데 제공된 Deployment Manager 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- GCP 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.
- GCP에서 VPC 및 관련 서브넷을 생성하고 구성하십시오.
- GCP에서 네트워킹 및 로드 밸런서를 생성하고 구성하십시오.
- 컨트롤 플레인 및 컴퓨팅 역할을 생성합니다.
- 부트스트랩 시스템을 생성합니다.
- 컨트롤 플레인 시스템을 생성합니다.

프로세스

1. 이 항목의 **작업자 시스템에 대한 Deployment Manager 템플릿** 섹션에서 템플릿을 복사하여 사용자 컴퓨터에 **06_worker.py**로 저장합니다. 이 템플릿에서 클러스터에 필요한 작업자 시스템을 설명합니다.
2. 리소스 정의가 사용하는 변수를 내보냅니다.
   a. 컴퓨팅 시스템을 호스팅하는 서브넷을 내보냅니다.

```
$ export COMPUTE_SUBNET=(`gcloud compute networks subnets describe
${HOST_PROJECT_COMPUTE_SUBNET} --region=${REGION} --project
${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT} --format json | jq -r
.selfLink`)
```

b. 서비스 계정의 이메일 주소를 내보냅니다.

```
$ export WORKER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

c. 컴퓨팅 시스템 Ignition 구성 파일의 위치를 내보냅니다.

```
$ export WORKER_IGNITION=`cat <installation_directory>/worker.ign`
```

3. **06_worker.yaml** 리소스 정의 파일을 생성합니다.

```
$ cat <<EOF >06_worker.yaml
imports:
- path: 06_worker.py

resources:
- name: 'worker-0' (1)
  type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}' (2)
    zone: '${ZONE_0}' (3)
    compute_subnet: '${COMPUTE_SUBNET}' (4)
    image: '${CLUSTER_IMAGE}' (5)
    machine_type: 'n1-standard-4' (6)
    root_volume_size: '128'
    service_account_email: '${WORKER_SERVICE_ACCOUNT}' (7)
    ignition: '${WORKER_IGNITION}' (8)
- name: 'worker-1'
  type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}' (9)
    zone: '${ZONE_1}' (10)
    compute_subnet: '${COMPUTE_SUBNET}' (11)
    image: '${CLUSTER_IMAGE}' (12)
    machine_type: 'n1-standard-4' (13)
    root_volume_size: '128'
    service_account_email: '${WORKER_SERVICE_ACCOUNT}' (14)
    ignition: '${WORKER_IGNITION}' (15)
EOF
```

(1) **name**은 작업자 머신의 이름입니다(예: **worker-0**).

(2) (9) **infra_id**는 추출 단계에서 가져온 **INFRA_ID** 인프라 이름입니다.

(3) (10) **zone**은 작업자 시스템을 배포할 영역입니다(예: **us-central1-a**).

(4) (11) **compute_subnet**은 컴퓨팅 서브넷에 대한 **selfLink** URL입니다.

5 12 **image**는 RHCOS 이미지에 대한 **selfLink** URL입니다. [1]

6 13 **machine_type**은 인스턴스의 시스템 유형입니다(예: **n1-standard-4)**.

7 14 **service_account_email**은 사용자가 생성한 작업자 서비스 계정의 이메일 주소입니다.

8 15 **ignition**은 **worker.ign** 파일의 내용입니다.

4. 선택사항: 추가 인스턴스를 시작하려면 **06_worker.yaml** 리소스 정의 파일에 **06_worker.py** 형식의 추가 리소스를 포함시키십시오.

5. **gcloud** CLI를 사용하여 배포를 생성합니다.

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-worker --config
06_worker.yaml
```

1. GCP Marketplace 이미지를 사용하려면 사용할 오퍼를 지정합니다.

   - OpenShift Container Platform: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-ocp-48-x86-64-202210040145**
   - OpenShift Platform Plus: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-opp-48-x86-64-202206140145**
   - OpenShift Kubernetes Engine: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-oke-48-x86-64-202206140145**

7.11.18.1. 작업자 시스템에 대한 Deployment Manager 템플릿

다음 Deloyment Manager 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 작업자 시스템을 배포할 수 있습니다.

예 7.18. 06_worker.py Deployment Manager 템플릿

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-' + context.env['name'],
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
```

```
                'value': context.properties['ignition']
            }]
        },
        'networkInterfaces': [{
            'subnetwork': context.properties['compute_subnet']
        }],
        'serviceAccounts': [{
            'email': context.properties['service_account_email'],
            'scopes': ['https://www.googleapis.com/auth/cloud-platform']
        }],
        'tags': {
            'items': [
                context.properties['infra_id'] + '-worker',
            ]
        },
        'zone': context.properties['zone']
    }
}]

return {'resources': resources}
```

7.11.19. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift CLI(**oc**)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 **oc**를 설치할 수 있습니다.

중요

이전 버전의 **oc**를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 **oc**를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 Linux에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Linux Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. **oc** 바이너리를 **PATH**에 있는 디렉터리에 배치합니다.
   PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치
다음 절차에 따라 Windows에 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Windows Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. ZIP 프로그램으로 아카이브의 압축을 풉니다.
5. **oc** 바이너리를 **PATH**에 있는 디렉터리로 이동합니다.
   PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

   ```
   C:\> path
   ```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치
다음 절차에 따라 macOS에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 MacOSX Client** 항목 옆에 있는 **지금 다운로드를** 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 해제하고 압축을 풉니다.
5. **oc** 바이너리 PATH의 디렉터리로 이동합니다.
   PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

   ```
   $ echo $PATH
   ```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

7.11.20. CLI를 사용하여 클러스터에 로그인

클러스터 **kubeconfig** 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. **kubeconfig** 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- **oc** CLI를 설치했습니다.

프로세스

1. **kubeadmin** 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 **oc** 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

7.11.21. 머신의 인증서 서명 요청 승인

클러스터에 시스템을 추가하면 추가한 시스템별로 보류 중인 인증서 서명 요청(CSR)이 두 개씩 생성됩니다. 이러한 CSR이 승인되었는지 확인해야 하며, 필요한 경우 이를 직접 승인해야 합니다. 클라이언트 요청을 먼저 승인한 다음 서버 요청을 승인해야 합니다.

사전 요구 사항

- 클러스터에 시스템을 추가했습니다.

프로세스

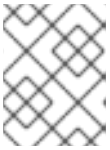
1. 클러스터가 시스템을 인식하는지 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0 Ready    master  63m  v1.22.1
master-1 Ready    master  63m  v1.22.1
master-2 Ready    master  64m  v1.22.1
```

출력에 생성된 모든 시스템이 나열됩니다.



참고

이전 출력에는 일부 CSR이 승인될 때까지 컴퓨팅 노드(작업자 노드라고도 함)가 포함되지 않을 수 있습니다.

2. 보류 중인 CSR을 검토하고 클러스터에 추가한 각 시스템에 대해 **Pending** 또는 **Approved** 상태의 클라이언트 및 서버 요청이 표시되는지 확인합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE    REQUESTOR                                                   CONDITION
csr-8b2br   15m    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

예에서는 두 시스템이 클러스터에 참여하고 있습니다. 목록에는 승인된 CSR이 더 많이 나타날 수도 있습니다.

3. CSR이 승인되지 않은 경우, 추가된 시스템에 대한 모든 보류 중인 CSR이 **Pending** 상태로 전환된 후 클러스터 시스템의 CSR을 승인합니다.

참고

CSR은 교체 주기가 자동으로 만료되므로 클러스터에 시스템을 추가한 후 1시간 이내에 CSR을 승인하십시오. 한 시간 내에 승인하지 않으면 인증서가 교체되고 각 노드에 대해 두 개 이상의 인증서가 표시됩니다. 이러한 인증서를 모두 승인해야 합니다. 클라이언트 CSR이 승인되면 Kubelet은 인증서에 대한 보조 CSR을 생성하므로 수동 승인이 필요합니다. 그러면 Kubelet에서 동일한 매개변수를 사용하여 새 인증서를 요청하는 경우 인증서 갱신 요청은 **machine-approver**에 의해 자동으로 승인됩니다.

참고

베어 메탈 및 기타 사용자 프로비저닝 인프라와 같이 머신 API를 사용하도록 활성화되지 않는 플랫폼에서 실행되는 클러스터의 경우 CSR(Kubelet service Certificate Request)을 자동으로 승인하는 방법을 구현해야 합니다. 요청이 승인되지 않으면 API 서버가 kubelet에 연결될 때 서비스 인증서가 필요하므로 **oc exec**, **oc rsh**, **oc logs** 명령을 성공적으로 수행할 수 없습니다. Kubelet 엔드 포인트에 연결하는 모든 작업을 수행하려면 이 인증서 승인이 필요합니다. 이 방법은 새 CSR을 감시하고 CSR이 **system:node** 또는 **system:admin** 그룹의 **node-bootstrapper** 서비스 계정에 의해 제출되었는지 확인하고 노드의 ID를 확인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>**은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

참고

일부 Operator는 일부 CSR이 승인될 때까지 사용할 수 없습니다.

4. 이제 클라이언트 요청이 승인되었으므로 클러스터에 추가한 각 머신의 서버 요청을 검토해야 합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 나머지 CSR이 승인되지 않고 **Pending** 상태인 경우 클러스터 머신의 CSR을 승인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>**은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. 모든 클라이언트 및 서버 CSR이 승인된 후 머신은 **Ready** 상태가 됩니다. 다음 명령을 실행하여 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.22.1
master-1  Ready     master  73m  v1.22.1
master-2  Ready     master  74m  v1.22.1
worker-0  Ready     worker  11m  v1.22.1
worker-1  Ready     worker  11m  v1.22.1
```

참고

머신이 **Ready** 상태로 전환하는 데 서버 CSR의 승인 후 몇 분이 걸릴 수 있습니다.

추가 정보

- CSR에 대한 자세한 내용은 인증서 서명 요청을 참조하십시오.

7.11.22. 인그레스 DNS 레코드 추가

Kubernetes 매니페스트를 생성하고 Ignition 구성을 생성할 때 DNS 영역 구성이 제거됩니다. 인그레스 로드 밸런서를 가리키는 DNS 레코드를 수동으로 생성해야 합니다. 와일드카드 ***.apps.{baseDomain}.** 또는 특정 레코드를 생성할 수 있습니다. 사용자 요구사항에 따라 A, CNAME 및 기타 레코드를 사용할 수 있습니다.

사전 요구 사항

- GCP 계정을 구성하십시오.
- Kubernetes 매니페스트를 생성하고 Ignition 구성을 생성할 때 DNS 영역 구성을 제거하십시오.
- GCP에서 VPC 및 관련 서브넷을 생성하고 구성하십시오.
- GCP에서 네트워킹 및 로드 밸런서를 생성하고 구성하십시오.
- 컨트롤 플레인 및 컴퓨팅 역할을 생성합니다.
- 부트스트랩 시스템을 생성합니다.
- 컨트롤 플레인 시스템을 생성합니다.
- 작업자 시스템을 생성합니다.

프로세스

1. 인그레스 라우터가 로드 밸런서를 생성하고 **EXTERNAL-IP** 필드를 채울 때까지 기다립니다.

```
$ oc -n openshift-ingress get service router-default
```

출력 예

```
NAME            TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)                      AGE
router-default  LoadBalancer   172.30.18.154  35.233.157.184
80:32288/TCP,443:31215/TCP   98
```

2. 영역에 A 레코드를 추가합니다.
   - A 레코드를 사용하려면,
     i. 라우터 IP 주소에 대한 변수를 내보냅니다.

```
$ export ROUTER_IP=`oc -n openshift-ingress get service router-default --no-
headers | awk '{print $4}'`
```

     ii. 프라이빗 영역에 A 레코드를 추가합니다.

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone --project
${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
$ gcloud dns record-sets transaction add ${ROUTER_IP} --name
\*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone
```

```
${INFRA_ID}-private-zone --project ${HOST_PROJECT} --account
${HOST_PROJECT_ACCOUNT}
$ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone --
project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
```

iii. 외부 클러스터의 경우 퍼블릭 영역에 A 레코드를 추가합니다.

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME} -
-project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
$ gcloud dns record-sets transaction add ${ROUTER_IP} --name
\*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone
${BASE_DOMAIN_ZONE_NAME} --project ${HOST_PROJECT} --account
${HOST_PROJECT_ACCOUNT}
$ gcloud dns record-sets transaction execute --zone
${BASE_DOMAIN_ZONE_NAME} --project ${HOST_PROJECT} --account
${HOST_PROJECT_ACCOUNT}
```

- 와일드카드를 사용하지 않고 명시적 도메인을 추가하려면 클러스터의 현재 경로에 대한 항목을 생성합니다.

```
$ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}
{"\n"}{end}{end}' routes
```

출력 예

```
oauth-openshift.apps.your.cluster.domain.example.com
console-openshift-console.apps.your.cluster.domain.example.com
downloads-openshift-console.apps.your.cluster.domain.example.com
alertmanager-main-openshift-monitoring.apps.your.cluster.domain.example.com
grafana-openshift-monitoring.apps.your.cluster.domain.example.com
prometheus-k8s-openshift-monitoring.apps.your.cluster.domain.example.com
```

7.11.23. 인그레스 방화벽 규칙 추가

클러스터에는 몇 가지 방화벽 규칙이 필요합니다. 공유 VPC를 사용하지 않는 경우, 이러한 규칙은 GCP 클라우드 공급자를 통해 Ingress Controller에 의해 생성됩니다. 공유 VPC를 사용할 경우에는 모든 서비스에 대한 클러스터 단위 방화벽 규칙을 생성하거나 클러스터가 액세스를 요청할 때 이벤트를 기반으로 각 규칙을 생성할 수 있습니다. 클러스터가 액세스를 요청할 때마다 규칙을 생성하면 필요한 방화벽 규칙을 정확하게 파악할 수 있습니다. 클러스터 단위 방화벽 규칙을 생성하면 여러 클러스터에 동일한 규칙 세트를 적용할 수 있습니다.

이벤트 기준으로 각 규칙을 생성하도록 선택하는 경우, 클러스터를 프로비저닝한 후에 그리고 클러스터 수명 동안 콘솔에서 규칙이 누락되었음을 알릴 때 방화벽 규칙을 생성해야 합니다. 다음 이벤트와 유사한 이벤트가 표시되면 필요한 방화벽 규칙을 추가해야 합니다.

```
$ oc get events -n openshift-ingress --field-selector="reason=LoadBalancerManualChange"
```

출력 예

```
Firewall change required by security admin: `gcloud compute firewall-rules create k8s-fw-
a26e631036a3f46cba28f8df67266d55 --network example-network --description "
{\"kubernetes.io/service-name\":\"openshift-ingress/router-default\", \"kubernetes.io/service-
```

```
ip\":\"35.237.236.234\"}\" --allow tcp:443,tcp:80 --source-ranges 0.0.0.0/0 --target-tags exampl-fqzq7-master,exampl-fqzq7-worker --project example-project`
```

이러한 규칙 기반 이벤트가 생성될 때 문제가 발생하면 클러스터가 실행되는 동안 클러스터 단위 방화벽 규칙을 구성할 수 있습니다.

7.11.23.1. GCP에서 공유 VPC에 대한 클러스터 단위 방화벽 규칙 생성

OpenShift Container Platform 클러스터에 필요한 액세스를 허용하기 위해 클러스터 단위 방화벽 규칙을 생성할 수 있습니다.

주의

클러스터 이벤트 기준으로 방화벽 규칙을 생성하지 않도록 선택하는 경우, 사용자가 클러스터 단위 방화벽 규칙을 생성해야 합니다.

사전 요구 사항

- 클러스터를 배포하기 위해 Deployment Manager 템플릿에 필요한 변수를 내보냈습니다.
- 클러스터에 필요한 네트워킹 및 로드 밸런싱 구성 요소를 GCP에 생성했습니다.

프로세스

1. 모든 서비스에 액세스할 수 있도록 Google Cloud Engine 상태 검사를 허용하는 단일 방화벽 규칙을 추가합니다. 이 규칙을 사용하면 인그레스 로드 밸런서가 인스턴스의 상태를 확인할 수 있습니다.

   ```
   $ gcloud compute firewall-rules create --allow='tcp:30000-32767,udp:30000-32767' --network="${CLUSTER_NETWORK}" --source-ranges='130.211.0.0/22,35.191.0.0/16,209.85.152.0/22,209.85.204.0/22' --target-tags="${INFRA_ID}-master,${INFRA_ID}-worker" ${INFRA_ID}-ingress-hc --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
   ```

2. 모든 클러스터 서비스에 액세스를 허용하는 단일 방화벽 규칙을 추가합니다.

   - 외부 클러스터의 경우:

     ```
     $ gcloud compute firewall-rules create --allow='tcp:80,tcp:443' --network="${CLUSTER_NETWORK}" --source-ranges="0.0.0.0/0" --target-tags="${INFRA_ID}-master,${INFRA_ID}-worker" ${INFRA_ID}-ingress --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
     ```

   - 프라이빗 클러스터의 경우:

     ```
     $ gcloud compute firewall-rules create --allow='tcp:80,tcp:443' --network="${CLUSTER_NETWORK}" --source-ranges=${NETWORK_CIDR} --target-tags="${INFRA_ID}-master,${INFRA_ID}-worker" ${INFRA_ID}-ingress --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
     ```

이 규칙은 TCP 포트 **80** 및 **443**의 트래픽만을 허용하므로 서비스에서 사용하는 포트를 빠짐없이 모두 추가해야 합니다.

7.11.24. 사용자 프로비저닝 인프라에서 GCP 설치 완료

GCP(Google Cloud Platform) 사용자 프로비저닝 인프라에서 OpenShift Container Platform 설치를 시작한 후 클러스터가 준비를 마칠 때까지 클러스터 이벤트를 모니터링할 수 있습니다.

사전 요구 사항

- 사용자 프로비저닝 GCP 인프라에 OpenShift Container Platform 클러스터용 부트스트랩 시스템을 배포하십시오.
- **oc** CLI를 설치하고 로그인하십시오.

프로세스

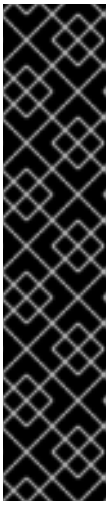
1. 클러스터 설치를 완료합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

출력 예

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.



중요

- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

2. 클러스터의 실행 상태를 관찰합니다.
   a. 다음 명령을 실행하여 현재 클러스터 버전과 상태를 확인합니다.

```
$ oc get clusterversion
```

출력 예

```
NAME     VERSION   AVAILABLE   PROGRESSING   SINCE   STATUS
version            False       True          24m     Working towards 4.5.4: 99% complete
```

b. 다음 명령을 실행하여 컨트롤 플레인에서 관리하되는 운영자를 CVO(Cluster Version Operator)별로 확인합니다.

```
$ oc get clusteroperators
```

출력 예

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.5.4     True        False         False      7m56s
cloud-credential                           4.5.4     True        False         False      31m
cluster-autoscaler                         4.5.4     True        False         False      16m
console                                    4.5.4     True        False         False      10m
csi-snapshot-controller                    4.5.4     True        False         False      16m
dns                                        4.5.4     True        False         False      22m
etcd                                       4.5.4     False       False         False      25s
image-registry                             4.5.4     True        False         False      16m
ingress                                    4.5.4     True        False         False      16m
insights                                   4.5.4     True        False         False      17m
kube-apiserver                             4.5.4     True        False         False      19m
kube-controller-manager                    4.5.4     True        False         False      20m
kube-scheduler                             4.5.4     True        False         False      20m
kube-storage-version-migrator              4.5.4     True        False         False      16m
machine-api                                4.5.4     True        False         False      22m
machine-config                             4.5.4     True        False         False      22m
marketplace                                4.5.4     True        False         False      16m
monitoring                                 4.5.4     True        False         False      10m
network                                    4.5.4     True        False         False      23m
node-tuning                                4.5.4     True        False         False      23m
openshift-apiserver                        4.5.4     True        False         False      17m
openshift-controller-manager               4.5.4     True        False         False      15m
openshift-samples                          4.5.4     True        False         False      16m
operator-lifecycle-manager                 4.5.4     True        False         False      22m
operator-lifecycle-manager-catalog         4.5.4     True        False         False      22m
operator-lifecycle-manager-packageserver   4.5.4     True        False         False      18m
service-ca                                 4.5.4     True        False         False      23m
service-catalog-apiserver                  4.5.4     True        False         False      23m
service-catalog-controller-manager         4.5.4     True        False         False      23m
storage                                    4.5.4     True        False         False      17m
```

c. 다음 명령어를 실행하여 클러스터 Pod를 확인합니다.

```
$ oc get pods --all-namespaces
```

출력 예

```
NAMESPACE                                       NAME
READY   STATUS    RESTARTS   AGE
kube-system                                     etcd-member-ip-10-0-3-111.us-east-
2.compute.internal             1/1     Running   0          35m
kube-system                                     etcd-member-ip-10-0-3-239.us-east-
2.compute.internal             1/1     Running   0          37m
kube-system                                     etcd-member-ip-10-0-3-24.us-east-
2.compute.internal             1/1     Running   0          35m
```

```
openshift-apiserver-operator                    openshift-apiserver-operator-6d6674f4f4-
h7t2t                    1/1     Running    1          37m
openshift-apiserver                             apiserver-fm48r
1/1     Running    0          30m
openshift-apiserver                             apiserver-fxkvv
1/1     Running    0          29m
openshift-apiserver                             apiserver-q85nm
1/1     Running    0          29m
...
openshift-service-ca-operator                   openshift-service-ca-operator-66ff6dc6cd-
9r257                    1/1     Running    0          37m
openshift-service-ca                            apiservice-cabundle-injector-695b6bcbc-cl5hm
1/1     Running    0          35m
openshift-service-ca                            configmap-cabundle-injector-8498544d7-
25qn6                    1/1     Running    0          35m
openshift-service-ca                            service-serving-cert-signer-6445fc9c6-wqdqn
1/1     Running    0          35m
openshift-service-catalog-apiserver-operator    openshift-service-catalog-apiserver-
operator-549f44668b-b5q2w    1/1     Running    0          32m
openshift-service-catalog-controller-manager-operator   openshift-service-catalog-
controller-manager-operator-b78cr2lnm    1/1     Running    0          31m
```

현재 클러스터 버전이 **AVAILABLE**이면 설치가 완료된 것입니다.

7.11.25. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보 를 참조하십시오.

7.11.26. 다음 단계

- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃 을 수행할 수 있습니다.

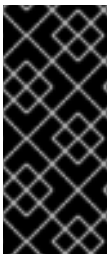
7.12. 사용자 프로비저닝 인프라가 있는 제한된 네트워크에서 GCP에 클러스터 설치

OpenShift Container Platform 4.9 버전에서는 사용자가 제공하는 인프라 및 설치 릴리스 콘텐츠의 내부 미러를 사용하는 GCP(Google Cloud Platform)에 클러스터를 설치할 수 있습니다.

중요

미러링된 설치 릴리스 콘텐츠를 사용하여 OpenShift Container Platform 클러스터를 설치할 수는 있지만 GCP API를 사용하려면 여전히 클러스터에 인터넷 액세스가 필요합니다.

사용자 제공 인프라 설치 단계가 여기에 요약되어 있습니다. 안내된 단계를 수행하거나 자체 모델링에 유용한 몇 가지 Deployment Manager 템플릿이 제공됩니다. 여러 다른 방법을 통해 필요한 리소스를 자유롭게 생성할 수도 있습니다.

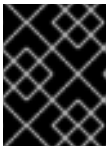


중요

사용자가 프로비저닝한 인프라 설치를 수행하는 단계는 예시용으로만 제공됩니다. 사용자가 제공하는 인프라를 사용하여 클러스터를 설치하려면 클라우드 공급자 및 OpenShift Container Platform 설치 프로세스에 대한 정보가 필요합니다. 안내된 단계를 수행하거나 자체 모델링에 유용한 몇 가지 Deployment Manager 템플릿이 제공됩니다. 여러 다른 방법을 통해 필요한 리소스를 자유롭게 생성할 수도 있습니다. 템플릿은 예시일 뿐입니다.

7.12.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비 에 대한 문서를 읽습니다.
- 미러 호스트에 레지스트리를 생성 하고 사용 중인 OpenShift Container Platform 버전의 **imageContentSources** 데이터를 가져옵니다.

  중요

  미러 호스트에 설치 미디어가 있으므로 해당 컴퓨터를 사용하여 모든 설치 단계를 완료하십시오.

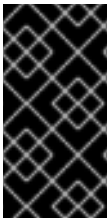
- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성 해야 합니다. 액세스 권한을 부여해야 할 사이트가 더 있을지도 모르지만 ***.googleapis.com**과 **accounts.google.com**에 액세스 권한은 반드시 부여해야 합니다.
- 사용자의 환경에서 클라우드 ID 및 액세스 관리(IAM) API에 액세스할 수 없거나 **kube-system** 네임스페이스에 관리자 수준의 인증 정보 시크릿을 저장하지 않으려면 IAM 인증 정보 수동 생성 및 유지 관리를 참조하십시오.

7.12.2. 네트워크가 제한된 환경에서의 설치 정보

OpenShift Container Platform 4.9에서는 소프트웨어 구성 요소를 받기 위한 인터넷 접속이 필요하지 않은 설치를 수행할 수 있습니다. 제한된 네트워크 설치는 클러스터를 설치하는 클라우드 플랫폼에 따라 설치 관리자 프로비저닝 인프라 또는 사용자 프로비저닝 인프라를 사용하여 완료할 수 있습니다.

클라우드 플랫폼에 제한된 네트워크 설치를 수행하는 방법을 선택해도 클라우드 API에 액세스는 가능해야 합니다. Amazon Web Service의 Route 53 DNS 및 IAM 서비스와 같은 일부 클라우드 기능에는 인터넷 액세스가 필요합니다. 사용 중인 네트워크에 따라 베어메탈 하드웨어 또는 VMware vSphere에 설치하기 위해 필요한 인터넷 액세스가 줄어들 수 있습니다.

제한된 네트워크 설치를 완료하려면 OpenShift Container Platform 레지스트리의 내용을 미러링하고 설치 미디어를 포함할 레지스트리를 생성해야 합니다. 인터넷과 폐쇄 네트워크에 모두 액세스하거나 제한 사항을 따르는 다른 방법을 통해 미러 호스트에 레지스트리를 생성할 수 있습니다.



중요

사용자 프로비저닝 설치의 구성이 복잡하므로 사용자 프로비저닝 인프라를 사용하여 제한된 네트워크 설치를 시도하기 전에 표준 사용자 프로비저닝 인프라 설치를 완료하는 것이 좋습니다. 이 테스트 설치를 완료하면 제한된 네트워크에 설치하는 동안 발생할 수 있는 문제를 보다 쉽게 파악 및 해결할 수 있습니다.

7.12.2.1. 추가 제한

제한된 네트워크의 클러스터에는 다음과 같은 추가 제한이 있습니다.

- **ClusterVersion** 상태에 **사용 가능한 업데이트를 검색할 수 없음** 오류가 포함되어 있습니다.
- 기본적으로 필요한 이미지 스트림 태그에 액세스할 수 없기 때문에 개발자 카탈로그의 내용을 사용할 수 없습니다.

7.12.3. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하기 위해 필요한 이미지를 받으려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.

중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

7.12.4. GCP 프로젝트 구성

OpenShift Container Platform을 설치하려면 먼저 호스팅할 GCP(Google Cloud Platform) 프로젝트를 구성해야 합니다.

7.12.4.1. GCP 프로젝트 생성

OpenShift Container Platform을 설치하려면 클러스터를 호스팅할 GCP(Google Cloud Platform) 계정에 프로젝트를 생성해야 합니다.

프로세스

- OpenShift Container Platform 클러스터를 호스팅할 프로젝트를 생성합니다. GCP 문서의 프로젝트 생성 및 관리 단원을 참조하십시오.

중요

설치 관리자 프로비저닝 인프라를 사용하는 경우 GCP 프로젝트는 Premium Network Service Tier를 사용해야 합니다. 설치 프로그램을 사용하여 설치된 클러스터의 Standard Network Service Tier는 지원되지 않습니다. 설치 프로그램은 **api-int.<cluster_name>.<base_domain>** URL에 대한 내부 로드 밸런싱을 구성합니다.

7.12.4.2. GCP에서 API 서비스 활성화

GCP(Google Cloud Platform) 프로젝트에서 OpenShift Container Platform 설치를 완료하려면 여러 API 서비스에 액세스해야 합니다.

사전 요구 사항

- 클러스터를 호스팅할 프로젝트 생성을 완료했습니다.

프로세스

- 클러스터를 호스팅하는 프로젝트에서 다음과 같은 필수 API 서비스를 활성화합니다. GCP 문서의 서비스 활성화 단원을 참조하십시오.

표 7.51. 필수 API 서비스

API 서비스	콘솔 서비스 이름
컴퓨팅 엔진 API	**compute.googleapis.com**
Google 클라우드 API	**cloudapis.googleapis.com**
클라우드 리소스 관리자 API	**cloudresourcemanager.googleapis.com**
Google DNS API	**dns.googleapis.com**
IAM 서비스 계정 자격 증명 API	**iamcredentials.googleapis.com**
IAM(ID 및 액세스 관리) API	**iam.googleapis.com**
서비스 관리 API	**servicemanagement.googleapis.com**
서비스 사용량 API	**serviceusage.googleapis.com**
Google 클라우드 스토리지 JSON API	**storage-api.googleapis.com**
클라우드 스토리지	**storage-component.googleapis.com**

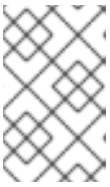
7.12.4.3. GCP용 DNS 구성

OpenShift Container Platform을 설치하려면 사용하는 GCP(Google Cloud Platform) 계정에 OpenShift Container Platform 클러스터를 호스팅하는 프로젝트와 동일한 프로젝트에 전용 퍼블릭 호스팅 영역이 있어야 합니다. 도메인에 대한 권한도 이 영역에 있어야 합니다. DNS 서비스는 클러스터와 외부 연결에 필

요한 클러스터 DNS 확인 및 이름 조회 기능을 제공합니다.

프로세스

1. 도메인 또는 하위 도메인과 등록 기관을 식별합니다. 기존 도메인 및 등록 기관을 이전하거나 GCP 또는 다른 소스를 통해 새 도메인과 등록 기관을 구할 수 있습니다.



참고

새 도메인을 구입하는 경우, 관련 DNS 변경사항이 전파되는 데 시간이 걸릴 수 있습니다. Google을 통한 도메인 구매에 대한 자세한 내용은 Google 도메인을 참조하십시오.

2. GCP 프로젝트에서 도메인 또는 하위 도메인의 퍼블릭 호스팅 영역을 생성합니다. GCP 문서의 퍼블릭 영역 생성 단원을 참조하십시오.
적절한 루트 도메인(예: **openshiftcorp.com**) 또는 하위 도메인(예: **clusters.openshiftcorp.com**)을 사용합니다.

3. 호스팅 영역 레코드에서 권한이 있는 새 이름 서버를 추출합니다. GCP 문서의 클라우드 DNS 이름 서버 조회 단원을 참조하십시오.
일반적으로 네 가지 이름 서버가 있습니다.

4. 도메인에서 사용하는 이름 서버의 등록 기관 레코드를 업데이트합니다. 예를 들어, 사용자 도메인을 Google 도메인에 등록한 경우 Google 도메인 도움말의 사용자 지정 이름 서버로 전환하는 방법 항목을 참조하십시오.

5. 루트 도메인을 Google Cloud DNS로 마이그레이션했으면 DNS 레코드를 마이그레이션합니다. GCP 문서의 클라우드 DNS로 마이그레이션을 참조하십시오.

6. 하위 도메인을 사용하는 경우, 회사의 프로시저에 따라 상위 도메인에 위임 레코드를 추가합니다. 이 과정에 회사의 IT 부서 또는 회사의 루트 도메인 및 DNS 서비스를 제어하는 부서에 요청하는 일도 포함될 수 있습니다.

7.12.4.4. GCP 계정 제한

OpenShift Container Platform 클러스터는 여러 GCP(Google Cloud Platform) 구성 요소를 사용하지만 기본 할당량이 기본 OpenShift Container Platform 클러스터 설치가 가능할지 여부에 영향을 미치지 않습니다.

컴퓨팅 및 컨트롤 플레인 시스템 세 개가 포함된 기본 클러스터는 다음과 같은 리소스를 사용합니다. 일부 리소스는 부트스트랩 프로세스 중에만 필요하며 클러스터 배포 후 제거됩니다.

표 7.52. 기본 클러스터에서 사용되는 GCP 리소스

서비스	구성 요소	위치	필요한 총 리소스	부트스트랩 후 제거된 리소스
서비스 계정	IAM	글로벌	5	0
방화벽 규칙	네트워킹	글로벌	11	1
전송 규칙	컴퓨팅	글로벌	2	0

서비스	구성 요소	위치	필요한 총 리소스	부트스트랩 후 제거된 리소스
상태 검사	컴퓨팅	글로벌	2	0
이미지	컴퓨팅	글로벌	1	0
네트워크	네트워킹	글로벌	1	0
라우터	네트워킹	글로벌	1	0
라우트	네트워킹	글로벌	2	0
서브네트워크	컴퓨팅	글로벌	2	0
대상 풀	네트워킹	글로벌	2	0

참고

설치하는 동안 할당량이 충분하지 않으면 설치 프로그램에서 초과된 할당량과 리전을 모두 안내하는 오류 메시지를 표시합니다.

실제 클러스터 크기, 예상 클러스터 증가, 계정과 연결된 다른 클러스터의 사용량을 모두 고려해야 합니다. CPU, 고정 IP 주소, 영구 디스크 SSD(스토리지) 할당량이 가장 부족하기 쉬운 할당량입니다.

다음 리전 중 하나에서 클러스터를 배포하려는 경우, 최대 스토리지 할당량을 초과할 것이며, CPU 할당량 제한을 초과할 가능성도 있습니다.

- **asia-east2**
- **asia-northeast2**
- **asia-south1**
- **australia-southeast1**
- **europe-north1**
- **europe-west2**
- **europe-west3**
- **europe-west6**
- **northamerica-northeast1**
- **southamerica-east1**
- **us-west2**

GCP 콘솔에서 리소스 할당량을 늘릴 수는 있지만 지원 티켓을 제출해야 할 수도 있습니다. OpenShift Container Platform 클러스터를 설치하기 전에 지원 티켓을 해결할 시간이 충분하도록 조기에 클러스터 크기를 계획해야 합니다.

7.12.4.5. GCP에서 서비스 계정 생성

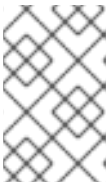
OpenShift Container Platform에는 Google API의 데이터에 액세스하기 위한 인증 및 승인을 제공하는 GCP(Google Cloud Platform) 서비스 계정이 필요합니다. 프로젝트에 필요한 역할이 포함된 기존 IAM 서비스 계정이 없으면 새로 생성해야 합니다.

사전 요구 사항

- 클러스터를 호스팅할 프로젝트 생성을 완료했습니다.

프로세스

1. OpenShift Container Platform 클러스터를 호스팅하는 데 사용하는 프로젝트에 서비스 계정을 생성합니다. GCP 문서의 서비스 계정 생성 단원을 참조하십시오.
2. 서비스 계정에 적절한 권한을 부여합니다. 뒤따르는 개별 권한을 부여하거나 **Owner** 역할을 할당할 수 있습니다. 서비스 계정에 특정 리소스에 대한 역할 부여 를 참조하십시오.



참고

서비스 계정을 프로젝트 소유자로 지정하는 것은 가장 쉽게 필요한 권한을 얻는 방법이며, 서비스 계정으로 프로젝트를 완전히 제어할 수 있음을 의미합니다. 해당 권한을 제공하는 데 따른 위험이 수용 가능한 수준인지 확인해봐야 합니다.

3. JSON 형식으로 서비스 계정 키를 생성합니다. GCP 문서의 서비스 계정 키 생성 단원을 참조하십시오.
   클러스터를 생성하기 위해서는 서비스 계정 키가 필요합니다.

7.12.4.5.1. 필요한 GCP 권한

생성하는 서비스 계정에 **Owner** 역할을 연결하면 OpenShift Container Platform 설치에 필요한 권한을 포함하여 모든 권한이 해당 서비스 계정에 부여됩니다. OpenShift Container Platform 클러스터를 배포하려면 서비스 계정에 다음과 같은 권한이 필요합니다. 기존 VPC에 클러스터를 배포할 때는 다음 목록에 제시된 특정 네트워킹 권한이 서비스 계정에 필요하지 않습니다.

설치 프로그램에 필요한 역할

- 컴퓨팅 관리자
- 보안 관리자
- 서비스 계정 관리자
- 서비스 계정 사용자
- 스토리지 관리자

설치 과정에서 네트워크 리소스를 생성하는 데 필요한 역할

- DNS 관리자

사용자 프로비저닝 GCP 인프라에 필요한 역할

- 배포 관리자 편집자
- 서비스 계정 키 관리자

선택적 역할

운영자를 위한 제한적 자격 증명을 새로 생성하는 클러스터의 경우 다음 역할을 추가합니다.

- 서비스 계정 키 관리자

컨트롤 플레인 및 컴퓨팅 시스템이 사용하는 서비스 계정에 적용되는 역할입니다.

표 7.53. GCP 서비스 계정 권한

계정	역할
컨트롤 플레인	**roles/compute.instanceAdmin**
	roles/compute.networkAdmin
	roles/compute.securityAdmin
	roles/storage.admin
	roles/iam.serviceAccountUser
컴퓨팅	**roles/compute.viewer**
	roles/storage.admin

7.12.4.6. 지원되는 GCP 리전

다음과 같은 GCP(Google Cloud Platform) 리전에 OpenShift Container Platform 클러스터를 배포할 수 있습니다.

- **asia-east1** (대만 장화현)
- **asia-east2** (홍콩)
- **asia-northeast1** (일본 도쿄)
- **asia-northeast2** (일본 오사카)
- **asia-northeast3** (한국 서울)
- **asia-south1** (인도 뭄바이)
- **asia-southeast1** (싱가포르 주롱 웨스트)
- **asia-southeast2** (인도네시아 자카르타)
- **australia-southeast1** (호주 시드니)

- **europe-central2** (폴란드 바르샤바)
- **europe-north1** (핀란드 하미나)
- **europe-west1** (벨기에 생기슬랭)
- **europe-west2** (영국 런던)
- **europe-west3** (독일 프랑크푸르트)
- **europe-west4** (네덜란드 엠스하벤)
- **europe-west6** (스위스 취리히)
- **northamerica-northeast1** (캐나다 퀘벡 주 몬트리올)
- **southamerica-east1** (브라질 상파울루)
- **us-central1** (미국 아이오와 주 카운실 블러프스)
- **us-east1** (미국 사우스 캐롤라이나 주 몽크스 코너)
- **us-east4** (미국 노던 버지니아 주 애쉬번)
- **us-west1** (미국 오레곤 주 댈러스)
- **us-west2** (미국 캘리포니아 주 로스앤젤레스)
- **us-west3** (미국 유타 주 솔트레이크시티)
- **us-west4** (미국 네바다 라스베거스)

7.12.4.7. GCP용 CLI 도구 설치 및 구성

사용자 프로비저닝 인프라를 사용하는 OpenShift Container Platform을 GCP(Google Cloud Platform)에 설치하려면 GCP용 CLI 도구를 설치하고 구성해야 합니다.

사전 요구 사항

- 클러스터를 호스팅할 프로젝트 생성을 완료했습니다.
- 서비스 계정 생성과 필요한 권한 부여 작업을 마쳤습니다.

프로세스

1. **$PATH에** 다음 바이너리를 설치합니다.
   - **gcloud**
   - **gsutil**

   GCP 문서의 최신 클라우드 SDK 버전 설치 단원을 참조하십시오.
2. 구성된 서비스 계정과 **gcloud** 도구를 사용하여 인증합니다.
   GCP 문서의 서비스 계정 인증을 참조하십시오.

7.12.5. 사용자 프로비저닝 인프라를 포함한 클러스터의 시스템 요구사항

사용자 프로비저닝 인프라가 포함된 클러스터의 경우, 필요한 모든 시스템을 배포해야 합니다.

이 섹션에서는 사용자 프로비저닝 인프라에 OpenShift Container Platform을 배포해야 하는 요구 사항에 대해 설명합니다.

7.12.5.1. 클러스터 설치에 필요한 시스템

최소 OpenShift Container Platform 클러스터에 다음과 같은 호스트가 필요합니다.

표 7.54. 최소 필수 호스트

호스트	설명
임시 부트스트랩 시스템 한 개	컨트롤 플레인 시스템 세 개에 OpenShift Container Platform 클러스터를 배포하기 위한 부트스트랩 시스템이 클러스터에 필요합니다. 클러스터를 설치한 후 부트스트랩 시스템을 제거할 수 있습니다.
컨트롤 플레인 시스템 세 개	컨트롤 플레인 시스템은 컨트롤 플레인을 구성하는 Kubernetes 및 OpenShift Container Platform 서비스를 실행합니다.
두 개 이상의 컴퓨팅 시스템(작업자 시스템이라고도 함).	OpenShift Container Platform 사용자가 요청한 워크로드는 컴퓨팅 머신에서 실행됩니다.

중요

클러스터의 고가용성을 유지하려면 이러한 클러스터 시스템에 대해 별도의 물리적 호스트를 사용하십시오.

부트스트랩, 컨트롤 플레인 시스템은 운영 체제로 RHCOS (Red Hat Enterprise Linux CoreOS)를 사용해야 합니다. 그러나 컴퓨팅 머신은 RHCOS(Red Hat Enterprise Linux CoreOS), RHEL (Red Hat Enterprise Linux) 7.9 또는 RHEL 8.4 중에서 선택할 수 있습니다.

RHCOS는 RHEL 8(Red Hat Enterprise Linux)을 기반으로하며 모든 하드웨어 인증 및 요구사항을 모두 이어받습니다. Red Hat Enterprise Linux 기술 기능 및 제한 을 참조하십시오.

7.12.5.2. 클러스터 설치를 위한 최소 리소스 요구 사항

각 클러스터 시스템이 다음과 같은 최소 요구사항을 충족해야 합니다.

표 7.55. 최소 리소스 요구사항

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
부트스트랩	RHCOS	4	16GB	100GB	300
컨트롤 플레인	RHCOS	4	16GB	100GB	300

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
컴퓨팅	RHCOS, RHEL 7.9 또는 RHEL 8.4 [3]	2	8GB	100GB	300

1. SMT(동시 멀티스레딩) 또는 하이퍼 스레딩이 활성화되지 않은 경우 하나의 vCPU는 하나의 물리적 코어와 동일합니다. 활성화하면 다음과 같은 공식을 사용하여 해당 비율을 계산합니다. (코어당 스레드 수 × 코어 수) × 소켓 수 = vCPU 수

2. OpenShift Container Platform 및 Kubernetes는 디스크 성능에 민감하며 특히 10ms p99 fsync 기간이 필요한 컨트롤 플레인 노드의 etcd에 더 빠른 스토리지가 권장됩니다. 많은 클라우드 플랫폼에서 스토리지 크기와 IOPS를 함께 확장되므로 충분한 성능을 얻으려면 스토리지 볼륨을 과도하게 할당해야 할 수 있습니다.

3. 사용자가 프로비저닝한 모든 설치와 마찬가지로 클러스터에서 RHEL 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. RHEL 7 컴퓨팅 머신 사용은 더 이상 사용되지 않으며 향후 OpenShift Container Platform 4 릴리스에서 제거될 예정입니다.

7.12.5.3. 사용자 정의 머신 유형 사용

OpenShift Container Platform 클러스터를 설치하기 위해 사용자 정의 머신 유형을 사용할 수 있습니다.

사용자 지정 머신 유형을 사용할 때는 다음을 고려하십시오.

- 사전 정의된 인스턴스 유형과 유사하게 사용자 정의 머신 유형은 컨트롤 플레인 및 컴퓨팅 시스템의 최소 리소스 요구 사항을 충족해야 합니다. 자세한 내용은 "클러스터 설치를 위한 최소 리소스 요구 사항"을 참조하십시오.

- 사용자 정의 머신 유형의 이름은 다음 구문을 준수해야 합니다.
  custom-<number_of_cpus>-<amount_of_memory_in_mb>

  예를 들면 **custom-6-20480** 입니다.

7.12.6. GCP용 설치 파일 생성

사용자 프로비저닝 인프라를 사용하는 OpenShift Container Platform을 GCP(Google Cloud Platform)에 설치하려면 설치 프로그램에서 클러스터를 배포하는 데 필요한 파일을 생성하고 클러스터가 사용할 시스템만을 생성하도록 파일을 수정해야 합니다. **install-config.yaml** 파일, Kubernetes 매니페스트 및 Ignition 구성 파일을 생성하고 사용자 지정합니다. 또한 설치 준비 단계에서 별도의 **var** 파티션을 먼저 설정할 수 있는 옵션이 있습니다.

7.12.6.1. 선택 사항: 별도의 /var 파티션 만들기

OpenShift Container Platform의 디스크 파티션 설정은 설치 프로그램에 맡기는 것이 좋습니다. 그러나 확장하려는 파일 시스템의 일부에 별도의 파티션을 생성해야 하는 경우가 있습니다.

OpenShift 컨테이너 플랫폼은 **/var** 파티션 또는 **/var**의 하위 디렉터리 중 하나에 스토리지를 연결하는 단일 파티션의 추가를 지원합니다. 예를 들면 다음과 같습니다.

- **/var/lib/containers**: 시스템에 더 많은 이미지와 컨테이너가 추가됨에 따라 확장될 수 있는 컨테이너 관련 콘텐츠를 보관합니다.

- **/var/lib/etcd**: etcd 스토리지의 성능 최적화와 같은 목적으로 별도로 보관할 데이터를 보관합니다.
- **/var**: 감사 등의 목적에 맞게 별도로 분리하여 보관해야 하는 데이터를 보관합니다.

/var 디렉터리의 콘텐츠를 별도로 저장하면 필요에 따라 해당 영역에 대한 스토리지 확장을 보다 용이하게 하고 나중에 OpenShift Container Platform을 다시 설치하여 해당 데이터를 그대로 보존할 수 있습니다. 이 방법을 사용하면 모든 컨테이너를 다시 가져올 필요가 없으며 시스템을 업데이트할 때 대용량 로그 파일을 복사할 필요도 없습니다.

RHCOS(Red Hat Enterprise Linux CoreOS)를 새로 설치하기 전에 **/var가 있어야 하므로 다음 절차에서는 OpenShift Container Platform 설치의 openshift-install 준비 단계 중에 삽입되는 머신 구성 매니페스트를 생성하여 별도의 /var** 파티션을 설정합니다.

중요

이 절차에서 별도의 **/var** 파티션을 생성하기 위해 단계에 따라 이 섹션의 뒷부분에서 설명한 대로 Kubernetes 매니페스트 및 Ignition 구성 파일을 다시 생성할 필요가 없습니다.

프로세스

1. OpenShift Container Platform 설치 파일을 저장할 디렉터리를 만듭니다.

```
$ mkdir $HOME/clusterconfig
```

2. **openshift-install**을 실행하여 **manifest** 및 **openshift** 하위 디렉터리에 파일 세트를 만듭니다. 프롬프트가 표시되면 시스템 질문에 대답합니다.

```
$ openshift-install create manifests --dir $HOME/clusterconfig
```

출력 예

```
? SSH Public Key ...
INFO Credentials loaded from the "myprofile" profile in file "/home/myuser/.aws/credentials"
INFO Consuming Install Config from target directory
INFO Manifests created in: $HOME/clusterconfig/manifests and
$HOME/clusterconfig/openshift
```

3. 선택 사항: 설치 프로그램이 **clusterconfig/openshift** 디렉터리에 매니페스트를 생성했는지 확인합니다.

```
$ ls $HOME/clusterconfig/openshift/
```

출력 예

```
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

4. 추가 파티션을 구성하는 Butane 구성을 생성합니다. 예를 들어 파일 이름을 **$HOME/clusterconfig/98-var-partition.bu** 로 지정하고, 디스크 장치 이름을 **worker** 시스템의 스토리지 장치 이름으로 변경하고 스토리지 크기를 적절하게 설정합니다. 이 예에서는 **/var** 디렉

토리를 별도의 파티션에 배치합니다.

```
variant: openshift
version: 4.9.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1 파티션을 설정해야하는 디스크 저장 장치 이름입니다.

2 데이터 파티션을 부트 디스크에 추가할 때 최소 25000MiB(메비 바이트)가 권장됩니다. 루트 파일 시스템은 지정된 오프셋까지 사용 가능한 모든 공간을 채우기 위해 자동으로 크기가 조정됩니다. 값이 지정되지 않거나 지정된 값이 권장 최소값보다 작으면 생성되는 루트 파일 시스템의 크기가 너무 작아지고 RHCOS를 나중에 다시 설치할 때 데이터 파티션의 첫 번째 부분을 덮어 쓸 수 있습니다.

3 데이터 파티션의 크기(MB)입니다.

4 컨테이너 스토리지에 사용되는 파일 시스템에 **prjquota** 마운트 옵션을 활성화해야 합니다.

참고

별도의 **/var** 파티션을 만들 때 다른 인스턴스 유형에 동일한 장치 이름이 없는 경우 작업자 노드에 다른 인스턴스 유형을 사용할 수 없습니다.

5. Butane 구성에서 매니페스트를 생성하여 **clusterconfig/openshift** 디렉터리에 저장합니다. 예를 들어 다음 명령을 실행합니다.

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

6. **openshift-install**을 다시 실행하여 **manifest** 및 **openshift** 하위 디렉터리의 파일 세트에서 Ignition 구성을 만듭니다.

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

이제 Ignition 구성 파일을 설치 절차에 대한 입력으로 사용하여 RHCOS (Red Hat Enterprise Linux CoreOS) 시스템을 설치할 수 있습니다.

7.12.6.2. 설치 구성 파일 만들기

GCP(Google Cloud Platform)에 설치하는 OpenShift Container Platform 클러스터를 사용자 지정할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다. 제한된 네트워크 설치의 경우, 해당 파일은 미러 호스트에 있습니다.
- 미러 레지스트리 작성 중에 생성된 **imageContentSources** 값이 있어야 합니다.
- 미러 레지스트리에 대한 인증서의 내용을 가져옵니다.
- 서브스크립션 수준에서 서비스 주체 권한을 획득합니다.

프로세스

1. **install-config.yaml** 파일을 생성합니다.
   a. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 **<installation_directory>**는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.

중요

비어 있는 디렉터리를 지정합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

   b. 화면에 나타나는 지시에 따라 클라우드에 대한 구성 세부 사항을 입력합니다.
      i. 선택사항: 클러스터 시스템에 액세스하는 데 사용할 SSH 키를 선택합니다.

참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.

      ii. 대상 플랫폼으로 **gcp**를 선택합니다.
      iii. 사용자 컴퓨터에 GCP 계정의 서비스 계정 키가 구성되어 있지 않은 경우, GCP에서 해당 키를 가져와 파일의 내용을 붙여 넣거나 파일의 전체 경로를 입력해야 합니다.

iv. 클러스터를 프로비저닝할 프로젝트 ID를 선택합니다. 구성한 서비스 계정에 의해 기본값이 지정됩니다.

v. 클러스터를 배포할 리전을 선택합니다.

vi. 클러스터를 배포할 기본 도메인을 선택합니다. 기본 도메인은 클러스터용으로 생성한 퍼블릭 DNS 영역에 해당합니다.

vii. 클러스터를 설명할 수 있는 이름을 입력합니다.

viii. Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 붙여넣습니다.

2. 제한된 네트워크에서의 설치에 필요한 추가 정보를 제공하려면 **install-config.yaml** 파일을 편집합니다.

a. 레지스트리의 인증 정보를 포함하도록 **pullSecret** 값을 업데이트합니다.

```
pullSecret: '{"auths":{"<mirror_host_name>:5000": {"auth": "<credentials>","email":
"you@example.com"}}}'
```

<mirror_host_name>의 경우 미러 레지스트리의 인증서에 지정한 레지스트리 도메인 이름을 지정하고 **<credentials>**의 경우 미러 레지스트리에 base64로 인코딩된 사용자 이름 및 암호를 지정합니다.

b. **additionalTrustBundle** 매개변수와 값을 추가합니다.

```
additionalTrustBundle: |
  -----BEGIN CERTIFICATE-----

  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
```

값은 미러 레지스트리에 사용한 인증서 파일의 내용이어야 하며, 신뢰할 수 있는 기존 인증 기관 또는 미러 레지스트리에 대해 생성한 자체 서명 인증서일 수 있습니다.

c. 상위 **platform.gcp** 필드 아래에 클러스터를 설치할 VPC의 네트워크 및 서브넷을 정의합니다.

```
network: <existing_vpc>
controlPlaneSubnet: <control_plane_subnet>
computeSubnet: <compute_subnet>
```

platform.gcp.network의 경우 기존 Google VPC의 이름을 지정합니다.
platform.gcp.controlPlaneSubnet 및 **platform.gcp.computeSubnet**의 경우 각각 컨트롤 플레인 시스템 및 컴퓨팅 시스템을 배포할 기존 서브넷을 지정합니다.

d. 다음과 같은 이미지 콘텐츠 리소스를 추가합니다.

```
imageContentSources:
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: registry.redhat.io/ocp/release
```

이러한 값을 완료하려면 미러 레지스트리 생성 중에 기록한 **imageContentSources**를 사용하십시오.

3. 필요한 **install-config.yaml** 파일을 수정합니다. 사용 가능한 매개변수에 대한 자세한 정보는 **Installation configuration parameters** 섹션에서 확인할 수 있습니다.

4. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.

중요

install-config.yaml 파일은 설치 과정에서 사용됩니다. 이 파일을 재사용하려면 지금 백업해야 합니다.

7.12.6.3. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 HTTP 또는 HTTPS 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 OpenShift Container Platform 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 API에 대한 호출을 포함하여 모든 클러스터 발신(Egress) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.

참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 Red Hat OpenStack Platform (RHOSP)에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. 클러스터 외부에서 HTTP 연결을 구축하는 데 사용할 프록시 URL입니다. URL 스키마는 **http**여야 합니다.
2. 클러스터 외부에서 HTTPS 연결을 구축하는 데 사용할 프록시 URL입니다.
3. 대상 도메인 이름, IP 주소 또는 프록시에서 제외할 기타 네트워크 CIDR로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. *를 사용하여 모든 대상에 대해 프록시를 바이패스합니다.
4. 이 값을 제공하면 설치 프로그램에서 추가 CA 인증서를 보유할 **openshift -config 네임스페이스에 user-ca- bundle** 이라는 구성 맵을 생성합니다. **additionalTrustBundle** 및 하나 이상의 프록시 설정을 제공하는 경우 **프록시** 오브젝트는 **trustedCA** 필드의 **user-ca-bundle** 구성 맵을 참조하도록 구성됩니다. 그러면 Cluster Network Operator에서 **trustedCA** 매개변수에 대해 지정된 콘텐츠를 RHCOS 신뢰 번들과 병합하는 **trusted-ca-bundle** 구성 맵을 생성합니다. 프록시의 ID 인증서를 RHCOS 트러스트 번들에 있는 기관에서 서명하지 않은 경우 **additionalTrustBundle** 필드가 있어야 합니다.

참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 OpenShift Container Platform을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 nil이 됩니다.

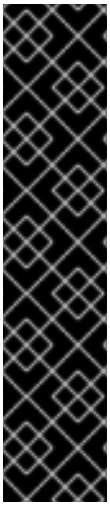
참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

7.12.6.4. Kubernetes 매니페스트 및 Ignition 설정 파일 생성

일부 클러스터 정의 파일을 수정하고 클러스터 시스템을 수동으로 시작해야 하므로 클러스터가 시스템을 구성하는 데 필요한 Kubernetes 매니페스트 및 Ignition 구성 파일을 사용자가 생성해야 합니다.

설치 구성 파일은 Kubernetes 매니페스트로 변환됩니다. 매니페스트는 나중에 클러스터 머신을 구성하는 데 사용되는 Ignition 구성 파일로 래핑됩니다.



중요

- OpenShift Container Platform 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램을 가져오셨습니다. 제한된 네트워크 설치의 경우, 해당 파일은 미러 호스트에 있습니다.
- **install-config.yaml** 설치 구성 파일을 생성하셨습니다.

절차

1. OpenShift Container Platform 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터에 대한 Kubernetes 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>**는 사용자가 만든 **install-config.yaml** 파일이 포함된 설치 디렉터리를 지정합니다.

2. 컨트롤 플레인 시스템을 정의하는 Kubernetes 매니페스트 파일을 제거합니다.

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

이러한 파일을 제거하면 클러스터가 컨트롤 플레인 시스템을 자동으로 생성하지 못합니다.

3. 선택사항: 클러스터가 컴퓨팅 시스템을 프로비저닝하지 않도록 하려면 작업자 시스템을 정의하는 Kubernetes 매니페스트 파일을 제거하십시오.

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

작업자 시스템은 사용자가 직접 생성하고 관리하기 때문에 초기화할 필요가 없습니다.

4. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes 매니페스트 파일의 **mastersSchedulable** 매개변수가 **false**로 설정되어 있는지 확인합니다. 이 설정으로 인해 컨트롤 플레인 머신에서 포드가 예약되지 않습니다.
   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** 파일을 엽니다.
   b. **mastersSchedulable** 매개변수를 찾아서 값을 **False**로 설정되어 있는지 확인합니다.
   c. 파일을 저장하고 종료합니다.
5. 선택사항: Ingress Operator가 사용자 대신 DNS 레코드를 생성하지 못하도록 하려면 **<installation_directory>/manifests/cluster-dns-02-config.yml** DNS 구성 파일에서 **privateZone** 및 **publicZone** 섹션을 제거합니다.

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
```

```
  publicZone: 2
    id: example.openshift.com
status: {}
```

1 2 이 섹션을 완전히 제거합니다.

제거한 경우 나중에 인그레스 DNS 레코드를 수동으로 추가해야 합니다.

6. Ignition 구성 파일을 생성하려면 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행합니다.

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>**는 동일한 설치 디렉터리를 지정합니다.

설치 디렉터리의 부트스트랩, 컨트롤 플레인 및 컴퓨팅 노드에 대한 Ignition 구성 파일이 생성됩니다. **kubeadmin-password** 및 **kubeconfig** 파일은 **./<installation_directory>/auth** 디렉터리에 생성됩니다.

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

추가 리소스

- 선택사항 : 인그레스 DNS 레코드 추가

7.12.7. 공통 변수 내보내기

7.12.7.1. 인프라 이름 추출

Ignition 구성 파일에는 GCP(Google Cloud Platform)에서 클러스터를 고유하게 식별하는 데 사용할 수 있는 고유한 클러스터 ID가 포함되어 있습니다. 인프라 이름은 OpenShift Container Platform 설치 중에 적절한 GCP 리소스를 찾는 데도 사용됩니다. 제공된 Deployment Manager 템플릿에 이 인프라 이름에 대한 참조가 포함되어 있으므로 이름을 추출해야 합니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받으셨습니다.
- 클러스터에 대한 Ignition 구성 파일을 생성하셨습니다.
- **jq** CLI를 설치하셨습니다.

절차

- Ignition 구성 파일 메타데이터에서 인프라 이름을 추출하여 확인하려면 다음 명령을 실행하십시오.

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

출력 예

```
openshift-vw9j6 1
```

1 이 명령의 출력은 클러스터 이름과 임의의 문자열입니다.

7.12.7.2. Deployment Manager 템플릿의 공통 변수 내보내기

GCP(Google Cloud Platform)에서 사용자 제공 인프라 설치를 지원하기 위해 제공된 Deployment Manager 템플릿과 함께 사용되는 공통 변수 세트를 내보내야 합니다.

참고

특정 Deployment Manager 템플릿에는 추가적으로 내보낸 변수도 필요할 수 있습니다. 관련 프로시저에서 자세히 설명합니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.
- **jq** 패키지를 설치합니다.

절차

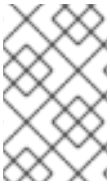
1. 제공된 Deployment Manager 템플릿에서 사용할 다음 공통 변수를 내보냅니다.

```
$ export BASE_DOMAIN='<base_domain>'
$ export BASE_DOMAIN_ZONE_NAME='<base_domain_zone_name>'
$ export NETWORK_CIDR='10.0.0.0/16'
$ export MASTER_SUBNET_CIDR='10.0.0.0/17'
$ export WORKER_SUBNET_CIDR='10.0.128.0/17'

$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
$ export CLUSTER_NAME=`jq -r .clusterName <installation_directory>/metadata.json`
$ export INFRA_ID=`jq -r .infraID <installation_directory>/metadata.json`
$ export PROJECT_NAME=`jq -r .gcp.projectID <installation_directory>/metadata.json`
$ export REGION=`jq -r .gcp.region <installation_directory>/metadata.json`
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

7.12.8. GCP에서 VPC 생성

OpenShift Container Platform 클러스터에서 사용할 VPC를 GCP(Google Cloud Platform)에 생성해야 합니다. 요구사항에 맞춰 VPC를 사용자 지정할 수 있습니다. VPC를 생성하는 한 가지 방법은 제공된 Deployment Manager 템플릿을 수정하는 것입니다.



참고

GCP 인프라를 생성하는 데 제공된 Deployment Manager 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- GCP 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.

프로세스

1. 이 항목의 **VPC에 대한 Deployment Manager 템플릿** 섹션에서 템플릿을 복사하여 사용자 컴퓨터에 **01_vpc.py**로 저장합니다. 이 템플릿은 클러스터에 필요한 VPC를 설명합니다.
2. **01_vpc.yaml** 리소스 정의 파일을 생성합니다.

```
$ cat <<EOF >01_vpc.yaml
imports:
- path: 01_vpc.py

resources:
- name: cluster-vpc
  type: 01_vpc.py
  properties:
    infra_id: '${INFRA_ID}' (1)
    region: '${REGION}' (2)
    master_subnet_cidr: '${MASTER_SUBNET_CIDR}' (3)
    worker_subnet_cidr: '${WORKER_SUBNET_CIDR}' (4)
EOF
```

(1) **infra_id**는 추출 단계에서 가져온 **INFRA_ID** 인프라 이름입니다.

(2) **region**은 클러스터를 배포할 영역입니다(예: **us-central1**).

(3) **master_subnet_cidr**은 마스터 서브넷의 CIDR입니다. (예: **10.0.0.0/17**)

(4) **worker_subnet_cidr**은 작업자 서브넷의 CIDR입니다. (예: **10.0.128.0/17**)

3. **gcloud** CLI를 사용하여 배포를 생성합니다.

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-vpc --config 01_vpc.yaml
```

7.12.8.1. VPC용 Deployment Manager 템플릿

다음 Deployment Manager 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 VPC를 배포할 수 있습니다.

예 7.19. 01_vpc.py Deployment Manager 템플릿

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-network',
        'type': 'compute.v1.network',
        'properties': {
            'region': context.properties['region'],
            'autoCreateSubnetworks': False
        }
    }, {
        'name': context.properties['infra_id'] + '-master-subnet',
        'type': 'compute.v1.subnetwork',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'ipCidrRange': context.properties['master_subnet_cidr']
        }
    }, {
        'name': context.properties['infra_id'] + '-worker-subnet',
        'type': 'compute.v1.subnetwork',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'ipCidrRange': context.properties['worker_subnet_cidr']
        }
    }, {
        'name': context.properties['infra_id'] + '-router',
        'type': 'compute.v1.router',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'nats': [{
                'name': context.properties['infra_id'] + '-nat-master',
                'natIpAllocateOption': 'AUTO_ONLY',
                'minPortsPerVm': 7168,
                'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
                'subnetworks': [{
                    'name': '$(ref.' + context.properties['infra_id'] + '-master-subnet.selfLink)',
                    'sourceIpRangesToNat': ['ALL_IP_RANGES']
                }]
            }, {
                'name': context.properties['infra_id'] + '-nat-worker',
                'natIpAllocateOption': 'AUTO_ONLY',
                'minPortsPerVm': 512,
                'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
                'subnetworks': [{
                    'name': '$(ref.' + context.properties['infra_id'] + '-worker-subnet.selfLink)',
                    'sourceIpRangesToNat': ['ALL_IP_RANGES']
                }]
            }]
        }
    }]

    return {'resources': resources}
```

7.12.9. 사용자 프로비저닝 인프라에 대한 네트워킹 요구사항

모든 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템이 부팅 중에 Ignition 구성 파일을 가져오려면 **initramfs**에 네트워킹을 구성해야 합니다.

7.12.9.1. DHCP를 통해 클러스터 노드의 호스트 이름 설정

RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 호스트 이름은 NetworkManager를 통해 설정됩니다. 기본적으로 시스템은 DHCP를 통해 호스트 이름을 가져옵니다. DHCP에서 호스트 이름을 제공하지 않거나, 커널 인수를 통해 정적으로 설정하거나 다른 방법을 통해 설정하면 역방향 DNS 조회를 통해 얻을 수 있습니다. 역방향 DNS 조회는 노드에서 네트워크를 초기화한 후 수행되며 확인하는 데 시간이 걸릴 수 있습니다. 다른 시스템 서비스는 이 보다 먼저 시작하여 호스트 이름을 **localhost** 등으로 감지할 수 있습니다. DHCP를 사용하여 각 클러스터 노드의 호스트 이름을 제공하여 이 문제를 방지할 수 있습니다.

또한 DHCP를 통해 호스트 이름을 설정하면 DNS 분할 수평 구현 환경에서 수동으로 DNS 레코드 이름 구성 오류를 무시할 수 있습니다.

7.12.9.2. 네트워크 연결 요구사항

OpenShift Container Platform 클러스터 구성 요소가 통신할 수 있도록 시스템 간 네트워크 연결을 구성해야 합니다. 각 시스템에서 클러스터에 있는 다른 모든 시스템의 호스트 이름을 확인할 수 있어야 합니다.

이 섹션에서는 필요한 포트에 대해 자세히 설명합니다.

표 7.56. 모든 시스템 간 통신에 사용되는 포트

프로토콜	포트	설명
ICMP	해당 없음	네트워크 연결성 테스트
TCP	**1936**	메트릭
	9000-**9999**	**9100**-**9101** 포트의 노드 내보내기 및 **9099** 포트의 Cluster Version Operator를 포함한 호스트 수준 서비스.
	10250-**10259**	Kubernetes에서 예약하는 기본 포트
	10256	openshift-sdn
UDP	**4789**	VXLAN 및 Geneve
	6081	VXLAN 및 Geneve
	9000-**9999**	**9100**-**9101** 포트의 노드 내보내기를 포함한 호스트 수준 서비스.
	500	IPsec IKE 패킷
	4500	IPsec NAT-T 패킷

프로토콜	포트	설명
TCP/UDP	**30000-32767**	Kubernetes 노드 포트
ESP	해당 없음	IPsec Encapsulating Security Payload (ESP)

표 7.57. 모든 시스템과 컨트롤 플레인 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**6443**	Kubernetes API

표 7.58. 컨트롤 플레인 머신 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**2379-2380**	etcd 서버 및 피어 포트

7.12.10. GCP에서 로드 밸런서 생성

OpenShift Container Platform 클러스터가 사용할 로드 밸런서를 GCP(Google Cloud Platform)에 구성해야 합니다. 이러한 구성 요소를 생성하는 한 가지 방법은 제공된 Deployment Manager 템플릿을 수정하는 것입니다.

참고

GCP 인프라를 생성하는 데 제공된 Deployment Manager 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- GCP 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.
- GCP에서 VPC 및 관련 서브넷을 생성하고 구성하십시오.

프로세스

1. 이 항목의 **내부 로드 밸런서에 대한 Deployment Manager 템플릿** 섹션에서 템플릿을 복사하여 사용자 컴퓨터에 **02_lb_int.py**로 저장합니다. 이 템플릿에서 클러스터에 필요한 내부 로드 밸런싱 개체를 설명합니다.
2. 외부 클러스터에 대해서도, 이 항목의 **외부 로드 밸런서에 대한 Deployment Manager 템플릿** 섹션에서 템플릿을 복사하여 사용자 컴퓨터에 **02_lb_ext.py**로 저장합니다. 이 템플릿에서 클러스터에 필요한 외부 로드 밸런싱 개체를 설명합니다.
3. 배포 템플릿이 사용하는 변수를 내보냅니다.

a. 클러스터 네트워크 위치를 내보냅니다.

```
$ export CLUSTER_NETWORK=(`gcloud compute networks describe ${INFRA_ID}-network --format json | jq -r .selfLink`)
```

b. 컨트롤 플레인 서브넷 위치를 내보냅니다.

```
$ export CONTROL_SUBNET=(`gcloud compute networks subnets describe ${INFRA_ID}-master-subnet --region=${REGION} --format json | jq -r .selfLink`)
```

c. 클러스터가 사용하는 세 영역을 내보냅니다.

```
$ export ZONE_0=(`gcloud compute regions describe ${REGION} --format=json | jq -r .zones[0] | cut -d "/" -f9`)
```

```
$ export ZONE_1=(`gcloud compute regions describe ${REGION} --format=json | jq -r .zones[1] | cut -d "/" -f9`)
```

```
$ export ZONE_2=(`gcloud compute regions describe ${REGION} --format=json | jq -r .zones[2] | cut -d "/" -f9`)
```

4. **02_infra.yaml** 리소스 정의 파일을 생성합니다.

```
$ cat <<EOF >02_infra.yaml
imports:
- path: 02_lb_ext.py
- path: 02_lb_int.py (1)
resources:
- name: cluster-lb-ext (2)
  type: 02_lb_ext.py
  properties:
    infra_id: '${INFRA_ID}' (3)
    region: '${REGION}' (4)
- name: cluster-lb-int
  type: 02_lb_int.py
  properties:
    cluster_network: '${CLUSTER_NETWORK}'
    control_subnet: '${CONTROL_SUBNET}' (5)
    infra_id: '${INFRA_ID}'
    region: '${REGION}'
    zones: (6)
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'
EOF
```

(1) (2) 외부 클러스터를 배포하는 경우에만 필요합니다.

(3) **infra_id**는 추출 단계에서 가져온 **INFRA_ID** 인프라 이름입니다.

(4) **region**은 클러스터를 배포할 영역입니다(예: **us-central1)**.

(5) **control_subnet**은 컨트롤 서브넷에 대한 URI입니다.

6 **zones**는 **us-east1-b**, **us-east1-c** 및 **us-east1-d**와 같이 컨트롤 플레인 인스턴스를 배포하는 영역입니다.

5. **gcloud** CLI를 사용하여 배포를 생성합니다.

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-infra --config 02_infra.yaml
```

6. 클러스터 IP 주소를 내보냅니다.

```
$ export CLUSTER_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-ip --
region=${REGION} --format json | jq -r .address`)
```

7. 외부 클러스터의 경우 클러스터 공용 IP 주소도 내보냅니다.

```
$ export CLUSTER_PUBLIC_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-
public-ip --region=${REGION} --format json | jq -r .address`)
```

7.12.10.1. 외부 로드 밸런서에 대한 Deployment Manager 템플릿

다음 Deployment Manager 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 외부 로드 밸런서를 배포할 수 있습니다.

예 7.20. 02_lb_ext.py Deployment Manager 템플릿

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-cluster-public-ip',
        'type': 'compute.v1.address',
        'properties': {
            'region': context.properties['region']
        }
    }, {
        # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check
probe for kube-apiserver
        'name': context.properties['infra_id'] + '-api-http-health-check',
        'type': 'compute.v1.httpHealthCheck',
        'properties': {
            'port': 6080,
            'requestPath': '/readyz'
        }
    }, {
        'name': context.properties['infra_id'] + '-api-target-pool',
        'type': 'compute.v1.targetPool',
        'properties': {
            'region': context.properties['region'],
            'healthChecks': ['$(ref.' + context.properties['infra_id'] + '-api-http-health-check.selfLink)'],
            'instances': []
        }
    }, {
        'name': context.properties['infra_id'] + '-api-forwarding-rule',
        'type': 'compute.v1.forwardingRule',
        'properties': {
```

```
            'region': context.properties['region'],
            'IPAddress': '$(ref.' + context.properties['infra_id'] + '-cluster-public-ip.selfLink)',
            'target': '$(ref.' + context.properties['infra_id'] + '-api-target-pool.selfLink)',
            'portRange': '6443'
        }
    }]

    return {'resources': resources}
```

7.12.10.2. 내부 로드 밸런서에 대한 Deployment Manager 템플릿

다음 Deployment Manager 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 내부 로드 밸런서를 배포할 수 있습니다.

예 7.21. 02_lb_int.py Deployment Manager 템플릿

```
def GenerateConfig(context):

    backends = []
    for zone in context.properties['zones']:
        backends.append({
            'group': '$(ref.' + context.properties['infra_id'] + '-master-' + zone + '-instance-group' +
'.selfLink)'
        })

    resources = [{
        'name': context.properties['infra_id'] + '-cluster-ip',
        'type': 'compute.v1.address',
        'properties': {
            'addressType': 'INTERNAL',
            'region': context.properties['region'],
            'subnetwork': context.properties['control_subnet']
        }
    }, {
        # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check
probe for kube-apiserver
        'name': context.properties['infra_id'] + '-api-internal-health-check',
        'type': 'compute.v1.healthCheck',
        'properties': {
            'httpsHealthCheck': {
                'port': 6443,
                'requestPath': '/readyz'
            },
            'type': "HTTPS"
        }
    }, {
        'name': context.properties['infra_id'] + '-api-internal-backend-service',
        'type': 'compute.v1.regionBackendService',
        'properties': {
            'backends': backends,
            'healthChecks': ['$(ref.' + context.properties['infra_id'] + '-api-internal-health-
check.selfLink)'],
            'loadBalancingScheme': 'INTERNAL',
            'region': context.properties['region'],
```

```
            'protocol': 'TCP',
            'timeoutSec': 120
        }
    }, {
        'name': context.properties['infra_id'] + '-api-internal-forwarding-rule',
        'type': 'compute.v1.forwardingRule',
        'properties': {
            'backendService': '$(ref.' + context.properties['infra_id'] + '-api-internal-backend-
service.selfLink)',
            'IPAddress': '$(ref.' + context.properties['infra_id'] + '-cluster-ip.selfLink)',
            'loadBalancingScheme': 'INTERNAL',
            'ports': ['6443','22623'],
            'region': context.properties['region'],
            'subnetwork': context.properties['control_subnet']
        }
    }]

    for zone in context.properties['zones']:
        resources.append({
            'name': context.properties['infra_id'] + '-master-' + zone + '-instance-group',
            'type': 'compute.v1.instanceGroup',
            'properties': {
                'namedPorts': [
                    {
                        'name': 'ignition',
                        'port': 22623
                    }, {
                        'name': 'https',
                        'port': 6443
                    }
                ],
                'network': context.properties['cluster_network'],
                'zone': zone
            }
        })

    return {'resources': resources}
```

외부 클러스터를 생성할 때 **02_lb_ext.py** 템플릿 외에도 이 템플릿이 필요합니다.

7.12.11. GCP에서 프라이빗 DNS 영역 생성

OpenShift Container Platform 클러스터가 사용할 프라이빗 DNS 영역을 GCP(Google Cloud Platform)에 구성해야 합니다. 이 구성 요소를 생성하는 한 가지 방법은 제공된 Deployment Manager 템플릿을 수정하는 것입니다.

참고

GCP 인프라를 생성하는 데 제공된 Deployment Manager 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- GCP 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.
- GCP에서 VPC 및 관련 서브넷을 생성하고 구성하십시오.

프로세스

1. 이 항목의 **프라이빗 DNS에 대한 Deployment Manager 템플릿** 섹션에서 템플릿을 복사하여 사용자 컴퓨터에 **02_dns.py**로 저장합니다. 이 템플릿에서 클러스터에 필요한 프라이빗 DNS 개체를 설명합니다.
2. **02_dns.yaml** 리소스 정의 파일을 생성합니다.

```
$ cat <<EOF >02_dns.yaml
imports:
- path: 02_dns.py

resources:
- name: cluster-dns
  type: 02_dns.py
  properties:
    infra_id: '${INFRA_ID}' 1
    cluster_domain: '${CLUSTER_NAME}.${BASE_DOMAIN}' 2
    cluster_network: '${CLUSTER_NETWORK}' 3
EOF
```

1. **infra_id**는 추출 단계에서 가져온 **INFRA_ID** 인프라 이름입니다.
2. **cluster_domain**은 클러스터의 도메인입니다(예: **openshift.example.com**).
3. **cluster_network**는 클러스터 네트워크에 대한 **selfLink** URL입니다.

3. **gcloud** CLI를 사용하여 배포를 생성합니다.

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-dns --config 02_dns.yaml
```

4. Deployment Manager의 제한으로 인해 템플릿을 통해 DNS 항목이 생성되지 않으므로 수동으로 생성해야 합니다.
   a. 내부 DNS 항목을 추가합니다.

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction add ${CLUSTER_IP} --name
api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-
private-zone
$ gcloud dns record-sets transaction add ${CLUSTER_IP} --name api-
int.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-
private-zone
$ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone
```

   b. 외부 클러스터의 경우 외부 DNS 항목도 추가합니다.

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction add ${CLUSTER_PUBLIC_IP} --name
api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone
${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction execute --zone ${BASE_DOMAIN_ZONE_NAME}
```

7.12.11.1. 프라이빗 DNS에 대한 Deployment Manager 템플릿

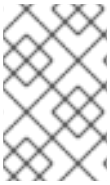
다음 Deployment Manager 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 프라이빗 DNS를 배포할 수 있습니다.

예 7.22. 02_dns.py Deployment Manager 템플릿

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-private-zone',
        'type': 'dns.v1.managedZone',
        'properties': {
            'description': '',
            'dnsName': context.properties['cluster_domain'] + '.',
            'visibility': 'private',
            'privateVisibilityConfig': {
                'networks': [{
                    'networkUrl': context.properties['cluster_network']
                }]
            }
        }
    }]

    return {'resources': resources}
```

7.12.12. GCP에서 방화벽 규칙 생성

OpenShift Container Platform 클러스터에서 사용할 방화벽 규칙을 GCP(Google Cloud Platform)에서 생성해야 합니다. 이러한 구성 요소를 생성하는 한 가지 방법은 제공된 Deployment Manager 템플릿을 수정하는 것입니다.



참고

GCP 인프라를 생성하는 데 제공된 Deployment Manager 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- GCP 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.
- GCP에서 VPC 및 관련 서브넷을 생성하고 구성하십시오.

프로세스

1. 이 항목의 **방화벽 규칙에 대한 Deployment Manager 템플릿** 섹션에서 템플릿을 복사하여 사용자 컴퓨터에 **03_firewall.py**로 저장합니다. 이 템플릿에서 클러스터에 필요한 보안 그룹을 설명합니다.

2. **03_firewall.yaml** 리소스 정의 파일을 생성합니다.

```
$ cat <<EOF >03_firewall.yaml
imports:
- path: 03_firewall.py

resources:
- name: cluster-firewall
  type: 03_firewall.py
  properties:
    allowed_external_cidr: '0.0.0.0/0' (1)
    infra_id: '${INFRA_ID}' (2)
    cluster_network: '${CLUSTER_NETWORK}' (3)
    network_cidr: '${NETWORK_CIDR}' (4)
EOF
```

(1) **allowed_external_cidr**은 부트스트랩 호스트에 클러스터 API 및 SSH 액세스가 가능한 CIDR 범위입니다. 내부 클러스터의 경우 이 값을 **${NETWORK_CIDR}**로 설정합니다.

(2) **infra_id**는 추출 단계에서 가져온 **INFRA_ID** 인프라 이름입니다.

(3) **cluster_network**는 클러스터 네트워크에 대한 **selfLink** URL입니다.

(4) **network_cidr**은 VPC 네트워크의 CIDR입니다(예: **10.0.0.0/16**).

3. **gcloud** CLI를 사용하여 배포를 생성합니다.

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-firewall --config
03_firewall.yaml
```

7.12.12.1. 방화벽 규칙에 대한 Deployment Manager 템플릿

다음 Deployment Manager 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 방화벽 규칙을 배포할 수 있습니다.

예 7.23. 03_firewall.py Deployment Manager 템플릿

```
def GenerateConfig(context):

  resources = [{
    'name': context.properties['infra_id'] + '-bootstrap-in-ssh',
    'type': 'compute.v1.firewall',
    'properties': {
      'network': context.properties['cluster_network'],
      'allowed': [{
        'IPProtocol': 'tcp',
        'ports': ['22']
      }],
```

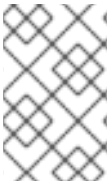
```
            'sourceRanges': [context.properties['allowed_external_cidr']],
            'targetTags': [context.properties['infra_id'] + '-bootstrap']
        }
    }, {
        'name': context.properties['infra_id'] + '-api',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['6443']
            }],
            'sourceRanges': [context.properties['allowed_external_cidr']],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-health-checks',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['6080', '6443', '22624']
            }],
            'sourceRanges': ['35.191.0.0/16', '130.211.0.0/22', '209.85.152.0/22', '209.85.204.0/22'],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-etcd',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['2379-2380']
            }],
            'sourceTags': [context.properties['infra_id'] + '-master'],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-control-plane',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['10257']
            },{
                'IPProtocol': 'tcp',
                'ports': ['10259']
            },{
                'IPProtocol': 'tcp',
                'ports': ['22623']
            }],
            'sourceTags': [
                context.properties['infra_id'] + '-master',
```

```
                context.properties['infra_id'] + '-worker'
            ],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-internal-network',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'icmp'
            },{
                'IPProtocol': 'tcp',
                'ports': ['22']
            }],
            'sourceRanges': [context.properties['network_cidr']],
            'targetTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ]
        }
    }, {
        'name': context.properties['infra_id'] + '-internal-cluster',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'udp',
                'ports': ['4789', '6081']
            },{
                'IPProtocol': 'udp',
                'ports': ['500', '4500']
            },{
                'IPProtocol': 'esp',
            },{
                'IPProtocol': 'tcp',
                'ports': ['9000-9999']
            },{
                'IPProtocol': 'udp',
                'ports': ['9000-9999']
            },{
                'IPProtocol': 'tcp',
                'ports': ['10250']
            },{
                'IPProtocol': 'tcp',
                'ports': ['30000-32767']
            },{
                'IPProtocol': 'udp',
                'ports': ['30000-32767']
            }],
            'sourceTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ],
            'targetTags': [
                context.properties['infra_id'] + '-master',
```

```
            context.properties['infra_id'] + '-worker'
        ]
    }
}]

return {'resources': resources}
```

7.12.13. GCP에서 IAM 역할 생성

OpenShift Container Platform 클러스터에서 사용할 IAM 역할을 GCP(Google Cloud Platform)에 생성해야 합니다. 이러한 구성 요소를 생성하는 한 가지 방법은 제공된 Deployment Manager 템플릿을 수정하는 것입니다.



참고

GCP 인프라를 생성하는 데 제공된 Deployment Manager 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- GCP 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.
- GCP에서 VPC 및 관련 서브넷을 생성하고 구성하십시오.

프로세스

1. 이 항목의 **IAM 역할에 대한 Deployment Manager 템플릿** 섹션에서 템플릿을 복사하여 사용자 컴퓨터에 **03_iam.py**로 저장합니다. 이 템플릿에서 클러스터에 필요한 IAM 역할을 설명합니다.
2. **03_iam.yaml** 리소스 정의 파일을 생성합니다.

```
$ cat <<EOF >03_iam.yaml
imports:
- path: 03_iam.py
resources:
- name: cluster-iam
  type: 03_iam.py
  properties:
    infra_id: '${INFRA_ID}' (1)
EOF
```

(1) **infra_id**는 추출 단계에서 가져온 **INFRA_ID** 인프라 이름입니다.

3. **gcloud** CLI를 사용하여 배포를 생성합니다.

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-iam --config 03_iam.yaml
```

4. 마스터 서비스 계정에 대한 변수를 내보냅니다.

```
$ export MASTER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-m@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

5. 작업자 서비스 계정에 대한 변수를 내보냅니다.

```
$ export WORKER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

6. 컴퓨팅 머신을 호스팅하는 서브넷의 변수를 내보냅니다.

```
$ export COMPUTE_SUBNET=(`gcloud compute networks subnets describe ${INFRA_ID}-
worker-subnet --region=${REGION} --format json | jq -r .selfLink`)
```

7. Deployment Manager의 제한으로 인해 템플릿을 통해 정책 바인딩이 생성되지 않으므로 수동으로 생성해야 합니다.

```
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.instanceAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.networkAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.securityAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/iam.serviceAccountUser"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/storage.admin"

$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/compute.viewer"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/storage.admin"
```

8. 서비스 계정 키를 생성하여 나중에 사용할 수 있도록 로컬로 저장합니다.

```
$ gcloud iam service-accounts keys create service-account-key.json --iam-
account=${MASTER_SERVICE_ACCOUNT}
```

7.12.13.1. IAM 역할에 대한 Deployment Manager 템플릿

다음 Deployment Manager 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 IAM 역할을 배포할 수 있습니다.

예 7.24. 03_iam.py Deployment Manager 템플릿

```
def GenerateConfig(context):

  resources = [{
    'name': context.properties['infra_id'] + '-master-node-sa',
    'type': 'iam.v1.serviceAccount',
    'properties': {
      'accountId': context.properties['infra_id'] + '-m',
      'displayName': context.properties['infra_id'] + '-master-node'
    }
```

```
    }, {
        'name': context.properties['infra_id'] + '-worker-node-sa',
        'type': 'iam.v1.serviceAccount',
        'properties': {
            'accountId': context.properties['infra_id'] + '-w',
            'displayName': context.properties['infra_id'] + '-worker-node'
        }
    }]

    return {'resources': resources}
```

7.12.14. GCP 인프라용 RHCOS 클러스터 이미지 생성

OpenShift Container Platform 노드에 유효한 GCP(Google Cloud Platform)용 RHCOS(Red Hat Enterprise Linux CoreOS) 이미지를 사용해야 합니다.

프로세스

1. RHCOS 이미지 미러 페이지에서 RHCOS 이미지를 가져옵니다.

중요

RHCOS 이미지는 OpenShift Container Platform 릴리스에 따라 변경되지 않을 수 있습니다. 설치하는 OpenShift Container Platform 버전과 같거나 그 이하의 버전 중 가장 최신 버전의 이미지를 다운로드해야 합니다. 지원되는 경우 OpenShift Container Platform 버전과 일치하는 이미지 버전을 사용합니다.

rhcos-<version>-<arch>-gcp.<arch>.tar.gz 형식의 OpenShift Container Platform 버전 번호가 파일 이름에 포함되어 있습니다.

2. Google 스토리지 버킷을 생성합니다.

```
$ gsutil mb gs://<bucket_name>
```

3. RHCOS 이미지를 Google 스토리지 버킷에 업로드합니다.

```
$ gsutil cp <downloaded_image_file_path>/rhcos-<version>-x86_64-gcp.x86_64.tar.gz
gs://<bucket_name>
```

4. 업로드된 RHCOS 이미지 위치를 변수로 내보냅니다.

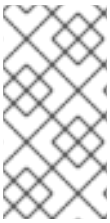
```
$ export IMAGE_SOURCE=gs://<bucket_name>/rhcos-<version>-x86_64-gcp.x86_64.tar.gz
```

5. 클러스터 이미지를 생성합니다.

```
$ gcloud compute images create "${INFRA_ID}-rhcos-image" \
    --source-uri="${IMAGE_SOURCE}"
```

7.12.15. GCP에서 부트스트랩 시스템 생성

OpenShift Container Platform 클러스터 초기화 과정에서 사용할 부트스트랩 시스템을 GCP(Google Cloud Platform)에 생성해야 합니다. 이 시스템을 생성하는 한 가지 방법은 제공된 Deployment Manager 템플릿을 수정하는 것입니다.



참고

부트스트랩 시스템을 생성하는 데 제공된 Deployment Manager 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- GCP 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.
- GCP에서 VPC 및 관련 서브넷을 생성하고 구성하십시오.
- GCP에서 네트워킹 및 로드 밸런서를 생성하고 구성하십시오.
- 컨트롤 플레인 및 컴퓨팅 역할을 생성합니다.
- pyOpenSSL이 설치되어 있는지 확인하십시오.

프로세스

1. 이 항목의 **부트스트랩 시스템에 대한 Deployment Manager 템플릿** 섹션에서 템플릿을 복사하여 사용자 컴퓨터에 **04_bootstrap.py**로 저장합니다. 이 템플릿은 클러스터에 필요한 부트스트랩 시스템을 설명합니다.
2. 설치 프로그램에 필요한 RHCOS(Red Hat Enterprise Linux CoreOS) 이미지의 위치를 내보냅니다.

```
$ export CLUSTER_IMAGE=(`gcloud compute images describe ${INFRA_ID}-rhcos-image --format json | jq -r .selfLink`)
```

3. 버킷을 생성하고 **bootstrap.ign** 파일을 업로드합니다.

```
$ gsutil mb gs://${INFRA_ID}-bootstrap-ignition
$ gsutil cp <installation_directory>/bootstrap.ign gs://${INFRA_ID}-bootstrap-ignition/
```

4. Ignition 구성 파일에 액세스하는 데 사용할 부트스트랩 인스턴스에 대한 서명된 URL을 생성합니다. 출력에서 URL을 변수로 내보냅니다.

```
$ export BOOTSTRAP_IGN=`gsutil signurl -d 1h service-account-key.json gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign | grep "^gs:" | awk '{print $5}'`
```

5. **04_bootstrap.yaml** 리소스 정의 파일을 생성합니다.

```
$ cat <<EOF >04_bootstrap.yaml
imports:
- path: 04_bootstrap.py

resources:
```

```
- name: cluster-bootstrap
  type: 04_bootstrap.py
  properties:
    infra_id: '${INFRA_ID}' (1)
    region: '${REGION}' (2)
    zone: '${ZONE_0}' (3)

    cluster_network: '${CLUSTER_NETWORK}' (4)
    control_subnet: '${CONTROL_SUBNET}' (5)
    image: '${CLUSTER_IMAGE}' (6)
    machine_type: 'n1-standard-4' (7)
    root_volume_size: '128' (8)

    bootstrap_ign: '${BOOTSTRAP_IGN}' (9)
EOF
```

(1) **infra_id**는 추출 단계에서 가져온 **INFRA_ID** 인프라 이름입니다.

(2) **region**은 클러스터를 배포할 영역입니다(예: **us-central1**).

(3) **zone**은 부트스트랩 인스턴스를 배포할 영역입니다(예: **us-central1-b**).

(4) **cluster_network**는 클러스터 네트워크에 대한 **selfLink** URL입니다.

(5) **control_subnet**은 컨트롤 서브넷에 대한 **selfLink** URL입니다.

(6) **image**는 RHCOS 이미지에 대한 **selfLink** URL입니다.

(7) **machine_type**은 인스턴스의 시스템 유형입니다(예: **n1-standard-4**).

(8) **root_volume_size**는 부트스트랩 시스템의 부팅 디스크 크기입니다.

(9) **bootstrap_ign**은 서명된 URL을 생성할 때 출력되는 URL입니다.

6. **gcloud** CLI를 사용하여 배포를 생성합니다.

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-bootstrap --config
04_bootstrap.yaml
```

7. Deployment Manager의 제한으로 인해 템플릿을 통해 로드 밸런서 멤버십이 관리되지 않으므로 수동으로 부트스트랩 시스템을 추가해야 합니다.

   a. 내부 로드 밸런서 인스턴스 그룹에 부트스트랩 인스턴스를 추가합니다.

```
$ gcloud compute instance-groups unmanaged add-instances \
    ${INFRA_ID}-bootstrap-instance-group --zone=${ZONE_0} --instances=${INFRA_ID}-
bootstrap
```

   b. 내부 로드 밸런서 백엔드 서비스에 부트스트랩 인스턴스 그룹을 추가합니다.

```
$ gcloud compute backend-services add-backend \
    ${INFRA_ID}-api-internal-backend-service --region=${REGION} --instance-
group=${INFRA_ID}-bootstrap-instance-group --instance-group-zone=${ZONE_0}
```

7.12.15.1. 부트스트랩 시스템에 대한 Deployment Manager 템플릿

다음 Deployment Mananger 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 부트스트랩 시스템을 배포할 수 있습니다.

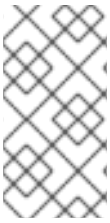
예 7.25. 04_bootstrap.py Deployment Manager 템플릿

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-bootstrap-public-ip',
        'type': 'compute.v1.address',
        'properties': {
            'region': context.properties['region']
        }
    }, {
        'name': context.properties['infra_id'] + '-bootstrap',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': '{"ignition":{"config":{"replace":{"source":"' + context.properties['bootstrap_ign']
+ '"}},"version":"3.1.0"}}',
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet'],
                'accessConfigs': [{
                    'natIP': '$(ref.' + context.properties['infra_id'] + '-bootstrap-public-ip.address)'
                }]
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                    context.properties['infra_id'] + '-bootstrap'
                ]
            },
            'zone': context.properties['zone']
        }
    }, {
        'name': context.properties['infra_id'] + '-bootstrap-instance-group',
        'type': 'compute.v1.instanceGroup',
        'properties': {
            'namedPorts': [
                {
```

```
                'name': 'ignition',
                'port': 22623
            }, {
                'name': 'https',
                'port': 6443
            }
        ],
        'network': context.properties['cluster_network'],
        'zone': context.properties['zone']
    }
}]

return {'resources': resources}
```

7.12.16. GCP에 컨트롤 플레인 시스템 생성

클러스터에서 사용할 컨트롤 플레인 시스템을 GCP(Google Cloud Platform)에 생성해야 합니다. 이러한 시스템을 생성하는 한 가지 방법은 제공된 Deployment Manager 템플릿을 수정하는 것입니다.



참고

컨트롤 플레인 시스템을 생성하는 데 제공된 Deployment Manager 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- GCP 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.
- GCP에서 VPC 및 관련 서브넷을 생성하고 구성하십시오.
- GCP에서 네트워킹 및 로드 밸런서를 생성하고 구성하십시오.
- 컨트롤 플레인 및 컴퓨팅 역할을 생성합니다.
- 부트스트랩 시스템을 생성합니다.

프로세스

1. 이 항목의 **컨트롤 플레인 시스템에 대한 Deployment Manager 템플릿** 섹션에서 템플릿을 복사하여 사용자 컴퓨터에 **05_control_plane.py**로 저장합니다. 이 템플릿은 클러스터에 필요한 컨트롤 플레인 시스템을 설명합니다.
2. 리소스 정의에 필요한 다음 변수를 내보냅니다.

```
$ export MASTER_IGNITION=`cat <installation_directory>/master.ign`
```

3. **05_control_plane.yaml** 리소스 정의 파일을 생성합니다.

```
$ cat <<EOF >05_control_plane.yaml
```

```
imports:
- path: 05_control_plane.py

resources:
- name: cluster-control-plane
  type: 05_control_plane.py
  properties:
    infra_id: '${INFRA_ID}' 1
    zones: 2
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'

    control_subnet: '${CONTROL_SUBNET}' 3
    image: '${CLUSTER_IMAGE}' 4
    machine_type: 'n1-standard-4' 5
    root_volume_size: '128'
    service_account_email: '${MASTER_SERVICE_ACCOUNT}' 6

    ignition: '${MASTER_IGNITION}' 7
EOF
```

1. **infra_id**는 추출 단계에서 가져온 **INFRA_ID** 인프라 이름입니다.
2. **zones**은 컨트롤 플레인 인스턴스를 배포할 영역입니다(예: **us-central1-a**, **us-central1-b**, **us-central1-c**).
3. **control_subnet**은 컨트롤 서브넷에 대한 **selfLink** URL입니다.
4. **image**는 RHCOS 이미지에 대한 **selfLink** URL입니다.
5. **machine_type**은 인스턴스의 시스템 유형입니다(예: **n1-standard-4**).
6. **service_account_email**은 사용자가 생성한 마스터 서비스 계정의 이메일 주소입니다.
7. **ignition**은 **master.ign** 파일의 내용입니다.

4. **gcloud** CLI를 사용하여 배포를 생성합니다.

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-control-plane --config
05_control_plane.yaml
```

5. Deployment Manager의 제한으로 인해 템플릿을 통해 로드 밸런서 멤버십이 관리되지 않으므로 수동으로 컨트롤 플레인 시스템을 추가해야 합니다.

- 다음 명령을 실행하여 적절한 인스턴스 그룹에 컨트롤 플레인 머신을 추가합니다.

```
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_0}-instance-group --zone=${ZONE_0} --instances=${INFRA_ID}-master-0
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_1}-instance-group --zone=${ZONE_1} --instances=${INFRA_ID}-master-1
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_2}-instance-group --zone=${ZONE_2} --instances=${INFRA_ID}-master-2
```

- 외부 클러스터의 경우 다음 명령을 실행하여 대상 풀에 컨트롤 플레인 머신을 추가해야 합니다.

```
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-zone="${ZONE_0}" --instances=${INFRA_ID}-master-0
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-zone="${ZONE_1}" --instances=${INFRA_ID}-master-1
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-zone="${ZONE_2}" --instances=${INFRA_ID}-master-2
```

7.12.16.1. 컨트롤 플레인 시스템에 대한 Deployment Manager 템플릿

다음 Deployment Mananger 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 컨트롤 플레인 시스템을 배포할 수 있습니다.

예 7.26. 05_control_plane.py Deployment Manager 템플릿

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-master-0',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'diskType': 'zones/' + context.properties['zones'][0] + '/diskTypes/pd-ssd',
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zones'][0] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                ]
            },
            'zone': context.properties['zones'][0]
        }
    }, {
```

```
        'name': context.properties['infra_id'] + '-master-1',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'diskType': 'zones/' + context.properties['zones'][1] + '/diskTypes/pd-ssd',
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zones'][1] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                ]
            },
            'zone': context.properties['zones'][1]
        }
    }, {
        'name': context.properties['infra_id'] + '-master-2',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'diskType': 'zones/' + context.properties['zones'][2] + '/diskTypes/pd-ssd',
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zones'][2] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
```

```
            'subnetwork': context.properties['control_subnet']
        }],
        'serviceAccounts': [{
            'email': context.properties['service_account_email'],
            'scopes': ['https://www.googleapis.com/auth/cloud-platform']
        }],
        'tags': {
            'items': [
                context.properties['infra_id'] + '-master',
            ]
        },
        'zone': context.properties['zones'][2]
    }
}]

return {'resources': resources}
```

7.12.17. 부트스트랩이 완료되길 기다렸다가 GCP에서 부트스트랩 리소스를 제거합니다.

GCP(Google Cloud Platform)에 필요한 인프라를 모두 생성한 후 설치 프로그램에 의해 생성된 Ignition 구성 파일을 사용하여 프로비저닝한 시스템에서 부트스트랩 프로세스가 완료될 때까지 기다립니다.

사전 요구 사항

- GCP 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.
- GCP에서 VPC 및 관련 서브넷을 생성하고 구성하십시오.
- GCP에서 네트워킹 및 로드 밸런서를 생성하고 구성하십시오.
- 컨트롤 플레인 및 컴퓨팅 역할을 생성합니다.
- 부트스트랩 시스템을 생성합니다.
- 컨트롤 플레인 시스템을 생성합니다.

프로세스

1. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

```
$ ./openshift-install wait-for bootstrap-complete --dir <installation_directory> \ 1
    --log-level info 2
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2 다른 설치 세부 사항을 보려면 **info** 대신 **warn**, **debug** 또는 **error**를 지정합니다.

FATAL 경고 없이 명령이 종료되면 프로덕션 컨트롤 플레인이 초기화된 것입니다.

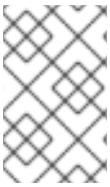
2. 부트스트랩 리소스를 삭제합니다.

```
$ gcloud compute backend-services remove-backend ${INFRA_ID}-api-internal-backend-
service --region=${REGION} --instance-group=${INFRA_ID}-bootstrap-instance-group --
instance-group-zone=${ZONE_0}
$ gsutil rm gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign
$ gsutil rb gs://${INFRA_ID}-bootstrap-ignition
$ gcloud deployment-manager deployments delete ${INFRA_ID}-bootstrap
```

7.12.18. GCP에 추가 작업자 시스템 생성

개별 인스턴스를 따로 시작하거나 자동 확장 그룹과 같은 클러스터 외부의 자동화된 프로세스를 통해 클러스터에서 사용할 작업자 시스템을 GCP(Google Cloud Platform)에 생성할 수 있습니다. OpenShift Container Platform의 기본 제공 클러스터 확장 메커니즘과 시스템 API를 활용할 수도 있습니다.

예에서는 Deployment Manager 템플릿을 사용하여 인스턴스 하나를 수동으로 시작합니다. 파일에 **06_worker.py** 유형의 추가 리소스를 포함시켜 추가 인스턴스를 시작할 수 있습니다.



참고

작업자 시스템을 생성하는 데 제공된 Deployment Manager 템플릿을 사용하지 않는 경우, 제공된 정보를 검토하고 수동으로 인프라를 생성해야 합니다. 클러스터가 올바르게 초기화되지 않은 경우, Red Hat 지원팀에 설치 로그를 제시하여 문의해야 할 수도 있습니다.

사전 요구 사항

- GCP 계정을 구성하십시오.
- 클러스터에 대한 Ignition 구성 파일을 생성하십시오.
- GCP에서 VPC 및 관련 서브넷을 생성하고 구성하십시오.
- GCP에서 네트워킹 및 로드 밸런서를 생성하고 구성하십시오.
- 컨트롤 플레인 및 컴퓨팅 역할을 생성합니다.
- 부트스트랩 시스템을 생성합니다.
- 컨트롤 플레인 시스템을 생성합니다.

프로세스

1. 이 항목의 **작업자 시스템에 대한 Deployment Manager 템플릿** 섹션에서 템플릿을 복사하여 사용자 컴퓨터에 **06_worker.py**로 저장합니다. 이 템플릿에서 클러스터에 필요한 작업자 시스템을 설명합니다.
2. 리소스 정의가 사용하는 변수를 내보냅니다.
   a. 컴퓨팅 시스템을 호스팅하는 서브넷을 내보냅니다.

   ```
   $ export COMPUTE_SUBNET=(`gcloud compute networks subnets describe
   ${INFRA_ID}-worker-subnet --region=${REGION} --format json | jq -r .selfLink`)
   ```

   b. 서비스 계정의 이메일 주소를 내보냅니다.

   ```
   $ export WORKER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
   "email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
   ```

c. 컴퓨팅 시스템 Ignition 구성 파일의 위치를 내보냅니다.

```
$ export WORKER_IGNITION=`cat <installation_directory>/worker.ign`
```

3. **06_worker.yaml** 리소스 정의 파일을 생성합니다.

```
$ cat <<EOF >06_worker.yaml
imports:
- path: 06_worker.py

resources:
- name: 'worker-0' (1)
  type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}' (2)
    zone: '${ZONE_0}' (3)
    compute_subnet: '${COMPUTE_SUBNET}' (4)
    image: '${CLUSTER_IMAGE}' (5)
    machine_type: 'n1-standard-4' (6)
    root_volume_size: '128'
    service_account_email: '${WORKER_SERVICE_ACCOUNT}' (7)
    ignition: '${WORKER_IGNITION}' (8)
- name: 'worker-1'
  type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}' (9)
    zone: '${ZONE_1}' (10)
    compute_subnet: '${COMPUTE_SUBNET}' (11)
    image: '${CLUSTER_IMAGE}' (12)
    machine_type: 'n1-standard-4' (13)
    root_volume_size: '128'
    service_account_email: '${WORKER_SERVICE_ACCOUNT}' (14)
    ignition: '${WORKER_IGNITION}' (15)
EOF
```

(1) **name**은 작업자 머신의 이름입니다(예: **worker-0**).

(2) (9) **infra_id**는 추출 단계에서 가져온 **INFRA_ID** 인프라 이름입니다.

(3) (10) **zone**은 작업자 시스템을 배포할 영역입니다(예: **us-central1-a**).

(4) (11) **compute_subnet**은 컴퓨팅 서브넷에 대한 **selfLink** URL입니다.

(5) (12) **image**는 RHCOS 이미지에 대한 **selfLink** URL입니다. [1]

(6) (13) **machine_type**은 인스턴스의 시스템 유형입니다(예: **n1-standard-4**).

(7) (14) **service_account_email**은 사용자가 생성한 작업자 서비스 계정의 이메일 주소입니다.

(8) (15) **ignition**은 **worker.ign** 파일의 내용입니다.

4. 선택사항: 추가 인스턴스를 시작하려면 **06_worker.yaml** 리소스 정의 파일에 **06_worker.py** 형식의 추가 리소스를 포함시키십시오.

5. **gcloud** CLI를 사용하여 배포를 생성합니다.

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-worker --config
06_worker.yaml
```

1. GCP Marketplace 이미지를 사용하려면 사용할 오퍼를 지정합니다.

   - OpenShift Container Platform: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-ocp-48-x86-64-202210040145**
   - OpenShift Platform Plus: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-opp-48-x86-64-202206140145**
   - OpenShift Kubernetes Engine: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-oke-48-x86-64-202206140145**

7.12.18.1. 작업자 시스템에 대한 Deployment Manager 템플릿

다음 Deloyment Manager 템플릿을 사용하여 OpenShift Container Platform 클러스터에 필요한 작업자 시스템을 배포할 수 있습니다.

예 7.27. 06_worker.py Deployment Manager 템플릿

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-' + context.env['name'],
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['compute_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
```

```
        }],
        'tags': {
            'items': [
                context.properties['infra_id'] + '-worker',
            ]
        },
        'zone': context.properties['zone']
    }
}]

return {'resources': resources}
```

7.12.19. CLI를 사용하여 클러스터에 로그인

클러스터 **kubeconfig** 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. **kubeconfig** 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- **oc** CLI를 설치했습니다.

프로세스

1. **kubeadmin** 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 **oc** 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

7.12.20. 기본 OperatorHub 소스 비활성화

Red Hat 및 커뮤니티 프로젝트에서 제공하는 콘텐츠를 소싱하는 Operator 카탈로그는 OpenShift Container Platform을 설치하는 동안 기본적으로 OperatorHub용으로 구성됩니다. 제한된 네트워크 환경에서는 클러스터 관리자로서 기본 카탈로그를 비활성화해야 합니다.

프로세스

- **OperatorHub** 오브젝트에 **disableAllDefaultSources: true**를 추가하여 기본 카탈로그의 소스를 비활성화합니다.

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

작은 정보

또는 웹 콘솔을 사용하여 카탈로그 소스를 관리할 수 있습니다. **관리 → 클러스터 설정 → 구성 → OperatorHub** 페이지에서 개별 소스를 생성, 삭제, 비활성화 및 활성화할 수 있는 소스 **탭을** 클릭합니다.

7.12.21. 머신의 인증서 서명 요청 승인

클러스터에 시스템을 추가하면 추가한 시스템별로 보류 중인 인증서 서명 요청(CSR)이 두 개씩 생성됩니다. 이러한 CSR이 승인되었는지 확인해야 하며, 필요한 경우 이를 직접 승인해야 합니다. 클라이언트 요청을 먼저 승인한 다음 서버 요청을 승인해야 합니다.

사전 요구 사항

- 클러스터에 시스템을 추가했습니다.

프로세스

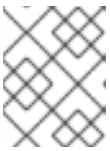
1. 클러스터가 시스템을 인식하는지 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0 Ready    master  63m  v1.22.1
master-1 Ready    master  63m  v1.22.1
master-2 Ready    master  64m  v1.22.1
```

출력에 생성된 모든 시스템이 나열됩니다.



참고

이전 출력에는 일부 CSR이 승인될 때까지 컴퓨팅 노드(작업자 노드라고도 함)가 포함되지 않을 수 있습니다.

2. 보류 중인 CSR을 검토하고 클러스터에 추가한 각 시스템에 대해 **Pending** 또는 **Approved** 상태의 클라이언트 및 서버 요청이 표시되는지 확인합니다.

```
$ oc get csr
```

출력 예

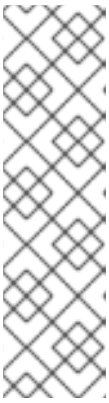
```
NAME        AGE    REQUESTOR                                                   CONDITION
csr-8b2br   15m    system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
csr-8vnps   15m    system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
...
```

예에서는 두 시스템이 클러스터에 참여하고 있습니다. 목록에는 승인된 CSR이 더 많이 나타날 수도 있습니다.

3. CSR이 승인되지 않은 경우, 추가된 시스템에 대한 모든 보류 중인 CSR이 **Pending** 상태로 전환된 후 클러스터 시스템의 CSR을 승인합니다.

참고

CSR은 교체 주기가 자동으로 만료되므로 클러스터에 시스템을 추가한 후 1시간 이내에 CSR을 승인하십시오. 한 시간 내에 승인하지 않으면 인증서가 교체되고 각 노드에 대해 두 개 이상의 인증서가 표시됩니다. 이러한 인증서를 모두 승인해야 합니다. 클라이언트 CSR이 승인되면 Kubelet은 인증서에 대한 보조 CSR을 생성하므로 수동 승인이 필요합니다. 그러면 Kubelet에서 동일한 매개변수를 사용하여 새 인증서를 요청하는 경우 인증서 갱신 요청은 **machine-approver**에 의해 자동으로 승인됩니다.



참고

베어 메탈 및 기타 사용자 프로비저닝 인프라와 같이 머신 API를 사용하도록 활성화되지 않는 플랫폼에서 실행되는 클러스터의 경우 CSR(Kubelet service Certificate Request)을 자동으로 승인하는 방법을 구현해야 합니다. 요청이 승인되지 않으면 API 서버가 kubelet에 연결될 때 서비스 인증서가 필요하므로 **oc exec**, **oc rsh**, **oc logs** 명령을 성공적으로 수행할 수 없습니다. Kubelet 엔드 포인트에 연결하는 모든 작업을 수행하려면 이 인증서 승인이 필요합니다. 이 방법은 새 CSR을 감시하고 CSR이 **system:node** 또는 **system:admin** 그룹의 **node-bootstrapper** 서비스 계정에 의해 제출되었는지 확인하고 노드의 ID를 확인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>**은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

참고

일부 Operator는 일부 CSR이 승인될 때까지 사용할 수 없습니다.

4. 이제 클라이언트 요청이 승인되었으므로 클러스터에 추가한 각 머신의 서버 요청을 검토해야 합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE    REQUESTOR                                                   CONDITION
csr-bfd72   5m26s  system:node:ip-10-0-50-126.us-east-2.compute.internal
```

```
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 나머지 CSR이 승인되지 않고 **Pending** 상태인 경우 클러스터 머신의 CSR을 승인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>**은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. 모든 클라이언트 및 서버 CSR이 승인된 후 머신은 **Ready** 상태가 됩니다. 다음 명령을 실행하여 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME      STATUS   ROLES   AGE  VERSION
master-0  Ready    master  73m  v1.22.1
master-1  Ready    master  73m  v1.22.1
master-2  Ready    master  74m  v1.22.1
worker-0  Ready    worker  11m  v1.22.1
worker-1  Ready    worker  11m  v1.22.1
```

참고

머신이 **Ready** 상태로 전환하는 데 서버 CSR의 승인 후 몇 분이 걸릴 수 있습니다.

추가 정보

- CSR에 대한 자세한 내용은 인증서 서명 요청을 참조하십시오.

7.12.22. 선택사항 : 인그레스 DNS 레코드 추가

Kubernetes 매니페스트를 생성하고 Ignition 구성을 생성할 때 DNS 영역 구성을 제거한 경우, 인그레스 로드 밸런서를 가리키는 DNS 레코드를 수동으로 생성해야 합니다. 와일드카드 ***.apps.{baseDomain}.** 또는 특정 레코드를 생성할 수 있습니다. 사용자 요구사항에 따라 A, CNAME 및 기타 레코드를 사용할 수 있습니다.

사전 요구 사항

- GCP 계정을 구성하십시오.
- Kubernetes 매니페스트를 생성하고 Ignition 구성을 생성할 때 DNS 영역 구성을 제거하십시오.

- GCP에서 VPC 및 관련 서브넷을 생성하고 구성하십시오.
- GCP에서 네트워킹 및 로드 밸런서를 생성하고 구성하십시오.
- 컨트롤 플레인 및 컴퓨팅 역할을 생성합니다.
- 부트스트랩 시스템을 생성합니다.
- 컨트롤 플레인 시스템을 생성합니다.
- 작업자 시스템을 생성합니다.

프로세스

1. 인그레스 라우터가 로드 밸런서를 생성하고 **EXTERNAL-IP** 필드를 채울 때까지 기다립니다.

```
$ oc -n openshift-ingress get service router-default
```

출력 예

```
NAME            TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)                      AGE
router-default  LoadBalancer   172.30.18.154   35.233.157.184
80:32288/TCP,443:31215/TCP   98
```

2. 영역에 A 레코드를 추가합니다.
   - A 레코드를 사용하려면,
     i. 라우터 IP 주소에 대한 변수를 내보냅니다.

```
$ export ROUTER_IP=`oc -n openshift-ingress get service router-default --no-
headers | awk '{print $4}'`
```

     ii. 프라이빗 영역에 A 레코드를 추가합니다.

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction add ${ROUTER_IP} --name
\*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone
${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone
```

     iii. 외부 클러스터의 경우 퍼블릭 영역에 A 레코드를 추가합니다.

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction add ${ROUTER_IP} --name
\*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone
${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction execute --zone
${BASE_DOMAIN_ZONE_NAME}
```

   - 와일드카드를 사용하지 않고 명시적 도메인을 추가하려면 클러스터의 현재 경로에 대한 항목을 생성합니다.

```
$ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}
{"\n"}{end}{end}' routes
```

출력 예

```
oauth-openshift.apps.your.cluster.domain.example.com
console-openshift-console.apps.your.cluster.domain.example.com
downloads-openshift-console.apps.your.cluster.domain.example.com
alertmanager-main-openshift-monitoring.apps.your.cluster.domain.example.com
grafana-openshift-monitoring.apps.your.cluster.domain.example.com
prometheus-k8s-openshift-monitoring.apps.your.cluster.domain.example.com
```

7.12.23. 사용자 프로비저닝 인프라에서 GCP 설치 완료

GCP(Google Cloud Platform) 사용자 프로비저닝 인프라에서 OpenShift Container Platform 설치를 시작한 후 클러스터가 준비를 마칠 때까지 클러스터 이벤트를 모니터링할 수 있습니다.

사전 요구 사항

- 사용자 프로비저닝 GCP 인프라에 OpenShift Container Platform 클러스터용 부트스트랩 시스템을 배포하십시오.
- **oc** CLI를 설치하고 로그인하십시오.

프로세스

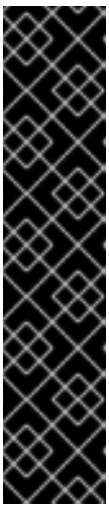
1. 클러스터 설치를 완료합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

출력 예

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.



중요

- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

2. 클러스터의 실행 상태를 관찰합니다.

   a. 다음 명령을 실행하여 현재 클러스터 버전과 상태를 확인합니다.

   ```
   $ oc get clusterversion
   ```

   출력 예

   ```
   NAME     VERSION   AVAILABLE   PROGRESSING   SINCE   STATUS
   version            False       True          24m     Working towards 4.5.4: 99% complete
   ```

   b. 다음 명령을 실행하여 컨트롤 플레인에서 관리하되는 운영자를 CVO(Cluster Version Operator)별로 확인합니다.

   ```
   $ oc get clusteroperators
   ```

   출력 예

   ```
   NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
   SINCE
   authentication                             4.5.4     True        False         False      7m56s
   cloud-credential                           4.5.4     True        False         False      31m
   cluster-autoscaler                         4.5.4     True        False         False      16m
   console                                    4.5.4     True        False         False      10m
   csi-snapshot-controller                    4.5.4     True        False         False      16m
   dns                                        4.5.4     True        False         False      22m
   etcd                                       4.5.4     False       False         False      25s
   image-registry                             4.5.4     True        False         False      16m
   ingress                                    4.5.4     True        False         False      16m
   insights                                   4.5.4     True        False         False      17m
   kube-apiserver                             4.5.4     True        False         False      19m
   kube-controller-manager                    4.5.4     True        False         False      20m
   kube-scheduler                             4.5.4     True        False         False      20m
   kube-storage-version-migrator              4.5.4     True        False         False      16m
   machine-api                                4.5.4     True        False         False      22m
   machine-config                             4.5.4     True        False         False      22m
   marketplace                                4.5.4     True        False         False      16m
   monitoring                                 4.5.4     True        False         False      10m
   network                                    4.5.4     True        False         False      23m
   node-tuning                                4.5.4     True        False         False      23m
   openshift-apiserver                        4.5.4     True        False         False      17m
   openshift-controller-manager               4.5.4     True        False         False      15m
   openshift-samples                          4.5.4     True        False         False      16m
   operator-lifecycle-manager                 4.5.4     True        False         False      22m
   operator-lifecycle-manager-catalog         4.5.4     True        False         False      22m
   operator-lifecycle-manager-packageserver   4.5.4     True        False         False      18m
   service-ca                                 4.5.4     True        False         False      23m
   service-catalog-apiserver                  4.5.4     True        False         False      23m
   service-catalog-controller-manager         4.5.4     True        False         False      23m
   storage                                    4.5.4     True        False         False      17m
   ```

   c. 다음 명령어를 실행하여 클러스터 Pod를 확인합니다.

   ```
   $ oc get pods --all-namespaces
   ```

출력 예

```
NAMESPACE                                           NAME
READY   STATUS    RESTARTS   AGE
kube-system                                         etcd-member-ip-10-0-3-111.us-east-
2.compute.internal              1/1     Running     0          35m
kube-system                                         etcd-member-ip-10-0-3-239.us-east-
2.compute.internal              1/1     Running     0          37m
kube-system                                         etcd-member-ip-10-0-3-24.us-east-
2.compute.internal              1/1     Running     0          35m
openshift-apiserver-operator                        openshift-apiserver-operator-6d6674f4f4-
h7t2t                           1/1     Running     1          37m
openshift-apiserver                                 apiserver-fm48r
1/1     Running     0          30m
openshift-apiserver                                 apiserver-fxkvv
1/1     Running     0          29m
openshift-apiserver                                 apiserver-q85nm
1/1     Running     0          29m
...
openshift-service-ca-operator                       openshift-service-ca-operator-66ff6dc6cd-
9r257                           1/1     Running     0          37m
openshift-service-ca                                apiservice-cabundle-injector-695b6bcbc-cl5hm
1/1     Running     0          35m
openshift-service-ca                                configmap-cabundle-injector-8498544d7-
25qn6                           1/1     Running     0          35m
openshift-service-ca                                service-serving-cert-signer-6445fc9c6-wqdqn
1/1     Running     0          35m
openshift-service-catalog-apiserver-operator        openshift-service-catalog-apiserver-
operator-549f44668b-b5q2w       1/1     Running     0          32m
openshift-service-catalog-controller-manager-operator   openshift-service-catalog-
controller-manager-operator-b78cr2lnm   1/1     Running     0          31m
```

현재 클러스터 버전이 **AVAILABLE**이면 설치가 완료된 것입니다.

7.12.24. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

7.12.25. 다음 단계

- 클러스터를 사용자 지정합니다.
- Cluster Samples Operator 및 **must-gather** 툴의 이미지 스트림을 구성합니다.

- 제한된 네트워크에서 Operator Lifecycle Manager (OLM) 사용 방법에 대해 살펴봅니다.
- 클러스터를 설치하는 데 사용한 미러 레지스트리에 신뢰할 수 있는 CA가 있는 경우 추가 신뢰 저장소를 구성하여 클러스터에 추가합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.

7.13. GCP의 클러스터 설치 제거

GCP(Google Cloud Platform)에 배포한 클러스터를 삭제할 수 있습니다.

7.13.1. 설치 관리자가 프로비저닝한 인프라를 사용하는 클러스터 제거

클라우드에서 설치 관리자 프로비저닝 인프라를 사용하는 클러스터를 제거할 수 있습니다.

참고

설치 제거 후 특히 UPI(User Provisioned Infrastructure) 클러스터에서 제거되지 않은 리소스에 대해 클라우드 공급자를 확인합니다. 설치 관리자가 생성하지 않았거나 설치 프로그램이 액세스할 수 없는 리소스가 있을 수 있습니다. 예를 들어 일부 Google Cloud 리소스에는 공유 VPC 호스트 프로젝트에서 IAM 권한이 필요하거나 삭제해야 하는 사용되지 않은 상태 점검이 있을 수 있습니다.

사전 요구 사항

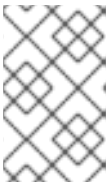
- 클러스터를 배포하는 데 사용한 설치 프로그램의 사본을 준비합니다.
- 클러스터를 만들 때 설치 프로그램에 의해 생성된 파일을 준비합니다.

프로세스

1. 클러스터를 설치하는 데 사용한 컴퓨터에 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행합니다.

```
$ ./openshift-install destroy cluster \
--dir <installation_directory> --log-level info 1 2
```

1. **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.
2. 다른 세부 사항을 보려면 **info** 대신 **warn**, **debug** 또는 **error**를 지정합니다.



참고

클러스터의 클러스터 정의 파일이 포함되어 있는 디렉터리를 지정해야 합니다. 설치 프로그램이 클러스터를 삭제하려면 이 디렉터리에 있는 **metadata.json** 파일이 필요합니다.

2. 선택사항: **<installation_directory>** 디렉터리와 OpenShift Container Platform 설치 프로그램을 삭제합니다.

8장. 베어 메탈에 설치

8.1. 베어 메탈 클러스터 설치 준비

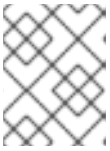
8.1.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비 에 대한 문서를 읽습니다.

8.1.2. OpenShift Virtualization을 위한 베어 메탈 클러스터 계획

OpenShift Virtualization을 사용하는 경우 베어 메탈 클러스터를 설치하기 전에 몇 가지 요구 사항을 알고 있어야 합니다.

- 실시간 마이그레이션 기능을 사용하려면 *클러스터 설치 시 작업자 노드가 여러 개* 있어야 합니다. 실시간 마이그레이션에는 클러스터 수준의 HA(고가용성) 플래그를 true로 설정해야 하기 때문입니다. HA 플래그는 클러스터가 설치될 때 설정되며 나중에 변경할 수 없습니다. 클러스터를 설치할 때 정의된 작업자 노드가 두 개 미만인 경우 클러스터 수명에 대해 HA 플래그가 false로 설정됩니다.



참고

단일 노드 클러스터에 OpenShift Virtualization을 설치할 수 있지만 단일 노드 OpenShift는 고가용성을 지원하지 않습니다.

- 실시간 마이그레이션에는 공유 스토리지가 필요합니다. OpenShift Virtualization의 스토리지에서는 RWX(ReadWriteMany) 액세스 모드를 지원하고 사용해야 합니다.
- SR-IOV(Single Root I/O Virtualization)를 사용하려는 경우 OpenShift Container Platform에서 NIC(네트워크 인터페이스 컨트롤러)를 지원하는지 확인합니다.

추가 리소스

- OpenShift Virtualization을 위한 클러스터 준비
- SR-IOV(Single Root I/O Virtualization) 하드웨어 네트워크 정보
- SR-IOV 네트워크에 가상 머신 연결

8.1.3. 베어 메탈에 OpenShift Container Platform을 설치할 방법 선택

설치 관리자 프로비저닝 또는 사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치할 수 있습니다. 기본 설치 유형은 설치 프로그램이 클러스터의 기본 인프라를 프로비저닝하는 설치 관리자 프로비저닝 인프라를 사용합니다. 제공하는 인프라에 OpenShift Container Platform도 설치할 수 있습니다. 설치 프로그램에서 프로비저닝한 인프라를 사용하지 않는 경우 클러스터 리소스를 직접 관리하고 유지보수해야 합니다.

설치 관리자 프로비저닝 및 사용자 프로비저닝 설치 프로세스에 대한 자세한 내용은 설치 프로세스를 참조하십시오.

8.1.3.1. 설치 관리자 프로비저닝 인프라를 사용하여 클러스터 설치

다음 방법을 사용하여 OpenShift Container Platform 설치 프로그램에서 프로비저닝한 베어 메칼 인프라에 클러스터를 설치할 수 있습니다.

- **베어 메탈에 설치 프로그램 프로비저닝 클러스터 설치**: 설치 관리자 프로비저닝을 사용하여 베어 메탈에 OpenShift Container Platform을 설치할 수 있습니다.

8.1.3.2. 사용자 프로비저닝 인프라를 사용하여 클러스터 설치

다음 방법 중 하나를 사용하여 프로비저닝하는 베어 메탈 인프라에 클러스터를 설치할 수 있습니다.

- **베어 메탈에 사용자 프로비저닝 클러스터 설치**: 프로비저닝하는 베어메탈 인프라에 OpenShift Container Platform을 설치할 수 있습니다. 사용자 프로비저닝 인프라가 포함된 클러스터의 경우, 필요한 모든 시스템을 배포해야 합니다.

- **네트워크 사용자 지정으로 사용자 프로비저닝 베어 메탈 클러스터 설치**: 네트워크 사용자 지정으로 사용자 프로비저닝 인프라에 베어 메탈 클러스터를 설치할 수 있습니다. 네트워크 구성을 사용자 지정할 경우, 클러스터가 사용자 환경의 기존 IP 주소 할당과 공존하고 기존 MTU 및 VXLAN 구성과 통합될 수 있습니다. 대부분의 네트워크 사용자 지정은 설치 단계에서 적용해야 합니다.

- **제한된 네트워크에 사용자 프로비저닝 베어 메탈 클러스터 설치**: 미러 레지스트리를 사용하여 제한된 또는 연결이 끊긴 네트워크에 사용자 프로비저닝 베어 메탈 클러스터를 설치할 수 있습니다. 이 설치 방법을 사용하여 클러스터가 외부 콘텐츠에 대해 조직의 제어 조건을 충족하는 컨테이너 이미지만 사용하도록 할 수 있습니다.

8.2. 베어 메탈에 사용자 프로비저닝 클러스터 설치

OpenShift Container Platform 4.9에서는 사용자가 프로비저닝하는 베어메탈 인프라에 클러스터를 설치할 수 있습니다.

중요

이 프로세스에 따라 가상화 또는 클라우드 환경에 클러스터를 배포할 수는 있지만 베어메탈 이외 플랫폼에 대한 추가적인 고려사항도 알고 있어야 합니다. 가상화 또는 클라우드 환경에서 OpenShift Container Platform 클러스터를 설치하기 전에 테스트되지 않은 플랫폼에 OpenShift Container Platform 배포 지침의 정보를 검토하십시오.

8.2.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비 에 대한 문서를 읽습니다.
- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성 해야 합니다.

참고

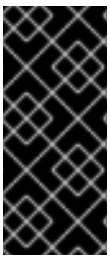
프록시를 구성하는 경우에도 해당 사이트 목록을 검토하십시오.

8.2.2. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.



중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

추가 리소스

- 프로비저닝하는 베어 메탈 인프라에서 제한된 네트워크 설치를 수행하는 방법에 대한 자세한 내용은 제한된 네트워크에 사용자 프로비저닝 베어 메탈 클러스터 설치 를 참조하십시오.

8.2.3. 사용자 프로비저닝 인프라를 포함한 클러스터의 시스템 요구사항

사용자 프로비저닝 인프라가 포함된 클러스터의 경우, 필요한 모든 시스템을 배포해야 합니다.

이 섹션에서는 사용자 프로비저닝 인프라에 OpenShift Container Platform을 배포해야 하는 요구 사항에 대해 설명합니다.

8.2.3.1. 클러스터 설치에 필요한 시스템

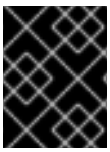
최소 OpenShift Container Platform 클러스터에 다음과 같은 호스트가 필요합니다.

표 8.1. 최소 필수 호스트

호스트	설명
임시 부트스트랩 시스템 한 개	컨트롤 플레인 시스템 세 개에 OpenShift Container Platform 클러스터를 배포하기 위한 부트스트랩 시스템이 클러스터에 필요합니다. 클러스터를 설치한 후 부트스트랩 시스템을 제거할 수 있습니다.
컨트롤 플레인 시스템 세 개	컨트롤 플레인 시스템은 컨트롤 플레인을 구성하는 Kubernetes 및 OpenShift Container Platform 서비스를 실행합니다.
두 개 이상의 컴퓨팅 시스템(작업자 시스템이라고도 함).	OpenShift Container Platform 사용자가 요청한 워크로드는 컴퓨팅 머신에서 실행됩니다.

참고

예외적으로 세 개의 컨트롤 플레인 시스템으로 구성된 베어 메탈 클러스터에 제로 컴퓨팅 머신을 실행할 수 있습니다. 이를 통해 클러스터 관리자와 개발자들이 테스트, 개발, 프로덕션에 사용할 수 있는 소형화되고 리소스 효율이 높은 클러스터를 제공합니다. 컴퓨팅 머신 하나를 실행하는 것은 지원되지 않습니다.



중요

클러스터의 고가용성을 유지하려면 이러한 클러스터 시스템에 대해 별도의 물리적 호스트를 사용하십시오.

부트스트랩, 컨트롤 플레인 시스템은 운영 체제로 RHCOS (Red Hat Enterprise Linux CoreOS)를 사용해야 합니다. 그러나 컴퓨팅 머신은 RHCOS(Red Hat Enterprise Linux CoreOS), RHEL (Red Hat Enterprise Linux) 7.9 또는 RHEL 8.4 중에서 선택할 수 있습니다.

RHCOS는 RHEL 8(Red Hat Enterprise Linux)을 기반으로하며 모든 하드웨어 인증 및 요구사항을 모두 이어받습니다. Red Hat Enterprise Linux 기술 기능 및 제한 을 참조하십시오.

8.2.3.2. 클러스터 설치를 위한 최소 리소스 요구 사항

각 클러스터 시스템이 다음과 같은 최소 요구사항을 충족해야 합니다.

표 8.2. 최소 리소스 요구사항

시스템	운영 체제	CPU [1]	RAM	스토리지	IOPS [2]
부트스트랩	RHCOS	4	16GB	100GB	300
컨트롤 플레인	RHCOS	4	16GB	100GB	300
컴퓨팅	RHCOS, RHEL 7.9 또는 RHEL 8.4 [3]	2	8GB	100GB	300

1. SMT(동시 멀티스레딩) 또는 하이퍼 스레딩이 활성화되지 않은 경우 하나의 CPU는 하나의 실제 코어에 해당합니다. 활성화하면 다음과 같은 공식을 사용하여 해당 비율을 계산합니다. (코어 당 스레드 수). (코어 당 스레드) 소켓s = CPU.

2. OpenShift Container Platform 및 Kubernetes는 디스크 성능에 민감하며 특히 10ms p99 fsync 기간이 필요한 컨트롤 플레인 노드의 etcd에 더 빠른 스토리지가 권장됩니다. 많은 클라우드 플랫폼에서 스토리지 크기와 IOPS를 함께 확장되므로 충분한 성능을 얻으려면 스토리지 볼륨을 과도하게 할당해야 할 수 있습니다.

3. 사용자가 프로비저닝한 모든 설치와 마찬가지로 클러스터에서 RHEL 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. RHEL 7 컴퓨팅 머신 사용은 더 이상 사용되지 않으며 향후 OpenShift Container Platform 4 릴리스에서 제거될 예정입니다.

8.2.3.3. 인증서 서명 요청 관리

사용자가 프로비저닝하는 인프라를 사용하는 경우 자동 시스템 관리 기능으로 인해 클러스터의 액세스가 제한되므로 설치한 후 클러스터 인증서 서명 요청(CSR)을 승인하는 메커니즘을 제공해야 합니다. **kube-**

controller-manager는 kubelet 클라이언트 CSR만 승인합니다. **machine-approver**는 올바른 시스템에서 발행한 요청인지 확인할 수 없기 때문에 kubelet 자격 증명을 사용하여 요청하는 서비스 인증서의 유효성을 보장할 수 없습니다. kubelet 서빙 인증서 요청의 유효성을 확인하고 요청을 승인하는 방법을 결정하여 구현해야 합니다.

추가 리소스

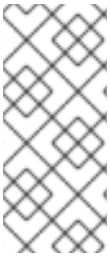
- 베어 메탈 환경에서 3-노드 클러스터를 배포하는 방법에 대한 자세한 내용은 3-노드 클러스터 구성을 참조하십시오.
- 설치 후 클러스터 인증서 서명 요청 승인에 대한 자세한 내용은 시스템의 인증서 서명 요청 승인을 참조하십시오.

8.2.3.4. 사용자 프로비저닝 인프라에 대한 네트워킹 요구사항

모든 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템이 부팅 중에 Ignition 구성 파일을 가져오려면 **initramfs**에 네트워킹을 구성해야 합니다.

초기 부팅 과정에서 시스템에 필요한 부팅 옵션을 제공하여 DHCP 서버를 통해 설정하거나 정적으로 설정하는 IP 주소 구성이 필요합니다. 네트워크 연결이 설정된 후 시스템은 HTTP 또는 HTTPS 서버에서 Ignition 구성 파일을 다운로드합니다. 그런 다음 Ignition 구성 파일을 사용하여 각 머신의 정확한 상태를 설정합니다. Machine Config Operator는 설치 후 새 인증서 또는 키 적용과 같은 머신에 대한 추가 변경을 완료합니다.

클러스터 시스템의 장기적인 관리를 위해 DHCP 서버를 사용하는 것이 좋습니다. DHCP 서버가 클러스터 시스템에 영구 IP 주소, DNS 서버 정보 및 호스트 이름을 제공하도록 구성되었는지 확인합니다.



참고

사용자 프로비저닝 인프라에 DHCP 서비스를 사용할 수 없는 경우 RHCOS 설치 시 노드에 IP 네트워킹 구성과 DNS 서버의 주소를 대신 제공할 수 있습니다. ISO 이미지에서 설치하는 경우 부팅 인수로 전달할 수 있습니다. 고정 IP 프로비저닝 *및 고급 네트워킹 옵션에 대한 자세한 내용은 RHCOS 설치 및 OpenShift Container Platform 부트스트랩 프로세스 시작* 섹션을 참조하십시오.

Kubernetes API 서버가 클러스터 시스템의 노드 이름을 확인할 수 있어야 합니다. API 서버와 작업자 노드가 서로 다른 영역에 있는 경우, API 서버가 노드 이름을 확인할 수 있도록 기본 DNS 검색 영역을 설정할 수 있습니다. 노드 개체와 모든 DNS 요청에서 항상 정규화된 도메인 이름으로 호스트를 가리키는 것도 지원되는 방법입니다

8.2.3.4.1. DHCP를 통해 클러스터 노드의 호스트 이름 설정

RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 호스트 이름은 NetworkManager를 통해 설정됩니다. 기본적으로 시스템은 DHCP를 통해 호스트 이름을 가져옵니다. DHCP에서 호스트 이름을 제공하지 않거나, 커널 인수를 통해 정적으로 설정하거나 다른 방법을 통해 설정하면 역방향 DNS 조회를 통해 얻을 수 있습니다. 역방향 DNS 조회는 노드에서 네트워크를 초기화한 후 수행되며 확인하는 데 시간이 걸릴 수 있습니다. 다른 시스템 서비스는 이 보다 먼저 시작하여 호스트 이름을 **localhost** 등으로 감지할 수 있습니다. DHCP를 사용하여 각 클러스터 노드의 호스트 이름을 제공하여 이 문제를 방지할 수 있습니다.

또한 DHCP를 통해 호스트 이름을 설정하면 DNS 분할 수평 구현 환경에서 수동으로 DNS 레코드 이름 구성 오류를 무시할 수 있습니다.

8.2.3.4.2. 네트워크 연결 요구사항

OpenShift Container Platform 클러스터 구성 요소가 통신할 수 있도록 시스템 간 네트워크 연결을 구성해야 합니다. 각 시스템에서 클러스터에 있는 다른 모든 시스템의 호스트 이름을 확인할 수 있어야 합니다.

이 섹션에서는 필요한 포트에 대해 자세히 설명합니다.

중요

연결된 OpenShift Container Platform 환경에서 모든 노드는 플랫폼 컨테이너의 이미지를 가져오고 Red Hat에 원격 측정 데이터를 제공하기 위해 인터넷에 액세스할 수 있어야 합니다.

표 8.3. 모든 시스템 간 통신에 사용되는 포트

프로토콜	포트	설명
ICMP	해당 없음	네트워크 연결성 테스트
TCP	**1936**	메트릭
	9000-9999	**9100-9101** 포트의 노드 내보내기 및 **9099** 포트의 Cluster Version Operator를 포함한 호스트 수준 서비스.
	10250-10259	Kubernetes에서 예약하는 기본 포트
	10256	openshift-sdn
UDP	**4789**	VXLAN 및 Geneve
	6081	VXLAN 및 Geneve
	9000-9999	**9100-9101** 포트의 노드 내보내기를 포함한 호스트 수준 서비스.
	500	IPsec IKE 패킷
	4500	IPsec NAT-T 패킷
TCP/UDP	**30000-32767**	Kubernetes 노드 포트
ESP	해당 없음	IPsec Encapsulating Security Payload (ESP)

표 8.4. 모든 시스템과 컨트롤 플레인 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**6443**	Kubernetes API

표 8.5. 컨트롤 플레인 머신 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**2379-2380**	etcd 서버 및 피어 포트

사용자 프로비저닝 인프라에 대한 NTP 구성
OpenShift Container Platform 클러스터는 기본적으로 공용 NTP(Network Time Protocol) 서버를 사용하도록 구성되어 있습니다. 로컬 엔터프라이즈 NTP 서버를 사용하거나 클러스터가 연결이 끊긴 네트워크에 배포되는 경우 특정 시간 서버를 사용하도록 클러스터를 구성할 수 있습니다. 자세한 내용은 *chrony 타임 서비스 설정* 문서를 참조하십시오.

DHCP 서버가 NTP 서버 정보를 제공하는 경우 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템의 chrony 타임 서비스에서 정보를 읽고 NTP 서버와 클럭을 동기화할 수 있습니다.

추가 리소스

- chrony 타임 서비스 설정

8.2.3.5. 사용자 프로비저닝 DNS 요구사항

OpenShift Container Platform 배포의 경우 다음 구성 요소에 DNS 이름을 확인해야 합니다.

- Kubernetes API
- OpenShift Container Platform 애플리케이션 와일드카드
- 부트스트랩, 컨트롤 플레인 및 컴퓨팅 시스템

Kubernetes API, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템에 대한 역방향 DNS 확인이 필요합니다.

DNS A/AAAA 또는 CNAME 레코드는 이름 확인에 사용되며 PTR 레코드는 역방향 이름 확인에 사용됩니다. RHCOS (Red Hat Enterprise Linux CoreOS)는 DHCP에서 호스트 이름을 제공하지 않는 한 모든 노드의 호스트 이름을 설정할 때 역방향 레코드를 사용하기 때문에 역방향 레코드가 중요합니다. 또한 역방향 레코드는 OpenShift Container Platform이 작동하는 데 필요한 인증서 서명 요청 (CSR)을 생성하는 데 사용됩니다.

참고

DHCP 서버를 사용하여 각 클러스터 노드에 호스트 이름을 제공하는 것이 좋습니다. 자세한 내용은 *사용자 프로비저닝 인프라 섹션에 대한 DHCP 권장 사항* 섹션을 참조하십시오.

사용자가 프로비저닝한 OpenShift Container Platform 클러스터에 대해 다음 DNS 레코드가 필요하며 설치 전에 있어야 합니다. 각 레코드에서 **<cluster_name>**은 클러스터 이름이고 **<base_domain>**은 **install-config.yaml** 파일에서 지정하는 기반 도메인입니다. 전체 DNS 레코드는 **<component>.<cluster_name>.<base_domain>** 형식입니다.

표 8.6. 필수 DNS 레코드

구성 요소	레코드	설명
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API 로드 밸런서를 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.
	api-int.<cluster_name>.<base_domain>.	내부적으로 API 로드 밸런서를 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다. **중요** API 서버는 Kubernetes에 기록된 호스트 이름으로 작업자 노드를 확인할 수 있어야 합니다. API 서버가 노드 이름을 확인할 수 없는 경우 프록시된 API 호출이 실패할 수 있으며 pod에서 로그를 검색할 수 없습니다.
라우트	***.apps.<cluster_name>.<base_domain>.**	애플리케이션 인그레스 로드 밸런서를 참조하는 와일드카드 DNS A/AAA 또는 CNAME 레코드입니다. 애플리케이션 인그레스 로드 밸런서는 Ingress 컨트롤러 Pod를 실행하는 머신을 대상으로 합니다. Ingress 컨트롤러 Pod는 기본적으로 컴퓨팅 머신에서 실행됩니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다. 예를 들어 **console-openshift-console.apps.<cluster_name>.<base_domain>**은 OpenShift Container Platform 콘솔의 와일드카드 경로로 사용됩니다.
부트스트랩 시스템	**bootstrap.<cluster_name>.<base_domain>.**	부트스트랩 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.
컨트롤 플레인 머신	**<master><n>.<cluster_name>.<base_domain>.**	컨트롤 플레인 노드의 각 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.
컴퓨팅 머신	**<worker><n>.<cluster_name>.<base_domain>.**	작업자 노드의 각 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.

참고

OpenShift Container Platform 4.4 이상에서는 DNS 구성에서 etcd 호스트 및 SRV 레코드를 지정할 필요가 없습니다.

작은 정보

dig 명령을 사용하여 이름과 역방향 이름을 확인할 수 있습니다. 자세한 검증 단계는 *사용자 프로비저닝 인프라의 DNS 확인* 섹션을 참조하십시오.

8.2.3.5.1. 사용자 프로비저닝 클러스터의 DNS 구성 예

이 섹션에서는 사용자 프로비저닝 인프라에 OpenShift Container Platform을 배포하기 위한 DNS 요구 사항을 충족하는 A 및 PTR 레코드 구성 샘플을 제공합니다. 샘플은 하나의 DNS 솔루션을 선택하기 위한 조언을 제공하기 위한 것이 아닙니다.

이 예제에서 클러스터 이름은 **ocp4**이고 기본 도메인은 **example.com**입니다.

사용자 프로비저닝 클러스터의 DNS A 레코드 구성 예

다음 BIND 영역 파일의 예제에서는 사용자가 프로비저닝한 클러스터의 이름 확인을 위한 샘플 A 레코드를 보여줍니다.

예 8.1. 샘플 DNS 영역 데이터베이스

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
master0.ocp4.example.com. IN A 192.168.1.97 5
master1.ocp4.example.com. IN A 192.168.1.98 6
master2.ocp4.example.com. IN A 192.168.1.99 7
;
worker0.ocp4.example.com. IN A 192.168.1.11 8
worker1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1 Kubernetes API의 이름 확인을 제공합니다. 레코드는 API 로드 밸런서의 IP 주소를 나타냅니다.

2 Kubernetes API의 이름 확인을 제공합니다. 레코드는 API 로드 밸런서의 IP 주소를 참조하며 내부 클러스터 통신에 사용됩니다.

3 와일드카드 경로의 이름 확인을 제공합니다. 레코드는 애플리케이션 인그레스 로드 밸런서의 IP 주소를 나타냅니다. 애플리케이션 인그레스 로드 밸런서는 Ingress 컨트롤러 Pod를 실행하는 머신을 대상으로 합니다. Ingress 컨트롤러 Pod는 기본적으로 컴퓨팅 머신에서 실행됩니다.

참고

이 예제에서는 Kubernetes API 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 API 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

4 부트스트랩 시스템의 이름 확인을 제공합니다.

5 6 7 컨트롤 플레인 시스템의 이름 확인을 제공합니다.

8 9 컴퓨팅 시스템의 이름 확인을 제공합니다.

사용자 프로비저닝 클러스터의 DNS PTR 레코드 구성 예

다음 예제 BIND 영역 파일은 사용자 프로비저닝 클러스터의 역방향 이름 확인을 위한 샘플 PTR 레코드를 보여줍니다.

예 8.2. 역방향 레코드의 샘플 DNS 영역 데이터베이스

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR master0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR master1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR master2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com. 8
;
;EOF
```

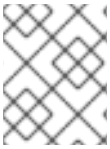
1 Kubernetes API의 역방향 DNS 확인을 제공합니다. PTR 레코드는 API 로드 밸런서의 레코드 이름을 참조합니다.

2 Kubernetes API의 역방향 DNS 확인을 제공합니다. PTR 레코드는 API 로드 밸런서의 레코드 이름을 참조하며 내부 클러스터 통신에 사용됩니다.

3 부트스트랩 시스템의 역방향 DNS 확인을 제공합니다.

4 5 6 컨트롤 플레인 시스템의 역방향 DNS 확인을 제공합니다.

7 8 컴퓨팅 시스템의 역방향 DNS 확인을 제공합니다.



참고

OpenShift Container Platform 애플리케이션 와일드카드에는 PTR 레코드가 필요하지 않습니다.

추가 리소스

- 사용자 프로비저닝 인프라에 대한 DNS 확인 검증

8.2.3.6. 사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구사항

OpenShift Container Platform을 설치하기 전에 API 및 애플리케이션 인그레스 로드 밸런싱 인프라를 프로비저닝해야 합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 API 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

참고

RHEL(Red Hat Enterprise Linux) 인스턴스를 사용하여 API 및 애플리케이션 인그레스 로드 밸런서를 배포하려면 RHEL 서브스크립션을 별도로 구매해야 합니다.

로드 밸런서 인프라는 다음 요구 사항을 충족해야 합니다.

1. **API 로드 밸런서**: 플랫폼과 상호 작용하고 플랫폼을 구성할 수 있도록 사용자(인간과 시스템 모두)에게 공통 끝점을 제공합니다. 다음 조건을 설정합니다.

   - Layer 4 로드 밸런싱 전용입니다. 이를 Raw TCP, SSL Passthrough 또는 SSL Bridge 모드라고 합니다. SSL Bridge 모드를 사용하는 경우, API 경로에 대해 SNI(Server Name Indication, 서버 이름 표시)를 활성화해야 합니다.

   - 스테이트리스 로드 밸런싱 알고리즘입니다. 옵션은 로드 밸런서 구현에 따라 달라집니다.

   참고

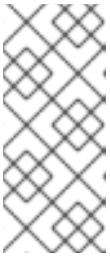
   API 로드 밸런서가 제대로 작동하기 위해 세션 지속성이 필요하지 않습니다.

   로드 밸런서의 전면과 후면 모두에서 다음 포트를 구성하십시오.

표 8.7. API 로드 밸런서

포트	백엔드 시스템(풀 멤버)	내부	외부	설명

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
6443	부트스트랩 및 컨트롤 플레인. 부트스트랩 시스템이 클러스터 컨트롤 플레인을 초기화한 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다. API 서버 상태 검사 프로브에 대한 **/readyz** 끝점을 구성해야 합니다.	X	X	Kubernetes API 서버
22623	부트스트랩 및 컨트롤 플레인. 부트스트랩 시스템이 클러스터 컨트롤 플레인을 초기화한 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다.	X		시스템 구성 서버



참고

API 서버가 **/readyz** 엔드포인트를 해제하는 시점부터 풀에서 API 서버 인스턴스가 제거되는 시점까지 시간이 30초를 넘지 않도록 로드 밸런서를 구성해야 합니다. **/readyz**가 오류를 반환하거나 정상 상태가 된 후 정해진 시간 안에 끝점이 제거 또는 추가되어야 합니다. 5초 또는 10초의 프로빙 주기(두 번의 성공적인 요청은 정상 상태, 세 번의 요청은 비정상 상태)는 충분한 테스트를 거친 값입니다.

2. **애플리케이션 인그레스 로드 밸런서**: 클러스터 외부에서 유입되는 애플리케이션 트래픽에 대한 인그레스 포인트를 제공합니다. 다음 조건을 설정합니다.

   - Layer 4 로드 밸런싱 전용입니다. 이를 Raw TCP, SSL Passthrough 또는 SSL Bridge 모드라고 합니다. SSL Bridge 모드를 사용하는 경우 인그레스 경로에 대해 SNI(Server Name Indication, 서버 이름 표시)를 활성화해야 합니다.
   - 사용 가능한 옵션과 플랫폼에서 호스팅되는 애플리케이션 유형에 따라 연결 기반 또는 세션 기반 지속성이 권장됩니다.

작은 정보

애플리케이션 인그레스 로드 밸런서에서 클라이언트의 실제 IP 주소를 확인할 수 있는 경우 소스 IP 기반 세션 지속성을 활성화하면 엔드 투 엔드 TLS 암호화를 사용하는 애플리케이션의 성능을 향상시킬 수 있습니다.

로드 밸런서의 전면과 후면 모두에서 다음 포트를 구성하십시오.

표 8.8. 애플리케이션 인그레스 로드 밸런서

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
443	기본적으로 인그레스 컨트롤러 pod, 컴퓨팅 또는 작업자를 실행하는 시스템입니다.	X	X	HTTPS 트래픽

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
80	기본적으로 인그레스 컨트롤러 pod, 컴퓨팅 또는 작업자를 실행하는 시스템입니다.	X	X	HTTP 트래픽
1936	기본적으로 Ingress 컨트롤러 Pod를 실행하는 작업자 노드입니다. 수신 상태 점검 프로브에 대해 **/healthz/ready** 끝점을 구성해야 합니다.	X	X	HTTP 트래픽

참고

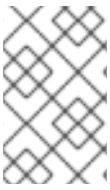
컴퓨팅 노드가 0인 3-노드 클러스터를 배포하는 경우 Ingress 컨트롤러 Pod는 컨트롤 플레인 노드에서 실행됩니다. 3-노드 클러스터 배포에서 HTTP 및 HTTPS 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다.

참고

인그레스 라우터에 대한 작업 구성이 OpenShift Container Platform 클러스터에 필요합니다. 컨트롤 플레인 초기화 후 인그레스 라우터를 설정해야 합니다.

8.2.3.6.1. 사용자 프로비저닝 클러스터의 로드 밸런서 구성 예

이 섹션에서는 사용자 프로비저닝 클러스터의 로드 밸런싱 요구 사항을 충족하는 API 및 애플리케이션 수신 로드 밸런서 구성 예를 제공합니다. 샘플은 HAProxy 로드 밸런서에 대한 **/etc/haproxy/haproxy.cfg** 구성입니다. 이 예제에서는 하나의 로드 밸런싱 솔루션을 선택하기 위한 조언을 제공하는 것을 목적으로 하지 않습니다.



참고

이 예제에서는 Kubernetes API 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 API 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

예 8.3. API 및 애플리케이션 인그레스 로드 밸런서 구성 샘플

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
```

```
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s
    maxconn                 3000
  frontend stats
    bind *:1936
    mode            http
    log             global
    maxconn 10
    stats enable
    stats hide-version
    stats refresh 30s
    stats show-node
    stats show-desc Stats for ocp4 cluster (1)
    stats auth admin:ocp4
    stats uri /stats
  listen api-server-6443 (2)
    bind *:6443
    mode tcp
    server bootstrap bootstrap.ocp4.example.com:6443 check inter 1s backup (3)
    server master0 master0.ocp4.example.com:6443 check inter 1s
    server master1 master1.ocp4.example.com:6443 check inter 1s
    server master2 master2.ocp4.example.com:6443 check inter 1s
  listen machine-config-server-22623 (4)
    bind *:22623
    mode tcp
    server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup (5)
    server master0 master0.ocp4.example.com:22623 check inter 1s
    server master1 master1.ocp4.example.com:22623 check inter 1s
    server master2 master2.ocp4.example.com:22623 check inter 1s
  listen ingress-router-443 (6)
    bind *:443
    mode tcp
    balance source
    server worker0 worker0.ocp4.example.com:443 check inter 1s
    server worker1 worker1.ocp4.example.com:443 check inter 1s
  listen ingress-router-80 (7)
    bind *:80
    mode tcp
    balance source
    server worker0 worker0.ocp4.example.com:80 check inter 1s
    server worker1 worker1.ocp4.example.com:80 check inter 1s
```

(1) 이 예에서 클러스터 이름은 **ocp4** 입니다.

(2) 포트 **6443**은 Kubernetes API 트래픽을 처리하고 컨트롤 플레인 시스템을 가리킵니다.

(3) (5) 부트스트랩 항목은 OpenShift Container Platform 클러스터 설치 전에 있어야 하며 부트스트랩 프로세스가 완료된 후 제거해야 합니다.

(4) 포트 **22623**은 머신 구성 서버 트래픽을 처리하고 컨트롤 플레인 시스템을 가리킵니다.

6 포트 **443**은 HTTPS 트래픽을 처리하고 Ingress 컨트롤러 Pod를 실행하는 시스템을 가리킵니다. Ingress 컨트롤러 Pod는 기본적으로 컴퓨팅 머신에서 실행됩니다.

7 포트 **80**은 HTTP 트래픽을 처리하고 Ingress 컨트롤러 Pod를 실행하는 머신을 가리킵니다. Ingress 컨트롤러 Pod는 기본적으로 컴퓨팅 머신에서 실행됩니다.

참고

컴퓨팅 노드가 0인 3-노드 클러스터를 배포하는 경우 Ingress 컨트롤러 Pod는 컨트롤 플레인 노드에서 실행됩니다. 3-노드 클러스터 배포에서 HTTP 및 HTTPS 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다.

작은 정보

HAProxy를 로드 밸런서로 사용하는 경우 HAProxy 노드에서 **netstat -nltupe**를 실행하여 **haproxy** 프로세스가 포트 **6443**, **22623**, **443** 및 **80**에서 수신 대기 중인지 확인할 수 있습니다.

참고

HAProxy를 로드 밸런서로 사용하고 SELinux가 **enforcing**으로 설정된 경우 HAProxy 서비스가 **setsebool -P haproxy_connect_any=1**을 실행하여 구성된 TCP 포트에 바인딩할 수 있는지 확인해야 합니다.

8.2.4. 사용자 프로비저닝 인프라 준비

사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치하기 전에 기본 인프라를 준비해야 합니다.

이 섹션에서는 OpenShift Container Platform 설치를 준비하기 위해 클러스터 인프라를 설정하는 데 필요한 높은 수준의 단계에 대해 자세히 설명합니다. 여기에는 클러스터 노드에 대한 IP 네트워킹 및 네트워크 연결 구성, 방화벽을 통해 필요한 포트 활성화, 필수 DNS 및 로드 밸런싱 인프라를 설정하는 작업이 포함됩니다.

준비 후 클러스터 인프라는 *사용자 프로비저닝 인프라가 있는 클러스터의 요구 사항* 섹션에 설명된 요구 사항을 충족해야 합니다.

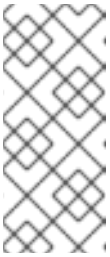
사전 요구 사항

- OpenShift Container Platform 4.x 테스트된 통합 페이지를 검토했습니다.
- *사용자 프로비저닝 인프라가 있는 클러스터의 요구 사항* 섹션에 자세히 설명된 인프라 요구 사항을 검토했습니다.

프로세스

1. DHCP를 사용하여 클러스터 노드에 IP 네트워킹 구성을 제공하는 경우 DHCP 서비스를 구성합니다.
   a. 노드의 영구 IP 주소를 DHCP 서버 구성에 추가합니다. 구성에서 관련 네트워크 인터페이스의 MAC 주소를 각 노드의 의도한 IP 주소와 일치시킵니다.

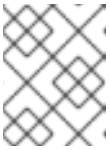
b. DHCP를 사용하여 클러스터 시스템의 IP 주소 지정을 구성하는 경우 시스템은 DHCP를 통해 DNS 서버 정보도 가져옵니다. DHCP 서버 구성을 통해 클러스터 노드에서 사용하는 영구 DNS 서버 주소를 정의합니다.



참고

DHCP 서비스를 사용하지 않는 경우 RHCOS 설치 시 IP 네트워킹 구성과 DNS 서버의 주소를 노드에 제공해야 합니다. ISO 이미지에서 설치하는 경우 부팅 인수로 전달할 수 있습니다. 고정 IP 프로비저닝 *및 고급 네트워킹 옵션에 대한 자세한 내용은 RHCOS 설치 및 OpenShift Container Platform 부트스트랩 프로세스 시작* 섹션을 참조하십시오.

c. DHCP 서버 구성에 클러스터 노드의 호스트 이름을 정의합니다. 호스트 이름 고려 사항에 대한 자세한 내용은 *DHCP를 통해 클러스터 노드 호스트 이름 설정* 섹션을 참조하십시오.



참고

DHCP 서비스를 사용하지 않는 경우 클러스터 노드는 역방향 DNS 조회를 통해 호스트 이름을 가져옵니다.

2. 네트워크 인프라가 클러스터 구성 요소 간 필수 네트워크 연결을 제공하는지 확인합니다. 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 인프라 섹션의 네트워킹 요구 사항* 섹션을 참조하십시오.

3. OpenShift Container Platform 클러스터 구성 요소가 통신하는 데 필요한 포트를 활성화하도록 방화벽을 구성합니다. 필요한 포트에 대한 자세한 내용은 *사용자 프로비저닝 인프라 섹션의 네트워킹 요구 사항* 섹션을 참조하십시오.

4. 클러스터에 필요한 DNS 인프라를 설정합니다.

   a. Kubernetes API, 애플리케이션 와일드카드, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템의 DNS 이름 확인을 구성합니다.

   b. Kubernetes API, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템에 대한 역방향 DNS 확인을 구성합니다.
   OpenShift Container Platform DNS 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 DNS 요구 사항* 섹션을 참조하십시오.

5. DNS 구성을 확인합니다.

   a. 설치 노드에서 Kubernetes API의 레코드 이름, 와일드카드 경로 및 클러스터 노드에 대해 DNS 조회를 실행합니다. 응답의 IP 주소가 올바른 구성 요소에 해당하는지 확인합니다.

   b. 설치 노드에서 로드 밸런서 및 클러스터 노드의 IP 주소에 대해 역방향 DNS 조회를 실행합니다. 응답의 레코드 이름이 올바른 구성 요소에 해당하는지 확인합니다.
   자세한 DNS 검증 단계는 *사용자 프로비저닝 인프라에 대한 DNS 확인* 섹션을 참조하십시오.

6. 필요한 API 및 애플리케이션 수신 로드 밸런싱 인프라를 프로비저닝합니다. 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구 사항* 섹션을 참조하십시오.

참고

일부 로드 밸런싱 솔루션에는 로드 밸런싱을 초기화하기 전에 클러스터 노드의 DNS 이름을 확인해야 합니다.

추가 리소스

- 사용자 프로비저닝 인프라를 포함한 클러스터의 시스템 요구사항
- RHCOS 설치 및 OpenShift Container Platform 부트스트랩 프로세스 시작
- DHCP를 통해 클러스터 노드의 호스트 이름 설정
- 고급 RHCOS 설치 구성 옵션
- 사용자 프로비저닝 인프라에 대한 네트워킹 요구사항
- 사용자 프로비저닝 DNS 요구사항
- 사용자 프로비저닝 인프라에 대한 DNS 확인 검증
- 사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구사항

8.2.5. 사용자 프로비저닝 인프라에 대한 DNS 확인 검증

사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치하기 전에 DNS 구성을 확인할 수 있습니다.

중요

클러스터를 설치하기 전에 이 섹션에 설명된 검증 단계를 성공해야 합니다.

사전 요구 사항

- 사용자 프로비저닝 인프라에 필요한 DNS 레코드를 구성했습니다.

프로세스

1. 설치 노드에서 Kubernetes API의 레코드 이름, 와일드카드 경로 및 클러스터 노드에 대해 DNS 조회를 실행합니다. 응답에 포함된 IP 주소가 올바른 구성 요소에 해당하는지 확인합니다.

   a. Kubernetes API 레코드 이름을 조회합니다. 결과가 API 로드 밸런서의 IP 주소를 가리키는지 확인합니다.

   ```
   $ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
   ```

   1 **<nameserver_ip>**를 네임서버의 IP 주소로, **<cluster_name>**을 클러스터 이름으로, **<base_domain>**을 기본 도메인 이름으로 바꿉니다.

   출력 예

   ```
   api.ocp4.example.com.  0 IN A 192.168.1.5
   ```

   b. Kubernetes 내부 API 레코드 이름을 조회합니다. 결과가 API 로드 밸런서의 IP 주소를 가리키는지 확인합니다.

   ```
   $ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
   ```

출력 예

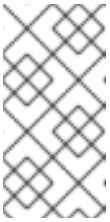
```
api-int.ocp4.example.com.  0 IN A 192.168.1.5
```

c. 예제 ***.apps.<cluster_name>.<base_domain>**을 테스트합니다. DNS 와일드카드를 조회합니다. 모든 애플리케이션 와일드카드 조회는 애플리케이션 인그레스 로드 밸런서의 IP 주소로 확인되어야 합니다.

```
$ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.<base_domain>
```

출력 예

```
random.apps.ocp4.example.com.  0 IN A 192.168.1.5
```



참고

예제 출력에서는 Kubernetes API 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 API 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

random 항목을 다른 와일드카드 값으로 교체할 수 있습니다. 예를 들어 OpenShift Container Platform 콘솔의 경로를 쿼리할 수 있습니다.

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

출력 예

```
console-openshift-console.apps.ocp4.example.com. 0 IN A 192.168.1.5
```

d. 부트스트랩 DNS 레코드 이름에 대해 조회를 실행합니다. 결과가 부트스트랩 노드의 IP 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.<base_domain>
```

출력 예

```
bootstrap.ocp4.example.com.  0 IN A 192.168.1.96
```

e. 이 방법을 사용하여 컨트롤 플레인 및 컴퓨팅 노드의 DNS 레코드 이름에 대해 조회를 수행합니다. 결과가 각 노드의 IP 주소에 해당하는지 확인합니다.

2. 설치 노드에서 로드 밸런서 및 클러스터 노드의 IP 주소에 대해 역방향 DNS 조회를 실행합니다. 응답에 포함된 레코드 이름이 올바른 구성 요소에 해당하는지 확인합니다.

a. API 로드 밸런서의 IP 주소에 대해 역방향 조회를 수행합니다. 응답에 Kubernetes API 및 Kubernetes 내부 API의 레코드 이름이 포함되어 있는지 확인합니다.

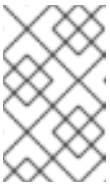
```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

출력 예

```
5.1.168.192.in-addr.arpa. 0 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 0 IN PTR api.ocp4.example.com. 2
```

1 Kubernetes 내부 API의 레코드 이름을 제공합니다.

2 Kubernetes API의 레코드 이름을 제공합니다.



참고

OpenShift Container Platform 애플리케이션 와일드카드에는 PTR 레코드가 필요하지 않습니다. 애플리케이션 인그레스 로드 밸런서의 IP 주소에 대한 역방향 DNS 확인에는 유효성 검사 단계가 필요하지 않습니다.

b. 부트스트랩 노드의 IP 주소에 대해 역방향 조회를 수행합니다. 결과가 부트스트랩 노드의 DNS 레코드 이름을 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

출력 예

```
96.1.168.192.in-addr.arpa. 0 IN PTR bootstrap.ocp4.example.com.
```

c. 이 방법을 사용하여 컨트롤 플레인 및 컴퓨팅 노드의 IP 주소에 대해 역방향 조회를 수행합니다. 결과가 각 노드의 DNS 레코드 이름과 일치하는지 확인합니다.

추가 리소스

- 사용자 프로비저닝 DNS 요구사항
- 사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구사항

8.2.6. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 **core** 사용자의 **~/.ssh/authorized_keys** 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 **core**로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다. SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. **./openshift-install gather** 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

참고

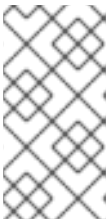
AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: **~/.ssh/id_ed25519**)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 **'~/.ssh** 디렉터리에 있는지 확인하십시오.



참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 **ed25519** 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 **rsa** 또는 **ecdsa** 알고리즘을 사용하는 키를 생성합니다.

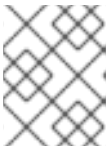
2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 **~/.ssh/id_ed25519.pub** 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 **./openshift-install gather** 명령을 사용하려는 경우 필요합니다.



참고

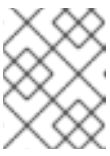
일부 배포에서는 **~/.ssh/id_rsa** 및 **~/.ssh/id_dsa**와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. **ssh-agent** 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```



참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. **ssh-agent**에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: **~/.ssh/id_ed25519**).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다. 프로비저닝하는 인프라에 클러스터를 설치하는 경우 설치 프로그램에 키를 제공해야 합니다.

추가 리소스

- 노드 상태 확인

8.2.7. 설치 프로그램 받기

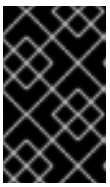
OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

프로세스

1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.
2. 인프라 공급자를 선택합니다.
3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.



중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.

중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 OpenShift Container Platform 설치 제거 절차를 완료해야 합니다.

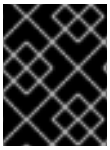
4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

8.2.8. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift CLI(**oc**)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 **oc**를 설치할 수 있습니다.



중요

이전 버전의 **oc**를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 **oc**를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 Linux에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Linux Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. **oc** 바이너리를 **PATH**에 있는 디렉터리에 배치합니다.
PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치

다음 절차에 따라 Windows에 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 Windows Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. ZIP 프로그램으로 아카이브의 압축을 풉니다.

5. **oc** 바이너리를 **PATH**에 있는 디렉터리로 이동합니다.
PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치
다음 절차에 따라 macOS에서 OpenShift CLI(**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.9 MacOSX Client** 항목 옆에 있는 **지금 다운로드를** 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 해제하고 압축을 풉니다.
5. **oc** 바이너리 PATH의 디렉터리로 이동합니다.
PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

8.2.9. 수동으로 설치 구성 파일 만들기

OpenShift Container Platform을 사용자가 프로비저닝한 설치의 경우 설치 구성 파일을 수동으로 생성합니다.

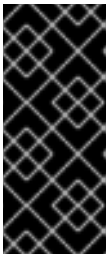
사전 요구 사항

- 로컬 시스템에 설치 프로그램에 제공할 SSH 공개 키가 있습니다. 키는 디버깅 및 재해 복구를 위해 클러스터 노드에 대한 SSH 인증에 사용됩니다.
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿이 있습니다.

프로세스

1. 필요한 설치 자산을 저장할 설치 디렉터리를 만듭니다.

```
$ mkdir <installation_directory>
```



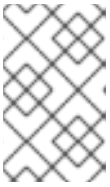
중요

디렉터리를 만들어야 합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

2. 샘플 **install-config.yaml** 파일 템플릿을 사용자 지정하여 **<installation_directory>**에 저장합니다.

참고

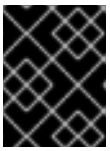
이 설정 파일의 이름을 **install-config.yaml**로 지정해야 합니다.



참고

일부 플랫폼 유형의 경우 대체로 **./openshift-install create install-config --dir <installation_directory>** 를 실행하여 **install-config.yaml** 파일을 생성할 수 있습니다. 프롬프트에서 클러스터 구성에 대한 세부 정보를 제공할 수 있습니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.



중요

install-config.yaml 파일은 설치 과정의 다음 단계에서 사용됩니다. 이 시점에서 이를 백업해야 합니다.

8.2.9.1. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 환경에 대한 세부 정보를 설명하는 사용자 지정 **install-config.yaml** 설치 구성 파일을 제공합니다.

참고

설치한 후에는 **install-config.yaml** 파일에서 이러한 매개변수를 수정할 수 없습니다.

중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

8.2.9.1.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 8.9. 필수 매개 변수

매개변수	설명	값

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	**dev** 와 같은 소문자 및 하이픈(**-**)의 문자열입니다.
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

매개변수	설명	값

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

8.2.9.1.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 8.10. 네트워크 매개변수

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체 **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.

매개변수	설명	값
networking.clusterNetwork	Pod의 IP 주소 블록입니다. 기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된 **cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16** **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

8.2.9.1.3. 선택적 구성 매개변수

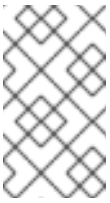
선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 8.11. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은**3**입니다.

매개변수	설명	값
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 기본값은 **External**입니다. 이 필드를 **Internal** 로 설정하는 것은 클라우드 이외의 플랫폼에서는 지원되지 않습니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

8.2.9.2. 베어 메탈의 샘플 install-config.yaml 파일

install-config.yaml 파일을 사용자 지정하여 OpenShift Container Platform 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.

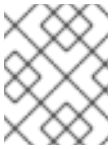
```
apiVersion: v1
baseDomain: example.com 1
compute: 2
- hyperthreading: Enabled 3
  name: worker
  replicas: 0 4
controlPlane: 5
  hyperthreading: Enabled 6
  name: master
  replicas: 3 7
metadata:
  name: test 8
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 9
    hostPrefix: 23 10
  networkType: OpenShiftSDN
  serviceNetwork: 11
  - 172.30.0.0/16
platform:
  none: {} 12
fips: false 13
pullSecret: '{"auths": ...}' 14
sshKey: 'ssh-ed25519 AAAA...' 15
```

1 클러스터의 기본 도메인입니다. 모든 DNS 레코드는 이 기본 도메인의 하위 도메인이어야 하며 클러스터 이름을 포함해야 합니다.

2 5 **controlPlane** 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 하나의 컨트롤 플레인 풀만 사용

됩니다.

3 6 동시 멀티스레딩(SMT) 또는 hyperthreading 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 SMT가 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. SMT를 비활성화하는 경우 모든 클러스터 머신에서 이를 비활성화해야 합니다. 여기에는 컨트롤 플레인과 컴퓨팅 머신이 모두 포함됩니다.



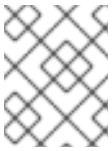
참고

SMT(동시 멀티 스레딩)는 기본적으로 활성화되어 있습니다. BIOS 설정에서 SMT를 활성화하지 않으면 **hyperthreading** 매개변수가 적용되지 않습니다.

중요

BIOS에서든 **install-config.yaml** 파일에서든 **hyperthreading**을 비활성화한 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.

4 사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치할 때 이 값을 **0**으로 설정해야 합니다. 설치 프로그램에서 제공하는 설치에서 매개 변수는 클러스터가 생성 및 관리하는 컴퓨팅 머신 수를 제어합니다. 사용자 프로비저닝 설치에서는 클러스터 설치를 완료하기 전에 컴퓨팅 시스템을 수동으로 배포해야 합니다.



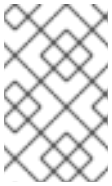
참고

3-노드 클러스터를 설치하는 경우 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템을 설치할 때 컴퓨팅 머신을 배포하지 마십시오.

7 클러스터에 추가하는 컨트롤 플레인 시스템의 수입니다. 클러스터에서 이 값을 클러스터의 etcd 끝점 수로 사용하므로 이 값은 배포하는 컨트롤 플레인 시스템의 수와 일치해야 합니다.

8 DNS 레코드에 지정한 클러스터 이름입니다.

9 Pod IP 주소가 할당되는 IP 주소 블록입니다. 이 블록은 기존 물리적 네트워크와 중복되지 않아야합니다. 이러한 IP 주소는 Pod 네트워크에 사용됩니다. 외부 네트워크에서 Pod에 액세스해야 하는 경우, 트래픽을 관리하도록 로드 밸런서와 라우터를 설정해야 합니다.



참고

클래스 E CIDR 범위는 향후 사용을 위해 예약되어 있습니다. 클래스 E CIDR 범위를 사용하려면 네트워킹 환경에서 클래스 E CIDR 범위 내에서 IP 주소를 수락하는지 확인해야 합니다.

10 개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하면 지정된 **cidr** 이외 **/23** 서브넷이 각 노드에 할당되어 510(2^(32 - 23) - 2) Pod IP 주소가 허용됩니다. 외부 네트워크에서 노드에 액세스해야 하는 경우 트래픽을 관리하도록 로드 밸런서와 라우터를 구성합니다.

11 서비스 IP 주소에 사용할 IP 주소 풀입니다. IP 주소 풀은 하나만 입력할 수 있습니다. 이 블록은 기존 물리적 네트워크와 중복되지 않아야합니다. 외부 네트워크에서 서비스에 액세스해야 하는 경우, 트래픽을 관리하도록 로드 밸런서와 라우터를 구성합니다.

12 플랫폼을 **none**으로 설정해야 합니다. 플랫폼에 대한 추가 플랫폼 구성 변수는 지정할 수 없습니다.

중요

플랫폼 유형으로 설치된 클러스터는 Machine API로 컴퓨팅 머신 관리와 같은 일부 기능을 사용할 수 없습니다. **이 제한은 클러스터에 연결된 컴퓨팅 시스템이 일반적으로 기능을 지원하는 플랫폼에 설치된 경우에도 적용됩니다. 이 매개변수는 설치 후 변경할 수 없습니다.**

13

FIPS 모드 활성화 또는 비활성화 여부입니다. 기본적으로 FIPS 모드는 비활성화됩니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.

중요

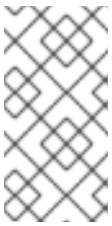
FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 x86_64 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.

14

Red Hat OpenShift Cluster Manager의 풀 시크릿. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

15

RHCOS(Red Hat Enterprise Linux CoreOS)의 core 사용자에 대한 SSH 공용 키입니다.



참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 ssh-agent 프로세스가 사용하는 SSH 키를 지정합니다.

추가 리소스

- **API 및 애플리케이션 수신 로드 밸런싱 요구 사항에 대한 자세한 내용은 사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구 사항을 참조하십시오.**

8.2.9.3. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 HTTP 또는 HTTPS 프록시를 사용

할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 **OpenShift Container Platform** 클러스터를 구성할 수 있습니다.

참고

바이메탈 설치의 경우, **install-config.yaml** 파일의 **networking.machineNetwork[].cidr** 필드에 지정된 범위 밖의 노드 **IP** 주소를 지정하지 않는 경우, **proxy.noProxy** 필드에 포함시켜야 합니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 **API**에 대한 호출을 포함하여 모든 클러스터 발신 **(Egress)** 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.

참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr, networking.clusterNetwork[].cidr, networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 **Red Hat OpenStack Platform (RHOSP)**에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점**(169.254.169.254)**로 채워집니다.

절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
```

```
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1

클러스터 외부에서 **HTTP** 연결을 구축하는 데 사용할 프록시 **URL**입니다. **URL** 스키마는 **http**여야 합니다.

2

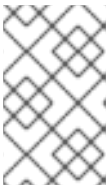
클러스터 외부에서 **HTTPS** 연결을 구축하는 데 사용할 프록시 **URL**입니다.

3

대상 도메인 이름, **IP** 주소 또는 프록시에서 제외할 기타 네트워크 **CIDR**로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. ***를 사용하여 모든 대상에 대해 프록시를 바이패스합니다.

4

이 값을 제공하면 설치 프로그램에서 추가 **CA** 인증서를 보유할 **openshift -config** 네임스페이스에 **user-ca- bundle** 이라는 구성 맵을 생성합니다. **additionalTrustBundle** 및 하나 이상의 프록시 설정을 제공하는 경우 프록시 오브젝트는 **trustedCA** 필드의 **user-ca-bundle** 구성 맵을 참조하도록 구성됩니다. 그러면 **Cluster Network Operator**에서 **trustedCA** 매개변수에 대해 지정된 콘텐츠를 **RHCOS** 신뢰 번들과 병합하는 **trusted-ca-bundle** 구성 맵을 생성합니다. 프록시의 **ID** 인증서를 **RHCOS** 트러스트 번들에 있는 기관에서 서명하지 않은 경우 **additionalTrustBundle** 필드가 있어야 합니다.



참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 **OpenShift Container Platform**을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 **nil**이 됩니다.

참고

cluster라는 Proxy 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

8.2.9.4. 3개의 노드 클러스터 구성

필요한 경우 세 개의 컨트롤 플레인 시스템으로 구성된 베어 메탈 클러스터에 제로 컴퓨팅 머신을 배포할 수 있습니다. 이를 통해 클러스터 관리자와 개발자들이 테스트, 개발, 프로덕션에 사용할 수 있는 소형화되고 리소스 효율이 높은 클러스터를 제공합니다.

3-노드 OpenShift Container Platform 환경에서 세 개의 컨트롤 플레인 머신을 예약할 수 있습니다. 즉, 애플리케이션 워크로드가 해당 플랫폼에서 실행되도록 예약됩니다.

사전 요구 사항

- 기존 install-config.yaml 파일이 있습니다.

절차

- install-config.yaml 파일에서 다음 compute 스탠자에 표시된 대로 컴퓨팅 복제본 수가 0으로 설정되어 있는지 확인합니다.

```
compute:
- name: worker
  platform: {}
  replicas: 0
```

참고

배포 중인 컴퓨팅 머신 수에 관계없이 사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치할 때 컴퓨팅 머신의 replicas 매개변수 값을 0으로 설정해야 합니다. 설치 프로그램에서 제공하는 설치에서 매개 변수는 클러스터가 생성 및 관리하는 컴퓨팅 머신 수를 제어합니다. 이 설정은 컴퓨팅 시스템이 수동으로 배포되는 사용자 프로비저닝 설치에는 적용되지 않습니다.

3-노드 클러스터 설치의 경우 다음 단계를 따르십시오.

- 컴퓨팅 노드가 0인 3-노드 클러스터를 배포하는 경우 Ingress 컨트롤러 Pod는 컨트롤 플레인 노드에서 실행됩니다. 3-노드 클러스터 배포에서 HTTP 및 HTTPS 트래픽을 컨트롤 플레인 노

드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다. 자세한 내용은 *사용자 프로비저닝 인프라 섹션에 대한 로드 밸런싱 요구 사항* 섹션을 참조하십시오.

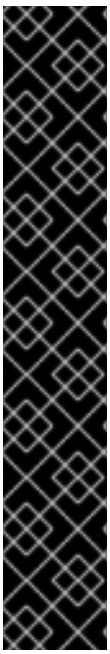
- 다음 절차에서 **Kubernetes** 매니페스트 파일을 생성할 때 **<installation_directory>/manifests/cluster-scheduler-02-config.yml** 파일의 **mastersSchedulable** 매개변수가 **true**로 설정되어 있는지 확인합니다. 이렇게 하면 애플리케이션 워크로드를 컨트롤 플레인 노드에서 실행할 수 있습니다.

- **RHCOS(Red Hat Enterprise Linux CoreOS)** 시스템을 생성할 때 컴퓨팅 노드를 배포하지 마십시오.

8.2.10. Kubernetes 매니페스트 및 Ignition 설정 파일 생성

일부 클러스터 정의 파일을 수정하고 클러스터 시스템을 수동으로 시작해야 하므로 클러스터가 시스템을 구성하는 데 필요한 **Kubernetes** 매니페스트 및 **Ignition** 구성 파일을 사용자가 생성해야 합니다.

설치 구성 파일은 **Kubernetes** 매니페스트로 변환됩니다. 매니페스트는 나중에 클러스터 머신을 구성하는 데 사용되는 **Ignition** 구성 파일로 래핑됩니다.



중요

- **OpenShift Container Platform** 설치 프로그램에서 생성하는 **Ignition** 구성 파일에 **24**시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 **24**시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 **kubelet** 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청**(CSR)**을 수동으로 승인해야 합니다. 자세한 내용은 ***Recovering from expired control plane certificates*** 문서를 참조하십시오.

- 클러스터를 설치한 후 **24**시간에서 **22**시간까지의 인증서가 교체되기 때문에 생성된 후 **12**시간 이내에 **Ignition** 구성 파일을 사용하는 것이 좋습니다. **12**시간 이내에 **Ignition** 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

사전 요구 사항

- **OpenShift Container Platform** 설치 프로그램을 가져오셨습니다.

- install-config.yaml 설치 구성 파일을 생성하셨습니다.

절차

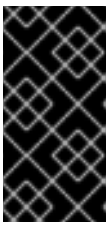
1. OpenShift Container Platform 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터에 대한 Kubernetes 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 <installation_directory>는 사용자가 만든 install-config.yaml 파일이 포함된 설치 디렉터리를 지정합니다.

주의

3 노드 클러스터를 실행 중이면 다음 단계를 건너 뛰어 컨트롤 플레인 노드 일정을 계획할 수 있도록 하십시오.



중요

예약할 수 없는 기본에서 컨트롤 플레인 노드를 구성하면 추가 서브스크립션이 필요합니다. 이는 컨트롤 플레인 노드가 작업자 노드가 되기 때문입니다.

2. <installation_directory>/manifests/cluster-scheduler-02-config.yml Kubernetes 매니페스트 파일의 mastersSchedulable 매개변수가 false로 설정되어 있는지 확인합니다. 이 설정으로 인해 컨트롤 플레인 머신에서 포드가 예약되지 않습니다.

   a. <installation_directory>/manifests/cluster-scheduler-02-config.yml 파일을 엽니다.

   b. mastersSchedulable 매개변수를 찾아서 값을 False로 설정되어 있는지 확인합니다.

c.
파일을 저장하고 종료합니다.

3.
Ignition 구성 파일을 생성하려면 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행합니다.

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1
<installation_directory>는 동일한 설치 디렉터리를 지정합니다.

설치 디렉터리의 부트스트랩, 컨트롤 플레인 및 컴퓨팅 노드에 대한 **Ignition** 구성 파일이 생성됩니다. **kubeadmin-password** 및 **kubeconfig** 파일은 **./<installation_directory>/auth** 디렉터리에 생성됩니다.

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

추가 리소스

- **kubelet** 인증서 복구에 대한 자세한 내용은 만료된 컨트롤 플레인 인증서 복구에서 참조하십시오.

8.2.11. RHCOS 설치 및 OpenShift Container Platform 부트스트랩 프로세스 시작

프로비저닝하는 베어메탈 인프라에 **OpenShift Container Platform**을 설치하려면 머신에 **RHCOS(Red Hat Enterprise Linux CoreOS)**를 설치해야 합니다. **RHCOS**를 설치할 때 설치 중인 머신 유형에 대해 **OpenShift Container Platform** 설치 프로그램에서 생성한 **Ignition** 구성 파일을 제공해야 합니다. 적합한 네트워킹, **DNS** 및 로드 밸런싱 인프라를 구성한 경우 **RHCOS** 머신이 재부팅된 후 **OpenShift Container Platform** 부트스트랩 프로세스가 자동으로 시작됩니다.

단계에 따라 **ISO** 이미지 또는 네트워크 **PXE** 부팅을 사용하여 시스템에 **RHCOS**를 설치합니다.

참고

이 설치 문서에 포함된 컴퓨팅 노드 배포 단계는 RHCOS에 따라 다릅니다. RHEL 기반 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. RHEL 7 컴퓨팅 머신 사용은 더 이상 사용되지 않으며 향후 OpenShift Container Platform 4 릴리스에서 제거될 예정입니다.

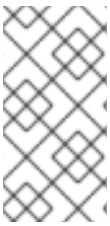
다음 방법을 사용하여 ISO 및 PXE 설치 중에 RHCOS를 구성할 수 있습니다.

- 커널 인수: 커널 인수를 사용하여 설치 관련 정보를 제공할 수 있습니다. 예를 들어 HTTP 서버에 업로드한 RHCOS 설치 파일의 위치와 설치 중인 노드 유형에 대한 Ignition 구성 파일의 위치를 지정할 수 있습니다. PXE 설치의 경우 APPEND 매개 변수를 사용하여 라이브 설치 프로그램의 커널에 인수를 전달할 수 있습니다. ISO 설치의 경우는 라이브 설치 부팅 프로세스를 중단하고 커널 매개 변수를 추가할 수 있습니다. 두 경우 모두 특정 coreos.inst.* 인수를 사용하여 라이브 설치 프로그램을 지시 할 수 있을 뿐 만 아니라 표준 커널 서비스를 활성화/비활성화하기 위해 표준 설치 부팅 인수를 사용할 수 있습니다.

- Ignition 구성: OpenShift Container Platform Ignition 구성 파일(*.ign)은 설치 중인 노드 유형에 따라 다릅니다. RHCOS 설치 중에 부트스트랩, 컨트롤 플레인 또는 컴퓨팅 노드 Ignition 구성 파일의 위치를 전달하여 첫 번째 부팅 시 적용됩니다. 특별한 경우에는 라이브 시스템으로 전달할 별도의 제한된 Ignition 설정을 만들 수 있습니다. 이 Ignition 설정은 설치 완료 후 프로비저닝 시스템에 설치가 성공적으로 완료되었는지를 보고하는 것과 같은 일련의 작업을 수행할 수 있습니다. 이러한 특수 Ignition 구성은 설치된 시스템의 처음 부팅 시 적용되는 coreos-installer에 의해 소비됩니다. 라이브 ISO에 표준 컨트롤 플레인 및 컴퓨팅 노드 Ignition 구성을 직접 제공하지 마십시오.

- coreos-installer : 처음 부팅하기 전에 다양한 방법으로 영구 시스템을 준비 할 수 있도록 쉘 프롬프트에서 라이브 ISO 설치 프로그램을 시작할 수 있습니다. coreos-installer 명령을 실행하여 추가하는 다양한 아티팩트를 식별하고 디스크 파티션을 사용하여 네트워크를 설정할 수 있습니다. 경우에 따라 라이브 시스템에서 기능을 구성하고 설치된 시스템에 복사할 수도 있습니다.

ISO 또는 PXE 설치 사용 여부는 상황에 따라 달라집니다. PXE 설치에는 사용 가능한 DHCP 서비스와 추가 준비가 필요하지만 설치 프로세스를 보다 자동화할 수 있습니다. ISO 설치는 주로 수동적인 프로세스에서 여러 시스템을 설정하는 경우 불편할 수 있습니다.



참고

OpenShift Container Platform 4.6부터 RHCOS ISO 및 기타 설치 아티팩트는 4K 섹터 디스크에 설치를 지원합니다.

8.2.11.1. ISO 이미지를 사용하여 RHCOS 설치

ISO 이미지를 사용하여 시스템에 RHCOS을 설치할 수 있습니다.

사전 요구 사항

- 클러스터에 대한 Ignition 구성 파일이 생성되어 있습니다.
- 적합한 네트워크, DNS 및 로드 밸런싱 인프라가 구성되어 있습니다.
- 사용자 컴퓨터에서 액세스할 수 있고 사용자가 생성한 시스템에서 액세스할 수 있는 HTTP 서버에 대한 액세스 권한을 확보합니다.
- 네트워킹 및 디스크 파티션과 같은 기능을 구성하는 다양한 방법에 대해서는 *고급 **RHCOS** 설치 구성*섹션을 살펴보십시오.

프로세스

1. 각 Ignition 구성 파일에 대해 SHA512 다이제스트를 가져옵니다. 예를 들어 Linux를 실행하는 시스템에서 다음을 사용하여 bootstrap.ign Ignition 구성 파일의 SHA512 다이제스트를 가져올 수 있습니다.

```
$ sha512sum <installation_directory>/bootstrap.ign
```

   다이제스트는 클러스터 노드에서 Ignition 구성 파일의 신뢰성을 검증하기 위해 이후 단계에서 coreos-installer에 제공됩니다.

2. 설치 프로그램에서 생성한 부트스트랩, 컨트롤 플레인, 컴퓨팅 노드 Ignition 구성 파일을 HTTP 서버에 업로드합니다. 해당 파일의 URL을 기록해 둡니다.

   중요

   HTTP 서버에 저장하기 전에 Ignition 구성에서 구성 설정을 추가하거나 변경할 수 있습니다. 설치를 마친 후 클러스터에 컴퓨팅 시스템을 더 추가하려면 Ignition 구성 파일을 삭제하지 마십시오.

3.

설치 호스트에서 Ignition 구성 파일을 URL에서 사용할 수 있는지 확인합니다. 다음 예에서는 부트스트랩 노드에 대한 Ignition 구성 파일을 가져옵니다.

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

출력 예

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-
rsa...
```

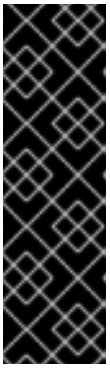
명령에서 bootstrap.ign을 master.ign 또는 worker.ign으로 교체하여 컨트롤 플레인 및 컴퓨팅 노드의 Ignition 구성 파일도 사용할 수 있는지 확인합니다.

4. RHCOS 이미지 미러 페이지에서 운영 체제 인스턴스 설치 방법에 필요한 RHCOS 이미지를 얻을 수 있지만 올바른 버전의 RHCOS 이미지를 얻는 것이 좋습니다. openshift-install 명령 출력에서 사용할 수 있습니다.

```
$ openshift-install coreos print-stream-json | grep '\.iso[^.]'
```

출력 예

```
"location": "<url>/art/storage/releases/rhcos-4.9-aarch64/<release>/aarch64/rhcos-
<release>-live.aarch64.iso",
"location": "<url>/art/storage/releases/rhcos-4.9-ppc64le/<release>/ppc64le/rhcos-
<release>-live.ppc64le.iso",
"location": "<url>/art/storage/releases/rhcos-4.9-s390x/<release>/s390x/rhcos-
<release>-live.s390x.iso",
"location": "<url>/art/storage/releases/rhcos-4.9/<release>/x86_64/rhcos-<release>-
live.x86_64.iso",
```



중요

RHCOS 이미지는 OpenShift Container Platform 릴리스에 따라 변경되지 않을 수 있습니다. 설치하는 OpenShift Container Platform 버전과 같거나 그 이하의 버전 중 가장 최신 버전의 이미지를 다운로드해야 합니다. 사용 가능한 경우 OpenShift Container Platform 버전과 일치하는 이미지 버전을 사용합니다. 이 프로세스에는 ISO 이미지만 사용하십시오. 이 설치 유형에서는 RHCOS qcow2 이미지가 지원되지 않습니다.

ISO 파일 이름은 다음 예와 유사합니다.

```
rhcos-<version>-live.<architecture>.iso
```

5. ISO를 사용하여 RHCOS 설치를 시작합니다. 다음 설치 옵션 중 하나를 사용합니다.

- ISO 이미지를 디스크에 굽고 직접 부팅합니다.
- LOM(Lightweight-out Management) 인터페이스를 사용하여 ISO 리디렉션을 사용합니다.

6. 옵션을 지정하거나 라이브 부팅 시퀀스를 중단하지 않고 RHCOS ISO 이미지를 부팅합니다. 설치 프로그램이 RHCOS 라이브 환경에서 쉘 프롬프트로 부팅될 때까지 기다립니다.

참고

커널 인수를 추가하기 위해 RHCOS 설치 부팅 프로세스를 중단할 수 있습니다. 하지만 이 ISO 프로세스에서는 커널 인수를 추가하지 않고 다음 단계에 설명된 대로 coreos-installer 명령을 사용해야 합니다.

7. coreos-installer 명령을 실행하고 설치 요구 사항을 충족하는 옵션을 지정합니다. 최소한 노드 유형에 대한 Ignition 구성 파일과 설치할 장치를 가리키는 URL을 지정해야 합니다.

```
$ sudo coreos-installer install --ignition-url=http://<HTTP_server>/<node_type>.ign
<device> --ignition-hash=sha512-<digest> 1 2
```

1 1

2

클러스터 노드에서 Ignition 구성 파일을 HTTP URL을 통해 가져오려면 --ignition-hash 옵션이 필요합니다. <digest>는 이전 단계에서 얻은 Ignition 구성 파일 SHA512 다이제스트입니다.

참고

TLS를 사용하는 HTTPS 서버를 통해 Ignition 구성 파일을 제공하려는 경우 coreos-installer를 실행하기 전에 내부 인증 기관(CA)을 시스템 신뢰 저장소에 추가할 수 있습니다.

다음 예제에서는 /dev/sda 장치에 부트스트랩 노드 설치를 초기화합니다. 부트스트랩 노드의 Ignition 구성 파일은 IP 주소 192.168.1.2가 있는 HTTP 웹 서버에서 가져옵니다.

```
$ sudo coreos-installer install --ignition-
url=http://192.168.1.2:80/installation_directory/bootstrap.ign /dev/sda --ignition-
hash=sha512-
a5a2d43879223273c9b60af66b44202a1d1248fc01cf156c46d4a79f552b6bad47bc8cc78dd
f0116e80c59d2ea9e32ba53bc807afbca581aa059311def2c3e3b
```

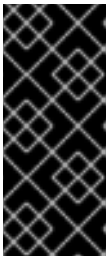
8. 머신 콘솔에서 RHCOS 설치 진행률을 모니터링합니다.

중요

OpenShift Container Platform 설치를 시작하기 전에 각 노드에서 성공적으로 설치되었는지 확인합니다. 설치 프로세스를 관찰하면 발생할 수 있는 RHCOS 설치 문제의 원인을 파악하는 데 도움이 될 수 있습니다.

9. RHCOS를 설치한 후 시스템을 재부팅해야 합니다. 시스템이 재부팅되는 동안 지정한 Ignition 구성 파일이 적용됩니다.

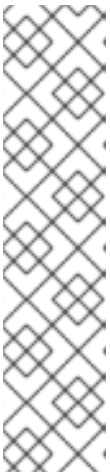
10. 계속해서 클러스터에 대한 나머지 시스템을 모두 생성합니다.



중요

이때 부트스트랩 및 컨트롤 플레인 시스템을 생성해야 합니다. 컨트롤 플레인 시스템이 예약 가능하지 않은 경우 **OpenShift Container Platform**을 설치하기 전에 두 개 이상의 컴퓨팅 시스템도 생성합니다.

적합한 네트워킹, **DNS** 및 로드 밸런싱 인프라가 있는 경우 **RHCOS** 노드가 재부팅된 후 **OpenShift Container Platform** 부트스트랩 프로세스가 자동으로 시작됩니다.



참고

RHCOS 노드에는 **core** 사용자의 기본 암호가 포함되지어 있지 않습니다. **install_config.yaml** 파일에 지정한 공개 키에 대한 액세스 권한이 있는 사용자로 **ssh core@<node>.<cluster_name>.<base_domain>**을 실행하여 노드에 액세스할 수 있습니다. **RHCOS**를 실행하는 **OpenShift Container Platform 4** 클러스터 노드는 변경할 수 없으며 **Operator**를 통해 클러스터 변경 사항을 적용합니다. **SSH**를 사용하여 클러스터 노드에 액세스하는 것은 권장되지 않습니다. 그러나 설치 문제를 조사할 때 **OpenShift Container Platform API**를 사용할 수 없거나 **kubelet**이 대상 노드에서 제대로 작동하지 않는 경우 디버깅 또는 재해 복구에 **SSH** 액세스가 필요할 수 있습니다.

8.2.11.2. PXE 또는 iPXE 부팅을 사용하여 RHCOS 설치

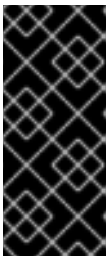
PXE 또는 **iPXE** 부팅을 사용하여 시스템에 **RHCOS**를 설치할 수 있습니다.

사전 요구 사항

- 클러스터에 대한 **Ignition** 구성 파일이 생성되어 있습니다.
- 적합한 네트워크, **DNS** 및 로드 밸런싱 인프라가 구성되어 있습니다.
- 적합한 **PXE** 또는 **iPXE** 인프라가 구성되어 있습니다.
- 사용자 컴퓨터에서 액세스할 수 있고 사용자가 생성한 시스템에서 액세스할 수 있는 **HTTP** 서버에 대한 액세스 권한을 확보합니다.
- 네트워킹 및 디스크 파티션과 같은 기능을 구성하는 다양한 방법에 대해서는 *고급 **RHCOS** 설치 구성*섹션을 살펴보십시오.

프로세스

1. 설치 프로그램에서 생성한 부트스트랩, 컨트롤 플레인, 컴퓨팅 노드 Ignition 구성 파일을 HTTP 서버에 업로드합니다. 해당 파일의 URL을 기록해 둡니다.



중요

HTTP 서버에 저장하기 전에 Ignition 구성에서 구성 설정을 추가하거나 변경할 수 있습니다. 설치를 마친 후 클러스터에 컴퓨팅 시스템을 더 추가하려면 Ignition 구성 파일을 삭제하지 마십시오.

2. 설치 호스트에서 Ignition 구성 파일을 URL에서 사용할 수 있는지 확인합니다. 다음 예에서는 부트스트랩 노드에 대한 Ignition 구성 파일을 가져옵니다.

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

출력 예

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-
rsa...
```

명령에서 bootstrap.ign을 master.ign 또는 worker.ign으로 교체하여 컨트롤 플레인 및 컴퓨팅 노드의 Ignition 구성 파일도 사용할 수 있는지 확인합니다.

3. RHCOS 커널,initramfs 및 rootfs 파일을 RHCOS 이미지 미러 페이지에서 원하는 운영 체제 인스턴스를 설치하는 데 필요한 경우, RHCOS 파일의 올바른 버전을 가져오는 권장 방법은 openshift-install 명령 출력에서 얻을 수 있습니다.

```
$ openshift-install coreos print-stream-json | grep -Eo '"https.*(kernel-
|initramfs.|rootfs.)\w+(\.img)?"'
```

출력 예

```
"<url>/art/storage/releases/rhcos-4.9-aarch64/<release>/aarch64/rhcos-<release>-live-kernel-aarch64"
"<url>/art/storage/releases/rhcos-4.9-aarch64/<release>/aarch64/rhcos-<release>-live-initramfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.9-aarch64/<release>/aarch64/rhcos-<release>-live-rootfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.9-ppc64le/49.84.202110081256-0/ppc64le/rhcos-<release>-live-kernel-ppc64le"
"<url>/art/storage/releases/rhcos-4.9-ppc64le/<release>/ppc64le/rhcos-<release>-live-initramfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.9-ppc64le/<release>/ppc64le/rhcos-<release>-live-rootfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.9-s390x/<release>/s390x/rhcos-<release>-live-kernel-s390x"
"<url>/art/storage/releases/rhcos-4.9-s390x/<release>/s390x/rhcos-<release>-live-initramfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.9-s390x/<release>/s390x/rhcos-<release>-live-rootfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.9/<release>/x86_64/rhcos-<release>-live-kernel-x86_64"
"<url>/art/storage/releases/rhcos-4.9/<release>/x86_64/rhcos-<release>-live-initramfs.x86_64.img"
"<url>/art/storage/releases/rhcos-4.9/<release>/x86_64/rhcos-<release>-live-rootfs.x86_64.img"
```

중요

OpenShift Container Platform의 모든 릴리스에서 **RHCOS** 아티팩트가 변경되지 않을 수도 있습니다. 설치하는 **OpenShift Container Platform** 버전과 같거나 그 이하의 버전 중 가장 최신 버전의 이미지를 다운로드해야 합니다. 이 프로시저에는 아래 설명된 적절한 **kernel, initramfs** 및 **rootfs** 아티팩트만 사용하십시오. 이 설치 유형에서는 **RHCOS QCOW2** 이미지가 지원되지 않습니다.

OpenShift Container Platform 버전 번호가 파일 이름에 포함됩니다. 다음 예와 유사합니다.

- **kernel: rhcos-<version>-live-kernel-<architecture>**
- **initramfs: rhcos-<version>-live-initramfs.<architecture>.img**

- rootfs: rhcos-<version>-live-rootfs.<architecture>.img

4. rootfs, kernel 및 initramfs 파일을 HTTP 서버에 업로드합니다.

중요

설치를 마친 후 클러스터에 컴퓨팅 시스템을 더 추가하려면 Ignition 구성 파일을 삭제하지 마십시오.

5. RHCOS가 설치된 후 시스템이 로컬 디스크에서 부팅되도록 네트워크 부팅 인프라를 구성하십시오.

6. RHCOS 이미지에 대한 PXE 또는 iPXE 설치를 구성하고 설치를 시작하십시오.

사용 환경에 대한 다음 예시 메뉴 항목 중 하나를 수정하고, 이미지 및 Ignition 파일에 적절히 접근할 수 있는지 확인하십시오.

- PXE의 경우:

```
DEFAULT pxeboot
TIMEOUT 20
PROMPT 0
LABEL pxeboot
    KERNEL http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> (1)
    APPEND initrd=http://<HTTP_server>/rhcos-<version>-live-initramfs.
<architecture>.img coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-
rootfs.<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign (2) (3)
```

(1) (1)

HTTP 서버에 업로드한 라이브 kernel 파일의 위치를 지정합니다. URL은 HTTP, TFTP 또는 FTP여야 합니다. HTTPS와 NFS는 지원되지 않습니다.

(2)

NIC를 여러 개 사용하는 경우 ip 옵션에 단일 인터페이스를 지정합니다. 예를 들어, eno1라는 NIC에서 DHCP를 사용하려면 ip=eno1:dhcp를 설정하십시오.

(3)

참고

이 구성은 그래픽 콘솔이 있는 시스템에서 직렬 콘솔 액세스를 활성화하지 않습니다. 다른 콘솔을 구성하려면 **APPEND** 행에 하나 이상의 **console=** 인수를 추가합니다. 예를 들어 **console=tty0 console=ttyS0**을 추가하여 첫 번째 **PC** 직렬 포트를 기본 콘솔로 설정하고 그래픽 콘솔을 보조 콘솔로 설정합니다. 자세한 내용은 **Red Hat Enterprise Linux에서 직렬 터미널 및/또는 콘솔 설정 방법**을 참조하십시오.

- **iPXE**의 경우 :

```
kernel http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> initrd=main
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign 1 2
initrd --name main http://<HTTP_server>/rhcos-<version>-live-initramfs.
<architecture>.img 3
boot
```

1

HTTP 서버에 업로드한 **RHCOS** 파일의 위치를 지정합니다. **kernel** 매개변수 값은 **kernel** 파일의 위치이고 **initrd=main** 인수는 **UEFI** 시스템에서 부팅하는 데 필요하며 **coreos.live.rootfs_url** 매개 변수 값은 **rootfs** 파일의 위치이며, **coreos.inst.ignition_url** 매개 변수 값은 부트스트랩 **Ignition** 설정 파일의 위치입니다.

2

NIC를 여러 개 사용하는 경우 **ip** 옵션에 단일 인터페이스를 지정합니다. 예를 들어, **eno1**라는 **NIC**에서 **DHCP**를 사용하려면 **ip=eno1:dhcp**를 설정하십시오.

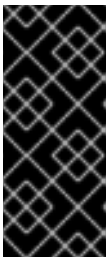
3

HTTP 서버에 업로드한 **initramfs** 파일의 위치를 지정합니다.

참고

이 구성은 그래픽 콘솔이 있는 시스템에서 직렬 콘솔 액세스를 활성화하지 않습니다. 다른 콘솔을 구성하려면 **kernel** 행에 하나 이상의 **console=** 인수를 추가합니다. 예를 들어 **console=tty0 console=ttyS0**을 추가하여 첫 번째 **PC** 직렬 포트를 기본 콘솔로 설정하고 그래픽 콘솔을 보조 콘솔로 설정합니다. 자세한 내용은 **Red Hat Enterprise Linux**에서 직렬 터미널 및/또는 콘솔 설정 방법을 참조하십시오.

7. 머신 콘솔에서 **RHCOS** 설치 진행률을 모니터링합니다.

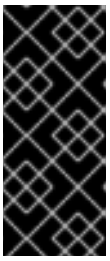


중요

OpenShift Container Platform 설치를 시작하기 전에 각 노드에서 성공적으로 설치되었는지 확인합니다. 설치 프로세스를 관찰하면 발생할 수 있는 **RHCOS** 설치 문제의 원인을 파악하는 데 도움이 될 수 있습니다.

8. **RHCOS**가 설치되면 시스템을 재부팅합니다. 시스템이 재부팅되는 동안 지정한 **Ignition** 구성 파일이 적용됩니다.

9. 클러스터용 시스템 생성을 계속합니다.



중요

이때 부트스트랩 및 컨트롤 플레인 시스템을 생성해야 합니다. 컨트롤 플레인 시스템이 예약 가능하지 않은 경우 클러스터를 설치하기 전에 두 개 이상의 컴퓨팅 시스템도 생성합니다.

적합한 네트워킹, **DNS** 및 로드 밸런싱 인프라가 있는 경우 **RHCOS** 노드가 재부팅된 후 **OpenShift Container Platform** 부트스트랩 프로세스가 자동으로 시작됩니다.

참고

RHCOS 노드에는 core 사용자의 기본 암호가 포함되지어 있지 않습니다. install_config.yaml 파일에 지정한 공개 키에 대한 액세스 권한이 있는 사용자로 ssh core@<node>.<cluster_name>.<base_domain>을 실행하여 노드에 액세스할 수 있습니다. RHCOS를 실행하는 OpenShift Container Platform 4 클러스터 노드는 변경할 수 없으며 Operator를 통해 클러스터 변경 사항을 적용합니다. SSH를 사용하여 클러스터 노드에 액세스하는 것은 권장되지 않습니다. 그러나 설치 문제를 조사할 때 OpenShift Container Platform API를 사용할 수 없거나 kubelet이 대상 노드에서 제대로 작동하지 않는 경우 디버깅 또는 재해 복구에 SSH 액세스가 필요할 수 있습니다.

8.2.11.3. 고급 RHCOS 설치 구성 옵션

OpenShift Container Platform용 RHCOS (Red Hat Enterprise Linux CoreOS) 노드를 수동으로 프로비저닝하는 주요 이점은 기본 OpenShift Container Platform 설치 방법을 통해 사용할 수 없는 구성을 수행할 수 있는 것입니다. 이 섹션에서는 다음과 같은 방법을 사용하여 수행할 수있는 몇 가지 구성에 대해 설명합니다.

- 라이브 설치 프로그램에 커널 인수 전달
- 라이브 시스템에서 수동으로 coreos-installer 실행
- ISO에 Ignition 구성 포함

이 섹션에 설명되어 있는 수동 Red Hat Enterprise Linux CoreOS(RHCOS) 설치에 대한 고급 구성 항목은 디스크 파티션 설정, 네트워킹 및 다양한 방식의 Ignition 구성 사용과 관련되어 있습니다.

8.2.11.3.1. PXE 및 ISO 설치를 위한 고급 네트워크 옵션 사용

OpenShift Container Platform 노드의 네트워크는 기본적으로 DHCP를 사용하여 필요한 모든 구성 설정을 수집합니다. 고정 IP 주소를 설정하거나 본딩과 같은 특정 설정을 구성하려면 다음 중 하나의 방법으로 수행할 수 있습니다.

- 라이브 설치 프로그램을 시작할 때 특수 커널 매개 변수를 전달합니다.
- 머신 구성을 사용하여 네트워크 파일을 설치된 시스템에 복사합니다.

- 라이브 설치 프로그램 쉘 프롬프트에서 네트워크를 구성한 다음 설치된 시스템에 복사하여 설치한 시스템을 처음 시작할 때 사용하도록합니다.

PXE 또는 iPXE 설치를 구성하려면 다음 옵션 중 하나를 사용합니다.

- "고급 RHCOS 설치 참조" 표를 참조하십시오.

- 머신 구성을 사용하여 네트워크 파일을 설치된 시스템에 복사합니다.

다음 프로세스에 따라 ISO 설치를 구성합니다.

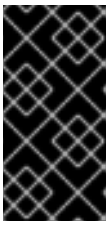
절차

1. ISO 설치 프로그램을 시작합니다.

2. 라이브 시스템 쉘 프롬프트에서 사용 가능한 RHEL 도구 (예: nmcli 또는 nmtui)를 사용하여 라이브 시스템의 네트워킹를 구성합니다.

3. coreos-installer 명령을 실행하여 시스템을 설치하고 --copy-network 옵션을 추가하여 네트워크 구성을 복사합니다. 예를 들면 다음과 같습니다.

```
$ sudo coreos-installer install --copy-network \
    --ignition-url=http://host/worker.ign /dev/sda
```



중요

copy-network 옵션은 /etc/NetworkManager/system-connections에 있는 네트워킹 구성만 복사합니다. 특히 시스템 호스트 이름을 복사하지 않습니다.

4. 설치된 시스템으로 재부팅하십시오.

추가 리소스

- nmcli 및 nmtui 툴에 대한 자세한 내용은 RHEL 8 설명서에서 nmcli로 시작하기 및 nmtui

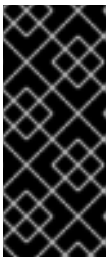
로 시작하기를 참조하십시오.

8.2.11.3.2. 디스크 파티션 설정

디스크 파티션은 **RHCOS(Red Hat Enterprise Linux CoreOS)** 설치 중에 **OpenShift Container Platform** 클러스터 노드에 생성됩니다. 특정 아키텍처의 각 **RHCOS** 노드는 기본 파티션 구성을 재정의하지 않는 한 동일한 파티션 레이아웃을 사용합니다. **RHCOS** 설치 중에 대상 장치에서 사용 가능한 나머지 공간을 사용하도록 루트 파일 시스템의 크기가 증가합니다.

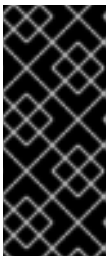
OpenShift Container Platform 클러스터 노드에 **RHCOS**를 설치할 때 다음과 같은 두 가지 경우 기본 파티셔닝을 재정의할 수 있습니다.

- 별도의 파티션 생성: 빈 디스크에 그린 필드 설치의 경우 파티션에 별도의 스토리지를 추가할 수 있습니다. 이는 **/var** 또는 **/var**의 하위 디렉터리 (예: **/var/lib/etcd**)중 하나를 별도의 파티션으로 만드는 경우에 공식적으로 지원되지만 둘 다 지원되지는 않습니다.



중요

100GB 이상의 디스크 크기 및 특히 **1TB**보다 큰 디스크 크기의 경우 별도의 **/var** 파티션을 만듭니다. 자세한 내용은 " 별도의 **/var** 파티션 생성" 및 이 Red Hat 지식베이스 문서 를 참조하십시오.



중요

Kubernetes는 두 개의 파일 시스템 파티션만 지원합니다. 원래 구성에 둘 이상의 파티션을 추가하는 경우 **Kubernetes**는 모든 파티션을 모니터링할 수 없습니다.

- 기존 파티션 유지: 브라운 필드 설치에서 기존 노드에 **OpenShift Container Platform**을 재설치하고 이전 운영 체제에 설치된 데이터 파티션을 유지해야 하는 경우 기존 데이터 파티션을 유지할 수 있는 **coreos-installer**에 부팅 인수와 옵션이 모두 있습니다.

주의

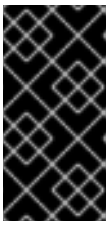
사용자 지정 파티션을 사용하면 OpenShift Container Platform에서 해당 파티션을 모니터링하지 않거나 경고를 받을 수 있습니다. 기본 파티션을 재정의하는 경우 OpenShift Container Platform이 호스트 파일 시스템을 모니터링하는 방법에 대한 자세한 내용은 OpenShift File System Monitoring(제거 조건) 이해를 참조하십시오.

8.2.11.3.2.1. 별도의 /var 파티션 만들기

일반적으로 RHCOS 설치 중에 생성된 기본 디스크 파티션을 사용해야 합니다. 그러나 확장하려는 디렉토리에 별도의 파티션을 생성해야 하는 경우가 있습니다.

OpenShift 컨테이너 플랫폼은 /var 디렉토리 또는 /var의 하위 디렉터리 중 하나에 스토리지를 연결하는 단일 파티션의 추가를 지원합니다. 예를 들면 다음과 같습니다.

- /var/lib/containers: 시스템에 더 많은 이미지와 컨테이너가 추가됨에 따라 확장될 수 있는 컨테이너 관련 콘텐츠를 보관합니다.
- /var/lib/etcd: etcd 스토리지의 성능 최적화와 같은 목적으로 별도로 보관할 데이터를 보관합니다.
- /var: 감사 등의 목적에 맞게 별도로 분리하여 보관해야 하는 데이터를 보관합니다.



중요

100GB보다 큰 디스크 크기 및 특히 1TB보다 큰 디스크의 경우 별도의 /var 파티션을 만듭니다.

/var 디렉터리의 콘텐츠를 별도로 저장하면 필요에 따라 해당 영역에 대한 스토리지 확장을 보다 용이하게 하고 나중에 OpenShift Container Platform을 다시 설치하여 해당 데이터를 그대로 보존할 수 있습니다. 이 방법을 사용하면 모든 컨테이너를 다시 가져올 필요가 없으며 시스템을 업데이트할 때 대용량 로그 파일을 복사할 필요도 없습니다.

/var 디렉토리 또는 /var의 하위 디렉토리에 대해 별도의 파티션을 사용하면 분할된 디렉토리의 데이터 증가로 루트 파일 시스템이 채워지는 것을 방지할 수 있습니다.

다음 절차에서는 설치 준비 단계에서 노드 유형의 Ignition 구성 파일에 래핑되는 머신 구성 매니페스트를 추가하여 별도의 /var 파티션을 설정합니다.

절차

1. 설치 호스트에서 OpenShift Container Platform 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터에 대한 Kubernetes 매니페스트를 생성합니다.

```
$ openshift-install create manifests --dir <installation_directory>
```

2. 추가 파티션을 구성하는 Butane 구성을 생성합니다. 예를 들어 파일 이름을 $HOME/clusterconfig/98-var-partition.bu 로 지정하고, 디스크 장치 이름을 worker 시스템의 스토리지 장치 이름으로 변경하고 스토리지 크기를 적절하게 설정합니다. 이 예에서는 /var 디렉토리를 별도의 파티션에 배치합니다.

```
variant: openshift
version: 4.9.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1 파티션을 설정해야하는 디스크 저장 장치 이름입니다.

2 데이터 파티션을 부트 디스크에 추가할 때 최소 오프셋 값 25000 메비 바이트가 권장됩니다. 루트 파일 시스템은 지정된 오프셋까지 사용 가능한 모든 공간을 채우기 위해 자동으로 크기가 조정됩니다. 오프셋 값이 지정되지 않거나 지정된 값이 권장 최소값보다 작으면

생성되는 루트 파일 시스템의 크기가 너무 작아지고 **RHCOS**를 나중에 다시 설치할 때 데이터 파티션의 첫 번째 부분을 덮어 쓸 수 있습니다.

3

데이터 파티션의 크기(**MB**)입니다.

4

컨테이너 스토리지에 사용되는 파일 시스템에 **prjquota** 마운트 옵션을 활성화해야 합니다.

참고

별도의 **/var** 파티션을 만들 때 다른 인스턴스 유형에 동일한 장치 이름이 없는 경우 컴퓨팅 노드에 다른 인스턴스 유형을 사용할 수 없습니다.

3. **Butane** 구성에서 매니페스트를 생성하여 **clusterconfig/openshift** 디렉터리에 저장합니다. 예를 들어 다음 명령을 실행합니다.

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o
$HOME/clusterconfig/openshift/98-var-partition.yaml
```

4. **Ignition** 구성 파일을 만듭니다.

```
$ openshift-install create ignition-configs --dir <installation_directory> 1
```

1

<installation_directory>는 동일한 설치 디렉터리를 지정합니다.

설치 디렉터리의 부트스트랩, 컨트롤 플레인 및 컴퓨팅 노드에 대한 **Ignition** 구성 파일이 생성됩니다.

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
```

```
├── master.ign
├── metadata.json
└── worker.ign
```

<installation_directory>/manifest 및 **<installation_directory>/openshift** 디렉터리의 파일은 **98-var-partition** 사용자 정의 **MachineConfig** 오브젝트가 포함된 파일을 포함하여 **Ignition** 구성 파일로 래핑됩니다.

다음 단계

- **RHCOS** 설치 중에 **Ignition** 구성 파일을 참조하여 사용자 정의 디스크 파티션을 적용할 수 있습니다.

8.2.11.3.2.2. 기존 파티션 유지

ISO 설치의 경우 설치 프로그램이 하나 이상의 기존 파티션을 유지하도록하는 옵션을 **coreos-installer** 명령에 추가할 수 있습니다. **PXE** 설치의 경우 **coreos.inst.*** 옵션을 **APPEND** 매개 변수에 추가하여 파티션을 유지할 수 있습니다.

저장된 파티션은 기존 **OpenShift Container Platform** 시스템의 데이터 파티션이 될 수 있습니다. 파티션 레이블 또는 번호 중 하나로 보관하려는 디스크 파티션을 확인할 수 있습니다.

참고

기존 파티션을 저장하고 해당 파티션이 **RHCOS**를 위한 충분한 공간을 남겨 두지 않으면 저장된 파티션이 손상되지는 않지만 설치에 실패합니다.

ISO 설치 중 기존 파티션 유지

이 예에서는 파티션 레이블이 **data (data*)**로 시작하는 모든 파티션을 유지합니다.

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign \
    --save-partlabel 'data*' /dev/sda
```

다음 예는 디스크의 여섯 번째 (6) 파티션을 유지하는 방식으로 **coreos-installer**를 실행하는 방법을 보여줍니다.

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign \
    --save-partindex 6 /dev/sda
```

이 예에서는 파티션 5 이상을 유지합니다.

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign
      --save-partindex 5- /dev/sda
```

파티션 저장이 사용된 이전 예에서 coreos-installer는 파티션을 즉시 다시 만듭니다.

PXE 설치 중 기존 파티션 유지

이 APPEND 옵션은 파티션 레이블이 'data'('data *')로 시작하는 모든 파티션을 유지합니다.

```
coreos.inst.save_partlabel=data*
```

이 APPEND 옵션은 파티션 5 이상을 유지합니다.

```
coreos.inst.save_partindex=5-
```

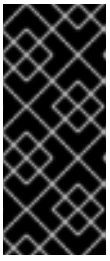
이 APPEND 옵션은 파티션 6을 유지합니다.

```
coreos.inst.save_partindex=6
```

8.2.11.3.3. Ignition 설정 확인

RHCOS 베어 메탈 설치를 수행할 때 제공할 수 있는 두 가지 유형의 Ignition 구성이 있으며 각 구성을 제공하는 이유도 각각 다릅니다.

- Permanent install Ignition config: 모든 수동 RHCOS 설치는 설치를 수행하기 위해 openshift-installer가 생성한 Ignition 구성 파일 (예: bootstrap.ign, master.ign 및 worker.ign) 중 하나를 전달해야 합니다.



중요

이러한 Ignition 구성 파일을 직접 수정하지 않는 것이 좋습니다. 이전 섹션의 예에 설명된 대로 Ignition 구성 파일로 래핑된 매니페스트 파일을 업데이트할 수 있습니다.

PXE 설치의 경우 coreos.inst.ignition_url= 옵션을 사용하여 APPEND 행에서 Ignition 구

성을 전달합니다. ISO 설치의 경우 쉘 프롬프트에서 ISO를 시작한 후 coreos-installer 명령 줄에서 --ignition-url= 옵션을 사용하여 Ignition 구성을 식별합니다. 두 경우 모두 HTTP 및 HTTPS 프로토콜만 지원됩니다.

- Live install Ignition config :이 유형은 수동으로 작성되어 Red Hat에서 지원되지 않으므로 가능하면 사용하지 않도록 해야 합니다. 이 방법을 사용하면 Ignition 구성이 라이브 설치 미디어로 전달되고 부팅시 즉시 실행되며 RHCOS 시스템이 디스크에 설치되기 전후에 설치 작업을 수행합니다. 이 방법은 시스템 구성을 사용하여 실행할 수 없는 고급 파티션 설정과 같이 한 번만 수행하고 나중에 다시 적용할 필요가 없는 작업의 실행에만 사용해야 합니다.

  PXE 또는 ISO 부팅의 경우 Ignition 설정을 만들고 ignition.config.url= 옵션에 APPEND를 실행하여 Ignition 설정 위치를 확인할 수 있습니다. 또한 ignition.firstboot ignition.platform.id = metal도 추가해야 합니다. 추가하지 않으면 ignition.config.url 옵션이 무시됩니다.

8.2.11.3.3.1. RHCOS ISO에 실시간 설치 Ignition 구성 포함

RHCOS ISO 이미지에 직접 라이브 설치 Ignition 구성을 포함할 수 있습니다. ISO 이미지를 부팅하면 내장된 구성이 자동으로 적용됩니다.

절차

1. 다음 이미지 미러 페이지에서 coreos-installer 바이너리를 다운로드합니다. https://mirror.openshift.com/pub/openshift-v4/clients/coreos-installer/latest/.

2. RHCOS ISO 이미지와 Ignition 구성 파일을 검색하고 이를 액세스 가능한 디렉터리 (예: /mnt)에 복사합니다.

```
# cp rhcos-<version>-live.x86_64.iso bootstrap.ign /mnt/
# chmod 644 /mnt/rhcos-<version>-live.x86_64.iso
```

3. 다음 명령을 실행하여 Ignition 구성을 ISO에 포함합니다.

```
# ./coreos-installer iso ignition embed -i /mnt/bootstrap.ign \
    /mnt/rhcos-<version>-live.x86_64.iso
```

이제 해당 ISO를 사용하여 지정된 라이브 설치 Ignition 구성을 사용하여 RHCOS를 설치할 수 있습니다.

중요

coreos-installer iso ignition embed를 사용하여 **bootstrap.ign, master.ign** 및 **worker.ign** 과 같이 **openshift-installer**에 의해 생성된 파일을 포함하는 것은 지원되지 않으며 권장되지 않습니다.

4. 포함된 **Ignition** 구성 내용을 표시하고 이를 파일로 보내려면 다음을 실행합니다.

```
# ./coreos-installer iso ignition show /mnt/rhcos-<version>-live.x86_64.iso >
mybootstrap.ign
```

```
# diff -s bootstrap.ign mybootstrap.ign
```

출력 예

```
Files bootstrap.ign and mybootstrap.ign are identical
```

5. **Ignition** 구성을 제거하고 다시 사용할 수 있도록 **ISO**를 초기 상태로 복원하려면 다음을 실행합니다.

```
# ./coreos-installer iso ignition remove /mnt/rhcos-<version>-live.x86_64.iso
```

이제 다른 **Ignition** 구성을 **ISO**에 포함하거나 초기 상태의 **ISO**를 사용할 수 있습니다.

8.2.11.3.4. 고급 RHCOS 설치 참조

여기서는 **RHCOS(Red Hat Enterprise Linux CoreOS)** 수동 설치 프로세스를 수정하는 데 사용할 수 있는 네트워킹 구성 및 기타 고급 옵션에 대해 설명합니다. 다음 표에서는 **RHCOS** 라이브 설치 프로그램 및 **coreos-installer** 명령과 함께 사용할 수있는 커널 인수 및 명령 줄 옵션에 대해 설명합니다.

8.2.11.3.4.1. ISO 설치를 위한 네트워킹 및 본딩 옵션

ISO 이미지에서 **RHCOS**를 설치하는 경우, 해당 이미지를 부팅할 때 수동으로 커널 인수를 추가하여 노드의 네트워킹을 구성할 수 있습니다. 네트워킹 인수를 지정하지 않으면 **RHCOS**에서 **Ignition** 구성 파일을 가져오는 데 네트워킹이 필요함을 감지하면 **initramfs**에서 **DHCP**가 활성화됩니다.

중요

네트워킹 인수를 수동으로 추가할 때 **initramfs**에서 네트워크를 가져오려면 **rd.neednet=1** 커널 인수도 추가해야 합니다.

다음 표는 **ISO** 설치를 위해 **RHCOS(Red Hat Enterprise Linux CoreOS)** 노드의 네트워킹 및 본딩 구성 예를 보여줍니다. 예제에서는 **ip=, nameserver=, bond=** 커널 인수를 사용하는 방법을 설명합니다.

참고

커널 인수를 추가할 때 순서가 중요합니다: **ip=, nameserver=** 및 **bond=** 입니다.

이는 시스템 부팅 중에 **dracut** 툴로 전달되는 네트워킹 옵션입니다. **dracut**에서 지원하는 네트워킹 옵션에 대한 자세한 내용은 **dracut.cmdline** 메뉴얼 페이지를 참조하십시오.

다음 정보는 **ISO** 설치를 위해 **RHCOS** 노드에서 네트워킹을 구성하는 예제를 제공합니다. 예제에서는 **ip=** 및 **nameserver=** 커널 인수를 사용하는 방법을 설명합니다.

참고

커널 인수를 추가할 때 순서가 중요합니다(**ip =** 및 **nameserver=).**

이는 시스템 부팅 중에 **dracut** 툴로 전달되는 네트워킹 옵션입니다. **dracut**에서 지원하는 네트워킹 옵션에 대한 자세한 내용은 **dracut.cmdline** 메뉴얼 페이지를 참조하십시오.

다음 예제는 **ISO** 설치를 위한 네트워킹 옵션입니다.

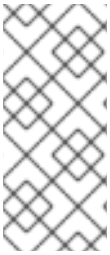
DHCP 또는 고정 **IP** 주소 구성

IP 주소를 구성하려면 **DHCP(ip=dhcp)**를 사용하거나 개별 고정 **IP** 주소(**ip=<host_ip>**)를 설정합니다. 정적 **IP**를 설정하는 경우 각 노드에서 **DNS** 서버 **IP** 주소 (**nameserver=<dns_ip>**)를 확인합니다. 다음 예제에서는 다음을 설정합니다.

- 노드의 **IP** 주소는 **10.10.10.2**로 설정됩니다.

- 게이트웨이 주소는 **10.10.10.254**로 설정됩니다.
- 넷마스크는 **255.255.255.0**으로
- **core0.example.com**에 대한 호스트 이름
- **4.4.4.41**의 **DNS** 서버 주소
- **auto-configuration** 값을 **none**으로 설정합니다. **IP** 네트워킹이 정적으로 구성되면 자동 구성이 필요하지 않습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
nameserver=4.4.4.41
```



참고

DHCP를 사용하여 **RHCOS** 시스템의 **IP** 주소 지정을 구성하는 경우 시스템은 **DHCP**를 통해 **DNS** 서버 정보도 가져옵니다. **DHCP** 기반 배포의 경우 **DHCP** 서버 구성을 통해 **RHCOS** 노드에서 사용할 **DNS** 서버 주소를 정의할 수 있습니다.

고정 호스트 이름 없이 **IP** 주소 구성

정적 호스트 이름을 할당하지 않고 **IP** 주소를 구성할 수 있습니다. 사용자가 정적 호스트 이름을 설정하지 않으면 역방향 **DNS** 조회에 의해 자동으로 선택됩니다. 정적 호스트 이름이 없는 **IP** 주소를 구성하려면 다음 예제를 참조합니다.

- 노드의 **IP** 주소는 **10.10.10.2**로 설정됩니다.
- 게이트웨이 주소는 **10.10.10.254**로 설정됩니다.
- 넷마스크는 **255.255.255.0**으로
- **4.4.4.41**의 **DNS** 서버 주소

- auto-configuration 값을 none으로 설정합니다. IP 네트워킹이 정적으로 구성되면 자동 구성이 필요하지 않습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0::enp1s0:none
nameserver=4.4.4.41
```

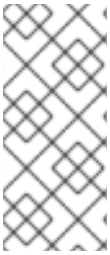
여러 네트워크 인터페이스 지정

여러 ip= 항목을 설정하여 여러 네트워크 인터페이스를 지정할 수 있습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

기본 게이트웨이 및 경로 구성

선택 사항: rd.route= 값을 설정하여 추가 네트워크에 대한 경로를 구성할 수 있습니다.



참고

하나 이상의 네트워크를 구성할 때 하나의 기본 게이트웨이가 필요합니다. 추가 네트워크 게이트웨이가 기본 네트워크 게이트웨이와 다른 경우 기본 게이트웨이가 기본 네트워크 게이트웨이어야 합니다.

- 다음 명령을 실행하여 기본 게이트웨이를 구성합니다.

```
ip=::10.10.10.254::::
```

- 다음 명령을 입력하여 추가 네트워크의 경로를 구성합니다.

```
rd.route=20.20.20.0/24:20.20.20.254:enp2s0
```

단일 인터페이스에서 DHCP 비활성화

두 개 이상의 네트워크 인터페이스가 있고 하나의 인터페이스만 사용되는 경우와 같이 단일 인터페이스에서 DHCP를 비활성화할 수 있습니다. 이 예에서 enp1s0 인터페이스에는 정적 네트워킹 구성이 있으며 enp2s0 용으로 DHCP가 사용되지 않습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=::::core0.example.com:enp2s0:none
```

DHCP 및 고정 IP 구성 결합

다음과 같이 시스템의 **DHCP** 및 고정 **IP** 구성을 여러 네트워크 인터페이스와 결합할 수 있습니다.

```
ip=enp1s0:dhcp
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

개별 인터페이스에서 **VLAN** 구성

선택 사항: **vlan=** 매개변수를 사용하여 개별 인터페이스에서 **VLAN**을 구성할 수 있습니다.

- 네트워크 인터페이스에서 **VLAN**을 구성하고 고정 **IP** 주소를 사용하려면 다음 명령을 실행합니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0.100:none
vlan=enp2s0.100:enp2s0
```

- 네트워크 인터페이스에서 **VLAN**을 구성하고 **DHCP**를 사용하려면 다음 명령을 실행합니다.

```
ip=enp2s0.100:dhcp
vlan=enp2s0.100:enp2s0
```

여러 **DNS** 서버 제공

각 서버에 **nameserver=** 항목을 추가하여 여러 **DNS** 서버를 제공할 수 있습니다. 예를 들면 다음과 같습니다.

```
nameserver=1.1.1.1
nameserver=8.8.8.8
```

단일 인터페이스에 여러 네트워크 인터페이스 본딩

선택 사항: **bond=** 옵션을 사용하여 여러 네트워크 인터페이스를 단일 인터페이스에 결합할 수 있습니다. 다음 예제를 참조하십시오.

- 결합된 인터페이스를 구성하는 구문: **bond = name [: network_interfaces] [: options]**

  name은 결합하는 기기 이름(**bond0**)이고 ***network_interfaces***는 쉼표로 구분되는 물리적(이더넷) 인터페이스 목록(**em1, em2**)이며, ***options***은 쉼표로 구분되는 결합 옵션 목록입니다. 사용 가능한 옵션을 보려면 **modinfo bonding**을 입력하십시오.

- **bond=**를 사용하여 결합된 인터페이스를 생성할 때 **IP** 주소가 할당되는 방법과 결합된 인터페이스에 대한 기타 정보를 지정해야 합니다.

- DHCP를 사용하도록 결합된 인터페이스를 구성하려면 bond의 IP 주소를 dhcp로 설정하십시오. 예를 들면 다음과 같습니다.

```
bond=bond0:em1,em2:mode=active-backup
ip=bond0:dhcp
```

- 고정 IP 주소를 사용하도록 결합된 인터페이스를 구성하려면 원하는 특정 IP 주소 및 관련 정보를 입력합니다. 예를 들면 다음과 같습니다.

```
bond=bond0:em1,em2:mode=active-backup
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
```

단일 인터페이스에 여러 네트워크 인터페이스 본딩

선택 사항: vlan= 매개변수를 사용하고 DHCP를 사용하도록 결합된 인터페이스에서 VLAN을 구성할 수 있습니다. 예를 들면 다음과 같습니다.

```
ip=bond0.100:dhcp
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

다음 예제를 사용하여 VLAN을 사용하여 결합된 인터페이스를 구성하고 고정 IP 주소를 사용합니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0.100:none
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

네트워크 팀 구성 사용

선택 사항: team= 매개변수를 사용하여 네트워크 팀을 결합의 대안으로 사용할 수 있습니다.

- 팀 인터페이스를 구성하는 구문은 team=name[:network_interfaces]입니다.

name 은 팀 장치 이름(team0)이고 *network_interfaces* 는 쉼표로 구분된 실제(이더넷) 인터페이스 목록(em1, em2)을 나타냅니다.

RHCOS가 향후 RHEL 버전으로 전환되면 티밍이 더 이상 사용되지 않을 예정입니다. 자세한 내용은 이 Red Hat 지식베이스 문서를 참조하십시오.

다음 예제를 사용하여 네트워크 팀을 구성합니다.

```
team=team0:em1,em2
ip=team0:dhcp
```

8.2.11.3.4.2. ISO 설치를 위한 coreos-installer 옵션

ISO 이미지에서 RHCOS 라이브 환경으로 부팅한 후 명령 프롬프트에서 **coreos-installer install <options> <device>**를 실행하여 RHCOS를 설치할 수 있습니다.

다음 표는 coreos-installer 명령으로 전달할 수 있는 하위 명령, 옵션 및 인수를 보여줍니다.

표 8.12. coreos-installer 하위 명령, 명령줄 옵션 및 인수

coreos-installer 설치 하위 명령	
하위 명령	*설명*
$ coreos-installer install <options> <device>	ISO 이미지에 Ignition 구성를 삽입합니다.
coreos-installer 설치 하위 명령 옵션	
옵션	*설명*
-u, **--image-url <url>**	이미지 URL을 수동으로 지정합니다.
-f, **--image-file <path>**	로컬 이미지 파일을 수동으로 지정합니다. 디버깅에 사용됩니다.
-i, --ignition-file <path>	파일의 Ignition 구성을 삽입합니다.
-I, **--ignition-url <URL>**	URL의 Ignition 구성을 삽입합니다.
--ignition-hash <digest>	Ignition 구성의 **type-value**를 요약합니다.
-p, **--platform <name>**	설치된 시스템의 Ignition 플랫폼 ID를 재정의합니다.
--append-karg <arg>…	설치된 시스템에 기본 커널 인수를 추가합니다.
--delete-karg <arg>…	설치된 시스템에서 기본 커널 인수를 삭제합니다.

-n, --copy-network	설치 환경의 네트워크 구성을 복사합니다.  **중요** **copy-network** 옵션은 **/etc/NetworkManager/system-connections**에 있는 네트워킹 구성만 복사합니다. 특히 시스템 호스트 이름을 복사하지 않습니다.
--network-dir <path>	**-n**과 함께 사용됩니다. 기본값은 **/etc/NetworkManager/system-connections/**입니다.
--save-partlabel <lx>..	이 레이블 glob로 파티션을 저장합니다.
--save-partindex <id>…	이 번호 또는 범위로 파티션을 저장합니다.
--insecure	서명 확인을 건너뜁니다.
--insecure-ignition	HTTPS 또는 해시 없는 Ignition URL을 허용합니다.
--architecture <name>	대상 CPU 아키텍처입니다. 기본값은 **x86_64**입니다.
--preserve-on-error	오류 발생한 파티션 테이블을 지우지 않습니다.
-h, --help	도움말 정보를 출력합니다.
coreos-install 설치 하위 명령 인수	
인수	***설명***
<device>	대상 장치입니다.
coreos-installer ISO Ignition 하위 명령	
하위 명령	***설명***
$ coreos-installer iso ignition embed <options> --ignition-file <file_path> <ISO_image>	ISO 이미지에 Ignition 구성를 삽입합니다.
coreos-installer iso ignition show <options> <ISO_image>	ISO 이미지에 삽입된 Ignition 구성를 표시합니다.

coreos-installer iso ignition remove <options> <ISO_image>	ISO 이미지에서 삽입된 Ignition 구성를 제거합니다.
coreos-installer ISO Ignition 하위 명령 옵션	
옵션	*설명*
-f, --force	기존 Ignition 구성를 덮어씁니다.
-i, --ignition-file <path>	사용할 Ignition 구성입니다. 기본값은 **stdin**입니다.
-o, --output <path>	새 출력 파일에 ISO를 씁니다.
-h, --help	도움말 정보를 출력합니다.
coreos-installer PXE Ignition 하위 명령	
하위 명령	*설명*
이러한 모든 옵션이 모든 하위 명령에서 허용되지는 않습니다.	
coreos-installer pxe ignition wrap <options>	Ignition 구성을 이미지로 래핑합니다.
coreos-installer pxe ignition unwrap <options> <image_name>	이미지에 래핑된 Ignition 구성를 표시합니다.
coreos-installer PXE Ignition 하위 명령 옵션	
옵션	*설명*
이러한 모든 옵션이 모든 하위 명령에서 허용되지는 않습니다.	
-i, --ignition-file <path>	사용할 Ignition 구성입니다. 기본값은 **stdin**입니다.
-o, --output <path>	새 출력 파일에 ISO를 씁니다.
-h, --help	도움말 정보를 출력합니다.

8.2.11.3.4.3. ISO 또는 PXE 설치를 위한 coreos.inst 부팅 옵션

coreos.inst 부팅 인수를 RHCOS 라이브 설치 프로그램에 전달하여 부팅 시 coreos-installer 옵션을 자동으로 호출할 수 있습니다. 이러한 매개 변수는 표준 부팅 인수 외에 제공됩니다.

- **ISO 설치의 경우 부트 로더 메뉴에서 자동 부팅을 중단하여 coreos.inst 옵션을 추가할 수 있습니다. RHEL CoreOS (Live) 메뉴 옵션이 강조 표시된 상태에서 TAB을 눌러 자동 부팅을 중**

단할 수 있습니다.

- PXE 또는 iPXE 설치의 경우 RHCOS 라이브 설치 프로그램을 부팅하기 전에 coreos.inst 옵션을 APPEND 줄에 추가해야 합니다.

다음 표는 ISO 및 PXE 설치를 위한 RHCOS 라이브 설치 관리자 coreos.inst 부팅 옵션을 보여줍니다.

표 8.13. coreos.inst 부팅 옵션

인수	설명
coreos.inst.install_dev	필수 항목입니다. 설치할 시스템의 블록 장치입니다. **sda**가 허용되더라도 전체 경로 (예:**/dev/sda**)를 사용하는 것이 좋습니다.
coreos.inst.ignition_url	선택사항: 설치된 시스템에 삽입할 Ignition 구성의 URL입니다. URL을 지정하지 않으면 Ignition 구성이 포함되지 않습니다. HTTP 및 HTTPS 프로토콜만 지원됩니다.
coreos.inst.save_partlabel	선택사항: 설치 중에 보존 할 파티션의 쉼표로 구분된 레이블입니다. Glob 스타일 와일드카드가 허용됩니다. 지정된 파티션이 존재할 필요는 없습니다.
coreos.inst.save_partindex	선택사항: 설치 도중 보존할 파티션 인덱스들입니다(쉼표로 구분됨). **m-n** 범위가 허용되며 **m** 또는 **n**은 생략할 수 있습니다. 지정된 파티션이 존재할 필요는 없습니다.
coreos.inst.insecure	선택사항: **coreos.inst.image_url**로 지정된 OS 이미지의 서명되지 않은 상태를 허용합니다.

인수	설명
coreos.inst.image_url	선택사항: 지정된 RHCOS 이미지를 다운로드하여 설치합니다. ● 이 인수는 프로덕션 환경에서 사용할 수 없으며, 디버깅 용도로만 사용됩니다. ● 이 인수를 사용하면 라이브 미디어와 일치하지 않는 RHCOS 버전을 설치할 수 있지만, 설치하려는 버전과 일치하는 미디어를 사용하는 것이 좋습니다. ● **coreos.inst.image_url**을 사용하는 경우 **coreos.inst.insecure**도 사용해야 합니다. 베어메탈 미디어가 OpenShift Container Platform용으로 GPG 서명되지 않았기 때문입니다. ● HTTP 및 HTTPS 프로토콜만 지원됩니다.
coreos.inst.skip_reboot	선택사항: 설치 후 시스템을 재부팅하지 않습니다. 설치가 완료되면 설치 과정에서 발생되는 상황을 검사할 수 있는 프롬프트가 표시됩니다. 이 인수는 프로덕션 환경에서 사용할 수 없으며, 디버깅 용도로만 사용됩니다.
coreos.inst.platform_id	선택사항: RHCOS 이미지가 설치되고 있는 플랫폼의 Ignition 플랫폼 ID입니다. 기본값은 **metal**입니다. 이 옵션에 따라 VMware와 같은 클라우드 공급자의 Ignition 구성를 요청할지 여부가 결정됩니다. 예: **coreos.inst.platform_id=vmware**.
ignition.config.url	선택사항: 라이브 부팅을 위한 Ignition 구성의 URL입니다. 예를 들어 **coreos-installer**가 호출되는 방식을 사용자 지정하거나 설치 전과 후에 코드를 실행하는 데 사용할 수 있습니다. 이 URL은 설치된 시스템의 Ignition 구성인 **coreos.inst.ignition_url**과 다릅니다.

8.2.11.4. RHCOS에서 커널 인수로 다중 경로 활성화

RHCOS는 이제 기본 디스크에서 멀티패스를 지원하므로 하드웨어 장애에 대한 탄력성이 강화된 호스트 가용성을 높일 수 있습니다.

OpenShift Container Platform 4.8 이상에서 프로비저닝된 노드의 설치 시 멀티패스를 활성화할 수 있습니다. 시스템 구성을 통해 멀티패스를 활성화하면 설치 후 지원을 사용할 수 있지만 설치 중에 멀피태스를 활성화하는 것이 좋습니다.

I/O에서 최적화된 경로로 인해 I/O 시스템 오류가 발생하는 설정에서 설치 시 멀티패스를 활성화해야 합니다.

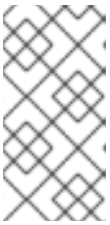
중요

IBM Z 및 **LinuxONE**에서는 설치 중에 클러스터를 구성하는 경우에만 다중 경로를 활성화할 수 있습니다. 자세한 내용은 *IBM Z* 및 *LinuxONE*에 *z/VM*으로 *클러스터 설치*의 **"RHCOS** 설치 및 **OpenShift Container Platform** 부트스트랩 프로세스 시작"을 참조하십시오.

다음 절차에서는 설치 시 멀티패스를 활성화하고 커널 인수를 **coreos-installer install** 명령에 추가하여 설치된 시스템 자체에서 첫 번째 부팅부터 시작된 멀티패스를 사용하도록 합니다.

사전 요구 사항

- 버전 **4.8** 이상을 사용하는 실행 중인 **OpenShift Container Platform** 클러스터가 있어야 합니다.



참고

OpenShift Container Platform은 **4.6** 또는 이전 버전에서 업그레이드된 노드에서 **Day-2** 활동으로 다중 경로를 활성화할 수 없습니다.

- 관리 권한이 있는 사용자로 클러스터에 로그인했습니다.

프로세스

1. 멀티패스를 활성화하고 **multipathd** 데몬을 시작하려면 다음 명령을 실행합니다.

```
$ mpathconf --enable && systemctl start multipathd.service
```

- 선택 사항: **PXE** 또는 **ISO**를 부팅하는 경우 커널 명령줄에서 **rd.multipath=default**를 추가하여 멀티패스를 활성화할 수 있습니다.

2. **coreos-installer** 프로그램을 호출하여 커널 인수를 추가합니다.

- 시스템에 연결된 멀티패스 장치가 하나뿐인 경우 경로 **/dev/mapper/mpatha**에서 사용할 수 있어야 합니다. 예를 들면 다음과 같습니다.

```
$ coreos-installer install /dev/mapper/mpatha \ 1
--append-karg rd.multipath=default \
--append-karg root=/dev/disk/by-label/dm-mpath-root \
--append-karg rw
```

1

단일 멀티패스 장치의 경로를 나타냅니다.

- 시스템에 연결된 멀티패스 장치가 여러 개 있는 경우 보다 명확하게 하려면 /dev/mapper/mpatha를 사용하는 대신 /dev/disk/by-id에서 사용할 수 있는 WWN(World Wide Name) 심볼릭 링크를 사용하는 것이 좋습니다. 예를 들면 다음과 같습니다.

```
$ coreos-installer install /dev/disk/by-id/wwn-<wwn_ID> \ 1
--append-karg rd.multipath=default \
--append-karg root=/dev/disk/by-label/dm-mpath-root \
--append-karg rw
```

1

멀티패스 대상 장치의 WWN ID를 나타냅니다. 예를 들면 0xx194e957fcedb4841입니다.

이 심볼릭 링크는 라이브 설치 프로그램을 지시하기 위해 특수 coreos.inst.** 인수를 사용할 때coreos.inst.install_dev 커널 인수로 사용될 수도 있습니다. 자세한 내용은 "RHCOS 설치 및 OpenShift Container Platform 부트스트랩 프로세스 시작"을 참조하십시오.

3. 작업자 노드 중 하나로 이동하고 커널 명령줄 인수 (호스트의 /proc/cmdline)를 나열하여 커널 인수가 작동하는지 확인합니다.

```
$ oc debug node/ip-10-0-141-105.ec2.internal
```

출력 예

```
Starting pod/ip-10-0-141-105ec2internal-debug ...
To use host binaries, run `chroot /host`

sh-4.2# cat /host/proc/cmdline
...
```

```
rd.multipath=default root=/dev/disk/by-label/dm-mpath-root
...

sh-4.2# exit
```

추가된 커널 인수가 표시되어야 합니다.

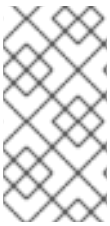
추가 리소스

- 특정 **coreos.inst.*** 인수를 사용하여 라이브 설치 프로그램을 보내는 방법에 대한 자세한 내용은 **RHCOS** 설치 프로그램 설치 및 **OpenShift Container Platform** 부트스트랩 프로세스 시작을 참조하십시오.

8.2.11.5. bootupd를 사용하여 부트로더 업데이트

bootupd를 사용하여 부트로더를 업데이트하려면 **RHCOS** 머신에 수동으로 **bootupd**를 설치하거나 **systemd** 유닛이 활성화된 머신 구성을 제공해야 합니다. **grubby** 또는 기타 부트로더 툴과는 달리 **bootupd**는 커널 인수 전달과 같은 커널 공간 구성을 관리하지 않습니다.

bootupd를 설치한 후 **OpenShift Container Platform** 클러스터에서 원격으로 관리할 수 있습니다.



참고

bootupd는 **BootHole** 취약점에 대한 보호와 같이 베어 메탈 또는 가상화 하이퍼바이저 설치에서만 사용하는 것이 좋습니다.

수동 설치 방법

bootctl 명령줄 툴을 사용하여 **bootupd**를 수동으로 설치할 수 있습니다.

1. 시스템 상태를 검사합니다.

```
# bootupctl status
```

출력 예

```
Component EFI
 Installed: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
 Update: At latest version
```

2. bootupd를 설치하지 않고 생성된 RHCOS 이미지에는 명시적 채택 단계가 필요합니다.

   시스템 상태가 Adoptable인 경우 채택을 수행합니다.

```
# bootupctl adopt-and-update
```

출력 예

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

3. 업데이트를 사용할 수 있는 경우 다음 재부팅에 변경 사항이 적용되도록 업데이트를 적용합니다.

```
# bootupctl update
```

출력 예

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

시스템 구성 방법

bootupd를 활성화하는 또 다른 방법은 머신 구성을 제공하는 것입니다.

- 다음 예와 같이 **systemd** 단위가 활성화된 머신 구성 파일을 제공합니다.

출력 예

```
variant: rhcos
version: 1.1.0
systemd:
  units:
    - name: custom-bootupd-auto.service
      enabled: true
      contents: |
        [Unit]
        Description=Bootupd automatic update

        [Service]
        ExecStart=/usr/bin/bootupctl update
        RemainAfterExit=yes

        [Install]
        WantedBy=multi-user.target
```

8.2.12. 부트스트랩 프로세스가 완료될 때까지 대기 중

OpenShift Container Platform 부트스트랩 프로세스는 클러스터 노드가 먼저 디스크에 설치된 영구 **RHCOS** 환경으로 부팅된 후에 시작됩니다. **Ignition** 구성 파일을 통해 제공되는 구성 정보는 부트스트랩 프로세스를 초기화하고 머신에 **OpenShift Container Platform**을 설치하는 데 사용됩니다. 부트스트랩 프로세스가 완료될 때까지 기다려야 합니다.

사전 요구 사항

- 클러스터에 대한 **Ignition** 구성 파일이 생성되어 있습니다.
- 적합한 네트워크, **DNS** 및 로드 밸런싱 인프라가 구성되어 있습니다.
- 설치 프로그램을 받아서 클러스터의 **Ignition** 구성 파일을 생성했습니다.
- 클러스터 머신에 **RHCOS**를 설치하고 **OpenShift Container Platform** 설치 프로그램에서 생성된 **Ignition** 구성 파일을 제공했습니다.

- 사용자 시스템에서 직접 인터넷에 액세스하거나 **HTTP** 또는 **HTTPS** 프록시를 사용할 수 있습니다.

절차

1. 부트스트랩 프로세스를 모니터링합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
  --log-level=info 2
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

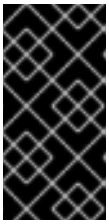
2 다른 설치 세부 사항을 보려면 **info** 대신 **warn, debug** 또는 **error**를 지정합니다.

출력 예

```
INFO Waiting up to 30m0s for the Kubernetes API at
https://api.test.example.com:6443...
INFO API v1.22.1 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

이 명령은 **Kubernetes API** 서버가 컨트롤 플레인 시스템에서 부트스트랩되었다는 신호를 보낼 때 성공합니다.

2. 부트스트랩 프로세스가 완료된 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다.



중요

이 시점에 로드 밸런서에서 부트스트랩 시스템을 제거해야 합니다. 부트스트랩 머신 자체를 제거하거나 다시 포맷할 수도 있습니다.

추가 리소스

- 설치 문제가 발생하는 경우 설치 로그 모니터링 및 진단 데이터 검색에 대한 자세한 내용은 설치 진행 상황 모니터링을 참조하십시오.

8.2.13. CLI를 사용하여 클러스터에 로그인

클러스터 kubeconfig 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. kubeconfig 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- oc CLI를 설치했습니다.

절차

1. kubeadmin 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 oc 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

8.2.14. 머신의 인증서 서명 요청 승인

클러스터에 시스템을 추가하면 추가한 시스템별로 보류 중인 인증서 서명 요청(CSR)이 두 개씩 생성됩니다. 이러한 CSR이 승인되었는지 확인해야 하며, 필요한 경우 이를 직접 승인해야 합니다. 클라이언트 요청을 먼저 승인한 다음 서버 요청을 승인해야 합니다.

사전 요구 사항

- 클러스터에 시스템을 추가했습니다.

절차

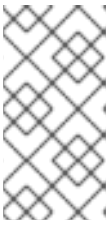
1. 클러스터가 시스템을 인식하는지 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0  Ready    master  63m  v1.22.1
master-1  Ready    master  63m  v1.22.1
master-2  Ready    master  64m  v1.22.1
```

출력에 생성된 모든 시스템이 나열됩니다.



참고

이전 출력에는 일부 CSR이 승인될 때까지 컴퓨팅 노드(작업자 노드라고도 함)가 포함되지 않을 수 있습니다.

2.

보류 중인 CSR을 검토하고 클러스터에 추가한 각 시스템에 대해 Pending 또는 Approved 상태의 클라이언트 및 서버 요청이 표시되는지 확인합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE    REQUESTOR                                                    CONDITION
csr-8b2br   15m    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

예에서는 두 시스템이 클러스터에 참여하고 있습니다. 목록에는 승인된 CSR이 더 많이 나타날 수도 있습니다.

3.

CSR이 승인되지 않은 경우, 추가된 시스템에 대한 모든 보류 중인 CSR이 Pending 상태로 전환된 후 클러스터 시스템의 CSR을 승인합니다.

참고

CSR은 교체 주기가 자동으로 만료되므로 클러스터에 시스템을 추가한 후 1 시간 이내에 CSR을 승인하십시오. 한 시간 내에 승인하지 않으면 인증서가 교체되고 각 노드에 대해 두 개 이상의 인증서가 표시됩니다. 이러한 인증서를 모두 승인해야 합니다. 클라이언트 CSR이 승인되면 Kubelet은 인증서에 대한 보조 CSR을 생성하므로 수동 승인이 필요합니다. 그러면 Kubelet에서 동일한 매개변수를 사용하여 새 인증서를 요청하는 경우 인증서 갱신 요청은 machine-approver에 의해 자동으로 승인됩니다.

참고

베어 메탈 및 기타 사용자 프로비저닝 인프라와 같이 머신 API를 사용하도록 활성화되지 않는 플랫폼에서 실행되는 클러스터의 경우 CSR(Kubelet service Certificate Request)을 자동으로 승인하는 방법을 구현해야 합니다. 요청이 승인되지 않으면 API 서버가 kubelet에 연결될 때 서비스 인증서가 필요하므로 oc exec, oc rsh, oc logs 명령을 성공적으로 수행할 수 없습니다. Kubelet 엔드 포인트에 연결하는 모든 작업을 수행하려면 이 인증서 승인이 필요합니다. 이 방법은 새 CSR을 감시하고 CSR이 system:node 또는 system:admin 그룹의 node-bootstrapper 서비스 계정에 의해 제출되었는지 확인하고 노드의 ID를 확인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 <csr_name>은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

참고

일부 Operator는 일부 CSR이 승인될 때까지 사용할 수 없습니다.

4. 이제 클라이언트 요청이 승인되었으므로 클러스터에 추가한 각 머신의 서버 요청을 검토해야 합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE    REQUESTOR                                                   CONDITION
csr-bfd72   5m26s  system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
```

```
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 나머지 CSR이 승인되지 않고 Pending 상태인 경우 클러스터 머신의 CSR을 승인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 <csr_name>은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs oc adm certificate approve
```

6. 모든 클라이언트 및 서버 CSR이 승인된 후 머신은 Ready 상태가 됩니다. 다음 명령을 실행하여 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0  Ready    master  73m  v1.22.1
master-1  Ready    master  73m  v1.22.1
master-2  Ready    master  74m  v1.22.1
worker-0  Ready    worker  11m  v1.22.1
worker-1  Ready    worker  11m  v1.22.1
```

참고

머신이 **Ready** 상태로 전환하는 데 서버 **CSR**의 승인 후 몇 분이 걸릴 수 있습니다.

추가 정보

- **CSR**에 대한 자세한 내용은 인증서 서명 요청을 참조하십시오.

8.2.15. Operator의 초기 설정

컨트롤 플레인이 초기화된 후 일부 **Operator**를 즉시 구성하여 모두 사용 가능하도록 해야 합니다.

사전 요구 사항

- 컨트롤 플레인이 초기화되어 있습니다.

절차

1. 클러스터 구성 요소가 온라인 상태인지 확인합니다.

```
$ watch -n5 oc get clusteroperators
```

출력 예

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.9.0     True        False         False      19m
baremetal                                  4.9.0     True        False         False      37m
cloud-credential                           4.9.0     True        False         False      40m
cluster-autoscaler                         4.9.0     True        False         False      37m
config-operator                            4.9.0     True        False         False      38m
console                                    4.9.0     True        False         False      26m
csi-snapshot-controller                    4.9.0     True        False         False      37m
dns                                        4.9.0     True        False         False      37m
etcd                                       4.9.0     True        False         False      36m
image-registry                             4.9.0     True        False         False      31m
ingress                                    4.9.0     True        False         False      30m
insights                                   4.9.0     True        False         False      31m
kube-apiserver                             4.9.0     True        False         False      26m
kube-controller-manager                    4.9.0     True        False         False      36m
kube-scheduler                             4.9.0     True        False         False      36m
```

```
kube-storage-version-migrator              4.9.0     True        False         False      37m
machine-api                                4.9.0     True        False         False      29m
machine-approver                           4.9.0     True        False         False      37m
machine-config                             4.9.0     True        False         False      36m
marketplace                                4.9.0     True        False         False      37m
monitoring                                 4.9.0     True        False         False      29m
network                                    4.9.0     True        False         False      38m
node-tuning                                4.9.0     True        False         False      37m
openshift-apiserver                        4.9.0     True        False         False      32m
openshift-controller-manager               4.9.0     True        False         False      30m
openshift-samples                          4.9.0     True        False         False      32m
operator-lifecycle-manager                 4.9.0     True        False         False      37m
operator-lifecycle-manager-catalog         4.9.0     True        False         False      37m
operator-lifecycle-manager-packageserver   4.9.0     True        False         False      32m
service-ca                                 4.9.0     True        False         False      38m
storage                                    4.9.0     True        False         False      37m
```

2. 사용할 수 없는 **Operator**를 구성합니다.

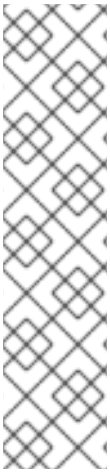
추가 리소스

- 실패한 **OpenShift Container Platform** 설치 시 데이터를 수집하는 방법에 대한 자세한 내용은 설치 실패에서 로그 수집을 참조하십시오.
- 클러스터 전체에서 **Operator Pod** 상태를 확인하고 진단을 위해 **Operator** 로그를 수집하는 단계는 **Operator** 문제 해결을 참조하십시오.

8.2.15.1. 설치 중 제거된 이미지 레지스트리

공유 가능한 개체 스토리지를 제공하지 않는 플랫폼에서 **OpenShift Image Registry Operator**는 자체적으로 **Removed**로 부트스트랩합니다. 이를 통해 **openshift-installer**가 이러한 플랫폼 유형에서 설치를 완료할 수 있습니다.

설치 후 **managementState**를 **Removed**에서 **Managed**로 전환하도록 **Image Registry Operator** 구성을 편집해야 합니다.



참고

Prometheus 콘솔은 **ImageRegistryRemoved** 경고를 제공합니다. 다음은 예시 경고입니다.

"이미지 레지스트리가 제거되었습니다. **ImageStreamTags**를 참조하는 **ImageStreamTags, BuildConfigs** 및 **DeploymentConfigs**가 예상대로 작동하지 않을 수 있습니다. 스토리지를 구성하고 **configs.imageregistry.operator.openshift.io**를 편집하여 구성을 **Managed** 상태로 업데이트하십시오."

8.2.15.2. 이미지 레지스트리 스토리지 구성

기본 스토리지를 제공하지 않는 플랫폼에서는 처음에 **Image Registry Operator**를 사용할 수 없습니다. 설치한 후에 스토리지를 사용하도록 레지스트리를 구성하여 **Registry Operator**를 사용 가능하도록 만들어야 합니다.

프로덕션 클러스터에 필요한 영구 볼륨을 구성하는 과정의 지침이 표시됩니다. 해당하는 경우, 프로덕션 환경 외 클러스터에서만 사용할 수 있는 저장 위치로서 빈 디렉터리를 구성하는 과정의 지침이 표시됩니다.

업그레이드 중에 **Recreate** 롤아웃 전략을 사용하여 이미지 레지스트리의 블록 스토리지 유형 사용을 허용하기 위한 추가 지침이 제공됩니다.

8.2.15.2.1. 베어메탈 및 기타 수동 설치를 위한 레지스트리 스토리지 구성

클러스터 관리자는 설치한 후 스토리지를 사용하도록 레지스트리를 구성해야 합니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- 베어 메탈과 같이 수동으로 프로비저닝된 **RHCOS(Red Hat Enterprise Linux CoreOS)** 노드를 사용하는 클러스터가 있어야 합니다.
- **Red Hat OpenShift Container Storage**와 같이 클러스터용 영구 스토리지 프로비저닝이 있습니다.

중요

OpenShift Container Platform은 복제본이 하나만 있는 경우 이미지 레지스트리 스토리지에 대한 **ReadWriteOnce** 액세스를 지원합니다. **ReadWriteOnce** 액세스에서는 레지스트리가 **Recreate** 롤아웃 전략을 사용해야 합니다. 두 개 이상의 복제본으로 고 가용성을 지원하는 이미지 레지스트리를 배포하려면 **ReadWriteMany** 액세스가 필요합니다.

- **"100Gi"** 용량이 필요합니다.

프로세스

1. 스토리지를 사용하도록 레지스트리를 구성하기 위해 **configs.imageregistry/cluster** 리소스에서 **spec.storage.pvc**를 변경합니다.

참고

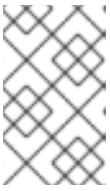
공유 스토리지를 사용할 때 보안 설정을 확인하여 외부에서의 액세스를 방지합니다.

2. 레지스트리 **pod**가 없는지 확인합니다.

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

출력 예

```
No resourses found in openshift-image-registry namespace
```



참고

출력에 레지스트리 **Pod**가 있는 경우 이 절차를 계속할 필요가 없습니다.

3. 레지스트리 구성을 확인합니다.

```
$ oc edit configs.imageregistry.operator.openshift.io
```

출력 예

```
storage:
  pvc:
    claim:
```

image-registry-storage PVC의 자동 생성을 허용하도록 claim 필드를 비워 둡니다.

4. clusteroperator 상태를 확인합니다.

```
$ oc get clusteroperator image-registry
```

출력 예

```
NAME              VERSION   AVAILABLE   PROGRESSING   DEGRADED   SINCE   MESSAGE
image-registry    4.9       True        False         False      6h50m
```

5. 이미지를 빌드 및 푸시할 수 있도록 레지스트리의 관리가 설정되어 있는지 확인하십시오.

- 다음을 실행합니다.

```
$ oc edit configs.imageregistry/cluster
```

다음으로 라인을 변경하십시오.

-

```
managementState: Removed
```

다음으로 변경

```
managementState: Managed
```

8.2.15.2.2. 프로덕션 환경 외 클러스터에서 이미지 레지스트리의 스토리지 구성

이미지 레지스트리 **Operator**에 대한 스토리지를 구성해야 합니다. 프로덕션 환경 외 클러스터의 경우, 이미지 레지스트리를 빈 디렉터리로 설정할 수 있습니다. 이렇게 하는 경우 레지스트리를 다시 시작하면 모든 이미지가 손실됩니다.

프로세스

- 이미지 레지스트리 스토리지를 빈 디렉터리로 설정하려면 다음을 수행하십시오.

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch
'{"spec":{"storage":{"emptyDir":{}}}}'
```

주의

프로덕션 환경 외 클러스터에 대해서만 이 옵션을 구성하십시오.

Image Registry Operator가 구성 요소를 초기화하기 전에 이 명령을 실행하면 **oc patch** 명령이 실패하며 다음 오류가 발생합니다.

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster"
not found
```

몇 분 후에 명령을 다시 실행하십시오.

8.2.15.2.3. 블록 레지스트리 스토리지 구성

클러스터 관리자로서 업그레이드 중에 이미지 레지스트리가 블록 스토리지 유형을 사용할 수 있도록 허용하기 위해 **Recreate** 롤아웃 전략을 사용할 수 있습니다.

중요

블록 스토리지 볼륨이 지원되지만 프로덕션 클러스터에서 이미지 레지스트리와 함께 사용하는 것은 권장되지 않습니다. 레지스트리가 블록 스토리지에 구성된 설치는 레지스트리가 둘 이상의 복제본을 가질 수 없기 때문에 가용성이 높지 않습니다.

프로세스

1. 이미지 레지스트리 스토리지를 블록 스토리지 유형으로 설정하려면 레지스트리가 **Recreate** 롤아웃 전략을 사용하고 하나의 **(1)** 복제본에서만 실행되도록 레지스트리를 패치합니다.

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. 블록 스토리지 장치에 **PV**를 프로비저닝하고 해당 볼륨의 **PVC**를 생성합니다. 요청된 블록 볼륨은 **RWO(ReadWriteOnce)** 액세스 모드를 사용합니다.

3. 올바른 **PVC**를 참조하도록 레지스트리 구성을 편집합니다.

8.2.16. 사용자 프로비저닝 인프라에 설치 완료

Operator 구성을 완료한 후 제공하는 인프라에 클러스터 설치를 완료할 수 있습니다.

사전 요구 사항

- 컨트롤 플레인이 초기화되어 있습니다.
- 초기 **Operator** 구성을 완료해야 합니다.

프로세스

1. 다음 명령을 사용하여 모든 클러스터 구성 요소가 온라인 상태인지 확인합니다.

```
$ watch -n5 oc get clusteroperators
```

출력 예

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.9.0     True        False         False      19m
baremetal                                  4.9.0     True        False         False      37m
cloud-credential                           4.9.0     True        False         False      40m
cluster-autoscaler                         4.9.0     True        False         False      37m
config-operator                            4.9.0     True        False         False      38m
console                                    4.9.0     True        False         False      26m
csi-snapshot-controller                    4.9.0     True        False         False      37m
dns                                        4.9.0     True        False         False      37m
etcd                                       4.9.0     True        False         False      36m
image-registry                             4.9.0     True        False         False      31m
ingress                                    4.9.0     True        False         False      30m
insights                                   4.9.0     True        False         False      31m
kube-apiserver                             4.9.0     True        False         False      26m
kube-controller-manager                    4.9.0     True        False         False      36m
kube-scheduler                             4.9.0     True        False         False      36m
kube-storage-version-migrator              4.9.0     True        False         False      37m
machine-api                                4.9.0     True        False         False      29m
machine-approver                           4.9.0     True        False         False      37m
machine-config                             4.9.0     True        False         False      36m
marketplace                                4.9.0     True        False         False      37m
monitoring                                 4.9.0     True        False         False      29m
network                                    4.9.0     True        False         False      38m
node-tuning                                4.9.0     True        False         False      37m
openshift-apiserver                        4.9.0     True        False         False      32m
openshift-controller-manager               4.9.0     True        False         False      30m
openshift-samples                          4.9.0     True        False         False      32m
operator-lifecycle-manager                 4.9.0     True        False         False      37m
operator-lifecycle-manager-catalog         4.9.0     True        False         False      37m
operator-lifecycle-manager-packageserver   4.9.0     True        False         False      32m
service-ca                                 4.9.0     True        False         False      38m
storage                                    4.9.0     True        False         False      37m
```

또는 다음 명령은 모든 클러스터를 사용할 수 있을 때 알립니다. 또한 인증 정보를 검색하고 표시합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

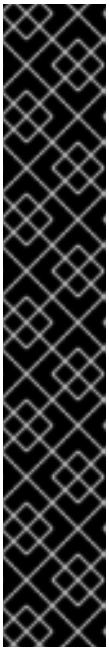
1

<installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

출력 예

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator가 Kubernetes API 서버에서 OpenShift Container Platform 클러스터 배포를 완료하면 명령이 성공합니다.



중요

- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 node-bootstrapper 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.

- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

2. Kubernetes API 서버가 Pod와 통신하고 있는지 확인합니다.

   a. 모든 Pod 목록을 보려면 다음 명령을 사용하십시오.

```
$ oc get pods --all-namespaces
```

출력 예

```
NAMESPACE                       NAME                                        READY   STATUS
RESTARTS   AGE
openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8
1/1     Running   1          9m
openshift-apiserver             apiserver-67b9g                             1/1     Running   0
```

```
3m
openshift-apiserver            apiserver-ljcmx                         1/1     Running     0
1m
openshift-apiserver            apiserver-z25h4                         1/1     Running     0
2m
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8
1/1     Running     0          5m
...
```

b. 다음 명령을 사용하여 이전 명령의 출력에 나열된 **Pod**의 로그를 표시합니다.

```
$ oc logs <pod_name> -n <namespace> 1
```

1 이전 명령의 출력에 표시된 대로 **Pod** 이름과 네임스페이스를 지정합니다.

Pod 로그가 표시되면 **Kubernetes API** 서버는 클러스터 시스템과 통신할 수 있습니다.

3. **FCP(Fibre Channel Protocol)**를 사용하는 설치에는 다중 경로를 활성화하기 위해 추가 단계가 필요합니다. 설치 중에 멀티패스를 활성화하지 마십시오.

자세한 내용은 *설치 후 머신 구성 작업* 설명서에서 **"RHCOS에서 커널 인수를 사용하여 멀티패스 활성화"**를 참조하십시오.

8.2.17. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 **Telemetry** 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 **Telemetry**가 자동으로 실행되고 **OpenShift Cluster Manager** 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 **OpenShift Cluster Manager**를 사용하여 자동으로 또는 **OpenShift Cluster Manager**를 사용하여 수동으로 유지 관리되는지 확인한 후 **subscription watch**를 사용하여 계정 또는 다중 클러스터 수준에서 **OpenShift Container Platform** 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

8.2.18. 다음 단계

- 설치를 확인합니다.
- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.
- 레지스트리를 설정하고 레지스트리 스토리지를 구성합니다.

8.3. 네트워크 사용자 지정이 포함된 사용자 프로비저닝 베어 메탈 클러스터를 설치

OpenShift Container Platform 4.9에서는 사용자 지정된 네트워크 구성 옵션으로 프로비저닝하는 클러스터를 베어메탈 인프라에 설치할 수 있습니다. 네트워크 구성을 사용자 지정할 경우, 클러스터가 사용자 환경의 기존 IP 주소 할당과 공존하고 기존 MTU 및 VXLAN 구성과 통합될 수 있습니다.

OpenShift Container Platform 네트워킹을 사용자 지정할 때 설치 중에 대부분의 네트워크 구성 매개변수를 설정해야 합니다. 실행중인 클러스터에서 kubeProxy 네트워크 구성 매개변수만 수정할 수 있습니다.

8.3.1. 사전 요구 사항

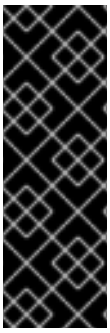
- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.
- 방화벽을 사용하며 Telemetry 서비스를 사용할 예정인 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성해야 합니다.

8.3.2. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 **Telemetry** 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.



중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

추가 리소스

- 프로비저닝하는 베어 메탈 인프라에서 제한된 네트워크 설치를 수행하는 방법에 대한 자세한 내용은 제한된 네트워크에 사용자 프로비저닝 베어 메탈 클러스터 설치를 참조하십시오.

8.3.3. 사용자 프로비저닝 인프라를 포함한 클러스터의 시스템 요구사항

사용자 프로비저닝 인프라가 포함된 클러스터의 경우, 필요한 모든 시스템을 배포해야 합니다.

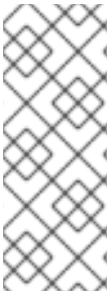
이 섹션에서는 사용자 프로비저닝 인프라에 **OpenShift Container Platform**을 배포해야 하는 요구 사항에 대해 설명합니다.

8.3.3.1. 클러스터 설치에 필요한 시스템

최소 **OpenShift Container Platform** 클러스터에 다음과 같은 호스트가 필요합니다.

표 8.14. 최소 필수 호스트

호스트	설명
임시 부트스트랩 시스템 한 개	컨트롤 플레인 시스템 세 개에 OpenShift Container Platform 클러스터를 배포하기 위한 부트스트랩 시스템이 클러스터에 필요합니다. 클러스터를 설치한 후 부트스트랩 시스템을 제거할 수 있습니다.
컨트롤 플레인 시스템 세 개	컨트롤 플레인 시스템은 컨트롤 플레인을 구성하는 Kubernetes 및 OpenShift Container Platform 서비스를 실행합니다.
두 개 이상의 컴퓨팅 시스템(작업자 시스템이라고도 함).	OpenShift Container Platform 사용자가 요청한 워크로드는 컴퓨팅 머신에서 실행됩니다.



참고

예외적으로 세 개의 컨트롤 플레인 시스템으로 구성된 베어 메탈 클러스터에 제로 컴퓨팅 머신을 실행할 수 있습니다. 이를 통해 클러스터 관리자와 개발자들이 테스트, 개발, 프로덕션에 사용할 수 있는 소형화되고 리소스 효율이 높은 클러스터를 제공합니다. 컴퓨팅 머신 하나를 실행하는 것은 지원되지 않습니다.

중요

클러스터의 고가용성을 유지하려면 이러한 클러스터 시스템에 대해 별도의 물리적 호스트를 사용하십시오.

부트스트랩, 컨트롤 플레인 시스템은 운영 체제로 RHCOS (Red Hat Enterprise Linux CoreOS)를 사용해야 합니다. 그러나 컴퓨팅 머신은 RHCOS(Red Hat Enterprise Linux CoreOS), RHEL (Red Hat Enterprise Linux) 7.9 또는 RHEL 8.4 중에서 선택할 수 있습니다.

RHCOS는 RHEL 8(Red Hat Enterprise Linux)을 기반으로하며 모든 하드웨어 인증 및 요구사항을 모두 이어받습니다. Red Hat Enterprise Linux 기술 기능 및 제한을 참조하십시오.

8.3.3.2. 클러스터 설치를 위한 최소 리소스 요구 사항

각 클러스터 시스템이 다음과 같은 최소 요구사항을 충족해야 합니다.

표 8.15. 최소 리소스 요구사항

시스템	운영 체제	CPU [1]	RAM	스토리지	IOPS [2]
부트스트랩	RHCOS	4	16GB	100GB	300
컨트롤 플레인	RHCOS	4	16GB	100GB	300
컴퓨팅	RHCOS, RHEL 7.9 또는 RHEL 8.4 [3]	2	8GB	100GB	300

1. **SMT(동시 멀티스레딩) 또는 하이퍼 스레딩이 활성화되지 않은 경우 하나의 CPU는 하나의 실제 코어에 해당합니다. 활성화하면 다음과 같은 공식을 사용하여 해당 비율을 계산합니다. (코어 당 스레드 수). (코어 당 스레드) 소켓s = CPU.**

2. **OpenShift Container Platform 및 Kubernetes는 디스크 성능에 민감하며 특히 10ms p99 fsync 기간이 필요한 컨트롤 플레인 노드의 etcd에 더 빠른 스토리지가 권장됩니다. 많은 클라우드 플랫폼에서 스토리지 크기와 IOPS를 함께 확장되므로 충분한 성능을 얻으려면 스토리지 볼륨을 과도하게 할당해야 할 수 있습니다.**

3. **사용자가 프로비저닝한 모든 설치와 마찬가지로 클러스터에서 RHEL 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. RHEL 7 컴퓨팅 머신 사용은 더 이상 사용되지 않으며 향후 OpenShift Container Platform 4 릴리스에서 제거될 예정입니다.**

8.3.3.3. 인증서 서명 요청 관리

사용자가 프로비저닝하는 인프라를 사용하는 경우 자동 시스템 관리 기능으로 인해 클러스터의 액세스가 제한되므로 설치한 후 클러스터 인증서 서명 요청(CSR)을 승인하는 메커니즘을 제공해야 합니다. kube-controller-manager는 kubelet 클라이언트 CSR만 승인합니다. machine-approver는 올바른 시스템에서 발행한 요청인지 확인할 수 없기 때문에 kubelet 자격 증명을 사용하여 요청하는 서비스 인증서의 유효성을 보장할 수 없습니다. kubelet 서빙 인증서 요청의 유효성을 확인하고 요청을 승인하는 방법을 결정하여 구현해야 합니다.

추가 리소스

- **베어 메탈 환경에서 3-노드 클러스터를 배포하는 방법에 대한 자세한 내용은 3-노드 클러스터 구성을 참조하십시오.**

- **설치 후 클러스터 인증서 서명 요청 승인에 대한 자세한 내용은 시스템의 인증서 서명 요청 승인을 참조하십시오.**

8.3.3.4. 사용자 프로비저닝 인프라에 대한 네트워킹 요구사항

모든 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템이 부팅 중에 Ignition 구성 파일을 가져오려면 initramfs에 네트워킹을 구성해야 합니다.

초기 부팅 과정에서 시스템에 필요한 부팅 옵션을 제공하여 DHCP 서버를 통해 설정하거나 정적으로 설정하는 IP 주소 구성이 필요합니다. 네트워크 연결이 설정된 후 시스템은 HTTP 또는 HTTPS 서버에서 Ignition 구성 파일을 다운로드합니다. 그런 다음 Ignition 구성 파일을 사용하여 각 머신의 정확한 상태를 설정합니다. Machine Config Operator는 설치 후 새 인증서 또는 키 적용과 같은 머신에 대한 추가 변경을 완료합니다.

클러스터 시스템의 장기적인 관리를 위해 DHCP 서버를 사용하는 것이 좋습니다. DHCP 서버가 클러스터 시스템에 영구 IP 주소, DNS 서버 정보 및 호스트 이름을 제공하도록 구성되었는지 확인합니다.

참고

사용자 프로비저닝 인프라에 DHCP 서비스를 사용할 수 없는 경우 RHCOS 설치 시 노드에 IP 네트워킹 구성과 DNS 서버의 주소를 대신 제공할 수 있습니다. ISO 이미지에서 설치하는 경우 부팅 인수로 전달할 수 있습니다. 고정 IP 프로비저닝 *및 고급 네트워킹 옵션에 대한 자세한 내용은 **RHCOS 설치 및 OpenShift Container Platform 부트스트랩 프로세스 시작* 섹션을 참조하십시오.

Kubernetes API 서버가 클러스터 시스템의 노드 이름을 확인할 수 있어야 합니다. API 서버와 작업자 노드가 서로 다른 영역에 있는 경우, API 서버가 노드 이름을 확인할 수 있도록 기본 DNS 검색 영역을 설정할 수 있습니다. 노드 개체와 모든 DNS 요청에서 항상 정규화된 도메인 이름으로 호스트를 가리키는 것도 지원되는 방법입니다

8.3.3.4.1. DHCP를 통해 클러스터 노드의 호스트 이름 설정

RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 호스트 이름은 NetworkManager를 통해 설정됩니다. 기본적으로 시스템은 DHCP를 통해 호스트 이름을 가져옵니다. DHCP에서 호스트 이름을 제공하지 않거나, 커널 인수를 통해 정적으로 설정하거나 다른 방법을 통해 설정하면 역방향 DNS 조회를 통해 얻을 수 있습니다. 역방향 DNS 조회는 노드에서 네트워크를 초기화한 후 수행되며 확인하는 데 시간이 걸릴 수 있습니다. 다른 시스템 서비스는 이 보다 먼저 시작하여 호스트 이름을 localhost 등으로 감지할 수 있습니다. DHCP를 사용하여 각 클러스터 노드의 호스트 이름을 제공하여 이 문제를 방지할 수 있습니다.

또한 DHCP를 통해 호스트 이름을 설정하면 DNS 분할 수평 구현 환경에서 수동으로 DNS 레코드 이름 구성 오류를 무시할 수 있습니다.

8.3.3.4.2. 네트워크 연결 요구사항

OpenShift Container Platform 클러스터 구성 요소가 통신할 수 있도록 시스템 간 네트워크 연결을 구성해야 합니다. 각 시스템에서 클러스터에 있는 다른 모든 시스템의 호스트 이름을 확인할 수 있어야 합니다.

이 섹션에서는 필요한 포트에 대해 자세히 설명합니다.

중요

연결된 **OpenShift Container Platform** 환경에서 모든 노드는 플랫폼 컨테이너의 이미지를 가져오고 **Red Hat**에 원격 측정 데이터를 제공하기 위해 인터넷에 액세스할 수 있어야 합니다.

표 8.16. 모든 시스템 간 통신에 사용되는 포트

프로토콜	포트	설명
ICMP	해당 없음	네트워크 연결성 테스트
TCP	**1936**	메트릭
	9000-**9999**	**9100**-**9101** 포트의 노드 내보내기 및 **9099** 포트의 Cluster Version Operator를 포함한 호스트 수준 서비스.
	10250-**10259**	Kubernetes에서 예약하는 기본 포트
	10256	openshift-sdn
UDP	**4789**	VXLAN 및 Geneve
	6081	VXLAN 및 Geneve
	9000-**9999**	**9100**-**9101** 포트의 노드 내보내기를 포함한 호스트 수준 서비스.
	500	IPsec IKE 패킷
	4500	IPsec NAT-T 패킷
TCP/UDP	**30000**-**32767**	Kubernetes 노드 포트
ESP	해당 없음	IPsec Encapsulating Security Payload (ESP)

표 8.17. 모든 시스템과 컨트롤 플레인 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**6443**	Kubernetes API

표 **8.18.** 컨트롤 플레인 머신 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**2379-2380**	etcd 서버 및 피어 포트

사용자 프로비저닝 인프라에 대한 **NTP** 구성

OpenShift Container Platform 클러스터는 기본적으로 공용 **NTP(Network Time Protocol)** 서버를 사용하도록 구성되어 있습니다. 로컬 엔터프라이즈 **NTP** 서버를 사용하거나 클러스터가 연결이 끊긴 네트워크에 배포되는 경우 특정 시간 서버를 사용하도록 클러스터를 구성할 수 있습니다. 자세한 내용은 ***chrony* 타임 서비스 설정** 문서를 참조하십시오.

DHCP 서버가 **NTP** 서버 정보를 제공하는 경우 **RHCOS(Red Hat Enterprise Linux CoreOS)** 시스템의 **chrony** 타임 서비스에서 정보를 읽고 **NTP** 서버와 클럭을 동기화할 수 있습니다.

추가 리소스

- **chrony** 타임 서비스 설정

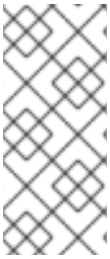
8.3.3.5. 사용자 프로비저닝 **DNS** 요구사항

OpenShift Container Platform 배포의 경우 다음 구성 요소에 **DNS** 이름을 확인해야 합니다.

- **Kubernetes API**
- **OpenShift Container Platform** 애플리케이션 와일드카드
- 부트스트랩, 컨트롤 플레인 및 컴퓨팅 시스템

Kubernetes API, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템에 대한 역방향 **DNS** 확인이 필요합니다.

DNS A/AAAA 또는 CNAME 레코드는 이름 확인에 사용되며 PTR 레코드는 역방향 이름 확인에 사용됩니다. RHCOS (Red Hat Enterprise Linux CoreOS)는 DHCP에서 호스트 이름을 제공하지 않는 한 모든 노드의 호스트 이름을 설정할 때 역방향 레코드를 사용하기 때문에 역방향 레코드가 중요합니다. 또한 역방향 레코드는 OpenShift Container Platform이 작동하는 데 필요한 인증서 서명 요청 (CSR)을 생성하는 데 사용됩니다.



참고

DHCP 서버를 사용하여 각 클러스터 노드에 호스트 이름을 제공하는 것이 좋습니다. 자세한 내용은 *사용자 프로비저닝 인프라 섹션에 대한 DHCP 권장 사항* 섹션을 참조하십시오.

사용자가 프로비저닝한 OpenShift Container Platform 클러스터에 대해 다음 DNS 레코드가 필요하며 설치 전에 있어야 합니다. 각 레코드에서 <cluster_name>은 클러스터 이름이고 <base_domain>은 install-config.yaml 파일에서 지정하는 기반 도메인입니다. 전체 DNS 레코드는 <component>.<cluster_name>.<base_domain> 형식입니다.

표 8.19. 필수 DNS 레코드

구성 요소	레코드	설명
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API 로드 밸런서를 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.
	api-int.<cluster_name>.<base_domain>.	내부적으로 API 로드 밸런서를 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.  **중요** API 서버는 Kubernetes에 기록된 호스트 이름으로 작업자 노드를 확인할 수 있어야 합니다. API 서버가 노드 이름을 확인할 수 없는 경우 프록시된 API 호출이 실패할 수 있으며 pod에서 로그를 검색할 수 없습니다.

구성 요소	레코드	설명
라우트	***.apps.<cluster_name>. <base_domain>.**	애플리케이션 인그레스 로드 밸런서를 참조하는 와일드카드 DNS A/AAA 또는 CNAME 레코드입니다. 애플리케이션 인그레스 로드 밸런서는 Ingress 컨트롤러 Pod를 실행하는 머신을 대상으로 합니다. Ingress 컨트롤러 Pod는 기본적으로 컴퓨팅 머신에서 실행됩니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다. 예를 들어 **console-openshift-console.apps. <cluster_name>.<base_domain>**은 OpenShift Container Platform 콘솔의 와일드카드 경로로 사용됩니다.
부트스트랩 시스템	**bootstrap.<cluster_name>. <base_domain>.**	부트스트랩 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.
컨트롤 플레인 머신	**<master><n>. <cluster_name>. <base_domain>.**	컨트롤 플레인 노드의 각 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.
컴퓨팅 머신	**<worker><n>. <cluster_name>. <base_domain>.**	작업자 노드의 각 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.

참고

OpenShift Container Platform 4.4 이상에서는 DNS 구성에서 etcd 호스트 및 SRV 레코드를 지정할 필요가 없습니다.

작은 정보

dig 명령을 사용하여 이름과 역방향 이름을 확인할 수 있습니다. 자세한 검증 단계는 *사용자 프로비저닝 인프라의 DNS 확인* 섹션을 참조하십시오.

8.3.3.5.1. 사용자 프로비저닝 클러스터의 DNS 구성 예

이 섹션에서는 사용자 프로비저닝 인프라에 OpenShift Container Platform을 배포하기 위한 DNS 요구 사항을 충족하는 A 및 PTR 레코드 구성 샘플을 제공합니다. 샘플은 하나의 DNS 솔루션을 선택하기 위한 조언을 제공하기 위한 것이 아닙니다.

이 예제에서 클러스터 이름은 **ocp4**이고 기본 도메인은 **example.com**입니다.

사용자 프로비저닝 클러스터의 **DNS A** 레코드 구성 예

다음 **BIND** 영역 파일의 예제에서는 사용자가 프로비저닝한 클러스터의 이름 확인을 위한 샘플 **A** 레코드를 보여줍니다.

예 8.4. 샘플 DNS 영역 데이터베이스

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
master0.ocp4.example.com. IN A 192.168.1.97 5
master1.ocp4.example.com. IN A 192.168.1.98 6
master2.ocp4.example.com. IN A 192.168.1.99 7
;
worker0.ocp4.example.com. IN A 192.168.1.11 8
worker1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1

Kubernetes API의 이름 확인을 제공합니다. 레코드는 **API** 로드 밸런서의 **IP** 주소를 나타냅니다.

2

3

와일드카드 경로의 이름 확인을 제공합니다. 레코드는 애플리케이션 인그레스 로드 밸런서의 IP 주소를 나타냅니다. 애플리케이션 인그레스 로드 밸런서는 Ingress 컨트롤러 Pod를 실행하는 머신을 대상으로 합니다. Ingress 컨트롤러 Pod는 기본적으로 컴퓨팅 머신에서 실행됩니다.

참고

이 예제에서는 Kubernetes API 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 API 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

4

부트스트랩 시스템의 이름 확인을 제공합니다.

5 6 7

컨트롤 플레인 시스템의 이름 확인을 제공합니다.

8 9

컴퓨팅 시스템의 이름 확인을 제공합니다.

사용자 프로비저닝 클러스터의 DNS PTR 레코드 구성 예

다음 예제 BIND 영역 파일은 사용자 프로비저닝 클러스터의 역방향 이름 확인을 위한 샘플 PTR 레코드를 보여줍니다.

예 8.5. 역방향 레코드의 샘플 DNS 영역 데이터베이스

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
```

```
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR master0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR master1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR master2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com. 8
;
;EOF
```

1

Kubernetes API의 역방향 DNS 확인을 제공합니다. PTR 레코드는 API 로드 밸런서의 레코드 이름을 참조합니다.

2

Kubernetes API의 역방향 DNS 확인을 제공합니다. PTR 레코드는 API 로드 밸런서의 레코드 이름을 참조하며 내부 클러스터 통신에 사용됩니다.

3

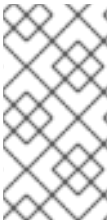
부트스트랩 시스템의 역방향 DNS 확인을 제공합니다.

4 5 6

컨트롤 플레인 시스템의 역방향 DNS 확인을 제공합니다.

7 8

컴퓨팅 시스템의 역방향 DNS 확인을 제공합니다.



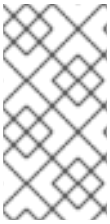
참고

OpenShift Container Platform 애플리케이션 와일드카드에는 PTR 레코드가 필요하지 않습니다.

- 사용자 프로비저닝 인프라에 대한 DNS 확인 검증

8.3.3.6. 사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구사항

OpenShift Container Platform을 설치하기 전에 **API** 및 애플리케이션 인그레스 로드 밸런싱 인프라를 프로비저닝해야 합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 **API** 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.



참고

RHEL(Red Hat Enterprise Linux) 인스턴스를 사용하여 **API** 및 애플리케이션 인그레스 로드 밸런서를 배포하려면 **RHEL** 서브스크립션을 별도로 구매해야 합니다.

로드 밸런서 인프라는 다음 요구 사항을 충족해야 합니다.

1. **API** 로드 밸런서: 플랫폼과 상호 작용하고 플랫폼을 구성할 수 있도록 사용자(인간과 시스템 모두)에게 공통 끝점을 제공합니다. 다음 조건을 설정합니다.

   - **Layer 4** 로드 밸런싱 전용입니다. 이를 **Raw TCP, SSL Passthrough** 또는 **SSL Bridge** 모드라고 합니다. **SSL Bridge** 모드를 사용하는 경우, **API** 경로에 대해 **SNI(Server Name Indication,** 서버 이름 표시)를 활성화해야 합니다.

   - 스테이트리스 로드 밸런싱 알고리즘입니다. 옵션은 로드 밸런서 구현에 따라 달라집니다.

참고

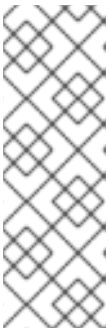
API 로드 밸런서가 제대로 작동하기 위해 세션 지속성이 필요하지 않습니다.

로드 밸런서의 전면과 후면 모두에서 다음 포트를 구성하십시오.

표 8.20. API 로드 밸런서

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
6443	부트스트랩 및 컨트롤 플레인. 부트스트랩 시스템이 클러스터 컨트롤 플레인을 초기화한 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다. API 서버 상태 검사 프로브에 대한 **/readyz** 끝점을 구성해야 합니다.	X	X	Kubernetes API 서버

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
22623	부트스트랩 및 컨트롤 플레인. 부트스트랩 시스템이 클러스터 컨트롤 플레인을 초기화한 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다.	X		시스템 구성 서버



참고

API 서버가 /readyz 엔드포인트를 해제하는 시점부터 풀에서 API 서버 인스턴스가 제거되는 시점까지 시간이 30초를 넘지 않도록 로드 밸런서를 구성해야 합니다. /readyz가 오류를 반환하거나 정상 상태가 된 후 정해진 시간 안에 끝점이 제거 또는 추가되어야 합니다. 5초 또는 10초의 프로빙 주기(두 번의 성공적인 요청은 정상 상태, 세 번의 요청은 비정상 상태)는 충분한 테스트를 거친 값입니다.

2. **애플리케이션 인그레스 로드 밸런서: 클러스터 외부에서 유입되는 애플리케이션 트래픽에 대한 인그래스 포인트를 제공합니다. 다음 조건을 설정합니다.**

- **Layer 4 로드 밸런싱 전용입니다. 이를 Raw TCP, SSL Passthrough 또는 SSL Bridge 모드라고 합니다. SSL Bridge 모드를 사용하는 경우 인그레스 경로에 대해 SNI(Server Name Indication, 서버 이름 표시)를 활성화해야 합니다.**

- **사용 가능한 옵션과 플랫폼에서 호스팅되는 애플리케이션 유형에 따라 연결 기반 또는 세션 기반 지속성이 권장됩니다.**

작은 정보

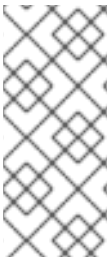
애플리케이션 인그레스 로드 밸런서에서 클라이언트의 실제 IP 주소를 확인할 수 있는 경우 소스 IP 기반 세션 지속성을 활성화하면 엔드 투 엔드 TLS 암호화를 사용하는 애플리케이션의 성능을 향상시킬 수 있습니다.

로드 밸런서의 전면과 후면 모두에서 다음 포트를 구성하십시오.

표 8.21. 애플리케이션 인그레스 로드 밸런서

포트	백엔드 시스템(풀 멤버)	내부	외부	설명

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
443	기본적으로 인그레스 컨트롤러 pod, 컴퓨팅 또는 작업자를 실행하는 시스템입니다.	X	X	HTTPS 트래픽
80	기본적으로 인그레스 컨트롤러 pod, 컴퓨팅 또는 작업자를 실행하는 시스템입니다.	X	X	HTTP 트래픽
1936	기본적으로 Ingress 컨트롤러 Pod를 실행하는 작업자 노드입니다. 수신 상태 점검 프로브에 대해 **/healthz/ready** 끝점을 구성해야 합니다.	X	X	HTTP 트래픽



참고

컴퓨팅 노드가 0인 3-노드 클러스터를 배포하는 경우 Ingress 컨트롤러 Pod는 컨트롤 플레인 노드에서 실행됩니다. 3-노드 클러스터 배포에서 HTTP 및 HTTPS 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다.

참고

인그레스 라우터에 대한 작업 구성이 OpenShift Container Platform 클러스터에 필요합니다. 컨트롤 플레인 초기화 후 인그레스 라우터를 설정해야 합니다.

8.3.3.6.1. 사용자 프로비저닝 클러스터의 로드 밸런서 구성 예

이 섹션에서는 사용자 프로비저닝 클러스터의 로드 밸런싱 요구 사항을 충족하는 API 및 애플리케이션 수신 로드 밸런서 구성 예를 제공합니다. 샘플은 HAProxy 로드 밸런서에 대한 /etc/haproxy/haproxy.cfg 구성입니다. 이 예제에서는 하나의 로드 밸런싱 솔루션을 선택하기 위한 조언을 제공하는 것을 목적으로 하지 않습니다.

참고

이 예제에서는 Kubernetes API 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 API 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

예 8.6. API 및 애플리케이션 인그레스 로드 밸런서 구성 샘플

```
global
```

```
    log         127.0.0.1 local2
    pidfile     /var/run/haproxy.pid
    maxconn     4000
    daemon
  defaults
    mode                    http
    log                     global
    option                  dontlognull
    option http-server-close
    option                  redispatch
    retries                 3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s
    maxconn                 3000
  frontend stats
    bind *:1936
    mode            http
    log             global
    maxconn 10
    stats enable
    stats hide-version
    stats refresh 30s
    stats show-node
    stats show-desc Stats for ocp4 cluster (1)
    stats auth admin:ocp4
    stats uri /stats
  listen api-server-6443 (2)
    bind *:6443
    mode tcp
    server bootstrap bootstrap.ocp4.example.com:6443 check inter 1s backup (3)
    server master0 master0.ocp4.example.com:6443 check inter 1s
    server master1 master1.ocp4.example.com:6443 check inter 1s
    server master2 master2.ocp4.example.com:6443 check inter 1s
  listen machine-config-server-22623 (4)
    bind *:22623
    mode tcp
    server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup (5)
    server master0 master0.ocp4.example.com:22623 check inter 1s
    server master1 master1.ocp4.example.com:22623 check inter 1s
    server master2 master2.ocp4.example.com:22623 check inter 1s
  listen ingress-router-443 (6)
    bind *:443
    mode tcp
    balance source
    server worker0 worker0.ocp4.example.com:443 check inter 1s
    server worker1 worker1.ocp4.example.com:443 check inter 1s
  listen ingress-router-80 (7)
    bind *:80
    mode tcp
```

```
balance source
server worker0 worker0.ocp4.example.com:80 check inter 1s
server worker1 worker1.ocp4.example.com:80 check inter 1s
```

1 이 예에서 클러스터 이름은 **ocp4** 입니다.

2 포트 **6443**은 **Kubernetes API** 트래픽을 처리하고 컨트롤 플레인 시스템을 가리킵니다.

3 5 부트스트랩 항목은 **OpenShift Container Platform** 클러스터 설치 전에 있어야 하며 부트스트랩 프로세스가 완료된 후 제거해야 합니다.

4 포트 **22623**은 머신 구성 서버 트래픽을 처리하고 컨트롤 플레인 시스템을 가리킵니다.

6 포트 **443**은 **HTTPS** 트래픽을 처리하고 **Ingress** 컨트롤러 **Pod**를 실행하는 시스템을 가리킵니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

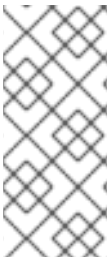
7 포트 **80**은 **HTTP** 트래픽을 처리하고 **Ingress** 컨트롤러 **Pod**를 실행하는 머신을 가리킵니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

참고

컴퓨팅 노드가 **0**인 **3-**노드 클러스터를 배포하는 경우 **Ingress** 컨트롤러 **Pod**는 컨트롤 플레인 노드에서 실행됩니다. **3-**노드 클러스터 배포에서 **HTTP** 및 **HTTPS** 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다.

작은 정보

HAProxy를 로드 밸런서로 사용하는 경우 **HAProxy** 노드에서 **netstat -nltupe**를 실행하여 **haproxy** 프로세스가 포트 **6443, 22623, 443** 및 **80**에서 수신 대기 중인지 확인할 수 있습니다.



참고

HAProxy를 로드 밸런서로 사용하고 SELinux가 enforcing으로 설정된 경우 HAProxy 서비스가 setsebool -P haproxy_connect_any=1을 실행하여 구성된 TCP 포트에 바인딩할 수 있는지 확인해야 합니다.

8.3.4. 사용자 프로비저닝 인프라 준비

사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치하기 전에 기본 인프라를 준비해야 합니다.

이 섹션에서는 OpenShift Container Platform 설치를 준비하기 위해 클러스터 인프라를 설정하는 데 필요한 높은 수준의 단계에 대해 자세히 설명합니다. 여기에는 클러스터 노드에 대한 IP 네트워킹 및 네트워크 연결 구성, 방화벽을 통해 필요한 포트 활성화, 필수 DNS 및 로드 밸런싱 인프라를 설정하는 작업이 포함됩니다.

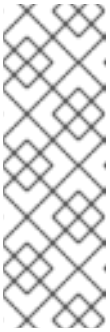
준비 후 클러스터 인프라는 *사용자 프로비저닝 인프라가 있는 클러스터의 요구 사항* 섹션에 설명된 요구 사항을 충족해야 합니다.

사전 요구 사항

- OpenShift Container Platform 4.x 테스트된 통합 페이지를 검토했습니다.
- *사용자 프로비저닝 인프라가 있는 클러스터의 요구 사항* 섹션에 자세히 설명된 인프라 요구 사항을 검토했습니다.

프로세스

1. DHCP를 사용하여 클러스터 노드에 IP 네트워킹 구성을 제공하는 경우 DHCP 서비스를 구성합니다.
   a. 노드의 영구 IP 주소를 DHCP 서버 구성에 추가합니다. 구성에서 관련 네트워크 인터페이스의 MAC 주소를 각 노드의 의도한 IP 주소와 일치시킵니다.
   b. DHCP를 사용하여 클러스터 시스템의 IP 주소 지정을 구성하는 경우 시스템은 DHCP를 통해 DNS 서버 정보도 가져옵니다. DHCP 서버 구성을 통해 클러스터 노드에서 사용하는 영구 DNS 서버 주소를 정의합니다.

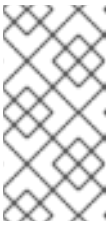


참고

DHCP 서비스를 사용하지 않는 경우 **RHCOS** 설치 시 **IP** 네트워킹 구성과 **DNS** 서버의 주소를 노드에 제공해야 합니다. **ISO** 이미지에서 설치하는 경우 부팅 인수로 전달할 수 있습니다. 고정 **IP** 프로비저닝 *및 고급 네트워킹 옵션에 대한 자세한 내용은 **RHCOS** 설치 및 **OpenShift Container Platform** 부트스트랩 프로세스 시작* 섹션을 참조하십시오.

c.

DHCP 서버 구성에 클러스터 노드의 호스트 이름을 정의합니다. 호스트 이름 고려 사항에 대한 자세한 내용은 ***DHCP**를 통해 클러스터 노드 호스트 이름 설정* 섹션을 참조하십시오.



참고

DHCP 서비스를 사용하지 않는 경우 클러스터 노드는 역방향 **DNS** 조회를 통해 호스트 이름을 가져옵니다.

2.

네트워크 인프라가 클러스터 구성 요소 간 필수 네트워크 연결을 제공하는지 확인합니다. 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 인프라 섹션의 네트워킹 요구 사항* 섹션을 참조하십시오.

3.

OpenShift Container Platform 클러스터 구성 요소가 통신하는 데 필요한 포트를 활성화하도록 방화벽을 구성합니다. 필요한 포트에 대한 자세한 내용은 *사용자 프로비저닝 인프라 섹션의 네트워킹 요구 사항* 섹션을 참조하십시오.

4.

클러스터에 필요한 **DNS** 인프라를 설정합니다.

a.

Kubernetes API, 애플리케이션 와일드카드, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템의 **DNS** 이름 확인을 구성합니다.

b.

Kubernetes API, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템에 대한 역방향 **DNS** 확인을 구성합니다.

OpenShift Container Platform DNS 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 **DNS** 요구 사항* 섹션을 참조하십시오.

5.

DNS 구성을 확인합니다.

a. 설치 노드에서 **Kubernetes API**의 레코드 이름, 와일드카드 경로 및 클러스터 노드에 대해 **DNS** 조회를 실행합니다. 응답의 **IP** 주소가 올바른 구성 요소에 해당하는지 확인합니다.

b. 설치 노드에서 로드 밸런서 및 클러스터 노드의 **IP** 주소에 대해 역방향 **DNS** 조회를 실행합니다. 응답의 레코드 이름이 올바른 구성 요소에 해당하는지 확인합니다.

자세한 **DNS** 검증 단계는 *사용자 프로비저닝 인프라에 대한 DNS 확인* 섹션을 참조하십시오.

6. 필요한 **API** 및 애플리케이션 수신 로드 밸런싱 인프라를 프로비저닝합니다. 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구 사항* 섹션을 참조하십시오.

참고

일부 로드 밸런싱 솔루션에는 로드 밸런싱을 초기화하기 전에 클러스터 노드의 **DNS** 이름을 확인해야 합니다.

추가 리소스

- 사용자 프로비저닝 인프라를 포함한 클러스터의 시스템 요구사항
- **RHCOS** 설치 및 **OpenShift Container Platform** 부트스트랩 프로세스 시작
- **DHCP**를 통해 클러스터 노드의 호스트 이름 설정
- 고급 **RHCOS** 설치 구성 옵션
- 사용자 프로비저닝 인프라에 대한 네트워킹 요구사항
- 사용자 프로비저닝 **DNS** 요구사항
- 사용자 프로비저닝 인프라에 대한 **DNS** 확인 검증

- 사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구사항

8.3.5. 사용자 프로비저닝 인프라에 대한 DNS 확인 검증

사용자 프로비저닝 인프라에 **OpenShift Container Platform**을 설치하기 전에 **DNS** 구성을 확인할 수 있습니다.

중요

클러스터를 설치하기 전에 이 섹션에 설명된 검증 단계를 성공해야 합니다.

사전 요구 사항

- 사용자 프로비저닝 인프라에 필요한 **DNS** 레코드를 구성했습니다.

프로세스

1. 설치 노드에서 **Kubernetes API**의 레코드 이름, 와일드카드 경로 및 클러스터 노드에 대해 **DNS** 조회를 실행합니다. 응답에 포함된 **IP** 주소가 올바른 구성 요소에 해당하는지 확인합니다.

   a. **Kubernetes API** 레코드 이름을 조회합니다. 결과가 **API** 로드 밸런서의 **IP** 주소를 가리키는지 확인합니다.

   ```
   $ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
   ```

   1 **<nameserver_ip>**를 네임서버의 **IP** 주소로, **<cluster_name>**을 클러스터 이름으로, **<base_domain>**을 기본 도메인 이름으로 바꿉니다.

   출력 예

   ```
   api.ocp4.example.com.  0 IN A 192.168.1.5
   ```

b.

Kubernetes 내부 API 레코드 이름을 조회합니다. 결과가 API 로드 밸런서의 IP 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
```

출력 예

```
api-int.ocp4.example.com.  0 IN A 192.168.1.5
```

c.

예제 *.apps.<cluster_name>.<base_domain>을 테스트합니다. DNS 와일드카드를 조회합니다. 모든 애플리케이션 와일드카드 조회는 애플리케이션 인그레스 로드 밸런서의 IP 주소로 확인되어야 합니다.

```
$ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.
<base_domain>
```

출력 예

```
random.apps.ocp4.example.com.  0 IN A 192.168.1.5
```

참고

예제 출력에서는 Kubernetes API 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 API 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

random 항목을 다른 와일드카드 값으로 교체할 수 있습니다. 예를 들어 OpenShift Container Platform 콘솔의 경로를 쿼리할 수 있습니다.

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

출력 예

```
console-openshift-console.apps.ocp4.example.com. 0 IN A 192.168.1.5
```

d. 부트스트랩 DNS 레코드 이름에 대해 조회를 실행합니다. 결과가 부트스트랩 노드의 IP 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.
<base_domain>
```

출력 예

```
bootstrap.ocp4.example.com.  0 IN A 192.168.1.96
```

e. 이 방법을 사용하여 컨트롤 플레인 및 컴퓨팅 노드의 DNS 레코드 이름에 대해 조회를 수행합니다. 결과가 각 노드의 IP 주소에 해당하는지 확인합니다.

2. 설치 노드에서 로드 밸런서 및 클러스터 노드의 IP 주소에 대해 역방향 DNS 조회를 실행합니다. 응답에 포함된 레코드 이름이 올바른 구성 요소에 해당하는지 확인합니다.

a. API 로드 밸런서의 IP 주소에 대해 역방향 조회를 수행합니다. 응답에 Kubernetes API 및 Kubernetes 내부 API의 레코드 이름이 포함되어 있는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

출력 예

```
5.1.168.192.in-addr.arpa. 0 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 0 IN PTR api.ocp4.example.com. 2
```

1

Kubernetes 내부 API의 레코드 이름을 제공합니다.

2

Kubernetes API의 레코드 이름을 제공합니다.

참고

OpenShift Container Platform 애플리케이션 와일드카드에는 PTR 레코드가 필요하지 않습니다. 애플리케이션 인그레스 로드 밸런서의 IP 주소에 대한 역방향 DNS 확인에는 유효성 검사 단계가 필요하지 않습니다.

b. 부트스트랩 노드의 IP 주소에 대해 역방향 조회를 수행합니다. 결과가 부트스트랩 노드의 DNS 레코드 이름을 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

출력 예

```
96.1.168.192.in-addr.arpa. 0 IN PTR bootstrap.ocp4.example.com.
```

c. 이 방법을 사용하여 컨트롤 플레인 및 컴퓨팅 노드의 IP 주소에 대해 역방향 조회를 수행합니다. 결과가 각 노드의 DNS 레코드 이름과 일치하는지 확인합니다.

추가 리소스

- 사용자 프로비저닝 DNS 요구사항
- 사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구사항

8.3.6. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 core 사용자의 ~/.ssh/authorized_keys 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 core로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. ./openshift-install gather 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

참고

AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1

새 SSH 키의 경로 및 파일 이름(예: ~/.ssh/id_ed25519)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 '~/.ssh 디렉터리에 있는지 확인하십시오.

참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 ed25519 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 rsa 또는 ecdsa 알고리즘을 사용하는 키를 생성합니다.

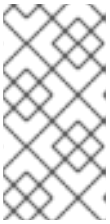
2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 ~/.ssh/id_ed25519.pub 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 ./openshift-install gather 명령을 사용하려는 경우 필요합니다.



참고

일부 배포에서는 ~/.ssh/id_rsa 및 ~/.ssh/id_dsa와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. ssh-agent 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. ssh-agent에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: ~/.ssh/id_ed25519).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

추가 리소스

- 노드 상태 확인

8.3.7. 설치 프로그램 받기

OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

프로세스

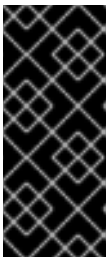
1. **OpenShift Cluster Manager** 사이트의 인프라 공급자 페이지에 액세스합니다. **Red Hat** 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.

2. 인프라 공급자를 선택합니다.

3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.

중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.



중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 **OpenShift Container Platform** 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 **Linux** 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 **OpenShift Container Platform** 구성 요소에 대한 컨테이너 이미지를 제공하는 **Quay.io**를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

8.3.8. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 **OpenShift Container Platform**과 상호 작용하기 위해 **OpenShift CLI(oc)**를 설치할 수 있습니다. **Linux, Windows** 또는 **macOS**에 **oc**를 설치할 수 있습니다.

중요

이전 버전의 oc를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 oc를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 Linux에서 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.
2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. OpenShift v4.9 Linux Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. oc 바이너리를 PATH에 있는 디렉터리에 배치합니다.

   PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치

다음 절차에 따라 Windows에 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.

2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.

3. OpenShift v4.9 Windows Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.

4. ZIP 프로그램으로 아카이브의 압축을 풉니다.

5. oc 바이너리를 PATH에 있는 디렉터리로 이동합니다.

   PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치

다음 절차에 따라 macOS에서 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.

2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.

3. OpenShift v4.9 MacOSX Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.

4. 아카이브의 압축을 해제하고 압축을 풉니다.

5. oc 바이너리 PATH의 디렉터리로 이동합니다.

   PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
$ oc <command>
```

8.3.9. 수동으로 설치 구성 파일 만들기

OpenShift Container Platform을 사용자가 프로비저닝한 설치의 경우 설치 구성 파일을 수동으로 생성합니다.

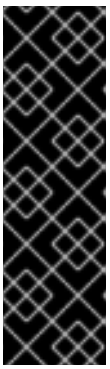
사전 요구 사항

- 로컬 시스템에 설치 프로그램에 제공할 SSH 공개 키가 있습니다. 키는 디버깅 및 재해 복구를 위해 클러스터 노드에 대한 SSH 인증에 사용됩니다.
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿이 있습니다.

프로세스

1. 필요한 설치 자산을 저장할 설치 디렉터리를 만듭니다.

```
$ mkdir <installation_directory>
```

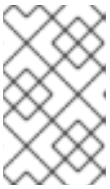


중요

디렉터리를 만들어야 합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

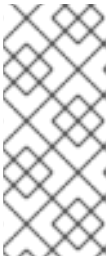
2.

샘플 **install-config.yaml** 파일 템플릿을 사용자 지정하여 **<installation_directory>**에 저장합니다.



참고

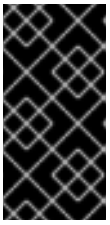
이 설정 파일의 이름을 **install-config.yaml**로 지정해야 합니다.



참고

일부 플랫폼 유형의 경우 대체로 **./openshift-install create install-config --dir <installation_directory>** 를 실행하여 **install-config.yaml** 파일을 생성할 수 있습니다. 프롬프트에서 클러스터 구성에 대한 세부 정보를 제공할 수 있습니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.



중요

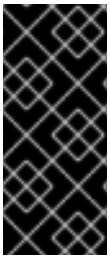
install-config.yaml 파일은 설치 과정의 다음 단계에서 사용됩니다. 이 시점에서 이를 백업해야 합니다.

8.3.9.1. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 환경에 대한 세부 정보를 설명하는 사용자 지정 **install-config.yaml** 설치 구성 파일을 제공합니다.

참고

설치한 후에는 **install-config.yaml** 파일에서 이러한 매개변수를 수정할 수 없습니다.



중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

8.3.9.1.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 8.22. 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	**dev** 와 같은 소문자 및 하이픈(**-**)의 문자열입니다.
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

매개변수	설명	값

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

8.3.9.1.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 8.23. 네트워크 매개변수

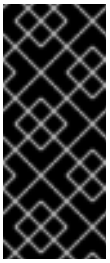
매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체 **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.

매개변수	설명	값
networking.clusterNetwork	Pod의 IP 주소 블록입니다. 기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된 **cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16** **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

8.3.9.1.3. 선택적 구성 매개변수

선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 8.24. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은**3**입니다.

매개변수	설명	값
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 기본값은 **External**입니다. 이 필드를 **Internal** 로 설정하는 것은 클라우드 이외의 플랫폼에서는 지원되지 않습니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

8.3.9.2. 베어 메탈의 샘플 install-config.yaml 파일

install-config.yaml 파일을 사용자 지정하여 OpenShift Container Platform 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.

```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 0 (4)
controlPlane: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3 (7)
metadata:
  name: test (8)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 (9)
    hostPrefix: 23 (10)
  networkType: OpenShiftSDN
  serviceNetwork: (11)
  - 172.30.0.0/16
platform:
  none: {} (12)
fips: false (13)
pullSecret: '{"auths": ...}' (14)
sshKey: 'ssh-ed25519 AAAA...' (15)
```

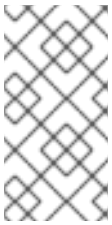
(1)

2 5

controlPlane 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

3 6

동시 멀티스레딩(**SMT**) 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 **SMT**가 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. **SMT**를 비활성화하는 경우 모든 클러스터 머신에서 이를 비활성화해야 합니다. 여기에는 컨트롤 플레인과 컴퓨팅 머신이 모두 포함됩니다.



참고

SMT(동시 멀티 스레딩)는 기본적으로 활성화되어 있습니다. **BIOS** 설정에서 **SMT**를 활성화하지 않으면 **hyperthreading** 매개변수가 적용되지 않습니다.

중요

BIOS에서든 **install-config.yaml** 파일에서든 **hyperthreading**을 비활성화한 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.

4

사용자 프로비저닝 인프라에 **OpenShift Container Platform**을 설치할 때 이 값을 **0**으로 설정해야 합니다. 설치 프로그램에서 제공하는 설치에서 매개 변수는 클러스터가 생성 및 관리하는 컴퓨팅 머신 수를 제어합니다. 사용자 프로비저닝 설치에서는 클러스터 설치를 완료하기 전에 컴퓨팅 시스템을 수동으로 배포해야 합니다.

참고

3-노드 클러스터를 설치하는 경우 **RHCOS(Red Hat Enterprise Linux CoreOS)** 시스템을 설치할 때 컴퓨팅 머신을 배포하지 마십시오.

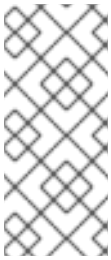
7

클러스터에 추가하는 컨트롤 플레인 시스템의 수입니다. 클러스터에서 이 값을 클러스터의 **etcd** 끝점 수로 사용하므로 이 값은 배포하는 컨트롤 플레인 시스템의 수와 일치해야 합니다.

8

9

Pod IP 주소가 할당되는 IP 주소 블록입니다. 이 블록은 기존 물리적 네트워크와 중복되지 않아야합니다. 이러한 IP 주소는 Pod 네트워크에 사용됩니다. 외부 네트워크에서 Pod에 액세스해야 하는 경우, 트래픽을 관리하도록 로드 밸런서와 라우터를 설정해야 합니다.



참고

클래스 E CIDR 범위는 향후 사용을 위해 예약되어 있습니다. 클래스 E CIDR 범위를 사용하려면 네트워킹 환경에서 클래스 E CIDR 범위 내에서 IP 주소를 수락하는지 확인해야 합니다.

10

개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 hostPrefix를 23으로 설정하면 지정된 cidr 이외 /23 서브넷이 각 노드에 할당되어 510(2^(32 - 23) - 2) Pod IP 주소가 허용됩니다. 외부 네트워크에서 노드에 액세스해야 하는 경우 트래픽을 관리하도록 로드 밸런서와 라우터를 구성합니다.

11

서비스 IP 주소에 사용할 IP 주소 풀입니다. IP 주소 풀은 하나만 입력할 수 있습니다. 이 블록은 기존 물리적 네트워크와 중복되지 않아야합니다. 외부 네트워크에서 서비스에 액세스해야 하는 경우, 트래픽을 관리하도록 로드 밸런서와 라우터를 구성합니다.

12

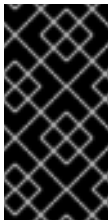
플랫폼을 none으로 설정해야 합니다. 플랫폼에 대한 추가 플랫폼 구성 변수는 지정할 수 없습니다.

중요

플랫폼 유형으로 설치된 클러스터는 Machine API로 컴퓨팅 머신 관리와 같은 일부 기능을 사용할 수 없습니다. 이 제한은 클러스터에 연결된 컴퓨팅 시스템이 일반적으로 기능을 지원하는 플랫폼에 설치된 경우에도 적용됩니다. 이 매개변수는 설치 후 변경할 수 없습니다.

13

FIPS 모드 활성화 또는 비활성화 여부입니다. 기본적으로 FIPS 모드는 비활성화됩니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.



중요

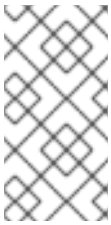
FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 **OpenShift Container Platform** 배포에서만 지원됩니다.

14

Red Hat OpenShift Cluster Manager의 풀 시크릿. 이 풀 시크릿을 사용하면 **OpenShift Container Platform** 구성 요소에 대한 컨테이너 이미지를 제공하는 **Quay.io**를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

15

RHCOS(Red Hat Enterprise Linux CoreOS)의 **core** 사용자에 대한 **SSH** 공용 키입니다.



참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 **OpenShift Container Platform** 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 **SSH** 키를 지정합니다.

추가 리소스

- **API** 및 애플리케이션 수신 로드 밸런싱 요구 사항에 대한 자세한 내용은 사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구 사항을 참조하십시오.

8.3.10. 네트워크 구성 단계

OpenShift Container Platform을 설치하기 전에 네트워크 구성을 사용자 지정할 수 있습니다.

1 단계

매니페스트 파일을 생성하기 전에 **install-config.yaml** 파일에서 다음 네트워크 관련 필드를 사용자 지정할 수 있습니다.

- **networking.networkType**
- **networking.clusterNetwork**

- **networking.serviceNetwork**

- **networking.machineNetwork**

  이러한 필드에 대한 자세한 내용은 *설치 구성 매개 변수*에서 참조하십시오.

참고

기본 **NIC**가 상주하는 **CIDR**과 일치하도록 **networking.machineNetwork**를 설정합니다.

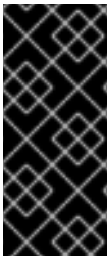
2 단계

openshift-install create manifests 를 실행하여 매니페스트 파일을 생성한 후 수정할 필드로 사용자 지정된 **Cluster Network Operator** 매니페스트를 정의할 수 있습니다. 매니페스트를 사용하여 고급 네트워크 구성을 지정할 수 있습니다.

install-config.yaml 파일에서 **1** 단계에 지정된 값을 **2** 단계에서 재정의할 수 없습니다. 그러나 **2** 단계에서 클러스터 네트워크 공급자를 추가로 사용자 지정할 수 있습니다.

8.3.11. 고급 네트워크 구성 지정

클러스터 네트워크 제공자의 고급 네트워크 구성을 사용하여 클러스터를 기존 네트워크 환경에 통합할 수 있습니다. 클러스터를 설치하기 전에만 고급 네트워크 구성을 지정할 수 있습니다.



중요

설치 프로그램에서 생성한 **OpenShift Container Platform** 매니페스트 파일을 수정하여 네트워크 구성을 사용자 정의하는 것은 지원되지 않습니다. 다음 절차에서와 같이 생성한 매니페스트 파일을 적용할 수 있습니다.

사전 요구 사항

- **install-config.yaml** 파일을 생성하고 수정 작업을 완료했습니다.

프로세스

1. 설치 프로그램이 포함된 디렉터리로 변경하고 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 <installation_directory>는 클러스터의 install-config.yaml 파일이 포함된 디렉터리의 이름을 지정합니다.

2. <installation_directory>/ manifests/ 디렉토리에 cluster-network-03-config.yml이라는 stub 매니페스트 파일을 만듭니다.

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
```

3. 다음과 같이 cluster-network-03-config.yml 파일에서 클러스터의 고급 네트워크 구성을 지정합니다.

OpenShift SDN 네트워크 공급자에 대한 다른 VXLAN 포트 지정

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    openshiftSDNConfig:
      vxlanPort: 4800
```

OVN-Kubernetes 네트워크 공급자의 IPsec 활성화

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
```

```
  name: cluster
spec:
  defaultNetwork:
    ovnKubernetesConfig:
      ipsecConfig: {}
```

4. 선택사항: **manifests/cluster-network-03-config.yml** 파일을 백업합니다. 설치 프로그램은 **Ignition** 구성 파일을 생성할 때 **manifests/** 디렉터리를 사용합니다.

8.3.12. CNO(Cluster Network Operator) 구성

클러스터 네트워크의 구성은 **CNO(Cluster Network Operator)** 구성의 일부로 지정되며 **cluster**라는 이름의 **CR**(사용자 정의 리소스) 오브젝트에 저장됩니다. **CR**은 **operator.openshift.io API** 그룹에서 **Network API**의 필드를 지정합니다.

CNO 구성은 **Network.config.openshift.io API** 그룹의 **Network API**에서 클러스터 설치 중에 다음 필드를 상속하며 이러한 필드는 변경할 수 없습니다.

clusterNetwork

Pod IP 주소가 할당되는 **IP** 주소 풀입니다.

serviceNetwork

서비스를 위한 **IP** 주소 풀입니다.

defaultNetwork.type

OpenShift SDN 또는 **OVN-Kubernetes**와 같은 클러스터 네트워크 공급자입니다.

cluster라는 **CNO** 오브젝트에서 **defaultNetwork** 오브젝트의 필드를 설정하여 클러스터의 클러스터 네트워크 공급자 구성을 지정할 수 있습니다.

8.3.12.1. CNO(Cluster Network Operator) 구성 오브젝트

CNO(Cluster Network Operator)의 필드는 다음 표에 설명되어 있습니다.

표 **8.25. CNO(Cluster Network Operator)** 구성 오브젝트

필드	유형	설명
metadata.name	**string**	CNO 개체 이름입니다. 이 이름은 항상 **cluster**입니다.
spec.clusterNetwork	**array**	Pod IP 주소가 할당되는 IP 주소 블록과 클러스터의 각 개별 노드에 할당된 서브넷 접두사 길이를 지정하는 목록입니다. 예를 들면 다음과 같습니다. spec: clusterNetwork: - cidr: 10.128.0.0/19 hostPrefix: 23 - cidr: 10.128.32.0/19 hostPrefix: 23 매니페스트를 생성하기 전에 **install-config.yaml** 파일에서만 이 필드를 사용자 지정할 수 있습니다. 값은 매니페스트 파일에서 읽기 전용입니다.
spec.serviceNetwork	**array**	서비스의 IP 주소 블록입니다. OpenShift SDN 및 OVN-Kubernetes CNI(Container Network Interface) 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다. 예를 들면 다음과 같습니다. spec: serviceNetwork: - 172.30.0.0/14 매니페스트를 생성하기 전에 **install-config.yaml** 파일에서만 이 필드를 사용자 지정할 수 있습니다. 값은 매니페스트 파일에서 읽기 전용입니다.
spec.defaultNetwork	**object**	클러스터 네트워크의 CNI(Container Network Interface) 클러스터 네트워크 공급자를 구성합니다.
spec.kubeProxyConfig	**object**	이 개체의 필드는 kube-proxy 구성을 지정합니다. OVN-Kubernetes 클러스터 네트워크 공급자를 사용하는 경우 kube-proxy 구성이 적용되지 않습니다.

defaultNetwork 오브젝트 구성

defaultNetwork 오브젝트의 값은 다음 표에 정의되어 있습니다.

표 8.26. defaultNetwork 오브젝트

필드	유형	설명

필드	유형	설명
type	**string**	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, 클러스터 네트워크 공급자가 설치 중에 선택됩니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다. **참고** OpenShift Container Platform은 기본적으로 OpenShift SDN CNI(Container Network Interface) 클러스터 네트워크 공급자를 사용합니다.
openshiftSDNConfig	**object**	이 오브젝트는 OpenShift SDN 클러스터 네트워크 공급자에만 유효합니다.
ovnKubernetesConfig	**object**	이 오브젝트는 OVN-Kubernetes 클러스터 네트워크 공급자에만 유효합니다.

OpenShift SDN CNI 네트워크 공급자에 대한 구성

다음 표에서는 OpenShift SDN Container Network Interface (CNI) 클러스터 네트워크 공급자의 구성 필드를 설명합니다.

표 8.27. openshiftSDNConfig 오브젝트

필드	유형	설명
mode	**string**	OpenShift SDN의 네트워크 격리 모드를 구성합니다. 기본값은 **NetworkPolicy**입니다. **Multitenant** 및 **Subnet** 값은 OpenShift Container Platform 3.x 버전과의 역호환할 수 있지만 권장되지는 않습니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
mtu	**integer**	VXLAN 오버레이 네트워크의 최대 전송 단위(MTU)입니다. 이는 기본 네트워크 인터페이스의 MTU를 기준으로 자동 탐지됩니다. 일반적으로 감지된 MTU를 재정의할 필요는 없습니다. 자동 감지 값이 예상 밖인 경우 노드의 기본 네트워크 인터페이스의 MTU가 올바른지 확인합니다. 이 옵션을 사용하여 노드의 기본 네트워크 인터페이스의 MTU 값을 변경할 수 없습니다. 클러스터에 다른 노드에 대한 다른 MTU 값이 필요한 경우 이 값을 클러스터의 가장 낮은 MTU 값보다 **50** 미만으로 설정해야 합니다. 예를 들어 클러스터의 일부 노드에는 **9001**의 MTU가 있고 일부에는 **1500**의 MTU가 있는 경우 이 값을 **1450**으로 설정해야 합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.

필드	유형	설명
vxlanPort	**integer**	모든 VXLAN 패킷에 사용할 포트입니다. 기본값은 **4789**입니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다. 기존 노드가 다른 VXLAN 네트워크에 속하는 가상 환경에서 실행 중인 경우에는 기본값을 변경해야 할 수도 있습니다. 예를 들어 VMware NSX-T 위에서 OpenShift SDN 오버레이를 실행할 때 두 SDN이 동일한 기본 VXLAN 포트 번호를 사용하므로 VXLAN의 대체 포트를 선택해야 합니다. AWS(Amazon Web Services)에서는 포트 **9000**과 포트 **9999** 사이에서 VXLAN의 대체 포트를 선택할 수 있습니다.

OpenShift SDN 구성 예

```
defaultNetwork:
  type: OpenShiftSDN
  openshiftSDNConfig:
    mode: NetworkPolicy
    mtu: 1450
    vxlanPort: 4789
```

OVN-Kubernetes CNI 클러스터 네트워크 공급자에 대한 구성

다음 표에서는 **OVN-Kubernetes CNI** 클러스터 네트워크 공급자의 구성 필드를 설명합니다.

표 8.28. ovnKubernetesConfig object

필드	유형	설명

필드	유형	설명
mtu	**integer**	Geneve(Generic Network Virtualization Encapsulation) 오버레이 네트워크의 MTU(최대 전송 단위)입니다. 이는 기본 네트워크 인터페이스의 MTU를 기준으로 자동 탐지됩니다. 일반적으로 감지된 MTU를 재정의할 필요는 없습니다. 자동 감지 값이 예상 밖인 경우 노드의 기본 네트워크 인터페이스의 MTU가 올바른지 확인합니다. 이 옵션을 사용하여 노드의 기본 네트워크 인터페이스의 MTU 값을 변경할 수 없습니다. 클러스터에 다른 노드에 대한 다른 MTU 값이 필요한 경우, 이 값을 클러스터의 가장 낮은 MTU 값보다 **100** 미만으로 설정해야 합니다. 예를 들어, 클러스터의 일부 노드에 **9001**의 MTU가 있고 일부에는 **1500**의 MTU가 있는 경우 이 값을 **1400**으로 설정해야 합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
genevePort	**integer**	모든 Geneve 패킷에 사용할 포트입니다. 기본값은 **6081**입니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
ipsecConfig	**object**	IPsec 암호화를 활성화하려면 빈 오브젝트를 지정합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
policyAuditConfig	**object**	네트워크 정책 감사 로깅을 사용자 정의할 구성 오브젝트를 지정합니다. 설정되지 않으면 기본값 감사 로그 설정이 사용됩니다.

표 8.29. policyAuditConfig object

필드	유형	설명
rateLimit	integer	노드당 1초마다 생성할 최대 메시지 수입니다. 기본값은 초당 **20** 개의 메시지입니다.
maxFileSize	integer	감사 로그의 최대 크기(바이트)입니다. 기본값은 **50000000 또는 50** MB입니다.
대상	string	다음 추가 감사 로그 대상 중 하나입니다. **libc** 호스트에서 journald 프로세스의 libc **syslog()** 함수입니다. **udp:<host>:<port>** syslog 서버입니다. **<host>:<port>**를 syslog 서버의 호스트 및 포트로 바꿉니다. **unix:<file>** **<file>**로 지정된 Unix Domain Socket 파일입니다. **null** 감사 로그를 추가 대상으로 보내지 마십시오.

필드	유형	설명
syslogFacility	string	RFC5424에 정의된 **kern**과 같은 syslog 기능입니다. 기본값은 **local0**입니다.

OVN-Kubernetes 구성 예

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
    ipsecConfig: {}
```

kubeProxyConfig 오브젝트 구성

kubeProxyConfig 오브젝트의 값은 다음 표에 정의되어 있습니다.

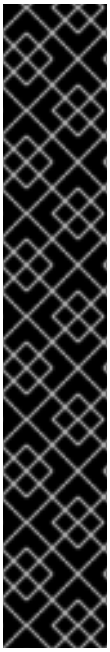
표 8.30. kubeProxyConfig object

필드	유형	설명
iptablesSyncPeriod	**string**	**iptables** 규칙의 새로 고침 간격입니다. 기본값은 **30s**입니다. 유효 접미사로 **s**, **m**, **h**가 있으며, 자세한 설명은 Go **time** 패키지 문서를 참조하십시오. **참고** OpenShift Container Platform 4.3 이상에서는 성능이 개선되어 더 이상 **iptablesSyncPeriod** 매개변수를 조정할 필요가 없습니다.

필드	유형	설명
proxyArguments.iptables-min-sync-period	**array**	**iptables** 규칙을 새로 고치기 전 최소 기간입니다. 이 필드를 통해 새로 고침 간격이 너무 짧지 않도록 조정할 수 있습니다. 유효 접미사로 **s**, **m**, **h**가 있으며, 자세한 설명은 Go **time** 패키지를 참조하십시오. 기본값은 다음과 같습니다. kubeProxyConfig: proxyArguments: iptables-min-sync-period: - 0s

8.3.13. Ignition 구성 파일 생성

클러스터 시스템을 수동으로 시작해야 하므로 클러스터가 시스템을 생성하는 데 필요한 **Ignition** 구성 파일을 사용자가 생성해야 합니다.



중요

- 설치 프로그램에서 생성하는 **Ignition** 구성 파일에 **24**시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 **24**시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 **kubelet** 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(**CSR**)을 수동으로 승인해야 합니다. 자세한 내용은 ***Recovering from expired control plane certificates*** 문서를 참조하십시오.
- 클러스터를 설치한 후 **24**시간에서 **22**시간까지의 인증서가 교체되기 때문에 생성된 후 **12**시간 이내에 **Ignition** 구성 파일을 사용하는 것이 좋습니다. **12**시간 이내에 **Ignition** 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

사전 요구 사항

- **OpenShift Container Platform** 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

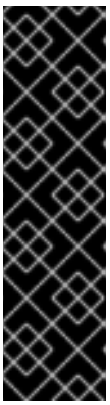
프로세스

- **Ignition** 구성 파일을 가져옵니다.

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1

<installation_directory>는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.



중요

install-config.yaml 파일을 생성한 경우 파일이 포함된 디렉터리를 지정하십시오. 그렇지 않으면 빈 디렉터리를 지정합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

다음 파일이 디렉터리에 생성됩니다.

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

8.3.14. RHCOS 설치 및 OpenShift Container Platform 부트스트랩 프로세스 시작

프로비저닝하는 베어메탈 인프라에 OpenShift Container Platform을 설치하려면 머신에 RHCOS(Red Hat Enterprise Linux CoreOS)를 설치해야 합니다. RHCOS를 설치할 때 설치 중인 머신 유형에 대해 OpenShift Container Platform 설치 프로그램에서 생성한 Ignition 구성 파일을 제공해야 합니다. 적합한 네트워킹, DNS 및 로드 밸런싱 인프라를 구성한 경우 RHCOS 머신이 재부팅된 후 OpenShift Container Platform 부트스트랩 프로세스가 자동으로 시작됩니다.

단계에 따라 ISO 이미지 또는 네트워크 PXE 부팅을 사용하여 시스템에 RHCOS를 설치합니다.

참고

이 설치 문서에 포함된 컴퓨팅 노드 배포 단계는 RHCOS에 따라 다릅니다. RHEL 기반 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. RHEL 7 컴퓨팅 머신 사용은 더 이상 사용되지 않으며 향후 OpenShift Container Platform 4 릴리스에서 제거될 예정입니다.

다음 방법을 사용하여 ISO 및 PXE 설치 중에 RHCOS를 구성할 수 있습니다.

- 커널 인수: 커널 인수를 사용하여 설치 관련 정보를 제공할 수 있습니다. 예를 들어 HTTP 서버에 업로드한 RHCOS 설치 파일의 위치와 설치 중인 노드 유형에 대한 Ignition 구성 파일의 위치를 지정할 수 있습니다. PXE 설치의 경우 APPEND 매개 변수를 사용하여 라이브 설치 프로그램의 커널에 인수를 전달할 수 있습니다. ISO 설치의 경우는 라이브 설치 부팅 프로세스를 중단하고 커널 매개 변수를 추가할 수 있습니다. 두 경우 모두 특정 coreos.inst.* 인수를 사용하여 라이브 설치 프로그램을 지시 할 수 있을 뿐 만 아니라 표준 커널 서비스를 활성화/비활성화하기 위해 표준 설치 부팅 인수를 사용할 수 있습니다.

- Ignition 구성: OpenShift Container Platform Ignition 구성 파일(*.ign)은 설치 중인 노드 유형에 따라 다릅니다. RHCOS 설치 중에 부트스트랩, 컨트롤 플레인 또는 컴퓨팅 노드 Ignition 구성 파일의 위치를 전달하여 첫 번째 부팅 시 적용됩니다. 특별한 경우에는 라이브 시스템으로 전달할 별도의 제한된 Ignition 설정을 만들 수 있습니다. 이 Ignition 설정은 설치 완료 후 프로비저닝 시스템에 설치가 성공적으로 완료되었는지를 보고하는 것과 같은 일련의 작업을 수행할 수 있습니다. 이러한 특수 Ignition 구성은 설치된 시스템의 처음 부팅 시 적용되는 coreos-installer에 의해 소비됩니다. 라이브 ISO에 표준 컨트롤 플레인 및 컴퓨팅 노드 Ignition 구성을 직접 제공하지 마십시오.

- coreos-installer : 처음 부팅하기 전에 다양한 방법으로 영구 시스템을 준비 할 수 있도록 쉘 프롬프트에서 라이브 ISO 설치 프로그램을 시작할 수 있습니다. coreos-installer 명령을 실행하여 추가하는 다양한 아티팩트를 식별하고 디스크 파티션을 사용하여 네트워크를 설정할 수 있습니다. 경우에 따라 라이브 시스템에서 기능을 구성하고 설치된 시스템에 복사할 수도 있습니다.

ISO 또는 PXE 설치 사용 여부는 상황에 따라 달라집니다. PXE 설치에는 사용 가능한 DHCP 서비스와 추가 준비가 필요하지만 설치 프로세스를 보다 자동화할 수 있습니다. ISO 설치는 주로 수동적인 프로세스에서 여러 시스템을 설정하는 경우 불편할 수 있습니다.

참고

OpenShift Container Platform 4.6부터 RHCOS ISO 및 기타 설치 아티팩트는 4K 섹터 디스크에 설치를 지원합니다.

8.3.14.1. ISO 이미지를 사용하여 RHCOS 설치

ISO 이미지를 사용하여 시스템에 RHCOS을 설치할 수 있습니다.

사전 요구 사항

- 클러스터에 대한 Ignition 구성 파일이 생성되어 있습니다.
- 적합한 네트워크, DNS 및 로드 밸런싱 인프라가 구성되어 있습니다.
- 사용자 컴퓨터에서 액세스할 수 있고 사용자가 생성한 시스템에서 액세스할 수 있는 HTTP 서버에 대한 액세스 권한을 확보합니다.
- 네트워킹 및 디스크 파티션과 같은 기능을 구성하는 다양한 방법에 대해서는 *고급 RHCOS 설치 구성*섹션을 살펴보십시오.

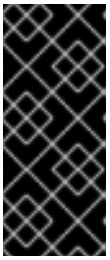
프로세스

1. 각 Ignition 구성 파일에 대해 SHA512 다이제스트를 가져옵니다. 예를 들어 Linux를 실행하는 시스템에서 다음을 사용하여 bootstrap.ign Ignition 구성 파일의 SHA512 다이제스트를 가져올 수 있습니다.

```
$ sha512sum <installation_directory>/bootstrap.ign
```

   다이제스트는 클러스터 노드에서 Ignition 구성 파일의 신뢰성을 검증하기 위해 이후 단계에서 coreos-installer에 제공됩니다.

2. 설치 프로그램에서 생성한 부트스트랩, 컨트롤 플레인, 컴퓨팅 노드 Ignition 구성 파일을 HTTP 서버에 업로드합니다. 해당 파일의 URL을 기록해 둡니다.



   중요

   HTTP 서버에 저장하기 전에 Ignition 구성에서 구성 설정을 추가하거나 변경할 수 있습니다. 설치를 마친 후 클러스터에 컴퓨팅 시스템을 더 추가하려면 Ignition 구성 파일을 삭제하지 마십시오.

3.

설치 호스트에서 Ignition 구성 파일을 URL에서 사용할 수 있는지 확인합니다. 다음 예에서는 부트스트랩 노드에 대한 Ignition 구성 파일을 가져옵니다.

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

출력 예

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-
rsa...
```

명령에서 bootstrap.ign을 master.ign 또는 worker.ign으로 교체하여 컨트롤 플레인 및 컴퓨팅 노드의 Ignition 구성 파일도 사용할 수 있는지 확인합니다.

4. RHCOS 이미지 미러 페이지에서 운영 체제 인스턴스 설치 방법에 필요한 RHCOS 이미지를 얻을 수 있지만 올바른 버전의 RHCOS 이미지를 얻는 것이 좋습니다. openshift-install 명령 출력에서 사용할 수 있습니다.

```
$ openshift-install coreos print-stream-json | grep '\.iso[^.]'
```

출력 예

```
"location": "<url>/art/storage/releases/rhcos-4.9-aarch64/<release>/aarch64/rhcos-
<release>-live.aarch64.iso",
"location": "<url>/art/storage/releases/rhcos-4.9-ppc64le/<release>/ppc64le/rhcos-
<release>-live.ppc64le.iso",
"location": "<url>/art/storage/releases/rhcos-4.9-s390x/<release>/s390x/rhcos-
<release>-live.s390x.iso",
"location": "<url>/art/storage/releases/rhcos-4.9/<release>/x86_64/rhcos-<release>-
live.x86_64.iso",
```

중요

RHCOS 이미지는 OpenShift Container Platform 릴리스에 따라 변경되지 않을 수 있습니다. 설치하는 OpenShift Container Platform 버전과 같거나 그 이하의 버전 중 가장 최신 버전의 이미지를 다운로드해야 합니다. 사용 가능한 경우 OpenShift Container Platform 버전과 일치하는 이미지 버전을 사용합니다. 이 프로세스에는 ISO 이미지만 사용하십시오. 이 설치 유형에서는 RHCOS qcow2 이미지가 지원되지 않습니다.

ISO 파일 이름은 다음 예와 유사합니다.

```
rhcos-<version>-live.<architecture>.iso
```

5. ISO를 사용하여 RHCOS 설치를 시작합니다. 다음 설치 옵션 중 하나를 사용합니다.

- ISO 이미지를 디스크에 굽고 직접 부팅합니다.
- LOM(Lightweight-out Management) 인터페이스를 사용하여 ISO 리디렉션을 사용합니다.

6. 옵션을 지정하거나 라이브 부팅 시퀀스를 중단하지 않고 RHCOS ISO 이미지를 부팅합니다. 설치 프로그램이 RHCOS 라이브 환경에서 쉘 프롬프트로 부팅될 때까지 기다립니다.

참고

커널 인수를 추가하기 위해 RHCOS 설치 부팅 프로세스를 중단할 수 있습니다. 하지만 이 ISO 프로세스에서는 커널 인수를 추가하지 않고 다음 단계에 설명된 대로 coreos-installer 명령을 사용해야 합니다.

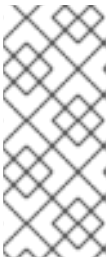
7. coreos-installer 명령을 실행하고 설치 요구 사항을 충족하는 옵션을 지정합니다. 최소한 노드 유형에 대한 Ignition 구성 파일과 설치할 장치를 가리키는 URL을 지정해야 합니다.

```
$ sudo coreos-installer install --ignition-url=http://<HTTP_server>/<node_type>.ign
<device> --ignition-hash=sha512-<digest> 1 2
```

1 1

2

클러스터 노드에서 Ignition 구성 파일을 HTTP URL을 통해 가져오려면 --ignition-hash 옵션이 필요합니다. <digest>는 이전 단계에서 얻은 Ignition 구성 파일 SHA512 다이제스트입니다.



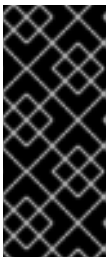
참고

TLS를 사용하는 HTTPS 서버를 통해 Ignition 구성 파일을 제공하려는 경우 coreos-installer를 실행하기 전에 내부 인증 기관(CA)을 시스템 신뢰 저장소에 추가할 수 있습니다.

다음 예제에서는 /dev/sda 장치에 부트스트랩 노드 설치를 초기화합니다. 부트스트랩 노드의 Ignition 구성 파일은 IP 주소 192.168.1.2가 있는 HTTP 웹 서버에서 가져옵니다.

```
$ sudo coreos-installer install --ignition-
url=http://192.168.1.2:80/installation_directory/bootstrap.ign /dev/sda --ignition-
hash=sha512-
a5a2d43879223273c9b60af66b44202a1d1248fc01cf156c46d4a79f552b6bad47bc8cc78dd
f0116e80c59d2ea9e32ba53bc807afbca581aa059311def2c3e3b
```

8. 머신 콘솔에서 RHCOS 설치 진행률을 모니터링합니다.



중요

OpenShift Container Platform 설치를 시작하기 전에 각 노드에서 성공적으로 설치되었는지 확인합니다. 설치 프로세스를 관찰하면 발생할 수 있는 RHCOS 설치 문제의 원인을 파악하는 데 도움이 될 수 있습니다.

9. RHCOS를 설치한 후 시스템을 재부팅해야 합니다. 시스템이 재부팅되는 동안 지정한 Ignition 구성 파일이 적용됩니다.

10. 계속해서 클러스터에 대한 나머지 시스템을 모두 생성합니다.

중요

이때 부트스트랩 및 컨트롤 플레인 시스템을 생성해야 합니다. 컨트롤 플레인 시스템이 예약 가능하지 않은 경우 **OpenShift Container Platform**을 설치하기 전에 두 개 이상의 컴퓨팅 시스템도 생성합니다.

적합한 네트워킹, **DNS** 및 로드 밸런싱 인프라가 있는 경우 **RHCOS** 노드가 재부팅된 후 **OpenShift Container Platform** 부트스트랩 프로세스가 자동으로 시작됩니다.

참고

RHCOS 노드에는 **core** 사용자의 기본 암호가 포함되지어 있지 않습니다. **install_config.yaml** 파일에 지정한 공개 키에 대한 액세스 권한이 있는 사용자로 **ssh core@<node>.<cluster_name>.<base_domain>**을 실행하여 노드에 액세스할 수 있습니다. **RHCOS**를 실행하는 **OpenShift Container Platform 4** 클러스터 노드는 변경할 수 없으며 **Operator**를 통해 클러스터 변경 사항을 적용합니다. **SSH**를 사용하여 클러스터 노드에 액세스하는 것은 권장되지 않습니다. 그러나 설치 문제를 조사할 때 **OpenShift Container Platform API**를 사용할 수 없거나 **kubelet**이 대상 노드에서 제대로 작동하지 않는 경우 디버깅 또는 재해 복구에 **SSH** 액세스가 필요할 수 있습니다.

8.3.14.2. PXE 또는 iPXE 부팅을 사용하여 RHCOS 설치

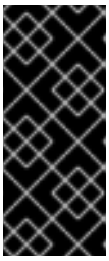
PXE 또는 **iPXE** 부팅을 사용하여 시스템에 **RHCOS**를 설치할 수 있습니다.

사전 요구 사항

- 클러스터에 대한 **Ignition** 구성 파일이 생성되어 있습니다.
- 적합한 네트워크, **DNS** 및 로드 밸런싱 인프라가 구성되어 있습니다.
- 적합한 **PXE** 또는 **iPXE** 인프라가 구성되어 있습니다.
- 사용자 컴퓨터에서 액세스할 수 있고 사용자가 생성한 시스템에서 액세스할 수 있는 **HTTP** 서버에 대한 액세스 권한을 확보합니다.
- 네트워킹 및 디스크 파티션과 같은 기능을 구성하는 다양한 방법에 대해서는 *고급 **RHCOS** 설치 구성*섹션을 살펴보십시오.

프로세스

1. 설치 프로그램에서 생성한 부트스트랩, 컨트롤 플레인, 컴퓨팅 노드 Ignition 구성 파일을 HTTP 서버에 업로드합니다. 해당 파일의 URL을 기록해 둡니다.



중요

HTTP 서버에 저장하기 전에 Ignition 구성에서 구성 설정을 추가하거나 변경할 수 있습니다. 설치를 마친 후 클러스터에 컴퓨팅 시스템을 더 추가하려면 Ignition 구성 파일을 삭제하지 마십시오.

2. 설치 호스트에서 Ignition 구성 파일을 URL에서 사용할 수 있는지 확인합니다. 다음 예에서는 부트스트랩 노드에 대한 Ignition 구성 파일을 가져옵니다.

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

출력 예

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-
rsa...
```

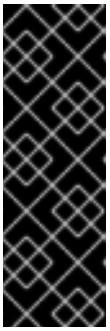
명령에서 bootstrap.ign을 master.ign 또는 worker.ign으로 교체하여 컨트롤 플레인 및 컴퓨팅 노드의 Ignition 구성 파일도 사용할 수 있는지 확인합니다.

3. RHCOS 커널,initramfs 및 rootfs 파일을 RHCOS 이미지 미러 페이지에서 원하는 운영 체제 인스턴스를 설치하는 데 필요한 경우, RHCOS 파일의 올바른 버전을 가져오는 권장 방법은 openshift-install 명령 출력에서 얻을 수 있습니다.

```
$ openshift-install coreos print-stream-json | grep -Eo '"https.*(kernel-
|initramfs.|rootfs.)\w+(\.img)?"'
```

출력 예

```
"<url>/art/storage/releases/rhcos-4.9-aarch64/<release>/aarch64/rhcos-<release>-live-kernel-aarch64"
"<url>/art/storage/releases/rhcos-4.9-aarch64/<release>/aarch64/rhcos-<release>-live-initramfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.9-aarch64/<release>/aarch64/rhcos-<release>-live-rootfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.9-ppc64le/49.84.202110081256-0/ppc64le/rhcos-<release>-live-kernel-ppc64le"
"<url>/art/storage/releases/rhcos-4.9-ppc64le/<release>/ppc64le/rhcos-<release>-live-initramfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.9-ppc64le/<release>/ppc64le/rhcos-<release>-live-rootfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.9-s390x/<release>/s390x/rhcos-<release>-live-kernel-s390x"
"<url>/art/storage/releases/rhcos-4.9-s390x/<release>/s390x/rhcos-<release>-live-initramfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.9-s390x/<release>/s390x/rhcos-<release>-live-rootfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.9/<release>/x86_64/rhcos-<release>-live-kernel-x86_64"
"<url>/art/storage/releases/rhcos-4.9/<release>/x86_64/rhcos-<release>-live-initramfs.x86_64.img"
"<url>/art/storage/releases/rhcos-4.9/<release>/x86_64/rhcos-<release>-live-rootfs.x86_64.img"
```



중요

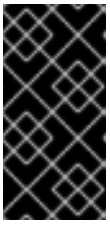
OpenShift Container Platform의 모든 릴리스에서 **RHCOS** 아티팩트가 변경되지 않을 수도 있습니다. 설치하는 **OpenShift Container Platform** 버전과 같거나 그 이하의 버전 중 가장 최신 버전의 이미지를 다운로드해야 합니다. 이 프로시저에는 아래 설명된 적절한 **kernel, initramfs** 및 **rootfs** 아티팩트만 사용하십시오. 이 설치 유형에서는 **RHCOS QCOW2** 이미지가 지원되지 않습니다.

OpenShift Container Platform 버전 번호가 파일 이름에 포함됩니다. 다음 예와 유사합니다.

- **kernel: rhcos-<version>-live-kernel-<architecture>**
- **initramfs: rhcos-<version>-live-initramfs.<architecture>.img**

- **rootfs: rhcos-<version>-live-rootfs.<architecture>.img**

4. **rootfs, kernel** 및 **initramfs** 파일을 **HTTP** 서버에 업로드합니다.



중요

설치를 마친 후 클러스터에 컴퓨팅 시스템을 더 추가하려면 **Ignition** 구성 파일을 삭제하지 마십시오.

5. **RHCOS**가 설치된 후 시스템이 로컬 디스크에서 부팅되도록 네트워크 부팅 인프라를 구성하십시오.

6. **RHCOS** 이미지에 대한 **PXE** 또는 **iPXE** 설치를 구성하고 설치를 시작하십시오.

사용 환경에 대한 다음 예시 메뉴 항목 중 하나를 수정하고, 이미지 및 **Ignition** 파일에 적절히 접근할 수 있는지 확인하십시오.

- **PXE**의 경우:

```
DEFAULT pxeboot
TIMEOUT 20
PROMPT 0
LABEL pxeboot
    KERNEL http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> (1)
    APPEND initrd=http://<HTTP_server>/rhcos-<version>-live-initramfs.
<architecture>.img coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-
rootfs.<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign (2) (3)
```

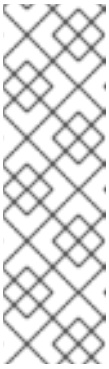
(1) (1)

HTTP 서버에 업로드한 라이브 **kernel** 파일의 위치를 지정합니다. **URL**은 **HTTP**, **TFTP** 또는 **FTP**여야 합니다. **HTTPS**와 **NFS**는 지원되지 않습니다.

(2)

NIC를 여러 개 사용하는 경우 **ip** 옵션에 단일 인터페이스를 지정합니다. 예를 들어, **eno1**라는 **NIC**에서 **DHCP**를 사용하려면 **ip=eno1:dhcp**를 설정하십시오.

(3)



참고

이 구성은 그래픽 콘솔이 있는 시스템에서 직렬 콘솔 액세스를 활성화하지 않습니다. 다른 콘솔을 구성하려면 **APPEND** 행에 하나 이상의 **console=** 인수를 추가합니다. 예를 들어 **console=tty0 console=ttyS0**을 추가하여 첫 번째 **PC** 직렬 포트를 기본 콘솔로 설정하고 그래픽 콘솔을 보조 콘솔로 설정합니다. 자세한 내용은 **Red Hat Enterprise Linux에서 직렬 터미널 및/또는 콘솔 설정 방법**을 참조하십시오.

- **iPXE**의 경우 :

```
kernel http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> initrd=main
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign 1 2
initrd --name main http://<HTTP_server>/rhcos-<version>-live-initramfs.
<architecture>.img 3
boot
```

1

HTTP 서버에 업로드한 **RHCOS** 파일의 위치를 지정합니다. **kernel** 매개변수 값은 **kernel** 파일의 위치이고 **initrd=main** 인수는 **UEFI** 시스템에서 부팅하는 데 필요하며 **coreos.live.rootfs_url** 매개 변수 값은 **rootfs** 파일의 위치이며, **coreos.inst.ignition_url** 매개 변수 값은 부트스트랩 **Ignition** 설정 파일의 위치입니다.

2

NIC를 여러 개 사용하는 경우 **ip** 옵션에 단일 인터페이스를 지정합니다. 예를 들어, **eno1**라는 **NIC**에서 **DHCP**를 사용하려면 **ip=eno1:dhcp**를 설정하십시오.

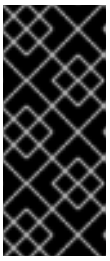
3

HTTP 서버에 업로드한 **initramfs** 파일의 위치를 지정합니다.

참고

이 구성은 그래픽 콘솔이 있는 시스템에서 직렬 콘솔 액세스를 활성화하지 않습니다. 다른 콘솔을 구성하려면 **kernel** 행에 하나 이상의 **console=** 인수를 추가합니다. 예를 들어 **console=tty0 console=ttyS0**을 추가하여 첫 번째 **PC** 직렬 포트를 기본 콘솔로 설정하고 그래픽 콘솔을 보조 콘솔로 설정합니다. 자세한 내용은 **Red Hat Enterprise Linux**에서 직렬 터미널 및/또는 콘솔 설정 방법을 참조하십시오.

7. 머신 콘솔에서 **RHCOS** 설치 진행률을 모니터링합니다.



중요

OpenShift Container Platform 설치를 시작하기 전에 각 노드에서 성공적으로 설치되었는지 확인합니다. 설치 프로세스를 관찰하면 발생할 수 있는 **RHCOS** 설치 문제의 원인을 파악하는 데 도움이 될 수 있습니다.

8. **RHCOS**가 설치되면 시스템을 재부팅합니다. 시스템이 재부팅되는 동안 지정한 **Ignition** 구성 파일이 적용됩니다.

9. 클러스터용 시스템 생성을 계속합니다.

중요

이때 부트스트랩 및 컨트롤 플레인 시스템을 생성해야 합니다. 컨트롤 플레인 시스템이 예약 가능하지 않은 경우 클러스터를 설치하기 전에 두 개 이상의 컴퓨팅 시스템도 생성합니다.

적합한 네트워킹, **DNS** 및 로드 밸런싱 인프라가 있는 경우 **RHCOS** 노드가 재부팅된 후 **OpenShift Container Platform** 부트스트랩 프로세스가 자동으로 시작됩니다.

참고

RHCOS 노드에는 core 사용자의 기본 암호가 포함되지어 있지 않습니다. install_config.yaml 파일에 지정한 공개 키에 대한 액세스 권한이 있는 사용자로 ssh core@<node>.<cluster_name>.<base_domain>을 실행하여 노드에 액세스할 수 있습니다. RHCOS를 실행하는 OpenShift Container Platform 4 클러스터 노드는 변경할 수 없으며 Operator를 통해 클러스터 변경 사항을 적용합니다. SSH를 사용하여 클러스터 노드에 액세스하는 것은 권장되지 않습니다. 그러나 설치 문제를 조사할 때 OpenShift Container Platform API를 사용할 수 없거나 kubelet이 대상 노드에서 제대로 작동하지 않는 경우 디버깅 또는 재해 복구에 SSH 액세스가 필요할 수 있습니다.

8.3.14.3. 고급 RHCOS 설치 구성 옵션

OpenShift Container Platform용 RHCOS (Red Hat Enterprise Linux CoreOS) 노드를 수동으로 프로비저닝하는 주요 이점은 기본 OpenShift Container Platform 설치 방법을 통해 사용할 수 없는 구성을 수행할 수 있는 것입니다. 이 섹션에서는 다음과 같은 방법을 사용하여 수행할 수있는 몇 가지 구성에 대해 설명합니다.

- 라이브 설치 프로그램에 커널 인수 전달
- 라이브 시스템에서 수동으로 coreos-installer 실행
- ISO에 Ignition 구성 포함

이 섹션에 설명되어 있는 수동 Red Hat Enterprise Linux CoreOS(RHCOS) 설치에 대한 고급 구성 항목은 디스크 파티션 설정, 네트워킹 및 다양한 방식의 Ignition 구성 사용과 관련되어 있습니다.

8.3.14.3.1. PXE 및 ISO 설치를 위한 고급 네트워크 옵션 사용

OpenShift Container Platform 노드의 네트워크는 기본적으로 DHCP를 사용하여 필요한 모든 구성 설정을 수집합니다. 고정 IP 주소를 설정하거나 본딩과 같은 특정 설정을 구성하려면 다음 중 하나의 방법으로 수행할 수 있습니다.

- 라이브 설치 프로그램을 시작할 때 특수 커널 매개 변수를 전달합니다.
- 머신 구성을 사용하여 네트워크 파일을 설치된 시스템에 복사합니다.

- 라이브 설치 프로그램 쉘 프롬프트에서 네트워크를 구성한 다음 설치된 시스템에 복사하여 설치한 시스템을 처음 시작할 때 사용하도록합니다.

PXE 또는 **iPXE** 설치를 구성하려면 다음 옵션 중 하나를 사용합니다.

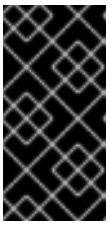
- "고급 **RHCOS** 설치 참조" 표를 참조하십시오.
- 머신 구성을 사용하여 네트워크 파일을 설치된 시스템에 복사합니다.

다음 프로세스에 따라 **ISO** 설치를 구성합니다.

절차

1. **ISO** 설치 프로그램을 시작합니다.
2. 라이브 시스템 쉘 프롬프트에서 사용 가능한 **RHEL** 도구 (예: **nmcli** 또는 **nmtui**)를 사용하여 라이브 시스템의 네트워킹를 구성합니다.
3. **coreos-installer** 명령을 실행하여 시스템을 설치하고 **--copy-network** 옵션을 추가하여 네트워크 구성을 복사합니다. 예를 들면 다음과 같습니다.

```
$ sudo coreos-installer install --copy-network \
    --ignition-url=http://host/worker.ign /dev/sda
```



중요

copy-network 옵션은 **/etc/NetworkManager/system-connections**에 있는 네트워킹 구성만 복사합니다. 특히 시스템 호스트 이름을 복사하지 않습니다.

4. 설치된 시스템으로 재부팅하십시오.

추가 리소스

- **nmcli** 및 **nmtui** 툴에 대한 자세한 내용은 **RHEL 8** 설명서에서 nmcli로 시작하기 및 nmtui

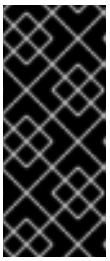
로 시작하기를 참조하십시오.

8.3.14.3.2. 디스크 파티션 설정

디스크 파티션은 RHCOS(Red Hat Enterprise Linux CoreOS) 설치 중에 OpenShift Container Platform 클러스터 노드에 생성됩니다. 특정 아키텍처의 각 RHCOS 노드는 기본 파티션 구성을 재정의하지 않는 한 동일한 파티션 레이아웃을 사용합니다. RHCOS 설치 중에 대상 장치에서 사용 가능한 나머지 공간을 사용하도록 루트 파일 시스템의 크기가 증가합니다.

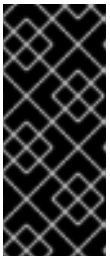
OpenShift Container Platform 클러스터 노드에 RHCOS를 설치할 때 다음과 같은 두 가지 경우 기본 파티셔닝을 재정의할 수 있습니다.

- 별도의 파티션 생성: 빈 디스크에 그린 필드 설치의 경우 파티션에 별도의 스토리지를 추가할 수 있습니다. 이는 /var 또는 /var의 하위 디렉터리 (예: /var/lib/etcd)중 하나를 별도의 파티션으로 만드는 경우에 공식적으로 지원되지만 둘 다 지원되지는 않습니다.



중요

100GB 이상의 디스크 크기 및 특히 1TB보다 큰 디스크 크기의 경우 별도의 /var 파티션을 만듭니다. 자세한 내용은 " 별도의 /var 파티션 생성" 및 이 Red Hat 지식베이스 문서 를 참조하십시오.



중요

Kubernetes는 두 개의 파일 시스템 파티션만 지원합니다. 원래 구성에 둘 이상의 파티션을 추가하는 경우 Kubernetes는 모든 파티션을 모니터링할 수 없습니다.

- 기존 파티션 유지: 브라운 필드 설치에서 기존 노드에 OpenShift Container Platform을 재설치하고 이전 운영 체제에 설치된 데이터 파티션을 유지해야 하는 경우 기존 데이터 파티션을 유지할 수 있는 coreos-installer에 부팅 인수와 옵션이 모두 있습니다.

주의

사용자 지정 파티션을 사용하면 **OpenShift Container Platform**에서 해당 파티션을 모니터링하지 않거나 경고를 받을 수 있습니다. 기본 파티션을 재정의하는 경우 **OpenShift Container Platform**이 호스트 파일 시스템을 모니터링하는 방법에 대한 자세한 내용은 **OpenShift File System Monitoring(제거 조건) 이해**를 참조하십시오.

8.3.14.3.2.1. 별도의 /var 파티션 만들기

일반적으로 **RHCOS** 설치 중에 생성된 기본 디스크 파티션을 사용해야 합니다. 그러나 확장하려는 디렉토리에 별도의 파티션을 생성해야 하는 경우가 있습니다.

OpenShift 컨테이너 플랫폼은 **/var** 디렉토리 또는 **/var**의 하위 디렉터리 중 하나에 스토리지를 연결하는 단일 파티션의 추가를 지원합니다. 예를 들면 다음과 같습니다.

- **/var/lib/containers:** 시스템에 더 많은 이미지와 컨테이너가 추가됨에 따라 확장될 수 있는 컨테이너 관련 콘텐츠를 보관합니다.
- **/var/lib/etcd: etcd** 스토리지의 성능 최적화와 같은 목적으로 별도로 보관할 데이터를 보관합니다.
- **/var:** 감사 등의 목적에 맞게 별도로 분리하여 보관해야 하는 데이터를 보관합니다.

중요

100GB보다 큰 디스크 크기 및 특히 **1TB**보다 큰 디스크의 경우 별도의 **/var** 파티션을 만듭니다.

/var 디렉터리의 콘텐츠를 별도로 저장하면 필요에 따라 해당 영역에 대한 스토리지 확장을 보다 용이하게 하고 나중에 **OpenShift Container Platform**을 다시 설치하여 해당 데이터를 그대로 보존할 수 있습니다. 이 방법을 사용하면 모든 컨테이너를 다시 가져올 필요가 없으며 시스템을 업데이트할 때 대용량 로그 파일을 복사할 필요도 없습니다.

/var 디렉토리 또는 /var의 하위 디렉토리에 대해 별도의 파티션을 사용하면 분할된 디렉토리의 데이터 증가로 루트 파일 시스템이 채워지는 것을 방지할 수 있습니다.

다음 절차에서는 설치 준비 단계에서 노드 유형의 Ignition 구성 파일에 래핑되는 머신 구성 매니페스트를 추가하여 별도의 /var 파티션을 설정합니다.

절차

1. 설치 호스트에서 OpenShift Container Platform 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터에 대한 Kubernetes 매니페스트를 생성합니다.

```
$ openshift-install create manifests --dir <installation_directory>
```

2. 추가 파티션을 구성하는 Butane 구성을 생성합니다. 예를 들어 파일 이름을 $HOME/clusterconfig/98-var-partition.bu 로 지정하고, 디스크 장치 이름을 worker 시스템의 스토리지 장치 이름으로 변경하고 스토리지 크기를 적절하게 설정합니다. 이 예에서는 /var 디렉토리를 별도의 파티션에 배치합니다.

```
variant: openshift
version: 4.9.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1 파티션을 설정해야하는 디스크 저장 장치 이름입니다.

2 데이터 파티션을 부트 디스크에 추가할 때 최소 오프셋 값 25000 메비 바이트가 권장됩니다. 루트 파일 시스템은 지정된 오프셋까지 사용 가능한 모든 공간을 채우기 위해 자동으로 크기가 조정됩니다. 오프셋 값이 지정되지 않거나 지정된 값이 권장 최소값보다 작으면

생성되는 루트 파일 시스템의 크기가 너무 작아지고 **RHCOS**를 나중에 다시 설치할 때 데이터 파티션의 첫 번째 부분을 덮어 쓸 수 있습니다.

3

데이터 파티션의 크기(**MB**)입니다.

4

컨테이너 스토리지에 사용되는 파일 시스템에 **prjquota** 마운트 옵션을 활성화해야 합니다.

참고

별도의 **/var** 파티션을 만들 때 다른 인스턴스 유형에 동일한 장치 이름이 없는 경우 컴퓨팅 노드에 다른 인스턴스 유형을 사용할 수 없습니다.

3. **Butane** 구성에서 매니페스트를 생성하여 **clusterconfig/openshift** 디렉터리에 저장합니다. 예를 들어 다음 명령을 실행합니다.

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o
$HOME/clusterconfig/openshift/98-var-partition.yaml
```

4. **Ignition** 구성 파일을 만듭니다.

```
$ openshift-install create ignition-configs --dir <installation_directory> 1
```

1

<installation_directory>는 동일한 설치 디렉터리를 지정합니다.

설치 디렉터리의 부트스트랩, 컨트롤 플레인 및 컴퓨팅 노드에 대한 **Ignition** 구성 파일이 생성됩니다.

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
```

```
├── master.ign
├── metadata.json
└── worker.ign
```

<installation_directory>/manifest 및 **<installation_directory>/openshift** 디렉터리의 파일은 **98-var-partition** 사용자 정의 **MachineConfig** 오브젝트가 포함된 파일을 포함하여 **Ignition** 구성 파일로 래핑됩니다.

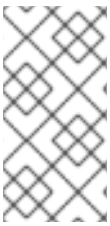
다음 단계

- **RHCOS** 설치 중에 **Ignition** 구성 파일을 참조하여 사용자 정의 디스크 파티션을 적용할 수 있습니다.

8.3.14.3.2.2. 기존 파티션 유지

ISO 설치의 경우 설치 프로그램이 하나 이상의 기존 파티션을 유지하도록하는 옵션을 **coreos-installer** 명령에 추가할 수 있습니다. **PXE** 설치의 경우 **coreos.inst.*** 옵션을 **APPEND** 매개 변수에 추가하여 파티션을 유지할 수 있습니다.

저장된 파티션은 기존 **OpenShift Container Platform** 시스템의 데이터 파티션이 될 수 있습니다. 파티션 레이블 또는 번호 중 하나로 보관하려는 디스크 파티션을 확인할 수 있습니다.



참고

기존 파티션을 저장하고 해당 파티션이 **RHCOS**를 위한 충분한 공간을 남겨 두지 않으면 저장된 파티션이 손상되지는 않지만 설치에 실패합니다.

ISO 설치 중 기존 파티션 유지

이 예에서는 파티션 레이블이 **data (data*)**로 시작하는 모든 파티션을 유지합니다.

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign \
    --save-partlabel 'data*' /dev/sda
```

다음 예는 디스크의 여섯 번째 (6) 파티션을 유지하는 방식으로 **coreos-installer**를 실행하는 방법을 보여줍니다.

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign \
    --save-partindex 6 /dev/sda
```

이 예에서는 파티션 5 이상을 유지합니다.

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign
      --save-partindex 5- /dev/sda
```

파티션 저장이 사용된 이전 예에서 coreos-installer는 파티션을 즉시 다시 만듭니다.

PXE 설치 중 기존 파티션 유지

이 APPEND 옵션은 파티션 레이블이 'data'('data *')로 시작하는 모든 파티션을 유지합니다.

```
coreos.inst.save_partlabel=data*
```

이 APPEND 옵션은 파티션 5 이상을 유지합니다.

```
coreos.inst.save_partindex=5-
```

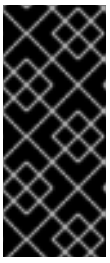
이 APPEND 옵션은 파티션 6을 유지합니다.

```
coreos.inst.save_partindex=6
```

8.3.14.3.3. Ignition 설정 확인

RHCOS 베어 메탈 설치를 수행할 때 제공할 수 있는 두 가지 유형의 Ignition 구성이 있으며 각 구성을 제공하는 이유도 각각 다릅니다.

- Permanent install Ignition config: 모든 수동 RHCOS 설치는 설치를 수행하기 위해 openshift-installer가 생성한 Ignition 구성 파일 (예: bootstrap.ign, master.ign 및 worker.ign) 중 하나를 전달해야 합니다.



중요

이러한 Ignition 구성 파일을 직접 수정하지 않는 것이 좋습니다. 이전 섹션의 예에 설명된 대로 Ignition 구성 파일로 래핑된 매니페스트 파일을 업데이트할 수 있습니다.

PXE 설치의 경우 coreos.inst.ignition_url= 옵션을 사용하여 APPEND 행에서 Ignition 구

성을 전달합니다. ISO 설치의 경우 쉘 프롬프트에서 ISO를 시작한 후 coreos-installer 명령 줄에서 --ignition-url= 옵션을 사용하여 Ignition 구성을 식별합니다. 두 경우 모두 HTTP 및 HTTPS 프로토콜만 지원됩니다.

- Live install Ignition config :이 유형은 수동으로 작성되어 Red Hat에서 지원되지 않으므로 가능하면 사용하지 않도록 해야 합니다. 이 방법을 사용하면 Ignition 구성이 라이브 설치 미디어로 전달되고 부팅시 즉시 실행되며 RHCOS 시스템이 디스크에 설치되기 전후에 설치 작업을 수행합니다. 이 방법은 시스템 구성을 사용하여 실행할 수 없는 고급 파티션 설정과 같이 한 번만 수행하고 나중에 다시 적용할 필요가 없는 작업의 실행에만 사용해야 합니다.

  PXE 또는 ISO 부팅의 경우 Ignition 설정을 만들고 ignition.config.url= 옵션에 APPEND를 실행하여 Ignition 설정 위치를 확인할 수 있습니다. 또한 ignition.firstboot ignition.platform.id = metal도 추가해야 합니다. 추가하지 않으면 ignition.config.url 옵션이 무시됩니다.

8.3.14.3.3.1. RHCOS ISO에 실시간 설치 Ignition 구성 포함

RHCOS ISO 이미지에 직접 라이브 설치 Ignition 구성을 포함할 수 있습니다. ISO 이미지를 부팅하면 내장된 구성이 자동으로 적용됩니다.

절차

1. 다음 이미지 미러 페이지에서 coreos-installer 바이너리를 다운로드합니다. https://mirror.openshift.com/pub/openshift-v4/clients/coreos-installer/latest/.

2. RHCOS ISO 이미지와 Ignition 구성 파일을 검색하고 이를 액세스 가능한 디렉터리 (예: /mnt)에 복사합니다.

```
# cp rhcos-<version>-live.x86_64.iso bootstrap.ign /mnt/
# chmod 644 /mnt/rhcos-<version>-live.x86_64.iso
```

3. 다음 명령을 실행하여 Ignition 구성을 ISO에 포함합니다.

```
# ./coreos-installer iso ignition embed -i /mnt/bootstrap.ign \
    /mnt/rhcos-<version>-live.x86_64.iso
```

이제 해당 ISO를 사용하여 지정된 라이브 설치 Ignition 구성을 사용하여 RHCOS를 설치할 수 있습니다.

중요

coreos-installer iso ignition embed를 사용하여 bootstrap.ign, master.ign 및 worker.ign 과 같이 openshift-installer에 의해 생성된 파일을 포함하는 것은 지원되지 않으며 권장되지 않습니다.

4. 포함된 Ignition 구성 내용을 표시하고 이를 파일로 보내려면 다음을 실행합니다.

```
# ./coreos-installer iso ignition show /mnt/rhcos-<version>-live.x86_64.iso >
mybootstrap.ign
```

```
# diff -s bootstrap.ign mybootstrap.ign
```

출력 예

```
Files bootstrap.ign and mybootstrap.ign are identical
```

5. Ignition 구성을 제거하고 다시 사용할 수 있도록 ISO를 초기 상태로 복원하려면 다음을 실행합니다.

```
# ./coreos-installer iso ignition remove /mnt/rhcos-<version>-live.x86_64.iso
```

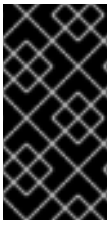
이제 다른 Ignition 구성을 ISO에 포함하거나 초기 상태의 ISO를 사용할 수 있습니다.

8.3.14.3.4. 고급 RHCOS 설치 참조

여기서는 RHCOS(Red Hat Enterprise Linux CoreOS) 수동 설치 프로세스를 수정하는 데 사용할 수 있는 네트워킹 구성 및 기타 고급 옵션에 대해 설명합니다. 다음 표에서는 RHCOS 라이브 설치 프로그램 및 coreos-installer 명령과 함께 사용할 수있는 커널 인수 및 명령 줄 옵션에 대해 설명합니다.

8.3.14.3.4.1. ISO 설치를 위한 네트워킹 및 본딩 옵션

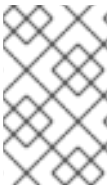
ISO 이미지에서 RHCOS를 설치하는 경우, 해당 이미지를 부팅할 때 수동으로 커널 인수를 추가하여 노드의 네트워킹을 구성할 수 있습니다. 네트워킹 인수를 지정하지 않으면 RHCOS에서 Ignition 구성 파일을 가져오는 데 네트워킹이 필요함을 감지하면 initramfs에서 DHCP가 활성화됩니다.



중요

네트워킹 인수를 수동으로 추가할 때 **initramfs**에서 네트워크를 가져오려면 **rd.neednet=1** 커널 인수도 추가해야 합니다.

다음 표는 **ISO** 설치를 위해 **RHCOS(Red Hat Enterprise Linux CoreOS)** 노드의 네트워킹 및 본딩 구성 예를 보여줍니다. 예제에서는 **ip=, nameserver=, bond=** 커널 인수를 사용하는 방법을 설명합니다.

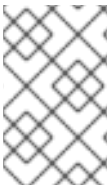


참고

커널 인수를 추가할 때 순서가 중요합니다: **ip=, nameserver=** 및 **bond=** 입니다.

이는 시스템 부팅 중에 **dracut** 툴로 전달되는 네트워킹 옵션입니다. **dracut**에서 지원하는 네트워킹 옵션에 대한 자세한 내용은 **dracut.cmdline** 메뉴얼 페이지를 참조하십시오.

다음 정보는 **ISO** 설치를 위해 **RHCOS** 노드에서 네트워킹을 구성하는 예제를 제공합니다. 예제에서는 **ip=** 및 **nameserver=** 커널 인수를 사용하는 방법을 설명합니다.



참고

커널 인수를 추가할 때 순서가 중요합니다(**ip =** 및 **nameserver= **).

이는 시스템 부팅 중에 **dracut** 툴로 전달되는 네트워킹 옵션입니다. **dracut**에서 지원하는 네트워킹 옵션에 대한 자세한 내용은 **dracut.cmdline** 메뉴얼 페이지를 참조하십시오.

다음 예제는 **ISO** 설치를 위한 네트워킹 옵션입니다.

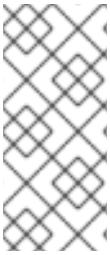
DHCP 또는 고정 **IP** 주소 구성

IP 주소를 구성하려면 **DHCP(ip=dhcp)**를 사용하거나 개별 고정 **IP** 주소(**ip=<host_ip>**)를 설정합니다. 정적 **IP**를 설정하는 경우 각 노드에서 **DNS** 서버 **IP** 주소 (**nameserver=<dns_ip>**)를 확인합니다. 다음 예제에서는 다음을 설정합니다.

- 노드의 **IP** 주소는 **10.10.10.2**로 설정됩니다.

- 게이트웨이 주소는 **10.10.10.254**로 설정됩니다.
- 넷마스크는 **255.255.255.0**으로
- **core0.example.com**에 대한 호스트 이름
- **4.4.4.41**의 **DNS** 서버 주소
- **auto-configuration** 값을 **none**으로 설정합니다. **IP** 네트워킹이 정적으로 구성되면 자동 구성이 필요하지 않습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
nameserver=4.4.4.41
```



참고

DHCP를 사용하여 **RHCOS** 시스템의 **IP** 주소 지정을 구성하는 경우 시스템은 **DHCP**를 통해 **DNS** 서버 정보도 가져옵니다. **DHCP** 기반 배포의 경우 **DHCP** 서버 구성을 통해 **RHCOS** 노드에서 사용할 **DNS** 서버 주소를 정의할 수 있습니다.

고정 호스트 이름 없이 **IP** 주소 구성

정적 호스트 이름을 할당하지 않고 **IP** 주소를 구성할 수 있습니다. 사용자가 정적 호스트 이름을 설정하지 않으면 역방향 **DNS** 조회에 의해 자동으로 선택됩니다. 정적 호스트 이름이 없는 **IP** 주소를 구성하려면 다음 예제를 참조합니다.

- 노드의 **IP** 주소는 **10.10.10.2**로 설정됩니다.
- 게이트웨이 주소는 **10.10.10.254**로 설정됩니다.
- 넷마스크는 **255.255.255.0**으로
- **4.4.4.41**의 **DNS** 서버 주소

- auto-configuration 값을 none으로 설정합니다. IP 네트워킹이 정적으로 구성되면 자동 구성이 필요하지 않습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0::enp1s0:none
nameserver=4.4.4.41
```

여러 네트워크 인터페이스 지정

여러 ip= 항목을 설정하여 여러 네트워크 인터페이스를 지정할 수 있습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

기본 게이트웨이 및 경로 구성

선택 사항: rd.route= 값을 설정하여 추가 네트워크에 대한 경로를 구성할 수 있습니다.

참고

하나 이상의 네트워크를 구성할 때 하나의 기본 게이트웨이가 필요합니다. 추가 네트워크 게이트웨이가 기본 네트워크 게이트웨이와 다른 경우 기본 게이트웨이가 기본 네트워크 게이트웨이어야 합니다.

- 다음 명령을 실행하여 기본 게이트웨이를 구성합니다.

```
ip=::10.10.10.254::::
```

- 다음 명령을 입력하여 추가 네트워크의 경로를 구성합니다.

```
rd.route=20.20.20.0/24:20.20.20.254:enp2s0
```

단일 인터페이스에서 DHCP 비활성화

두 개 이상의 네트워크 인터페이스가 있고 하나의 인터페이스만 사용되는 경우와 같이 단일 인터페이스에서 DHCP를 비활성화할 수 있습니다. 이 예에서 enp1s0 인터페이스에는 정적 네트워킹 구성이 있으며 enp2s0 용으로 DHCP가 사용되지 않습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=::::core0.example.com:enp2s0:none
```

DHCP 및 고정 IP 구성 결합

다음과 같이 시스템의 **DHCP** 및 고정 **IP** 구성을 여러 네트워크 인터페이스와 결합할 수 있습니다.

```
ip=enp1s0:dhcp
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

개별 인터페이스에서 VLAN 구성

선택 사항: **vlan=** 매개변수를 사용하여 개별 인터페이스에서 **VLAN**을 구성할 수 있습니다.

- 네트워크 인터페이스에서 **VLAN**을 구성하고 고정 **IP** 주소를 사용하려면 다음 명령을 실행합니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0.100:none
vlan=enp2s0.100:enp2s0
```

- 네트워크 인터페이스에서 **VLAN**을 구성하고 **DHCP**를 사용하려면 다음 명령을 실행합니다.

```
ip=enp2s0.100:dhcp
vlan=enp2s0.100:enp2s0
```

여러 DNS 서버 제공

각 서버에 **nameserver=** 항목을 추가하여 여러 **DNS** 서버를 제공할 수 있습니다. 예를 들면 다음과 같습니다.

```
nameserver=1.1.1.1
nameserver=8.8.8.8
```

단일 인터페이스에 여러 네트워크 인터페이스 본딩

선택 사항: **bond=** 옵션을 사용하여 여러 네트워크 인터페이스를 단일 인터페이스에 결합할 수 있습니다. 다음 예제를 참조하십시오.

- 결합된 인터페이스를 구성하는 구문: **bond = name [: network_interfaces] [: options]**

  name은 결합하는 기기 이름(**bond0**)이고 ***network_interfaces***는 쉼표로 구분되는 물리적(이더넷) 인터페이스 목록(**em1, em2**)이며, ***options***은 쉼표로 구분되는 결합 옵션 목록입니다. 사용 가능한 옵션을 보려면 **modinfo bonding**을 입력하십시오.

- **bond=**를 사용하여 결합된 인터페이스를 생성할 때 **IP** 주소가 할당되는 방법과 결합된 인터페이스에 대한 기타 정보를 지정해야 합니다.

- DHCP를 사용하도록 결합된 인터페이스를 구성하려면 bond의 IP 주소를 dhcp로 설정하십시오. 예를 들면 다음과 같습니다.

```
bond=bond0:em1,em2:mode=active-backup
ip=bond0:dhcp
```

- 고정 IP 주소를 사용하도록 결합된 인터페이스를 구성하려면 원하는 특정 IP 주소 및 관련 정보를 입력합니다. 예를 들면 다음과 같습니다.

```
bond=bond0:em1,em2:mode=active-backup
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
```

단일 인터페이스에 여러 네트워크 인터페이스 본딩

선택 사항: vlan= 매개변수를 사용하고 DHCP를 사용하도록 결합된 인터페이스에서 VLAN을 구성할 수 있습니다. 예를 들면 다음과 같습니다.

```
ip=bond0.100:dhcp
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

다음 예제를 사용하여 VLAN을 사용하여 결합된 인터페이스를 구성하고 고정 IP 주소를 사용합니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0.100:none
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

네트워크 팀 구성 사용

선택 사항: team= 매개변수를 사용하여 네트워크 팀을 결합의 대안으로 사용할 수 있습니다.

- 팀 인터페이스를 구성하는 구문은 team=name[:network_interfaces]입니다.

name 은 팀 장치 이름(team0)이고 *network_interfaces* 는 쉼표로 구분된 실제(이더넷) 인터페이스 목록(em1, em2)을 나타냅니다.

RHCOS가 향후 RHEL 버전으로 전환되면 티밍이 더 이상 사용되지 않을 예정입니다. 자세한 내용은 이 Red Hat 지식베이스 문서를 참조하십시오.

다음 예제를 사용하여 네트워크 팀을 구성합니다.

```
team=team0:em1,em2
ip=team0:dhcp
```

8.3.14.3.4.2. ISO 설치를 위한 coreos-installer 옵션

ISO 이미지에서 RHCOS 라이브 환경으로 부팅한 후 명령 프롬프트에서 **coreos-installer install <options> <device>**를 실행하여 RHCOS를 설치할 수 있습니다.

다음 표는 coreos-installer 명령으로 전달할 수 있는 하위 명령, 옵션 및 인수를 보여줍니다.

표 8.31. coreos-installer 하위 명령, 명령줄 옵션 및 인수

coreos-installer 설치 하위 명령	
하위 명령	*설명*
$ coreos-installer install <options> <device>	ISO 이미지에 Ignition 구성를 삽입합니다.
coreos-installer 설치 하위 명령 옵션	
옵션	*설명*
-u, **--image-url <url>**	이미지 URL을 수동으로 지정합니다.
-f, **--image-file <path>**	로컬 이미지 파일을 수동으로 지정합니다. 디버깅에 사용됩니다.
-i, --ignition-file <path>	파일의 Ignition 구성을 삽입합니다.
-I, **--ignition-url <URL>**	URL의 Ignition 구성을 삽입합니다.
--ignition-hash <digest>	Ignition 구성의 **type-value**를 요약합니다.
-p, **--platform <name>**	설치된 시스템의 Ignition 플랫폼 ID를 재정의합니다.
--append-karg <arg>…	설치된 시스템에 기본 커널 인수를 추가합니다.
--delete-karg <arg>…	설치된 시스템에서 기본 커널 인수를 삭제합니다.

-n, **--copy-network**	설치 환경의 네트워크 구성을 복사합니다.  **중요** **copy-network** 옵션은 **/etc/NetworkManager/system-connections**에 있는 네트워킹 구성만 복사합니다. 특히 시스템 호스트 이름을 복사하지 않습니다.
--network-dir <path>	**-n**과 함께 사용됩니다. 기본값은 **/etc/NetworkManager/system-connections/**입니다.
--save-partlabel <lx>..	이 레이블 glob로 파티션을 저장합니다.
--save-partindex <id>…	이 번호 또는 범위로 파티션을 저장합니다.
--insecure	서명 확인을 건너뜁니다.
--insecure-ignition	HTTPS 또는 해시 없는 Ignition URL을 허용합니다.
--architecture <name>	대상 CPU 아키텍처입니다. 기본값은 **x86_64**입니다.
--preserve-on-error	오류 발생한 파티션 테이블을 지우지 않습니다.
-h, **--help**	도움말 정보를 출력합니다.
coreos-install 설치 하위 명령 인수	
인수	*설명*
<device>	대상 장치입니다.
coreos-installer ISO Ignition 하위 명령	
하위 명령	*설명*
$ coreos-installer iso ignition embed <options> --ignition-file <file_path> <ISO_image>	ISO 이미지에 Ignition 구성를 삽입합니다.
coreos-installer iso ignition show <options> <ISO_image>	ISO 이미지에 삽입된 Ignition 구성를 표시합니다.

coreos-installer iso ignition remove <options> <ISO_image>	ISO 이미지에서 삽입된 Ignition 구성를 제거합니다.
coreos-installer ISO Ignition 하위 명령 옵션	
옵션	*설명*
-f, --force	기존 Ignition 구성를 덮어씁니다.
-i, --ignition-file <path>	사용할 Ignition 구성입니다. 기본값은 **stdin**입니다.
-o, --output <path>	새 출력 파일에 ISO를 씁니다.
-h, --help	도움말 정보를 출력합니다.
coreos-installer PXE Ignition 하위 명령	
하위 명령	*설명*
이러한 모든 옵션이 모든 하위 명령에서 허용되지는 않습니다.	
coreos-installer pxe ignition wrap <options>	Ignition 구성을 이미지로 래핑합니다.
coreos-installer pxe ignition unwrap <options> <image_name>	이미지에 래핑된 Ignition 구성를 표시합니다.
coreos-installer PXE Ignition 하위 명령 옵션	
옵션	*설명*
이러한 모든 옵션이 모든 하위 명령에서 허용되지는 않습니다.	
-i, --ignition-file <path>	사용할 Ignition 구성입니다. 기본값은 **stdin**입니다.
-o, --output <path>	새 출력 파일에 ISO를 씁니다.
-h, --help	도움말 정보를 출력합니다.

8.3.14.3.4.3. ISO 또는 PXE 설치를 위한 coreos.inst 부팅 옵션

coreos.inst 부팅 인수를 RHCOS 라이브 설치 프로그램에 전달하여 부팅 시 coreos-installer 옵션을 자동으로 호출할 수 있습니다. 이러한 매개 변수는 표준 부팅 인수 외에 제공됩니다.

- **ISO 설치의 경우 부트 로더 메뉴에서 자동 부팅을 중단하여 coreos.inst 옵션을 추가할 수 있습니다. RHEL CoreOS (Live) 메뉴 옵션이 강조 표시된 상태에서 TAB을 눌러 자동 부팅을 중**

단할 수 있습니다.

- PXE 또는 iPXE 설치의 경우 RHCOS 라이브 설치 프로그램을 부팅하기 전에 coreos.inst 옵션을 APPEND 줄에 추가해야 합니다.

다음 표는 ISO 및 PXE 설치를 위한 RHCOS 라이브 설치 관리자 coreos.inst 부팅 옵션을 보여줍니다.

표 8.32. coreos.inst 부팅 옵션

인수	설명
coreos.inst.install_dev	필수 항목입니다. 설치할 시스템의 블록 장치입니다. **sda**가 허용되더라도 전체 경로 (예:**/dev/sda**)를 사용하는 것이 좋습니다.
coreos.inst.ignition_url	선택사항: 설치된 시스템에 삽입할 Ignition 구성의 URL입니다. URL을 지정하지 않으면 Ignition 구성이 포함되지 않습니다. HTTP 및 HTTPS 프로토콜만 지원됩니다.
coreos.inst.save_partlabel	선택사항: 설치 중에 보존 할 파티션의 쉼표로 구분된 레이블입니다. Glob 스타일 와일드카드가 허용됩니다. 지정된 파티션이 존재할 필요는 없습니다.
coreos.inst.save_partindex	선택사항: 설치 도중 보존할 파티션 인덱스들입니다(쉼표로 구분됨). **m-n** 범위가 허용되며 **m** 또는 **n**은 생략할 수 있습니다. 지정된 파티션이 존재할 필요는 없습니다.
coreos.inst.insecure	선택사항: **coreos.inst.image_url**로 지정된 OS 이미지의 서명되지 않은 상태를 허용합니다.

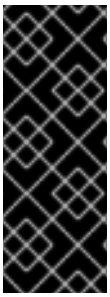
인수	설명
coreos.inst.image_url	선택사항: 지정된 RHCOS 이미지를 다운로드하여 설치합니다. ● 이 인수는 프로덕션 환경에서 사용할 수 없으며, 디버깅 용도로만 사용됩니다. ● 이 인수를 사용하면 라이브 미디어와 일치하지 않는 RHCOS 버전을 설치할 수 있지만, 설치하려는 버전과 일치하는 미디어를 사용하는 것이 좋습니다. ● **coreos.inst.image_url**을 사용하는 경우 **coreos.inst.insecure**도 사용해야 합니다. 베어메탈 미디어가 OpenShift Container Platform용으로 GPG 서명되지 않았기 때문입니다. ● HTTP 및 HTTPS 프로토콜만 지원됩니다.
coreos.inst.skip_reboot	선택사항: 설치 후 시스템을 재부팅하지 않습니다. 설치가 완료되면 설치 과정에서 발생되는 상황을 검사할 수 있는 프롬프트가 표시됩니다. 이 인수는 프로덕션 환경에서 사용할 수 없으며, 디버깅 용도로만 사용됩니다.
coreos.inst.platform_id	선택사항: RHCOS 이미지가 설치되고 있는 플랫폼의 Ignition 플랫폼 ID입니다. 기본값은 **metal**입니다. 이 옵션에 따라 VMware와 같은 클라우드 공급자의 Ignition 구성를 요청할지 여부가 결정됩니다. 예: **coreos.inst.platform_id=vmware**.
ignition.config.url	선택사항: 라이브 부팅을 위한 Ignition 구성의 URL입니다. 예를 들어 **coreos-installer**가 호출되는 방식을 사용자 지정하거나 설치 전과 후에 코드를 실행하는 데 사용할 수 있습니다. 이 URL은 설치된 시스템의 Ignition 구성인 **coreos.inst.ignition_url**과 다릅니다.

8.3.14.4. RHCOS에서 커널 인수로 다중 경로 활성화

RHCOS는 이제 기본 디스크에서 멀티패스를 지원하므로 하드웨어 장애에 대한 탄력성이 강화된 호스트 가용성을 높일 수 있습니다.

OpenShift Container Platform 4.8 이상에서 프로비저닝된 노드의 설치 시 멀티패스를 활성화할 수 있습니다. 시스템 구성을 통해 멀티패스를 활성화하면 설치 후 지원을 사용할 수 있지만 설치 중에 멀피태스를 활성화하는 것이 좋습니다.

I/O에서 최적화된 경로로 인해 I/O 시스템 오류가 발생하는 설정에서 설치 시 멀티패스를 활성화해야 합니다.



중요

IBM Z 및 **LinuxONE**에서는 설치 중에 클러스터를 구성하는 경우에만 다중 경로를 활성화할 수 있습니다. 자세한 내용은 ***IBM Z* 및 *LinuxONE*에 *z/VM*으로 *클러스터 설치*의 "RHCOS 설치 및 OpenShift Container Platform 부트스트랩 프로세스 시작"을 참조하십시오.

다음 절차에서는 설치 시 멀티패스를 활성화하고 커널 인수를 **coreos-installer install** 명령에 추가하여 설치된 시스템 자체에서 첫 번째 부팅부터 시작된 멀티패스를 사용하도록 합니다.

사전 요구 사항

- 버전 **4.8** 이상을 사용하는 실행 중인 **OpenShift Container Platform** 클러스터가 있어야 합니다.

참고

OpenShift Container Platform은 **4.6** 또는 이전 버전에서 업그레이드된 노드에서 **Day-2** 활동으로 다중 경로를 활성화할 수 없습니다.

- 관리 권한이 있는 사용자로 클러스터에 로그인했습니다.

프로세스

1. 멀티패스를 활성화하고 **multipathd** 데몬을 시작하려면 다음 명령을 실행합니다.

```
$ mpathconf --enable && systemctl start multipathd.service
```

- 선택 사항: **PXE** 또는 **ISO**를 부팅하는 경우 커널 명령줄에서 **rd.multipath=default**를 추가하여 멀티패스를 활성화할 수 있습니다.

2. **coreos-installer** 프로그램을 호출하여 커널 인수를 추가합니다.

- 시스템에 연결된 멀티패스 장치가 하나뿐인 경우 경로 **/dev/mapper/mpatha**에서 사용할 수 있어야 합니다. 예를 들면 다음과 같습니다.

```
$ coreos-installer install /dev/mapper/mpatha \ 1
--append-karg rd.multipath=default \
--append-karg root=/dev/disk/by-label/dm-mpath-root \
--append-karg rw
```

1

단일 멀티패스 장치의 경로를 나타냅니다.

- 시스템에 연결된 멀티패스 장치가 여러 개 있는 경우 보다 명확하게 하려면 /dev/mapper/mpatha를 사용하는 대신 /dev/disk/by-id에서 사용할 수 있는 WWN(World Wide Name) 심볼릭 링크를 사용하는 것이 좋습니다. 예를 들면 다음과 같습니다.

```
$ coreos-installer install /dev/disk/by-id/wwn-<wwn_ID> \ 1
--append-karg rd.multipath=default \
--append-karg root=/dev/disk/by-label/dm-mpath-root \
--append-karg rw
```

1

멀티패스 대상 장치의 WWN ID를 나타냅니다. 예를 들면 0xx194e957fcedb4841입니다.

이 심볼릭 링크는 라이브 설치 프로그램을 지시하기 위해 특수 coreos.inst.** 인수를 사용할 때coreos.inst.install_dev 커널 인수로 사용될 수도 있습니다. 자세한 내용은 "RHCOS 설치 및 OpenShift Container Platform 부트스트랩 프로세스 시작"을 참조하십시오.

3. 작업자 노드 중 하나로 이동하고 커널 명령줄 인수 (호스트의 /proc/cmdline)를 나열하여 커널 인수가 작동하는지 확인합니다.

```
$ oc debug node/ip-10-0-141-105.ec2.internal
```

출력 예

```
Starting pod/ip-10-0-141-105ec2internal-debug ...
To use host binaries, run `chroot /host`

sh-4.2# cat /host/proc/cmdline
...
```

```
rd.multipath=default root=/dev/disk/by-label/dm-mpath-root
...

sh-4.2# exit
```

추가된 커널 인수가 표시되어야 합니다.

8.3.14.5. bootupd를 사용하여 부트로더 업데이트

bootupd를 사용하여 부트로더를 업데이트하려면 **RHCOS** 머신에 수동으로 **bootupd**를 설치하거나 **systemd** 유닛이 활성화된 머신 구성을 제공해야 합니다. **grubby** 또는 기타 부트로더 툴과는 달리 **bootupd**는 커널 인수 전달과 같은 커널 공간 구성을 관리하지 않습니다.

bootupd를 설치한 후 **OpenShift Container Platform** 클러스터에서 원격으로 관리할 수 있습니다.

참고

bootupd는 **BootHole** 취약점에 대한 보호와 같이 베어 메탈 또는 가상화 하이퍼바이저 설치에서만 사용하는 것이 좋습니다.

수동 설치 방법

bootctl 명령줄 툴을 사용하여 **bootupd**를 수동으로 설치할 수 있습니다.

1. 시스템 상태를 검사합니다.

```
# bootupctl status
```

출력 예

```
Component EFI
  Installed: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
  Update: At latest version
```

2. **bootupd**를 설치하지 않고 생성된 **RHCOS** 이미지에는 명시적 채택 단계가 필요합니다.

   시스템 상태가 **Adoptable**인 경우 채택을 수행합니다.

```
# bootupctl adopt-and-update
```

   출력 예

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

3. 업데이트를 사용할 수 있는 경우 다음 재부팅에 변경 사항이 적용되도록 업데이트를 적용합니다.

```
# bootupctl update
```

   출력 예

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

시스템 구성 방법

bootupd를 활성화하는 또 다른 방법은 머신 구성을 제공하는 것입니다.

- 다음 예와 같이 **systemd** 단위가 활성화된 머신 구성 파일을 제공합니다.

  출력 예

```
variant: rhcos
version: 1.1.0
systemd:
  units:
    - name: custom-bootupd-auto.service
      enabled: true
      contents: |
        [Unit]
        Description=Bootupd automatic update

        [Service]
        ExecStart=/usr/bin/bootupctl update
        RemainAfterExit=yes

        [Install]
        WantedBy=multi-user.target
```

8.3.15. 부트스트랩 프로세스가 완료될 때까지 대기 중

OpenShift Container Platform 부트스트랩 프로세스는 클러스터 노드가 먼저 디스크에 설치된 영구 RHCOS 환경으로 부팅된 후에 시작됩니다. Ignition 구성 파일을 통해 제공되는 구성 정보는 부트스트랩 프로세스를 초기화하고 머신에 OpenShift Container Platform을 설치하는 데 사용됩니다. 부트스트랩 프로세스가 완료될 때까지 기다려야 합니다.

사전 요구 사항

- 클러스터에 대한 Ignition 구성 파일이 생성되어 있습니다.
- 적합한 네트워크, DNS 및 로드 밸런싱 인프라가 구성되어 있습니다.
- 설치 프로그램을 받아서 클러스터의 Ignition 구성 파일을 생성했습니다.
- 클러스터 머신에 RHCOS를 설치하고 OpenShift Container Platform 설치 프로그램에서 생성된 Ignition 구성 파일을 제공했습니다.
- 사용자 시스템에서 직접 인터넷에 액세스하거나 HTTP 또는 HTTPS 프록시를 사용할 수 있습니다.

절차

1. 부트스트랩 프로세스를 모니터링합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

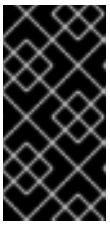
2 다른 설치 세부 사항을 보려면 info 대신 warn, debug 또는 error를 지정합니다.

출력 예

```
INFO Waiting up to 30m0s for the Kubernetes API at
https://api.test.example.com:6443...
INFO API v1.22.1 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

이 명령은 Kubernetes API 서버가 컨트롤 플레인 시스템에서 부트스트랩되었다는 신호를 보낼 때 성공합니다.

2. 부트스트랩 프로세스가 완료된 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다.



중요

이 시점에 로드 밸런서에서 부트스트랩 시스템을 제거해야 합니다. 부트스트랩 머신 자체를 제거하거나 다시 포맷할 수도 있습니다.

추가 리소스

- 설치 문제가 발생하는 경우 설치 로그 모니터링 및 진단 데이터 검색에 대한 자세한 내용은 설치 진행 상황 모니터링을 참조하십시오.

8.3.16. CLI를 사용하여 클러스터에 로그인

클러스터 kubeconfig 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. kubeconfig 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- oc CLI를 설치했습니다.

절차

1. kubeadmin 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 oc 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

8.3.17. 머신의 인증서 서명 요청 승인

클러스터에 시스템을 추가하면 추가한 시스템별로 보류 중인 인증서 서명 요청(CSR)이 두 개씩 생성

됩니다. 이러한 CSR이 승인되었는지 확인해야 하며, 필요한 경우 이를 직접 승인해야 합니다. 클라이언트 요청을 먼저 승인한 다음 서버 요청을 승인해야 합니다.

사전 요구 사항

- 클러스터에 시스템을 추가했습니다.

절차

1. 클러스터가 시스템을 인식하는지 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0  Ready    master  63m  v1.22.1
master-1  Ready    master  63m  v1.22.1
master-2  Ready    master  64m  v1.22.1
```

출력에 생성된 모든 시스템이 나열됩니다.

참고

이전 출력에는 일부 CSR이 승인될 때까지 컴퓨팅 노드(작업자 노드라고도 함)가 포함되지 않을 수 있습니다.

2. 보류 중인 CSR을 검토하고 클러스터에 추가한 각 시스템에 대해 Pending 또는 Approved 상태의 클라이언트 및 서버 요청이 표시되는지 확인합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE    REQUESTOR                                                    CONDITION
csr-8b2br   15m    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

예에서는 두 시스템이 클러스터에 참여하고 있습니다. 목록에는 승인된 **CSR**이 더 많이 나타날 수도 있습니다.

3.

CSR이 승인되지 않은 경우, 추가된 시스템에 대한 모든 보류 중인 **CSR**이 **Pending** 상태로 전환된 후 클러스터 시스템의 **CSR**을 승인합니다.

참고

CSR은 교체 주기가 자동으로 만료되므로 클러스터에 시스템을 추가한 후 **1** 시간 이내에 **CSR**을 승인하십시오. 한 시간 내에 승인하지 않으면 인증서가 교체되고 각 노드에 대해 두 개 이상의 인증서가 표시됩니다. 이러한 인증서를 모두 승인해야 합니다. 클라이언트 **CSR**이 승인되면 **Kubelet**은 인증서에 대한 보조 **CSR**을 생성하므로 수동 승인이 필요합니다. 그러면 **Kubelet**에서 동일한 매개변수를 사용하여 새 인증서를 요청하는 경우 인증서 갱신 요청은 **machine-approver**에 의해 자동으로 승인됩니다.

참고

베어 메탈 및 기타 사용자 프로비저닝 인프라와 같이 머신 **API**를 사용하도록 활성화되지 않는 플랫폼에서 실행되는 클러스터의 경우 **CSR(Kubelet service Certificate Request)**을 자동으로 승인하는 방법을 구현해야 합니다. 요청이 승인되지 않으면 **API** 서버가 **kubelet**에 연결될 때 서비스 인증서가 필요하므로 **oc exec, oc rsh, oc logs** 명령을 성공적으로 수행할 수 없습니다. **Kubelet** 엔드 포인트에 연결하는 모든 작업을 수행하려면 이 인증서 승인이 필요합니다. 이 방법은 새 **CSR**을 감시하고 **CSR**이 **system:node** 또는 **system:admin** 그룹의 **node-bootstrapper** 서비스 계정에 의해 제출되었는지 확인하고 노드의 **ID**를 확인합니다.

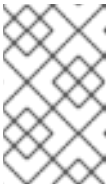
- 개별적으로 승인하려면 유효한 **CSR** 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```



참고

일부 Operator는 일부 CSR이 승인될 때까지 사용할 수 없습니다.

4. 이제 클라이언트 요청이 승인되었으므로 클러스터에 추가한 각 머신의 서버 요청을 검토해야 합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 나머지 CSR이 승인되지 않고 Pending 상태인 경우 클러스터 머신의 CSR을 승인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 <csr_name>은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs oc adm certificate approve
```

6. 모든 클라이언트 및 서버 CSR이 승인된 후 머신은 Ready 상태가 됩니다. 다음 명령을 실행하여 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0 Ready    master  73m  v1.22.1
master-1 Ready    master  73m  v1.22.1
master-2 Ready    master  74m  v1.22.1
worker-0 Ready    worker  11m  v1.22.1
worker-1 Ready    worker  11m  v1.22.1
```

참고

머신이 Ready 상태로 전환하는 데 서버 CSR의 승인 후 몇 분이 걸릴 수 있습니다.

추가 정보

- CSR에 대한 자세한 내용은 인증서 서명 요청을 참조하십시오.

8.3.18. Operator의 초기 설정

컨트롤 플레인이 초기화된 후 일부 Operator를 즉시 구성하여 모두 사용 가능하도록 해야 합니다.

사전 요구 사항

- 컨트롤 플레인이 초기화되어 있습니다.

절차

1. 클러스터 구성 요소가 온라인 상태인지 확인합니다.

```
$ watch -n5 oc get clusteroperators
```

출력 예

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.9.0     True        False         False      19m
baremetal                                  4.9.0     True        False         False      37m
cloud-credential                           4.9.0     True        False         False      40m
cluster-autoscaler                         4.9.0     True        False         False      37m
config-operator                            4.9.0     True        False         False      38m
console                                    4.9.0     True        False         False      26m
csi-snapshot-controller                    4.9.0     True        False         False      37m
dns                                        4.9.0     True        False         False      37m
etcd                                       4.9.0     True        False         False      36m
image-registry                             4.9.0     True        False         False      31m
ingress                                    4.9.0     True        False         False      30m
insights                                   4.9.0     True        False         False      31m
kube-apiserver                             4.9.0     True        False         False      26m
kube-controller-manager                    4.9.0     True        False         False      36m
kube-scheduler                             4.9.0     True        False         False      36m
kube-storage-version-migrator              4.9.0     True        False         False      37m
machine-api                                4.9.0     True        False         False      29m
machine-approver                           4.9.0     True        False         False      37m
machine-config                             4.9.0     True        False         False      36m
marketplace                                4.9.0     True        False         False      37m
monitoring                                 4.9.0     True        False         False      29m
network                                    4.9.0     True        False         False      38m
node-tuning                                4.9.0     True        False         False      37m
openshift-apiserver                        4.9.0     True        False         False      32m
openshift-controller-manager               4.9.0     True        False         False      30m
openshift-samples                          4.9.0     True        False         False      32m
operator-lifecycle-manager                 4.9.0     True        False         False      37m
operator-lifecycle-manager-catalog         4.9.0     True        False         False      37m
operator-lifecycle-manager-packageserver   4.9.0     True        False         False      32m
service-ca                                 4.9.0     True        False         False      38m
storage                                    4.9.0     True        False         False      37m
```

2. 사용할 수 없는 **Operator**를 구성합니다.

추가 리소스

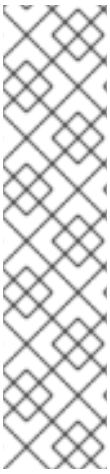
- 실패한 OpenShift Container Platform 설치 시 데이터를 수집하는 방법에 대한 자세한 내용은 설치 실패에서 로그 수집을 참조하십시오.

- 클러스터 전체에서 Operator Pod 상태를 확인하고 진단을 위해 Operator 로그를 수집하는 단계는 Operator 문제 해결을 참조하십시오.

8.3.18.1. 설치 중 제거된 이미지 레지스트리

공유 가능한 개체 스토리지를 제공하지 않는 플랫폼에서 OpenShift Image Registry Operator는 자체적으로 Removed로 부트스트랩합니다. 이를 통해 openshift-installer가 이러한 플랫폼 유형에서 설치를 완료할 수 있습니다.

설치 후 managementState를 Removed에서 Managed로 전환하도록 Image Registry Operator 구성을 편집해야 합니다.



참고

Prometheus 콘솔은 ImageRegistryRemoved 경고를 제공합니다. 다음은 예시 경고입니다.

"이미지 레지스트리가 제거되었습니다. ImageStreamTags를 참조하는 ImageStreamTags, BuildConfigs 및 DeploymentConfigs가 예상대로 작동하지 않을 수 있습니다. 스토리지를 구성하고 configs.imageregistry.operator.openshift.io를 편집하여 구성을 Managed 상태로 업데이트하십시오."

8.3.18.2. 이미지 레지스트리 스토리지 구성

기본 스토리지를 제공하지 않는 플랫폼에서는 처음에 Image Registry Operator를 사용할 수 없습니다. 설치한 후에 스토리지를 사용하도록 레지스트리를 구성하여 Registry Operator를 사용 가능하도록 만들어야 합니다.

프로덕션 클러스터에 필요한 영구 볼륨을 구성하는 과정의 지침이 표시됩니다. 해당하는 경우, 프로덕션 환경 외 클러스터에서만 사용할 수 있는 저장 위치로서 빈 디렉터리를 구성하는 과정의 지침이 표시됩니다.

업그레이드 중에 Recreate 롤아웃 전략을 사용하여 이미지 레지스트리의 블록 스토리지 유형 사용을 허용하기 위한 추가 지침이 제공됩니다.

8.3.18.3. 블록 레지스트리 스토리지 구성

클러스터 관리자로서 업그레이드 중에 이미지 레지스트리가 블록 스토리지 유형을 사용할 수 있도록 허용하기 위해 **Recreate** 롤아웃 전략을 사용할 수 있습니다.

중요

블록 스토리지 볼륨이 지원되지만 프로덕션 클러스터에서 이미지 레지스트리와 함께 사용하는 것은 권장되지 않습니다. 레지스트리가 블록 스토리지에 구성된 설치는 레지스트리가 둘 이상의 복제본을 가질 수 없기 때문에 가용성이 높지 않습니다.

프로세스

1. 이미지 레지스트리 스토리지를 블록 스토리지 유형으로 설정하려면 레지스트리가 **Recreate** 롤아웃 전략을 사용하고 하나의 **(1)** 복제본에서만 실행되도록 레지스트리를 패치합니다.

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. 블록 스토리지 장치에 **PV**를 프로비저닝하고 해당 볼륨의 **PVC**를 생성합니다. 요청된 블록 볼륨은 **RWO(ReadWriteOnce)** 액세스 모드를 사용합니다.

3. 올바른 **PVC**를 참조하도록 레지스트리 구성을 편집합니다.

8.3.19. 사용자 프로비저닝 인프라에 설치 완료

Operator 구성을 완료한 후 제공하는 인프라에 클러스터 설치를 완료할 수 있습니다.

사전 요구 사항

- 컨트롤 플레인이 초기화되어 있습니다.
- 초기 **Operator** 구성을 완료해야 합니다.

프로세스

1. 다음 명령을 사용하여 모든 클러스터 구성 요소가 온라인 상태인지 확인합니다.

```
$ watch -n5 oc get clusteroperators
```

출력 예

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.9.0     True        False         False      19m
baremetal                                  4.9.0     True        False         False      37m
cloud-credential                           4.9.0     True        False         False      40m
cluster-autoscaler                         4.9.0     True        False         False      37m
config-operator                            4.9.0     True        False         False      38m
console                                    4.9.0     True        False         False      26m
csi-snapshot-controller                    4.9.0     True        False         False      37m
dns                                        4.9.0     True        False         False      37m
etcd                                       4.9.0     True        False         False      36m
image-registry                             4.9.0     True        False         False      31m
ingress                                    4.9.0     True        False         False      30m
insights                                   4.9.0     True        False         False      31m
kube-apiserver                             4.9.0     True        False         False      26m
kube-controller-manager                    4.9.0     True        False         False      36m
kube-scheduler                             4.9.0     True        False         False      36m
kube-storage-version-migrator              4.9.0     True        False         False      37m
machine-api                                4.9.0     True        False         False      29m
machine-approver                           4.9.0     True        False         False      37m
machine-config                             4.9.0     True        False         False      36m
marketplace                                4.9.0     True        False         False      37m
monitoring                                 4.9.0     True        False         False      29m
network                                    4.9.0     True        False         False      38m
node-tuning                                4.9.0     True        False         False      37m
openshift-apiserver                        4.9.0     True        False         False      32m
openshift-controller-manager               4.9.0     True        False         False      30m
openshift-samples                          4.9.0     True        False         False      32m
operator-lifecycle-manager                 4.9.0     True        False         False      37m
operator-lifecycle-manager-catalog         4.9.0     True        False         False      37m
operator-lifecycle-manager-packageserver   4.9.0     True        False         False      32m
service-ca                                 4.9.0     True        False         False      38m
storage                                    4.9.0     True        False         False      37m
```

또는 다음 명령은 모든 클러스터를 사용할 수 있을 때 알립니다. 또한 인증 정보를 검색하고 표시합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1

<installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

출력 예

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator가 Kubernetes API 서버에서 OpenShift Container Platform 클러스터 배포를 완료하면 명령이 성공합니다.

중요

- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 node-bootstrapper 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

2. Kubernetes API 서버가 Pod와 통신하고 있는지 확인합니다.

   a. 모든 Pod 목록을 보려면 다음 명령을 사용하십시오.

   ```
   $ oc get pods --all-namespaces
   ```

   출력 예

   -

```
NAMESPACE                  NAME                                        READY   STATUS
RESTARTS   AGE
openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8
1/1    Running    1        9m
openshift-apiserver        apiserver-67b9g                             1/1     Running    0
3m
openshift-apiserver        apiserver-ljcmx                             1/1     Running    0
1m
openshift-apiserver        apiserver-z25h4                             1/1     Running    0
2m
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8
1/1    Running    0        5m
...
```

b. 다음 명령을 사용하여 이전 명령의 출력에 나열된 Pod의 로그를 표시합니다.

```
$ oc logs <pod_name> -n <namespace> 1
```

1 이전 명령의 출력에 표시된 대로 Pod 이름과 네임스페이스를 지정합니다.

Pod 로그가 표시되면 Kubernetes API 서버는 클러스터 시스템과 통신할 수 있습니다.

3. FCP(Fibre Channel Protocol)를 사용하는 설치에는 다중 경로를 활성화하기 위해 추가 단계가 필요합니다. 설치 중에 멀티패스를 활성화하지 마십시오.

자세한 내용은 설치 후 머신 구성 작업 설명서에서 "RHCOS에서 커널 인수를 사용하여 멀티패스 활성화"를 참조하십시오.

8.3.20. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription

watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

8.3.21. 다음 단계

- 설치를 확인합니다.
- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.
- 레지스트리를 설정하고 레지스트리 스토리지를 구성합니다.

8.4. 제한된 네트워크에서 사용자가 프로비저닝한 베어 메탈 클러스터를 설치

OpenShift Container Platform 4.9에서는 제한된 네트워크에서 프로비저닝하는 베어메탈 인프라에 클러스터를 설치할 수 있습니다.

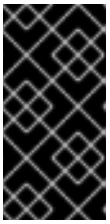
중요

이 프로세스에 따라 가상화 또는 클라우드 환경에 클러스터를 배포할 수는 있지만 베어 메탈 이외 플랫폼에 대한 추가적인 고려사항도 알고 있어야 합니다. 가상화 또는 클라우드 환경에서 OpenShift Container Platform 클러스터를 설치하기 전에 테스트되지 않은 플랫폼에 OpenShift Container Platform 배포 지침의 정보를 검토하십시오.

8.4.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.

- 미러 호스트에 레지스트리를 생성하고 사용 중인 **OpenShift Container Platform** 버전의 **imageContentSources** 데이터를 가져옵니다.

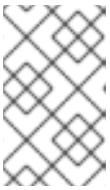


중요

미러 호스트에 설치 미디어가 있으므로 해당 컴퓨터를 사용하여 모든 설치 단계를 완료하십시오.

- 클러스터용 영구 스토리지를 프로비저닝합니다. 프라이빗 이미지 레지스트리를 배포하려면 스토리지에서 **ReadWriteMany** 액세스 모드를 제공해야 합니다.

- 방화벽을 사용하며 **Telemetry** 서비스를 사용할 예정인 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성해야 합니다.



참고

프록시를 구성하는 경우에도 해당 사이트 목록을 검토하십시오.

8.4.2. 네트워크가 제한된 환경에서의 설치 정보

OpenShift Container Platform 4.9에서는 소프트웨어 구성 요소를 받기 위한 인터넷 접속이 필요하지 않은 설치를 수행할 수 있습니다. 제한된 네트워크 설치는 클러스터를 설치하는 클라우드 플랫폼에 따라 설치 관리자 프로비저닝 인프라 또는 사용자 프로비저닝 인프라를 사용하여 완료할 수 있습니다.

클라우드 플랫폼에 제한된 네트워크 설치를 수행하는 방법을 선택해도 클라우드 **API**에 액세스는 가능해야 합니다. **Amazon Web Service**의 **Route 53 DNS** 및 **IAM** 서비스와 같은 일부 클라우드 기능에는 인터넷 액세스가 필요합니다. 사용 중인 네트워크에 따라 베어메탈 하드웨어 또는 **VMware vSphere**에 설치하기 위해 필요한 인터넷 액세스가 줄어들 수 있습니다.

제한된 네트워크 설치를 완료하려면 **OpenShift Container Platform** 레지스트리의 내용을 미러링하고 설치 미디어를 포함할 레지스트리를 생성해야 합니다. 인터넷과 폐쇄 네트워크에 모두 액세스하거나 제한 사항을 따르는 다른 방법을 통해 미러 호스트에 레지스트리를 생성할 수 있습니다.

중요

사용자 프로비저닝 설치의 구성이 복잡하므로 사용자 프로비저닝 인프라를 사용하여 제한된 네트워크 설치를 시도하기 전에 표준 사용자 프로비저닝 인프라 설치를 완료하는 것이 좋습니다. 이 테스트 설치를 완료하면 제한된 네트워크에 설치하는 동안 발생할 수 있는 문제를 보다 쉽게 파악 및 해결할 수 있습니다.

8.4.2.1. 추가 제한

제한된 네트워크의 클러스터에는 다음과 같은 추가 제한이 있습니다.

- **ClusterVersion** 상태에 사용 가능한 업데이트를 검색할 수 없음 오류가 포함되어 있습니다.
- 기본적으로 필요한 이미지 스트림 태그에 액세스할 수 없기 때문에 개발자 카탈로그의 내용을 사용할 수 없습니다.

8.4.3. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하기 위해 필요한 이미지를 받으려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 **Telemetry** 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.

중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

8.4.4. 사용자 프로비저닝 인프라를 포함한 클러스터의 시스템 요구사항

사용자 프로비저닝 인프라가 포함된 클러스터의 경우, 필요한 모든 시스템을 배포해야 합니다.

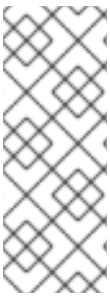
이 섹션에서는 사용자 프로비저닝 인프라에 **OpenShift Container Platform**을 배포해야 하는 요구 사항에 대해 설명합니다.

8.4.4.1. 클러스터 설치에 필요한 시스템

최소 **OpenShift Container Platform** 클러스터에 다음과 같은 호스트가 필요합니다.

표 8.33. 최소 필수 호스트

호스트	설명
임시 부트스트랩 시스템 한 개	컨트롤 플레인 시스템 세 개에 OpenShift Container Platform 클러스터를 배포하기 위한 부트스트랩 시스템이 클러스터에 필요합니다. 클러스터를 설치한 후 부트스트랩 시스템을 제거할 수 있습니다.
컨트롤 플레인 시스템 세 개	컨트롤 플레인 시스템은 컨트롤 플레인을 구성하는 Kubernetes 및 OpenShift Container Platform 서비스를 실행합니다.
두 개 이상의 컴퓨팅 시스템(작업자 시스템이라고도 함).	OpenShift Container Platform 사용자가 요청한 워크로드는 컴퓨팅 머신에서 실행됩니다.



참고

예외적으로 세 개의 컨트롤 플레인 시스템으로 구성된 베어 메탈 클러스터에 제로 컴퓨팅 머신을 실행할 수 있습니다. 이를 통해 클러스터 관리자와 개발자들이 테스트, 개발, 프로덕션에 사용할 수 있는 소형화되고 리소스 효율이 높은 클러스터를 제공합니다. 컴퓨팅 머신 하나를 실행하는 것은 지원되지 않습니다.

중요

클러스터의 고가용성을 유지하려면 이러한 클러스터 시스템에 대해 별도의 물리적 호스트를 사용하십시오.

부트스트랩, 컨트롤 플레인 시스템은 운영 체제로 **RHCOS (Red Hat Enterprise Linux CoreOS)**를 사용해야 합니다. 그러나 컴퓨팅 머신은 **RHCOS(Red Hat Enterprise Linux CoreOS), RHEL (Red Hat Enterprise Linux) 7.9** 또는 **RHEL 8.4** 중에서 선택할 수 있습니다.

RHCOS는 **RHEL 8(Red Hat Enterprise Linux)**을 기반으로하며 모든 하드웨어 인증 및 요구사항을 모두 이어받습니다. **Red Hat Enterprise Linux** 기술 기능 및 제한을 참조하십시오.

8.4.4.2. 클러스터 설치를 위한 최소 리소스 요구 사항

각 클러스터 시스템이 다음과 같은 최소 요구사항을 충족해야 합니다.

표 8.34. 최소 리소스 요구사항

시스템	운영 체제	CPU [1]	RAM	스토리지	IOPS [2]
부트스트랩	RHCOS	4	16GB	100GB	300
컨트롤 플레인	RHCOS	4	16GB	100GB	300
컴퓨팅	RHCOS, RHEL 7.9 또는 RHEL 8.4 [3]	2	8GB	100GB	300

1. **SMT(**동시 멀티스레딩) 또는 하이퍼 스레딩이 활성화되지 않은 경우 하나의 **CPU**는 하나의 실제 코어에 해당합니다. 활성화하면 다음과 같은 공식을 사용하여 해당 비율을 계산합니다. **(**코어 당 스레드 수**). (**코어 당 스레드**)** 소켓**s = CPU.**

2. **OpenShift Container Platform** 및 **Kubernetes**는 디스크 성능에 민감하며 특히 **10ms p99 fsync** 기간이 필요한 컨트롤 플레인 노드의 **etcd**에 더 빠른 스토리지가 권장됩니다. 많은 클라우드 플랫폼에서 스토리지 크기와 **IOPS**를 함께 확장되므로 충분한 성능을 얻으려면 스토리지 볼륨을 과도하게 할당해야 할 수 있습니다.

3. 사용자가 프로비저닝한 모든 설치와 마찬가지로 클러스터에서 **RHEL** 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. **RHEL 7** 컴퓨팅 머신 사용

은 더 이상 사용되지 않으며 향후 OpenShift Container Platform 4 릴리스에서 제거될 예정입니다.

8.4.4.3. 인증서 서명 요청 관리

사용자가 프로비저닝하는 인프라를 사용하는 경우 자동 시스템 관리 기능으로 인해 클러스터의 액세스가 제한되므로 설치한 후 클러스터 인증서 서명 요청(CSR)을 승인하는 메커니즘을 제공해야 합니다. kube-controller-manager는 kubelet 클라이언트 CSR만 승인합니다. machine-approver는 올바른 시스템에서 발행한 요청인지 확인할 수 없기 때문에 kubelet 자격 증명을 사용하여 요청하는 서비스 인증서의 유효성을 보장할 수 없습니다. kubelet 서빙 인증서 요청의 유효성을 확인하고 요청을 승인하는 방법을 결정하여 구현해야 합니다.

추가 리소스

- 베어 메탈 환경에서 3-노드 클러스터를 배포하는 방법에 대한 자세한 내용은 3-노드 클러스터 구성을 참조하십시오.
- 설치 후 클러스터 인증서 서명 요청 승인에 대한 자세한 내용은 시스템의 인증서 서명 요청 승인을 참조하십시오.

8.4.4.4. 사용자 프로비저닝 인프라에 대한 네트워킹 요구사항

모든 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템이 부팅 중에 Ignition 구성 파일을 가져오려면 initramfs에 네트워킹을 구성해야 합니다.

초기 부팅 과정에서 시스템에 필요한 부팅 옵션을 제공하여 DHCP 서버를 통해 설정하거나 정적으로 설정하는 IP 주소 구성이 필요합니다. 네트워크 연결이 설정된 후 시스템은 HTTP 또는 HTTPS 서버에서 Ignition 구성 파일을 다운로드합니다. 그런 다음 Ignition 구성 파일을 사용하여 각 머신의 정확한 상태를 설정합니다. Machine Config Operator는 설치 후 새 인증서 또는 키 적용과 같은 머신에 대한 추가 변경을 완료합니다.

클러스터 시스템의 장기적인 관리를 위해 DHCP 서버를 사용하는 것이 좋습니다. DHCP 서버가 클러스터 시스템에 영구 IP 주소, DNS 서버 정보 및 호스트 이름을 제공하도록 구성되었는지 확인합니다.

참고

사용자 프로비저닝 인프라에 DHCP 서비스를 사용할 수 없는 경우 RHCOS 설치 시 노드에 IP 네트워킹 구성과 DNS 서버의 주소를 대신 제공할 수 있습니다. ISO 이미지에서 설치하는 경우 부팅 인수로 전달할 수 있습니다. 고정 IP 프로비저닝 *및 고급 네트워킹 옵션에 대한 자세한 내용은 **RHCOS** 설치 및 **OpenShift Container Platform** 부트스트랩 프로세스 시작* 섹션을 참조하십시오.

Kubernetes API 서버가 클러스터 시스템의 노드 이름을 확인할 수 있어야 합니다. API 서버와 작업자 노드가 서로 다른 영역에 있는 경우, API 서버가 노드 이름을 확인할 수 있도록 기본 DNS 검색 영역을 설정할 수 있습니다. 노드 개체와 모든 DNS 요청에서 항상 정규화된 도메인 이름으로 호스트를 가리키는 것도 지원되는 방법입니다

8.4.4.4.1. DHCP를 통해 클러스터 노드의 호스트 이름 설정

RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 호스트 이름은 NetworkManager를 통해 설정됩니다. 기본적으로 시스템은 DHCP를 통해 호스트 이름을 가져옵니다. DHCP에서 호스트 이름을 제공하지 않거나, 커널 인수를 통해 정적으로 설정하거나 다른 방법을 통해 설정하면 역방향 DNS 조회를 통해 얻을 수 있습니다. 역방향 DNS 조회는 노드에서 네트워크를 초기화한 후 수행되며 확인하는 데 시간이 걸릴 수 있습니다. 다른 시스템 서비스는 이 보다 먼저 시작하여 호스트 이름을 localhost 등으로 감지할 수 있습니다. DHCP를 사용하여 각 클러스터 노드의 호스트 이름을 제공하여 이 문제를 방지할 수 있습니다.

또한 DHCP를 통해 호스트 이름을 설정하면 DNS 분할 수평 구현 환경에서 수동으로 DNS 레코드 이름 구성 오류를 무시할 수 있습니다.

8.4.4.4.2. 네트워크 연결 요구사항

OpenShift Container Platform 클러스터 구성 요소가 통신할 수 있도록 시스템 간 네트워크 연결을 구성해야 합니다. 각 시스템에서 클러스터에 있는 다른 모든 시스템의 호스트 이름을 확인할 수 있어야 합니다.

이 섹션에서는 필요한 포트에 대해 자세히 설명합니다.

표 8.35. 모든 시스템 간 통신에 사용되는 포트

프로토콜	포트	설명
ICMP	해당 없음	네트워크 연결성 테스트
TCP	**1936**	메트릭

프로토콜	포트	설명
	9000-**9999**	**9100**-**9101** 포트의 노드 내보내기 및 **9099** 포트의 Cluster Version Operator를 포함한 호스트 수준 서비스.
	10250-**10259**	Kubernetes에서 예약하는 기본 포트
	10256	openshift-sdn
UDP	**4789**	VXLAN 및 Geneve
	6081	VXLAN 및 Geneve
	9000-**9999**	**9100**-**9101** 포트의 노드 내보내기를 포함한 호스트 수준 서비스.
	500	IPsec IKE 패킷
	4500	IPsec NAT-T 패킷
TCP/UDP	**30000**-**32767**	Kubernetes 노드 포트
ESP	해당 없음	IPsec Encapsulating Security Payload (ESP)

표 8.36. 모든 시스템과 컨트롤 플레인 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**6443**	Kubernetes API

표 8.37. 컨트롤 플레인 머신 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**2379**-**2380**	etcd 서버 및 피어 포트

사용자 프로비저닝 인프라에 대한 NTP 구성

OpenShift Container Platform 클러스터는 기본적으로 공용 **NTP(Network Time Protocol)** 서버를 사용하도록 구성되어 있습니다. 로컬 엔터프라이즈 **NTP** 서버를 사용하거나 클러스터가 연결이 끊긴 네트워크에 배포되는 경우 특정 시간 서버를 사용하도록 클러스터를 구성할 수 있습니다. 자세한 내용은 ***chrony** 타임 서비스 설정* 문서를 참조하십시오.

DHCP 서버가 **NTP** 서버 정보를 제공하는 경우 **RHCOS(Red Hat Enterprise Linux CoreOS)** 시스템의 **chrony** 타임 서비스에서 정보를 읽고 **NTP** 서버와 클럭을 동기화할 수 있습니다.

추가 리소스

- chrony 타임 서비스 설정

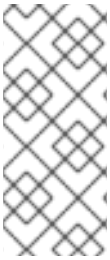
8.4.4.5. 사용자 프로비저닝 DNS 요구사항

OpenShift Container Platform 배포의 경우 다음 구성 요소에 DNS 이름을 확인해야 합니다.

- Kubernetes API
- OpenShift Container Platform 애플리케이션 와일드카드
- 부트스트랩, 컨트롤 플레인 및 컴퓨팅 시스템

Kubernetes API, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템에 대한 역방향 DNS 확인이 필요합니다.

DNS A/AAAA 또는 CNAME 레코드는 이름 확인에 사용되며 PTR 레코드는 역방향 이름 확인에 사용됩니다. RHCOS (Red Hat Enterprise Linux CoreOS)는 DHCP에서 호스트 이름을 제공하지 않는 한 모든 노드의 호스트 이름을 설정할 때 역방향 레코드를 사용하기 때문에 역방향 레코드가 중요합니다. 또한 역방향 레코드는 OpenShift Container Platform이 작동하는 데 필요한 인증서 서명 요청 (CSR)을 생성하는 데 사용됩니다.



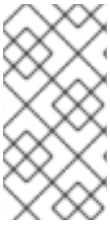
참고

DHCP 서버를 사용하여 각 클러스터 노드에 호스트 이름을 제공하는 것이 좋습니다. 자세한 내용은 *사용자 프로비저닝 인프라 섹션에 대한 DHCP 권장 사항* 섹션을 참조하십시오.

사용자가 프로비저닝한 OpenShift Container Platform 클러스터에 대해 다음 DNS 레코드가 필요하며 설치 전에 있어야 합니다. 각 레코드에서 <cluster_name>은 클러스터 이름이고 <base_domain>은 install-config.yaml 파일에서 지정하는 기반 도메인입니다. 전체 DNS 레코드는 <component>.<cluster_name>.<base_domain> 형식입니다.

표 8.38. 필수 DNS 레코드

구성 요소	레코드	설명
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API 로드 밸런서를 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.
	api-int.<cluster_name>.<base_domain>.	내부적으로 API 로드 밸런서를 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.  **중요** API 서버는 Kubernetes에 기록된 호스트 이름으로 작업자 노드를 확인할 수 있어야 합니다. API 서버가 노드 이름을 확인할 수 없는 경우 프록시된 API 호출이 실패할 수 있으며 pod에서 로그를 검색할 수 없습니다.
라우트	***.apps.<cluster_name>.<base_domain>.**	애플리케이션 인그레스 로드 밸런서를 참조하는 와일드카드 DNS A/AAA 또는 CNAME 레코드입니다. 애플리케이션 인그레스 로드 밸런서는 Ingress 컨트롤러 Pod를 실행하는 머신을 대상으로 합니다. Ingress 컨트롤러 Pod는 기본적으로 컴퓨팅 머신에서 실행됩니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다. 예를 들어 **console-openshift-console.apps.<cluster_name>.<base_domain>**은 OpenShift Container Platform 콘솔의 와일드카드 경로로 사용됩니다.
부트스트랩 시스템	**bootstrap.<cluster_name>.<base_domain>.**	부트스트랩 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.
컨트롤 플레인 머신	**<master><n>.<cluster_name>.<base_domain>.**	컨트롤 플레인 노드의 각 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.
컴퓨팅 머신	**<worker><n>.<cluster_name>.<base_domain>.**	작업자 노드의 각 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.



참고

OpenShift Container Platform 4.4 이상에서는 DNS 구성에서 etcd 호스트 및 SRV 레코드를 지정할 필요가 없습니다.

작은 정보

dig 명령을 사용하여 이름과 역방향 이름을 확인할 수 있습니다. 자세한 검증 단계는 *사용자 프로비저닝 인프라의 DNS 확인* 섹션을 참조하십시오.

8.4.4.5.1. 사용자 프로비저닝 클러스터의 DNS 구성 예

이 섹션에서는 사용자 프로비저닝 인프라에 OpenShift Container Platform을 배포하기 위한 DNS 요구 사항을 충족하는 A 및 PTR 레코드 구성 샘플을 제공합니다. 샘플은 하나의 DNS 솔루션을 선택하기 위한 조언을 제공하기 위한 것이 아닙니다.

이 예제에서 클러스터 이름은 ocp4이고 기본 도메인은 example.com입니다.

사용자 프로비저닝 클러스터의 DNS A 레코드 구성 예

다음 BIND 영역 파일의 예제에서는 사용자가 프로비저닝한 클러스터의 이름 확인을 위한 샘플 A 레코드를 보여줍니다.

예 8.7. 샘플 DNS 영역 데이터베이스

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
master0.ocp4.example.com. IN A 192.168.1.97 5
master1.ocp4.example.com. IN A 192.168.1.98 6
master2.ocp4.example.com. IN A 192.168.1.99 7
```

```
;
worker0.ocp4.example.com. IN A 192.168.1.11 8
worker1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1

Kubernetes API의 이름 확인을 제공합니다. 레코드는 **API** 로드 밸런서의 **IP** 주소를 나타냅니다.

2

Kubernetes API의 이름 확인을 제공합니다. 레코드는 **API** 로드 밸런서의 **IP** 주소를 참조하며 내부 클러스터 통신에 사용됩니다.

3

와일드카드 경로의 이름 확인을 제공합니다. 레코드는 애플리케이션 인그레스 로드 밸런서의 **IP** 주소를 나타냅니다. 애플리케이션 인그레스 로드 밸런서는 **Ingress** 컨트롤러 **Pod**를 실행하는 머신을 대상으로 합니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

참고

이 예제에서는 **Kubernetes API** 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 **API** 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

4

부트스트랩 시스템의 이름 확인을 제공합니다.

5 6 7

컨트롤 플레인 시스템의 이름 확인을 제공합니다.

8 9

컴퓨팅 시스템의 이름 확인을 제공합니다.

사용자 프로비저닝 클러스터의 **DNS PTR** 레코드 구성 예

다음 예제 BIND 영역 파일은 사용자 프로비저닝 클러스터의 역방향 이름 확인을 위한 샘플 PTR 레코드를 보여줍니다.

예 8.8. 역방향 레코드의 샘플 DNS 영역 데이터베이스

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR master0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR master1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR master2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com. 8
;
;EOF
```

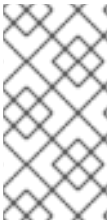
1 Kubernetes API의 역방향 DNS 확인을 제공합니다. PTR 레코드는 API 로드 밸런서의 레코드 이름을 참조합니다.

2 Kubernetes API의 역방향 DNS 확인을 제공합니다. PTR 레코드는 API 로드 밸런서의 레코드 이름을 참조하며 내부 클러스터 통신에 사용됩니다.

3 부트스트랩 시스템의 역방향 DNS 확인을 제공합니다.

4 5 6 컨트롤 플레인 시스템의 역방향 DNS 확인을 제공합니다.

7 8 컴퓨팅 시스템의 역방향 DNS 확인을 제공합니다.



참고

OpenShift Container Platform 애플리케이션 와일드카드에는 **PTR** 레코드가 필요하지 않습니다.

추가 리소스

- 사용자 프로비저닝 인프라에 대한 DNS 확인 검증

8.4.4.6. 사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구사항

OpenShift Container Platform을 설치하기 전에 **API** 및 애플리케이션 인그레스 로드 밸런싱 인프라를 프로비저닝해야 합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 **API** 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

참고

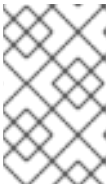
RHEL(Red Hat Enterprise Linux) 인스턴스를 사용하여 **API** 및 애플리케이션 인그레스 로드 밸런서를 배포하려면 **RHEL** 서브스크립션을 별도로 구매해야 합니다.

로드 밸런서 인프라는 다음 요구 사항을 충족해야 합니다.

1. **API** 로드 밸런서: 플랫폼과 상호 작용하고 플랫폼을 구성할 수 있도록 사용자(인간과 시스템 모두)에게 공통 끝점을 제공합니다. 다음 조건을 설정합니다.

   - **Layer 4** 로드 밸런싱 전용입니다. 이를 **Raw TCP, SSL Passthrough** 또는 **SSL Bridge** 모드라고 합니다. **SSL Bridge** 모드를 사용하는 경우, **API** 경로에 대해 **SNI(Server Name Indication,** 서버 이름 표시)를 활성화해야 합니다.

   - 스테이트리스 로드 밸런싱 알고리즘입니다. 옵션은 로드 밸런서 구현에 따라 달라집니다.



참고

API 로드 밸런서가 제대로 작동하기 위해 세션 지속성이 필요하지 않습니다.

로드 밸런서의 전면과 후면 모두에서 다음 포트를 구성하십시오.

표 8.39. API 로드 밸런서

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
6443	부트스트랩 및 컨트롤 플레인. 부트스트랩 시스템이 클러스터 컨트롤 플레인을 초기화한 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다. API 서버 상태 검사 프로브에 대한 **/readyz** 끝점을 구성해야 합니다.	X	X	Kubernetes API 서버
22623	부트스트랩 및 컨트롤 플레인. 부트스트랩 시스템이 클러스터 컨트롤 플레인을 초기화한 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다.	X		시스템 구성 서버

참고

API 서버가 /readyz 엔드포인트를 해제하는 시점부터 풀에서 API 서버 인스턴스가 제거되는 시점까지 시간이 30초를 넘지 않도록 로드 밸런서를 구성해야 합니다. /readyz가 오류를 반환하거나 정상 상태가 된 후 정해진 시간 안에 끝점이 제거 또는 추가되어야 합니다. 5초 또는 10초의 프로빙 주기(두 번의 성공적인 요청은 정상 상태, 세 번의 요청은 비정상 상태)는 충분한 테스트를 거친 값입니다.

2. 애플리케이션 인그레스 로드 밸런서: 클러스터 외부에서 유입되는 애플리케이션 트래픽에 대한 인그레스 포인트를 제공합니다. 다음 조건을 설정합니다.

- Layer 4 로드 밸런싱 전용입니다. 이를 Raw TCP, SSL Passthrough 또는 SSL Bridge 모드라고 합니다. SSL Bridge 모드를 사용하는 경우 인그레스 경로에 대해 SNI(Server Name Indication, 서버 이름 표시)를 활성화해야 합니다.

- 사용 가능한 옵션과 플랫폼에서 호스팅되는 애플리케이션 유형에 따라 연결 기반 또는 세션 기반 지속성이 권장됩니다.

작은 정보

애플리케이션 인그레스 로드 밸런서에서 클라이언트의 실제 IP 주소를 확인할 수 있는 경우 소스 IP 기반 세션 지속성을 활성화하면 엔드 투 엔드 TLS 암호화를 사용하는 애플리케이션의 성능을 향상시킬 수 있습니다.

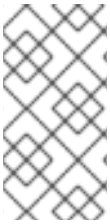
로드 밸런서의 전면과 후면 모두에서 다음 포트를 구성하십시오.

표 8.40. 애플리케이션 인그레스 로드 밸런서

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
443	기본적으로 인그레스 컨트롤러 pod, 컴퓨팅 또는 작업자를 실행하는 시스템입니다.	X	X	HTTPS 트래픽
80	기본적으로 인그레스 컨트롤러 pod, 컴퓨팅 또는 작업자를 실행하는 시스템입니다.	X	X	HTTP 트래픽
1936	기본적으로 Ingress 컨트롤러 Pod를 실행하는 작업자 노드입니다. 수신 상태 점검 프로브에 대해 **/healthz/ready** 끝점을 구성해야 합니다.	X	X	HTTP 트래픽

참고

컴퓨팅 노드가 0인 3-노드 클러스터를 배포하는 경우 Ingress 컨트롤러 Pod는 컨트롤 플레인 노드에서 실행됩니다. 3-노드 클러스터 배포에서 HTTP 및 HTTPS 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다.



참고

인그레스 라우터에 대한 작업 구성이 OpenShift Container Platform 클러스터에 필요합니다. 컨트롤 플레인 초기화 후 인그레스 라우터를 설정해야 합니다.

8.4.4.6.1. 사용자 프로비저닝 클러스터의 로드 밸런서 구성 예

이 섹션에서는 사용자 프로비저닝 클러스터의 로드 밸런싱 요구 사항을 충족하는 API 및 애플리케이션 수신 로드 밸런서 구성 예를 제공합니다. 샘플은 HAProxy 로드 밸런서에 대한 **/etc/haproxy/haproxy.cfg** 구성입니다. 이 예제에서는 하나의 로드 밸런싱 솔루션을 선택하기 위한 조언을 제공하는 것을 목적으로 하지 않습니다.

참고

이 예제에서는 **Kubernetes API** 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 **API** 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

예 8.9. API 및 애플리케이션 인그레스 로드 밸런서 구성 샘플

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
frontend stats
  bind *:1936
  mode            http
  log             global
  maxconn 10
  stats enable
  stats hide-version
  stats refresh 30s
  stats show-node
  stats show-desc Stats for ocp4 cluster (1)
  stats auth admin:ocp4
  stats uri /stats
listen api-server-6443 (2)
  bind *:6443
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:6443 check inter 1s backup (3)
  server master0 master0.ocp4.example.com:6443 check inter 1s
  server master1 master1.ocp4.example.com:6443 check inter 1s
  server master2 master2.ocp4.example.com:6443 check inter 1s
listen machine-config-server-22623 (4)
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup (5)
```

```
    server master0 master0.ocp4.example.com:22623 check inter 1s
    server master1 master1.ocp4.example.com:22623 check inter 1s
    server master2 master2.ocp4.example.com:22623 check inter 1s
  listen ingress-router-443 6
    bind *:443
    mode tcp
    balance source
    server worker0 worker0.ocp4.example.com:443 check inter 1s
    server worker1 worker1.ocp4.example.com:443 check inter 1s
  listen ingress-router-80 7
    bind *:80
    mode tcp
    balance source
    server worker0 worker0.ocp4.example.com:80 check inter 1s
    server worker1 worker1.ocp4.example.com:80 check inter 1s
```

1

이 예에서 클러스터 이름은 **ocp4** 입니다.

2

포트 **6443**은 **Kubernetes API** 트래픽을 처리하고 컨트롤 플레인 시스템을 가리킵니다.

3 5

부트스트랩 항목은 **OpenShift Container Platform** 클러스터 설치 전에 있어야 하며 부트스트랩 프로세스가 완료된 후 제거해야 합니다.

4

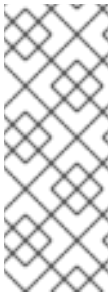
포트 **22623**은 머신 구성 서버 트래픽을 처리하고 컨트롤 플레인 시스템을 가리킵니다.

6

포트 **443**은 **HTTPS** 트래픽을 처리하고 **Ingress** 컨트롤러 **Pod**를 실행하는 시스템을 가리킵니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

7

포트 **80**은 **HTTP** 트래픽을 처리하고 **Ingress** 컨트롤러 **Pod**를 실행하는 머신을 가리킵니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

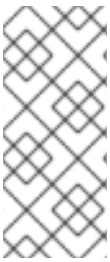


참고

컴퓨팅 노드가 0인 3-노드 클러스터를 배포하는 경우 Ingress 컨트롤러 Pod는 컨트롤 플레인 노드에서 실행됩니다. 3-노드 클러스터 배포에서 HTTP 및 HTTPS 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다.

작은 정보

HAProxy를 로드 밸런서로 사용하는 경우 HAProxy 노드에서 netstat -nltupe를 실행하여 haproxy 프로세스가 포트 6443, 22623, 443 및 80에서 수신 대기 중인지 확인할 수 있습니다.



참고

HAProxy를 로드 밸런서로 사용하고 SELinux가 enforcing으로 설정된 경우 HAProxy 서비스가 setsebool -P haproxy_connect_any=1을 실행하여 구성된 TCP 포트에 바인딩할 수 있는지 확인해야 합니다.

8.4.5. 사용자 프로비저닝 인프라 준비

사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치하기 전에 기본 인프라를 준비해야 합니다.

이 섹션에서는 OpenShift Container Platform 설치를 준비하기 위해 클러스터 인프라를 설정하는 데 필요한 높은 수준의 단계에 대해 자세히 설명합니다. 여기에는 클러스터 노드에 대한 IP 네트워킹 및 네트워크 연결 구성, 방화벽을 통해 필요한 포트 활성화, 필수 DNS 및 로드 밸런싱 인프라를 설정하는 작업이 포함됩니다.

준비 후 클러스터 인프라는 *사용자 프로비저닝 인프라가 있는 클러스터의 요구 사항* 섹션에 설명된 요구 사항을 충족해야 합니다.

사전 요구 사항

- OpenShift Container Platform 4.x 테스트된 통합 페이지를 검토했습니다.
- *사용자 프로비저닝 인프라가 있는 클러스터의 요구 사항* 섹션에 자세히 설명된 인프라 요구 사항을 검토했습니다.

프로세스

1. **DHCP**를 사용하여 클러스터 노드에 **IP** 네트워킹 구성을 제공하는 경우 **DHCP** 서비스를 구성합니다.

   a. 노드의 영구 **IP** 주소를 **DHCP** 서버 구성에 추가합니다. 구성에서 관련 네트워크 인터페이스의 **MAC** 주소를 각 노드의 의도한 **IP** 주소와 일치시킵니다.

   b. **DHCP**를 사용하여 클러스터 시스템의 **IP** 주소 지정을 구성하는 경우 시스템은 **DHCP**를 통해 **DNS** 서버 정보도 가져옵니다. **DHCP** 서버 구성을 통해 클러스터 노드에서 사용하는 영구 **DNS** 서버 주소를 정의합니다.

   참고

   DHCP 서비스를 사용하지 않는 경우 **RHCOS** 설치 시 **IP** 네트워킹 구성과 **DNS** 서버의 주소를 노드에 제공해야 합니다. **ISO** 이미지에서 설치하는 경우 부팅 인수로 전달할 수 있습니다. 고정 **IP** 프로비저닝 *및 고급 네트워킹 옵션에 대한 자세한 내용은 **RHCOS** 설치 및 **OpenShift Container Platform** 부트스트랩 프로세스 시작* 섹션을 참조하십시오.

   c. **DHCP** 서버 구성에 클러스터 노드의 호스트 이름을 정의합니다. 호스트 이름 고려 사항에 대한 자세한 내용은 ***DHCP**를 통해 클러스터 노드 호스트 이름 설정* 섹션을 참조하십시오.

   참고

   DHCP 서비스를 사용하지 않는 경우 클러스터 노드는 역방향 **DNS** 조회를 통해 호스트 이름을 가져옵니다.

2. 네트워크 인프라가 클러스터 구성 요소 간 필수 네트워크 연결을 제공하는지 확인합니다. 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 인프라 섹션의 네트워킹 요구 사항* 섹션을 참조하십시오.

3. **OpenShift Container Platform** 클러스터 구성 요소가 통신하는 데 필요한 포트를 활성화하도록 방화벽을 구성합니다. 필요한 포트에 대한 자세한 내용은 *사용자 프로비저닝 인프라 섹션의 네트워킹 요구 사항* 섹션을 참조하십시오.

4. 클러스터에 필요한 **DNS** 인프라를 설정합니다.

   a. **Kubernetes API**, 애플리케이션 와일드카드, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템의 **DNS** 이름 확인을 구성합니다.

   b. **Kubernetes API**, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템에 대한 역방향 **DNS** 확인을 구성합니다.

   OpenShift Container Platform DNS 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 DNS 요구 사항* 섹션을 참조하십시오.

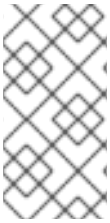
5. **DNS** 구성을 확인합니다.

   a. 설치 노드에서 **Kubernetes API**의 레코드 이름, 와일드카드 경로 및 클러스터 노드에 대해 **DNS** 조회를 실행합니다. 응답의 **IP** 주소가 올바른 구성 요소에 해당하는지 확인합니다.

   b. 설치 노드에서 로드 밸런서 및 클러스터 노드의 **IP** 주소에 대해 역방향 **DNS** 조회를 실행합니다. 응답의 레코드 이름이 올바른 구성 요소에 해당하는지 확인합니다.

   자세한 **DNS** 검증 단계는 *사용자 프로비저닝 인프라에 대한 DNS 확인* 섹션을 참조하십시오.

6. 필요한 **API** 및 애플리케이션 수신 로드 밸런싱 인프라를 프로비저닝합니다. 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구 사항* 섹션을 참조하십시오.



참고

일부 로드 밸런싱 솔루션에는 로드 밸런싱을 초기화하기 전에 클러스터 노드의 **DNS** 이름을 확인해야 합니다.

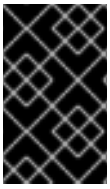
추가 리소스

- 사용자 프로비저닝 인프라를 포함한 클러스터의 시스템 요구사항

- RHCOS 설치 및 OpenShift Container Platform 부트스트랩 프로세스 시작
- DHCP를 통해 클러스터 노드의 호스트 이름 설정
- 고급 RHCOS 설치 구성 옵션
- 사용자 프로비저닝 인프라에 대한 네트워킹 요구사항
- 사용자 프로비저닝 DNS 요구사항
- 사용자 프로비저닝 인프라에 대한 DNS 확인 검증
- 사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구사항

8.4.6. 사용자 프로비저닝 인프라에 대한 DNS 확인 검증

사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치하기 전에 DNS 구성을 확인할 수 있습니다.



중요

클러스터를 설치하기 전에 이 섹션에 설명된 검증 단계를 성공해야 합니다.

사전 요구 사항

- 사용자 프로비저닝 인프라에 필요한 DNS 레코드를 구성했습니다.

프로세스

1. 설치 노드에서 Kubernetes API의 레코드 이름, 와일드카드 경로 및 클러스터 노드에 대해 DNS 조회를 실행합니다. 응답에 포함된 IP 주소가 올바른 구성 요소에 해당하는지 확인합니다.

   a. Kubernetes API 레코드 이름을 조회합니다. 결과가 API 로드 밸런서의 IP 주소를 가리

키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
```

1

<nameserver_ip>를 네임서버의 IP 주소로, <cluster_name>을 클러스터 이름으로, <base_domain>을 기본 도메인 이름으로 바꿉니다.

출력 예

```
api.ocp4.example.com.  0 IN A 192.168.1.5
```

b.

Kubernetes 내부 API 레코드 이름을 조회합니다. 결과가 API 로드 밸런서의 IP 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
```

출력 예

```
api-int.ocp4.example.com.  0 IN A 192.168.1.5
```

c.

예제 *.apps.<cluster_name>.<base_domain>을 테스트합니다. DNS 와일드카드를 조회합니다. 모든 애플리케이션 와일드카드 조회는 애플리케이션 인그레스 로드 밸런서의 IP 주소로 확인되어야 합니다.

```
$ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.
<base_domain>
```

출력 예

```
random.apps.ocp4.example.com.  0 IN A 192.168.1.5
```

참고

예제 출력에서는 Kubernetes API 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 API 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

random 항목을 다른 와일드카드 값으로 교체할 수 있습니다. 예를 들어 OpenShift Container Platform 콘솔의 경로를 쿼리할 수 있습니다.

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

출력 예

```
console-openshift-console.apps.ocp4.example.com. 0 IN A 192.168.1.5
```

d.

부트스트랩 DNS 레코드 이름에 대해 조회를 실행합니다. 결과가 부트스트랩 노드의 IP 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.
<base_domain>
```

출력 예

```
bootstrap.ocp4.example.com.  0 IN A 192.168.1.96
```

e.

이 방법을 사용하여 컨트롤 플레인 및 컴퓨팅 노드의 DNS 레코드 이름에 대해 조회를 수행합니다. 결과가 각 노드의 IP 주소에 해당하는지 확인합니다.

2.

설치 노드에서 로드 밸런서 및 클러스터 노드의 IP 주소에 대해 역방향 DNS 조회를 실행합니다. 응답에 포함된 레코드 이름이 올바른 구성 요소에 해당하는지 확인합니다.

a.

API 로드 밸런서의 IP 주소에 대해 역방향 조회를 수행합니다. 응답에 Kubernetes API 및 Kubernetes 내부 API의 레코드 이름이 포함되어 있는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

출력 예

```
5.1.168.192.in-addr.arpa. 0 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 0 IN PTR api.ocp4.example.com. 2
```

1

Kubernetes 내부 API의 레코드 이름을 제공합니다.

2

Kubernetes API의 레코드 이름을 제공합니다.

참고

OpenShift Container Platform 애플리케이션 와일드카드에는 PTR 레코드가 필요하지 않습니다. 애플리케이션 인그레스 로드 밸런서의 IP 주소에 대한 역방향 DNS 확인에는 유효성 검사 단계가 필요하지 않습니다.

b.

부트스트랩 노드의 IP 주소에 대해 역방향 조회를 수행합니다. 결과가 부트스트랩 노드의 DNS 레코드 이름을 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

출력 예

```
96.1.168.192.in-addr.arpa. 0 IN PTR bootstrap.ocp4.example.com.
```

c. 이 방법을 사용하여 컨트롤 플레인 및 컴퓨팅 노드의 IP 주소에 대해 역방향 조회를 수행합니다. 결과가 각 노드의 DNS 레코드 이름과 일치하는지 확인합니다.

추가 리소스

- 사용자 프로비저닝 DNS 요구사항
- 사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구사항

8.4.7. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 core 사용자의 ~/.ssh/authorized_keys 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 core로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. ./openshift-install gather 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

참고

AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: ~/.ssh/id_ed25519)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 '~/.ssh 디렉터리에 있는지 확인하십시오.

참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 ed25519 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 rsa 또는 ecdsa 알고리즘을 사용하는 키를 생성합니다.

2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 ~/.ssh/id_ed25519.pub 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 ./openshift-install gather 명령을 사용하려는 경우 필요합니다.

참고

일부 배포에서는 ~/.ssh/id_rsa 및 ~/.ssh/id_dsa와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. ssh-agent 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. ssh-agent에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: ~/.ssh/id_ed25519).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- **OpenShift Container Platform**을 설치할 때 **SSH** 공개 키를 설치 프로그램에 지정합니다. 프로비저닝하는 인프라에 클러스터를 설치하는 경우 설치 프로그램에 키를 제공해야 합니다.

추가 리소스

- 노드 상태 확인

8.4.8. 수동으로 설치 구성 파일 만들기

OpenShift Container Platform을 사용자가 프로비저닝한 설치의 경우 설치 구성 파일을 수동으로 생성합니다.

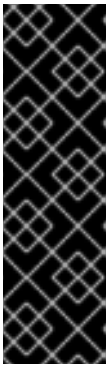
사전 요구 사항

- 로컬 시스템에 설치 프로그램에 제공할 **SSH** 공개 키가 있습니다. 키는 디버깅 및 재해 복구를 위해 클러스터 노드에 대한 **SSH** 인증에 사용됩니다.
- **OpenShift Container Platform** 설치 프로그램과 클러스터의 풀 시크릿이 있습니다.
- 명령 출력에서 **imageContentSources** 섹션을 가져와서 리포지토리를 미러링합니다.
- 미러 레지스트리에 대한 인증서의 내용을 가져옵니다.

프로세스

1. 필요한 설치 자산을 저장할 설치 디렉터리를 만듭니다.

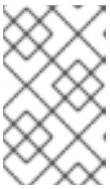
```
$ mkdir <installation_directory>
```



중요

디렉터리를 만들어야 합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

2. 샘플 install-config.yaml 파일 템플릿을 사용자 지정하여 <installation_directory>에 저장합니다.



참고

이 설정 파일의 이름을 install-config.yaml로 지정해야 합니다.

- docker.io와 같이 RHCOS가 기본적으로 신뢰하는 레지스트리를 사용하는 경우를 제외하고, additionalTrustBundle 섹션에 있는 미러 리포지토리에 대한 인증서 내용을 제공해야 합니다. 대부분의 경우 미러 인증서를 제공해야 합니다.

- 리포지토리를 미러링하려면 명령 출력의 imageContentSources 섹션을 삽입해야 합니다.

참고

일부 플랫폼 유형의 경우 대체로 ./openshift-install create install-config --dir <installation_directory> 를 실행하여 install-config.yaml 파일을 생성할 수 있습니다. 프롬프트에서 클러스터 구성에 대한 세부 정보를 제공할 수 있습니다.

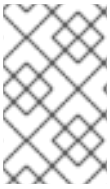
3. 여러 클러스터를 설치하는 데 사용할 수 있도록 install-config.yaml 파일을 백업합니다.

중요

install-config.yaml 파일은 설치 과정의 다음 단계에서 사용됩니다. 이 시점에서 이를 백업해야 합니다.

8.4.8.1. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 환경에 대한 세부 정보를 설명하는 사용자 지정 **install-config.yaml** 설치 구성 파일을 제공합니다.



참고

설치한 후에는 install-config.yaml 파일에서 이러한 매개변수를 수정할 수 없습니다.

중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

8.4.8.1.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 8.41. 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체

매개변수	설명	값
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	**dev** 와 같은 소문자 및 하이픈(-)의 문자열입니다.
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	{
		"auths":{
		"cloud.openshift.com":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		},
		"quay.io":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		}
		}
		}

8.4.8.1.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 8.42. 네트워크 매개변수

매개변수	설명	값

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체 **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.
networking.clusterNetwork	Pod의 IP 주소 블록입니다. 기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된 **cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16

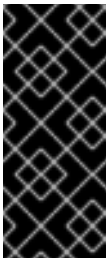
매개변수	설명	값
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16** **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

8.4.8.1.3. 선택적 구성 매개변수

선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 8.43. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열

매개변수	설명	값
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은 **3**입니다.
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열

매개변수	설명	값
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 기본값은 **External**입니다. 이 필드를 **Internal** 로 설정하는 것은 클라우드 이외의 플랫폼에서는 지원되지 않습니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

8.4.8.2. 베어 메탈의 샘플 install-config.yaml 파일

install-config.yaml 파일을 사용자 지정하여 **OpenShift Container Platform** 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.

```
apiVersion: v1
baseDomain: example.com 1
compute: 2
- hyperthreading: Enabled 3
  name: worker
  replicas: 0 4
controlPlane: 5
  hyperthreading: Enabled 6
  name: master
  replicas: 3 7
metadata:
  name: test 8
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 9
    hostPrefix: 23 10
  networkType: OpenShiftSDN
  serviceNetwork: 11
  - 172.30.0.0/16
platform:
  none: {} 12
fips: false 13
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email":
"you@example.com"}}}' 14
sshKey: 'ssh-ed25519 AAAA...' 15
additionalTrustBundle: | 16
  -----BEGIN CERTIFICATE-----
  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
```

```
  -----END CERTIFICATE-----
imageContentSources: 17
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

1

클러스터의 기본 도메인입니다. 모든 **DNS** 레코드는 이 기본 도메인의 하위 도메인이어야 하며 클러스터 이름을 포함해야 합니다.

2 5

controlPlane 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

3 6

동시 멀티스레딩(**SMT**) 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 **SMT**가 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. **SMT**를 비활성화하는 경우 모든 클러스터 머신에서 이를 비활성화해야 합니다. 여기에는 컨트롤 플레인과 컴퓨팅 머신이 모두 포함됩니다.

참고

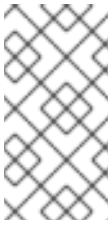
SMT(동시 멀티 스레딩)는 기본적으로 활성화되어 있습니다. **BIOS** 설정에서 **SMT**를 활성화하지 않으면 **hyperthreading** 매개변수가 적용되지 않습니다.

중요

BIOS에서든 **install-config.yaml** 파일에서든 **hyperthreading**을 비활성화한 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.

4

사용자 프로비저닝 인프라에 **OpenShift Container Platform**을 설치할 때 이 값을 **0**으로 설정해야 합니다. 설치 프로그램에서 제공하는 설치에서 매개 변수는 클러스터가 생성 및 관리하는 컴퓨팅 머신 수를 제어합니다. 사용자 프로비저닝 설치에서는 클러스터 설치를 완료하기 전에 컴퓨팅 시스템을 수동으로 배포해야 합니다.



참고

3-노드 클러스터를 설치하는 경우 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템을 설치할 때 컴퓨팅 머신을 배포하지 마십시오.

7

클러스터에 추가하는 컨트롤 플레인 시스템의 수입니다. 클러스터에서 이 값을 클러스터의 etcd 끝점 수로 사용하므로 이 값은 배포하는 컨트롤 플레인 시스템의 수와 일치해야 합니다.

8

DNS 레코드에 지정한 클러스터 이름입니다.

9

Pod IP 주소가 할당되는 IP 주소 블록입니다. 이 블록은 기존 물리적 네트워크와 중복되지 않아야합니다. 이러한 IP 주소는 Pod 네트워크에 사용됩니다. 외부 네트워크에서 Pod에 액세스해야 하는 경우, 트래픽을 관리하도록 로드 밸런서와 라우터를 설정해야 합니다.

참고

클래스 E CIDR 범위는 향후 사용을 위해 예약되어 있습니다. 클래스 E CIDR 범위를 사용하려면 네트워킹 환경에서 클래스 E CIDR 범위 내에서 IP 주소를 수락하는지 확인해야 합니다.

10

개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 hostPrefix를 23으로 설정하면 지정된 cidr 이외 /23 서브넷이 각 노드에 할당되어 510(2^(32 - 23) - 2) Pod IP 주소가 허용됩니다. 외부 네트워크에서 노드에 액세스해야 하는 경우 트래픽을 관리하도록 로드 밸런서와 라우터를 구성합니다.

11

서비스 IP 주소에 사용할 IP 주소 풀입니다. IP 주소 풀은 하나만 입력할 수 있습니다. 이 블록은 기존 물리적 네트워크와 중복되지 않아야합니다. 외부 네트워크에서 서비스에 액세스해야 하는 경우, 트래픽을 관리하도록 로드 밸런서와 라우터를 구성합니다.

12

플랫폼을 none으로 설정해야 합니다. 플랫폼에 대한 추가 플랫폼 구성 변수는 지정할 수 없습니다.

중요

플랫폼 유형으로 설치된 클러스터는 **Machine API**로 컴퓨팅 머신 관리와 같은 일부 기능을 사용할 수 없습니다. 이 제한은 클러스터에 연결된 컴퓨팅 시스템이 일반적으로 기능을 지원하는 플랫폼에 설치된 경우에도 적용됩니다. 이 매개변수는 설치 후 변경할 수 없습니다.

13

FIPS 모드 활성화 또는 비활성화 여부입니다. 기본적으로 **FIPS** 모드는 비활성화됩니다. **FIPS** 모드가 활성화되면 **OpenShift Container Platform**이 실행되는 **RHCOS(Red Hat Enterprise Linux CoreOS)** 시스템에서 기본 **Kubernetes** 암호화 제품군은 우회하고 **RHCOS**와 함께 제공되는 암호화 모듈을 대신 사용합니다.

중요

FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 **OpenShift Container Platform** 배포에서만 지원됩니다.

14

<local_registry>는 미러 레지스트리가 해당 내용을 제공하는 데 사용하는 레지스트리 도메인 이름과 포트(선택사항)를 지정합니다. 예: **registry.example.com** 또는 **registry.example.com:5000 <credentials>**는 미러 레지스트리의 **base64** 인코딩 사용자 이름과 암호를 지정합니다.

15

RHCOS(Red Hat Enterprise Linux CoreOS)의 **core** 사용자에 대한 **SSH** 공용 키입니다.

참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 **OpenShift Container Platform** 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 **SSH** 키를 지정합니다.

16

미러 레지스트리에 사용한 인증서 파일의 내용을 제공하십시오.

17

명령 출력에서 **imageContentSources** 섹션을 제공하여 리포지토리를 미러링하십시오.

추가 리소스

- API 및 애플리케이션 수신 로드 밸런싱 요구 사항에 대한 자세한 내용은 사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구 사항을 참조하십시오.

8.4.8.3. 설치 중 클러스터 단위 프록시 구성

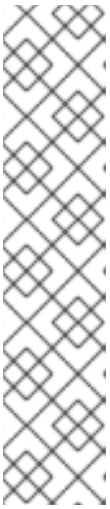
프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 HTTP 또는 HTTPS 프록시를 사용할 수 있습니다. install-config.yaml 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 OpenShift Container Platform 클러스터를 구성할 수 있습니다.

참고

바이메탈 설치의 경우, install-config.yaml 파일의 networking.machineNetwork[].cidr 필드에 지정된 범위 밖의 노드 IP 주소를 지정하지 않는 경우, proxy.noProxy 필드에 포함시켜야 합니다.

사전 요구 사항

- 기존 install-config.yaml 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 API에 대한 호출을 포함하여 모든 클러스터 발신(Egress) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 Proxy 오브젝트의 spec.noProxy 필드에 사이트를 추가했습니다.



참고

Proxy 오브젝트의 status.noProxy 필드는 설치 구성에 있는 networking.machineNetwork[].cidr, networking.clusterNetwork[].cidr, networking.serviceNetwork[] 필드의 값으로 채워집니다.

Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 Red Hat OpenStack Platform (RHOSP)에 설치하는 경우 Proxy 오브젝트 status.noProxy 필드도 인스턴스 메타데이터 끝점(169.254.169.254)로 채워집니다.

절차

1. install-config.yaml 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니

다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1

클러스터 외부에서 **HTTP** 연결을 구축하는 데 사용할 프록시 **URL**입니다. **URL** 스키마는 **http**여야 합니다.

2

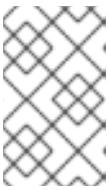
클러스터 외부에서 **HTTPS** 연결을 구축하는 데 사용할 프록시 **URL**입니다.

3

대상 도메인 이름, **IP** 주소 또는 프록시에서 제외할 기타 네트워크 **CIDR**로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. ***를 사용하여 모든 대상에 대해 프록시를 바이패스합니다.

4

이 값을 제공하면 설치 프로그램에서 추가 **CA** 인증서를 보유할 **openshift -config** 네임스페이스에 **user-ca- bundle** 이라는 구성 맵을 생성합니다. **additionalTrustBundle** 및 하나 이상의 프록시 설정을 제공하는 경우 프록시 오브젝트는 **trustedCA** 필드의 **user-ca-bundle** 구성 맵을 참조하도록 구성됩니다. 그러면 **Cluster Network Operator**에서 **trustedCA** 매개변수에 대해 지정된 콘텐츠를 **RHCOS** 신뢰 번들과 병합하는 **trusted-ca-bundle** 구성 맵을 생성합니다. 프록시의 **ID** 인증서를 **RHCOS** 트러스트 번들에 있는 기관에서 서명하지 않은 경우 **additionalTrustBundle** 필드가 있어야 합니다.

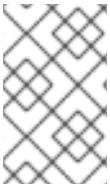


참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 **OpenShift Container Platform**을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 **nil**이 됩니다.



참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

8.4.8.4. 3개의 노드 클러스터 구성

필요한 경우 세 개의 컨트롤 플레인 시스템으로 구성된 베어 메탈 클러스터에 제로 컴퓨팅 머신을 배포할 수 있습니다. 이를 통해 클러스터 관리자와 개발자들이 테스트, 개발, 프로덕션에 사용할 수 있는 소형화되고 리소스 효율이 높은 클러스터를 제공합니다.

3-노드 **OpenShift Container Platform** 환경에서 세 개의 컨트롤 플레인 머신을 예약할 수 있습니다. 즉, 애플리케이션 워크로드가 해당 플랫폼에서 실행되도록 예약됩니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.

절차

- **install-config.yaml** 파일에서 다음 **compute** 스탠자에 표시된 대로 컴퓨팅 복제본 수가 **0**으로 설정되어 있는지 확인합니다.

```
compute:
- name: worker
  platform: {}
  replicas: 0
```

참고

배포 중인 컴퓨팅 머신 수에 관계없이 사용자 프로비저닝 인프라에 **OpenShift Container Platform**을 설치할 때 컴퓨팅 머신의 **replicas** 매개변수 값을 **0**으로 설정해야 합니다. 설치 프로그램에서 제공하는 설치에서 매개 변수는 클러스터가 생성 및 관리하는 컴퓨팅 머신 수를 제어합니다. 이 설정은 컴퓨팅 시스템이 수동으로 배포되는 사용자 프로비저닝 설치에는 적용되지 않습니다.

3-노드 클러스터 설치의 경우 다음 단계를 따르십시오.

- 컴퓨팅 노드가 0인 3-노드 클러스터를 배포하는 경우 Ingress 컨트롤러 Pod는 컨트롤 플레인 노드에서 실행됩니다. 3-노드 클러스터 배포에서 HTTP 및 HTTPS 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다. 자세한 내용은 *사용자 프로비저닝 인프라 섹션에 대한 로드 밸런싱 요구 사항* 섹션을 참조하십시오.

- 다음 절차에서 Kubernetes 매니페스트 파일을 생성할 때 <installation_directory>/manifests/cluster-scheduler-02-config.yml 파일의 mastersSchedulable 매개변수가 true로 설정되어 있는지 확인합니다. 이렇게 하면 애플리케이션 워크로드를 컨트롤 플레인 노드에서 실행할 수 있습니다.

- RHCOS(Red Hat Enterprise Linux CoreOS) 시스템을 생성할 때 컴퓨팅 노드를 배포하지 마십시오.

8.4.9. Kubernetes 매니페스트 및 Ignition 설정 파일 생성

일부 클러스터 정의 파일을 수정하고 클러스터 시스템을 수동으로 시작해야 하므로 클러스터가 시스템을 구성하는 데 필요한 Kubernetes 매니페스트 및 Ignition 구성 파일을 사용자가 생성해야 합니다.

설치 구성 파일은 Kubernetes 매니페스트로 변환됩니다. 매니페스트는 나중에 클러스터 머신을 구성하는 데 사용되는 Ignition 구성 파일로 래핑됩니다.

중요

- OpenShift Container Platform 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 node-bootstrapper 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 ***Recovering from expired control plane certificates*** 문서를 참조하십시오.

- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램을 가져오셨습니다. 제한된 네트워크 설치의 경우, 해당 파일은 미러 호스트에 있습니다.
- install-config.yaml 설치 구성 파일을 생성하셨습니다.

절차

1. OpenShift Container Platform 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터에 대한 Kubernetes 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 <installation_directory>는 사용자가 만든 install-config.yaml 파일이 포함된 설치 디렉터리를 지정합니다.

주의

3 노드 클러스터를 실행 중이면 다음 단계를 건너 뛰어 컨트롤 플레인 노드 일정을 계획할 수 있도록 하십시오.

중요

예약할 수 없는 기본에서 컨트롤 플레인 노드를 구성하면 추가 서브스크립션이 필요합니다. 이는 컨트롤 플레인 노드가 작업자 노드가 되기 때문입니다.

2. <installation_directory>/manifests/cluster-scheduler-02-config.yml Kubernetes 매니페스트 파일의 mastersSchedulable 매개변수가 false로 설정되어 있는지 확인합니다. 이 설정으로 인해 컨트롤 플레인 머신에서 포드가 예약되지 않습니다.

a. <installation_directory>/manifests/cluster-scheduler-02-config.yml 파일을 엽니다.

b. **mastersSchedulable** 매개변수를 찾아서 값을 **False**로 설정되어 있는지 확인합니다.

c. 파일을 저장하고 종료합니다.

3. **Ignition** 구성 파일을 생성하려면 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행합니다.

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>**는 동일한 설치 디렉터리를 지정합니다.

설치 디렉터리의 부트스트랩, 컨트롤 플레인 및 컴퓨팅 노드에 대한 **Ignition** 구성 파일이 생성됩니다. **kubeadmin-password** 및 **kubeconfig** 파일은 **./<installation_directory>/auth** 디렉터리에 생성됩니다.

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

추가 리소스

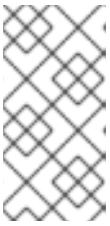
- **kubelet** 인증서 복구에 대한 자세한 내용은 만료된 컨트롤 플레인 인증서 복구에서 참조하십시오.

8.4.10. chrony 타임 서비스 설정

chrony.conf 파일의 내용을 수정하고 해당 내용을 머신 구성으로 노드에 전달하여 **chrony** 타임 서비스 **(chronyd)**에서 사용하는 시간 서버 및 관련 구성을 설정해야 합니다.

절차

1. **chrony.conf** 파일의 내용을 포함하여 **Butane config**를 만듭니다. 예를 들어 작업자 노드에 **chrony**를 구성하려면 **99-worker-chrony.bu** 파일을 만듭니다.



참고

Butane에 대한 자세한 내용은 “Butane 을 사용하여 머신 구성 생성”을 참조하십시오.

```
variant: openshift
version: 4.9.0
metadata:
  name: 99-worker-chrony (1)
  labels:
    machineconfiguration.openshift.io/role: worker (2)
storage:
  files:
  - path: /etc/chrony.conf
    mode: 0644 (3)
    overwrite: true
    contents:
      inline: |
        pool 0.rhel.pool.ntp.org iburst (4)
        driftfile /var/lib/chrony/drift
        makestep 1.0 3
        rtcsync
        logdir /var/log/chrony
```

(1) (2)

컨트롤 플레인 노드에서 두 위치에 있는 master를 worker로 대체합니다.

(3)

시스템 구성 파일에서 mode 필드의 8진수 값 모드를 지정합니다. 파일을 만들고 변경 사항을 적용하면 모드가 10진수 값으로 변환됩니다. oc get mc <mc-name> -o yaml 명령을 사용하여 YAML 파일을 확인할 수 있습니다.

(4)

DHCP 서버에서 제공하는 것과 같은 유효한 시간 소스를 지정합니다.

2. Butane을 사용하여 노드에 전달할 구성이 포함된 MachineConfig 파일 99-worker-chrony.yaml을 생성합니다.

```
$ butane 99-worker-chrony.bu -o 99-worker-chrony.yaml
```

3. 다음 두 가지 방법 중 하나로 설정을 적용하십시오.

- 클러스터가 아직 실행되지 않은 경우 매니페스트 파일을 생성한 후 **<installation_directory>/openshift** 디렉터리에 **MachineConfig** 개체 파일을 추가한 다음 클러스터를 계속 작성합니다.

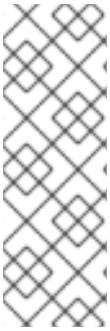
- 클러스터가 이미 실행중인 경우 다음과 같은 파일을 적용합니다.

```
$ oc apply -f ./99-worker-chrony.yaml
```

8.4.11. RHCOS 설치 및 OpenShift Container Platform 부트스트랩 프로세스 시작

프로비저닝하는 베어메탈 인프라에 **OpenShift Container Platform**을 설치하려면 머신에 **RHCOS(Red Hat Enterprise Linux CoreOS)**를 설치해야 합니다. **RHCOS**를 설치할 때 설치 중인 머신 유형에 대해 **OpenShift Container Platform** 설치 프로그램에서 생성한 **Ignition** 구성 파일을 제공해야 합니다. 적합한 네트워킹, **DNS** 및 로드 밸런싱 인프라를 구성한 경우 **RHCOS** 머신이 재부팅된 후 **OpenShift Container Platform** 부트스트랩 프로세스가 자동으로 시작됩니다.

단계에 따라 **ISO** 이미지 또는 네트워크 **PXE** 부팅을 사용하여 시스템에 **RHCOS**를 설치합니다.



참고

이 설치 문서에 포함된 컴퓨팅 노드 배포 단계는 **RHCOS**에 따라 다릅니다. **RHEL** 기반 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. **RHEL 7** 컴퓨팅 머신 사용은 더 이상 사용되지 않으며 향후 **OpenShift Container Platform 4** 릴리스에서 제거될 예정입니다.

다음 방법을 사용하여 **ISO** 및 **PXE** 설치 중에 **RHCOS**를 구성할 수 있습니다.

- 커널 인수: 커널 인수를 사용하여 설치 관련 정보를 제공할 수 있습니다. 예를 들어 **HTTP** 서버에 업로드한 **RHCOS** 설치 파일의 위치와 설치 중인 노드 유형에 대한 **Ignition** 구성 파일의 위치를 지정할 수 있습니다. **PXE** 설치의 경우 **APPEND** 매개 변수를 사용하여 라이브 설치 프로그램의 커널에 인수를 전달할 수 있습니다. **ISO** 설치의 경우는 라이브 설치 부팅 프로세스를 중단하고 커널 매개 변수를 추가할 수 있습니다. 두 경우 모두 특정 **coreos.inst.** * 인수를 사용하여 라이브 설치 프로그램을 지시 할 수 있을 뿐 만 아니라 표준 커널 서비스를 활성화/비활성화하기 위해 표준 설치 부팅 인수를 사용할 수 있습니다.

-

Ignition 구성: OpenShift Container Platform Ignition 구성 파일(*.ign)은 설치 중인 노드 유형에 따라 다릅니다. **RHCOS** 설치 중에 부트스트랩, 컨트롤 플레인 또는 컴퓨팅 노드 **Ignition** 구성 파일의 위치를 전달하여 첫 번째 부팅 시 적용됩니다. 특별한 경우에는 라이브 시스템으로 전달할 별도의 제한된 **Ignition** 설정을 만들 수 있습니다. 이 **Ignition** 설정은 설치 완료 후 프로비저닝 시스템에 설치가 성공적으로 완료되었는지를 보고하는 것과 같은 일련의 작업을 수행할 수 있습니다. 이러한 특수 **Ignition** 구성은 설치된 시스템의 처음 부팅 시 적용되는 **coreos-installer**에 의해 소비됩니다. 라이브 **ISO**에 표준 컨트롤 플레인 및 컴퓨팅 노드 **Ignition** 구성을 직접 제공하지 마십시오.

- **coreos-installer :** 처음 부팅하기 전에 다양한 방법으로 영구 시스템을 준비 할 수 있도록 쉘 프롬프트에서 라이브 **ISO** 설치 프로그램을 시작할 수 있습니다. **coreos-installer** 명령을 실행하여 추가하는 다양한 아티팩트를 식별하고 디스크 파티션을 사용하여 네트워크를 설정할 수 있습니다. 경우에 따라 라이브 시스템에서 기능을 구성하고 설치된 시스템에 복사할 수도 있습니다.

ISO 또는 **PXE** 설치 사용 여부는 상황에 따라 달라집니다. **PXE** 설치에는 사용 가능한 **DHCP** 서비스와 추가 준비가 필요하지만 설치 프로세스를 보다 자동화할 수 있습니다. **ISO** 설치는 주로 수동적인 프로세스에서 여러 시스템을 설정하는 경우 불편할 수 있습니다.

참고

OpenShift Container Platform 4.6부터 **RHCOS ISO** 및 기타 설치 아티팩트는 **4K** 섹터 디스크에 설치를 지원합니다.

8.4.11.1. ISO 이미지를 사용하여 RHCOS 설치

ISO 이미지를 사용하여 시스템에 **RHCOS**을 설치할 수 있습니다.

사전 요구 사항

- 클러스터에 대한 **Ignition** 구성 파일이 생성되어 있습니다.
- 적합한 네트워크, **DNS** 및 로드 밸런싱 인프라가 구성되어 있습니다.
- 사용자 컴퓨터에서 액세스할 수 있고 사용자가 생성한 시스템에서 액세스할 수 있는 **HTTP** 서버에 대한 액세스 권한을 확보합니다.
- 네트워킹 및 디스크 파티션과 같은 기능을 구성하는 다양한 방법에 대해서는 *고급 **RHCOS** 설치 구성*섹션을 살펴보십시오.

프로세스

1. 각 Ignition 구성 파일에 대해 SHA512 다이제스트를 가져옵니다. 예를 들어 Linux를 실행하는 시스템에서 다음을 사용하여 bootstrap.ign Ignition 구성 파일의 SHA512 다이제스트를 가져올 수 있습니다.

```
$ sha512sum <installation_directory>/bootstrap.ign
```

다이제스트는 클러스터 노드에서 Ignition 구성 파일의 신뢰성을 검증하기 위해 이후 단계에서 coreos-installer에 제공됩니다.

2. 설치 프로그램에서 생성한 부트스트랩, 컨트롤 플레인, 컴퓨팅 노드 Ignition 구성 파일을 HTTP 서버에 업로드합니다. 해당 파일의 URL을 기록해 둡니다.

중요

HTTP 서버에 저장하기 전에 Ignition 구성에서 구성 설정을 추가하거나 변경할 수 있습니다. 설치를 마친 후 클러스터에 컴퓨팅 시스템을 더 추가하려면 Ignition 구성 파일을 삭제하지 마십시오.

3. 설치 호스트에서 Ignition 구성 파일을 URL에서 사용할 수 있는지 확인합니다. 다음 예에서는 부트스트랩 노드에 대한 Ignition 구성 파일을 가져옵니다.

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

출력 예

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-
rsa...
```

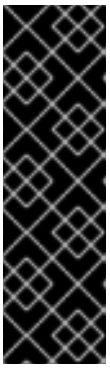
명령에서 bootstrap.ign을 master.ign 또는 worker.ign으로 교체하여 컨트롤 플레인 및 컴퓨팅 노드의 Ignition 구성 파일도 사용할 수 있는지 확인합니다.

4. RHCOS 이미지 미러 페이지에서 운영 체제 인스턴스 설치 방법에 필요한 RHCOS 이미지를 얻을 수 있지만 올바른 버전의 RHCOS 이미지를 얻는 것이 좋습니다. openshift-install 명령 출력에서 사용할 수 있습니다.

```
$ openshift-install coreos print-stream-json | grep '\.iso[^.]'
```

출력 예

```
"location": "<url>/art/storage/releases/rhcos-4.9-aarch64/<release>/aarch64/rhcos-
<release>-live.aarch64.iso",
"location": "<url>/art/storage/releases/rhcos-4.9-ppc64le/<release>/ppc64le/rhcos-
<release>-live.ppc64le.iso",
"location": "<url>/art/storage/releases/rhcos-4.9-s390x/<release>/s390x/rhcos-
<release>-live.s390x.iso",
"location": "<url>/art/storage/releases/rhcos-4.9/<release>/x86_64/rhcos-<release>-
live.x86_64.iso",
```



중요

RHCOS 이미지는 OpenShift Container Platform 릴리스에 따라 변경되지 않을 수 있습니다. 설치하는 OpenShift Container Platform 버전과 같거나 그 이하의 버전 중 가장 최신 버전의 이미지를 다운로드해야 합니다. 사용 가능한 경우 OpenShift Container Platform 버전과 일치하는 이미지 버전을 사용합니다. 이 프로세스에는 ISO 이미지만 사용하십시오. 이 설치 유형에서는 RHCOS qcow2 이미지가 지원되지 않습니다.

ISO 파일 이름은 다음 예와 유사합니다.

```
rhcos-<version>-live.<architecture>.iso
```

5. ISO를 사용하여 RHCOS 설치를 시작합니다. 다음 설치 옵션 중 하나를 사용합니다.

- ISO 이미지를 디스크에 굽고 직접 부팅합니다.
- LOM(Lightweight-out Management) 인터페이스를 사용하여 ISO 리디렉션을 사용합

니다.

6. 옵션을 지정하거나 라이브 부팅 시퀀스를 중단하지 않고 **RHCOS ISO** 이미지를 부팅합니다. 설치 프로그램이 **RHCOS** 라이브 환경에서 쉘 프롬프트로 부팅될 때까지 기다립니다.

참고

커널 인수를 추가하기 위해 **RHCOS** 설치 부팅 프로세스를 중단할 수 있습니다. 하지만 이 **ISO** 프로세스에서는 커널 인수를 추가하지 않고 다음 단계에 설명된 대로 **coreos-installer** 명령을 사용해야 합니다.

7. **coreos-installer** 명령을 실행하고 설치 요구 사항을 충족하는 옵션을 지정합니다. 최소한 노드 유형에 대한 **Ignition** 구성 파일과 설치할 장치를 가리키는 **URL**을 지정해야 합니다.

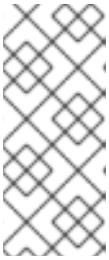
```
$ sudo coreos-installer install --ignition-url=http://<HTTP_server>/<node_type>.ign
<device> --ignition-hash=sha512-<digest> 1 2
```

1 1

core 사용자에게 설치를 수행하는 데 필요한 **root** 권한이 없으므로 **sudo**를 사용하여 **coreos-installer** 명령을 실행해야 합니다.

2

클러스터 노드에서 **Ignition** 구성 파일을 **HTTP URL**을 통해 가져오려면 **--ignition-hash** 옵션이 필요합니다. **<digest>**는 이전 단계에서 얻은 **Ignition** 구성 파일 **SHA512** 다이제스트입니다.



참고

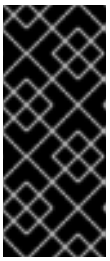
TLS를 사용하는 **HTTPS** 서버를 통해 **Ignition** 구성 파일을 제공하려는 경우 **coreos-installer**를 실행하기 전에 내부 인증 기관(**CA**)을 시스템 신뢰 저장소에 추가할 수 있습니다.

다음 예제에서는 **/dev/sda** 장치에 부트스트랩 노드 설치를 초기화합니다. 부트스트랩 노드의 **Ignition** 구성 파일은 **IP** 주소 **192.168.1.2**가 있는 **HTTP** 웹 서버에서 가져옵니다.

```
$ sudo coreos-installer install --ignition-
url=http://192.168.1.2:80/installation_directory/bootstrap.ign /dev/sda --ignition-
hash=sha512-
```

```
a5a2d43879223273c9b60af66b44202a1d1248fc01cf156c46d4a79f552b6bad47bc8cc78dd
f0116e80c59d2ea9e32ba53bc807afbca581aa059311def2c3e3b
```

8. 머신 콘솔에서 RHCOS 설치 진행률을 모니터링합니다.

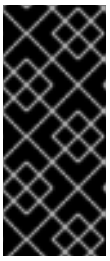


중요

OpenShift Container Platform 설치를 시작하기 전에 각 노드에서 성공적으로 설치되었는지 확인합니다. 설치 프로세스를 관찰하면 발생할 수 있는 RHCOS 설치 문제의 원인을 파악하는 데 도움이 될 수 있습니다.

9. RHCOS를 설치한 후 시스템을 재부팅해야 합니다. 시스템이 재부팅되는 동안 지정한 Ignition 구성 파일이 적용됩니다.

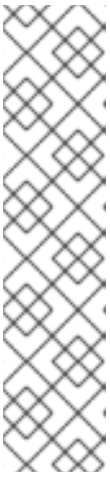
10. 계속해서 클러스터에 대한 나머지 시스템을 모두 생성합니다.



중요

이때 부트스트랩 및 컨트롤 플레인 시스템을 생성해야 합니다. 컨트롤 플레인 시스템이 예약 가능하지 않은 경우 OpenShift Container Platform을 설치하기 전에 두 개 이상의 컴퓨팅 시스템도 생성합니다.

적합한 네트워킹, DNS 및 로드 밸런싱 인프라가 있는 경우 RHCOS 노드가 재부팅된 후 OpenShift Container Platform 부트스트랩 프로세스가 자동으로 시작됩니다.



참고

RHCOS 노드에는 core 사용자의 기본 암호가 포함되지어 있지 않습니다. install_config.yaml 파일에 지정한 공개 키에 대한 액세스 권한이 있는 사용자로 ssh core@<node>.<cluster_name>.<base_domain>을 실행하여 노드에 액세스할 수 있습니다. RHCOS를 실행하는 OpenShift Container Platform 4 클러스터 노드는 변경할 수 없으며 Operator를 통해 클러스터 변경 사항을 적용합니다. SSH를 사용하여 클러스터 노드에 액세스하는 것은 권장되지 않습니다. 그러나 설치 문제를 조사할 때 OpenShift Container Platform API를 사용할 수 없거나 kubelet이 대상 노드에서 제대로 작동하지 않는 경우 디버깅 또는 재해 복구에 SSH 액세스가 필요할 수 있습니다.

8.4.11.2. PXE 또는 iPXE 부팅을 사용하여 RHCOS 설치

PXE 또는 iPXE 부팅을 사용하여 시스템에 RHCOS를 설치할 수 있습니다.

사전 요구 사항

- 클러스터에 대한 Ignition 구성 파일이 생성되어 있습니다.
- 적합한 네트워크, DNS 및 로드 밸런싱 인프라가 구성되어 있습니다.
- 적합한 PXE 또는 iPXE 인프라가 구성되어 있습니다.
- 사용자 컴퓨터에서 액세스할 수 있고 사용자가 생성한 시스템에서 액세스할 수 있는 HTTP 서버에 대한 액세스 권한을 확보합니다.
- 네트워킹 및 디스크 파티션과 같은 기능을 구성하는 다양한 방법에 대해서는 *고급 RHCOS 설치 구성*섹션을 살펴보십시오.

프로세스

1. 설치 프로그램에서 생성한 부트스트랩, 컨트롤 플레인, 컴퓨팅 노드 Ignition 구성 파일을 HTTP 서버에 업로드합니다. 해당 파일의 URL을 기록해 둡니다.

중요

HTTP 서버에 저장하기 전에 Ignition 구성에서 구성 설정을 추가하거나 변경할 수 있습니다. 설치를 마친 후 클러스터에 컴퓨팅 시스템을 더 추가하려면 Ignition 구성 파일을 삭제하지 마십시오.

2. 설치 호스트에서 Ignition 구성 파일을 URL에서 사용할 수 있는지 확인합니다. 다음 예에서는 부트스트랩 노드에 대한 Ignition 구성 파일을 가져옵니다.

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

출력 예

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-
rsa...
```

명령에서 **bootstrap.ign**을 **master.ign** 또는 **worker.ign**으로 교체하여 컨트롤 플레인 및 컴퓨팅 노드의 **Ignition** 구성 파일도 사용할 수 있는지 확인합니다.

3. **RHCOS** 커널,**initramfs** 및 **rootfs** 파일을 **RHCOS** 이미지 미러 페이지에서 원하는 운영 체제 인스턴스를 설치하는 데 필요한 경우, **RHCOS** 파일의 올바른 버전을 가져오는 권장 방법은 **openshift-install** 명령 출력에서 얻을 수 있습니다.

```
$ openshift-install coreos print-stream-json | grep -Eo '"https.*(kernel-
|initramfs.|rootfs.)\w+(\.img)?"'
```

출력 예

```
"<url>/art/storage/releases/rhcos-4.9-aarch64/<release>/aarch64/rhcos-<release>-live-
kernel-aarch64"
"<url>/art/storage/releases/rhcos-4.9-aarch64/<release>/aarch64/rhcos-<release>-live-
initramfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.9-aarch64/<release>/aarch64/rhcos-<release>-live-
rootfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.9-ppc64le/49.84.202110081256-0/ppc64le/rhcos-
<release>-live-kernel-ppc64le"
"<url>/art/storage/releases/rhcos-4.9-ppc64le/<release>/ppc64le/rhcos-<release>-live-
initramfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.9-ppc64le/<release>/ppc64le/rhcos-<release>-live-
rootfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.9-s390x/<release>/s390x/rhcos-<release>-live-
kernel-s390x"
"<url>/art/storage/releases/rhcos-4.9-s390x/<release>/s390x/rhcos-<release>-live-
initramfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.9-s390x/<release>/s390x/rhcos-<release>-live-
rootfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.9/<release>/x86_64/rhcos-<release>-live-kernel-
x86_64"
"<url>/art/storage/releases/rhcos-4.9/<release>/x86_64/rhcos-<release>-live-
initramfs.x86_64.img"
"<url>/art/storage/releases/rhcos-4.9/<release>/x86_64/rhcos-<release>-live-
rootfs.x86_64.img"
```

중요

OpenShift Container Platform의 모든 릴리스에서 RHCOS 아티팩트가 변경되지 않을 수도 있습니다. 설치하는 OpenShift Container Platform 버전과 같거나 그 이하의 버전 중 가장 최신 버전의 이미지를 다운로드해야 합니다. 이 프로시저에는 아래 설명된 적절한 kernel, initramfs 및 rootfs 아티팩트만 사용하십시오. 이 설치 유형에서는 RHCOS QCOW2 이미지가 지원되지 않습니다.

OpenShift Container Platform 버전 번호가 파일 이름에 포함됩니다. 다음 예와 유사합니다.

- kernel: rhcos-<version>-live-kernel-<architecture>
- initramfs: rhcos-<version>-live-initramfs.<architecture>.img
- rootfs: rhcos-<version>-live-rootfs.<architecture>.img

4. rootfs, kernel 및 initramfs 파일을 HTTP 서버에 업로드합니다.

중요

설치를 마친 후 클러스터에 컴퓨팅 시스템을 더 추가하려면 Ignition 구성 파일을 삭제하지 마십시오.

5. RHCOS가 설치된 후 시스템이 로컬 디스크에서 부팅되도록 네트워크 부팅 인프라를 구성하십시오.

6. RHCOS 이미지에 대한 PXE 또는 iPXE 설치를 구성하고 설치를 시작하십시오.

사용 환경에 대한 다음 예시 메뉴 항목 중 하나를 수정하고, 이미지 및 Ignition 파일에 적절히 접근할 수 있는지 확인하십시오.

- **PXE**의 경우:

```
DEFAULT pxeboot
TIMEOUT 20
PROMPT 0
LABEL pxeboot
    KERNEL http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> (1)
    APPEND initrd=http://<HTTP_server>/rhcos-<version>-live-initramfs.
<architecture>.img coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-
rootfs.<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign (2) (3)
```

(1) (1)

HTTP 서버에 업로드한 라이브 **kernel** 파일의 위치를 지정합니다. **URL**은 **HTTP**, **TFTP** 또는 **FTP**여야 합니다. **HTTPS**와 **NFS**는 지원되지 않습니다.

(2)

NIC를 여러 개 사용하는 경우 **ip** 옵션에 단일 인터페이스를 지정합니다. 예를 들어, **eno1**라는 **NIC**에서 **DHCP**를 사용하려면 **ip=eno1:dhcp**를 설정하십시오.

(3)

HTTP 서버에 업로드한 **RHCOS** 파일의 위치를 지정합니다. **initrd** 매개변수 값은 **initramfs** 파일의 위치, **coreos.live.rootfs_url** 매개변수 값은 **rootfs** 파일의 위치, **coreos.inst.ignition_url** 매개변수 값은 부트스트랩 **Ignition** 구성 파일의 위치입니다. **APPEND** 줄에 커널 인수를 더 추가하여 네트워킹 또는 기타 부팅 옵션도 구성할 수 있습니다.

참고

이 구성은 그래픽 콘솔이 있는 시스템에서 직렬 콘솔 액세스를 활성화하지 않습니다. 다른 콘솔을 구성하려면 **APPEND** 행에 하나 이상의 **console=** 인수를 추가합니다. 예를 들어 **console=tty0 console=ttyS0**을 추가하여 첫 번째 **PC** 직렬 포트를 기본 콘솔로 설정하고 그래픽 콘솔을 보조 콘솔로 설정합니다. 자세한 내용은 **Red Hat Enterprise Linux**에서 직렬 터미널 및/또는 콘솔 설정 방법을 참조하십시오.

- **iPXE**의 경우 :

```
kernel http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> initrd=main
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign (1) (2)
```

```
initrd --name main http://<HTTP_server>/rhcos-<version>-live-initramfs.
<architecture>.img (3)
boot
```

(1)

HTTP 서버에 업로드한 **RHCOS** 파일의 위치를 지정합니다. **kernel** 매개변수 값은 **kernel** 파일의 위치이고 **initrd=main** 인수는 **UEFI** 시스템에서 부팅하는 데 필요하며 **coreos.live.rootfs_url** 매개 변수 값은 **rootfs** 파일의 위치이며, **coreos.inst.ignition_url** 매개 변수 값은 부트스트랩 **Ignition** 설정 파일의 위치입니다.

(2)

NIC를 여러 개 사용하는 경우 **ip** 옵션에 단일 인터페이스를 지정합니다. 예를 들어, **eno1**라는 **NIC**에서 **DHCP**를 사용하려면 **ip=eno1:dhcp**를 설정하십시오.

(3)

HTTP 서버에 업로드한 **initramfs** 파일의 위치를 지정합니다.

참고

이 구성은 그래픽 콘솔이 있는 시스템에서 직렬 콘솔 액세스를 활성화하지 않습니다. 다른 콘솔을 구성하려면 **kernel** 행에 하나 이상의 **console=** 인수를 추가합니다. 예를 들어 **console=tty0 console=ttyS0**을 추가하여 첫 번째 **PC** 직렬 포트를 기본 콘솔로 설정하고 그래픽 콘솔을 보조 콘솔로 설정합니다. 자세한 내용은 **Red Hat Enterprise Linux에서 직렬 터미널 및/또는 콘솔 설정 방법**을 참조하십시오.

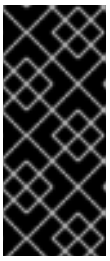
7. 머신 콘솔에서 **RHCOS** 설치 진행률을 모니터링합니다.

중요

OpenShift Container Platform 설치를 시작하기 전에 각 노드에서 성공적으로 설치되었는지 확인합니다. 설치 프로세스를 관찰하면 발생할 수 있는 **RHCOS** 설치 문제의 원인을 파악하는 데 도움이 될 수 있습니다.

8. **RHCOS**가 설치되면 시스템을 재부팅합니다. 시스템이 재부팅되는 동안 지정한 **Ignition** 구성 파일이 적용됩니다.

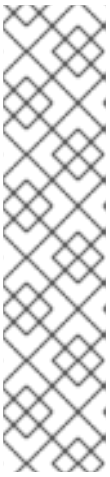
9. 클러스터용 시스템 생성을 계속합니다.



중요

이때 부트스트랩 및 컨트롤 플레인 시스템을 생성해야 합니다. 컨트롤 플레인 시스템이 예약 가능하지 않은 경우 클러스터를 설치하기 전에 두 개 이상의 컴퓨팅 시스템도 생성합니다.

적합한 네트워킹, DNS 및 로드 밸런싱 인프라가 있는 경우 RHCOS 노드가 재부팅된 후 OpenShift Container Platform 부트스트랩 프로세스가 자동으로 시작됩니다.



참고

RHCOS 노드에는 core 사용자의 기본 암호가 포함되지어 있지 않습니다. install_config.yaml 파일에 지정한 공개 키에 대한 액세스 권한이 있는 사용자로 ssh core@<node>.<cluster_name>.<base_domain>을 실행하여 노드에 액세스할 수 있습니다. RHCOS를 실행하는 OpenShift Container Platform 4 클러스터 노드는 변경할 수 없으며 Operator를 통해 클러스터 변경 사항을 적용합니다. SSH를 사용하여 클러스터 노드에 액세스하는 것은 권장되지 않습니다. 그러나 설치 문제를 조사할 때 OpenShift Container Platform API를 사용할 수 없거나 kubelet이 대상 노드에서 제대로 작동하지 않는 경우 디버깅 또는 재해 복구에 SSH 액세스가 필요할 수 있습니다.

8.4.11.3. 고급 RHCOS 설치 구성 옵션

OpenShift Container Platform용 RHCOS (Red Hat Enterprise Linux CoreOS) 노드를 수동으로 프로비저닝하는 주요 이점은 기본 OpenShift Container Platform 설치 방법을 통해 사용할 수 없는 구성을 수행할 수 있는 것입니다. 이 섹션에서는 다음과 같은 방법을 사용하여 수행할 수있는 몇 가지 구성에 대해 설명합니다.

- 라이브 설치 프로그램에 커널 인수 전달
- 라이브 시스템에서 수동으로 coreos-installer 실행
- ISO에 Ignition 구성 포함

이 섹션에 설명되어 있는 수동 Red Hat Enterprise Linux CoreOS(RHCOS) 설치에 대한 고급 구성 항목은 디스크 파티션 설정, 네트워킹 및 다양한 방식의 Ignition 구성 사용과 관련되어 있습니다.

8.4.11.3.1. PXE 및 ISO 설치를 위한 고급 네트워크 옵션 사용

OpenShift Container Platform 노드의 네트워크는 기본적으로 DHCP를 사용하여 필요한 모든 구성 설정을 수집합니다. 고정 IP 주소를 설정하거나 본딩과 같은 특정 설정을 구성하려면 다음 중 하나의 방법으로 수행할 수 있습니다.

- 라이브 설치 프로그램을 시작할 때 특수 커널 매개 변수를 전달합니다.
- 머신 구성을 사용하여 네트워크 파일을 설치된 시스템에 복사합니다.
- 라이브 설치 프로그램 쉘 프롬프트에서 네트워크를 구성한 다음 설치된 시스템에 복사하여 설치한 시스템을 처음 시작할 때 사용하도록합니다.

PXE 또는 iPXE 설치를 구성하려면 다음 옵션 중 하나를 사용합니다.

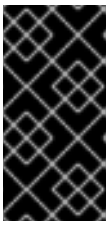
- "고급 RHCOS 설치 참조" 표를 참조하십시오.
- 머신 구성을 사용하여 네트워크 파일을 설치된 시스템에 복사합니다.

다음 프로세스에 따라 ISO 설치를 구성합니다.

프로세스

1. ISO 설치 프로그램을 시작합니다.
2. 라이브 시스템 쉘 프롬프트에서 사용 가능한 RHEL 도구 (예: nmcli 또는 nmtui)를 사용하여 라이브 시스템의 네트워킹를 구성합니다.
3. coreos-installer 명령을 실행하여 시스템을 설치하고 --copy-network 옵션을 추가하여 네트워크 구성을 복사합니다. 예를 들면 다음과 같습니다.

```
$ sudo coreos-installer install --copy-network \
    --ignition-url=http://host/worker.ign /dev/sda
```



중요

copy-network 옵션은 /etc/NetworkManager/system-connections에 있는 네트워킹 구성만 복사합니다. 특히 시스템 호스트 이름을 복사하지 않습니다.

4. 설치된 시스템으로 재부팅하십시오.

추가 리소스

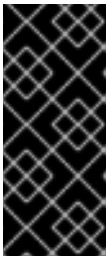
- nmcli 및 nmtui 툴에 대한 자세한 내용은 RHEL 8 설명서에서 nmcli로 시작하기 및 nmtui로 시작하기를 참조하십시오.

8.4.11.3.2. 디스크 파티션 설정

디스크 파티션은 RHCOS(Red Hat Enterprise Linux CoreOS) 설치 중에 OpenShift Container Platform 클러스터 노드에 생성됩니다. 특정 아키텍처의 각 RHCOS 노드는 기본 파티션 구성을 재정의하지 않는 한 동일한 파티션 레이아웃을 사용합니다. RHCOS 설치 중에 대상 장치에서 사용 가능한 나머지 공간을 사용하도록 루트 파일 시스템의 크기가 증가합니다.

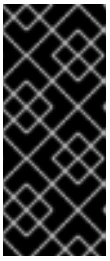
OpenShift Container Platform 클러스터 노드에 RHCOS를 설치할 때 다음과 같은 두 가지 경우 기본 파티셔닝을 재정의할 수 있습니다.

- 별도의 파티션 생성: 빈 디스크에 그린 필드 설치의 경우 파티션에 별도의 스토리지를 추가할 수 있습니다. 이는 /var 또는 /var의 하위 디렉터리 (예: /var/lib/etcd)중 하나를 별도의 파티션으로 만드는 경우에 공식적으로 지원되지만 둘 다 지원되지는 않습니다.



중요

100GB 이상의 디스크 크기 및 특히 1TB보다 큰 디스크 크기의 경우 별도의 /var 파티션을 만듭니다. 자세한 내용은 " 별도의 /var 파티션 생성" 및 이 Red Hat 지식베이스 문서 를 참조하십시오.



중요

Kubernetes는 두 개의 파일 시스템 파티션만 지원합니다. 원래 구성에 둘 이상의 파티션을 추가하는 경우 **Kubernetes**는 모든 파티션을 모니터링할 수 없습니다.

- 기존 파티션 유지: 브라운 필드 설치에서 기존 노드에 **OpenShift Container Platform**을 재설치하고 이전 운영 체제에 설치된 데이터 파티션을 유지해야 하는 경우 기존 데이터 파티션을 유지할 수 있는 **coreos-installer**에 부팅 인수와 옵션이 모두 있습니다.

주의

사용자 지정 파티션을 사용하면 **OpenShift Container Platform**에서 해당 파티션을 모니터링하지 않거나 경고를 받을 수 있습니다. 기본 파티션을 재정의하는 경우 **OpenShift Container Platform**이 호스트 파일 시스템을 모니터링하는 방법에 대한 자세한 내용은 **OpenShift File System Monitoring(제거 조건) 이해**를 참조하십시오.

8.4.11.3.2.1. 별도의 /var 파티션 만들기

일반적으로 **RHCOS** 설치 중에 생성된 기본 디스크 파티션을 사용해야 합니다. 그러나 확장하려는 디렉토리에 별도의 파티션을 생성해야 하는 경우가 있습니다.

OpenShift 컨테이너 플랫폼은 **/var** 디렉토리 또는 **/var**의 하위 디렉터리 중 하나에 스토리지를 연결하는 단일 파티션의 추가를 지원합니다. 예를 들면 다음과 같습니다.

- **/var/lib/containers:** 시스템에 더 많은 이미지와 컨테이너가 추가됨에 따라 확장될 수 있는 컨테이너 관련 콘텐츠를 보관합니다.
- **/var/lib/etcd: etcd** 스토리지의 성능 최적화와 같은 목적으로 별도로 보관할 데이터를 보관합니다.
- **/var:** 감사 등의 목적에 맞게 별도로 분리하여 보관해야 하는 데이터를 보관합니다.

중요

100GB보다 큰 디스크 크기 및 특히 1TB보다 큰 디스크의 경우 별도의 /var 파티션을 만듭니다.

/var 디렉터리의 콘텐츠를 별도로 저장하면 필요에 따라 해당 영역에 대한 스토리지 확장을 보다 용이하게 하고 나중에 OpenShift Container Platform을 다시 설치하여 해당 데이터를 그대로 보존할 수 있습니다. 이 방법을 사용하면 모든 컨테이너를 다시 가져올 필요가 없으며 시스템을 업데이트할 때 대용량 로그 파일을 복사할 필요도 없습니다.

/var 디렉토리 또는 /var의 하위 디렉토리에 대해 별도의 파티션을 사용하면 분할된 디렉토리의 데이터 증가로 루트 파일 시스템이 채워지는 것을 방지할 수 있습니다.

다음 절차에서는 설치 준비 단계에서 노드 유형의 Ignition 구성 파일에 래핑되는 머신 구성 매니페스트를 추가하여 별도의 /var 파티션을 설정합니다.

프로세스

1. 설치 호스트에서 OpenShift Container Platform 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터에 대한 Kubernetes 매니페스트를 생성합니다.

```
$ openshift-install create manifests --dir <installation_directory>
```

2. 추가 파티션을 구성하는 Butane 구성을 생성합니다. 예를 들어 파일 이름을 $HOME/clusterconfig/98-var-partition.bu 로 지정하고, 디스크 장치 이름을 worker 시스템의 스토리지 장치 이름으로 변경하고 스토리지 크기를 적절하게 설정합니다. 이 예에서는 /var 디렉토리를 별도의 파티션에 배치합니다.

```
variant: openshift
version: 4.9.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
  filesystems:
    - device: /dev/disk/by-partlabel/var
```

```
path: /var
format: xfs
mount_options: [defaults, prjquota] 4
with_mount_unit: true
```

1

파티션을 설정해야하는 디스크 저장 장치 이름입니다.

2

데이터 파티션을 부트 디스크에 추가할 때 최소 오프셋 값 **25000** 메비 바이트가 권장됩니다. 루트 파일 시스템은 지정된 오프셋까지 사용 가능한 모든 공간을 채우기 위해 자동으로 크기가 조정됩니다. 오프셋 값이 지정되지 않거나 지정된 값이 권장 최소값보다 작으면 생성되는 루트 파일 시스템의 크기가 너무 작아지고 **RHCOS**를 나중에 다시 설치할 때 데이터 파티션의 첫 번째 부분을 덮어 쓸 수 있습니다.

3

데이터 파티션의 크기(**MB**)입니다.

4

컨테이너 스토리지에 사용되는 파일 시스템에 **prjquota** 마운트 옵션을 활성화해야 합니다.

참고

별도의 **/var** 파티션을 만들 때 다른 인스턴스 유형에 동일한 장치 이름이 없는 경우 컴퓨팅 노드에 다른 인스턴스 유형을 사용할 수 없습니다.

3. **Butane** 구성에서 매니페스트를 생성하여 **clusterconfig/openshift** 디렉터리에 저장합니다. 예를 들어 다음 명령을 실행합니다.

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o
$HOME/clusterconfig/openshift/98-var-partition.yaml
```

4. **Ignition** 구성 파일을 만듭니다.

```
$ openshift-install create ignition-configs --dir <installation_directory> 1
```

1

<installation_directory>는 동일한 설치 디렉터리를 지정합니다.

설치 디렉터리의 부트스트랩, 컨트롤 플레인 및 컴퓨팅 노드에 대한 **Ignition** 구성 파일이 생성됩니다.

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

<installation_directory>/manifest 및 **<installation_directory>/openshift** 디렉터리의 파일은 **98-var-partition** 사용자 정의 **MachineConfig** 오브젝트가 포함된 파일을 포함하여 **Ignition** 구성 파일로 래핑됩니다.

다음 단계

- **RHCOS** 설치 중에 **Ignition** 구성 파일을 참조하여 사용자 정의 디스크 파티션을 적용할 수 있습니다.

8.4.11.3.2.2. 기존 파티션 유지

ISO 설치의 경우 설치 프로그램이 하나 이상의 기존 파티션을 유지하도록하는 옵션을 **coreos-installer** 명령에 추가할 수 있습니다. **PXE** 설치의 경우 **coreos.inst.*** 옵션을 **APPEND** 매개 변수에 추가하여 파티션을 유지할 수 있습니다.

저장된 파티션은 기존 **OpenShift Container Platform** 시스템의 데이터 파티션이 될 수 있습니다. 파티션 레이블 또는 번호 중 하나로 보관하려는 디스크 파티션을 확인할 수 있습니다.

참고

기존 파티션을 저장하고 해당 파티션이 **RHCOS**를 위한 충분한 공간을 남겨 두지 않으면 저장된 파티션이 손상되지는 않지만 설치에 실패합니다.

ISO 설치 중 기존 파티션 유지

이 예에서는 파티션 레이블이 **data (data*)**로 시작하는 모든 파티션을 유지합니다.

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign \
      --save-partlabel 'data*' /dev/sda
```

다음 예는 디스크의 여섯 번째 (6) 파티션을 유지하는 방식으로 **coreos-installer**를 실행하는 방법을 보여줍니다.

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign \
      --save-partindex 6 /dev/sda
```

이 예에서는 파티션 **5** 이상을 유지합니다.

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign
      --save-partindex 5- /dev/sda
```

파티션 저장이 사용된 이전 예에서 **coreos-installer**는 파티션을 즉시 다시 만듭니다.

PXE 설치 중 기존 파티션 유지

이 **APPEND** 옵션은 파티션 레이블이 **'data'('data *')**로 시작하는 모든 파티션을 유지합니다.

```
coreos.inst.save_partlabel=data*
```

이 **APPEND** 옵션은 파티션 **5** 이상을 유지합니다.

```
coreos.inst.save_partindex=5-
```

이 **APPEND** 옵션은 파티션 **6**을 유지합니다.

```
coreos.inst.save_partindex=6
```

8.4.11.3.3. Ignition 설정 확인

RHCOS 베어 메탈 설치를 수행할 때 제공할 수 있는 두 가지 유형의 **Ignition** 구성이 있으며 각 구성을 제공하는 이유도 각각 다릅니다.

-

Permanent install Ignition config: 모든 수동 **RHCOS** 설치는 설치를 수행하기 위해 **openshift-installer**가 생성한 **Ignition** 구성 파일 (예: **bootstrap.ign, master.ign** 및 **worker.ign**) 중 하나를 전달해야 합니다.

중요

이러한 **Ignition** 구성 파일을 직접 수정하지 않는 것이 좋습니다. 이전 섹션의 예에 설명된 대로 **Ignition** 구성 파일로 래핑된 매니페스트 파일을 업데이트할 수 있습니다.

PXE 설치의 경우 **coreos.inst.ignition_url=** 옵션을 사용하여 **APPEND** 행에서 **Ignition** 구성을 전달합니다. **ISO** 설치의 경우 쉘 프롬프트에서 **ISO**를 시작한 후 **coreos-installer** 명령 줄에서 **--ignition-url=** 옵션을 사용하여 **Ignition** 구성을 식별합니다. 두 경우 모두 **HTTP** 및 **HTTPS** 프로토콜만 지원됩니다.

- **Live install Ignition config :**이 유형은 수동으로 작성되어 **Red Hat**에서 지원되지 않으므로 가능하면 사용하지 않도록 해야 합니다. 이 방법을 사용하면 **Ignition** 구성이 라이브 설치 미디어로 전달되고 부팅시 즉시 실행되며 **RHCOS** 시스템이 디스크에 설치되기 전후에 설치 작업을 수행합니다. 이 방법은 시스템 구성을 사용하여 실행할 수 없는 고급 파티션 설정과 같이 한 번만 수행하고 나중에 다시 적용할 필요가 없는 작업의 실행에만 사용해야 합니다.

  PXE 또는 **ISO** 부팅의 경우 **Ignition** 설정을 만들고 **ignition.config.url=** 옵션에 **APPEND**를 실행하여 **Ignition** 설정 위치를 확인할 수 있습니다. 또한 **ignition.firstboot ignition.platform.id = metal**도 추가해야 합니다. 추가하지 않으면 **ignition.config.url** 옵션이 무시됩니다.

8.4.11.3.3.1. RHCOS ISO에 실시간 설치 Ignition 구성 포함

RHCOS ISO 이미지에 직접 라이브 설치 **Ignition** 구성을 포함할 수 있습니다. **ISO** 이미지를 부팅하면 내장된 구성이 자동으로 적용됩니다.

절차

1. 다음 이미지 미러 페이지에서 **coreos-installer** 바이너리를 다운로드합니다. **https://mirror.openshift.com/pub/openshift-v4/clients/coreos-installer/latest/.**

2. **RHCOS ISO** 이미지와 **Ignition** 구성 파일을 검색하고 이를 액세스 가능한 디렉터리 (예: **/mnt)**에 복사합니다.

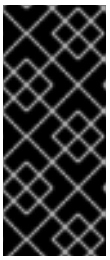
```
# cp rhcos-<version>-live.x86_64.iso bootstrap.ign /mnt/
# chmod 644 /mnt/rhcos-<version>-live.x86_64.iso
```

–

3. 다음 명령을 실행하여 Ignition 구성을 ISO에 포함합니다.

```
# ./coreos-installer iso ignition embed -i /mnt/bootstrap.ign \
    /mnt/rhcos-<version>-live.x86_64.iso
```

이제 해당 ISO를 사용하여 지정된 라이브 설치 Ignition 구성을 사용하여 RHCOS를 설치할 수 있습니다.



중요

coreos-installer iso ignition embed를 사용하여 bootstrap.ign, master.ign 및 worker.ign 과 같이 openshift-installer에 의해 생성된 파일을 포함하는 것은 지원되지 않으며 권장되지 않습니다.

4. 포함된 Ignition 구성 내용을 표시하고 이를 파일로 보내려면 다음을 실행합니다.

```
# ./coreos-installer iso ignition show /mnt/rhcos-<version>-live.x86_64.iso >
mybootstrap.ign
```

```
# diff -s bootstrap.ign mybootstrap.ign
```

출력 예

```
Files bootstrap.ign and mybootstrap.ign are identical
```

5. Ignition 구성을 제거하고 다시 사용할 수 있도록 ISO를 초기 상태로 복원하려면 다음을 실행합니다.

```
# ./coreos-installer iso ignition remove /mnt/rhcos-<version>-live.x86_64.iso
```

이제 다른 Ignition 구성을 ISO에 포함하거나 초기 상태의 ISO를 사용할 수 있습니다.

8.4.11.3.4. 고급 RHCOS 설치 참조

여기서는 **RHCOS(Red Hat Enterprise Linux CoreOS)** 수동 설치 프로세스를 수정하는 데 사용할 수 있는 네트워킹 구성 및 기타 고급 옵션에 대해 설명합니다. 다음 표에서는 **RHCOS** 라이브 설치 프로그램 및 **coreos-installer** 명령과 함께 사용할 수있는 커널 인수 및 명령 줄 옵션에 대해 설명합니다.

8.4.11.3.4.1. ISO 설치를 위한 네트워킹 및 본딩 옵션

ISO 이미지에서 **RHCOS**를 설치하는 경우, 해당 이미지를 부팅할 때 수동으로 커널 인수를 추가하여 노드의 네트워킹을 구성할 수 있습니다. 네트워킹 인수를 지정하지 않으면 **RHCOS**에서 **Ignition** 구성 파일을 가져오는 데 네트워킹이 필요함을 감지하면 **initramfs**에서 **DHCP**가 활성화됩니다.

중요

네트워킹 인수를 수동으로 추가할 때 **initramfs**에서 네트워크를 가져오려면 **rd.neednet=1** 커널 인수도 추가해야 합니다.

다음 표는 **ISO** 설치를 위해 **RHCOS(Red Hat Enterprise Linux CoreOS)** 노드의 네트워킹 및 본딩 구성 예를 보여줍니다. 예제에서는 **ip=, nameserver=, bond=** 커널 인수를 사용하는 방법을 설명합니다.

참고

커널 인수를 추가할 때 순서가 중요합니다: **ip=, nameserver=** 및 **bond=** 입니다.

이는 시스템 부팅 중에 **dracut** 툴로 전달되는 네트워킹 옵션입니다. **dracut**에서 지원하는 네트워킹 옵션에 대한 자세한 내용은 **dracut.cmdline** 메뉴얼 페이지를 참조하십시오.

다음 정보는 **ISO** 설치를 위해 **RHCOS** 노드에서 네트워킹을 구성하는 예제를 제공합니다. 예제에서는 **ip=** 및 **nameserver=** 커널 인수를 사용하는 방법을 설명합니다.

참고

커널 인수를 추가할 때 순서가 중요합니다(**ip =** 및 **nameserver= **).

이는 시스템 부팅 중에 **dracut** 툴로 전달되는 네트워킹 옵션입니다. **dracut**에서 지원하는 네트워킹 옵션에 대한 자세한 내용은 **dracut.cmdline** 메뉴얼 페이지를 참조하십시오.

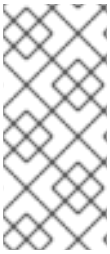
다음 예제는 ISO 설치를 위한 네트워킹 옵션입니다.

DHCP 또는 고정 IP 주소 구성

IP 주소를 구성하려면 DHCP(ip=dhcp)를 사용하거나 개별 고정 IP 주소(ip=<host_ip>)를 설정합니다. 정적 IP를 설정하는 경우 각 노드에서 DNS 서버 IP 주소 (nameserver=<dns_ip>)를 확인합니다. 다음 예제에서는 다음을 설정합니다.

- 노드의 IP 주소는 10.10.10.2로 설정됩니다.
- 게이트웨이 주소는 10.10.10.254로 설정됩니다.
- 넷마스크는 255.255.255.0으로
- core0.example.com에 대한 호스트 이름
- 4.4.4.41의 DNS 서버 주소
- auto-configuration 값을 none으로 설정합니다. IP 네트워킹이 정적으로 구성되면 자동 구성이 필요하지 않습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
nameserver=4.4.4.41
```



참고

DHCP를 사용하여 RHCOS 시스템의 IP 주소 지정을 구성하는 경우 시스템은 DHCP를 통해 DNS 서버 정보도 가져옵니다. DHCP 기반 배포의 경우 DHCP 서버 구성을 통해 RHCOS 노드에서 사용할 DNS 서버 주소를 정의할 수 있습니다.

고정 호스트 이름 없이 IP 주소 구성

정적 호스트 이름을 할당하지 않고 IP 주소를 구성할 수 있습니다. 사용자가 정적 호스트 이름을 설정하지 않으면 역방향 DNS 조회에 의해 자동으로 선택됩니다. 정적 호스트 이름이 없는 IP 주소를 구성하려면 다음 예제를 참조합니다.

- 노드의 **IP** 주소는 **10.10.10.2**로 설정됩니다.
- 게이트웨이 주소는 **10.10.10.254**로 설정됩니다.
- 넷마스크는 **255.255.255.0**으로
- **4.4.4.41**의 **DNS** 서버 주소
- **auto-configuration** 값을 **none**으로 설정합니다. **IP** 네트워킹이 정적으로 구성되면 자동 구성이 필요하지 않습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0::enp1s0:none
nameserver=4.4.4.41
```

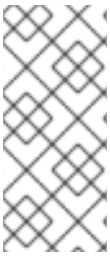
여러 네트워크 인터페이스 지정

여러 **ip=** 항목을 설정하여 여러 네트워크 인터페이스를 지정할 수 있습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

기본 게이트웨이 및 경로 구성

선택 사항: **rd.route=** 값을 설정하여 추가 네트워크에 대한 경로를 구성할 수 있습니다.



참고

하나 이상의 네트워크를 구성할 때 하나의 기본 게이트웨이가 필요합니다. 추가 네트워크 게이트웨이가 기본 네트워크 게이트웨이와 다른 경우 기본 게이트웨이가 기본 네트워크 게이트웨이어야 합니다.

- 다음 명령을 실행하여 기본 게이트웨이를 구성합니다.

```
ip=::10.10.10.254::::
```

- 다음 명령을 입력하여 추가 네트워크의 경로를 구성합니다.

```
rd.route=20.20.20.0/24:20.20.20.254:enp2s0
```

단일 인터페이스에서 DHCP 비활성화

두 개 이상의 네트워크 인터페이스가 있고 하나의 인터페이스만 사용되는 경우와 같이 단일 인터페이스에서 DHCP를 비활성화할 수 있습니다. 이 예에서 enp1s0 인터페이스에는 정적 네트워킹 구성이 있으며 enp2s0 용으로 DHCP가 사용되지 않습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=::::core0.example.com:enp2s0:none
```

DHCP 및 고정 IP 구성 결합

다음과 같이 시스템의 DHCP 및 고정 IP 구성을 여러 네트워크 인터페이스와 결합할 수 있습니다.

```
ip=enp1s0:dhcp
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

개별 인터페이스에서 VLAN 구성

선택 사항: vlan= 매개변수를 사용하여 개별 인터페이스에서 VLAN을 구성할 수 있습니다.

- 네트워크 인터페이스에서 VLAN을 구성하고 고정 IP 주소를 사용하려면 다음 명령을 실행합니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0.100:none
vlan=enp2s0.100:enp2s0
```

- 네트워크 인터페이스에서 VLAN을 구성하고 DHCP를 사용하려면 다음 명령을 실행합니다.

```
ip=enp2s0.100:dhcp
vlan=enp2s0.100:enp2s0
```

여러 DNS 서버 제공

각 서버에 nameserver= 항목을 추가하여 여러 DNS 서버를 제공할 수 있습니다. 예를 들면 다음과 같습니다.

```
nameserver=1.1.1.1
nameserver=8.8.8.8
```

단일 인터페이스에 여러 네트워크 인터페이스 본딩

선택 사항: bond= 옵션을 사용하여 여러 네트워크 인터페이스를 단일 인터페이스에 결합할 수 있습니다. 다음 예제를 참조하십시오.

- 결합된 인터페이스를 구성하는 구문: **bond = name [: network_interfaces] [: options]**

  name은 결합하는 기기 이름(**bond0**)이고 ***network_interfaces***는 쉼표로 구분되는 물리적(이더넷) 인터페이스 목록(**em1, em2**)이며, ***options***은 쉼표로 구분되는 결합 옵션 목록입니다. 사용 가능한 옵션을 보려면 **modinfo bonding**을 입력하십시오.

- **bond=**를 사용하여 결합된 인터페이스를 생성할 때 **IP** 주소가 할당되는 방법과 결합된 인터페이스에 대한 기타 정보를 지정해야 합니다.

- **DHCP**를 사용하도록 결합된 인터페이스를 구성하려면 **bond**의 **IP** 주소를 **dhcp**로 설정하십시오. 예를 들면 다음과 같습니다.

```
bond=bond0:em1,em2:mode=active-backup
ip=bond0:dhcp
```

- 고정 **IP** 주소를 사용하도록 결합된 인터페이스를 구성하려면 원하는 특정 **IP** 주소 및 관련 정보를 입력합니다. 예를 들면 다음과 같습니다.

```
bond=bond0:em1,em2:mode=active-backup
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
```

단일 인터페이스에 여러 네트워크 인터페이스 본딩

선택 사항: **vlan=** 매개변수를 사용하고 **DHCP**를 사용하도록 결합된 인터페이스에서 **VLAN**을 구성할 수 있습니다. 예를 들면 다음과 같습니다.

```
ip=bond0.100:dhcp
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

다음 예제를 사용하여 **VLAN**을 사용하여 결합된 인터페이스를 구성하고 고정 **IP** 주소를 사용합니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0.100:none
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

네트워크 팀 구성 사용

선택 사항: **team=** 매개변수를 사용하여 네트워크 팀을 결합의 대안으로 사용할 수 있습니다.

- 팀 인터페이스를 구성하는 구문은 **team=name[:network_interfaces]**입니다.

  name 은 팀 장치 이름(**team0**)이고 ***network_interfaces*** 는 쉼표로 구분된 실제(이더넷) 인터페이스 목록(**em1, em2**)을 나타냅니다.

RHCOS가 향후 **RHEL** 버전으로 전환되면 티밍이 더 이상 사용되지 않을 예정입니다. 자세한 내용은 이 **Red Hat** 지식베이스 문서를 참조하십시오.

다음 예제를 사용하여 네트워크 팀을 구성합니다.

```
team=team0:em1,em2
ip=team0:dhcp
```

8.4.11.3.4.2. ISO 설치를 위한 coreos-installer 옵션

ISO 이미지에서 **RHCOS** 라이브 환경으로 부팅한 후 명령 프롬프트에서 **coreos-installer install <options> <device>**를 실행하여 **RHCOS**를 설치할 수 있습니다.

다음 표는 **coreos-installer** 명령으로 전달할 수 있는 하위 명령, 옵션 및 인수를 보여줍니다.

표 8.44. coreos-installer 하위 명령, 명령줄 옵션 및 인수

coreos-installer 설치 하위 명령	
하위 명령	*설명*
$ coreos-installer install <options> <device>	ISO 이미지에 Ignition 구성를 삽입합니다.
coreos-installer 설치 하위 명령 옵션	
옵션	*설명*
-u, **--image-url <url>**	이미지 URL을 수동으로 지정합니다.
-f, **--image-file <path>**	로컬 이미지 파일을 수동으로 지정합니다. 디버깅에 사용됩니다.
-i, **--ignition-file <path>**	파일의 Ignition 구성을 삽입합니다.
-I, **--ignition-url <URL>**	URL의 Ignition 구성을 삽입합니다.

--ignition-hash <digest>	Ignition 구성의 **type-value**를 요약합니다.
-p, --platform <name>	설치된 시스템의 Ignition 플랫폼 ID를 재정의합니다.
--append-karg <arg>...	설치된 시스템에 기본 커널 인수를 추가합니다.
--delete-karg <arg>...	설치된 시스템에서 기본 커널 인수를 삭제합니다.
-n, --copy-network	설치 환경의 네트워크 구성을 복사합니다.  **중요** **copy-network** 옵션은 **/etc/NetworkManager/system-connections**에 있는 네트워킹 구성만 복사합니다. 특히 시스템 호스트 이름을 복사하지 않습니다.
--network-dir <path>	**-n**과 함께 사용됩니다. 기본값은 **/etc/NetworkManager/system-connections/**입니다.
--save-partlabel <lx>..	이 레이블 glob로 파티션을 저장합니다.
--save-partindex <id>...	이 번호 또는 범위로 파티션을 저장합니다.
--insecure	서명 확인을 건너뜁니다.
--insecure-ignition	HTTPS 또는 해시 없는 Ignition URL을 허용합니다.
--architecture <name>	대상 CPU 아키텍처입니다. 기본값은 **x86_64**입니다.
--preserve-on-error	오류 발생한 파티션 테이블을 지우지 않습니다.
-h, --help	도움말 정보를 출력합니다.
coreos-install 설치 하위 명령 인수	
인수	***설명***
<device>	대상 장치입니다.
coreos-installer ISO Ignition 하위 명령	
하위 명령	***설명***

$ coreos-installer iso ignition embed <options> --ignition-file <file_path> <ISO_image>	ISO 이미지에 Ignition 구성를 삽입합니다.
coreos-installer iso ignition show <options> <ISO_image>	ISO 이미지에 삽입된 Ignition 구성를 표시합니다.
coreos-installer iso ignition remove <options> <ISO_image>	ISO 이미지에서 삽입된 Ignition 구성를 제거합니다.
coreos-installer ISO Ignition 하위 명령 옵션	
옵션	***설명***
-f, --force	기존 Ignition 구성를 덮어씁니다.
-i, --ignition-file <path>	사용할 Ignition 구성입니다. 기본값은 **stdin**입니다.
-o, --output <path>	새 출력 파일에 ISO를 씁니다.
-h, --help	도움말 정보를 출력합니다.
coreos-installer PXE Ignition 하위 명령	
하위 명령	***설명***
이러한 모든 옵션이 모든 하위 명령에서 허용되지는 않습니다.	
coreos-installer pxe ignition wrap <options>	Ignition 구성을 이미지로 래핑합니다.
coreos-installer pxe ignition unwrap <options> <image_name>	이미지에 래핑된 Ignition 구성를 표시합니다.
coreos-installer PXE Ignition 하위 명령 옵션	
옵션	***설명***
이러한 모든 옵션이 모든 하위 명령에서 허용되지는 않습니다.	
-i, --ignition-file <path>	사용할 Ignition 구성입니다. 기본값은 **stdin**입니다.
-o, --output <path>	새 출력 파일에 ISO를 씁니다.
-h, --help	도움말 정보를 출력합니다.

8.4.11.3.4.3. ISO 또는 PXE 설치를 위한 coreos.inst 부팅 옵션

coreos.inst 부팅 인수를 RHCOS 라이브 설치 프로그램에 전달하여 부팅 시 coreos-installer 옵션을 자동으로 호출할 수 있습니다. 이러한 매개 변수는 표준 부팅 인수 외에 제공됩니다.

- **ISO 설치의 경우 부트 로더 메뉴에서 자동 부팅을 중단하여 coreos.inst 옵션을 추가할 수 있습니다. RHEL CoreOS (Live) 메뉴 옵션이 강조 표시된 상태에서 TAB을 눌러 자동 부팅을 중단할 수 있습니다.**

- **PXE 또는 iPXE 설치의 경우 RHCOS 라이브 설치 프로그램을 부팅하기 전에 coreos.inst 옵션을 APPEND 줄에 추가해야 합니다.**

다음 표는 ISO 및 PXE 설치를 위한 RHCOS 라이브 설치 관리자 coreos.inst 부팅 옵션을 보여줍니다.

표 8.45. coreos.inst 부팅 옵션

인수	설명
coreos.inst.install_dev	필수 항목입니다. 설치할 시스템의 블록 장치입니다. **sda**가 허용되더라도 전체 경로 (예:**/dev/sda**)를 사용하는 것이 좋습니다.
coreos.inst.ignition_url	선택사항: 설치된 시스템에 삽입할 Ignition 구성의 URL입니다. URL을 지정하지 않으면 Ignition 구성이 포함되지 않습니다. HTTP 및 HTTPS 프로토콜만 지원됩니다.
coreos.inst.save_partlabel	선택사항: 설치 중에 보존 할 파티션의 쉼표로 구분된 레이블입니다. Glob 스타일 와일드카드가 허용됩니다. 지정된 파티션이 존재할 필요는 없습니다.
coreos.inst.save_partindex	선택사항: 설치 도중 보존할 파티션 인덱스들입니다(쉼표로 구분됨). **m-n** 범위가 허용되며 **m** 또는 **n**은 생략할 수 있습니다. 지정된 파티션이 존재할 필요는 없습니다.
coreos.inst.insecure	선택사항: **coreos.inst.image_url**로 지정된 OS 이미지의 서명되지 않은 상태를 허용합니다.

인수	설명
coreos.inst.image_url	선택사항: 지정된 RHCOS 이미지를 다운로드하여 설치합니다. ● 이 인수는 프로덕션 환경에서 사용할 수 없으며, 디버깅 용도로만 사용됩니다. ● 이 인수를 사용하면 라이브 미디어와 일치하지 않는 RHCOS 버전을 설치할 수 있지만, 설치하려는 버전과 일치하는 미디어를 사용하는 것이 좋습니다. ● **coreos.inst.image_url**을 사용하는 경우 **coreos.inst.insecure**도 사용해야 합니다. 베어메탈 미디어가 OpenShift Container Platform용으로 GPG 서명되지 않았기 때문입니다. ● HTTP 및 HTTPS 프로토콜만 지원됩니다.
coreos.inst.skip_reboot	선택사항: 설치 후 시스템을 재부팅하지 않습니다. 설치가 완료되면 설치 과정에서 발생되는 상황을 검사할 수 있는 프롬프트가 표시됩니다. 이 인수는 프로덕션 환경에서 사용할 수 없으며, 디버깅 용도로만 사용됩니다.
coreos.inst.platform_id	선택사항: RHCOS 이미지가 설치되고 있는 플랫폼의 Ignition 플랫폼 ID입니다. 기본값은 **metal**입니다. 이 옵션에 따라 VMware와 같은 클라우드 공급자의 Ignition 구성를 요청할지 여부가 결정됩니다. 예: **coreos.inst.platform_id=vmware**.
ignition.config.url	선택사항: 라이브 부팅을 위한 Ignition 구성의 URL입니다. 예를 들어 **coreos-installer**가 호출되는 방식을 사용자 지정하거나 설치 전과 후에 코드를 실행하는 데 사용할 수 있습니다. 이 URL은 설치된 시스템의 Ignition 구성인 **coreos.inst.ignition_url**과 다릅니다.

8.4.11.4. RHCOS에서 커널 인수로 다중 경로 활성화

RHCOS는 이제 기본 디스크에서 멀티패스를 지원하므로 하드웨어 장애에 대한 탄력성이 강화된 호스트 가용성을 높일 수 있습니다.

OpenShift Container Platform 4.8 이상에서 프로비저닝된 노드의 설치 시 멀티패스를 활성화할 수 있습니다. 시스템 구성을 통해 멀티패스를 활성화하면 설치 후 지원을 사용할 수 있지만 설치 중에 멀피태스를 활성화하는 것이 좋습니다.

I/O에서 최적화된 경로로 인해 I/O 시스템 오류가 발생하는 설정에서 설치 시 멀티패스를 활성화해야

합니다.

중요

IBM Z 및 **LinuxONE**에서는 설치 중에 클러스터를 구성하는 경우에만 다중 경로를 활성화할 수 있습니다. 자세한 내용은 ***IBM Z 및 LinuxONE에 z/VM으로 클러스터 설치***의 **"RHCOS** 설치 및 **OpenShift Container Platform** 부트스트랩 프로세스 시작"을 참조하십시오.

다음 절차에서는 설치 시 멀티패스를 활성화하고 커널 인수를 **coreos-installer install** 명령에 추가하여 설치된 시스템 자체에서 첫 번째 부팅부터 시작된 멀티패스를 사용하도록 합니다.

사전 요구 사항

- 버전 **4.8** 이상을 사용하는 실행 중인 **OpenShift Container Platform** 클러스터가 있어야 합니다.

참고

OpenShift Container Platform은 **4.6** 또는 이전 버전에서 업그레이드된 노드에서 **Day-2** 활동으로 다중 경로를 활성화할 수 없습니다.

- 관리 권한이 있는 사용자로 클러스터에 로그인했습니다.

프로세스

1. 멀티패스를 활성화하고 **multipathd** 데몬을 시작하려면 다음 명령을 실행합니다.

```
$ mpathconf --enable && systemctl start multipathd.service
```

- 선택 사항: **PXE** 또는 **ISO**를 부팅하는 경우 커널 명령줄에서 **rd.multipath=default**를 추가하여 멀티패스를 활성화할 수 있습니다.

2. **coreos-installer** 프로그램을 호출하여 커널 인수를 추가합니다.

- 시스템에 연결된 멀티패스 장치가 하나뿐인 경우 경로 **/dev/mapper/mpatha**에서 사용

할 수 있어야 합니다. 예를 들면 다음과 같습니다.

```
$ coreos-installer install /dev/mapper/mpatha \ 1
--append-karg rd.multipath=default \
--append-karg root=/dev/disk/by-label/dm-mpath-root \
--append-karg rw
```

1

단일 멀티패스 장치의 경로를 나타냅니다.

- 시스템에 연결된 멀티패스 장치가 여러 개 있는 경우 보다 명확하게 하려면 /dev/mapper/mpatha를 사용하는 대신 /dev/disk/by-id에서 사용할 수 있는 WWN(World Wide Name) 심볼릭 링크를 사용하는 것이 좋습니다. 예를 들면 다음과 같습니다.

```
$ coreos-installer install /dev/disk/by-id/wwn-<wwn_ID> \ 1
--append-karg rd.multipath=default \
--append-karg root=/dev/disk/by-label/dm-mpath-root \
--append-karg rw
```

1

멀티패스 대상 장치의 WWN ID를 나타냅니다. 예를 들면 0xx194e957fcedb4841입니다.

이 심볼릭 링크는 라이브 설치 프로그램을 지시하기 위해 특수 coreos.inst.** 인수를 사용할 때coreos.inst.install_dev 커널 인수로 사용될 수도 있습니다. 자세한 내용은 "RHCOS 설치 및 OpenShift Container Platform 부트스트랩 프로세스 시작"을 참조하십시오.

3. 작업자 노드 중 하나로 이동하고 커널 명령줄 인수 (호스트의 /proc/cmdline)를 나열하여 커널 인수가 작동하는지 확인합니다.

```
$ oc debug node/ip-10-0-141-105.ec2.internal
```

출력 예

```
Starting pod/ip-10-0-141-105ec2internal-debug ...
To use host binaries, run `chroot /host`

sh-4.2# cat /host/proc/cmdline
```

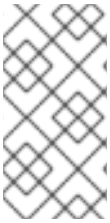
```
...
rd.multipath=default root=/dev/disk/by-label/dm-mpath-root
...

sh-4.2# exit
```

추가된 커널 인수가 표시되어야 합니다.

8.4.11.5. bootupd를 사용하여 부트로더 업데이트

bootupd를 사용하여 부트로더를 업데이트하려면 **RHCOS** 머신에 수동으로 **bootupd**를 설치하거나 **systemd** 유닛이 활성화된 머신 구성을 제공해야 합니다. **grubby** 또는 기타 부트로더 툴과는 달리 **bootupd**는 커널 인수 전달과 같은 커널 공간 구성을 관리하지 않습니다.

bootupd를 설치한 후 **OpenShift Container Platform** 클러스터에서 원격으로 관리할 수 있습니다.



참고

bootupd는 **BootHole** 취약점에 대한 보호와 같이 베어 메탈 또는 가상화 하이퍼바이저 설치에서만 사용하는 것이 좋습니다.

수동 설치 방법

bootctl 명령줄 툴을 사용하여 **bootupd**를 수동으로 설치할 수 있습니다.

1. 시스템 상태를 검사합니다.

```
# bootupctl status
```

출력 예

```
Component EFI
  Installed: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
  Update: At latest version
```

2. **bootupd**를 설치하지 않고 생성된 **RHCOS** 이미지에는 명시적 채택 단계가 필요합니다.

   시스템 상태가 **Adoptable**인 경우 채택을 수행합니다.

```
# bootupctl adopt-and-update
```

   출력 예

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

3. 업데이트를 사용할 수 있는 경우 다음 재부팅에 변경 사항이 적용되도록 업데이트를 적용합니다.

```
# bootupctl update
```

   출력 예

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

시스템 구성 방법

bootupd를 활성화하는 또 다른 방법은 머신 구성을 제공하는 것입니다.

- 다음 예와 같이 **systemd** 단위가 활성화된 머신 구성 파일을 제공합니다.

  출력 예

```
variant: rhcos
version: 1.1.0
systemd:
  units:
    - name: custom-bootupd-auto.service
      enabled: true
      contents: |
        [Unit]
        Description=Bootupd automatic update

        [Service]
        ExecStart=/usr/bin/bootupctl update
        RemainAfterExit=yes

        [Install]
        WantedBy=multi-user.target
```

8.4.12. 부트스트랩 프로세스가 완료될 때까지 대기 중

OpenShift Container Platform 부트스트랩 프로세스는 클러스터 노드가 먼저 디스크에 설치된 영구 RHCOS 환경으로 부팅된 후에 시작됩니다. Ignition 구성 파일을 통해 제공되는 구성 정보는 부트스트랩 프로세스를 초기화하고 머신에 OpenShift Container Platform을 설치하는 데 사용됩니다. 부트스트랩 프로세스가 완료될 때까지 기다려야 합니다.

사전 요구 사항

- 클러스터에 대한 Ignition 구성 파일이 생성되어 있습니다.
- 적합한 네트워크, DNS 및 로드 밸런싱 인프라가 구성되어 있습니다.
- 설치 프로그램을 받아서 클러스터의 Ignition 구성 파일을 생성했습니다.
- 클러스터 머신에 RHCOS를 설치하고 OpenShift Container Platform 설치 프로그램에서 생성된 Ignition 구성 파일을 제공했습니다.

절차

1. 부트스트랩 프로세스를 모니터링합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1

<installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2

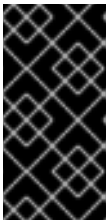
다른 설치 세부 사항을 보려면 info 대신 warn, debug 또는 error를 지정합니다.

출력 예

```
INFO Waiting up to 30m0s for the Kubernetes API at
https://api.test.example.com:6443...
INFO API v1.22.1 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

이 명령은 Kubernetes API 서버가 컨트롤 플레인 시스템에서 부트스트랩되었다는 신호를 보낼 때 성공합니다.

2. 부트스트랩 프로세스가 완료된 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다.



중요

이 시점에 로드 밸런서에서 부트스트랩 시스템을 제거해야 합니다. 부트스트랩 머신 자체를 제거하거나 다시 포맷할 수도 있습니다.

추가 리소스

- 설치 문제가 발생하는 경우 설치 로그 모니터링 및 진단 데이터 검색에 대한 자세한 내용은 설치 진행 상황 모니터링을 참조하십시오.

8.4.13. CLI를 사용하여 클러스터에 로그인

클러스터 kubeconfig 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. kubeconfig 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- oc CLI를 설치했습니다.

절차

1. kubeadmin 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 oc 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

8.4.14. 머신의 인증서 서명 요청 승인

클러스터에 시스템을 추가하면 추가한 시스템별로 보류 중인 인증서 서명 요청(CSR)이 두 개씩 생성됩니다. 이러한 CSR이 승인되었는지 확인해야 하며, 필요한 경우 이를 직접 승인해야 합니다. 클라이언트

요청을 먼저 승인한 다음 서버 요청을 승인해야 합니다.

사전 요구 사항

- 클러스터에 시스템을 추가했습니다.

절차

1. 클러스터가 시스템을 인식하는지 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0  Ready    master  63m  v1.22.1
master-1  Ready    master  63m  v1.22.1
master-2  Ready    master  64m  v1.22.1
```

출력에 생성된 모든 시스템이 나열됩니다.

참고

이전 출력에는 일부 **CSR**이 승인될 때까지 컴퓨팅 노드(작업자 노드라고도 함)가 포함되지 않을 수 있습니다.

2. 보류 중인 **CSR**을 검토하고 클러스터에 추가한 각 시스템에 대해 **Pending** 또는 **Approved** 상태의 클라이언트 및 서버 요청이 표시되는지 확인합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE    REQUESTOR                                                   CONDITION
csr-8b2br   15m    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

예에서는 두 시스템이 클러스터에 참여하고 있습니다. 목록에는 승인된 **CSR**이 더 많이 나타날 수도 있습니다.

3. **CSR**이 승인되지 않은 경우, 추가된 시스템에 대한 모든 보류 중인 **CSR**이 **Pending** 상태로 전환된 후 클러스터 시스템의 **CSR**을 승인합니다.

참고

CSR은 교체 주기가 자동으로 만료되므로 클러스터에 시스템을 추가한 후 **1** 시간 이내에 **CSR**을 승인하십시오. 한 시간 내에 승인하지 않으면 인증서가 교체되고 각 노드에 대해 두 개 이상의 인증서가 표시됩니다. 이러한 인증서를 모두 승인해야 합니다. 클라이언트 **CSR**이 승인되면 **Kubelet**은 인증서에 대한 보조 **CSR**을 생성하므로 수동 승인이 필요합니다. 그러면 **Kubelet**에서 동일한 매개변수를 사용하여 새 인증서를 요청하는 경우 인증서 갱신 요청은 **machine-approver**에 의해 자동으로 승인됩니다.

참고

베어 메탈 및 기타 사용자 프로비저닝 인프라와 같이 머신 **API**를 사용하도록 활성화되지 않는 플랫폼에서 실행되는 클러스터의 경우 **CSR(Kubelet service Certificate Request)**을 자동으로 승인하는 방법을 구현해야 합니다. 요청이 승인되지 않으면 **API** 서버가 **kubelet**에 연결될 때 서비스 인증서가 필요하므로 **oc exec, oc rsh, oc logs** 명령을 성공적으로 수행할 수 없습니다. **Kubelet** 엔드 포인트에 연결하는 모든 작업을 수행하려면 이 인증서 승인이 필요합니다. 이 방법은 새 **CSR**을 감시하고 **CSR**이 **system:node** 또는 **system:admin** 그룹의 **node-bootstrapper** 서비스 계정에 의해 제출되었는지 확인하고 노드의 **ID**를 확인합니다.

- 개별적으로 승인하려면 유효한 **CSR** 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

참고

일부 Operator는 일부 CSR이 승인될 때까지 사용할 수 없습니다.

4. 이제 클라이언트 요청이 승인되었으므로 클러스터에 추가한 각 머신의 서버 요청을 검토해야 합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 나머지 CSR이 승인되지 않고 Pending 상태인 경우 클러스터 머신의 CSR을 승인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 <csr_name>은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs oc adm certificate approve
```

6. 모든 클라이언트 및 서버 CSR이 승인된 후 머신은 Ready 상태가 됩니다. 다음 명령을 실행하여 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0  Ready    master  73m  v1.22.1
master-1  Ready    master  73m  v1.22.1
master-2  Ready    master  74m  v1.22.1
worker-0  Ready    worker  11m  v1.22.1
worker-1  Ready    worker  11m  v1.22.1
```

참고

머신이 Ready 상태로 전환하는 데 서버 CSR의 승인 후 몇 분이 걸릴 수 있습니다.

추가 정보

- CSR에 대한 자세한 내용은 인증서 서명 요청을 참조하십시오.

8.4.15. Operator의 초기 설정

컨트롤 플레인이 초기화된 후 일부 Operator를 즉시 구성하여 모두 사용 가능하도록 해야 합니다.

사전 요구 사항

- 컨트롤 플레인이 초기화되어 있습니다.

프로세스

1. 클러스터 구성 요소가 온라인 상태인지 확인합니다.

```
$ watch -n5 oc get clusteroperators
```

출력 예

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.9.0     True        False         False      19m
baremetal                                  4.9.0     True        False         False      37m
cloud-credential                           4.9.0     True        False         False      40m
cluster-autoscaler                         4.9.0     True        False         False      37m
config-operator                            4.9.0     True        False         False      38m
console                                    4.9.0     True        False         False      26m
csi-snapshot-controller                    4.9.0     True        False         False      37m
dns                                        4.9.0     True        False         False      37m
etcd                                       4.9.0     True        False         False      36m
image-registry                             4.9.0     True        False         False      31m
ingress                                    4.9.0     True        False         False      30m
insights                                   4.9.0     True        False         False      31m
kube-apiserver                             4.9.0     True        False         False      26m
kube-controller-manager                    4.9.0     True        False         False      36m
kube-scheduler                             4.9.0     True        False         False      36m
kube-storage-version-migrator              4.9.0     True        False         False      37m
machine-api                                4.9.0     True        False         False      29m
machine-approver                           4.9.0     True        False         False      37m
machine-config                             4.9.0     True        False         False      36m
marketplace                                4.9.0     True        False         False      37m
monitoring                                 4.9.0     True        False         False      29m
network                                    4.9.0     True        False         False      38m
node-tuning                                4.9.0     True        False         False      37m
openshift-apiserver                        4.9.0     True        False         False      32m
openshift-controller-manager               4.9.0     True        False         False      30m
openshift-samples                          4.9.0     True        False         False      32m
operator-lifecycle-manager                 4.9.0     True        False         False      37m
operator-lifecycle-manager-catalog         4.9.0     True        False         False      37m
operator-lifecycle-manager-packageserver   4.9.0     True        False         False      32m
service-ca                                 4.9.0     True        False         False      38m
storage                                    4.9.0     True        False         False      37m
```

2. 사용할 수 없는 **Operator**를 구성합니다.

추가 리소스

- 실패한 OpenShift Container Platform 설치 시 데이터를 수집하는 방법에 대한 자세한 내용은 설치 실패에서 로그 수집을 참조하십시오.

- 클러스터 전체에서 Operator Pod 상태를 확인하고 진단을 위해 Operator 로그를 수집하는 단계는 Operator 문제 해결을 참조하십시오.

8.4.15.1. 기본 OperatorHub 소스 비활성화

Red Hat 및 커뮤니티 프로젝트에서 제공하는 콘텐츠를 소싱하는 Operator 카탈로그는 OpenShift Container Platform을 설치하는 동안 기본적으로 OperatorHub용으로 구성됩니다. 제한된 네트워크 환경에서는 클러스터 관리자로서 기본 카탈로그를 비활성화해야 합니다.

프로세스

- OperatorHub 오브젝트에 disableAllDefaultSources: true를 추가하여 기본 카탈로그의 소스를 비활성화합니다.

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

작은 정보

또는 웹 콘솔을 사용하여 카탈로그 소스를 관리할 수 있습니다. 관리 → 클러스터 설정 → 구성 → OperatorHub 페이지에서 개별 소스를 생성, 삭제, 비활성화 및 활성화할 수 있는 소스 탭을 클릭합니다.

8.4.15.2. 이미지 레지스트리 스토리지 구성

기본 스토리지를 제공하지 않는 플랫폼에서는 처음에 Image Registry Operator를 사용할 수 없습니다. 설치한 후에 스토리지를 사용하도록 레지스트리를 구성하여 Registry Operator를 사용 가능하도록 만들어야 합니다.

프로덕션 클러스터에 필요한 영구 볼륨을 구성하는 과정의 지침이 표시됩니다. 해당하는 경우, 프로덕션 환경 외 클러스터에서만 사용할 수 있는 저장 위치로서 빈 디렉터리를 구성하는 과정의 지침이 표시됩니다.

업그레이드 중에 Recreate 롤아웃 전략을 사용하여 이미지 레지스트리의 블록 스토리지 유형 사용을 허용하기 위한 추가 지침이 제공됩니다.

8.4.15.2.1. 이미지 레지스트리의 관리 상태 변경

이미지 레지스트리를 시작하려면 **Image Registry Operator** 구성의 **managementState**를 **Removed**에서 **Managed**로 변경해야 합니다.

프로세스

- **managementState Image Registry Operator** 구성을 **Removed**에서 **Managed**로 변경합니다. 예를 들면 다음과 같습니다.

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch
'{"spec":{"managementState":"Managed"}}'
```

8.4.15.2.2. 베어메탈 및 기타 수동 설치를 위한 레지스트리 스토리지 구성

클러스터 관리자는 설치한 후 스토리지를 사용하도록 레지스트리를 구성해야 합니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- 베어 메탈과 같이 수동으로 프로비저닝된 **RHCOS(Red Hat Enterprise Linux CoreOS)** 노드를 사용하는 클러스터가 있어야 합니다.
- **Red Hat OpenShift Container Storage**와 같이 클러스터용 영구 스토리지 프로비저닝이 있습니다.

중요

OpenShift Container Platform은 복제본이 하나만 있는 경우 이미지 레지스트리 스토리지에 대한 **ReadWriteOnce** 액세스를 지원합니다. **ReadWriteOnce** 액세스에서는 레지스트리가 **Recreate** 롤아웃 전략을 사용해야 합니다. 두 개 이상의 복제본으로 고 가용성을 지원하는 이미지 레지스트리를 배포하려면 **ReadWriteMany** 액세스가 필요합니다.

- **"100Gi"** 용량이 필요합니다.

프로세스

1. 스토리지를 사용하도록 레지스트리를 구성하기 위해 **configs.imageregistry/cluster** 리소스에서 **spec.storage.pvc**를 변경합니다.

참고

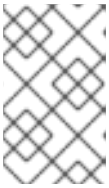
공유 스토리지를 사용할 때 보안 설정을 확인하여 외부에서의 액세스를 방지합니다.

2. 레지스트리 **pod**가 없는지 확인합니다.

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

출력 예

```
No resourses found in openshift-image-registry namespace
```



참고

출력에 레지스트리 **Pod**가 있는 경우 이 절차를 계속할 필요가 없습니다.

3. 레지스트리 구성을 확인합니다.

```
$ oc edit configs.imageregistry.operator.openshift.io
```

출력 예

```
storage:
  pvc:
    claim:
```

image-registry-storage PVC의 자동 생성을 허용하도록 **claim** 필드를 비워 둡니다.

4. **clusteroperator** 상태를 확인합니다.

```
$ oc get clusteroperator image-registry
```

출력 예

```
NAME             VERSION       AVAILABLE   PROGRESSING   DEGRADED   SINCE
MESSAGE
image-registry   4.9           True        False         False      6h50m
```

5. 이미지를 빌드 및 푸시할 수 있도록 레지스트리의 관리가 설정되어 있는지 확인하십시오.

- 다음을 실행합니다.

```
$ oc edit configs.imageregistry/cluster
```

다음으로 라인을 변경하십시오.

```
managementState: Removed
```

다음으로 변경

```
managementState: Managed
```

8.4.15.2.3. 프로덕션 환경 외 클러스터에서 이미지 레지스트리의 스토리지 구성

이미지 레지스트리 **Operator**에 대한 스토리지를 구성해야 합니다. 프로덕션 환경 외 클러스터의 경우, 이미지 레지스트리를 빈 디렉터리로 설정할 수 있습니다. 이렇게 하는 경우 레지스트리를 다시 시작하면 모든 이미지가 손실됩니다.

프로세스

- 이미지 레지스트리 스토리지를 빈 디렉터리로 설정하려면 다음을 수행하십시오.

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch
'{"spec":{"storage":{"emptyDir":{}}}}'
```

주의

프로덕션 환경 외 클러스터에 대해서만 이 옵션을 구성하십시오.

Image Registry Operator가 구성 요소를 초기화하기 전에 이 명령을 실행하면 oc patch 명령이 실패하며 다음 오류가 발생합니다.

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster"
not found
```

몇 분 후에 명령을 다시 실행하십시오.

8.4.15.2.4. 블록 레지스트리 스토리지 구성

클러스터 관리자로서 업그레이드 중에 이미지 레지스트리가 블록 스토리지 유형을 사용할 수 있도록 허용하기 위해 Recreate 롤아웃 전략을 사용할 수 있습니다.

중요

블록 스토리지 볼륨이 지원되지만 프로덕션 클러스터에서 이미지 레지스트리와 함께 사용하는 것은 권장되지 않습니다. 레지스트리가 블록 스토리지에 구성된 설치는 레지스트리가 둘 이상의 복제본을 가질 수 없기 때문에 가용성이 높지 않습니다.

프로세스

1. 이미지 레지스트리 스토리지를 블록 스토리지 유형으로 설정하려면 레지스트리가 Recreate 롤아웃 전략을 사용하고 하나의 (1) 복제본에서만 실행되도록 레지스트리를 패치합니다.

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. 블록 스토리지 장치에 PV를 프로비저닝하고 해당 볼륨의 PVC를 생성합니다. 요청된 블록 볼륨은 RWO(ReadWriteOnce) 액세스 모드를 사용합니다.

3. 올바른 PVC를 참조하도록 레지스트리 구성을 편집합니다.

8.4.16. 사용자 프로비저닝 인프라에 설치 완료

Operator 구성을 완료한 후 제공하는 인프라에 클러스터 설치를 완료할 수 있습니다.

사전 요구 사항

- 컨트롤 플레인이 초기화되어 있습니다.
- 초기 Operator 구성을 완료해야 합니다.

프로세스

1. 다음 명령을 사용하여 모든 클러스터 구성 요소가 온라인 상태인지 확인합니다.

```
$ watch -n5 oc get clusteroperators
```

출력 예

```
NAME                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication             4.9.0     True        False         False      19m
baremetal                  4.9.0     True        False         False      37m
cloud-credential           4.9.0     True        False         False      40m
cluster-autoscaler         4.9.0     True        False         False      37m
config-operator            4.9.0     True        False         False      38m
console                    4.9.0     True        False         False      26m
csi-snapshot-controller    4.9.0     True        False         False      37m
dns                        4.9.0     True        False         False      37m
etcd                       4.9.0     True        False         False      36m
image-registry             4.9.0     True        False         False      31m
ingress                    4.9.0     True        False         False      30m
```

```
insights                                   4.9.0     True        False         False      31m
kube-apiserver                             4.9.0     True        False         False      26m
kube-controller-manager                    4.9.0     True        False         False      36m
kube-scheduler                             4.9.0     True        False         False      36m
kube-storage-version-migrator              4.9.0     True        False         False      37m
machine-api                                4.9.0     True        False         False      29m
machine-approver                           4.9.0     True        False         False      37m
machine-config                             4.9.0     True        False         False      36m
marketplace                                4.9.0     True        False         False      37m
monitoring                                 4.9.0     True        False         False      29m
network                                    4.9.0     True        False         False      38m
node-tuning                                4.9.0     True        False         False      37m
openshift-apiserver                        4.9.0     True        False         False      32m
openshift-controller-manager               4.9.0     True        False         False      30m
openshift-samples                          4.9.0     True        False         False      32m
operator-lifecycle-manager                 4.9.0     True        False         False      37m
operator-lifecycle-manager-catalog         4.9.0     True        False         False      37m
operator-lifecycle-manager-packageserver   4.9.0     True        False         False      32m
service-ca                                 4.9.0     True        False         False      38m
storage                                    4.9.0     True        False         False      37m
```

또는 다음 명령은 모든 클러스터를 사용할 수 있을 때 알립니다. 또한 인증 정보를 검색하고 표시합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1

<installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

출력 예

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator가 Kubernetes API 서버에서 OpenShift Container Platform 클러스터 배포를 완료하면 명령이 성공합니다.

중요

- 설치 프로그램에서 생성하는 **Ignition** 구성 파일에 **24**시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 **24**시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 **kubelet** 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(**CSR**)을 수동으로 승인해야 합니다. 자세한 내용은 ***Recovering from expired control plane certificates*** 문서를 참조하십시오.

- 클러스터를 설치한 후 **24**시간에서 **22**시간까지의 인증서가 교체되기 때문에 생성된 후 **12**시간 이내에 **Ignition** 구성 파일을 사용하는 것이 좋습니다. **12**시간 이내에 **Ignition** 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

2. **Kubernetes API** 서버가 **Pod**와 통신하고 있는지 확인합니다.

   a. 모든 **Pod** 목록을 보려면 다음 명령을 사용하십시오.

   ```
   $ oc get pods --all-namespaces
   ```

   출력 예

   ```
   NAMESPACE                       NAME                                        READY   STATUS
   RESTARTS   AGE
   openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8
   1/1     Running   1          9m
   openshift-apiserver             apiserver-67b9g                             1/1     Running   0
   3m
   openshift-apiserver             apiserver-ljcmx                             1/1     Running   0
   1m
   openshift-apiserver             apiserver-z25h4                             1/1     Running   0
   2m
   openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8
   1/1     Running   0          5m
   ...
   ```

   b. 다음 명령을 사용하여 이전 명령의 출력에 나열된 **Pod**의 로그를 표시합니다.

   -

```
$ oc logs <pod_name> -n <namespace> 1
```

1

이전 명령의 출력에 표시된 대로 **Pod** 이름과 네임스페이스를 지정합니다.

Pod 로그가 표시되면 **Kubernetes API** 서버는 클러스터 시스템과 통신할 수 있습니다.

3. **FCP(Fibre Channel Protocol)**를 사용하는 설치에는 다중 경로를 활성화하기 위해 추가 단계가 필요합니다. 설치 중에 멀티패스를 활성화하지 마십시오.

   자세한 내용은 *설치 후 머신 구성 작업* 설명서에서 **"RHCOS에서 커널 인수를 사용하여 멀티패스 활성화"**를 참조하십시오.

4. 클러스터 등록 페이지에서 클러스터를 등록합니다.

8.4.17. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 **Telemetry** 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 **Telemetry**가 자동으로 실행되고 **OpenShift Cluster Manager** 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 **OpenShift Cluster Manager**를 사용하여 자동으로 또는 **OpenShift Cluster Manager**를 사용하여 수동으로 유지 관리되는지 확인한 후 **subscription watch**를 사용하여 계정 또는 다중 클러스터 수준에서 **OpenShift Container Platform** 서브스크립션을 추적합니다.

추가 리소스

- **Telemetry** 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

8.4.18. 다음 단계

- 설치를 확인합니다.

- 클러스터를 사용자 지정합니다.
- Cluster Samples Operator 및 must-gather 툴의 이미지 스트림을 구성합니다.
- 제한된 네트워크에서 Operator Lifecycle Manager (OLM) 사용 방법에 대해 살펴봅니다.
- 클러스터를 설치하는 데 사용한 미러 레지스트리에 신뢰할 수 있는 CA가 있는 경우 추가 신뢰 저장소를 구성하여 클러스터에 추가합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.

9장. 단일 노드에 설치

9.1. 단일 노드에 설치 준비

9.1.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.

9.1.2. 단일 노드에서 OpenShift 정보

표준 설치 방법으로 단일 노드 클러스터를 생성할 수 있습니다. 단일 노드의 OpenShift Container Platform은 특수 Ignition 구성 ISO를 생성해야 하는 특수 설치입니다. 주요 사용 사례는 기본 스테이션과 가까운 간헐적인 연결, 이동식 클라우드 및 RAN(Radio Access Network)을 비롯한 에지 컴퓨팅 워크로드를 위한 것입니다. 단일 노드에 설치하는 주요 장단점은 고가용성이 없다는 점입니다.

중요

OpenShift는 단일 노드 OpenShift와 함께 OpenShiftSDN을 사용하는 것은 더 이상 사용되지 않습니다. OVN-Kubernetes는 단일 노드 OpenShift 배포를 위한 기본 네트워킹 솔루션입니다.

9.1.3. 단일 노드에 OpenShift를 설치하기 위한 요구사항

단일 노드에 OpenShift Container Platform을 설치하면 고가용성 및 대규모 클러스터에 대한 일부 요구사항이 완화됩니다. 그러나 다음 요구 사항을 해결해야 합니다.

- 관리 호스트: ISO를 준비하고, USB 부팅 드라이브를 만들고 설치를 모니터링할 수 있는 컴퓨터가 있어야 합니다.
- 베어 메탈 설치: 베어 메탈에 단일 노드에 OpenShift Container Platform을 설치하려면 install-config .yaml 구성 파일에서 platform. none: {} 매개변수를 지정해야 합니다.
- 프로덕션급 서버: 단일 노드에 OpenShift Container Platform을 설치하려면 OpenShift Container Platform 서비스 및 프로덕션 워크로드를 실행하기에 충분한 리소스가 있는 서버가 필요합니다.

표 9.1. 최소 리소스 요구사항

Profile	vCPU	메모리	스토리지
최소	8개의 vCPU 코어	32GB RAM	120GB

참고

SMT(동시 멀티스레딩) 또는 하이퍼 스레딩이 활성화되지 않은 경우 하나의 vCPU는 하나의 물리적 코어와 동일합니다. 활성화되면 다음 공식을 사용하여 해당 비율을 계산합니다.

(코어당 스레드)소켓 = vCPU

가상 미디어로 부팅할 때 서버에 BMC(Baseboard Management Controller)가 있어야 합니다.

- **네트워킹: 서버가 인터넷에 액세스할 수 있거나 라우팅 가능한 네트워크에 연결되어 있지 않은 경우 로컬 레지스트리에 액세스할 수 있어야 합니다. 서버에는 Kubernetes API, 인그레스 경로 및 클러스터 노드 도메인 이름의 DHCP 예약 또는 고정 IP 주소가 있어야 합니다. IP 주소를 다음의 FQDN(정규화된 도메인 이름) 각각으로 확인하도록 DNS를 구성해야 합니다.**

표 9.2. 필수 DNS 레코드

사용법	FQDN	설명
Kubernetes API	**api.<cluster_name>.<base_domain>**	DNS A/AAAA 또는 CNAME 레코드를 추가합니다. 이 레코드는 클러스터 외부 클라이언트가 확인할 수 있어야 합니다.
내부 API	**api-int.<cluster_name>.<base_domain>**	ISO를 수동으로 생성할 때 DNS A/AAAA 또는 CNAME 레코드를 추가합니다. 이 레코드는 클러스터 내 노드에서 확인할 수 있어야 합니다.
인그레스 경로	***.apps.<cluster_name>.<base_domain>**	노드를 대상으로 하는 와일드카드 DNS A/AAAA 또는 CNAME 레코드를 추가합니다. 이 레코드는 클러스터 외부 클라이언트가 확인할 수 있어야 합니다.

영구적인 IP 주소가 없으면 apiserver 와 etcd 간의 통신이 실패할 수 있습니다.

9.2. 단일 노드에 OPENSHIFT 설치

9.2.1. Assisted Installer를 사용하여 검색 ISO 생성

단일 노드에 OpenShift Container Platform을 설치하려면 검색 ISO가 필요합니다. 이 ISO는 Assisted Installer(AI)에서 클러스터 이름, 기본 도메인, SSH(Secure Shell) 공개 키 및 풀 시크릿으로 생성할 수 있습니다.

프로세스

1. 관리 노드에서 브라우저를 열고 Assisted Installer를 사용하여 OpenShift 설치로 이동합니다.

2. Create Cluster (클러스터 만들기)를 클릭하여 새 클러스터를 생성합니다.

3. 클러스터 이름 필드에 클러스터 이름을 입력합니다.

4. Base 도메인 필드에 기본 도메인을 입력합니다. 예를 들면 다음과 같습니다.

```
example.com
```

모든 DNS 레코드는 이 기본 도메인의 하위 도메인이어야 하며 클러스터 이름을 포함해야 합니다. 클러스터 설치 후에는 기본 도메인을 변경할 수 없습니다. 예를 들면 다음과 같습니다.

```
<cluster-name>.example.com
```

5. Install single node OpenShift (SNO)(단일 노드 OpenShift(SNO) 설치를 선택합니다.

6. 4.9 릴리스 노트를 읽어 보십시오. OpenShift Container Platform을 단일 노드에 설치하기 위한 몇 가지 제한 사항이 요약되어 있습니다.

7. OpenShift Container Platform 버전을 선택합니다.

8. 선택 사항: 풀 시크릿을 편집합니다.

9. 다음을 클릭합니다.

10. Generate Discovery ISO(검색 ISO 생성)를 클릭합니다.

11. Full image file (전체 이미지 파일)을 선택하여 USB 드라이브 또는 PXE로 부팅합니다. Minimal image file (최소 이미지 파일)을 선택하여 가상 미디어로 부팅합니다.

12. 관리 노드의 SSH 공개 키를 공개 키 필드에 추가합니다.

13. Generate Discovery ISO(검색 ISO 생성)를 클릭합니다.

14. 검색 ISO 다운로드.

15. 가상 미디어로 설치할 검색 ISO URL을 기록합니다.

9.2.2. 수동으로 검색 ISO 생성

단일 노드에 OpenShift Container Platform을 설치하려면 다음 절차에 따라 생성할 수 있는 검색 ISO 가 필요합니다.

프로세스

1. OpenShift Container Platform 클라이언트(oc)를 다운로드하고 다음 명령을 입력하여 사용할 수 있도록 합니다.

```
$ curl -k https://mirror.openshift.com/pub/openshift-v4/clients/ocp/latest/openshift-client-linux.tar.gz > oc.tar.gz
```

```
$ tar zxf oc.tar.gz
```

```
$ chmod +x oc
```

2. OpenShift Container Platform 버전을 설정합니다.

```
$ OCP_VERSION=<ocp_version> 1
```

1 <ocp_version> 을 현재 버전으로 바꿉니다. 예를 들면 다음과 같습니다. latest-4.9

3. OpenShift Container Platform 설치 프로그램을 다운로드하고 다음 명령을 입력하여 사용할 수 있도록 합니다.

```
$ curl -k https://mirror.openshift.com/pub/openshift-v4/clients/ocp/$OCP_VERSION/openshift-install-linux.tar.gz > openshift-install-linux.tar.gz
```

```
$ tar zxvf openshift-install-linux.tar.gz
```

```
$ chmod +x openshift-install
```

4. RHCOS ISO URL을 검색합니다.

```
$ ISO_URL=$(./openshift-install coreos print-stream-json | grep location | grep x86_64 | grep iso | cut -d\" -f4)
```

5. RHCOS ISO를 다운로드합니다.

```
$ curl -L $ISO_URL > rhcos-live.x86_64.iso
```

6. install-config.yaml 파일을 준비합니다.

```
apiVersion: v1
baseDomain: <domain> 1
compute:
- name: worker
  replicas: 0 2
controlPlane:
  name: master
  replicas: 1 3
metadata:
  name: <name> 4
networking:
```

```
  networkType: OVNKubernetes
  clusterNetwork:
  - cidr: <IP_address>/<prefix> 5
    hostPrefix: <prefix> 6
  serviceNetwork:
  - <IP_address>/<prefix> 7
platform:
  none: {}
bootstrapInPlace:
  installationDisk: <path_to_install_drive> 8
pullSecret: '<pull_secret>' 9
sshKey: |
  <ssh_key> 10
```

1

클러스터 도메인 이름을 추가합니다.

2

컴퓨팅 복제본을 **0** 으로 설정합니다. 이렇게 하면 컨트롤 플레인 노드를 예약할 수 있습니다.

3

controlPlane 복제본을 **1** 로 설정합니다. 이 설정은 이전 **compute** 설정과 함께 클러스터가 단일 노드에서 실행되도록 합니다.

4

메타데이터 이름을 클러스터 이름으로 설정합니다.

5

clusterNetwork CIDR을 설정합니다.

6

clusterNetwork 호스트 접두사를 설정합니다. 포드는 이 풀에서 **IP** 주소를 수신합니다.

7

serviceNetwork CIDR을 설정합니다. 서비스는 이 풀에서 **IP** 주소를 수신합니다.

8

설치 디스크 드라이브의 경로를 설정합니다.

9

Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 복사합니다. 1단계에서 Download pull secret(pull secret 다운로드)을 클릭하고 이 구성 설정에 내용을 추가합니다.

10

설치 후 클러스터에 로그인할 수 있도록 관리 호스트에서 공용 SSH 키를 추가합니다.

7. OpenShift Container Platform 자산을 생성합니다.

```
$ mkdir ocp
```

```
$ cp install-config.yaml ocp
```

```
$ ./openshift-install --dir=ocp create single-node-ignition-config
```

8. Ignition 데이터를 RHCOS ISO에 포함합니다.

```
$ alias coreos-installer='podman run --privileged --pull always --rm \
      -v /dev:/dev -v /run/udev:/run/udev -v $PWD:/data \
      -w /data quay.io/coreos/coreos-installer:release'
```

```
$ cp ocp/bootstrap-in-place-for-live-iso.ign iso.ign
```

```
$ coreos-installer iso ignition embed -fi iso.ign rhcos-live.x86_64.iso
```

9.2.3. USB 미디어로 설치

USB 미디어로 설치하려면 관리 노드에서 검색 ISO를 사용하여 부팅 가능한 USB 드라이브를 생성해야 합니다. USB 드라이브로 서버를 부팅하면 단일 노드 설치를 위해 노드를 준비합니다.

프로세스

1. 관리 노드에서 USB 드라이브를 USB 포트에 삽입합니다.

2. 부팅 가능한 USB 드라이브를 생성합니다.

```
# dd if=<path-to-iso> of=<path/to/usb> status=progress
```

-

예를 들면 다음과 같습니다.

```
# dd if=discovery_image_sno.iso of=/dev/sdb status=progress
```

ISO를 USB 드라이브에 복사한 후에는 USB 드라이브를 사용하여 OpenShift Container Platform을 설치할 수 있습니다.

3. 서버에서 USB 드라이브를 USB 포트에 삽입합니다.

4. 서버를 재부팅하고 재부팅 시 BIOS 설정을 입력합니다.

5. 부팅 드라이브 순서를 변경하여 USB 드라이브 부팅을 먼저 만듭니다.

6. BIOS 설정을 저장하고 종료합니다. 서버가 검색 이미지로 부팅됩니다.

9.2.4. Assisted Installer를 사용하여 설치 모니터링

Assisted Installer를 사용하여 ISO를 생성한 경우 다음 절차를 사용하여 설치를 모니터링합니다.

프로세스

1. 관리 호스트에서 브라우저로 돌아가서 페이지를 새로 고칩니다. 필요한 경우 Assisted Installer(설치 프로그램) 페이지를 사용하여 Install OpenShift 를 다시 로드하고 클러스터 이름을 선택합니다.

2. 3단계, 네트워킹 에 도달할 때까지 Next (다음)를 클릭합니다.

3. 사용 가능한 서브넷에서 서브넷을 선택합니다.

4. 계속 선택한 동일한 호스트 검색 SSH 키를 사용합니다. 필요한 경우 SSH 공개 키를 변경할 수 있습니다.

5. **Next (다음)**를 클릭하여 **Review and Create(검토 및 생성)** 단계를 진행합니다.

6. 클러스터 설치를 클릭합니다.

7. 설치 진행 상황을 모니터링합니다. 클러스터 이벤트를 확인합니다. 설치 프로세스가 서버 드라이브에 검색 이미지 쓰기를 완료하면 서버가 다시 시작됩니다. **USB** 드라이브를 제거하고 **BIOS**를 재설정하여 **USB** 드라이브가 아닌 서버의 로컬 미디어로 부팅합니다.

서버가 여러 번 다시 시작하여 컨트롤 플레인을 배포합니다.

9.2.5. 수동으로 설치 모니터링

ISO를 수동으로 만든 경우 이 절차를 사용하여 설치를 모니터링합니다.

프로세스

1. 설치를 모니터링합니다.

```
$ ./openshift-install --dir=ocp wait-for install-complete
```

컨트롤 플레인을 배포하는 동안 서버가 여러 번 재시작됩니다.

2. 선택 사항: 설치가 완료된 후 환경을 확인합니다.

```
$ export KUBECONFIG=ocp/auth/kubeconfig
```

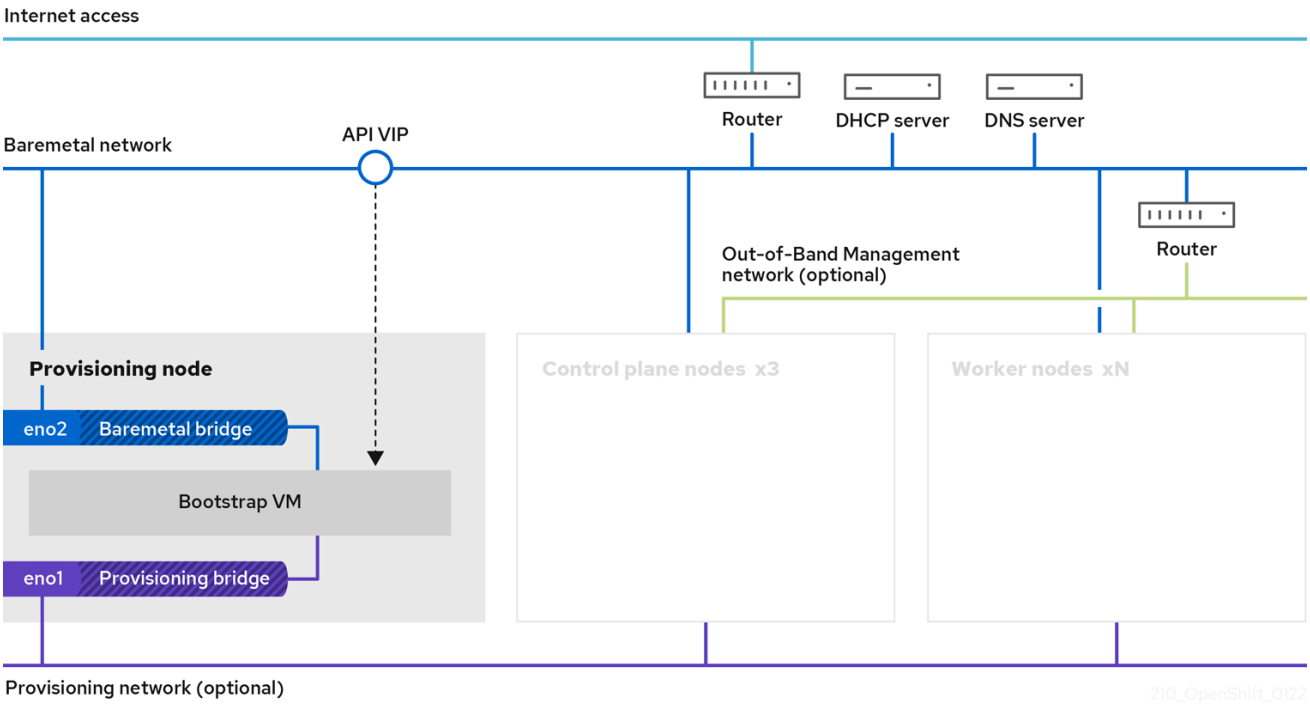
```
$ oc get nodes
```

```
$ oc get clusterversion
```

10장. 베어 메탈에 설치 프로그램이 프로비저닝 한 클러스터 배포

10.1. 개요

베어 메탈 노드에 설치 프로그램이 프로비저닝한 설치는 **OpenShift Container Platform** 클러스터가 실행되는 인프라를 배포하고 구성합니다. 이 가이드에서는 설치 관리자 프로비저닝 베어 메탈 설치를 수행하는 방법을 제공합니다. 다음 다이어그램에서는 배포 **1** 단계에 있는 설치 환경을 보여줍니다.

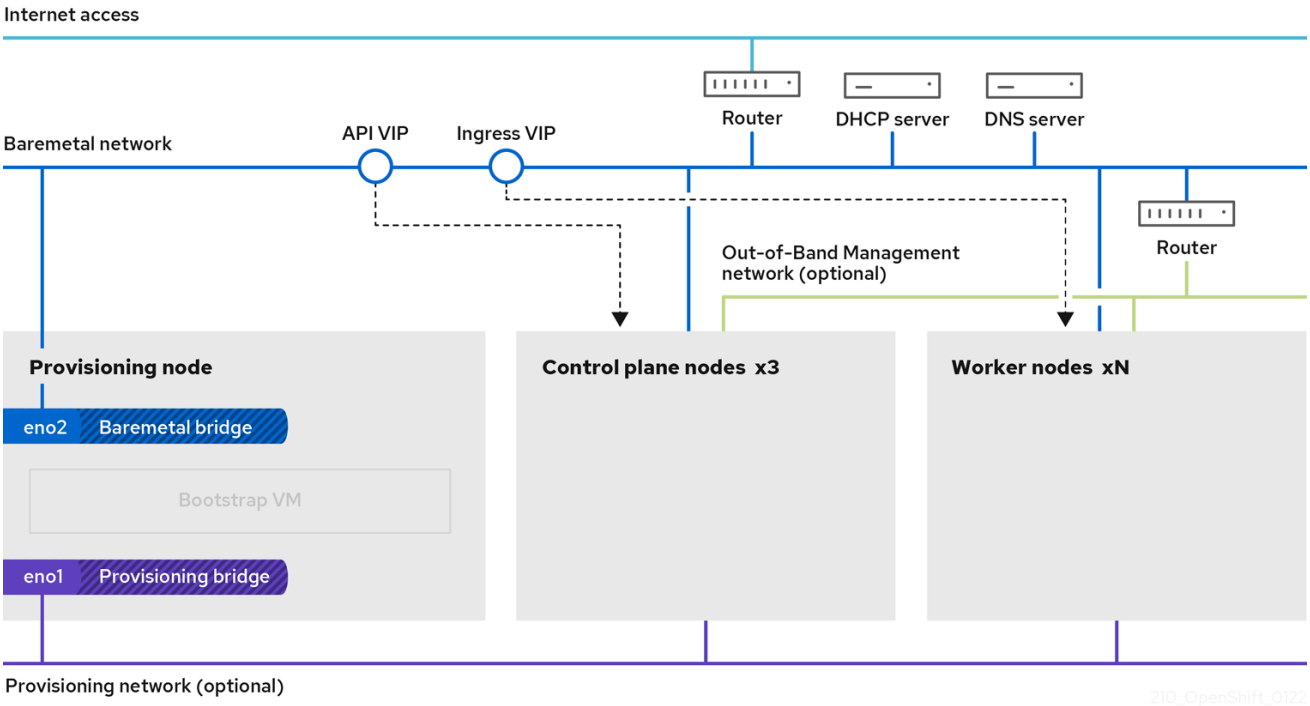


설치 후 프로비저닝 노드를 제거할 수 있습니다.

- **provisioner:** 설치 프로그램을 실행하고 부트스트랩 **VM**을 호스팅하는 물리적 머신으로 새 **OpenShift Container Platform** 클러스터의 컨트롤러를 배포합니다.
- 부트스트랩 **VM: OpenShift Container Platform** 클러스터 배포 프로세스에 사용되는 가상 머신입니다.
- 네트워크 브리지**:** 부트스트랩 **VM**은 네트워크 브리지**, eno1** 및 **eno2** 를 통해 베어 메탈 네트워크 및 **provisioning** 네트워크에 연결됩니다.

배포 **2** 단계에서 프로비저너는 부트스트랩 **VM**을 자동으로 제거하고 **VIP**(가상 **IP** 주소)를 적절한 노드로 이동합니다. **API VIP**는 컨트롤 플레인 노드로 이동하고 **Ingress VIP**는 작업자 노드로 이동합니다.

다음 다이어그램은 배포의 2단계를 보여줍니다.



중요

프로비저닝 네트워크는 선택 사항이지만 **PXE** 부팅에 필요합니다. **provisioning** 네트워크 없이 배포하는 경우 **redfish-virtualmedia** 또는 **idrac-virtualmedia** 와 같은 가상 미디어 **BMC** 주소 지정 옵션을 사용해야 합니다.

10.2. 사전 요구 사항

OpenShift Container Platform 설치 프로그램으로 프로비저닝된 설치에는 다음이 필요합니다.

1. **Red Hat Enterprise Linux (RHEL) 8.x**가 설치된 프로비저너 노드 **1**개 설치 후 프로비저닝 노드를 제거할 수 있습니다.

2. 컨트롤 플레인 노드 **3** 개

3. 각 노드의 베이스 보드 관리 컨트롤러 **(BMC)** 액세스

4. 하나 이상의 네트워크:

   a. 라우팅 가능한 필수 네트워크 1개

   b. 노드 프로비저닝을 위한 선택적 네트워크 1개

   c. 선택적 관리 네트워크 1개

OpenShift Container Platform 설치 프로그램으로 프로비저닝 설치를 시작하기 전에 하드웨어 환경이 다음 요구 사항을 충족하는지 확인합니다.

10.2.1. 노드 요구 사항

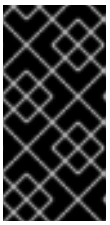
설치 프로그램에서 제공하는 설치에는 여러 하드웨어 노드 요구 사항이 있습니다.

- CPU 아키텍처: 모든 노드는 x86_64 CPU 아키텍처를 사용해야 합니다.
- 유사한 노드: Red Hat은 노드가 역할별로 동일한 구성을 지정할 것을 권장합니다. 즉, Red Hat은 동일한 CPU, 메모리, 스토리지 설정의 브랜드 및 모델의 노드를 사용할 것을 권장하고 있습니다.
- 베이스 보드 관리 컨트롤러 : provisioner 노드는 각 OpenShift Container Platform 클러스터 노드의 베이스 보드 관리 컨트롤러 (BMC)에 액세스할 수 있습니다. IPMI, Redfish 또는 전용 프로토콜을 사용할 수 있습니다.
- 최근 생성: 노드는 최근 생성된 노드여야합니다. 설치 프로그램에서 제공하는 설치는 노드간에 호환되어야 하는 BMC 프로토콜을 사용합니다. 또한 RHEL 8에는 RAID 컨트롤러 용 최신 드라이버가 포함되어 있습니다. 노드가 RHEL 8을 실행하기 위해 provisioner 노드를 지원하고 RHCOS 8을 실행하기 위해 컨트롤 플레인 및 작업자 노드를 지원하기에 충분한 지 확인합니다.
- 레지스트리 노드: (선택 사항) 연결이 끊어진 미러링된 레지스트리를 설정하는 경우 레지스트리가 자체 노드에 상주하는 것이 좋습니다.
- 프로비저너 노드 : 설치 프로그램이 제공하는 설치에는 하나의 provisioner 노드가 필요합니

다.

- 컨트롤 플레인: 설치 프로그램에서 프로비저닝한 설치에는 고가용성을 위해 3 개의 컨트롤 플레인 노드가 필요합니다. 컨트롤 플레인 노드가 3개뿐인 OpenShift Container Platform 클러스터를 배포하여 컨트롤 플레인 노드를 작업자 노드로 예약할 수 있습니다. 소규모 클러스터는 개발, 프로덕션 및 테스트 중에 관리자와 개발자에게 더 많은 리소스를 제공합니다.

- 작업자 노드: 필수는 아니지만 일반적인 프로덕션 클러스터에는 두 개 이상의 작업자 노드가 있습니다.



중요

클러스터는 성능이 저하된 상태에서 라우터 및 인그레스 트래픽과 함께 배포되므로 하나의 작업자 노드만 있는 클러스터를 배포하지 마십시오.

- 네트워크 인터페이스: 각 노드에는 라우팅 가능한 baremetal 네트워크에 대해 하나 이상의 네트워크 인터페이스가 있어야 합니다. 배포에 provisioning 네트워크를 사용할 때 각 노드에는 provisioning 네트워크에 대해 하나의 네트워크 인터페이스가 있어야합니다. provisioning 네트워크를 사용하는 것이 기본 구성입니다.

- UEFI (Unified Extensible Firmware Interface): 설치 프로그램이 프로비저닝한 설치에는 provisioning 네트워크에서 IPv6 주소를 사용하는 경우 모든 OpenShift Container Platform 노드에서 UEFI 부팅이 필요합니다. 또한 provisioning 네트워크 NIC에서 IPv6 프로토콜을 사용하도록 UEFI 장치 PXE 설정을 설정해야하지만 provisioning 네트워크를 생략하면 이 요구 사항이 제거됩니다.

중요

ISO 이미지와 같은 가상 미디어에서 설치를 시작할 때 이전 UEFI 부팅 테이블 항목을 모두 삭제합니다. 부팅 테이블에 펌웨어가 제공한 일반 항목이 아닌 항목이 포함된 경우 설치에 실패할 수 있습니다.

- Secure Boot: Secure Boot가 활성화된 노드를 사용하려면 UEFI 펌웨어 드라이버, EFI 애플리케이션 및 운영 체제와 같은 신뢰할 수 있는 소프트웨어에서만 노드를 부팅해야 합니다. Secure Boot를 사용하여 수동으로 배포하거나 관리할 수 있습니다.

1. 수동형: Secure Boot를 사용하여 OpenShift Container Platform 클러스터를 수동으로 배포하려면 각 컨트롤 플레인 노드와 각 작업자 노드에서 UEFI 부팅 모드 및 Secure Boot를 활성화해야 합니다. Red Hat은 설치 관리자 프로비저닝 설치에서 Redfish 가상 미디어를

사용하는 경우에만 수동으로 활성화된 **UEFI** 및 **Secure Boot**를 사용하여 **Secure Boot**를 지원합니다. 자세한 내용은 "노드 구성" 섹션의 "**Secure Boot**을 위해 수동으로 노드 구성"을 참조하십시오.

2. 관리형: **Secure Boot**를 사용하여 **OpenShift Container Platform** 클러스터를 배포하려면 **install-config.yaml** 파일에서 **bootMode** 값을 **UEFISecureBoot**로 설정해야 합니다. **Red Hat**은 펌웨어 버전 **2.75.75.75** 이상을 실행하는 **10**세대 **HPE** 하드웨어와 **13**세대 **Dell** 하드웨어에 대한 관리형 **Secure Boot**를 사용하는 설치 관리자 프로비저닝 설치를 지원합니다. 관리형 **Secure Boot**를 사용하여 배포하는 경우 **Redfish** 가상 미디어가 필요하지 않습니다. 자세한 내용은 "**OpenShift** 설치를 위한 환경 설정" 섹션의 "관리형 **Secure Boot** 구성"을 참조하십시오.

참고

Red Hat은 자체 생성되는 키가 있는 **Secure Boot**를 지원하지 않습니다.

10.2.2. OpenShift Virtualization을 위한 베어 메탈 클러스터 계획

OpenShift Virtualization을 사용하는 경우 베어 메탈 클러스터를 설치하기 전에 몇 가지 요구 사항을 알고 있어야 합니다.

- 실시간 마이그레이션 기능을 사용하려면 *클러스터 설치 시 작업자 노드가 여러 개* 있어야 합니다. 실시간 마이그레이션에는 클러스터 수준의 **HA**(고가용성) 플래그를 **true**로 설정해야 하기 때문입니다. **HA** 플래그는 클러스터가 설치될 때 설정되며 나중에 변경할 수 없습니다. 클러스터를 설치할 때 정의된 작업자 노드가 두 개 미만인 경우 클러스터 수명에 대해 **HA** 플래그가 **false**로 설정됩니다.

참고

단일 노드 클러스터에 **OpenShift Virtualization**을 설치할 수 있지만 단일 노드 **OpenShift**는 고가용성을 지원하지 않습니다.

- 실시간 마이그레이션에는 공유 스토리지가 필요합니다. **OpenShift Virtualization**의 스토리지에서는 **RWX(ReadWriteMany)** 액세스 모드를 지원하고 사용해야 합니다.

- **SR-IOV(Single Root I/O Virtualization)**를 사용하려는 경우 **OpenShift Container Platform**에서 **NIC**(네트워크 인터페이스 컨트롤러)를 지원하는지 확인합니다.

추가 리소스

- OpenShift Virtualization을 위한 클러스터 준비
- SR-IOV(Single Root I/O Virtualization) 하드웨어 네트워크 정보
- SR-IOV 네트워크에 가상 머신 연결

10.2.3. 가상 미디어를 사용하여 설치를 위한 펌웨어 요구 사항

설치 관리자 프로비저닝 **OpenShift Container Platform** 클러스터의 설치 프로그램은 **Redfish** 가상 미디어와의 하드웨어 및 펌웨어 호환성을 검증합니다. 다음 표에는 **Redfish** 가상 미디어를 사용하여 배포한 설치 관리자 프로비저닝 **OpenShift Container Platform** 클러스터에 대해 테스트 및 인증된 최소 펌웨어 버전이 나열되어 있습니다.

표 10.1. Redfish Virtual Media의 펌웨어 호환성

하드웨어	모델	관리	펌웨어 버전
HP	10세대	iLO5	2.63 이상
Dell	14세대	iDRAC 9	v4.20.20.20 - v4.40.00.00만
	13세대	iDRAC 8	v2.75.75.75 이상

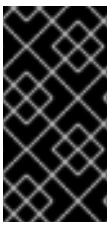
참고

Red Hat은 펌웨어, 하드웨어 또는 기타 타사 구성 요소의 모든 조합을 테스트하지 않습니다. 타사 지원에 대한 자세한 내용은 Red Hat 타사 지원 정책 을 참조하십시오.

펌웨어 업데이트에 대한 정보는 노드의 하드웨어 설명서를 참조하거나 하드웨어 공급업체에 문의하십시오.

HP 서버의 경우 Ironic은 가상 미디어에서 iLO4를 지원하지 않기 때문에 iLO4를 실행하는 9세대 시스템에서 Redfish 가상 미디어가 지원되지 않습니다.

Dell 서버의 경우 OpenShift Container Platform 클러스터 노드에 iDRAC 콘솔을 통해 AutoAttach가 활성화되어 있는지 확인합니다. 메뉴 경로는 구성 → 가상 미디어 → 연결 모드 → 자동 연결입니다 . iDRAC 9 펌웨어 버전 04.40.00.00 을 사용하면 가상 콘솔 플러그인이 기본적으로 eHTML5 로 설정되어 InsertVirtualMedia 워크플로우에 문제가 발생합니다. 이 문제를 방지하려면 플러그인을 HTML5로 설정합니다. 메뉴 경로는 설정 → 가상 콘솔 → 플러그인 유형 → HTML5입니다.

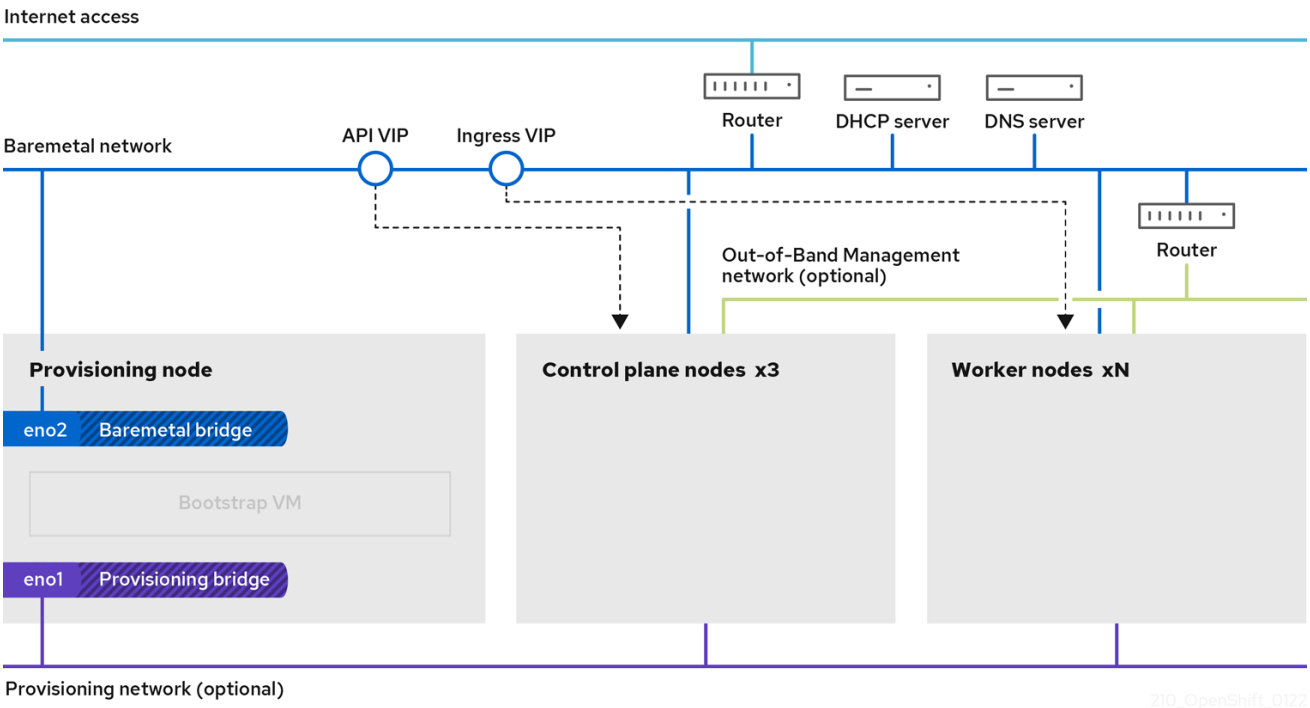


중요

노드 펌웨어가 가상 미디어와 함께 설치하는 버전 아래에 있는 경우 설치 관리자는 노드에 설치를 시작하지 않습니다.

10.2.4. 네트워크 요구 사항

OpenShift Container Platform의 설치 프로그램 프로비저닝 설치에는 몇 가지 네트워크 요구 사항이 있습니다. 먼저 설치 프로그램에서 프로비저닝하는 설치에는 각 베어 메탈 노드에서 운영 체제를 프로비저닝하기위한 라우팅 불가능한 provisioning 네트워크 (선택 사항)가 포함됩니다. 그리고 설치 프로그램에서 프로비저닝하는 설치에는 라우팅 가능한 baremetal 네트워크가 포함됩니다.



10.2.4.1. 네트워크 MTU 증가

OpenShift Container Platform을 배포하기 전에 네트워크 최대 전송 단위(**MTU**)를 **1500** 이상으로 늘립니다. **MTU**가 **1500** 미만이면 노드를 부팅하는 데 사용되는 **Ironic** 이미지가 **Ironic** 검사기 **Pod**와 통신하지 못할 수 있으며 검사가 실패합니다. 이 경우 노드를 설치에 사용할 수 없으므로 설치가 중지됩니다.

10.2.4.2. NIC 설정

OpenShift Container Platform은 다음 두 가지 네트워크를 사용하여 배포합니다.

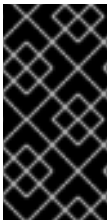
- **provisioning: provisioning** 네트워크는 **OpenShift Container Platform** 클러스터의 일부인 각 노드에서 기본 운영 체제를 프로비저닝하는데 사용되는 선택 옵션인 라우팅할 수 없는 네트워크입니다. 각 클러스터 노드에서 **provisioning** 네트워크의 네트워크 인터페이스에는 **PXE** 부팅으로 구성된 **BIOS** 또는 **UEFI**가 있어야 합니다.

  provisioningNetworkInterface 구성 설정은 컨트롤 플레인 노드에서 프로비저닝 네트워크 **NIC** 이름을 지정합니다. 이 이름은 컨트롤 플레인 노드에서 동일해야 합니다. **bootMACAddress** 구성 설정은 **provisioning** 네트워크의 각 노드에서 특정 **NIC**를 지정하는 방법을 제공합니다 .

  프로비저닝 네트워크는 선택 사항이지만 **PXE** 부팅에 필요합니다. **provisioning** 네트워크 없이 배포하는 경우 **redfish-virtualmedia** 또는 **idrac-virtualmedia** 와 같은 가상 미디어 **BMC** 주소 지정 옵션을 사용해야 합니다.

-

baremetal: baremetal 네트워크는 라우팅 가능한 네트워크입니다. **NIC**가 **provisioning** 네트워크를 사용하도록 구성되지 않은 경우 **baremetal** 네트워크와 인터페이스하는 데 **NIC**를 사용할 수 있습니다 .



중요

VLAN을 사용하는 경우 각 **NIC**는 적절한 네트워크에 해당하는 별도의 **VLAN**에 있어야 합니다.

10.2.4.3. DNS 요구 사항

클라이언트는 **baremetal** 네트워크를 통해 **OpenShift Container Platform** 클러스터 노드에 액세스합니다. 네트워크 관리자는 정식 이름 확장이 클러스터 이름인 하위 도메인 또는 하위 영역을 구성해야 합니다.

```
<cluster_name>.<base_domain>
```

예를 들면 다음과 같습니다.

```
test-cluster.example.com
```

OpenShift Container Platform에는 클러스터 멤버십 정보를 사용하여 **A/AAAA** 레코드를 생성하는 기능이 포함되어 있습니다. 이렇게 하면 노드 이름이 해당 **IP** 주소로 확인됩니다. 노드가 **API**에 등록되면 클러스터에서 **CoreDNS-mDNS**를 사용하지 않고 노드 정보를 분산할 수 있습니다. 그러면 멀티캐스트 **DNS**와 연결된 네트워크 트래픽이 제거됩니다.

OpenShift Container Platform 배포의 경우 다음 구성 요소에 **DNS** 이름을 확인해야 합니다.

- **Kubernetes API**
- **OpenShift Container Platform** 애플리케이션 와일드카드 수신 **API**

A/AAAA 레코드는 이름 확인에 사용되며 **PTR** 레코드는 역방향 이름 확인에 사용됩니다. **RHCOS(Red Hat Enterprise Linux CoreOS)**는 역방향 레코드 또는 **DHCP**를 사용하여 모든 노드의 호스트 이름을 설정합니다.

설치 프로그램에서 제공하는 설치에는 클러스터 멤버십 정보를 사용하여 **A/AAAA** 레코드를 생성하는

기능이 포함되어 있습니다. 이렇게 하면 노드 이름이 해당 IP 주소로 확인됩니다. 각 레코드에서 <cluster_name>은 클러스터 이름이고 <base_domain>은 install-config.yaml 파일에서 지정하는 기반 도메인입니다. 전체 DNS 레코드는 <component>.<cluster_name>.<base_domain> 형식입니다.

표 10.2. 필수 DNS 레코드

구성 요소	레코드	설명
Kubernetes API	**api.<cluster_name>.<base_domain>.**	A/AAAA 레코드와 PTR 레코드는 API 로드 밸런서를 식별합니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.
라우트	***.apps.<cluster_name>.<base_domain>.**	와일드카드 A/AAAA 레코드는 애플리케이션 인그레스 로드 밸런서를 나타냅니다. 애플리케이션 인그레스 로드 밸런서는 Ingress 컨트롤러 Pod를 실행하는 노드를 대상으로 합니다. Ingress 컨트롤러 Pod는 기본적으로 작업자 노드에서 실행됩니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다. 예를 들어 **console-openshift-console.apps.<cluster_name>.<base_domain>**은 OpenShift Container Platform 콘솔의 와일드카드 경로로 사용됩니다.

작은 정보

dig 명령을 사용하여 DNS 확인을 확인할 수 있습니다.

10.2.4.4. DHCP(Dynamic Host Configuration Protocol) 요구 사항

기본적으로 설치 프로그램에서 제공하는 설치는 provisioning 네트워크에 DHCP가 활성화된 ironic-dnsmasq를 배포합니다. provisioningNetwork 구성 설정이 기본값인 managed로 설정된 경우 provisioning 네트워크에서 다른 DHCP 서버가 실행되고 있지 않아야 합니다. provisioning 네트워크에서 실행 중인 DHCP 서버가 있는 경우 install-config.yaml 파일에서 provisioningNetwork 구성 설정을 Unmanaged로 설정해야 합니다.

네트워크 관리자는 외부 DHCP 서버의 baremetal 네트워크에 대해 OpenShift Container Platform 클러스터의 각 노드에 대한 IP 주소를 예약해야 합니다.

10.2.4.5. DHCP 서버를 사용하여 노드의 IP 주소 예약

baremetal 네트워크의 경우 네트워크 관리자는 다음을 포함하여 여러 IP 주소를 예약해야 합니다.

1. 두 개의 고유한 가상 IP 주소

   - API 엔드포인트에 대한 가상 IP 주소 1개
   - 와일드 카드 Ingress 엔드 포인트에 대한 가상 IP 주소 1 개

2. 프로비저너 노드 중 하나의 IP 주소.

3. 각 컨트롤 플레인 (마스터) 노드의 하나의 IP 주소

4. 각 작업자 노드에 대해 하나의 IP 주소 (해당되는 경우)

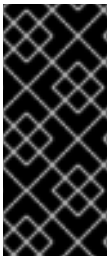
고정 IP 주소가 되도록 IP 주소 예약

일부 관리자는 각 노드의 IP 주소가 DHCP 서버에서 일정하게 유지되도록 고정 IP 주소를 사용하는 것을 선호합니다. OpenShift Container Platform 클러스터에서 고정 IP 주소를 사용하려면 무한 리스로 IP 주소를 예약합니다. 배포 중에 설치 프로그램은 DHCP에서 할당된 주소에서 고정 IP 주소로 NIC를 재구성합니다. 무한이 아닌 DHCP 리스가 있는 NIC는 DHCP를 사용하도록 설정되어 있습니다.

무한 리스로 IP 주소를 설정하는 것은 Machine Config Operator를 사용하여 배포된 네트워크 구성과 호환되지 않습니다.

DHCP 서버가 무한 리스를 제공할 수 있는지 확인

DHCP 서버에서는 rfc2131 에서 지정하는 대로 무한 리스를 올바르게 설정하려면 4294967295초의 DHCP 만료 시간을 제공해야 합니다. DHCP 무한 리스 시간에 대해 더 적은 값을 반환하면 노드는 오류를 보고하고 노드에 대해 영구 IP가 설정되지 않습니다. RHEL 8에서는 dhcpd 가 무한 리스를 제공하지 않습니다. 프로비저너 노드를 사용하여 무한 리스 시간에 동적 IP 주소를 제공하려면 dhcpd 대신 dnsmasq 를 사용합니다.



외부 로드 밸런서와 컨트롤 플레인 노드 간 네트워킹

외부 로드 밸런싱 서비스와 컨트롤 플레인 노드는 동일한 **L2** 네트워크에서 실행해야 하며 **VLAN**을 사용하여 로드 밸런싱 서비스와 컨트롤 플레인 노드 간에 트래픽을 라우팅할 때 동일한 **VLAN**에서 실행해야 합니다.

배포 후 **IP** 주소를 수동으로 변경하지 마십시오.

배포 후 작업자 노드의 **IP** 주소를 수동으로 변경하지 마십시오. 배포 후 작업자 노드의 **IP** 주소를 변경하려면 작업자 노드를 예약 불가로 표시하고 **pod**를 비우고 노드를 삭제하고 새 **IP** 주소로 다시 생성해야 합니다. 자세한 내용은 "노드 작업"을 참조하십시오. 배포 후 컨트롤 플레인 노드의 **IP** 주소를 변경하려면 지원부에 문의하십시오.

스토리지 인터페이스에 **DHCP** 예약이 필요합니다.

다음 표에서는 정규화된 도메인 이름의 구체적 구현을 제공합니다. **API** 및 **Nameserver** 주소는 표준 이름 확장으로 시작됩니다. 컨트롤 플레인 및 작업자 노드의 호스트 이름은 예외이므로 원하는 호스트 이름 지정 규칙을 사용할 수 있습니다.

사용법	호스트 이름	IP
API	**api.<cluster_name>.<base_domain>**	**<ip>**
Ingress LB (apps)	***.apps.<cluster_name>.<base_domain>**	**<ip>**
Provisioner node	**provisioner.<cluster_name>.<base_domain>**	**<ip>**
Master-0	**openshift-master-0.<cluster_name>.<base_domain>**	**<ip>**
Master-1	**openshift-master-1.<cluster_name>-.<base_domain>**	**<ip>**
Master-2	**openshift-master-2.<cluster_name>.<base_domain>**	**<ip>**
Worker-0	**openshift-worker-0.<cluster_name>.<base_domain>**	**<ip>**
Worker-1	**openshift-worker-1.<cluster_name>.<base_domain>**	**<ip>**

사용법	호스트 이름	IP
Worker-n	**openshift-worker-n.<cluster_name>.<base_domain>**	**<ip>**

참고

DHCP 예약을 생성하지 않는 경우 설치 프로그램에는 **Kubernetes API** 노드, 프로비저너 노드, 컨트롤 플레인 노드 및 작업자 노드의 호스트 이름을 설정하기 위해 역방향 **DNS** 확인이 필요합니다.

10.2.4.6. Network Time Protocol (NTP)

클러스터의 각 **OpenShift Container Platform** 노드는 **NTP** 서버에 액세스할 수 있습니다. **OpenShift Container Platform** 노드는 **NTP**를 사용하여 클럭을 동기화합니다. 예를 들어 클러스터 노드는 검증이 필요한 **SSL** 인증서를 사용하므로 노드 간 날짜와 시간이 동기화되지 않은 경우 인증서가 실패할 수 있습니다.

중요

각 클러스터 노드의 **BIOS** 설정에서 일관된 클럭 날짜 및 시간 형식을 정의하지 않으면 설치에 실패할 수 있습니다.

연결이 끊긴 클러스터에서 **NTP** 서버로 작동하도록 컨트롤 플레인 노드를 재구성하고 컨트롤 플레인 노드에서 시간을 검색하도록 작업자 노드를 재구성할 수 있습니다.

10.2.4.7. 상태 중심 네트워크 구성 요구 사항(기술 프리뷰)

OpenShift Container Platform은 **kubernetes-nmstate**를 사용하여 클러스터 노드의 보조 네트워크 인터페이스에 추가 설치 후 상태 기반 네트워크 구성을 지원합니다. 예를 들어 시스템 관리자는 스토리지 네트워크에 설치한 후 클러스터 노드에 보조 네트워크 인터페이스를 구성할 수 있습니다.

참고

pod를 예약하기 전에 설정이 수행되어야 합니다.

상태 중심 네트워크 구성에는 **kubernetes-nmstate**를 설치해야 하며 클러스터 노드에서 실행 중인 **Network Manager**도 필요합니다. 자세한 내용은 **OpenShift Virtualization > Kubernetes NMState**(기술

프리뷰)를 참조하십시오.

10.2.4.8. 대역 외 관리 IP 주소의 포트 액세스

대역 외 관리 IP 주소는 노드와 별도의 네트워크에 있습니다. 대역 외 관리가 설치 중에 프로비저닝 노드와 통신할 수 있도록 대역 외 관리 IP 주소에는 부트스트랩 호스트의 포트 80 및 OpenShift Container Platform 컨트롤 플레인 호스트의 포트 6180 에 대한 액세스 권한이 부여되어야 합니다.

10.2.5. 노드 설정

provisioning 네트워크를 사용할 때 노드 설정

클러스터의 각 노드는 적절한 설치를 위해 다음과 같은 설정이 필요합니다.

주의

노드간에 일치하지 않으면 설치에 실패합니다.

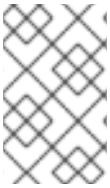
클러스터 노드에는 두 개 이상의 NIC가 포함될 수 있지만 설치 프로세스는 처음 두 개의 NIC에만 중점을 둡니다. 다음 표에서 NIC1은 OpenShift Container Platform 클러스터 설치에만 사용되는 라우팅 불가능한 네트워크(프로비저닝)입니다.

NIC	네트워크	VLAN
NIC1	**provisioning**	**<provisioning_vlan>**
NIC2	**baremetal**	**<baremetal_vlan>**

Provisioner 노드의 **RHEL (Red Hat Enterprise Linux) 8.x** 설치 프로세스는 다를 수 있습니다. 로컬 **Satellite** 서버 **PXE** 서버 **PXE** 지원 **NIC2**를 사용하여 **Red Hat Enterprise Linux (RHEL) 8.x**를 설치하려면 다음과 같이합니다.

PXE	부팅 순서
NIC1 PXE 지원 **provisioning** 네트워크	1

PXE	부팅 순서
NIC2 **baremetal** 네트워크 PXE 사용은 선택 사항입니다.	2



참고

다른 모든 **NIC**에서 **PXE**가 비활성화되어 있는지 확인합니다.

다음과 같이 컨트롤 플레인 및 작업자 노드를 설정합니다.

PXE	부팅 순서
NIC1 PXE 활성화 (프로비저닝 네트워크)	1

provisioning 네트워크없이 노드 설정

설치 프로세스에는 하나의 **NIC**가 필요합니다.

NIC	네트워크	VLAN
NICx	**baremetal**	**<baremetal_vlan>**

NICx는 **OpenShift Container Platform** 클러스터 설치에 사용되는 라우팅 가능한 네트워크 **(baremetal)**이며 인터넷으로 라우팅될 수 있습니다.

중요

프로비저닝 네트워크는 선택 사항이지만 **PXE** 부팅에 필요합니다. **provisioning** 네트워크 없이 배포하는 경우 **redfish-virtualmedia** 또는 **idrac-virtualmedia** 와 같은 가상 미디어 **BMC** 주소 지정 옵션을 사용해야 합니다.

Secure Boot를 위해 노드를 수동으로 설정합니다.

Secure Boot는 **UEFI** 펌웨어 드라이버, **EFI** 애플리케이션 및 운영 체제와 같은 신뢰할 수 있는 소프트웨어만 사용하는지 확인하지 않는 한 노드를 부팅하지 않습니다.

참고

Red Hat은 Redfish 가상 미디어를 사용하여 배포하는 경우에만 수동으로 구성된 Secure Boot를 지원합니다.

Secure Boot를 수동으로 활성화하려면 노드의 하드웨어 가이드를 참조하여 다음을 실행합니다.

프로세스

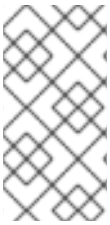
1. 노드를 부팅하고 BIOS 메뉴를 입력합니다.
2. 노드의 부팅 모드를 UEFI Enabled로 설정합니다.
3. Secure Boot를 활성화합니다.

중요

Red Hat은 자체 생성되는 키가 있는 Secure Boot를 지원하지 않습니다.

Fujitsu iRMC의 호환성 지원 모듈 구성

CSM(호환성 지원 모듈) 구성은 UEFI 시스템과의 기존 BIOS 이전 버전과의 호환성을 지원합니다. Fujitsu iRMC를 사용하여 클러스터를 배포할 때 CSM을 구성해야 합니다. 그렇지 않으면 설치에 실패할 수 있습니다.



참고

특정 노드 유형에 대한 CSM 구성에 대한 자세한 내용은 노드의 하드웨어 가이드를 참조하십시오.

사전 요구 사항

- Secure Boot Control을 비활성화했는지 확인합니다. 보안 → Secure Boot Configuration → Secure Boot Control 에서 기능을 비활성화할 수 있습니다.

프로세스

1. 노드를 부팅하고 **BIOS** 메뉴를 선택합니다.

2. 고급 탭의 목록에서 **CSM Configuration** 을 선택합니다.

3. **Launch CSM** 옵션을 활성화하고 다음 값을 설정합니다.

항목	값
부팅 옵션 필터	UEFI 및 레거시
PXE OpROM 정책 시작	UEFI 전용
Storage OpROM 정책 시작	UEFI 전용
기타 PCI 장치om 우선순위	UEFI 전용

10.2.6. 대역 외 관리

일반적으로 노드에는 베이스 보드 관리 컨트롤러 **(BMC)**에서 사용하는 추가 **NIC**가 있습니다. 이러한 **BMC**는 **provisioner** 노드에서 액세스할 수 있어야 합니다.

각 노드는 대역 외 관리를 통해 액세스할 수 있어야합니다. 대역 외 관리 네트워크를 사용하는 경우 **provisioner** 노드는 **OpenShift Container Platform 4**를 성공적으로 설치하기 위해 대역 외 관리 네트워크에 액세스해야 합니다.

대역 외 관리 설정은 이 문서에서 다루지 않습니다. 대역 외 관리를 위해 별도의 관리 네트워크를 설정하는 것이 좋습니다. 그러나 **provisioning** 네트워크 또는 **baremetal** 네트워크를 사용하는 것은 유효한 옵션입니다.

10.2.7. 설치에 필요한 데이터

OpenShift Container Platform 클러스터를 설치하기 전에 모든 클러스터 노드에서 다음 정보를 수집하십시오.

- 대역 외 관리 **IP**
  - 예

- Dell (iDRAC) IP
- HP (iLO) IP
- Fujitsu (iRMC) IP

provisioning 네트워크를 사용하는 경우

- NIC (프로비저닝) MAC 주소
- NIC (baremetal) MAC 주소

provisioning 네트워크를 생략하는 경우

- NIC (baremetal) MAC 주소

10.2.8. 노드의 유효성 검사 체크리스트

provisioning 네트워크를 사용하는 경우

❑ **NIC1 VLAN**은 **provisioning** 네트워크에 대해 설정되어 있습니다.

프로비저닝 네트워크의 **NIC1**은 프로비저너, 컨트롤 플레인 (마스터) 및 작업자 노드에서 **PXE**를 활성화합니다.

❑ **NIC2 VLAN**은 **baremetal** 네트워크에 대해 설정되어 있습니다.

❑ 다른 모든 **NIC**에서 **PXE**는 비활성화되어 있습니다.

DNS는 **API** 및 **Ingress** 끝점으로 구성됩니다.

❑ 컨트롤 플레인 및 작업자 노드가 설정되어 있습니다.

❑ 모든 노드는 대역 외 관리를 통해 액세스할 수 있습니다.

ECDHE (선택 사항) 별도의 관리 네트워크가 생성되었습니다.

❑ 설치에 필요한 데이터.

provisioning 네트워크를 생략하는 경우

NIC1 VLAN은 baremetal 네트워크에 맞게 구성되어 있습니다.

DNS는 API 및 Ingress 끝점으로 구성됩니다.

❑ 컨트롤 플레인 및 작업자 노드가 설정되어 있습니다.

❑ 모든 노드는 대역 외 관리를 통해 액세스할 수 있습니다.

ECDHE (선택 사항) 별도의 관리 네트워크가 생성되었습니다.

❑ 설치에 필요한 데이터.

10.3. OPENSHIFT 설치를 위한 환경 설정

10.3.1. 프로비저너 노드에 RHEL 설치

네트워킹 구성이 완료되면 다음 단계는 프로비저너 노드에 RHEL 8.x를 설치하는 것입니다. 설치 프로그램은 OpenShift Container Platform 클러스터를 설치하는 동안 프로비저너 노드를 오케스트레이터로 사용합니다. 이 문서에서는 프로비저너 노드에 RHEL을 설치하는 것은 다루지 않습니다. 선택 옵션에는 RHEL Satellite 서버, PXE 또는 설치 미디어 사용이 포함되지어 있지만 이에 국한되지는 않습니다.

10.3.2. OpenShift Container Platform 설치를 위한 provisioner 노드 준비

환경을 준비하려면 다음 단계를 수행하십시오.

프로세스

1. ssh를 통해 프로비저너 노드에 로그인합니다.

2. root가 아닌 사용자 (kni)를 만들고 해당 사용자에게 sudo 권한을 부여합니다.

```
# useradd kni
# passwd kni
# echo "kni ALL=(root) NOPASSWD:ALL" | tee -a /etc/sudoers.d/kni
# chmod 0440 /etc/sudoers.d/kni
```

3. 새 사용자에 대한 ssh 키를 만듭니다.

```
# su - kni -c "ssh-keygen -t ed25519 -f /home/kni/.ssh/id_rsa -N ""
```

4. 프로비저너 노드에서 새 사용자로 로그인합니다.

```
# su - kni
$
```

5. Red Hat Subscription Manager를 사용하여 프로비저닝 노드를 등록합니다.

```
$ sudo subscription-manager register --username=<user> --password=<pass> --auto-attach
$ sudo subscription-manager repos --enable=rhel-8-for-x86_64-appstream-rpms --enable=rhel-8-for-x86_64-baseos-rpms
```

참고

Red Hat Subscription Manager에 대한 자세한 내용은 Using and Configuring Red Hat Subscription Manager에서 참조하십시오.

6. 다음 패키지를 설치합니다.

```
$ sudo dnf install -y libvirt qemu-kvm mkisofs python3-devel jq ipmitool
```

7. 사용자를 변경하여 **libvirt** 그룹을 새로 만든 사용자에 추가합니다.

```
$ sudo usermod --append --groups libvirt <user>
```

8. **firewalld**를 다시 시작하고 **http** 서비스를 활성화합니다.

```
$ sudo systemctl start firewalld
$ sudo firewall-cmd --zone=public --add-service=http --permanent
$ sudo firewall-cmd --reload
```

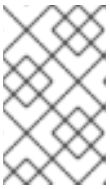
9. **libvirtd** 서비스를 시작하고 활성화합니다.

```
$ sudo systemctl enable libvirtd --now
```

10. **default** 스토리지 풀을 생성하고 시작합니다.

```
$ sudo virsh pool-define-as --name default --type dir --target /var/lib/libvirt/images
$ sudo virsh pool-start default
$ sudo virsh pool-autostart default
```

11. 네트워킹을 설정합니다.



참고

웹 콘솔에서 네트워킹을 구성할 수도 있습니다.

baremetal 네트워크 **NIC** 이름을 내보냅니다.

```
$ export PUB_CONN=<baremetal_nic_name>
```

baremetal 네트워크를 구성합니다.

```
$ sudo nohup bash -c "
    nmcli con down \"$PUB_CONN\"
    nmcli con delete \"$PUB_CONN\"
    # RHEL 8.1 appends the word \"System\" in front of the connection, delete in case it
exists
```

```
  nmcli con down \"System $PUB_CONN\"
  nmcli con delete \"System $PUB_CONN\"
  nmcli connection add ifname baremetal type bridge con-name baremetal bridge.stp
no
  nmcli con add type bridge-slave ifname \"$PUB_CONN\" master baremetal
  pkill dhclient;dhclient baremetal
"
```

provisioning 네트워크를 사용하여 배포하는 경우 **provisioning** 네트워크 **NIC** 이름을 내보냅니다.

```
$ export PROV_CONN=<prov_nic_name>
```

provisioning 네트워크를 사용하여 배포하는 경우 **provisioning** 네트워크를 구성합니다.

```
$ sudo nohup bash -c "
  nmcli con down \"$PROV_CONN\"
  nmcli con delete \"$PROV_CONN\"
  nmcli connection add ifname provisioning type bridge con-name provisioning
  nmcli con add type bridge-slave ifname \"$PROV_CONN\" master provisioning
  nmcli connection modify provisioning ipv6.addresses fd00:1101::1/64 ipv6.method
manual
  nmcli con down provisioning
  nmcli con up provisioning
"
```

참고

이 단계를 실행한 후 **ssh** 연결이 끊어 질 수 있습니다.

baremetal 네트워크를 통해 라우팅 할 수 없는 한 **IPv6** 주소는 모든 주소를 사용할 수 있습니다.

Pv6 주소를 사용하는 경우 **UEFI**가 활성화되고 **UEFI PXE** 설정이 **IPv6** 프로토콜로 설정되어 있는지 확인하십시오.

12. **provisioning** 네트워크 연결에서 **IPv4** 주소를 구성합니다.

```
$ nmcli connection modify provisioning ipv4.addresses 172.22.0.254/24 ipv4.method
manual
```

13. provisioner 노드로 ssh를 실행합니다 (필요한 경우).

```
# ssh kni@provisioner.<cluster-name>.<domain>
```

14. 연결 브리지가 제대로 생성되었는지 확인합니다.

```
$ sudo nmcli con show
```

```
NAME               UUID                                  TYPE      DEVICE
baremetal          4d5133a5-8351-4bb9-bfd4-3af264801530  bridge    baremetal
provisioning       43942805-017f-4d7d-a2c2-7cb3324482ed  bridge    provisioning
virbr0             d9bca40f-eee1-410b-8879-a2d4bb0465e7  bridge    virbr0
bridge-slave-eno1  76a8ed50-c7e5-4999-b4f6-6d9014dd0812  ethernet  eno1
bridge-slave-eno2  f31c3353-54b7-48de-893a-02d2b34c4736  ethernet  eno2
```

15. pull-secret.txt 파일을 만듭니다.

```
$ vim pull-secret.txt
```

웹 브라우저에서 설치 관리자 프로비저닝 인프라가 있는 베어 메탈에 OpenShift 설치 로 이동하고 Downloads 섹션 아래로 스크롤합니다. Copy pull secret을 클릭합니다. pull-secret.txt 파일에 내용을 붙여 넣고 kni 사용자의 홈 디렉터리에 저장합니다.

10.3.3. OpenShift Container Platform 설치 프로그램 검색

설치 프로그램의 stable-4.x 버전을 사용하여 일반적으로 사용 가능한 안정적인 OpenShift Container Platform 버전을 배포합니다.

```
$ export VERSION=stable-4.9
export RELEASE_IMAGE=$(curl -s https://mirror.openshift.com/pub/openshift-v4/clients/ocp/$VERSION/release.txt | grep 'Pull From: quay.io' | awk -F ' ' '{print $3}')
```

10.3.4. OpenShift Container Platform 설치 프로그램 추출

설치 프로그램을 가져온 후 다음 단계로 압축을 풉니다.

프로세스

1. 환경 변수를 설정합니다.

```
$ export cmd=openshift-baremetal-install
$ export pullsecret_file=~/pull-secret.txt
$ export extract_dir=$(pwd)
```

2. **oc** 바이너리를 가져옵니다.

```
$ curl -s https://mirror.openshift.com/pub/openshift-
v4/clients/ocp/$VERSION/openshift-client-linux.tar.gz | tar zxvf - oc
```

3. 설치 프로그램 압축을 풉니다.

```
$ sudo cp oc /usr/local/bin
$ oc adm release extract --registry-config "${pullsecret_file}" --command=$cmd --to
"${extract_dir}" ${RELEASE_IMAGE}
$ sudo cp openshift-baremetal-install /usr/local/bin
```

10.3.5. RHCOS 이미지 캐시 만들기 (선택 사항)

이미지 캐싱을 사용하려면 부트스트랩 VM에서 사용하는 RHCOS (Red Hat Enterprise Linux CoreOS) 이미지와 다른 노드를 프로비저닝하기 위해 설치 프로그램에서 사용하는 RHCOS 이미지의 두 가지 이미지를 다운로드해야 합니다. 이미지 캐싱은 선택 사항이지만 대역폭이 제한된 네트워크에서 설치 프로그램을 실행할 때 특히 유용합니다.

대역폭이 제한된 네트워크에서 설치 프로그램을 실행 중이고 RHCOS 이미지 다운로드에 15-20 분 이상 걸리는 경우 설치 프로그램이 시간 초과됩니다. 이러한 경우 웹 서버에서 이미지를 캐시할 수 있습니다.

이미지가 포함된 컨테이너를 설치합니다.

프로세스

1. podman을 설치합니다.

```
$ sudo dnf install -y podman
```

2. RHCOS 이미지 캐싱에 사용할 방화벽 포트 8080 을 엽니다.

```
$ sudo firewall-cmd --add-port=8080/tcp --zone=public --permanent
```

```
$ sudo firewall-cmd --reload
```

3. bootstraposimage 및 clusterosimage를 저장할 디렉터리를 만듭니다.

```
$ mkdir /home/kni/rhcos_image_cache
```

4. 새로 생성된 디렉터리에 적절한 SELinux 컨텍스트를 설정합니다.

```
$ sudo semanage fcontext -a -t httpd_sys_content_t
"/home/kni/rhcos_image_cache(/.*)?"
```

```
$ sudo restorecon -Rv /home/kni/rhcos_image_cache/
```

5. 설치 프로그램이 부트스트랩 VM에 배포할 RHCOS 이미지의 URI를 가져옵니다.

```
$ export RHCOS_QEMU_URI=$(/usr/local/bin/openshift-baremetal-install coreos print-
stream-json | jq -r --arg ARCH "$(arch)"
'.architectures[$ARCH].artifacts.qemu.formats["qcow2.gz"].disk.location')
```

6. 설치 프로그램이 부트스트랩 VM에 배포할 이미지의 이름을 가져옵니다.

```
$ export export RHCOS_QEMU_NAME=${RHCOS_QEMU_URI##*/}
```

7. 노드에 배포되는 RHCOS 이미지의 SHA 해시를 가져옵니다.

```
$ export RHCOS_QEMU_UNCOMPRESSED_SHA256=$(/usr/local/bin/openshift-
baremetal-install coreos print-stream-json | jq -r --arg ARCH "$(arch)"
'.architectures[$ARCH].artifacts.qemu.formats["qcow2.gz"].disk["uncompressed-
sha256"]')
```

8. 설치 프로그램이 클러스터 노드에 배포할 이미지의 URI를 가져옵니다.

```
$ export RHCOS_OPENSTACK_URI=$(/usr/local/bin/openshift-baremetal-install coreos
print-stream-json | jq -r --arg ARCH "$(arch)"
'.architectures[$ARCH].artifacts.openstack.formats["qcow2.gz"].disk.location')
```

9. 설치 프로그램이 클러스터 노드에 배포할 이미지의 이름을 가져옵니다.

```
$ export RHCOS_OPENSTACK_NAME=${RHCOS_OPENSTACK_URI##*/}
```

10. 설치 프로그램이 클러스터 노드에 배포할 이미지의 **SHA** 해시를 가져옵니다.

```
$ export RHCOS_OPENSTACK_UNCOMPRESSED_SHA256=$(/usr/local/bin/openshift-
baremetal-install coreos print-stream-json | jq -r --arg ARCH "$(arch)"
'.architectures[$ARCH].artifacts.openstack.formats["qcow2.gz"].disk["uncompressed-
sha256"]')
```

11. 이미지를 다운로드하여 **/home/kni/rhcos_image_cache** 디렉터리에 저장합니다.

```
$ curl -L ${RHCOS_QEMU_URI} -o
/home/kni/rhcos_image_cache/${RHCOS_QEMU_NAME}
```

```
$ curl -L ${RHCOS_OPENSTACK_URI} -o
/home/kni/rhcos_image_cache/${RHCOS_OPENSTACK_NAME}
```

12. **SELinux** 유형이 새로 생성된 파일의 **httpd_sys_content_t**임을 확인합니다.

```
$ ls -Z /home/kni/rhcos_image_cache
```

13. **Pod**를 생성합니다.

```
$ podman run -d --name rhcos_image_cache \
-v /home/kni/rhcos_image_cache:/var/www/html \
-p 8080:8080/tcp \
quay.io/centos7/httpd-24-centos7:latest
```

위의 명령은 배포용 이미지를 제공하는 **rhcos_image_cache**라는 이름의 캐싱 웹 서버를 생성합니다. 첫 번째 이미지 **${RHCOS_PATH}${RHCOS_QEMU_URI}?sha256=${RHCOS_QEMU_SHA_UNCOMPRESSED}**는 **bootstrapOSImage**이고 두 번째 이미지 **${RHCOS_PATH}${RHCOS_OP ENSTACK_URI}?sha256=${RHCOS_OPENSTACK_SHA_COMPRESSED}**는 **install-config.yaml** 파일의 **clusterOSImage**입니다.

14. **bootstrapOSImage** 및 **clusterOSImage** 구성을 만듭니다.

```
$ export BAREMETAL_IP=$(ip addr show dev baremetal | awk '/inet /{print $2}' | cut -
d"/" -f1)
```

```
$ export
BOOTSTRAP_OS_IMAGE="http://${BAREMETAL_IP}:8080/${RHCOS_QEMU_NAME}?
sha256=${RHCOS_QEMU_UNCOMPRESSED_SHA256}"
```

```
$ export
CLUSTER_OS_IMAGE="http://${BAREMETAL_IP}:8080/${RHCOS_OPENSTACK_NAME
}?sha256=${RHCOS_OPENSTACK_UNCOMPRESSED_SHA256}"
```

```
$ echo "    bootstrapOSImage=${BOOTSTRAP_OS_IMAGE}"
```

```
$ echo "    clusterOSImage=${CLUSTER_OS_IMAGE}"
```

15. platform.baremetal 아래의 install-config.yaml 파일에 필요한 구성을 추가합니다.

```
platform:
  baremetal:
    bootstrapOSImage: <bootstrap_os_image> 1
    clusterOSImage: <cluster_os_image> 2
```

1 & lt;bootstrap_os_image >를 $BOOTSTRAP_OS_IMAGE 값으로 바꿉니다.

2 & lt;cluster_os_image& gt;를 $CLUSTER_OS_IMAGE 값으로 바꿉니다.

자세한 내용은 "구성 파일" 섹션을 참조하십시오.

10.3.6. 구성 파일

10.3.6.1. install-config.yaml 파일을 생성합니다.

install-config.yaml 파일에는 몇 가지 추가 정보가 필요합니다. 대부분의 정보는 설치된 클러스터가 사용 가능한 하드웨어를 완벽하게 관리하는 데 필요한 정보를 충분히 갖도록 설치 프로세스를 안내하는 데 사용됩니다.

1. install-config.yaml을 설정합니다. pullSecret 및 sshKey 등 환경에 맞게 적절한 변수를 변경합니다.

```
apiVersion: v1
baseDomain: <domain>
```

```
metadata:
  name: <cluster-name>
networking:
  machineNetwork:
  - cidr: <public-cidr>
  networkType: OVNKubernetes
compute:
- name: worker
  replicas: 2 (1)
controlPlane:
  name: master
  replicas: 3
  platform:
    baremetal: {}
platform:
  baremetal:
    apiVIP: <api-ip>
    ingressVIP: <wildcard-ip>
    provisioningNetworkCIDR: <CIDR>
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: ipmi://<out-of-band-ip> (2)
          username: <user>
          password: <password>
        bootMACAddress: <NIC1-mac-address>
        rootDeviceHints:
         deviceName: "/dev/disk/by-id/<disk_id>" (3)
      - name: <openshift-master-1>
        role: master
        bmc:
          address: ipmi://<out-of-band-ip> (4)
          username: <user>
          password: <password>
        bootMACAddress: <NIC1-mac-address>
        rootDeviceHints:
         deviceName: "/dev/disk/by-id/<disk_id>" (5)
      - name: <openshift-master-2>
        role: master
        bmc:
          address: ipmi://<out-of-band-ip> (6)
          username: <user>
          password: <password>
        bootMACAddress: <NIC1-mac-address>
        rootDeviceHints:
         deviceName: "/dev/disk/by-id/<disk_id>" (7)
      - name: <openshift-worker-0>
        role: worker
        bmc:
          address: ipmi://<out-of-band-ip> (8)
          username: <user>
          password: <password>
        bootMACAddress: <NIC1-mac-address>
      - name: <openshift-worker-1>
```

```
      role: worker
      bmc:
        address: ipmi://<out-of-band-ip>
        username: <user>
        password: <password>
      bootMACAddress: <NIC1-mac-address>
      rootDeviceHints:
       deviceName: "/dev/disk/by-id/<disk_id>" 9
pullSecret: '<pull_secret>'
sshKey: '<ssh_pub_key>'
```

1

OpenShift Container Platform 클러스터의 일부인 작업자 노드 수를 기준으로 작업자 머신을 스케일링합니다. replicas 값의 유효한 옵션은 0 과 2 보다 크거나 같은 정수입니다. 세 개의 컨트롤 플레인 시스템만 포함된 3 노드 클러스터를 배포하려면 복제본 수를 0 으로 설정합니다. 3-노드 클러스터는 테스트, 개발 및 프로덕션에 사용할 수 있는 더 작고 리소스 효율적인 클러스터입니다. 작업자가 하나만 있는 클러스터를 설치할 수 없습니다.

2 4 6 8

자세한 옵션은 BMC 주소 지정 섹션을 참조하십시오.

3 5 7 9

설치 디스크 드라이브의 경로를 설정합니다(예: /dev/disk/by-id/wwn-0x64cd98f04fde100024684cf3034da5c2).

2. 클러스터 설정을 저장할 디렉터리를 만듭니다.

```
$ mkdir ~/clusterconfigs
$ cp install-config.yaml ~/clusterconfigs
```

3. OpenShift Container Platform 클러스터를 설치하기 전에 모든 베어 메탈 노드의 전원이 꺼져 있는지 확인합니다.

```
$ ipmitool -I lanplus -U <user> -P <password> -H <management-server-ip> power off
```

4. 이전 배포 시도에서 이전 부트스트랩 리소스가 남아 있는 경우 이전 부트스트랩 리소스를 삭제합니다.

```
for i in $(sudo virsh list | tail -n +3 | grep bootstrap | awk {'print $2'});
do
  sudo virsh destroy $i;
  sudo virsh undefine $i;
```

```
  sudo virsh vol-delete $i --pool $i;
  sudo virsh vol-delete $i.ign --pool $i;
  sudo virsh pool-destroy $i;
  sudo virsh pool-undefine $i;
done
```

10.3.6.2. install-config.yaml 파일 내에서 프록시 설정 (선택 사항)

프록시를 사용하여 **OpenShift Container Platform** 클러스터를 배포하려면 **install-config.yaml** 파일을 다음과 같이 변경합니다.

```
apiVersion: v1
baseDomain: <domain>
proxy:
  httpProxy: http://USERNAME:PASSWORD@proxy.example.com:PORT
  httpsProxy: https://USERNAME:PASSWORD@proxy.example.com:PORT
  noProxy: <WILDCARD_OF_DOMAIN>,<PROVISIONING_NETWORK/CIDR>,
<BMC_ADDRESS_RANGE/CIDR>
```

다음은 값을 포함하는 **noProxy**의 예입니다.

```
noProxy: .example.com,172.22.0.0/24,10.10.0.0/24
```

프록시가 활성화된 상태에서 해당 키 / 값 쌍에 적절한 프록시 값을 설정합니다.

주요 고려 사항:

- 프록시에 **HTTPS** 프록시가 없는 경우 **httpsProxy** 값을 **https://**에서 **http://**로 변경합니다.
- 프로비저닝 네트워크를 사용하는 경우 이를 **noProxy** 설정에 포함하십시오. 그렇지 않으면 설치 프로그램이 실패합니다.
- 모든 프록시 설정을 프로버저너 노드 내에서 환경 변수로 설정합니다. 예를 들어, **HTTP_PROXY, https_proxy, NO_PROXY**가 포함됩니다.

참고

IPv6으로 프로비저닝할 때 noProxy 설정에서 CIDR 주소 블록을 정의할 수 없습니다. 각 주소를 개별적으로 정의해야 합니다.

10.3.6.3. provisioning 네트워크가 없는 경우 install-config.yaml 파일 변경 (선택 사항)

provisioning 네트워크없이 OpenShift Container Platform 클러스터를 배포하려면 install-config.yaml 파일을 다음과 같이 변경하십시오.

```
platform:
  baremetal:
    apiVIP: <api_VIP>
    ingressVIP: <ingress_VIP>
    provisioningNetwork: "Disabled" 1
```

1

필요한 경우 provisioningNetwork 구성 설정을 추가하고 Disabled 로 설정합니다.

중요

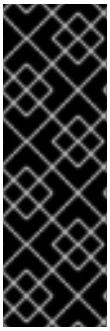
프로비저닝 네트워크는 PXE 부팅에 필요합니다. provisioning 네트워크 없이 배포하는 경우 redfish-virtualmedia 또는 idrac-virtualmedia 와 같은 가상 미디어 BMC 주소 지정 옵션을 사용해야 합니다. 자세한 내용은 "HPE iLO용 BMC 주소 지정" 섹션의 "Redfish virtual media for HPE iLO"를 참조하거나 "Dell iDRAC용 BMC 주소 지정" 섹션의 "Redfish virtual media for Dell iDRAC" 섹션을 참조하십시오.

10.3.6.4. 이중 스택 네트워크의 install-config.yaml 파일 변경 (선택 사항)

듀얼 스택 네트워킹으로 OpenShift Container Platform 클러스터를 배포하려면 install-config.yaml 파일에서 machineNetwork,clusterNetwork 및 serviceNetwork 구성 설정을 편집합니다. 각 설정에는 각각 두 개의 CIDR 항목이 있어야 합니다. 첫 번째 CIDR 항목이 IPv4 설정이며 두 번째 CIDR 항목이 IPv6 설정인지 확인합니다.

```
machineNetwork:
- cidr: {{ extcidrnet }}
- cidr: {{ extcidrnet6 }}
clusterNetwork:
- cidr: 10.128.0.0/14
  hostPrefix: 23
- cidr: fd02::/48
  hostPrefix: 64
```

```
serviceNetwork:
- 172.30.0.0/16
- fd03::/112
```



중요

듀얼 스택 네트워킹을 사용하는 경우 API VIP IP 주소와 Ingress VIP 주소가 기본 IP 주소 제품군이어야 합니다. 현재 Red Hat은 IPv6를 기본 IP 주소 제품군으로 사용하여 듀얼 스택 VIP 또는 듀얼 스택 네트워킹을 지원하지 않습니다. 하지만 Red Hat은 IPv4를 기본 IP 주소 제품군으로 사용하는 듀얼 스택 네트워킹을 지원합니다. 따라서 IPv4 항목은 IPv6 항목보다 먼저 이동해야 합니다.

10.3.6.5. install-config.yaml 파일에서 관리형 Secure Boot 구성 (선택 사항)

redfish, redfish -virtualmedia 또는 idrac-virtualmedia 와 같은 Redfish BMC 주소 지정을 사용하여 설치 관리자 프로비저닝 클러스터를 배포할 때 관리형 Secure Boot를 활성화할 수 있습니다. 관리형 Secure Boot를 활성화하려면 각 노드에 bootMode 구성 설정을 추가합니다.

예제

```
hosts:
  - name: openshift-master-0
    role: master
    bmc:
      address: redfish://<out_of_band_ip> 1
      username: <user>
      password: <password>
    bootMACAddress: <NIC1_mac_address>
    rootDeviceHints:
     deviceName: "/dev/sda"
    bootMode: UEFISecureBoot 2
```

1

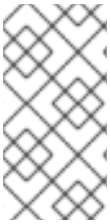
the bmc.address 설정이 redfish, redfish -virtualmedia 또는 idrac-virtualmedia를 프로토콜로 사용하는지 확인합니다. 자세한 내용은 "HPE iLO용 BMC 주소 지정" 또는 "Dell iDRAC의 BMC 주소 지정"을 참조하십시오.

2

bootMode 설정의 기본값은 UEFI입니다. 이 파일을 UEFISecureBoot로 변경하여 관리되는 Secure Boot를 활성화합니다.

참고

노드가 관리형 **Secure Boot**를 지원할 수 있는지 확인하려면 "사전 요구 사항"의 "노드 구성"을 참조하십시오. 노드가 관리형 **Secure Boot**를 지원하지 않는 경우 "노드 구성" 섹션의 "수동으로 **Secure Boot**의 노드 구성"을 참조하십시오. **Secure Boot**를 수동으로 구성하려면 **Redfish** 가상 미디어가 필요합니다.



참고

IPMI는 **Secure Boot** 관리 기능을 제공하지 않으므로 **Red Hat**은 **IPMI**를 사용하여 **Secure Boot**를 지원하지 않습니다.

10.3.6.6. 추가 install-config 매개 변수

install-config.yaml 파일의 필수 매개 변수 **hosts** 매개 변수 및 **bmc** 매개 변수는 다음 표를 참조하십시오.

표 10.3. 필수 매개 변수

매개 변수	기본	설명
baseDomain		클러스터의 도메인 이름입니다. 예: **example.com**
bootMode	**UEFI**	노드의 부팅 모드입니다. 옵션은 **legacy**, **UEFI** 및 **UEFISecureBoot**입니다. **bootMode**가 설정되지 않은 경우 Ironic은 노드를 검사하는 동안 해당 노드를 설정합니다.
sshKey		**sshKey** 구성 설정에는 컨트롤 플레인 노드 및 작업자 노드에 액세스하는 데 필요한 **~/.ssh/id_rsa.pub** 파일의 키가 포함되어 있습니다. 일반적으로 이 키는 **provisioner** 노드에서 가져옵니다.
pullSecret		**pullSecret** 구성 설정에는 프로비저너 노드를 준비할 때 Install OpenShift on Bare Metal 페이지에서 다운로드한 풀 시크릿 사본이 포함되어 있습니다.

매개 변수	기본	설명
metadata: name:		OpenShift Container Platform 클러스터에 지정되는 이름입니다. 예: **openshift**

networking:
  machineNetwork:
  - cidr:
		외부 네트워크의 공개 CIDR (Classless Inter-Domain Routing) 입니다. 예: **10.0.0.0/24**

compute: - name: worker		OpenShift Container Platform 클러스터에는 노드가 없는 경우에도 작업자 (또는 컴퓨팅) 노드에 이름을 지정해야 합니다.
compute: replicas: 2		복제는 OpenShift Container Platform 클러스터의 작업자 (또는 컴퓨팅) 노드 수를 설정합니다.
controlPlane: name: master		OpenShift Container Platform 클러스터에는 컨트롤 플레인 (마스터) 노드의 이름이 필요합니다.
controlPlane: replicas: 3		복제는 OpenShift Container Platform 클러스터의 일부로 포함된 컨트롤 플레인 (마스터) 노드의 수를 설정합니다.
provisioningNetworkInterface		**provisioning** 네트워크에 연결된 노드의 네트워크 인터페이스 이름입니다. OpenShift Container Platform 4.9 이상 릴리스의 경우 **bootMACAddress** 구성 설정을 사용하여 NIC 이름을 식별하는 대신 **provisioningNetworkInterface** 구성 설정을 사용하는 대신 NIC의 IP 주소를 식별할 수 있도록 Ironic을 활성화합니다.
defaultMachinePlatform		플랫폼 구성없이 머신 풀에 사용되는 기본 설정입니다.

매개 변수	기본	설명
apiVIP		(선택 사항) Kubernetes API 통신의 가상 IP 주소입니다. 이 설정은 **install-config.yaml** 파일에서 MachineNetwork에서 예약된 IP로 제공되거나 기본 이름이 올바르게 확인되도록 DNS에서 사전 구성해야 합니다. **install-config.yaml** 파일의 **apiVIP** 구성 설정에 값을 추가할 때 FQDN이 아닌 가상 IP 주소를 사용합니다. 듀얼 스택 네트워킹을 사용하는 경우 IP 주소는 기본 IPv4 네트워크에서여야 합니다. 설정하지 않으면 설치 프로그램에서 **api.<cluster_name>.<base_domain>** 을 사용하여 DNS에서 IP 주소를 파생합니다.
disableCertificateVerification	**False**	**redfish** 및 **redfish-virtualmedia** 는 BMC 주소를 관리하기 위해 이 매개 변수가 필요합니다. BMC 주소에 자체 서명된 인증서를 사용하는 경우 값은 **True** 여야합니다.
ingressVIP		(선택 사항) 수신 트래픽의 가상 IP 주소입니다. 이 설정은 **install-config.yaml** 파일에서 MachineNetwork에서 예약된 IP로 제공되거나 기본 이름이 올바르게 확인되도록 DNS에서 사전 구성해야 합니다. **install-config.yaml** 파일의 **ingressVIP** 구성 설정에 값을 추가할 때 FQDN이 아닌 가상 IP 주소를 사용합니다. 듀얼 스택 네트워킹을 사용하는 경우 IP 주소는 기본 IPv4 네트워크에서여야 합니다. 설정되지 않은 경우 설치 프로그램은 **test.apps.<cluster_name>.<base_domain>** 을 사용하여 DNS에서 IP 주소를 파생합니다.

표 10.4. 선택적 매개변수

매개 변수	기본	설명
provisioningDHCPRange	**172.22.0.10,172.22.0.100**	**provisioning** 네트워크에서 노드의 IP 범위를 정의합니다.
provisioningNetworkCIDR	**172.22.0.0/24**	프로비저닝에 사용할 네트워크의 CIDR입니다. 이 옵션은 **provisioning** 네트워크에서 기본 주소 범위를 사용하지 않는 경우에 필요합니다.
clusterProvisioningIP	**provisioningNetworkCIDR**의 세 번째 IP 주소입니다.	프로비저닝 서비스가 실행되는 클러스터 내의 IP 주소입니다. 기본값은 **provisioning** 서브넷의 세 번째 IP 주소입니다. 예: **172.22.0.3**
bootstrapProvisioningIP	**provisioningNetworkCIDR**의 두 번째 IP 주소입니다.	설치 프로그램이 컨트롤 플레인 (마스터) 노드를 배포하는 동안 프로비저닝 서비스가 실행되는 부트스트랩 VM의 IP 주소입니다. 기본값은 **provisioning** 서브넷의 두 번째 IP 주소입니다. 예를 들면 **172.22.0.2** 또는 **2620:52:0:1307::2** 입니다.
externalBridge	**baremetal**	**baremetal** 네트워크에 연결된 하이퍼 바이저의 **baremetal** 브리지 이름입니다.
provisioningBridge	**provisioning**	**provisioning** 네트워크에 연결된 **provisioner** 호스트의 **provisioning** 브리지 이름입니다.
defaultMachinePlatform		플랫폼 구성없이 머신 풀에 사용되는 기본 설정입니다.
bootstrapOSImage		부트스트랩 노드의 기본 운영 체제 이미지를 재정의하는 URL입니다. URL에는 이미지의 SHA-256 해시가 포함되어 있어야합니다. 예: **https://mirror.openshift.com/rhcos-<version>-qemu.qcow2.gz?sha256=<uncompressed_sha256>** .
clusterOSImage		클러스터 노드의 기본 운영 체제를 재정의하는 URL입니다. URL에는 이미지의 SHA-256 해시가 포함되어 있어야 합니다. 예: **https://mirror.openshift.com/images/rhcos-<version>-openstack.qcow2.gz?sha256=<compressed_sha256>**.

매개 변수	기본	설명
provisioningNetwork		**provisioningNetwork** 구성 설정은 클러스터가 **provisioning** 네트워크를 사용하는지 여부를 결정합니다. 이 기능이 있는 경우 구성 설정은 클러스터가 네트워크를 관리하는지도 결정합니다. **disabled**: **provisioning** 네트워크의 요구 사항을 비활성화하려면 이 매개변수를 **Disabled** 로 설정합니다. **Disabled(비활성화됨)**로 설정하는 경우 가상 미디어 기반 프로비저닝만 사용하거나 지원 설치 프로그램을 사용하여 클러스터를 가져와야 합니다. **Disabled** 및 power management를 사용하는 경우 **baremetal** 네트워크에서 BMC에 액세스할 수 있어야 합니다. **Disabled** 인 경우 프로비저닝 서비스에 사용되는 **baremetal** 네트워크에 두 개의 IP 주소를 제공해야 합니다. **Managed**: DHCP, TFTP 등을 포함한 프로비저닝 네트워크를 완전히 관리하려면 이 매개 변수를 기본값인 **Managed** 로 설정합니다. **Unmanaged**: 이 매개변수를 **Unmanaged** 로 설정하여 프로비저닝 네트워크를 활성화하지만 DHCP 수동 구성을 처리합니다. 가상 미디어의 프로비저닝이 권장되지만 필요한 경우 PXE를 계속 사용할 수 있습니다.
httpProxy		이 매개 변수를 환경 내에서 사용되는 적절한 HTTP 프록시로 설정합니다.
httpsProxy		이 매개 변수를 환경 내에서 사용되는 적절한 HTTPS 프록시로 설정합니다.
noProxy		이 매개 변수를 환경 내 프록시 사용에 대한 적절한 예외 목록으로 설정합니다.

호스트

hosts 매개 변수는 클러스터를 빌드하는 데 사용되는 별도의 베어 메탈 자산 목록입니다.

표 10.5. 호스트

이름	기본	설명
name		세부 정보와 연결할 **BareMetalHost** 리소스의 이름입니다. 예: **openshift-master-0**
role		베어 메탈 노드의 역할입니다. **master** 또는 **worker** 중 하나입니다.
bmc		베이스 보드 관리 컨트롤러에 대한 연결 세부 정보입니다. 자세한 내용은 BMC 주소 지정 섹션을 참조하십시오.

이름	기본	설명
bootMACAddress		호스트가 **provisioning** 네트워크에 사용하는 NIC의 MAC 주소입니다. Ironic은 **bootMACAddress** 구성 설정을 사용하여 IP 주소를 검색합니다. 그런 다음 호스트에 바인딩됩니다. **참고** **provisioning** 네트워크를 비활성화한 경우 호스트에서 유효한 MAC 주소를 제공해야 합니다.

10.3.6.7. BMC 주소 지정

대부분의 공급업체는 **IPMI(Intelligent Platform Management Interface)**를 사용하여 **BMC(Baseboard Management Controller)** 처리를 지원합니다. **IPMI**는 통신을 암호화하지 않습니다. 보안 또는 전용 관리 네트워크를 통해 데이터 센터 내에서 사용하기에 적합합니다. 공급 업체에 문의하여 **Redfish** 네트워크 부팅을 지원하는지 확인합니다. **Redfish**는 통합, 하이브리드 **IT** 및 소프트웨어 정의 데이터 센터 **(SDDC)**를 위한 간편한 보안 관리 기능을 제공합니다. **Redfish**는 사람이 읽을 수 있고 머신을 사용할 수 있으며 일반적인 인터넷 및 웹 서비스 표준을 활용하여 최신 툴 체인에 직접 정보를 노출합니다. 하드웨어에서 **Redfish** 네트워크 부팅을 지원하지 않는 경우 **IPMI**를 사용합니다.

IPMI

IPMI를 사용하는 호스트는 **ipmi://<out-of-band-ip>:<port>** 주소 형식을 사용하며 지정되지 않은 경우 기본적으로 포트 **623**으로 설정됩니다. 다음 예제는 **install-config.yaml** 파일 내의 **IPMI** 설정을 보여줍니다.

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: ipmi://<out-of-band-ip>
          username: <user>
          password: <password>
```

중요

BMC 주소 지정에 **IPMI**를 사용하여 **PXE** 부팅 시 프로비저닝 네트워크가 필요합니다. **provisioning** 네트워크없이는 **PXE** 부팅 호스트를 사용할 수 없습니다. **provisioning** 네트워크 없이 배포하는 경우 **redfish-virtualmedia** 또는 **idrac-virtualmedia** 와 같은 가상 미디어 **BMC** 주소 지정 옵션을 사용해야 합니다. 자세한 내용은 **"HPE iLO용 BMC 주소 지정"** 섹션의 **"Redfish virtual media for HPE iLO"**를 참조하거나 **"Dell iDRAC용 BMC 주소 지정"** 섹션의 **"Redfish virtual media for Dell iDRAC"** 섹션을 참조하십시오.

Redfish 네트워크 부팅

Redfish를 활성화하려면 **redfish://** 또는 **redfish+http://**를 사용하여 **TLS**를 비활성화합니다. 설치 프로그램에는 호스트 이름 또는 **IP** 주소와 시스템 **ID** 경로가 모두 필요합니다. 다음 예제는 **install-config.yaml** 파일 내의 **Redfish** 설정을 보여줍니다.

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: redfish://<out-of-band-ip>/redfish/v1/Systems/1
          username: <user>
          password: <password>
```

대역 외 관리 주소에 대한 인증 기관의 인증서를 사용하는 것이 좋지만 자체 서명 된 인증서를 사용하는 경우 **bmc** 설정에 **disableCertificateVerification:True**를 포함해야 합니다. 다음 예제는 **install-config.yaml** 파일 내의 **disableCertificateVerification:True** 설정 매개 변수를 사용하는 **Redfish** 설정을 보여줍니다.

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: redfish://<out-of-band-ip>/redfish/v1/Systems/1
          username: <user>
          password: <password>
          disableCertificateVerification: True
```

10.3.6.8. Dell iDRAC용 BMC 주소 지정

각 **bmc** 항목의 **address** 필드는 **URL** 체계의 컨트롤러 유형 및 네트워크에서의 위치를 포함하여 **OpenShift Container Platform** 클러스터 노드에 연결하기 위한 **URL**입니다.

```
platform:
  baremetal:
    hosts:
      - name: <hostname>
        role: <master | worker>
        bmc:
          address: <address> 1
          username: <user>
          password: <password>
```

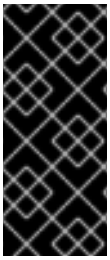
1

address 구성 설정은 프로토콜을 지정합니다.

Dell 하드웨어의 경우 Red Hat은 통합 iDRAC(Dell Remote Access Controller) 가상 미디어, Redfish 네트워크 부팅 및 IPMI를 지원합니다.

Dell iDRAC의 BMC 주소 형식

프로토콜	주소 형식
iDRAC 가상 미디어	idrac-virtualmedia://<out-of-band-ip>/redfish/v1/Systems/System.Embedded.1
Redfish 네트워크 부팅	redfish://<out-of-band-ip>/redfish/v1/Systems/System.Embedded.1
IPMI	ipmi://<out-of-band-ip>



중요

Redfish 가상 미디어의 프로토콜로 idrac-virtualmedia를 사용합니다. redfish-virtualmedia는 Dell 하드웨어에서 작동하지 않습니다. Dell의 idrac-virtualmedia는 Dell의 OEM 확장 기능과 함께 Redfish 표준을 사용합니다.

자세한 내용은 다음 섹션을 참조하십시오.

Dell iDRAC 용 Redfish 가상 미디어

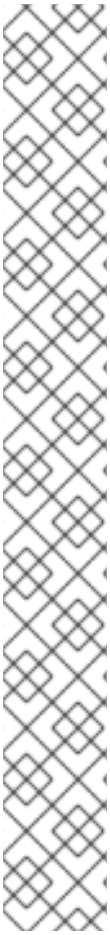
Dell 서버의 Redfish 가상 미디어의 경우 address 설정에서 idrac-virtualmedia://를 사용합니다. redfish-virtualmedia://를 사용하는 것은 작동하지 않습니다.

다음 예는 install-config.yaml 파일 내에서 iDRAC 가상 미디어를 사용하는 방법을 보여줍니다.

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: idrac-virtualmedia://<out-of-band-ip>/redfish/v1/Systems/System.Embedded.1
          username: <user>
          password: <password>
```

대역 외 관리 주소에 대한 인증 기관의 인증서를 사용하는 것이 좋지만 자체 서명 된 인증서를 사용하는 경우 bmc 설정에 disableCertificateVerification:True를 포함해야 합니다. 다음 예제는 install-config.yaml 파일 내의 disableCertificateVerification:True 설정 매개 변수를 사용하는 Redfish 설정을 보여줍니다.

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: idrac-virtualmedia://<out-of-band-ip>/redfish/v1/Systems/System.Embedded.1
          username: <user>
          password: <password>
          disableCertificateVerification: True
```



참고

현재 Redfish는 베어 메탈 배포 환경에서 설치 관리자가 프로비저닝한 설치에 대해 iDRAC 펌웨어 버전 4.20.20.20 ~ 04.40.00.00을 사용하는 Dell에서만 지원됩니다. 04.40.00.00 버전에 알려진 문제가 있습니다. iDRAC 9 펌웨어 버전 04.40.00.00 을 사용하면 가상 콘솔 플러그인이 기본적으로 eHTML5 로 설정되어 InsertVirtualMedia 워크플로우에 문제가 발생합니다. 이 문제를 방지하려면 플러그인을 HTML5 로 설정합니다. 메뉴 경로는 설정 → 가상 콘솔 → 플러그인 유형 → HTML5입니다.

OpenShift Container Platform 클러스터 노드에 iDRAC 콘솔을 통해 AutoAttach가 활성화되어 있는지 확인합니다. 메뉴 경로는 구성 → 가상 미디어 → 연결 모드 → 자동 연결입니다 .

idrac-virtualmedia://를 Redfish 가상 미디어의 프로토콜로 사용합니다. idrac-virtualmedia:// 프로토콜은 Ironic의 idrac 하드웨어 유형 및 Redfish 프로토콜에 해당하므로 redfish-virtualmedia://를 사용하면 Dell 하드웨어에서 작동하지 않습니다. Dell의 idrac-virtualmedia:// 프로토콜은 Dell의 OEM 확장에 Redfish 표준을 사용합니다. Ironic은 WSMAN 프로토콜로 idrac 유형도 지원합니다. 따라서 Dell 하드웨어에서 가상 미디어와 함께 Redfish를 사용하도록 선택할 때 예기치 않은 동작을 방지하려면 idrac-virtualmedia://를 지정해야 합니다.

iDRAC 용 Redfish 네트워크 부팅

Redfish를 활성화하려면 redfish:// 또는 redfish+http://를 사용하여 TLS(transport layer security)를 비활성화합니다. 설치 프로그램에는 호스트 이름 또는 IP 주소와 시스템 ID 경로가 모두 필요합니다. 다음 예제는 install-config.yaml 파일 내의 Redfish 설정을 보여줍니다.

```
platform:
  baremetal:
    hosts:
```

```
    - name: openshift-master-0
      role: master
      bmc:
        address: redfish://<out-of-band-ip>/redfish/v1/Systems/System.Embedded.1
        username: <user>
        password: <password>
```

대역 외 관리 주소에 대한 인증 기관의 인증서를 사용하는 것이 좋지만 자체 서명 된 인증서를 사용하는 경우 **bmc** 설정에 **disableCertificateVerification:True**를 포함해야 합니다. 다음 예제는 **install-config.yaml** 파일 내의 **disableCertificateVerification:True** 설정 매개 변수를 사용하는 **Redfish** 설정을 보여줍니다.

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: redfish://<out-of-band-ip>/redfish/v1/Systems/System.Embedded.1
          username: <user>
          password: <password>
          disableCertificateVerification: True
```

참고

현재 **Redfish**는 베어 메탈 배포 환경에서 설치 관리자가 프로비저닝한 설치에 대해 **iDRAC** 펌웨어 버전 **4.20.20.20 ~ 04.40.00.00**을 사용하는 **Dell** 하드웨어에서만 지원됩니다. **04.40.00.00** 버전에 알려진 문제가 있습니다. **iDRAC 9** 펌웨어 버전 **04.40.00.00** 을 사용하면 가상 콘솔 플러그인이 기본적으로 **eHTML5** 로 설정되어 **InsertVirtualMedia** 워크플로우에 문제가 발생합니다. 이 문제를 방지하려면 플러그인을 **HTML5** 로 설정합니다. 메뉴 경로는 설정 → 가상 콘솔 → 플러그인 유형 → **HTML5**입니다.

OpenShift Container Platform 클러스터 노드에 **iDRAC** 콘솔을 통해 **AutoAttach**가 활성화되어 있는지 확인합니다. 메뉴 경로는 구성 → 가상 미디어 → 연결 모드 → 자동 연결입니다 .

redfish:// URL 프로토콜은 **Ironic**의 **redfish** 하드웨어 유형에 해당합니다.

10.3.6.9. HPE iLO용 BMC 주소 지정

각 **bmc** 항목의 **address** 필드는 **URL** 체계의 컨트롤러 유형 및 네트워크에서의 위치를 포함하여 **OpenShift Container Platform** 클러스터 노드에 연결하기 위한 **URL**입니다.

```
platform:
```

```
baremetal:
  hosts:
    - name: <hostname>
      role: <master | worker>
      bmc:
        address: <address> 1
        username: <user>
        password: <password>
```

1

address 구성 설정은 프로토콜을 지정합니다.

HPE iLO(integrated Lights Out)의 경우 Red Hat은 Redfish 가상 미디어, Redfish 네트워크 부팅 및 IPMI를 지원합니다.

표 10.6. HPE iLO의 BMC 주소 형식

프로토콜	주소 형식
RedFish 가상 미디어	redfish-virtualmedia://<out-of-band-ip>/redfish/v1/Systems/1
Redfish 네트워크 부팅	redfish://<out-of-band-ip>/redfish/v1/Systems/1
IPMI	ipmi://<out-of-band-ip>

자세한 내용은 다음 섹션을 참조하십시오.

HPE iLO용 Redfish 가상 미디어

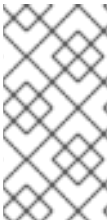
HPE 서버용 Redfish 가상 미디어를 활성화하려면 address 설정에서 redfish-virtualmedia://를 사용합니다. 다음 예제는 install-config.yaml 파일 내에서 Redfish 가상 미디어를 사용하는 방법을 보여줍니다.

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: redfish-virtualmedia://<out-of-band-ip>/redfish/v1/Systems/1
          username: <user>
          password: <password>
```

대역 외 관리 주소에 대한 인증 기관의 인증서를 사용하는 것이 좋지만 자체 서명 된 인증서를 사용하는 경우 bmc 설정에 disableCertificateVerification:True를 포함해야 합니다. 다음 예제는 install-

config.yaml 파일 내의 disableCertificateVerification:True 설정 매개 변수를 사용하는 Redfish 설정을 보여줍니다.

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: redfish-virtualmedia://<out-of-band-ip>/redfish/v1/Systems/1
          username: <user>
          password: <password>
          disableCertificateVerification: True
```



참고

Ironic은 가상 미디어에서 iLO4를 지원하지 않으므로 iLO4를 실행하는 9세대 시스템에서 Redfish 가상 미디어가 지원되지 않습니다.

HPE iLO용 Redfish 네트워크 부팅

Redfish를 활성화하려면 redfish:// 또는 redfish+http://를 사용하여 TLS를 비활성화합니다. 설치 프로그램에는 호스트 이름 또는 IP 주소와 시스템 ID 경로가 모두 필요합니다. 다음 예제는 install-config.yaml 파일 내의 Redfish 설정을 보여줍니다.

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: redfish://<out-of-band-ip>/redfish/v1/Systems/1
          username: <user>
          password: <password>
```

대역 외 관리 주소에 대한 인증 기관의 인증서를 사용하는 것이 좋지만 자체 서명 된 인증서를 사용하는 경우 bmc 설정에 disableCertificateVerification:True를 포함해야 합니다. 다음 예제는 install-config.yaml 파일 내의 disableCertificateVerification:True 설정 매개 변수를 사용하는 Redfish 설정을 보여줍니다.

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: redfish://<out-of-band-ip>/redfish/v1/Systems/1
```

```
      username: <user>
      password: <password>
      disableCertificateVerification: True
```

10.3.6.10. Fujitsu iRMC용 BMC 주소 지정

각 bmc 항목의 address 필드는 URL 체계의 컨트롤러 유형 및 네트워크에서의 위치를 포함하여 OpenShift Container Platform 클러스터 노드에 연결하기 위한 URL입니다.

```
platform:
  baremetal:
    hosts:
      - name: <hostname>
        role: <master | worker>
        bmc:
          address: <address> 1
          username: <user>
          password: <password>
```

1

address 구성 설정은 프로토콜을 지정합니다.

Fujitsu 하드웨어의 경우 Red Hat은 통합된 iRMC(Remote Management Controller) 및 IPMI를 지원합니다.

표 10.7. Fujitsu iRMC의 BMC 주소 형식

프로토콜	주소 형식
iRMC	irmc://<out-of-band-ip>
IPMI	ipmi://<out-of-band-ip>

iRMC

Fujitsu 노드는 irmc://<out-of-band-ip >를 사용할 수 있으며 기본값은 포트 443 입니다. 다음 예제는 install-config.yaml 파일 내의 iRMC 설정을 보여줍니다.

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
```

```
address: irmc://<out-of-band-ip>
username: <user>
password: <password>
```

참고

현재 Fujitsu는 베어 메탈에 설치 관리자 프로비저닝 설치를 위해 iRMC S5 펌웨어 버전 3.05P 이상을 지원합니다.

10.3.6.11. 루트 장치 팁

rootDeviceHints 매개 변수를 사용하면 설치 프로그램이 RHCOS (Red Hat Enterprise Linux CoreOS) 이미지를 특정 장치에 프로비저닝할 수 있습니다. 설치 프로그램은 장치를 검색한 순서대로 검사하고 검색된 값을 팁과 비교합니다. 설치 프로그램은 팁과 일치하는 첫 번째 검색된 장치를 사용합니다. 이 설정은 여러 팁을 결합할 수 있지만 장치는 설치 프로그램이이를 선택할 수 있도록 모든 팁과 일치해야 합니다.

표 10.8. 서브 필드

서브 필드	설명
deviceName	**/dev/vda**와 같은 Linux 장치 이름을 포함하는 문자열. 팁은 실제 값과 정확히 일치해야 합니다.
hctl	**0:0:0:0**과 같은 SCSI 버스 주소를 포함하는 문자열. 팁은 실제 값과 정확히 일치해야 합니다.
model	공급 업체별 장치 식별자가 포함된 문자열. 팁은 실제 값의 하위 문자열입니다.
vendor	장치의 공급 업체 또는 제조업체 이름이 포함된 문자열입니다. 팁은 실제 값의 하위 문자열입니다.
serialNumber	장치 일련 번호가 포함된 문자열입니다. 팁은 실제 값과 정확히 일치해야 합니다.
minSizeGigabytes	장치의 최소 크기 (기가 바이트)를 나타내는 정수입니다.
wwn	고유 저장소 식별자를 포함하는 문자열입니다. 팁은 실제 값과 정확히 일치해야 합니다.
wwnWithExtension	공급 업체 확장이 추가된 고유 한 저장소 식별자가 포함된 문자열입니다. 팁은 실제 값과 정확히 일치해야 합니다.

서브 필드	설명
wwnVendorExtension	고유 공급 업체 저장소 식별자를 포함하는 문자열입니다. 팁은 실제 값과 정확히 일치해야 합니다.
rotational	장치가 회전 디스크 여야하는지 (true) 아닌지 (false)를 나타내는 부울 값입니다.

사용 예

```
 - name: master-0
   role: master
   bmc:
     address: ipmi://10.10.0.3:6203
     username: admin
     password: redhat
   bootMACAddress: de:ad:be:ef:00:40
   rootDeviceHints:
     deviceName: "/dev/sda"
```

10.3.6.12. OpenShift Container Platform 매니페스트 만들기

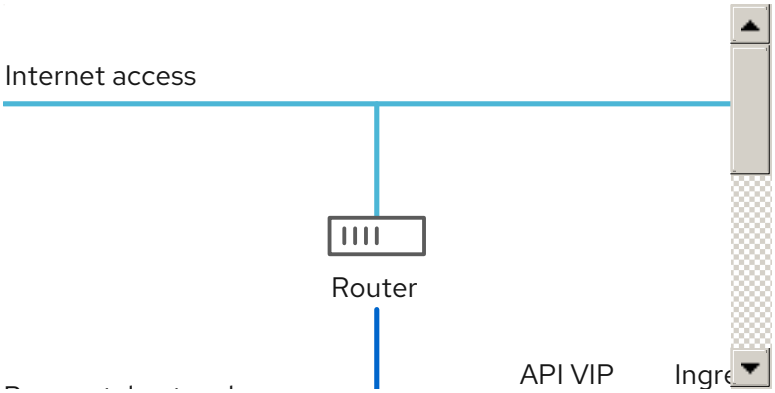
1. **OpenShift Container Platform 매니페스트를 만듭니다.**

```
$ ./openshift-baremetal-install --dir ~/clusterconfigs create manifests
```

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true
for Scheduler cluster settings
WARNING Discarding the OpenShift Manifest that was provided in the target directory
because its dependencies are dirty and it needs to be regenerated
```

10.3.6.13. 연결이 끊긴 클러스터의 NTP 구성 (선택 사항)

OpenShift Container Platform은 클러스터 노드에 chrony Network Time Protocol(NTP) 서비스를 설치합니다.



OpenShift Container Platform 노드는 올바로 실행되려면 날짜와 시간에 동의해야 합니다. 작업자 노드는 컨트롤 플레인 노드의 **NTP** 서버에서 날짜와 시간을 검색하면 라우팅 가능한 네트워크에 연결되지 않은 클러스터를 설치 및 운영할 수 있으므로 상위 계층 **NTP** 서버에 액세스할 수 없습니다.

절차

1. 컨트롤 플레인 노드에 대한 **chrony.conf** 파일의 콘텐츠를 포함하여 **Butane** 구성, **99-master-chrony-conf-override.bu**를 만듭니다.

참고

Butane에 대한 자세한 내용은 **"Butane** 을 사용하여 머신 구성 생성"을 참조하십시오.

Butane 구성의 예

```
variant: openshift
version: 4.9.0
metadata:
  name: 99-master-chrony-conf-override
  labels:
    machineconfiguration.openshift.io/role: master
storage:
  files:
    - path: /etc/chrony.conf
      mode: 0644
      overwrite: true
      contents:
        inline: |
          # Use public servers from the pool.ntp.org project.
          # Please consider joining the pool (https://www.pool.ntp.org/join.html).

          # The Machine Config Operator manages this file
          server openshift-master-0.<cluster-name>.<domain> iburst 1
```

```
server openshift-master-1.<cluster-name>.<domain> iburst
server openshift-master-2.<cluster-name>.<domain> iburst

stratumweight 0
driftfile /var/lib/chrony/drift
rtcsync
makestep 10 3
bindcmdaddress 127.0.0.1
bindcmdaddress ::1
keyfile /etc/chrony.keys
commandkey 1
generatecommandkey
noclientlog
logchange 0.5
logdir /var/log/chrony

# Configure the control plane nodes to serve as local NTP servers
# for all worker nodes, even if they are not in sync with an
# upstream NTP server.

# Allow NTP client access from the local network.
allow all
# Serve time even if not synchronized to a time source.
local stratum 3 orphan
```

1

<cluster-name>을 클러스터 이름으로 바꾸고 **<domain>**을 정규화된 도메인 이름으로 교체해야 합니다.

2. **Butane**을 사용하여 컨트롤 플레인 노드에 전달할 구성이 포함된 **MachineConfig** 파일 **99-master-chrony-conf-override.yaml**을 생성합니다.

```
$ butane 99-master-chrony-conf-override.bu -o 99-master-chrony-conf-override.yaml
```

3. 컨트롤 플레인 노드의 **NTP** 서버를 참조하는 작업자 노드의 **chrony.conf** 파일의 내용을 포함하여 **Butane** 구성 **99-worker-chrony-conf-override.bu**를 만듭니다.

Butane 구성의 예

```
variant: openshift
version: 4.9.0
metadata:
```

```
  name: 99-worker-chrony-conf-override
  labels:
    machineconfiguration.openshift.io/role: worker
storage:
  files:
    - path: /etc/chrony.conf
      mode: 0644
      overwrite: true
      contents:
        inline: |
          # The Machine Config Operator manages this file.
          server openshift-master-0.<cluster-name>.<domain> iburst 1
          server openshift-master-1.<cluster-name>.<domain> iburst
          server openshift-master-2.<cluster-name>.<domain> iburst

          stratumweight 0
          driftfile /var/lib/chrony/drift
          rtcsync
          makestep 10 3
          bindcmdaddress 127.0.0.1
          bindcmdaddress ::1
          keyfile /etc/chrony.keys
          commandkey 1
          generatecommandkey
          noclientlog
          logchange 0.5
          logdir /var/log/chrony
```

1

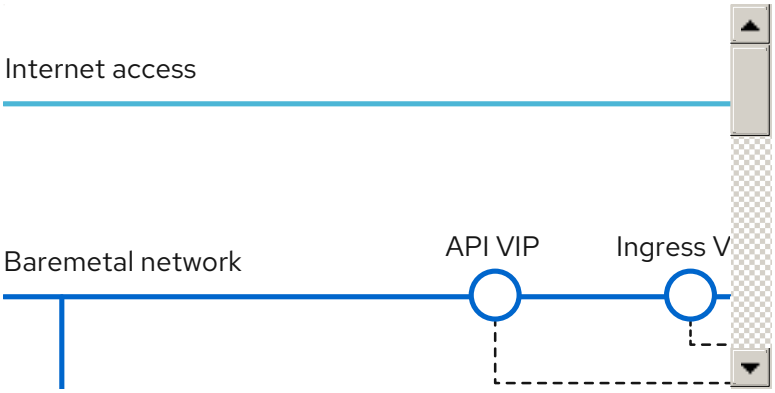
<cluster-name>을 클러스터 이름으로 바꾸고 **<domain>**을 정규화된 도메인 이름으로 교체해야 합니다.

4. **Butane**을 사용하여 작업자 노드로 전달할 구성이 포함된 **MachineConfig** 개체 파일 **99-worker-chrony-conf-override.yaml**을 생성합니다.

```
$ butane 99-worker-chrony-conf-override.bu -o 99-worker-chrony-conf-override.yaml
```

10.3.6.14. (선택 사항) 컨트롤 플레인에서 실행되도록 네트워크 구성 요소 구성

컨트롤 플레인 노드에서 독점적으로 실행되도록 네트워킹 구성 요소를 구성할 수 있습니다. 기본적으로 **OpenShift Container Platform**에서는 머신 구성 풀의 모든 노드가 **ingressVIP** 가상 **IP** 주소를 호스팅할 수 있습니다. 그러나 일부 환경에서는 컨트롤 플레인 노드와 별도의 서브넷에 작업자 노드를 배포합니다. 별도의 서브넷에 원격 작업자를 배포하는 경우 **ingressVIP** 가상 **IP** 주소를 컨트롤 플레인 노드 전용으로 배치해야 합니다.



절차

1. **install-config.yaml** 파일을 저장하는 디렉터리로 변경합니다.

```
$ cd ~/clusterconfigs
```

2. **manifests** 하위 디렉터리로 전환합니다.

```
$ cd manifests
```

3. **cluster-network-avoid-workers-99-config.yaml**이라는 파일을 생성합니다.

```
$ touch cluster-network-avoid-workers-99-config.yaml
```

4. 편집기에서 **cluster-network-avoid-workers-99-config.yaml** 파일을 열고 **Operator** 구성을 설명하는 **CR**(사용자 정의 리소스)을 입력합니다.

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: 50-worker-fix-ipi-rwn
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
        - path: /etc/kubernetes/manifests/keepalived.yaml
          mode: 0644
          contents:
            source: data:,
```

이 매니페스트는 컨트롤 플레인 노드에 **ingressVIP** 가상 **IP** 주소를 배치합니다. 또한 이 매니페스트는 컨트롤 플레인 노드에 다음 프로세스를 배포합니다.

- **openshift-ingress-operator**
- **keepalived**

5. **cluster-network-avoid-workers-99-config.yaml** 파일을 저장합니다.

6. **manifests/cluster-ingress-default-ingresscontroller.yaml** 파일을 생성합니다.

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: default
  namespace: openshift-ingress-operator
spec:
  nodePlacement:
    nodeSelector:
      matchLabels:
        node-role.kubernetes.io/master: ""
```

7. **manifests** 디렉터리를 백업하는 것이 좋습니다. 설치 프로그램은 클러스터를 생성할 때 **manifests/** 디렉터리를 삭제합니다.

8. **mastersSchedulable** 필드를 **true**로 설정하여 컨트롤 플레인 노드를 예약할 수 있도록 **cluster-scheduler-02-config.yml** 매니페스트를 수정합니다. 기본적으로 컨트롤 플레인 노드는 예약할 수 없습니다. 예를 들면 다음과 같습니다.

```
$ sed -i "s;mastersSchedulable: false;mastersSchedulable: true;g"
clusterconfigs/manifests/cluster-scheduler-02-config.yml
```

참고

이 절차를 완료한 후 컨트롤 플레인 노드를 예약할 수 없는 경우 클러스터 배포가 실패합니다.

10.3.6.15. 작업자 노드용 BIOS 구성

다음 절차에서는 설치 프로세스 중에 작업자 노드에 대한 BIOS를 구성합니다.

절차

1. 매니페스트를 생성합니다.

2. 작업자에 해당하는 BMH 파일을 수정합니다.

```
$ vim clusterconfigs/openshift/99_openshift-cluster-api_hosts-3.yaml
```

3. BMH 파일의 spec 섹션에 BIOS 구성을 추가합니다.

```
spec:
  firmware:
    simultaneousMultithreadingEnabled: true
    sriovEnabled: true
    virtualizationEnabled: true
```

참고

1. Red Hat은 다음 세 가지 BIOS 설정을 지원합니다. 자세한 내용은 BMH 설명서 를 참조하십시오. bmc 유형 irmc 가 있는 서버만 지원됩니다. 다른 유형의 서버는 현재 지원되지 않습니다.

4. 클러스터 생성.

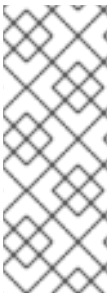
10.3.7. 비 연결 레지스트리 만들기 (선택 사항)

설치 레지스트리의 로컬 사본을 사용하여 OpenShift Container Platform 클러스터를 설치해야하는 경우가 있습니다. 이는 클러스터 노드가 인터넷에 액세스할 수 없는 네트워크에 있기 때문에 네트워크 효율성을 향상시키기 위한 것일 수 있습니다.

로컬 또는 미러링된 레지스트리 사본에는 다음이 필요합니다.

- 레지스트리 노드의 인증서. 자체 서명된 인증서를 사용할 수 있습니다.

- 시스템의 컨테이너가 제공할 웹 서버입니다.
- 인증서 및 로컬 저장소 정보를 포함하는 업데이트된 풀 시크릿.



참고

레지스트리 노드에 연결되지 않은 레지스트리를 만드는 것은 선택 사항입니다. 다음 섹션은 선택 사항으로 레지스트리 노드에 연결되지 않은 레지스트리를 만드는 경우에만 실행해야 하는 단계입니다. 레지스트리 노드에 연결되지 않은 레지스트리를 만들 때 "(선택 사항)"레이블이 붙은 모든 후속 하위 섹션을 실행해야 합니다.

10.3.7.1. 미러링된 레지스트리를 호스팅할 레지스트리 노드 준비 (선택 사항)

레지스트리 노드를 다음과 같이 변경합니다.

절차

1. 레지스트리 노드에서 방화벽 포트를 엽니다.

```
$ sudo firewall-cmd --add-port=5000/tcp --zone=libvirt  --permanent
$ sudo firewall-cmd --add-port=5000/tcp --zone=public   --permanent
$ sudo firewall-cmd --reload
```

2. 레지스트리 노드에 필요한 패키지를 설치합니다.

```
$ sudo yum -y install python3 podman httpd httpd-tools jq
```

3. 저장소 정보가 보관될 디렉터리 구조를 만듭니다.

```
$ sudo mkdir -p /opt/registry/{auth,certs,data}
```

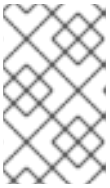
10.3.7.2. 자체 서명 인증서 생성 (선택 사항)

레지스트리 노드의 자체 서명된 인증서를 생성하고 이를 **/opt/registry/certs** 디렉터리에 배치합니다.

절차

1. 인증서 정보를 적절하게 조정합니다.

```
$ host_fqdn=$( hostname --long )
$ cert_c="<Country Name>"   # Country Name (C, 2 letter code)
$ cert_s="<State>"          # Certificate State (S)
$ cert_l="<Locality>"       # Certificate Locality (L)
$ cert_o="<Organization>"   # Certificate Organization (O)
$ cert_ou="<Org Unit>"      # Certificate Organizational Unit (OU)
$ cert_cn="${host_fqdn}"    # Certificate Common Name (CN)

$ openssl req \
   -newkey rsa:4096 \
   -nodes \
   -sha256 \
   -keyout /opt/registry/certs/domain.key \
   -x509 \
   -days 365 \
   -out /opt/registry/certs/domain.crt \
   -addext "subjectAltName = DNS:${host_fqdn}" \
   -subj
"/C=${cert_c}/ST=${cert_s}/L=${cert_l}/O=${cert_o}/OU=${cert_ou}/CN=${cert_cn}"
```



참고

<Country Name>을 바꾸는 경우에는 두 문자 만 사용합니다. 예: US

2. 새 인증서로 레지스트리 노드의 ca-trust를 업데이트합니다.

```
$ sudo cp /opt/registry/certs/domain.crt /etc/pki/ca-trust/source/anchors/
$ sudo update-ca-trust extract
```

10.3.7.3. 레지스트리 podman 컨테이너 만들기 (선택 사항)

레지스트리 컨테이너는 인증서, 인증 파일 및 데이터 파일 저장에 /opt/registry 디렉터리를 사용합니다.

레지스트리 컨테이너는 httpd를 사용하며 인증을 위해 htpasswd 파일이 필요합니다.

절차

1. 컨테이너에서 사용할 htpasswd 파일을 /opt/registry/auth에 만듭니다.

```
$ htpasswd -bBc /opt/registry/auth/htpasswd <user> <passwd>
```

<user>를 사용자 이름으로, <passwd>를 암호로 바꿉니다.

2. 레지스트리 컨테이너를 만들고 시작합니다.

```
$ podman create \
  --name ocpdiscon-registry \
  -p 5000:5000 \
  -e "REGISTRY_AUTH=htpasswd" \
  -e "REGISTRY_AUTH_HTPASSWD_REALM=Registry" \
  -e "REGISTRY_HTTP_SECRET=ALongRandomSecretForRegistry" \
  -e "REGISTRY_AUTH_HTPASSWD_PATH=/auth/htpasswd" \
  -e "REGISTRY_HTTP_TLS_CERTIFICATE=/certs/domain.crt" \
  -e "REGISTRY_HTTP_TLS_KEY=/certs/domain.key" \
  -e "REGISTRY_COMPATIBILITY_SCHEMA1_ENABLED=true" \
  -v /opt/registry/data:/var/lib/registry:z \
  -v /opt/registry/auth:/auth:z \
  -v /opt/registry/certs:/certs:z \
  docker.io/library/registry:2
```

```
$ podman start ocpdiscon-registry
```

10.3.7.4. 풀 시크릿(pull-secret) 복사 및 업데이트 (선택 사항)

프로비저너 노드에서 레지스트리 노드로 풀 시크릿 파일을 복사하고 이를 새 레지스트리 노드의 인증 정보를 포함하도록 변경합니다.

절차

1. pull-secret.txt 파일을 복사합니다.

```
$ scp kni@provisioner:/home/kni/pull-secret.txt pull-secret.txt
```

2. host_fqdn 환경 변수를 레지스트리 노드의 완전한 도메인 이름으로 업데이트합니다.

```
$ host_fqdn=$( hostname --long )
```

3. htpasswd 파일을 만드는 데 사용되는 http 인증 정보의 base64 인코딩으로 b64auth 환경 변수를 업데이트합니다.

```
$ b64auth=$( echo -n '<username>:<passwd>' | openssl base64 )
```

<username>을 사용자 이름으로, <passwd>를 암호로 바꿉니다.

4. base64 승인 문자열을 사용하도록 AUTHSTRING 환경 변수를 설정합니다. $USER 변수는 현재 사용자의 이름을 포함하는 환경 변수입니다.

```
$ AUTHSTRING="{\"$host_fqdn:5000\": {\"auth\": \"$b64auth\",\"email\":
\"$USER@redhat.com\"}}"
```

5. pull-secret.txt 파일을 업데이트합니다.

```
$ jq ".auths += $AUTHSTRING" < pull-secret.txt > pull-secret-update.txt
```

10.3.7.5. 저장소 미러링 (선택 사항)

프로세스

1. 프로비저너 노드에서 레지스트리 노드에 oc 바이너리를 복사합니다.

```
$ sudo scp kni@provisioner:/usr/local/bin/oc /usr/local/bin
```

2. 필요한 환경 변수를 설정합니다.

   a. 릴리스 버전을 설정합니다.

```
$ VERSION=<release_version>
```

   <release_version> 에 대해 설치할 OpenShift Container Platform 버전에 해당하는 태그를 지정합니다 (예: 4.9).

   b. 로컬 레지스트리 이름 및 호스트 포트를 설정합니다.

```
$ LOCAL_REG='<local_registry_host_name>:<local_registry_host_port>'
```

   <local_registry_host_name>의 경우 미러 저장소의 레지스트리 도메인 이름을 지정

하고 <local_registry_host_port>의 경우 콘텐츠를 제공하는데 사용되는 포트를 지정합니다.

c. 로컬 리포지터리 이름을 설정합니다.

```
$ LOCAL_REPO='<local_repository_name>'
```

<local_repository_name>의 경우 레지스트리에 작성할 저장소 이름 (예: ocp4/openshift4)을 지정합니다.

3. 원격 설치 이미지를 로컬 저장소에 미러링합니다.

```
$ /usr/local/bin/oc adm release mirror \
  -a pull-secret-update.txt \
  --from=$UPSTREAM_REPO \
  --to-release-image=$LOCAL_REG/$LOCAL_REPO:${VERSION} \
  --to=$LOCAL_REG/$LOCAL_REPO
```

10.3.7.6. 비 연결 레지스트리를 사용하도록 install-config.yaml 파일 변경 (선택 사항)

프로비저너 노드에서 install-config.yaml 파일은 pull-secret-update.txt 파일에서 새로 생성된 풀 시크릿(pull-secret)을 사용해야 합니다. install-config.yaml 파일에는 연결되지 않은 레지스트리 노드 인증서 및 레지스트리 정보도 포함되어야합니다.

프로세스

1. 비 연결 레지스트리 노드의 인증서를 install-config.yaml 파일에 추가합니다. 인증서는 "additionalTrustBundle:|" 뒤에 있어야 하며 보통 두 개의 공백으로 적절하게 들여 쓰기합니다.

```
$ echo "additionalTrustBundle: |" >> install-config.yaml
$ sed -e 's/^/  /' /opt/registry/certs/domain.crt >> install-config.yaml
```

2. 레지스트리의 미러 정보를 install-config.yaml 파일에 추가합니다.

```
$ echo "imageContentSources:" >> install-config.yaml
$ echo "- mirrors:" >> install-config.yaml
$ echo "  - registry.example.com:5000/ocp4/openshift4" >> install-config.yaml
$ echo "  source: quay.io/openshift-release-dev/ocp-release" >> install-config.yaml
$ echo "- mirrors:" >> install-config.yaml
```

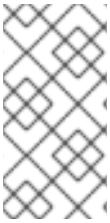
```
$ echo "  - registry.example.com:5000/ocp4/openshift4" >> install-config.yaml
$ echo "  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev" >> install-
config.yaml
```

참고

registry.example.com을 레지스트리의 정규화된 도메인 이름으로 바꿉니다.

10.3.8. 작업자 노드에 라우터 배치

설치 중에 설치 프로그램은 작업자 노드에 라우터 pod를 배포합니다. 기본적으로 설치 프로그램은 두 개의 라우터 pod를 설치합니다. 초기 클러스터에 작업자 노드가 하나만 있거나 배포된 클러스터에서 OpenShift Container Platform 클러스터 내의 서비스에 대해 외부 트래픽 로드를 처리하기 위해 추가 라우터가 필요한 경우 yaml 파일을 작성하여 적절한 라우터 복제 수를 설정할 수 있습니다.



참고

기본적으로 설치 프로그램은 두 개의 라우터를 배포합니다. 클러스터에 작업자 노드가 두 개 이상있는 경우 이 섹션을 생략할 수 있습니다.

참고

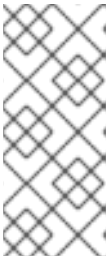
클러스터에 작업자 노드가 없는 경우 설치 프로그램은 기본적으로 컨트롤 플레인 노드에 두 개의 라우터를 배포합니다. 클러스터에 작업자 노드가없는 경우 이 섹션을 생략할 수 있습니다.

프로세스

1. router-replicas.yaml 파일을 만듭니다.

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: default
  namespace: openshift-ingress-operator
spec:
  replicas: <num-of-router-pods>
  endpointPublishingStrategy:
    type: HostNetwork
  nodePlacement:
    nodeSelector:
      matchLabels:
        node-role.kubernetes.io/worker: ""
```

-



참고

<num-of-router-pods>를 적절한 값으로 바꿉니다. 하나의 작업자 노드로만 작업하는 경우 replicas : 를 1로 설정합니다. 3 개 이상의 작업자 노드로 작업하는 경우 필요에 따라 replicas:을 기본값 2에서 늘릴 수 있습니다.

2. router-replicas.yaml 파일을 clusterconfigs/openshift 디렉터리에 저장 및 복사합니다.

```
cp ~/router-replicas.yaml clusterconfigs/openshift/99_router-replicas.yaml
```

10.3.9. 설치 확인 체크리스트

❑ OpenShift Container Platform 설치 프로그램이 검색되었습니다.

❑ OpenShift Container Platform 설치 프로그램이 추출되었습니다.

❑ install-config.yaml의 필수 매개 변수가 설정되었습니다.

❑ install-config.yaml의 hosts 매개 변수가 구성되었습니다.

❑ install-config.yaml의 bmc 매개 변수가 구성되었습니다.

❑ bmc address 필드에 구성된 값에 대한 규칙이 적용되었습니다.

❑ 비 연결 레지스트리를 생성했습니다 (선택 사항).

❑ (선택 사항) 비 연결 레지스트리 설정을 사용하고 있는 경우 이를 확인합니다.

❑ (선택 사항) 작업자 노드에 라우터를 배포했습니다.

10.3.10. OpenShift Container Platform 설치 프로그램을 통해 클러스터 배포

OpenShift Container Platform 설치 프로그램을 실행합니다.

```
$ ./openshift-baremetal-install --dir ~/clusterconfigs --log-level debug create cluster
```

10.3.11. 설치 후

배포 프로세스 중에 설치 디렉터리 폴더의 .openshift_install.log 로그 파일에 tail 명령을 실행하여 설치의 전체 상태를 확인할 수 있습니다.

```
$ tail -f /path/to/install-dir/.openshift_install.log
```

10.3.12. 고정 IP 주소 구성 확인

클러스터 노드의 DHCP 예약에서 무한 리스를 지정하는 경우 설치 프로그램이 노드를 성공적으로 프로비저닝한 후 디스패치 스크립트에서 노드의 네트워크 구성을 확인합니다. 스크립트에서 네트워크 구성에 무한 DHCP 리스가 포함되어 있음을 확인하면 DHCP 리스의 IP 주소를 고정 IP 주소로 사용하여 새 연결을 생성합니다.

참고

클러스터에 있는 다른 노드의 프로비저닝이 진행되는 동안 디스패치 스크립트는 성공적으로 프로비저닝된 노드에서 실행될 수 있습니다.

네트워크 구성이 제대로 작동하는지 확인합니다.

프로세스

1. 노드의 네트워크 인터페이스 구성을 확인합니다.

2. DHCP 서버를 끄고 OpenShift Container Platform 노드를 재부팅하고 네트워크 구성이 제대로 작동하는지 확인합니다.

추가 리소스

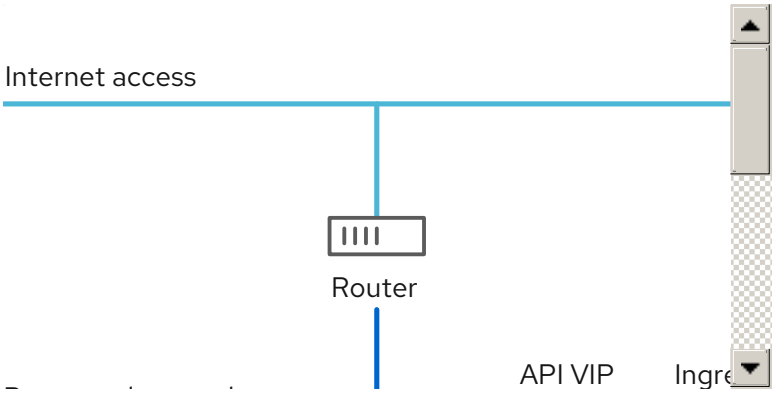
- 다양한 릴리스 채널에 대한 자세한 내용은 OpenShift Container Platform 업그레이드 채널 및 릴리스를 참조하십시오.

10.4. 설치 관리자가 프로비저닝한 설치 후 구성

설치 관리자 프로비저닝 클러스터를 성공적으로 배포한 후 다음 설치 후 절차를 고려하십시오.

10.4.1. 연결이 끊긴 클러스터의 NTP 구성 (선택 사항)

OpenShift Container Platform은 클러스터 노드에 **chrony Network Time Protocol(NTP)** 서비스를 설치합니다. 다음 절차에 따라 컨트롤 플레인 노드에서 **NTP** 서버를 구성하고 성공적으로 배포 후 작업자 노드를 컨트롤 플레인 노드의 **NTP** 클라이언트로 구성합니다.



OpenShift Container Platform 노드는 올바로 실행되려면 날짜와 시간에 동의해야 합니다. 작업자 노드는 컨트롤 플레인 노드의 **NTP** 서버에서 날짜와 시간을 검색하면 라우팅 가능한 네트워크에 연결되지 않은 클러스터를 설치 및 운영할 수 있으므로 상위 계층 **NTP** 서버에 액세스할 수 없습니다.

프로세스

1. 컨트롤 플레인 노드에 대한 **chrony.conf** 파일의 콘텐츠를 포함하여 **Butane** 구성, **99-master-chrony-conf-override.bu**를 만듭니다.

참고

Butane에 대한 자세한 내용은 **"Butane** 을 사용하여 머신 구성 생성"을 참조하십시오.

Butane 구성의 예

```
variant: openshift
version: 4.9.0
metadata:
```

```
  name: 99-master-chrony-conf-override
  labels:
    machineconfiguration.openshift.io/role: master
storage:
  files:
    - path: /etc/chrony.conf
      mode: 0644
      overwrite: true
      contents:
        inline: |
          # Use public servers from the pool.ntp.org project.
          # Please consider joining the pool (https://www.pool.ntp.org/join.html).

          # The Machine Config Operator manages this file
          server openshift-master-0.<cluster-name>.<domain> iburst 1
          server openshift-master-1.<cluster-name>.<domain> iburst
          server openshift-master-2.<cluster-name>.<domain> iburst

          stratumweight 0
          driftfile /var/lib/chrony/drift
          rtcsync
          makestep 10 3
          bindcmdaddress 127.0.0.1
          bindcmdaddress ::1
          keyfile /etc/chrony.keys
          commandkey 1
          generatecommandkey
          noclientlog
          logchange 0.5
          logdir /var/log/chrony

          # Configure the control plane nodes to serve as local NTP servers
          # for all worker nodes, even if they are not in sync with an
          # upstream NTP server.

          # Allow NTP client access from the local network.
          allow all
          # Serve time even if not synchronized to a time source.
          local stratum 3 orphan
```

1

<cluster-name>을 클러스터 이름으로 바꾸고 **<domain>**을 정규화된 도메인 이름으로 교체해야 합니다.

2. **Butane**을 사용하여 컨트롤 플레인 노드에 전달할 구성이 포함된 **MachineConfig** 파일 **99-master-chrony-conf-override.yaml**을 생성합니다.

```
$ butane 99-master-chrony-conf-override.bu -o 99-master-chrony-conf-override.yaml
```

3. 컨트롤 플레인 노드의 NTP 서버를 참조하는 작업자 노드의 chrony.conf 파일의 내용을 포함하여 Butane 구성 99-worker-chrony-conf-override.bu를 만듭니다.

Butane 구성의 예

```
variant: openshift
version: 4.9.0
metadata:
  name: 99-worker-chrony-conf-override
  labels:
    machineconfiguration.openshift.io/role: worker
storage:
  files:
    - path: /etc/chrony.conf
      mode: 0644
      overwrite: true
      contents:
        inline: |
          # The Machine Config Operator manages this file.
          server openshift-master-0.<cluster-name>.<domain> iburst 1
          server openshift-master-1.<cluster-name>.<domain> iburst
          server openshift-master-2.<cluster-name>.<domain> iburst

          stratumweight 0
          driftfile /var/lib/chrony/drift
          rtcsync
          makestep 10 3
          bindcmdaddress 127.0.0.1
          bindcmdaddress ::1
          keyfile /etc/chrony.keys
          commandkey 1
          generatecommandkey
          noclientlog
          logchange 0.5
          logdir /var/log/chrony
```

1 <cluster-name>을 클러스터 이름으로 바꾸고 <domain>을 정규화된 도메인 이름으로 교체해야 합니다.

4. Butane을 사용하여 작업자 노드로 전달할 구성이 포함된 MachineConfig 개체 파일 99-

worker-chrony-conf-override.yaml을 생성합니다.

```
$ butane 99-worker-chrony-conf-override.bu -o 99-worker-chrony-conf-override.yaml
```

5. **99-master-chrony-conf-override.yaml** 정책을 컨트롤 플레인 노드에 적용합니다.

```
$ oc apply -f 99-master-chrony-conf-override.yaml
```

출력 예

```
machineconfig.machineconfiguration.openshift.io/99-master-chrony-conf-override
created
```

6. **99-worker-chrony-conf-override.yaml** 정책을 작업자 노드에 적용합니다.

```
$ oc apply -f 99-worker-chrony-conf-override.yaml
```

출력 예

```
machineconfig.machineconfiguration.openshift.io/99-worker-chrony-conf-override
created
```

7. 적용된 **NTP** 설정 상태를 확인합니다.

```
$ oc describe machineconfigpool
```

10.4.2. 설치 후 프로비저닝 네트워크 활성화

베어 메탈 클러스터에 지원되는 설치 프로그램 및 설치 관리자 프로비저닝 설치는 **provisioning** 네트워크 없이 클러스터를 배포하는 기능을 제공합니다. 이 기능은 개념 증명 클러스터 또는 각 노드의 베이스

보드 관리 컨트롤러를 **baremetal** 네트워크를 통해 라우팅할 수 있는 **Redfish** 가상 미디어 전용 배포와 같은 시나리오에 적합합니다.

CBO(Cluster Baremetal Operator)를 사용하여 설치 후 **provisioning** 네트워크를 활성화할 수 있습니다.

사전 요구 사항

- 모든 작업자 및 컨트롤 플레인 노드에 연결된 전용 물리적 네트워크가 있어야 합니다.
- 태그가 지정되지 않은 기본 물리적 네트워크를 분리해야 합니다.
- **provisioningNetwork** 구성 설정이 **Managed**로 설정된 경우 네트워크에 **DHCP** 서버가 있을 수 없습니다.
- **bootMACAddress** 구성 설정을 사용하도록 **OpenShift Container Platform 4.9**에서 **provisioningInterface** 설정을 생략할 수 있습니다.

프로세스

1. **provisioningInterface** 설정을 설정할 때 먼저 클러스터 노드의 프로비저닝 인터페이스 이름을 확인합니다. 예를 들어 **eth0** 또는 **eno1** 입니다.
2. 클러스터 노드의 **provisioning** 네트워크 인터페이스에서 **PXE(Preboot eXecution Environment)**를 활성화합니다.
3. **provisioning** 네트워크의 현재 상태를 검색하여 **provisioning CR(사용자 정의 리소스)** 파일에 저장합니다.

```
$ oc get provisioning -o yaml > enable-provisioning-nw.yaml
```

4. **provisioning CR** 파일을 수정합니다.

```
$ vim ~/enable-provisioning-nw.yaml
```

아래로 스크롤하여 provisioningNetwork 구성 설정으로 이동한 후 Disabled에서 Managed로 변경합니다. 그런 다음 provisioningNetwork 설정 후 provisioningOSDownloadURL, provisioningIP, provisioningNetworkCIDR, provisioningDHCPRange, provisioningInterface, watchAllNameSpaces 구성 설정을 추가합니다. 각 설정에 적절한 값을 제공합니다.

```
apiVersion: v1
items:
- apiVersion: metal3.io/v1alpha1
  kind: Provisioning
  metadata:
    name: provisioning-configuration
  spec:
    provisioningNetwork: 1
    provisioningOSDownloadURL: 2
    provisioningIP: 3
    provisioningNetworkCIDR: 4
    provisioningDHCPRange: 5
    provisioningInterface: 6
    watchAllNameSpaces: 7
```

1

provisioningNetwork는 Managed, Unmanaged 또는 Disabled 중 하나입니다. Managed로 설정하면 Metal3에서 프로비저닝 네트워크를 관리하고 CBO는 구성된 DHCP 서버를 사용하여 Metal3 pod를 배포합니다. Unmanaged로 설정하면 시스템 관리자가 DHCP 서버를 수동으로 구성합니다.

2

provisioningOSDownloadURL은 유효한 sha 256 체크섬이 있는 HTTPS URL로 Metal3 pod에서 .qcow2.gz 또는 .qcow2.xz로 끝나는 qcow2 운영 체제 이미지를 다운로드할 수 있습니다. 이 필드는 프로비저닝 네트워크가 Managed, Unmanaged 또는 Disabled인지 여부에 관계없이 필요합니다. 예: http://192.168.0.1/images/rhcos-*<version>*.x86_64.qcow2.gz?sha256=*<sha>*.

3

provisioningIP는 DHCP 서버와 ironic에서 네트워크를 프로비저닝하는 데 사용하는 고정 IP 주소입니다. 이 고정 IP 주소는 provisioning 서브넷 내에 있어야 하며 DHCP 범위 외부에 있어야 합니다. 이 설정을 구성하는 경우 provisioning 네트워크가 Disabled인 경우에도 유효한 IP 주소가 있어야 합니다. 고정 IP 주소는 metal3 pod에 바인딩됩니다. metal3 pod에 장애가 발생하여 다른 서버로 이동하는 경우 고정 IP 주소도 새 서버로 이동합니다.

4

CIDR(Classless Inter-Domain Routing) 주소입니다. 이 설정을 구성하는 경우 provisioning 네트워크가 Disabled인 경우에도 유효한 CIDR 주소가 있어야 합니다. 예: 192.168.0.1/24

5

DHCP 범위입니다. 이 설정은 **Managed** 프로비저닝 네트워크에만 적용할 수 있습니다. **provisioning** 네트워크가 **Disabled**인 경우 이 구성 설정을 생략합니다. 예: **192.168.0.64, 192.168.0.253.**

6

클러스터 노드의 **provisioning** 인터페이스의 **NIC** 이름입니다. **provisioningInterface** 설정은 **Managed** 및 **Unmanaged** 프로비저닝 네트워크에만 적용할 수 있습니다. **provisioning** 네트워크 가 **Disabled**인 경우 **provisioning Interface** 구성 설정을 생략합니다. **bootMACAddress** 구성 설정을 사용하도록 **provisioningInterface** 구성 설정을 생략합니다.

7

metal3가 기본 **openshift-machine-api** 네임스페이스 이외의 네임스페이스를 감시하도록 하려면 이 설정을 **true**로 설정합니다. 기본값은 **false**입니다.

5. **provisioning CR** 파일에 변경 사항을 저장합니다.

6. **provisioning CR** 파일을 클러스터에 적용합니다.

```
$ oc apply -f enable-provisioning-nw.yaml
```

10.4.3. 외부 로드 밸런서 구성

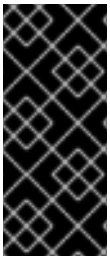
기본 로드 밸런서 대신 외부 로드 밸런서를 사용하도록 **OpenShift Container Platform** 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 로드 밸런서에서 포트 **6443, 443** 및 **80**을 통한 **TCP**는 시스템의 모든 사용자가 사용할 수 있어야 합니다.
- 각 컨트롤 플레인 노드 간에 **API** 포트 **6443**을 로드 밸런싱합니다.
- 모든 컴퓨팅 노드 간에 애플리케이션 포트 **443** 및 **80**을 로드 밸런싱합니다.
-

로드 밸런서에서 노드에 Ignition 시작 구성을 제공하는 데 사용되는 포트 22623은 클러스터 외부에 노출되지 않습니다.

- 로드 밸런서는 클러스터의 모든 머신에 액세스할 수 있어야 합니다. 이러한 액세스를 허용하는 방법은 다음과 같습니다.
  - 클러스터의 머신 서브넷에 로드 밸런서를 연결합니다.
  - 로드 밸런서를 사용하는 머신에 유동 IP 주소를 연결합니다.



중요

외부 로드 밸런싱 서비스와 컨트롤 플레인 노드는 동일한 L2 네트워크에서 실행해야 하며 VLAN을 사용하여 로드 밸런싱 서비스와 컨트롤 플레인 노드 간에 트래픽을 라우팅할 때 동일한 VLAN에서 실행해야 합니다.

절차

1. 포트 6443, 443, 80의 로드 밸런서에서 클러스터에 대한 액세스를 활성화합니다.

   예를 들어 이 HAProxy 구성에 유의하십시오.

   샘플 HAProxy 구성 섹션

```
...
listen my-cluster-api-6443
   bind 0.0.0.0:6443
   mode tcp
   balance roundrobin
   server my-cluster-master-2 192.0.2.2:6443 check
   server my-cluster-master-0 192.0.2.3:6443 check
   server my-cluster-master-1 192.0.2.1:6443 check
listen my-cluster-apps-443
      bind 0.0.0.0:443
      mode tcp
      balance roundrobin
      server my-cluster-worker-0 192.0.2.6:443 check
      server my-cluster-worker-1 192.0.2.5:443 check
      server my-cluster-worker-2 192.0.2.4:443 check
listen my-cluster-apps-80
      bind 0.0.0.0:80
```

```
mode tcp
balance roundrobin
server my-cluster-worker-0 192.0.2.7:80 check
server my-cluster-worker-1 192.0.2.9:80 check
server my-cluster-worker-2 192.0.2.8:80 check
```

2. 클러스터 API의 DNS 서버에 레코드를 추가하고 로드 밸런서를 통해 앱을 추가합니다. 예를 들면 다음과 같습니다.

```
<load_balancer_ip_address> api.<cluster_name>.<base_domain>
<load_balancer_ip_address> apps.<cluster_name>.<base_domain>
```

3. 명령줄에서 curl을 사용하여 외부 로드 밸런서 및 DNS 구성이 작동하는지 확인합니다.

   a. 클러스터 API에 액세스할 수 있는지 확인합니다.

   ```
   $ curl https://<loadbalancer_ip_address>:6443/version --insecure
   ```

   구성이 올바르면 응답으로 JSON 오브젝트가 표시됩니다.

   ```
   {
    "major": "1",
    "minor": "11+",
    "gitVersion": "v1.11.0+ad103ed",
    "gitCommit": "ad103ed",
    "gitTreeState": "clean",
    "buildDate": "2019-01-09T06:44:10Z",
    "goVersion": "go1.10.3",
    "compiler": "gc",
    "platform": "linux/amd64"
   }
   ```

   b. 클러스터 애플리케이션에 액세스할 수 있는지 확인합니다.

   참고

   웹 브라우저에서 OpenShift Container Platform 콘솔을 여는 방식으로 애플리케이션 액세스 가능성을 확인할 수도 있습니다.

```
$ curl http://console-openshift-console.apps.<cluster_name>.<base_domain> -I -L
--insecure
```

구성이 올바르면 **HTTP** 응답이 표시됩니다.

```
HTTP/1.1 302 Found
content-length: 0
location: https://console-openshift-console.apps.<cluster-name>.<base domain>/
cache-control: no-cacheHTTP/1.1 200 OK
referrer-policy: strict-origin-when-cross-origin
set-cookie: csrf-
token=39HoZgztDnzjJkq/JuLJMeoKNXlfiVv2YgZc09c3TBOBU4Nl6kDXaJH1LdicNh
N1UsQWzon4Dor9GWGfopaTEQ==; Path=/; Secure
x-content-type-options: nosniff
x-dns-prefetch-control: off
x-frame-options: DENY
x-xss-protection: 1; mode=block
date: Tue, 17 Nov 2020 08:42:10 GMT
content-type: text/html; charset=utf-8
set-cookie:
1e2670d92730b515ce3a1bb65da45062=9b714eb87e93cf34853e87a92d6894be;
path=/; HttpOnly; Secure; SameSite=None
cache-control: private
```

10.5. 클러스터 확장

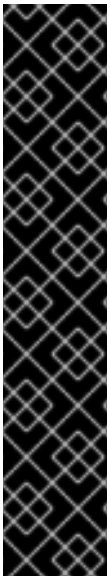
설치 관리자 프로비저닝 **OpenShift Container Platform** 클러스터를 배포한 후 다음 절차를 사용하여 작업자 노드 수를 확장할 수 있습니다. 가능한 각 작업자 노드가 사전 요구 사항을 충족하는지 확인합니다.

참고

RedFish Virtual Media를 사용하여 클러스터를 구성하려면 최소 펌웨어 요구 사항을 충족해야 합니다. **RedFish Virtual Media**를 사용하여 클러스터를 설정할 때 추가 세부 사항은 사전 요구 사항 섹션에서 가상 미디어를 사용하여 설치를 위한 펌웨어 요구 사항을 참조하십시오.

10.5.1. 베어 메탈 노드 준비

클러스터를 확장하려면 **DHCP** 서버가 필요합니다. 각 노드에는 **DHCP** 예약이 있어야 합니다.



고정 IP 주소가 되도록 IP 주소 예약

일부 관리자는 각 노드의 IP 주소가 DHCP 서버에서 일정하게 유지되도록 고정 IP 주소를 사용하는 것을 선호합니다. OpenShift Container Platform 클러스터에서 고정 IP 주소를 사용하려면 무한 리스로 DHCP 서버의 IP 주소를 예약합니다. 설치 프로그램이 노드를 성공적으로 프로비저닝하면 디스패치 스크립트에서 노드의 네트워크 구성을 확인합니다. 디스패치 스크립트에서 네트워크 구성에 DHCP 무한 리스가 포함되어 있음을 발견하면 DHCP 무한 리스에서 IP 주소를 사용하여 고정 IP 연결로 연결을 다시 생성합니다. DHCP 무한 리스가 없는 NIC는 수정되지 않은 상태로 유지됩니다.

무한 리스로 IP 주소를 설정하는 것은 Machine Config Operator를 사용하여 배포된 네트워크 구성과 호환되지 않습니다.

베어 메탈 노드를 준비하려면 프로비저너 노드에서 다음 절차를 실행해야 합니다.

절차

1. 필요한 경우 oc 바이너리를 가져옵니다. 이는 이미 프로비저너 노드에 있어야 합니다.

```
$ curl -s https://mirror.openshift.com/pub/openshift-
v4/clients/ocp/$VERSION/openshift-client-linux-$VERSION.tar.gz | tar zxvf - oc
```

```
$ sudo cp oc /usr/local/bin
```

2. 베이스 보드 관리 컨트롤러를 사용하여 베어 메탈 노드의 전원을 끄고 해제되었는지 확인합니다.

3. 베어 메탈 노드의 베이스 보드 관리 컨트롤러의 사용자 이름 및 암호를 검색합니다. 그런 다음 사용자 이름과 암호에서 base64 문자열을 생성합니다.

```
$ echo -ne "root" | base64
```

```
$ echo -ne "password" | base64
```

4. 베어 메탈 노드에 대한 구성 파일을 생성합니다.

```
$ vim bmh.yaml
```

```
---
```

```
apiVersion: v1
kind: Secret
metadata:
  name: openshift-worker-<num>-bmc-secret
type: Opaque
data:
  username: <base64-of-uid>
  password: <base64-of-pwd>
---
apiVersion: metal3.io/v1alpha1
kind: BareMetalHost
metadata:
  name: openshift-worker-<num>
spec:
  online: true
  bootMACAddress: <NIC1-mac-address>
  bmc:
    address: <protocol>://<bmc-ip>
    credentialsName: openshift-worker-<num>-bmc-secret
```

두 개의 **<num>** 필드와 **credentialsName** 필드에서 베어 메탈 노드의 작업자 번호 **<num>**을 바꿉니다 . **<base64-of-uid>**를 사용자 이름의 **base64** 문자열로 바꿉니다. **<base64-of-pwd>**를 암호의 **base64** 문자열로 바꿉니다. **<NIC1-mac-address>**를 베어 메탈 노드의 첫 번째 **NIC**의 **MAC** 주소로 바꿉니다.

추가 **BMC** 구성 옵션은 **BMC** 주소 지정 섹션을 참조하십시오. **<protocol>**을 **IPMI, RedFish** 또는 기타와 같은 **BMC** 프로토콜로 바꿉니다. **<bmc-ip>**를 베어 메탈 노드의 베이스 보드 관리 컨트롤러의 **IP** 주소로 바꿉니다.

참고

기존 베어 메탈 노드의 **MAC** 주소가 프로비저닝하려는 베어 메탈 호스트의 **MAC** 주소와 일치하면 **Ironic** 설치가 실패합니다. 호스트 등록, 검사, 정리 또는 기타 **Ironic** 단계가 실패하면 **Bare Metal Operator**에서 설치를 지속적으로 다시 시도합니다. 자세한 내용은 호스트 중복 **MAC** 주소 진단을 참조하십시오.

5. 베어 메탈 노드를 생성합니다.

```
$ oc -n openshift-machine-api create -f bmh.yaml
```

```
secret/openshift-worker-<num>-bmc-secret created
baremetalhost.metal3.io/openshift-worker-<num> created
```

여기서 **<num>**은 작업자 번호입니다.

6. 베어 메탈 노드의 전원을 켜고 검사합니다.

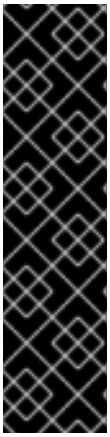
```
$ oc -n openshift-machine-api get bmh openshift-worker-<num>
```

여기서 <num>은 작업자 노드 번호입니다.

```
NAME                 STATE   CONSUMER   ONLINE   ERROR
openshift-worker-<num>  ready           true
```

10.5.2. 베어 메탈 컨트롤 플레인 노드 교체

설치 관리자 프로비저닝 OpenShift Container Platform 컨트롤 플레인 노드를 교체하려면 다음 절차를 사용하십시오.



중요

기존 컨트롤 플레인 호스트에서 BareMetalHost 오브젝트 정의를 재사용하는 경우 external Provisioned 필드를 true 로 설정하지 마십시오.

기존 컨트롤 플레인 BareMetalHost 오브젝트에 OpenShift Container Platform 설치 프로그램에서 프로비저닝한 경우 external Provisioned 플래그가 true 로 설정될 수 있습니다.

사전 요구 사항

- cluster-admin 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- etcd 백업이 수행되었습니다.

중요

문제가 발생하면 클러스터를 복원할 수 있도록 이 절차를 수행하기 전에 etcd 백업을 수행하십시오. etcd 백업에 대한 자세한 내용은 *추가 리소스* 섹션을 참조하십시오.

프로세스

1. Bare Metal Operator를 사용할 수 있는지 확인합니다.

```
$ oc get clusteroperator baremetal
```

출력 예

```
NAME       VERSION   AVAILABLE   PROGRESSING   DEGRADED   SINCE   MESSAGE
baremetal  4.9.0     True        False         False      3d15h
```

2. 이전 BareMetalHost 및 Machine 개체를 제거합니다.

```
$ oc delete bmh -n openshift-machine-api <host_name>
$ oc delete machine -n openshift-machine-api <machine_name>
```

& lt;host_name >을 호스트 이름으로 바꾸고 < machine_name >을 시스템 이름으로 바꿉니다. 머신 이름이 CONSUMER 필드에 표시됩니다.

BareMetalHost 및 Machine 오브젝트를 제거한 후 머신 컨트롤러에서 Node 오브젝트를 자동으로 삭제합니다.

3. BMC 인증 정보를 저장할 새 BareMetalHost 오브젝트와 시크릿을 생성합니다.

```
$ cat <<EOF | oc apply -f -
apiVersion: v1
kind: Secret
metadata:
  name: control-plane-<num>-bmc-secret 1
  namespace: openshift-machine-api
data:
  username: <base64_of_uid> 2
  password: <base64_of_pwd> 3
type: Opaque
---
apiVersion: metal3.io/v1alpha1
kind: BareMetalHost
metadata:
  name: control-plane-<num> 4
  namespace: openshift-machine-api
```

```
spec:
  automatedCleaningMode: disabled
  bmc:
    address: <protocol>://<bmc_ip> (5)
    credentialsName: control-plane-<num>-bmc-secret (6)
  bootMACAddress: <NIC1_mac_address> (7)
  bootMode: UEFI
  externallyProvisioned: false
  hardwareProfile: unknown
  online: true
EOF
```

(1) (4) (6)

name 필드와 credentialsName 필드에서 베어 메탈 노드의 컨트롤 플레인 번호 < num >을 바꿉니다.

(2)

& lt;base64_of_uid& gt;를 사용자 이름의 base64 문자열로 바꿉니다.

(3)

& lt;base64_of_pwd >를 암호의 base64 문자열로 바꿉니다.

(5)

& lt;protocol >을 redfish , redfish -virtualmedia,idrac-virtualmedia 등과 같은 BMC 프로토콜로 바꿉니다. & lt;bmc_ip >를 베어 메탈 노드의 베이스 보드 관리 컨트롤러의 IP 주소로 바꿉니다. 추가 BMC 구성 옵션은 *추가 리소스* 섹션의 "BMC 주소 지정"을 참조하십시오.

(7)

& lt;NIC1_mac_address >를 베어 메탈 노드의 첫 번째 NIC의 MAC 주소로 바꿉니다.

검사가 완료되면 BareMetalHost 오브젝트가 생성되고 프로비저닝할 수 있습니다.

4. 사용 가능한 BareMetalHost 오브젝트를 확인합니다.

```
$ oc get bmh -n openshift-machine-api
```

출력 예

```
NAME                          STATE                  CONSUMER          ONLINE   ERROR   AGE
control-plane-1.example.com   available              control-plane-1   true             1h10m
control-plane-2.example.com   externally provisioned control-plane-2   true             4h53m
control-plane-3.example.com   externally provisioned control-plane-3   true             4h53m
compute-1.example.com         provisioned            compute-1-ktmmx   true             4h53m
compute-1.example.com         provisioned            compute-2-l2zmb   true             4h53m
```

컨트롤 플레인 노드에 대한 **MachineSet** 오브젝트가 없으므로 대신 **Machine** 오브젝트를 생성해야 합니다. 다른 컨트롤 플레인 머신 오브젝트에서 **providerSpec** 을 복사할 수 있습니다.

5. **Machine** 오브젝트를 생성합니다.

```
$ cat <<EOF | oc apply -f -
apiVersion: machine.openshift.io/v1beta1
kind: Machine
metadata:
  annotations:
    metal3.io/BareMetalHost: openshift-machine-api/control-plane-<num> 1
  labels:
    machine.openshift.io/cluster-api-cluster: control-plane-<num> 2
    machine.openshift.io/cluster-api-machine-role: master
    machine.openshift.io/cluster-api-machine-type: master
  name: control-plane-<num> 3
  namespace: openshift-machine-api
spec:
  metadata: {}
  providerSpec:
    value:
      apiVersion: baremetal.cluster.k8s.io/v1alpha1
      customDeploy:
        method: install_coreos
      hostSelector: {}
      image:
        checksum: ""
        url: ""
      kind: BareMetalMachineProviderSpec
      metadata:
        creationTimestamp: null
      userData:
        name: master-user-data-managed
EOF
```

1 2 3

이름 ,라벨 및 주석 필드에서 베어 메탈 노드의 컨트롤 플레인 번호 <num >을 바꿉니다.

6. BareMetalHost 오브젝트를 보려면 다음 명령을 실행합니다.

```
$ oc get bmh -A
```

출력 예

```
NAME                 STATE                  CONSUMER             ONLINE   ERROR   AGE
control-plane-1.example.com   provisioned            control-plane-1      true
2h53m
control-plane-2.example.com   externally provisioned   control-plane-2      true
5h53m
control-plane-3.example.com   externally provisioned   control-plane-3      true
5h53m
compute-1.example.com         provisioned            compute-1-ktmmx      true
5h53m
compute-2.example.com         provisioned            compute-2-l2zmb      true
5h53m
```

7. RHCOS 설치 후 BareMetalHost 가 클러스터에 추가되었는지 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME                 STATUS   ROLES    AGE   VERSION
control-plane-1.example.com   available   master   4m2s  v1.18.2
control-plane-2.example.com   available   master   141m  v1.18.2
control-plane-3.example.com   available   master   141m  v1.18.2
compute-1.example.com         available   worker   87m   v1.18.2
compute-2.example.com         available   worker   87m   v1.18.2
```

참고

새 컨트롤 플레인 노드를 교체한 후 새 노드에서 실행중인 **etcd pod**는 크래시 루프 상태에 있습니다. 자세한 내용은 *추가 리소스* 섹션의 "**앓던 etcd 멤버 교체**"를 참조하십시오.

추가 리소스

- 비정상적인 **etcd** 멤버 교체
- **etcd** 백업
- **BMC** 주소 지정

10.5.3. 가상 네트워크에서 가상 미디어를 사용하여 배포 준비

provisioning 네트워크가 활성화되어 있고 **baremetal** 네트워크에서 가상 미디어를 사용하여 클러스터를 확장하려면 다음 절차를 사용하십시오.

사전 요구 사항

- **baremetal** 네트워크와 **provisioning** 네트워크가 있는 기존 클러스터가 있습니다.

프로세스

1. **provisioning CR**(사용자 정의 리소스)을 편집하여 **baremetal** 네트워크에서 가상 미디어를 사용하여 배포할 수 있습니다.

```
oc edit provisioning
```

```
apiVersion: metal3.io/v1alpha1
kind: Provisioning
metadata:
  creationTimestamp: "2021-08-05T18:51:50Z"
  finalizers:
  - provisioning.metal3.io
  generation: 8
  name: provisioning-configuration
  resourceVersion: "551591"
  uid: f76e956f-24c6-4361-aa5b-feaf72c5b526
```

```
  spec:
    preProvisioningOSDownloadURLs: {}
    provisioningDHCPRange: 172.22.0.10,172.22.0.254
    provisioningIP: 172.22.0.3
    provisioningInterface: enp1s0
    provisioningNetwork: Managed
    provisioningNetworkCIDR: 172.22.0.0/24
    provisioningOSDownloadURL: http://192.168.111.1/images/rhcos-
<version>.x86_64.qcow2.gz?sha256=<sha256>
    virtualMediaViaExternalNetwork: true (1)
  status:
    generations:
    - group: apps
      hash: ""
      lastGeneration: 7
      name: metal3
      namespace: openshift-machine-api
      resource: deployments
    - group: apps
      hash: ""
      lastGeneration: 1
      name: metal3-image-cache
      namespace: openshift-machine-api
      resource: daemonsets
    observedGeneration: 8
    readyReplicas: 0
```

(1) provisioning CR에 virtualœViaExternalNetwork: true를 추가합니다.

2. API VIP 주소를 사용하도록 머신 세트를 편집합니다.

```
oc edit machineset
```

```
  apiVersion: machine.openshift.io/v1beta1
  kind: MachineSet
  metadata:
    creationTimestamp: "2021-08-05T18:51:52Z"
    generation: 11
    labels:
      machine.openshift.io/cluster-api-cluster: ostest-hwmdt
      machine.openshift.io/cluster-api-machine-role: worker
      machine.openshift.io/cluster-api-machine-type: worker
    name: ostest-hwmdt-worker-0
    namespace: openshift-machine-api
    resourceVersion: "551513"
    uid: fad1c6e0-b9da-4d4a-8d73-286f78788931
  spec:
    replicas: 2
    selector:
```

```
      matchLabels:
        machine.openshift.io/cluster-api-cluster: ostest-hwmdt
        machine.openshift.io/cluster-api-machineset: ostest-hwmdt-worker-0
    template:
      metadata:
        labels:
          machine.openshift.io/cluster-api-cluster: ostest-hwmdt
          machine.openshift.io/cluster-api-machine-role: worker
          machine.openshift.io/cluster-api-machine-type: worker
          machine.openshift.io/cluster-api-machineset: ostest-hwmdt-worker-0
      spec:
        metadata: {}
        providerSpec:
          value:
            apiVersion: baremetal.cluster.k8s.io/v1alpha1
            hostSelector: {}
            image:
              checksum: http:/172.22.0.3:6181/images/rhcos-<version>.x86_64.qcow2.
<md5sum> (1)
              url: http://172.22.0.3:6181/images/rhcos-<version>.x86_64.qcow2 (2)
            kind: BareMetalMachineProviderSpec
            metadata:
              creationTimestamp: null
            userData:
              name: worker-user-data
    status:
      availableReplicas: 2
      fullyLabeledReplicas: 2
      observedGeneration: 11
      readyReplicas: 2
      replicas: 2
```

(1) API VIP 주소를 사용하도록 checksum URL을 편집합니다.

(2) API VIP 주소를 사용하도록 url URL을 편집합니다.

10.5.4. 클러스터에서 새 호스트를 프로비저닝할 때 중복된 MAC 주소 진단

클러스터에 있는 기존 베어 메탈 노드의 MAC 주소가 클러스터에 추가하려는 베어 메탈 호스트의 MAC 주소와 일치하는 경우 베어 메탈 Operator는 기존 노드와 호스트를 연결합니다. 호스트 등록, 검사, 정리 또는 기타 Ironic 단계가 실패하면 Bare Metal Operator에서 설치를 지속적으로 다시 시도합니다. 실패한 베어 메탈 호스트에 대한 등록 오류가 표시됩니다.

openshift-machine-api 네임스페이스에서 실행 중인 베어 메탈 호스트를 검사하여 중복된 MAC 주소를 진단할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 베어 메탈에 설치합니다.
- OpenShift Container Platform CLI oc를 설치합니다.
- cluster-admin 권한이 있는 사용자로 로그인합니다.

프로세스

프로비저닝에 실패하는 베어 메탈 호스트에 기존 노드와 동일한 MAC 주소가 있는지 확인하려면 다음을 수행하십시오.

1. openshift-machine-api 네임스페이스에서 베어 메탈 호스트를 실행합니다.

```
$ oc get bmh -n openshift-machine-api
```

출력 예

```
NAME                 STATUS   PROVISIONING STATUS      CONSUMER
openshift-master-0   OK       externally provisioned   openshift-zpwpq-master-0
openshift-master-1   OK       externally provisioned   openshift-zpwpq-master-1
openshift-master-2   OK       externally provisioned   openshift-zpwpq-master-2
openshift-worker-0   OK       provisioned              openshift-zpwpq-worker-0-lv84n
openshift-worker-1   OK       provisioned              openshift-zpwpq-worker-0-zd8lm
openshift-worker-2   error    registering
```

2. 장애가 발생한 호스트의 상태에 대한 자세한 정보를 보려면 다음 명령을 실행하여 <bare_metal_host_name>을 호스트 이름으로 교체합니다.

```
$ oc get -n openshift-machine-api bmh <bare_metal_host_name> -o yaml
```

출력 예

```
...
status:
  errorCount: 12
  errorMessage: MAC address b4:96:91:1d:7c:20 conflicts with existing node
openshift-worker-1
  errorType: registration error
...
```

10.5.5. 베어 메탈 노드 프로비저닝

베어 메탈 노드를 프로비저닝하려면 프로비저너 노드에서 다음 절차를 실행해야 합니다.

프로세스

1. 베어 메탈 노드를 프로비저닝하기 전에 **STATE** 가 준비 되었는지 확인합니다.

```
$  oc -n openshift-machine-api get bmh openshift-worker-<num>
```

여기서 **<num>**은 작업자 노드 번호입니다.

```
NAME                 STATE   CONSUMER    ONLINE   ERROR
openshift-worker-<num>   ready             true
```

2. 작업자 노드 수를 계산합니다.

```
$ oc get nodes
```

```
NAME                                 STATUS   ROLES         AGE     VERSION
provisioner.openshift.example.com              Ready    master        30h     v1.22.1
openshift-master-1.openshift.example.com       Ready    master        30h     v1.22.1
openshift-master-2.openshift.example.com       Ready    master        30h     v1.22.1
openshift-master-3.openshift.example.com       Ready    master        30h     v1.22.1
openshift-worker-0.openshift.example.com       Ready    master        30h     v1.22.1
openshift-worker-1.openshift.example.com       Ready    master        30h     v1.22.1
```

3. 머신 세트를 가져옵니다.

-

```
$ oc get machinesets -n openshift-machine-api
```

```
NAME                          DESIRED   CURRENT   READY   AVAILABLE   AGE
...
openshift-worker-0.example.com   1         1         1       1           55m
openshift-worker-1.example.com   1         1         1       1           55m
```

4. 작업자 노드 수를 하나씩 늘립니다.

```
$ oc scale --replicas=<num> machineset <machineset> -n openshift-machine-api
```

<num>을 새 작업자 노드 수로 바꿉니다. <machineset>를 이전 단계의 머신 세트 이름으로 바꿉니다.

5. 베어 메탈 노드 상태를 확인합니다.

```
$ oc -n openshift-machine-api get bmh openshift-worker-<num>
```

여기서 <num>은 작업자 노드 번호입니다. STATE가 준비 상태에서 프로비저닝 으로 변경됩니다.

```
NAME                 STATE          CONSUMER                    ONLINE   ERROR
openshift-worker-<num>  provisioning     openshift-worker-<num>-65tjz     true
```

provisioning 상태는 OpenShift Container Platform 클러스터가 노드를 프로비저닝할 때까지 유지됩니다. 이 작업을 수행하는 데 30분 이상 걸릴 수 있습니다. 노드를 프로비저닝하면 상태가 provisioned 로 변경됩니다.

```
NAME                 STATE          CONSUMER                    ONLINE   ERROR
openshift-worker-<num>  provisioning     openshift-worker-<num>-65tjz     true
```

6. 프로비저닝이 완료되면 베어 메탈 노드가 준비되었는지 확인합니다.

```
$ oc get nodes
```

```
NAME                                STATUS   ROLES   AGE    VERSION
provisioner.openshift.example.com          Ready    master  30h    v1.22.1
openshift-master-1.openshift.example.com     Ready    master  30h    v1.22.1
openshift-master-2.openshift.example.com     Ready    master  30h    v1.22.1
openshift-master-3.openshift.example.com     Ready    master  30h    v1.22.1
```

```
openshift-worker-0.openshift.example.com     Ready    master  30h    v1.22.1
openshift-worker-1.openshift.example.com     Ready    master  30h    v1.22.1
openshift-worker-<num>.openshift.example.com Ready    worker  3m27s  v1.22.1
```

kubelet도 확인할 수 있습니다.

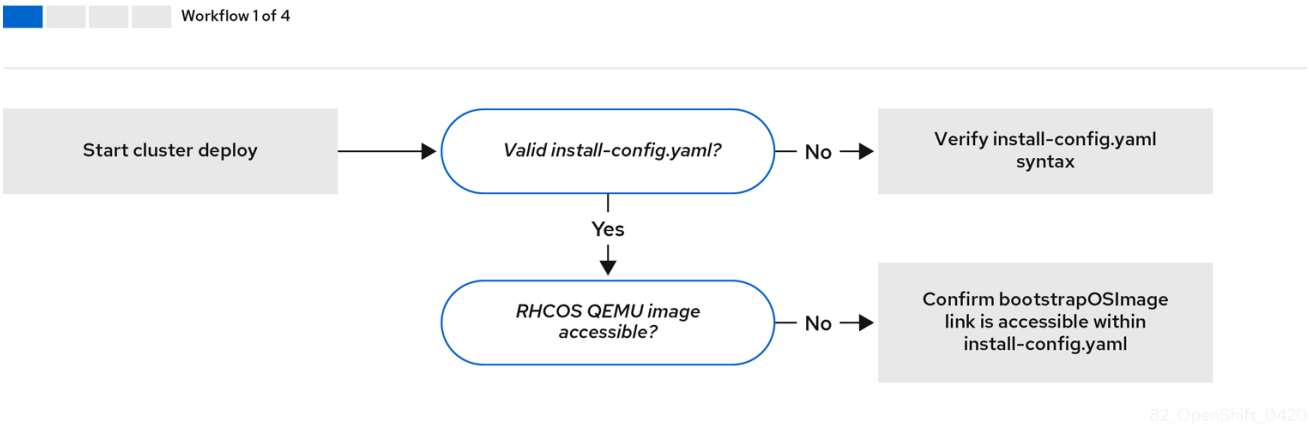
```
$ ssh openshift-worker-<num>
```

```
[kni@openshift-worker-<num>]$ journalctl -fu kubelet
```

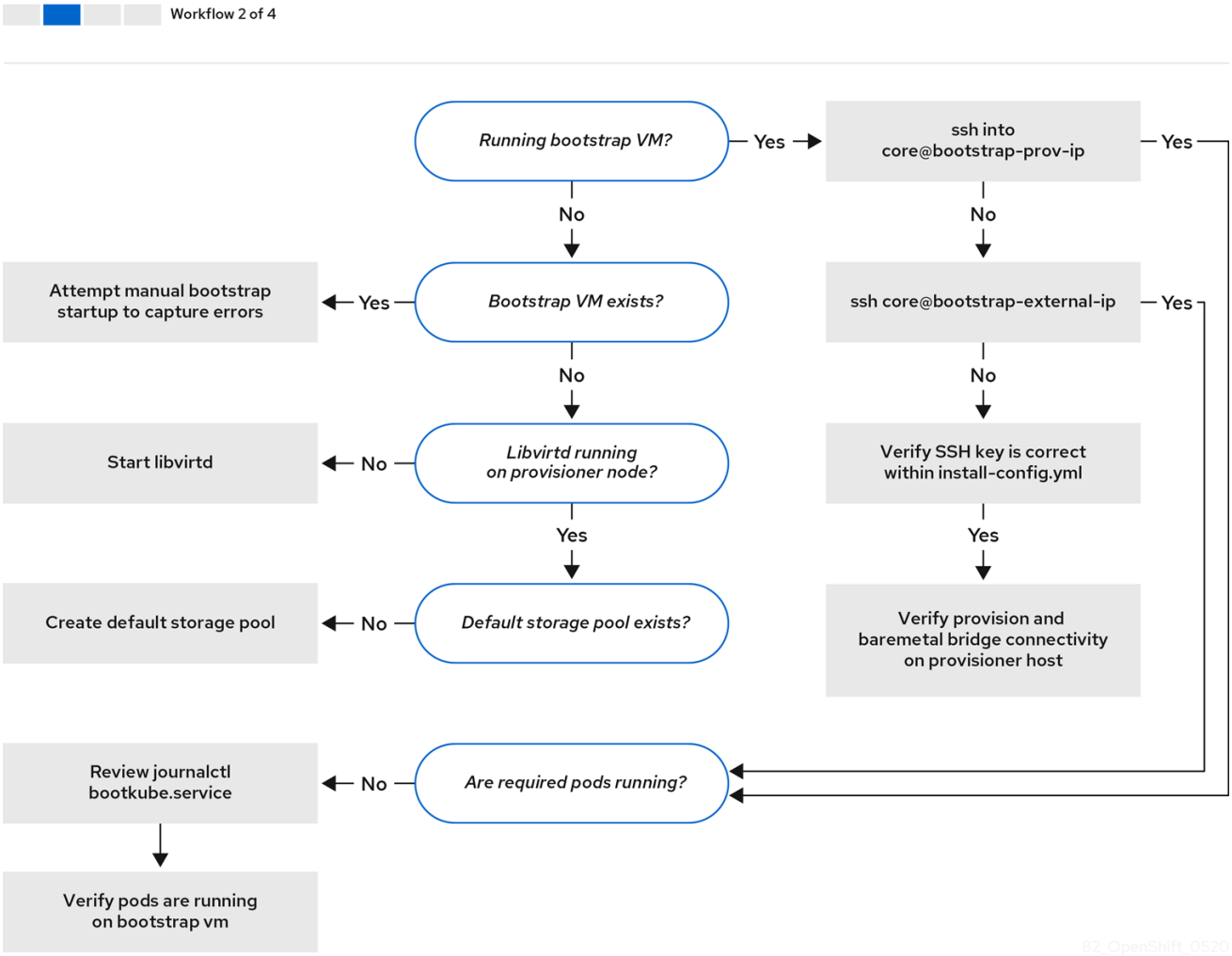
10.6. 문제 해결

10.6.1. 설치 프로그램 워크 플로우 문제 해결

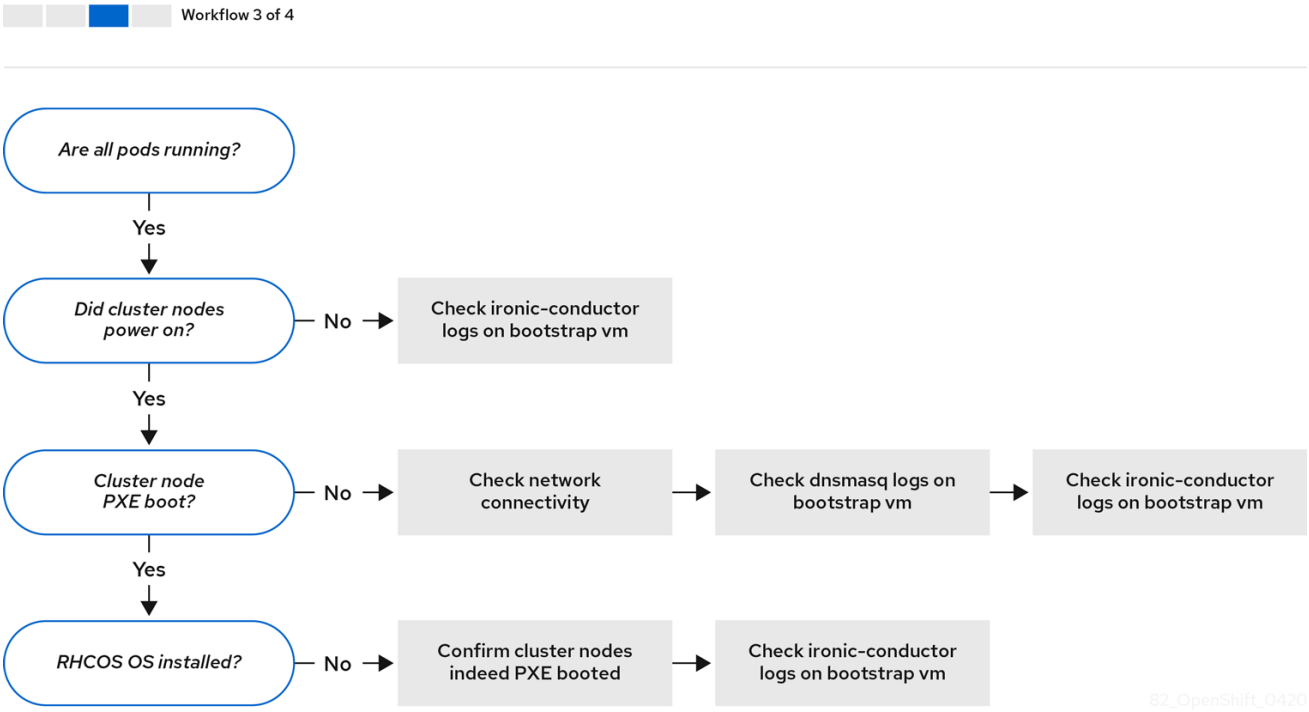
설치 환경의 문제를 해결하기 전에 베어 메탈에서 설치 프로그램이 제공하는 설치의 전체 흐름을 이해하는 것이 중요합니다. 아래 다이어그램은 환경에서 단계별 분석과 함께 문제 해결 흐름을 보여줍니다.



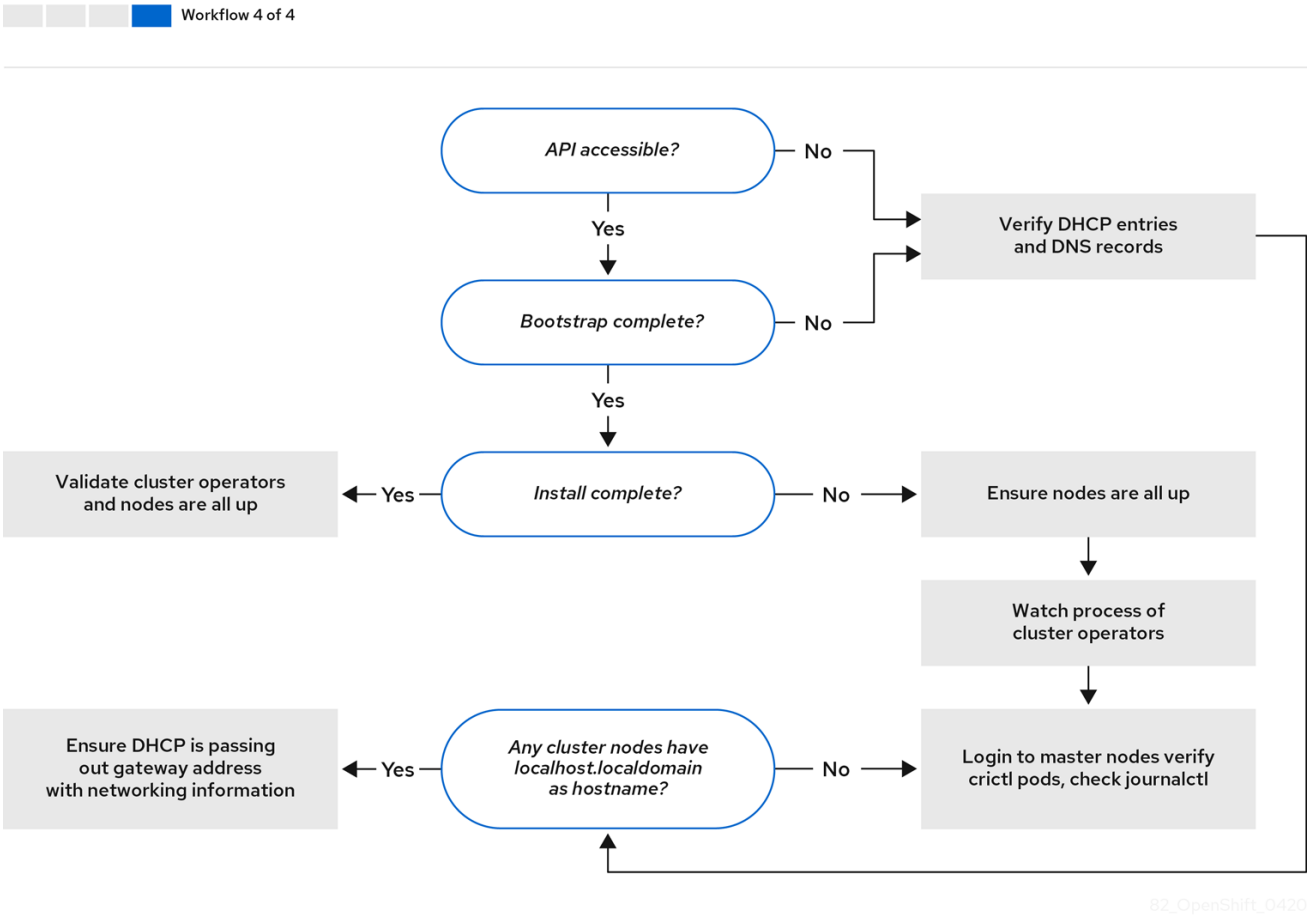
*워크 플로우 1/4*은 **install-config.yaml** 파일에 오류가 있거나 **RHCOS (Red Hat Enterprise Linux CoreOS)** 이미지에 액세스할 수 없는 경우 문제 해결의 워크 플로우를 보여줍니다. 문제 해결에 대한 제안은 **Troubleshooting install-config.yaml**에서 참조하십시오.



*워크 플로우 2/4*는 부트스트랩 VM 문제, 클러스터 노드를 부팅할 수 없는 부트스트랩 VM 및 로그 검사에 대한 문제 해결 워크 플로우를 보여줍니다. **provisioning** 네트워크없이 **OpenShift Container Platform** 클러스터를 설치할 때 이 워크플로는 적용되지 않습니다.



*워크 플로우 3/4*은 **PXE 부팅이 되지 않는 클러스터 노드**에 대한 문제 해결의 워크 플로우를 보여줍니다. **RedFish Virtual Media**를 사용하여 설치하는 경우 각 노드가 노드를 배포하는 데 필요한 최소 펌웨어 요구 사항을 충족해야 합니다. 자세한 내용은 사전 요구 사항 섹션에서 가상 미디어를 사용하여 설치를 위한 펌웨어 요구 사항을 참조하십시오.



*워크 플로우 4/4*는 액세스할 수 없는 **API**부터 검증된 설치까지의 문제 해결 워크 플로우를 보여줍니다.

10.6.2. install-config.yaml 문제 해결

install-config.yaml 설정 파일은 **OpenShift Container Platform** 클러스터의 일부인 모든 노드를 나타냅니다. 이 파일에는 **apiVersion, baseDomain, imageContentSources** 및 가상 **IP** 주소로 구성되지만 이에 국한되지 않는 필수 옵션이 포함되어 있습니다. **OpenShift Container Platform** 클러스터 배포 초기에 오류가 발생하면 **install-config.yaml** 구성 파일에 오류가 있을 수 있습니다.

절차

1. YAML-tips의 지침을 사용합니다.

2. syntax-check를 사용하여 **YAML** 구문이 올바른지 확인합니다.

3. **RHCOS (Red Hat Enterprise Linux CoreOS) QEMU** 이미지가 올바르게 정의되어 있고 **install-config.yaml**에서 제공되는 **URL**을 통해 액세스할 수 있는지 확인합니다. 예를 들면 다음

과 같습니다.

```
$ curl -s -o /dev/null -I -w "%{http_code}\n" http://webserver.example.com:8080/rhcos-44.81.202004250133-0-qemu.x86_64.qcow2.gz?sha256=7d884b46ee54fe87bbc3893bf2aa99af3b2d31f2e19ab5529c60636fbd0f1ce7
```

출력이 200이면 부트스트랩 VM 이미지를 저장하는 웹 서버의 유효한 응답이 있습니다.

10.6.3. 부트스트랩 VM 문제

OpenShift Container Platform 설치 프로그램은 OpenShift Container Platform 클러스터 노드 프로비저닝을 처리하는 부트스트랩 노드 가상 머신을 생성합니다.

절차

1. 설치 프로그램을 트리거한 후 약 10~15분 후에 virsh 명령을 사용하여 부트스트랩 VM이 작동하는지 확인합니다.

```
$ sudo virsh list
```

```
 Id    Name                           State
----------------------------------------------------
 12    openshift-xf6fq-bootstrap      running
```

참고

부트스트랩 VM의 이름은 항상 클러스터 이름으로 시작하여 그 뒤에 임의의 문자 집합이 있고 "bootstrap"이라는 단어로 끝납니다.

부트스트랩 VM이 10-15 분 후에도 실행되지 않으면 실행되지 않은 이유에 대한 문제를 해결합니다. 발생 가능한 문제는 다음과 같습니다.

2. libvirtd가 시스템에서 실행 중인지 확인합니다.

```
$ systemctl status libvirtd
```

```
● libvirtd.service - Virtualization daemon
   Loaded: loaded (/usr/lib/systemd/system/libvirtd.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2020-03-03 21:21:07 UTC; 3 weeks 5 days ago
```

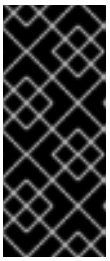
```
     Docs: man:libvirtd(8)
           https://libvirt.org
 Main PID: 9850 (libvirtd)
    Tasks: 20 (limit: 32768)
   Memory: 74.8M
   CGroup: /system.slice/libvirtd.service
           ├─ 9850 /usr/sbin/libvirtd
```

부트스트랩 VM이 작동하는 경우 로그인하십시오.

3. virsh console 명령을 사용하여 부트스트랩 VM의 IP 주소를 찾습니다.

```
$ sudo virsh console example.com
```

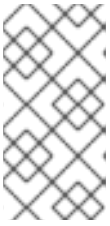
```
Connected to domain example.com
Escape character is ^]
Red Hat Enterprise Linux CoreOS 43.81.202001142154.0 (Ootpa) 4.3
SSH host key: SHA256:BRWJktXZgQQRY5zjuAV0IKZ4WM7i4TiUyMVanqu9Pqg
(ED25519)
SSH host key: SHA256:7+iKGA7VtG5szmk2jB5gl/5EZ+SNcJ3a2g23o0lnlio (ECDSA)
SSH host key: SHA256:DH5VWhvhvagOTaLsYiVNse9ca+ZSW/30OOMed8rIGOc (RSA)
ens3:  fd35:919d:4042:2:c7ed:9a9f:a9ec:7
ens4: 172.22.0.2 fe80::1d05:e52e:be5d:263f
localhost login:
```



중요

provisioning 네트워크 없이 OpenShift Container Platform 클러스터를 배포하는 경우 172.22.0.2 와 같은 개인 IP 주소가 아닌 공용 IP 주소를 사용해야 합니다.

4. IP 주소를 가져온 후 ssh 명령을 사용하여 부트스트랩 VM에 로그인합니다.



참고

이전 단계의 콘솔 출력에서 ens3에서 제공되는 IPv6 IP 주소 또는 ens4에서 제공되는 IPv4 IP를 사용할 수 있습니다.

```
$ ssh core@172.22.0.2
```

부트스트랩 VM에 성공적으로 로그인하지 못한 경우 다음 시나리오 중 하나가 발생했을 가능성이 있습니다.

- 172.22.0.0/24 네트워크에 연결할 수 없습니다. 프로비저너와 provisioning 네트워크 브리지 간의 네트워크 연결을 확인합니다. 이 문제는 프로비저닝 네트워크를 사용하는 경우 발생할 수 있습니다.

- 공용 네트워크를 통해 부트스트랩 VM에 연결할 수 없습니다. baremetal 네트워크에서 SSH를 시도할 때 provisioner 호스트, 특히 baremetal 네트워크 브리지의 연결을 확인합니다.

- Permission denied (publickey, password, keyboard-interactive)문제가 발생했습니다. 부트스트랩 VM에 액세스하려고하면 Permission denied 오류가 발생할 수 있습니다. VM에 로그인하려는 사용자의 SSH 키가 install-config.yaml 파일에 설정되어 있는지 확인합니다.

10.6.3.1. 부트스트랩 VM은 클러스터 노드를 부팅할 수 없습니다.

배포 중에 부트스트랩 VM이 클러스터 노드를 부팅하지 못하여 VM이 RHCOS 이미지로 노드를 프로비저닝하지 못할 수 있습니다. 이 시나리오는 다음과 같은 이유로 발생할 수 있습니다.

- install-config.yaml 파일 관련 문제

- baremetal 네트워크를 사용할 때 대역 외 네트워크 액세스 문제

이 문제를 확인하기 위해 ironic과 관련된 세 가지 컨테이너를 사용할 수 있습니다.

- ironic-api

- ironic-conductor

- ironic-inspector

프로세스

1. 부트스트랩 VM에 로그인합니다.

```
$ ssh core@172.22.0.2
```

2. 컨테이너 로그를 확인하려면 다음을 실행합니다.

```
[core@localhost ~]$ sudo podman logs -f <container-name>
```

<container-name>을 ironic-api, ironic-conductor 또는 ironic-inspector 중 하나로 바꿉니다. 컨트롤 플레인 노드가 PXE를 통해 부팅되지 않는 문제가 발생하면 ironic-conductor Pod를 확인하십시오. ironic-conductor Pod는 IPMI를 통해 노드에 로그인을 시도하기 때문에 클러스터 노드 부팅 시도에 대한 가장 자세한 정보가 포함되어 있습니다.

가능한 이유

배포가 시작되면 클러스터 노드가 ON 상태 일 수 있습니다.

해결책

IPMI를 통해 설치를 시작하기 전에 OpenShift Container Platform 클러스터 노드의 전원을 끄십시오.

```
$ ipmitool -I lanplus -U root -P <password> -H <out-of-band-ip> power off
```

10.6.3.2. 로그 검사

RHCOS 이미지를 다운로드하거나 액세스하는 데 문제가 발생하면 먼저 install-config.yaml 구성 파일에서 URL이 올바른지 확인합니다.

RHCOS 이미지를 호스팅하는 내부 웹 서버의 예

```
bootstrapOSImage: http://<ip:port>/rhcos-43.81.202001142154.0-qemu.x86_64.qcow2.gz?
sha256=9d999f55ff1d44f7ed7c106508e5deecd04dc3c06095d34d36bf1cd127837e0c
clusterOSImage: http://<ip:port>/rhcos-43.81.202001142154.0-openstack.x86_64.qcow2.gz?
sha256=a1bda656fa0892f7b936fdc6b6a6086bddaed5dafacedcd7a1e811abb78fe3b0
```

ipa-downloader 및 coreos-downloader 컨테이너는 install-config.yaml 구성 파일에 지정된 웹 서버 또는 외부 quay.io 레지스트리에서 리소스를 다운로드합니다. 다음 두 컨테이너가 실행 중인지 확인하

고 필요에 따라 로그를 검사합니다.

- ipa-downloader
- coreos-downloader

프로세스

1. 부트스트랩 VM에 로그인합니다.

```
$ ssh core@172.22.0.2
```

2. 부트스트랩 VM 내에서 ipa-downloader 및 coreos-downloader 컨테이너의 상태를 확인합니다.

```
[core@localhost ~]$ sudo podman logs -f ipa-downloader
```

```
[core@localhost ~]$ sudo podman logs -f coreos-downloader
```

부트스트랩 VM이 이미지의 URL에 액세스할 수 없는 경우 curl 명령을 사용하여 VM이 이미지에 액세스할 수 있는지 확인합니다.

3. 배포 단계에서 모든 컨테이너가 시작되었는지 여부를 나타내는 bootkube 로그를 검사하려면 다음을 실행합니다.

```
[core@localhost ~]$ journalctl -xe
```

```
[core@localhost ~]$ journalctl -b -f -u bootkube.service
```

4. dnsmasq, mariadb, httpd 및 ironic를 포함한 모든 Pod가 실행 중인지 확인합니다.

```
[core@localhost ~]$ sudo podman ps
```

5. Pod에 문제가 있는 경우 문제가있는 컨테이너의 로그를 확인합니다. ironic-api의 로그를 확인하려면 다음을 실행합니다.

```
[core@localhost ~]$ sudo podman logs <ironic-api>
```

10.6.4. 클러스터 노드는 PXE 부팅 불가능

OpenShift Container Platform 클러스터 노드가 PXE 부팅을 하지 않는 경우 PXE 부팅이 되지 않는 클러스터 노드에서 다음 검사를 실행합니다. 이 절차는 provisioning 네트워크없이 OpenShift Container Platform 클러스터를 설치할 때 적용되지 않습니다.

프로세스

1. provisioning 네트워크에 대한 네트워크 연결을 확인하십시오.

2. provisioning 네트워크의 NIC에서 PXE가 활성화되어 있고 다른 모든 NIC에 대해 PXE가 비활성화되어 있는지 확인합니다.

3. install-config.yaml 구성 파일에 provisioning 네트워크에 연결된 NIC에 대한 적절한 하드웨어 프로필과 부팅 MAC 주소가 있는지 확인합니다. 예를 들면 다음과 같습니다.

컨트롤 플레인 노드 설정

```
bootMACAddress: 24:6E:96:1B:96:90 # MAC of bootable provisioning NIC
hardwareProfile: default          #control plane node settings
```

작업자 노드 설정

```
bootMACAddress: 24:6E:96:1B:96:90 # MAC of bootable provisioning NIC
hardwareProfile: unknown          #worker node settings
```

10.6.5. API에 액세스 불가능

클러스터가 실행 중이고 클라이언트가 API에 액세스할 수 없는 경우 도메인 이름 확인 문제가 API에

대한 액세스를 방해할 수 있습니다.

프로세스

1. 호스트 이름 확인: 클러스터 노드에 localhost.localdomain 뿐만 아니라 완전한 도메인 이름이 있는지 확인합니다. 예를 들면 다음과 같습니다.

```
$ hostname
```

호스트 이름이 설정되지 않은 경우 올바른 호스트 이름을 설정합니다. 예를 들면 다음과 같습니다.

```
$ hostnamectl set-hostname <hostname>
```

2. 잘못된 이름 확인: dig 및 nslookup을 사용하여 DNS 서버에서 각 노드의 이름을 올바르게 확인할 수 있는지 확인합니다. 예를 들면 다음과 같습니다.

```
$ dig api.<cluster-name>.example.com
```

```
; <<>> DiG 9.11.4-P2-RedHat-9.11.4-26.P2.el8 <<>> api.<cluster-name>.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 37551
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 866929d2f8e8563582af23f05ec44203d313e50948d43f60 (good)
;; QUESTION SECTION:
;api.<cluster-name>.example.com. IN A

;; ANSWER SECTION:
api.<cluster-name>.example.com. 10800 IN A 10.19.13.86

;; AUTHORITY SECTION:
<cluster-name>.example.com. 10800 IN NS <cluster-name>.example.com.

;; ADDITIONAL SECTION:
<cluster-name>.example.com. 10800 IN A 10.19.14.247

;; Query time: 0 msec
;; SERVER: 10.19.14.247#53(10.19.14.247)
;; WHEN: Tue May 19 20:30:59 UTC 2020
;; MSG SIZE  rcvd: 140
```

이 예의 출력은 api. <cluster-name> .example.com VIP의 적절한 IP 주소가 10.19.13.86임

을 보여줍니다. 이 IP 주소는 baremetal 네트워크에 있어야합니다.

10.6.6. 이전 설치 정리

이전 배포에 실패한 경우 OpenShift Container Platform을 다시 배포하기 전에 실패한 시도에서 아티팩트를 제거합니다.

프로세스

1. OpenShift Container Platform 클러스터를 설치하기 전에 모든 베어 메탈 노드의 전원을 끕니다.

```
$ ipmitool -I lanplus -U <user> -P <password> -H <management-server-ip> power off
```

2. 이전 배포에서 남은 모든 이전 부트스트랩 리소스를 제거합니다.

```
for i in $(sudo virsh list | tail -n +3 | grep bootstrap | awk {'print $2'});
do
  sudo virsh destroy $i;
  sudo virsh undefine $i;
  sudo virsh vol-delete $i --pool $i;
  sudo virsh vol-delete $i.ign --pool $i;
  sudo virsh pool-destroy $i;
  sudo virsh pool-undefine $i;
done
```

3. Terraform이 실패하지 않도록 clusterconfigs 디렉터리에서 다음을 제거합니다.

```
$ rm -rf ~/clusterconfigs/auth ~/clusterconfigs/terraform* ~/clusterconfigs/tls
~/clusterconfigs/metadata.json
```

10.6.7. 레지스트리 생성 문제

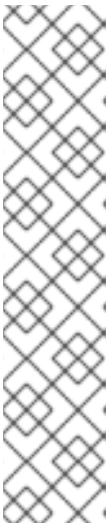
비 연결 레지스트리를 만들 때 레지스트리 미러링을 시도하는 경우 "User Not Authorized" 오류가 발생할 수 있습니다. 이 오류는 기존 pull-secret.txt 파일에 새 인증을 추가할 수 없는 경우 발생할 수 있습니다.

프로세스

1. 인증이 성공했는지 확인합니다.

-

```
$ /usr/local/bin/oc adm release mirror \
  -a pull-secret-update.json
  --from=$UPSTREAM_REPO \
  --to-release-image=$LOCAL_REG/$LOCAL_REPO:${VERSION} \
  --to=$LOCAL_REG/$LOCAL_REPO
```



참고

설치 이미지 미러링에 사용되는 변수의 출력 예:

```
UPSTREAM_REPO=${RELEASE_IMAGE}
LOCAL_REG=<registry_FQDN>:<registry_port>
LOCAL_REPO='ocp4/openshift4'
```

RELEASE_IMAGE 및 **VERSION**의 값은 **OpenShift** 설치 환경 설정 섹션의 **OpenShift** 설치 프로그램 가져오기 단계에서 설정됩니다.

2. 레지스트리를 미러링 후 연결이 끊긴 환경에서 이에 액세스할 수 있는지 확인합니다.

```
$ curl -k -u <user>:<password> https://registry.example.com:<registry-
port>/v2/_catalog
{"repositories":["<Repo-Name>"]}
```

10.6.8. 기타 문제

10.6.8.1. runtime network not ready 오류 해결

클러스터 배포 후 다음과 같은 오류가 발생할 수 있습니다.

```
`runtime network not ready: NetworkReady=false reason:NetworkPluginNotReady message:Network
plugin returns error: Missing CNI default network`
```

Cluster Network Operator 설치 프로그램이 생성한 특수 개체에 대응하여 네트워킹 구성 요소를 배포해야 합니다. 컨트롤 플레인 (마스터) 노드가 시작된 후 부트스트랩 컨트롤 플레인이 중지되기 전에 설치 프로세스 초기에 실행됩니다. 컨트롤 플레인 (마스터) 노드를 시작할 때 오랜 지연이나 **apiserver** 통신 문제와 같은 미묘한 설치 프로그램 문제가 표시될 수 있습니다.

프로세스

1. **openshift-network-operator** 네임 스페이스에서 **Pod**를 검사합니다.

```
$ oc get all -n openshift-network-operator
```

```
NAME                                READY STATUS          RESTARTS   AGE
pod/network-operator-69dfd7b577-bg89v   0/1   ContainerCreating 0          149m
```

2. provisioner 노드에서 네트워크 구성이 존재하는지 확인합니다.

```
$ kubectl get network.config.openshift.io cluster -oyaml
```

```
apiVersion: config.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  serviceNetwork:
  - 172.30.0.0/16
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  networkType: OpenShiftSDN
```

존재하지 않는 경우 설치 프로그램이 이를 생성하지 않은 것입니다. 설치 프로그램이 생성하지 않은 이유를 확인하려면 다음을 실행합니다.

```
$ openshift-install create manifests
```

3. network-operator가 실행되고 있는지 확인합니다.

```
$ kubectl -n openshift-network-operator get pods
```

4. 로그를 검색합니다.

```
$ kubectl -n openshift-network-operator logs -l "name=network-operator"
```

3 개 이상의 컨트롤 플레인 (마스터) 노드가 있는 고가용성 클러스터에서 Operator는 리더의 선택을 실행하고 다른 Operator는 절전 모드로 전환합니다. 자세한 내용은 Troubleshooting을 참조하십시오.

10.6.8.2. DHCP를 통해 올바른 IPv6 주소를 얻지 못하는 클러스터 노드

클러스터 노드가 DHCP를 통해 올바른 IPv6 주소를 얻지 못하는 경우 다음을 확인합니다.

1. 예약 된 IPv6 주소가 DHCP 범위 밖에 있는지 확인합니다.

2. DHCP 서버의 IP 주소 예약에서 예약에 올바른 DHCP 고유 식별자 (DUID)가 지정되어 있는지 확인합니다. 예를 들면 다음과 같습니다.

```
# This is a dnsmasq dhcp reservation, 'id:00:03:00:01' is the client id and
'18:db:f2:8c:d5:9f' is the MAC Address for the NIC
id:00:03:00:01:18:db:f2:8c:d5:9f,openshift-master-1,[2620:52:0:1302::6]
```

3. 경로 알림 (Route Announcement)이 제대로 작동하는지 확인합니다.

4. DHCP 서버가 IP 주소 범위를 제공하는 데 필요한 인터페이스에서 수신하고 있는지 확인합니다.

10.6.8.3. DHCP를 통해 올바른 호스트 이름을 얻지 못하는 클러스터 노드

IPv6 배포 중에 클러스터 노드는 DHCP를 통해 호스트 이름을 검색해야 합니다. 경우에 따라 NetworkManager가 호스트 이름을 즉시 할당하지 않을 수 있습니다. 컨트롤 플레인 (마스터) 노드는 다음과 같은 오류를 보고할 수 있습니다.

```
Failed Units: 2
  NetworkManager-wait-online.service
  nodeip-configuration.service
```

이 오류는 클러스터 노드가 DHCP 서버에서 호스트 이름을 받지 않고 부팅되었을 가능성이 있음을 나타냅니다. 이로 인해 kubelet이 localhost.localdomain 호스트 이름으로 부팅됩니다. 이 문제를 해결하려면 노드가 호스트 이름을 업데이트하도록 합니다.

프로세스

1. hostname을 검색합니다.

```
[core@master-X ~]$ hostname
```

   호스트 이름이 localhost인 경우 다음 단계를 진행합니다.

참고

여기서 **X**는 컨트롤 플레인 노드 번호입니다.

2. 클러스터 노드가 **DHCP** 임대를 갱신하도록 합니다.

```
[core@master-X ~]$ sudo nmcli con up "<bare-metal-nic>"
```

<bare-metal-nic>을 **baremetal** 네트워크에 해당하는 유선 연결로 바꿉니다.

3. **hostname** 다시 확인하십시오.

```
[core@master-X ~]$ hostname
```

4. 호스트 이름이 여전히 **localhost.localdomain**인 경우 **NetworkManager**를 다시 시작합니다.

```
[core@master-X ~]$ sudo systemctl restart NetworkManager
```

5. 호스트 이름이 여전히 **localhost.localdomain** 인 경우 몇 분 기다린 후 다시 확인하십시오. 호스트 이름이 **localhost.localdomain**으로 남아 있으면 이전 단계를 반복합니다.

6. **nodeip-configuration** 서비스를 다시 시작합니다.

```
[core@master-X ~]$ sudo systemctl restart nodeip-configuration.service
```

이 서비스는 올바른 호스트 이름 참조를 사용하여 **kubelet** 서비스를 재구성합니다.

7. 이전 단계에서 **kubelet**이 변경되었으므로 단위 파일 정의를 다시 로드하십시오.

```
[core@master-X ~]$ sudo systemctl daemon-reload
```

8. **kubelet** 서비스를 다시 시작합니다.

```
[core@master-X ~]$ sudo systemctl restart kubelet.service
```

9. kubelet 이 올바른 호스트 이름으로 부팅되었는지 확인합니다.

```
[core@master-X ~]$ sudo journalctl -fu kubelet.service
```

클러스터 가동 후 (예: 클러스터를 다시 시작) 클러스터 노드가 DHCP를 통해 올바른 호스트 이름을 얻지 못하는 경우 클러스터에 csr은 보류 처리됩니다. csr을 승인 하지 마십시오. 그렇지 않으면 다른 문제가 발생할 수 있습니다.

CSR 처리

1. 클러스터에서 CSR을 가져옵니다.

```
$ oc get csr
```

2. 보류중인 CSR 에 Subject Name: localhost.localdomain이 포함되어 있는지 확인합니다.

```
$ oc get csr <pending_csr> -o jsonpath='{.spec.request}' | base64 --decode | openssl
req -noout -text
```

3. Subject Name: localhost.localdomain이 포함된 모든 csr을 제거합니다.

```
$ oc delete csr <wrong_csr>
```

10.6.8.4. 루트가 엔드 포인트에 도달하지 않음

설치 프로세스 중에 VRRP (Virtual Router Redundancy Protocol) 충돌이 발생할 수 있습니다. 특정 클러스터 이름을 사용하여 클러스터 배포의 일부였던 이전에 사용된 OpenShift Container Platform 노드가 여전히 실행 중이지만 동일한 클러스터 이름을 사용하는 현재 OpenShift Container Platform 클러스터 배포의 일부가 아닌 경우 이러한 충돌이 발생할 수 있습니다. 예를 들어 클러스터는 클러스터 이름 openshift를 사용하여 3 개의 컨트롤 플레인 (마스터) 노드와 3 개의 작업자 노드를 배포합니다. 나중에 다른 설치에서 동일한 클러스터 이름 openshift를 사용하지만 이 재배포에서는 3 개의 컨트롤 플레인 (마스터) 노드 만 설치하여 이전 배포의 작업자 노드 3 개를 ON 상태로 유지합니다. 이로 인해 VRID (Virtual Router Identifier) 충돌 및 VRRP 충돌이 발생할 수 있습니다.

1. 루트를 가져옵니다.

```
$ oc get route oauth-openshift
```

2. 서비스 엔드 포인트를 확인합니다.

```
$ oc get svc oauth-openshift
```

```
NAME              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
oauth-openshift   ClusterIP   172.30.19.162   <none>        443/TCP   59m
```

3. 컨트롤 플레인 (마스터) 노드에서 서비스에 연결을 시도합니다.

```
[core@master0 ~]$ curl -k https://172.30.19.162
```

```
{
  "kind": "Status",
  "apiVersion": "v1",
  "metadata": {
  },
  "status": "Failure",
  "message": "forbidden: User \"system:anonymous\" cannot get path \"/\"",
  "reason": "Forbidden",
  "details": {
  },
  "code": 403
```

4. provisioner 노드에서 authentication-operator 오류를 식별합니다.

```
$ oc logs deployment/authentication-operator -n openshift-authentication-operator
```

```
Event(v1.ObjectReference{Kind:"Deployment", Namespace:"openshift-authentication-
operator", Name:"authentication-operator", UID:"225c5bd5-b368-439b-9155-
5fd3c0459d98", APIVersion:"apps/v1", ResourceVersion:"", FieldPath:""}): type:
'Normal' reason: 'OperatorStatusChanged' Status for clusteroperator/authentication
changed: Degraded message changed from "IngressStateEndpointsDegraded: All 2
endpoints for oauth-server are reporting"
```

해결책

1. 모든 배포의 클러스터 이름이 고유한지 확인하여 충돌이 발생하지 않도록합니다.

2. 동일한 클러스터 이름을 사용하는 클러스터 배포의 일부가 아닌 잘못된 노드 모두를 종료합니다. 그렇지 않으면 OpenShift Container Platform 클러스터의 인증 pod가 정상적으로 시작되지 않을 수 있습니다.

10.6.8.5. Firstboot 동안 Ignition 실패

Firstboot 중에 Ignition 설정이 실패할 수 있습니다.

프로세스

1. Ignition 설정이 실패한 노드에 연결합니다.

```
Failed Units: 1
  machine-config-daemon-firstboot.service
```

2. machine-config-daemon-firstboot 서비스를 다시 시작합니다.

```
[core@worker-X ~]$ sudo systemctl restart machine-config-daemon-firstboot.service
```

10.6.8.6. NTP가 동기화되지 않음

OpenShift Container Platform 클러스터를 배포하려면 클러스터 노드 간의 NTP 시계가 동기화되어야합니다. 동기화된 시계가 없으면 시간 차이가 2 초보다 크면 클럭 드리프트로 인해 배포 실패할 수 있습니다.

프로세스

1. 클러스터 노드의 AGE 차이를 확인하십시오. 예를 들면 다음과 같습니다.

```
$ oc get nodes
```

```
NAME                         STATUS   ROLES    AGE   VERSION
master-0.cloud.example.com   Ready    master   145m   v1.22.1
master-1.cloud.example.com   Ready    master   135m   v1.22.1
master-2.cloud.example.com   Ready    master   145m   v1.22.1
worker-2.cloud.example.com   Ready    worker   100m   v1.22.1
```

2. 클럭 드리프트로 인한 일관성없는 시간 지연을 확인하십시오. 예를 들면 다음과 같습니다.

```
$ oc get bmh -n openshift-machine-api
```

```
master-1   error registering master-1  ipmi://<out-of-band-ip>
```

```
$ sudo timedatectl
```

```
               Local time: Tue 2020-03-10 18:20:02 UTC
```

```
               Universal time: Tue 2020-03-10 18:20:02 UTC
                     RTC time: Tue 2020-03-10 18:36:53
                    Time zone: UTC (UTC, +0000)
System clock synchronized: no
                  NTP service: active
              RTC in local TZ: no
```

기존 클러스터에서 클럭 드리프트 처리

1. 노드에 전송할 **chrony.conf** 파일의 내용을 포함하여 **Butane** 구성 파일을 만듭니다. 다음 예제에서 **99-master-chrony.bu**를 생성하여 파일을 컨트롤 플레인 노드에 추가합니다. 작업자 노드의 파일을 변경하거나 작업자 역할에 대해 이 절차를 반복할 수 있습니다.

참고

Butane에 대한 자세한 내용은 **"Butane** 을 사용하여 머신 구성 생성**"**을 참조하십시오.

```
variant: openshift
version: 4.9.0
metadata:
  name: 99-master-chrony
  labels:
    machineconfiguration.openshift.io/role: master
storage:
  files:
  - path: /etc/chrony.conf
    mode: 0644
    overwrite: true
    contents:
      inline: |
        server <NTP-server> iburst (1)
        stratumweight 0
        driftfile /var/lib/chrony/drift
        rtcsync
        makestep 10 3
        bindcmdaddress 127.0.0.1
        bindcmdaddress ::1
        keyfile /etc/chrony.keys
        commandkey 1
        generatecommandkey
        noclientlog
        logchange 0.5
        logdir /var/log/chrony
```

(1) **<NTP-server>**를 **NTP** 서버의 **IP** 주소로 바꿉니다.

2. Butane을 사용하여 노드에 전달할 구성이 포함된 MachineConfig 파일 99-master-chrony.yaml을 생성합니다.

```
$ butane 99-master-chrony.bu -o 99-master-chrony.yaml
```

3. MachineConfig 오브젝트를 적용합니다.

```
$ oc apply -f 99-master-chrony.yaml
```

4. System clock synchronized 값이 yes 인지 확인하십시오.

```
$ sudo timedatectl
```

```
               Local time: Tue 2020-03-10 19:10:02 UTC
           Universal time: Tue 2020-03-10 19:10:02 UTC
                 RTC time: Tue 2020-03-10 19:36:53
                Time zone: UTC (UTC, +0000)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

배포 전에 클럭 동기화를 설정하려면 매니페스트 파일을 생성하고이 파일을 openshift 디렉터리에 추가합니다. 예를 들면 다음과 같습니다.

```
$ cp chrony-masters.yaml ~/clusterconfigs/openshift/99_masters-chrony-configuration.yaml
```

그런 다음 계속해서 클러스터를 만듭니다.

10.6.9. 설치 확인

설치 후 설치 프로그램이 노드 및 pod를 성공적으로 배포했는지 확인합니다.

프로세스

1. OpenShift Container Platform 클러스터 노드가 적절하게 설치되면 STATUS 열에 Ready 상태가 표시됩니다.

```
$ oc get nodes
```

```
NAME                 STATUS   ROLES         AGE  VERSION
master-0.example.com   Ready    master,worker   4h   v1.22.1
master-1.example.com   Ready    master,worker   4h   v1.22.1
master-2.example.com   Ready    master,worker   4h   v1.22.1
```

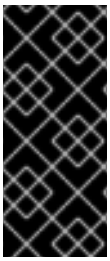
2. 설치 프로그램이 모든 **pod**를 성공적으로 배포했는지 확인합니다. 다음 명령은 아직 실행 중이거나 출력의 일부로 완료된 모든 **pod**를 제거합니다.

```
$ oc get pods --all-namespaces | grep -iv running | grep -iv complete
```

11장. IBM CLOUD에 설치 프로그램 프로비저닝 클러스터 배포

11.1. 사전 요구 사항

설치 관리자 프로비저닝 설치를 사용하여 IBM Cloud® 노드에 OpenShift Container Platform을 설치할 수 있습니다. 이 문서에서는 IBM Cloud 노드에 OpenShift Container Platform을 설치할 때 사전 요구 사항 및 절차를 설명합니다.



중요

Red Hat은 provisioning 네트워크에서만 IPMI 및 PXE를 지원합니다. Red Hat은 IBM Cloud 배포 시 Secure Boot와 같은 Red Fish, 가상 미디어 또는 기타 보완 기술을 테스트하지 않았습니다. 프로비저닝 네트워크가 필요합니다.

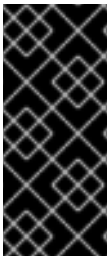
OpenShift Container Platform 설치 프로그램으로 프로비저닝된 설치에는 다음이 필요합니다.

- RHCOS(Red Hat Enterprise Linux CoreOS) 8.x가 설치된 provisioner 노드 1개
- 컨트롤 플레인 노드 세 개
- 라우팅 가능한 네트워크 1개
- 노드 프로비저닝용 네트워크 1개

IBM Cloud에서 OpenShift Container Platform의 설치 관리자 프로비저닝 설치를 시작하기 전에 다음 사전 요구 사항 및 요구 사항을 해결합니다.

11.1.1. IBM Cloud 인프라 설정

IBM Cloud®에 OpenShift Container Platform 클러스터를 배포하려면 먼저 IBM Cloud 노드를 프로비저닝해야 합니다.



중요

Red Hat은 provisioning 네트워크에서만 IPMI 및 PXE를 지원합니다. Red Hat은 IBM Cloud 배포 시 Secure Boot와 같은 Red Fish, 가상 미디어 또는 기타 보완 기술을 테스트하지 않았습니다. 프로비저닝 네트워크가 필요합니다.

IBM Cloud API를 사용하여 IBM Cloud 노드를 사용자 지정할 수 있습니다. IBM Cloud 노드를 생성할 때 다음 요구 사항을 고려해야 합니다.

클러스터당 데이터 센터 1개 사용

OpenShift Container Platform 클러스터의 모든 노드는 동일한 IBM Cloud 데이터 센터에서 실행해야 합니다.

공용 및 개인 VLAN 만들기

단일 공용 VLAN 및 단일 개인 VLAN을 사용하여 모든 노드를 생성합니다.

서브넷에 충분한 IP 주소가 있는지 확인하십시오

IBM Cloud 공용 VLAN 서브넷에서는 기본적으로 16개의 IP 주소를 제공하는 /28 접두사를 사용합니다. 베어 메탈 네트워크의 API VIP 및 Ingress VIP에 대한 세 개의 컨트롤 플레인 노드, 작업자 노드 4개, IP 주소 2개로 구성된 클러스터에 충분합니다. 대규모 클러스터의 경우 더 작은 접두사가 필요할 수 있습니다.

IBM Cloud 프라이빗 VLAN 서브넷에서는 기본적으로 64개의 IP 주소를 제공하는 /26 접두사를 사용합니다. IBM Cloud는 사설 네트워크 IP 주소를 사용하여 각 노드의 BMC(Baseboard Management Controller)에 액세스합니다. OpenShift Container Platform은 provisioning 네트워크에 대한 추가 서브넷을 생성합니다 . 사설 VLAN을 통해 프로비저닝 네트워크 서브넷 경로에 대한 네트워크 트래픽. 대규모 클러스터의 경우 더 작은 접두사가 필요할 수 있습니다.

표 11.1. 접두사당 IP 주소

IP 주소	접두사
32	/27
64	/26
128	/25
256	/24

NIC 설정

OpenShift Container Platform은 다음 두 가지 네트워크를 사용하여 배포합니다.

- **provisioning: provisioning** 네트워크는 **OpenShift Container Platform** 클러스터의 일부인 각 노드에서 기본 운영 체제를 프로비저닝하는 데 사용되는 라우팅 불가능한 네트워크입니다.

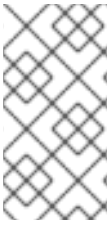
- **baremetal: baremetal** 네트워크는 라우팅 가능한 네트워크입니다. **provisioning NetworkInterface** 구성 설정에 지정된 **NIC** 또는 **provisioning** 네트워크의 **bootMACAddress** 구성 설정에 연결된 **NIC**가 아닌 경우 **baremetal** 네트워크와 상호 작용할 수 있습니다.

클러스터 노드에는 두 개 이상의 **NIC**가 포함될 수 있지만 설치 프로세스는 처음 두 개의 **NIC**에만 중점을 둡니다. 예를 들면 다음과 같습니다.

NIC	네트워크	VLAN
NIC1	**provisioning**	<provisioning_vlan>
NIC2	**baremetal**	<baremetal_vlan>

이전 예에서 모든 컨트롤 플레인 및 작업자 노드의 **NIC1**은 **OpenShift Container Platform** 클러스터 설치에만 사용되는 라우팅 불가능한 네트워크 (프로비저닝)에 연결됩니다. 모든 컨트롤 플레인 및 작업자 노드의 **NIC2**는 라우팅 가능한 **baremetal** 네트워크에 연결됩니다.

PXE	부팅 순서
NIC1 PXE 지원 **provisioning** 네트워크	1
NIC2 **baremetal** 네트워크	2



참고

provisioning 네트워크에 사용된 **NIC**에서 **PXE**가 활성화되어 있고 다른 모든 **NIC**에서 비활성화되어 있는지 확인합니다.

정식 이름 구성

클라이언트는 **baremetal** 네트워크를 통해 **OpenShift Container Platform** 클러스터 노드에 액세스합니다. 정식 이름 확장이 클러스터 이름인 하위 영역 또는 **IBM Cloud** 하위 도메인을 구성합니다.

```
<cluster_name>.<domain>
```

예를 들면 다음과 같습니다.

```
test-cluster.example.com
```

DNS 항목 생성

다음을 위해 퍼블릭 서브넷에서 사용되지 않는 IP 주소를 확인하는 DNS A 레코드 항목을 생성해야 합니다.

사용법	호스트 이름	IP
API	api.<cluster_name>.<domain>	<ip>
Ingress LB (apps)	*.apps.<cluster_name>.<domain>	<ip>

컨트롤 플레인 및 작업자 노드에는 프로비저닝 후 이미 DNS 항목이 있습니다.

다음 테이블에서는 정규화된 도메인 이름의 예를 제공합니다. API 및 Nameserver 주소는 표준 이름 확장으로 시작됩니다. 컨트롤 플레인 및 작업자 노드의 호스트 이름은 예제이므로 원하는 호스트 이름 지정 규칙을 사용할 수 있습니다.

사용법	호스트 이름	IP
API	api.<cluster_name>.<domain>	<ip>
Ingress LB (apps)	*.apps.<cluster_name>.<domain>	<ip>
Provisioner node	provisioner.<cluster_name>.<domain>	<ip>
Master-0	openshift-master-0.<cluster_name>.<domain>	<ip>
Master-1	openshift-master-1.<cluster_name>.<domain>	<ip>
Master-2	openshift-master-2.<cluster_name>.<domain>	<ip>

사용법	호스트 이름	IP
Worker-0	openshift-worker-0. <cluster_name>.<domain>	<ip>
Worker-1	openshift-worker-1. <cluster_name>.<domain>	<ip>
Worker-n	openshift-worker-n. <cluster_name>.<domain>	<ip>

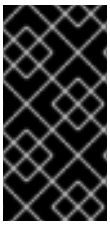
OpenShift Container Platform에는 클러스터 멤버십 정보를 사용하여 **A** 레코드를 생성하는 기능이 포함되어 있습니다. 이렇게 하면 노드 이름이 해당 **IP** 주소로 확인됩니다. 노드가 **API**에 등록되면 클러스터에서 **CoreDNS-mDNS**를 사용하지 않고 노드 정보를 분산할 수 있습니다. 그러면 멀티캐스트 **DNS**와 연결된 네트워크 트래픽이 제거됩니다.

중요

IBM Cloud 노드를 프로비저닝한 후 **CoreDNS**를 제거하면 로컬 항목이 사라지기 때문에 외부 **DNS**에서 **api.<cluster_name>.<domain>** 도메인 이름에 대한 **DNS** 항목을 생성해야 합니다. 외부 **DNS** 서버의 **api.<cluster_name>.<domain>** 도메인 이름에 대한 **DNS** 레코드를 생성하지 않으면 작업자 노드가 클러스터에 참여하지 못하게 합니다.

Network Time Protocol (NTP)

클러스터의 각 **OpenShift Container Platform** 노드는 **NTP** 서버에 액세스할 수 있습니다. **OpenShift Container Platform** 노드는 **NTP**를 사용하여 클럭을 동기화합니다. 예를 들어 클러스터 노드는 검증이 필요한 **SSL** 인증서를 사용하므로 노드 간 날짜와 시간이 동기화되지 않은 경우 인증서가 실패할 수 있습니다.



중요

각 클러스터 노드의 **BIOS** 설정에서 일관된 클럭 날짜 및 시간 형식을 정의하지 않으면 설치에 실패할 수 있습니다.

DHCP 서버 설정

IBM Cloud는 공용 또는 개인 **VLAN**에서 **DHCP**를 실행하지 않습니다. **IBM Cloud** 노드를 프로비저닝한 후 **OpenShift Container Platform**의 **baremetal** 네트워크에 해당하는 공용 **VLAN**의 **DHCP** 서버를 설정해야 합니다.

참고

각 노드에 할당된 IP 주소는 IBM Cloud 프로비저닝 시스템에서 할당한 IP 주소와 일치하지 않아도 됩니다.

자세한 내용은 "공용 서브넷 구성" 섹션을 참조하십시오.

BMC 액세스 권한 확인

대시보드의 각 노드의 "원격 관리" 페이지에는 노드의 IPMI(Intelligent Platform Management Interface) 자격 증명이 포함되어 있습니다. 기본 IPMI 권한을 사용하면 사용자가 특정 부팅 대상을 변경할 수 없습니다. Ironic에서 이러한 변경을 수행할 수 있도록 권한 수준을 OPERATOR 로 변경해야 합니다.

install-config.yaml 파일에서 각 BMC를 구성하는 데 사용되는 URL에 privilegelevel 매개변수를 추가합니다. 자세한 내용은 "install-config.yaml 파일 구성" 섹션을 참조하십시오. 예를 들면 다음과 같습니다.

```
ipmi://<IP>:<port>?privilegelevel=OPERATOR
```

또는 IBM Cloud 지원 센터에 문의하여 각 노드의 IPMI 권한을 ADMINISTRATOR 로 증가하도록 요청합니다.

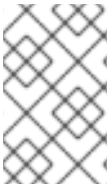
베어 메탈 서버 생성

리소스 → 베어 메탈 서버 만들기로 이동하여 IBM Cloud 대시보드에 베어 메탈 서버를 만듭니다.

또는 ibmcloud CLI 유틸리티를 사용하여 베어 메탈 서버를 생성할 수도 있습니다. 예를 들면 다음과 같습니다.

```
$ ibmcloud sl hardware create --hostname <SERVERNAME> \
                    --domain <DOMAIN> \
                    --size <SIZE> \
                    --os <OS-TYPE> \
                    --datacenter <DC-NAME> \
                    --port-speed <SPEED> \
                    --billing <BILLING>
```

IBM Cloud CLI 설치에 대한 자세한 내용은 독립 실행형 IBM Cloud CLI 설치를 참조하십시오.



참고

IBM 클라우드 서버를 사용할 수 있는 데 3~5시간이 걸릴 수 있습니다.

11.2. OPENSHIFT CONTAINER PLATFORM 설치를 위한 환경 설정

11.2.1. IBM Cloud에 OpenShift Container Platform 설치를 위한 프로비저너 노드 준비

다음 단계를 수행하여 프로비저너 노드를 준비합니다.

프로세스

1. ssh를 통해 프로비저너 노드에 로그인합니다.

2. root가 아닌 사용자 (kni)를 만들고 해당 사용자에게 sudo 권한을 부여합니다.

```
# useradd kni
```

```
# passwd kni
```

```
# echo "kni ALL=(root) NOPASSWD:ALL" | tee -a /etc/sudoers.d/kni
```

```
# chmod 0440 /etc/sudoers.d/kni
```

3. 새 사용자에 대한 ssh 키를 만듭니다.

```
# su - kni -c "ssh-keygen -f /home/kni/.ssh/id_rsa -N ""
```

4. 프로비저너 노드에서 새 사용자로 로그인합니다.

```
# su - kni
```

5. Red Hat Subscription Manager를 사용하여 프로비저닝 노드를 등록합니다.

```
$ sudo subscription-manager register --username=<user> --password=<pass> --auto-attach
```

```
$ sudo subscription-manager repos --enable=rhel-8-for-x86_64-appstream-rpms \
                          --enable=rhel-8-for-x86_64-baseos-rpms
```

참고

Red Hat Subscription Manager에 대한 자세한 내용은 Using and Configuring Red Hat Subscription Manager에서 참조하십시오.

6. 다음 패키지를 설치합니다.

```
$ sudo dnf install -y libvirt qemu-kvm mkisofs python3-devel jq ipmitool
```

7. 사용자를 변경하여 libvirt 그룹을 새로 만든 사용자에 추가합니다.

```
$ sudo usermod --append --groups libvirt kni
```

8. firewalld 시작 :

```
$ sudo systemctl start firewalld
```

9. firewalld 활성화 :

```
$ sudo systemctl enable firewalld
```

10. http 서비스를 시작합니다.

```
$ sudo firewall-cmd --zone=public --add-service=http --permanent
```

```
$ sudo firewall-cmd --reload
```

11. libvirtd 서비스를 시작하고 활성화합니다.

```
$ sudo systemctl enable libvirtd --now
```

12. 프로비저너 노드의 ID를 설정합니다.

```
$ PRVN_HOST_ID=<ID>
```

다음 **ibmcloud** 명령을 사용하여 **ID**를 볼 수 있습니다.

```
$ ibmcloud sl hardware list
```

13. 공용 서브넷의 **ID**를 설정합니다.

```
$ PUBLICSUBNETID=<ID>
```

다음 **ibmcloud** 명령을 사용하여 **ID**를 볼 수 있습니다.

```
$ ibmcloud sl subnet list
```

14. 사설 서브넷의 **ID**를 설정합니다.

```
$ PRIVSUBNETID=<ID>
```

다음 **ibmcloud** 명령을 사용하여 **ID**를 볼 수 있습니다.

```
$ ibmcloud sl subnet list
```

15. 프로비저너 노드 공용 **IP** 주소를 설정합니다.

```
$ PRVN_PUB_IP=$(ibmcloud sl hardware detail $PRVN_HOST_ID --output JSON | jq
.primaryIpAddress -r)
```

16. 공용 네트워크의 **CIDR**을 설정합니다.

```
$ PUBLICCIDR=$(ibmcloud sl subnet detail $PUBLICSUBNETID --output JSON | jq
.cidr)
```

17. 공용 네트워크의 **IP** 주소 및 **CIDR**을 설정합니다.

```
$ PUB_IP_CIDR=$PRVN_PUB_IP/$PUBLICCIDR
```

18. **public** 네트워크의 게이트웨이를 설정합니다.

```
$ PUB_GATEWAY=$(ibmcloud sl subnet detail $PUBLICSUBNETID --output JSON | jq
.gateway -r)
```

19. 프로비저너 노드의 개인 **IP** 주소를 설정합니다.

```
$ PRVN_PRIV_IP=$(ibmcloud sl hardware detail $PRVN_HOST_ID --output JSON | \
          jq .primaryBackendIpAddress -r)
```

20. 사설 네트워크의 **CIDR**을 설정합니다.

```
$ PRIVCIDR=$(ibmcloud sl subnet detail $PRIVSUBNETID --output JSON | jq .cidr)
```

21. 사설 네트워크의 **IP** 주소 및 **CIDR**을 설정합니다.

```
$ PRIV_IP_CIDR=$PRVN_PRIV_IP/$PRIVCIDR
```

22. 사설 네트워크의 게이트웨이를 설정합니다.

```
$ PRIV_GATEWAY=$(ibmcloud sl subnet detail $PRIVSUBNETID --output JSON | jq
.gateway -r)
```

23. **baremetal** 및 **provisioning** 네트워크의 브리지를 설정합니다.

```
$ sudo nohup bash -c "
   nmcli --get-values UUID con show | xargs -n 1 nmcli con delete
   nmcli connection add ifname provisioning type bridge con-name provisioning
   nmcli con add type bridge-slave ifname eth1 master provisioning
   nmcli connection add ifname baremetal type bridge con-name baremetal
   nmcli con add type bridge-slave ifname eth2 master baremetal
   nmcli connection modify baremetal ipv4.addresses $PUB_IP_CIDR ipv4.method
manual ipv4.gateway $PUB_GATEWAY
   nmcli connection modify provisioning ipv4.addresses 172.22.0.1/24,$PRIV_IP_CIDR
ipv4.method manual
   nmcli connection modify provisioning +ipv4.routes \"10.0.0.0/8 $PRIV_GATEWAY\"
   nmcli con down baremetal
   nmcli con up baremetal
   nmcli con down provisioning
   nmcli con up provisioning
   init 6
"
```

참고

필요에 따라 **eth1** 및 **eth2** 의 경우 적절한 인터페이스 이름을 대체합니다.

24. 필요한 경우 **provisioner** 노드로 다시 **SSH** 연결을 수행합니다.

```
# ssh kni@provisioner.<cluster-name>.<domain>
```

25. 연결 브리지가 올바르게 생성되었는지 확인합니다.

```
$ sudo nmcli con show
```

출력 예

```
NAME               UUID                                  TYPE      DEVICE
baremetal          4d5133a5-8351-4bb9-bfd4-3af264801530  bridge    baremetal
provisioning       43942805-017f-4d7d-a2c2-7cb3324482ed  bridge    provisioning
virbr0             d9bca40f-eee1-410b-8879-a2d4bb0465e7  bridge    virbr0
bridge-slave-eth1  76a8ed50-c7e5-4999-b4f6-6d9014dd0812  ethernet  eth1
bridge-slave-eth2  f31c3353-54b7-48de-893a-02d2b34c4736  ethernet  eth2
```

26. **pull-secret.txt** 파일을 생성합니다.

```
$ vim pull-secret.txt
```

웹 브라우저에서 Install on Bare Metal with user-provisioned infrastructure 로 이동합니다. 1단계에서 **Download pull secret(pull secret** 다운로드)을 클릭합니다. **pull-secret.txt** 파일에 내용을 붙여 넣고 **kni** 사용자의 홈 디렉터리에 저장합니다.

11.2.2. 공용 서브넷 구성

모든 **OpenShift Container Platform** 클러스터 노드는 퍼블릭 서브넷에 있어야 합니다. **IBM Cloud®** 는 서브넷에 **DHCP** 서버를 제공하지 않습니다. 프로비저너 노드에서 별도로 설정합니다.

프로비저너 노드를 준비할 때 정의된 **BASH** 변수를 재설정해야 합니다. 준비 후 프로비저너 노드를 재부팅하면 이전에 설정한 **BASH** 변수가 삭제됩니다.

프로세스

1. **dnsmasq** 를 설치하십시오 :

```
$ sudo dnf install dnsmasq
```

2. **dnsmasq** 구성 파일을 엽니다.

```
$ sudo vi /etc/dnsmasq.conf
```

3. **dnsmasq** 구성 파일에 다음 구성을 추가합니다.

```
interface=baremetal
except-interface=lo
bind-dynamic
log-dhcp

dhcp-range=<ip_addr>,<ip_addr>,<pub_cidr> (1)
dhcp-option=baremetal,121,0.0.0.0/0,<pub_gateway>,<prvn_priv_ip>,<prvn_pub_ip> (2)

dhcp-hostsfile=/var/lib/dnsmasq/dnsmasq.hostsfile
```

(1) **DHCP** 범위를 설정합니다. **baremetal** 네트워크의 **dhcp-range** 가 동일한 **IP** 주소로 시작되고 종료되도록 **<ip_addr>** 의 두 인스턴스를 공용 서브넷에서 사용하지 않는 **IP** 주소로 바꿉니다. **<pub_cidr>** 을 공용 서브넷의 **CIDR**로 바꿉니다.

(2) **DHCP** 옵션을 설정합니다. **<pub_gateway>** 를 **baremetal** 네트워크의 게이트웨이 **IP** 주소로 바꿉니다. **<prvn_priv_ip>** 를 **provisioning** 네트워크의 프로비저너 노드 개인 **IP** 주소의 **IP** 주소로 바꿉니다 . **<prvn_pub_ip>** 를 **baremetal** 네트워크에 있는 프로비저너 노드의 공용 **IP** 주소의 **IP** 주소로 바꿉니다.

<pub_cidr> 값을 검색하려면 다음을 실행합니다.

```
$ ibmcloud sl subnet detail <publicsubnetid> --output JSON | jq .cidr
```

<publicsubnetid> 를 공용 서브넷의 **ID**로 바꿉니다.

<pub_gateway> 값을 검색하려면 다음을 실행합니다.

```
$ ibmcloud sl subnet detail <publicsubnetid> --output JSON | jq .gateway -r
```

<publicsubnetid> 를 공용 서브넷의 **ID**로 바꿉니다.

<prvn_priv_ip> 값을 검색하려면 다음을 실행합니다.

```
$ ibmcloud  sl hardware detail <id> --output JSON | \
        jq .primaryBackendIpAddress -r
```

<id> 를 프로비저너 노드의 **ID**로 바꿉니다.

<prvn_pub_ip> 값을 검색하려면 다음을 실행합니다.

```
$ ibmcloud sl hardware detail <id> --output JSON | jq .primaryIpAddress -r
```

<id> 를 프로비저너 노드의 **ID**로 바꿉니다.

4. 클러스터의 하드웨어 목록을 가져옵니다.

```
$ ibmcloud sl hardware list
```

5. 각 노드의 **MAC** 주소 및 **IP** 주소를 가져옵니다.

```
$ ibmcloud sl hardware detail <id> --output JSON | \
  jq '.networkComponents[] | \
  "\(.primaryIpAddress) \(.macAddress)"' | grep -v null
```

<id> 를 노드 **ID**로 바꿉니다.

출력 예

```
"10.196.130.144 00:e0:ed:6a:ca:b4"
"141.125.65.215 00:e0:ed:6a:ca:b5"
```

공용 네트워크의 **MAC** 주소와 **IP** 주소를 기록합니다. 나중에 **install-config.yaml** 파일에서 사용할 사설 네트워크의 **MAC** 주소를 별도로 기록합니다. 공용 베어 메탈 네트워크의 모든 공용 **MAC** 및 **IP** 주소 및 개인 프로비저닝 네트워크의 **MAC** 주소가 있을 때까지 각 노드에 대해 이 절차를 반복합니다.

6. 각 노드의 공용 **baremetal** 네트워크의 **MAC** 및 **IP** 주소 쌍을 **dnsmasq.hostsfile** 파일에 추가합니다.

```
$ sudo vim /var/lib/dnsmasq/dnsmasq.hostsfile
```

입력 예

```
00:e0:ed:6a:ca:b5,141.125.65.215,master-0
<mac>,<ip>,master-1
<mac>,<ip>,master-2
<mac>,<ip>,worker-0
<mac>,<ip>,worker-1
...
```

<mac>,<ip> 를 해당 노드 이름의 공용 **MAC** 주소 및 공용 **IP** 주소로 바꿉니다.

7. **dnsmasq** 시작 :

```
$ sudo systemctl start dnsmasq
```

8. 노드를 부팅할 때 시작되도록 **dnsmasq** 를 활성화합니다.

```
$ sudo systemctl enable dnsmasq
```

9. dnsmasq 가 실행 중인지 확인합니다.

```
$ sudo systemctl status dnsmasq
```

출력 예

```
● dnsmasq.service - DNS caching server.
Loaded: loaded (/usr/lib/systemd/system/dnsmasq.service; enabled; vendor preset:
disabled)
Active: active (running) since Tue 2021-10-05 05:04:14 CDT; 49s ago
Main PID: 3101 (dnsmasq)
Tasks: 1 (limit: 204038)
Memory: 732.0K
CGroup: /system.slice/dnsmasq.service
└─3101 /usr/sbin/dnsmasq -k
```

10. UDP 프로토콜로 포트 53 및 67 을 엽니다.

```
$ sudo firewall-cmd --add-port 53/udp --permanent
```

```
$ sudo firewall-cmd --add-port 67/udp --permanent
```

11. masquerade를 사용하여 외부 영역에 프로비저닝 을 추가합니다.

```
$ sudo firewall-cmd --change-zone=provisioning --zone=external --permanent
```

이 단계에서는 관리 서브넷에 대한 IPMI 호출에 대한 네트워크 주소 변환을 수행합니다.

12. firewalld 구성을 다시 로드합니다.

```
$ sudo firewall-cmd --reload
```

11.2.3. OpenShift Container Platform 설치 프로그램 검색

설치 프로그램의 stable-4.x 버전을 사용하여 일반적으로 사용 가능한 안정적인 OpenShift Container

Platform 버전을 배포합니다.

```
$ export VERSION=stable-4.9
export RELEASE_IMAGE=$(curl -s https://mirror.openshift.com/pub/openshift-
v4/clients/ocp/$VERSION/release.txt | grep 'Pull From: quay.io' | awk -F ' ' '{print $3}')
```

11.2.4. OpenShift Container Platform 설치 프로그램 추출

설치 프로그램을 가져온 후 다음 단계로 압축을 풉니다.

절차

1. 환경 변수를 설정합니다.

```
$ export cmd=openshift-baremetal-install
$ export pullsecret_file=~/pull-secret.txt
$ export extract_dir=$(pwd)
```

2. oc 바이너리를 가져옵니다.

```
$ curl -s https://mirror.openshift.com/pub/openshift-
v4/clients/ocp/$VERSION/openshift-client-linux.tar.gz | tar zxvf - oc
```

3. 설치 프로그램 압축을 풉니다.

```
$ sudo cp oc /usr/local/bin
$ oc adm release extract --registry-config "${pullsecret_file}" --command=$cmd --to
"${extract_dir}" ${RELEASE_IMAGE}
$ sudo cp openshift-baremetal-install /usr/local/bin
```

11.2.5. install-config.yaml 파일 구성

install-config.yaml 파일에는 몇 가지 추가 정보가 필요합니다. 대부분의 정보는 사용 가능한 IBM Cloud® 하드웨어에 대해 설치 프로그램 및 결과 클러스터에 충분한 정보를 제공하여 이를 완전히 관리할 수 있도록 하는 것입니다. 베어 메탈에 설치하고 IBM Cloud에 설치하는 것과 중요한 차이점은 install-config.yaml 파일의 BMC 섹션에서 IPMI의 권한 수준을 명시적으로 설정해야 한다는 것입니다.

절차

1. install-config.yaml을 설정합니다. pullSecret 및 sshKey 등 환경에 맞게 적절한 변수를 변경합니다.

```
apiVersion: v1
baseDomain: <domain>
metadata:
  name: <cluster_name>
networking:
  machineNetwork:
  - cidr: <public-cidr>
  networkType: OVNKubernetes
compute:
- name: worker
  replicas: 2
controlPlane:
  name: master
  replicas: 3
  platform:
    baremetal: {}
platform:
  baremetal:
    apiVIP: <api_ip>
    ingressVIP: <wildcard_ip>
    provisioningNetworkInterface: <NIC1>
    provisioningNetworkCIDR: <CIDR>
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: ipmi://10.196.130.145?privilegelevel=OPERATOR 1
          username: root
          password: <password>
        bootMACAddress: 00:e0:ed:6a:ca:b4 2
        rootDeviceHints:
          deviceName: "/dev/sda"
      - name: openshift-worker-0
        role: worker
        bmc:
          address: ipmi://<out-of-band-ip>?privilegelevel=OPERATOR 3
          username: <user>
          password: <password>
        bootMACAddress: <NIC1_mac_address> 4
        rootDeviceHints:
          deviceName: "/dev/sda"
pullSecret: '<pull_secret>'
sshKey: '<ssh_pub_key>'
```

1 3

The bmc.address 는 OPERATOR 로 설정된 값이 있는 권한 수준 구성 설정을 제공합니다. 이는 IBM Cloud에 필요합니다.

2 4

해당 노드에 대한 개인 프로비저닝 네트워크 NIC의 MAC 주소를 추가합니다.

참고

ibmcloud 명령줄 유틸리티를 사용하여 암호를 검색할 수 있습니다.

```
$ ibmcloud sl hardware detail <id> --output JSON | \
  jq '"(.networkManagementIpAddress)
(.remoteManagementAccounts[0].password)"'
```

<id> 를 노드 ID로 바꿉니다.

2. 클러스터 구성을 저장할 디렉터리를 생성합니다.

```
$ mkdir ~/clusterconfigs
```

3. install-config.yaml 파일을 디렉터리에 복사합니다.

```
$ cp install-config.yaml ~/clusterconfig
```

4. OpenShift Container Platform 클러스터를 설치하기 전에 모든 베어 메탈 노드의 전원이 꺼져 있는지 확인합니다.

```
$ ipmitool -I lanplus -U <user> -P <password> -H <management_server_ip> power off
```

5. 이전 배포 시도에서 이전 부트스트랩 리소스가 남아 있는 경우 이전 부트스트랩 리소스를 제거합니다.

```
for i in $(sudo virsh list | tail -n +3 | grep bootstrap | awk {'print $2'});
do
  sudo virsh destroy $i;
  sudo virsh undefine $i;
  sudo virsh vol-delete $i --pool $i;
  sudo virsh vol-delete $i.ign --pool $i;
  sudo virsh pool-destroy $i;
  sudo virsh pool-undefine $i;
done
```

11.2.6. 추가 install-config 매개 변수

install-config.yaml 파일의 필수 매개 변수 **hosts** 매개 변수 및 **bmc** 매개 변수는 다음 표를 참조하십시오.

표 11.2. 필수 매개 변수

매개 변수	기본	설명
baseDomain		클러스터의 도메인 이름입니다. 예: **example.com**
bootMode	**UEFI**	노드의 부팅 모드입니다. 옵션은 **legacy**, **UEFI** 및 **UEFISecureBoot**입니다. **bootMode**가 설정되지 않은 경우 Ironic은 노드를 검사하는 동안 해당 노드를 설정합니다.
sshKey		**sshKey** 구성 설정에는 컨트롤 플레인 노드 및 작업자 노드에 액세스하는 데 필요한 **~/.ssh/id_rsa.pub** 파일의 키가 포함되어 있습니다. 일반적으로 이 키는 **provisioner** 노드에서 가져옵니다.
pullSecret		**pullSecret** 구성 설정에는 프로비저너 노드를 준비할 때 Install OpenShift on Bare Metal 페이지에서 다운로드한 풀 시크릿 사본이 포함되어 있습니다.
metadata: name:		OpenShift Container Platform 클러스터에 지정되는 이름입니다. 예: **openshift**

networking:
  machineNetwork:
  - cidr:
		외부 네트워크의 공개 CIDR (Classless Inter-Domain Routing)입니다. 예: **10.0.0.0/24**

compute: - name: worker		OpenShift Container Platform 클러스터에는 노드가 없는 경우에도 작업자 (또는 컴퓨팅) 노드에 이름을 지정해야 합니다.

매개 변수	기본	설명
compute: replicas: 2		복제는 OpenShift Container Platform 클러스터의 작업자 (또는 컴퓨팅) 노드 수를 설정합니다.
controlPlane: name: master		OpenShift Container Platform 클러스터에는 컨트롤 플레인 (마스터) 노드의 이름이 필요합니다.
controlPlane: replicas: 3		복제는 OpenShift Container Platform 클러스터의 일부로 포함된 컨트롤 플레인 (마스터) 노드의 수를 설정합니다.
provisioningNetworkInterface		**provisioning** 네트워크에 연결된 노드의 네트워크 인터페이스 이름입니다. OpenShift Container Platform 4.9 이상 릴리스의 경우 **bootMACAddress** 구성 설정을 사용하여 NIC 이름을 식별하는 대신 **provisioningNetworkInterface** 구성 설정을 사용하는 대신 NIC의 IP 주소를 식별할 수 있도록 Ironic을 활성화합니다.
defaultMachinePlatform		플랫폼 구성없이 머신 풀에 사용되는 기본 설정입니다.
apiVIP		(선택 사항) Kubernetes API 통신의 가상 IP 주소입니다. 이 설정은 **install-config.yaml** 파일에서 MachineNetwork에서 예약된 IP로 제공되거나 기본 이름이 올바르게 확인되도록 DNS에서 사전 구성해야 합니다. **install-config.yaml** 파일의 **apiVIP** 구성 설정에 값을 추가할 때 FQDN이 아닌 가상 IP 주소를 사용합니다. 듀얼 스택 네트워킹을 사용하는 경우 IP 주소는 기본 IPv4 네트워크에서여야 합니다. 설정하지 않으면 설치 프로그램에서 **api.<cluster_name>.<base_domain>** 을 사용하여 DNS에서 IP 주소를 파생합니다.

매개 변수	기본	설명
disableCertificateVerification	**False**	**redfish** 및 **redfish-virtualmedia** 는 BMC 주소를 관리하기 위해 이 매개 변수가 필요합니다. BMC 주소에 자체 서명된 인증서를 사용하는 경우 값은 **True** 여야합니다.
ingressVIP		(선택 사항) 수신 트래픽의 가상 IP 주소입니다. 이 설정은 **install-config.yaml** 파일에서 MachineNetwork에서 예약된 IP로 제공되거나 기본 이름이 올바르게 확인되도록 DNS에서 사전 구성해야 합니다. **install-config.yaml** 파일의 **ingressVIP** 구성 설정에 값을 추가할 때 FQDN이 아닌 가상 IP 주소를 사용합니다. 듀얼 스택 네트워킹을 사용하는 경우 IP 주소는 기본 IPv4 네트워크에 서여야 합니다. 설정되지 않은 경우 설치 프로그램은 **test.apps.<cluster_name>.<base_domain>** 을 사용하여 DNS에서 IP 주소를 파생합니다.

표 11.3. 선택적 매개변수

매개 변수	기본	설명
provisioningDHCPRange	**172.22.0.10,172.22.0.100**	**provisioning** 네트워크에서 노드의 IP 범위를 정의합니다.
provisioningNetworkCIDR	**172.22.0.0/24**	프로비저닝에 사용할 네트워크의 CIDR입니다. 이 옵션은 **provisioning** 네트워크에서 기본 주소 범위를 사용하지 않는 경우에 필요합니다.
clusterProvisioningIP	**provisioningNetworkCIDR**의 세 번째 IP 주소입니다.	프로비저닝 서비스가 실행되는 클러스터 내의 IP 주소입니다. 기본값은 **provisioning** 서브넷의 세 번째 IP 주소입니다. 예: **172.22.0.3**
bootstrapProvisioningIP	**provisioningNetworkCIDR**의 두 번째 IP 주소입니다.	설치 프로그램이 컨트롤 플레인 (마스터) 노드를 배포하는 동안 프로비저닝 서비스가 실행되는 부트스트랩 VM의 IP 주소입니다. 기본값은 **provisioning** 서브넷의 두 번째 IP 주소입니다. 예를 들면 **172.22.0.2** 또는 **2620:52:0:1307::2** 입니다.

매개 변수	기본	설명
externalBridge	**baremetal**	**baremetal** 네트워크에 연결된 하이퍼 바이저의 **baremetal** 브리지 이름입니다.
provisioningBridge	**provisioning**	**provisioning** 네트워크에 연결된 **provisioner** 호스트의 **provisioning** 브리지 이름입니다.
defaultMachinePlatform		플랫폼 구성없이 머신 풀에 사용되는 기본 설정입니다.
bootstrapOSImage		부트스트랩 노드의 기본 운영 체제 이미지를 재정의하는 URL입니다. URL에는 이미지의 SHA-256 해시가 포함되어 있어야합니다. 예: **https://mirror.openshift.com/rhcos-<version>-qemu.qcow2.gz?sha256=<uncompressed_sha256>** .
clusterOSImage		클러스터 노드의 기본 운영 체제를 재정의하는 URL입니다. URL에는 이미지의 SHA-256 해시가 포함되어 있어야 합니다. 예: **https://mirror.openshift.com/images/rhcos-<version>-openstack.qcow2.gz?sha256=<compressed_sha256>**.
provisioningNetwork		**provisioningNetwork** 구성 설정은 클러스터가 **provisioning** 네트워크를 사용하는지 여부를 결정합니다. 이 기능이 있는 경우 구성 설정은 클러스터가 네트워크를 관리하는지도 결정합니다. **disabled**: **provisioning** 네트워크의 요구 사항을 비활성화하려면 이 매개변수를 **Disabled** 로 설정합니다. **Disabled(비활성화됨)** 로 설정하는 경우 가상 미디어 기반 프로비저닝만 사용하거나 지원 설치 프로그램을 사용하여 클러스터를 가져와야 합니다. **Disabled** 및 power management를 사용하는 경우 **baremetal** 네트워크에서 BMC에 액세스할 수 있어야 합니다. **Disabled** 인 경우 프로비저닝 서비스에 사용되는 **baremetal** 네트워크에 두 개의 IP 주소를 제공해야 합니다. **Managed**: DHCP, TFTP 등을 포함한 프로비저닝 네트워크를 완전히 관리하려면 이 매개 변수를 기본값인 **Managed** 로 설정합니다. **Unmanaged**: 이 매개변수를 **Unmanaged** 로 설정하여 프로비저닝 네트워크를 활성화하지만 DHCP 수동 구성을 처리합니다. 가상 미디어의 프로비저닝이 권장되지만 필요한 경우 PXE를 계속 사용할 수 있습니다.
httpProxy		이 매개 변수를 환경 내에서 사용되는 적절한 HTTP 프록시로 설정합니다.
httpsProxy		이 매개 변수를 환경 내에서 사용되는 적절한 HTTPS 프록시로 설정합니다.
noProxy		이 매개 변수를 환경 내 프록시 사용에 대한 적절한 예외 목록으로 설정합니다.

호스트

hosts 매개 변수는 클러스터를 빌드하는 데 사용되는 별도의 베어 메탈 자산 목록입니다.

표 11.4. 호스트

이름	기본	설명
name		세부 정보와 연결할 **BareMetalHost** 리소스의 이름입니다. 예: **openshift-master-0**
role		베어 메탈 노드의 역할입니다. **master** 또는 **worker** 중 하나입니다.
bmc		베이스 보드 관리 컨트롤러에 대한 연결 세부 정보입니다. 자세한 내용은 BMC 주소 지정 섹션을 참조하십시오.
bootMACAddress		호스트가 **provisioning** 네트워크에 사용하는 NIC의 MAC 주소입니다. Ironic은 **bootMACAddress** 구성 설정을 사용하여 IP 주소를 검색합니다. 그런 다음 호스트에 바인딩됩니다. **참고** **provisioning** 네트워크를 비활성화한 경우 호스트에서 유효한 MAC 주소를 제공해야 합니다.

11.2.7. 루트 장치 팁

rootDeviceHints 매개 변수를 사용하면 설치 프로그램이 RHCOS (Red Hat Enterprise Linux CoreOS) 이미지를 특정 장치에 프로비저닝할 수 있습니다. 설치 프로그램은 장치를 검색한 순서대로 검사하고 검색된 값을 팁과 비교합니다. 설치 프로그램은 팁과 일치하는 첫 번째 검색된 장치를 사용합니다. 이 설정은 여러 팁을 결합할 수 있지만 장치는 설치 프로그램이이를 선택할 수 있도록 모든 팁과 일치해야 합니다.

표 11.5. 서브 필드

서브 필드	설명
deviceName	**/dev/vda**와 같은 Linux 장치 이름을 포함하는 문자열. 팁은 실제 값과 정확히 일치해야 합니다.
hctl	**0:0:0:0**과 같은 SCSI 버스 주소를 포함하는 문자열. 팁은 실제 값과 정확히 일치해야 합니다.

서브 필드	설명
model	공급 업체별 장치 식별자가 포함된 문자열. 팁은 실제 값의 하위 문자열입니다.
vendor	장치의 공급 업체 또는 제조업체 이름이 포함된 문자열입니다. 팁은 실제 값의 하위 문자열입니다.
serialNumber	장치 일련 번호가 포함된 문자열입니다. 팁은 실제 값과 정확히 일치해야 합니다.
minSizeGigabytes	장치의 최소 크기 (기가 바이트)를 나타내는 정수입니다.
wwn	고유 저장소 식별자를 포함하는 문자열입니다. 팁은 실제 값과 정확히 일치해야 합니다.
wwnWithExtension	공급 업체 확장이 추가된 고유 한 저장소 식별자가 포함된 문자열입니다. 팁은 실제 값과 정확히 일치해야 합니다.
wwnVendorExtension	고유 공급 업체 저장소 식별자를 포함하는 문자열입니다. 팁은 실제 값과 정확히 일치해야 합니다.
rotational	장치가 회전 디스크 여야하는지 (true) 아닌지 (false)를 나타내는 부울 값입니다.

사용 예

```
 - name: master-0
   role: master
   bmc:
     address: ipmi://10.10.0.3:6203
     username: admin
     password: redhat
   bootMACAddress: de:ad:be:ef:00:40
   rootDeviceHints:
     deviceName: "/dev/sda"
```

11.2.8. OpenShift Container Platform 매니페스트 만들기

1. **OpenShift Container Platform 매니페스트를 만듭니다.**

```
$ ./openshift-baremetal-install --dir ~/clusterconfigs create manifests
```

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true
for Scheduler cluster settings
WARNING Discarding the OpenShift Manifest that was provided in the target directory
because its dependencies are dirty and it needs to be regenerated
```

11.2.9. OpenShift Container Platform 설치 프로그램을 통해 클러스터 배포

OpenShift Container Platform 설치 프로그램을 실행합니다.

```
$ ./openshift-baremetal-install --dir ~/clusterconfigs --log-level debug create cluster
```

11.2.10. 설치 후

배포 프로세스 중에 설치 디렉터리 폴더의 **.openshift_install.log** 로그 파일에 **tail** 명령을 실행하여 설치의 전체 상태를 확인할 수 있습니다.

```
$ tail -f /path/to/install-dir/.openshift_install.log
```

12장. IBM Z 및 LINUXONE에 Z/VM으로 설치

12.1. IBM Z 및 LINUXONE에 Z/VM으로 클러스터 설치 준비

12.1.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.

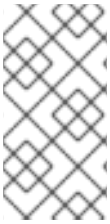
12.1.2. IBM Z 또는 Linux ONE에서 z/VM과 함께 OpenShift Container Platform을 설치할 방법 선택

다음 방법 중 하나를 사용하여 프로비저닝하는 IBM Z 또는 LinuxONE 인프라에 z/VM으로 클러스터를 설치할 수 있습니다.

- IBM Z 및 Linux ONE에 z/VM과 함께 클러스터 설치: 프로비저닝하는 IBM Z 또는 Linux ONE 인프라에 z/VM과 함께 OpenShift Container Platform을 설치할 수 있습니다.
- 네트워크가 제한된 환경에서 IBM Z 및 LinuxONE에 z/VM으로 클러스터 설치: 설치 릴리스 콘텐츠의 내부 미러를 사용하여 제한되거나 연결이 끊긴 네트워크에서 프로비저닝하는 IBM Z 또는 LinuxONE 인프라에 z/VM으로 OpenShift Container Platform을 설치할 수 있습니다. 이 방법을 사용하여 소프트웨어 구성 요소를 받기 위해 활성 인터넷 연결이 필요하지 않은 클러스터를 설치할 수 있습니다. 이 설치 방법을 사용하여 클러스터가 외부 콘텐츠에 대해 조직의 제어 조건을 충족하는 컨테이너 이미지만 사용하도록 할 수 있습니다.

12.2. IBM Z 및 LINUXONE에 Z/VM으로 클러스터 설치

OpenShift Container Platform 버전 4.9에서는 사용자가 제공하는 IBM Z 또는 LinuxONE 인프라에 클러스터를 설치할 수 있습니다.



참고

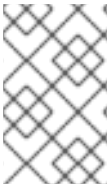
이 문서는 IBM Z에 대해서만 설명하지만 여기에 있는 모든 정보는 LinuxONE에도 적용됩니다.

중요

베어 메탈 이외의 플랫폼의 경우 추가 고려 사항이 있습니다. **OpenShift Container Platform** 클러스터를 설치하기 전에 **guidelines for deploying OpenShift Container Platform on non-tested platforms**에 있는 내용을 확인하십시오.

12.2.1. 사전 요구 사항

- **OpenShift Container Platform 설치 및 업데이트** 프로세스에 대한 세부 사항을 검토합니다.
- **클러스터 설치 방법 선택 및 사용자를 위한 준비**에 대한 문서를 읽습니다.
- 설치 프로세스를 시작하기 전에 설치 디렉터리를 정리해야 합니다. 이렇게 하면 설치 프로세스 중에 필요한 설치 파일이 생성되고 업데이트됩니다.
- 클러스터 용 **NFS를 사용하여 영구 스토리지**를 프로비저닝합니다. 프라이빗 이미지 레지스트리를 배포하려면 스토리지에서 **ReadWriteMany** 액세스 모드를 제공해야 합니다.
- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 **사이트를 허용하도록 방화벽을 구성**해야 합니다.



참고

프록시를 구성하는 경우에도 해당 사이트 목록을 검토하십시오.

12.2.2. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- **OpenShift Cluster Manager** 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 **Telemetry** 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.

- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.

중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

12.2.3. 사용자 프로비저닝 인프라를 포함한 클러스터의 시스템 요구사항

사용자 프로비저닝 인프라가 포함된 클러스터의 경우, 필요한 모든 시스템을 배포해야 합니다.

이 섹션에서는 사용자 프로비저닝 인프라에 **OpenShift Container Platform**을 배포해야 하는 요구 사항에 대해 설명합니다.

12.2.3.1. 클러스터 설치에 필요한 시스템

최소 **OpenShift Container Platform** 클러스터에 다음과 같은 호스트가 필요합니다.

표 12.1. 최소 필수 호스트

호스트	설명
임시 부트스트랩 시스템 한 개	컨트롤 플레인 시스템 세 개에 OpenShift Container Platform 클러스터를 배포하기 위한 부트스트랩 시스템이 클러스터에 필요합니다. 클러스터를 설치한 후 부트스트랩 시스템을 제거할 수 있습니다.
컨트롤 플레인 시스템 세 개	컨트롤 플레인 시스템은 컨트롤 플레인을 구성하는 Kubernetes 및 OpenShift Container Platform 서비스를 실행합니다.
두 개 이상의 컴퓨팅 시스템(작업자 시스템이라고도 함).	OpenShift Container Platform 사용자가 요청한 워크로드는 컴퓨팅 머신에서 실행됩니다.

중요

클러스터의 고가용성을 개선하려면 컨트롤 플레인 시스템을 두 개 이상의 물리적 시스템의 서로 다른 z/VM 인스턴스에 배포하십시오.

부트스트랩, 컨트롤 플레인 시스템은 운영 체제로 RHCOS (Red Hat Enterprise Linux CoreOS)를 사용해야 합니다. 그러나 컴퓨팅 머신은 RHCOS(Red Hat Enterprise Linux CoreOS), RHEL (Red Hat Enterprise Linux) 7.9 또는 RHEL 8.4 중에서 선택할 수 있습니다.

RHCOS는 RHEL 8(Red Hat Enterprise Linux)을 기반으로하며 모든 하드웨어 인증 및 요구사항을 모두 이어받습니다. Red Hat Enterprise Linux 기술 기능 및 제한을 참조하십시오.

12.2.3.2. 클러스터 설치를 위한 최소 리소스 요구 사항

각 클러스터 시스템이 다음과 같은 최소 요구사항을 충족해야 합니다.

표 12.2. 최소 리소스 요구사항

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS
부트스트랩	RHCOS	4	16GB	100GB	해당 없음
컨트롤 플레인	RHCOS	4	16GB	100GB	해당 없음
컴퓨팅	RHCOS	2	8GB	100GB	해당 없음

1. SMT-2가 활성화된 경우 하나의 물리적 코어(IFL)는 두 개의 논리 코어(스레드)를 제공합니다. 하이퍼바이저는 두 개 이상의 vCPU를 제공할 수 있습니다.

12.2.3.3. 최소 IBM Z 시스템 환경

다음 IBM 하드웨어에 OpenShift Container Platform 버전 4.9를 설치할 수 있습니다.

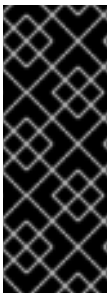
- IBM z15(모든 모델), IBM z14(모든 모델), IBM z13 및 IBM z13s
- LinuxONE, 모든 버전

하드웨어 요구 사항

- 각 클러스터에 대해 SMT2를 사용할 수 있는 6개의 IFL과 동등합니다.
- LoadBalancer 서비스에 연결하고 클러스터 외부 트래픽에 대한 데이터를 제공하는 네트워크 연결은 하나 이상 있습니다.

참고

전용 또는 공유 IFL을 사용하여 충분한 컴퓨팅 리소스를 할당할 수 있습니다. 리소스 공유는 IBM Z의 주요 강점 중 하나입니다. 그러나 각 하이퍼바이저 계층에서 용량을 올바르게 조정하고 모든 OpenShift Container Platform 클러스터에 충분한 리소스를 확인해야 합니다.



중요

클러스터의 전반적인 성능에 영향을 미칠 수 있으므로 OpenShift Container Platform 클러스터를 설정하는 데 사용되는 LPAR은 충분한 컴퓨팅 용량을 제공해야 합니다. 이 컨텍스트에서 하이퍼바이저 수준의 LPAR 가중치 관리, 권한 부여 및 CPU 공유는 중요한 역할을 합니다.

운영 체제 요구 사항

- z/VM 7.1 이상의 인스턴스 1개

z/VM 인스턴스에서 다음을 설정합니다.

- OpenShift Container Platform 컨트롤 플레인 시스템 용 게스트 가상 머신 3대
- OpenShift Container Platform 컴퓨팅 시스템 용 게스트 가상 머신 2 대
- 임시 OpenShift Container Platform 부트스트랩 머신 용 게스트 가상 머신 1 대

IBM Z 네트워크 연결 요구 사항

IBM Z의 z/VM에 설치하려면 레이어 2 모드의 단일 z/VM 가상 NIC가 필요합니다. 또한 다음이 필요합니다.

- 직접 연결된 OSA 또는 RoCE 네트워크 어댑터

- z/VM VSwitch 설정. 권장 설정의 경우 OSA 링크 통합을 사용합니다.

z/VM 게스트 가상 머신 용 디스크 스토리지

- FICON 연결 디스크 스토리지 (DASD). 이는 z/VM 미니 디스크, 풀팩 미니 디스크 또는 전용 DASD가 포함될 수 있으며, 모두 기본 CDL로 포맷되어야 합니다. Red Hat Enterprise Linux CoreOS (RHCOS) 설치에 필요한 최소 DASD 크기에 도달하려면 확장 주소 볼륨 (EAV)이 필요합니다. 사용 가능한 경우 HyperPAV를 사용하여 최적의 성능을 보장합니다.

- FCP 연결 디스크 스토리지

스토리지 / 메인 메모리

- OpenShift Container Platform 컨트롤 플레인 시스템용 16GB

- OpenShift Container Platform 컴퓨팅 머신 용 8GB

- 임시 OpenShift Container Platform 부트스트랩 머신 용 16GB

12.2.3.4. 권장되는 IBM Z 시스템 환경

하드웨어 요구 사항

- 각 클러스터에 대해 SMT2가 활성화된 6개의 IFL과 동등한 3개의 LPARS가 있습니다.

- LoadBalancer 서비스에 연결하고 클러스터 외부의 트래픽에 대한 데이터를 제공하는 두 개의 네트워크 연결

- Hipersockets - 하나의 장치에 직접 연결되거나 z/VM 게스트에 투명하도록 하나의 z/VM VSWITCH와 브리징하여 노드에 연결됩니다. HiperSockets을 노드에 직접 연결하려면 RHEL 8 게스트를 통해 외부 네트워크의 게이트웨이를 설정하여 HiperSockets 네트워크에 연결해야 합니다.

운영 체제 요구 사항

- 고가용성을 위해 z/VM 7.1 이상의 인스턴스 2개 또는 3개

z/VM 인스턴스에서 다음을 설정합니다.

- OpenShift Container Platform 컨트롤 플레인 시스템용 게스트 가상 머신 3개(z/VM 인스턴스당 1개)
- z/VM 인스턴스에 분산된 OpenShift Container Platform 컴퓨팅 머신용 게스트 가상 머신 6개 이상
- 임시 OpenShift Container Platform 부트스트랩 시스템용 게스트 가상 머신 1개
- 오버 커밋된 환경에서 통합 구성 요소를 사용하려면 CP 명령 SET SHARE 를 사용하여 컨트롤 플레인의 우선 순위를 높입니다. 인프라 노드가 있는 경우 동일한 작업을 수행합니다. IBM 문서의 SET SHARE를 참조하십시오.

IBM Z 네트워크 연결 요구 사항

IBM Z의 z/VM에 설치하려면 레이어 2 모드의 단일 z/VM 가상 NIC가 필요합니다. 또한 다음이 필요합니다.

- 직접 연결된 OSA 또는 RoCE 네트워크 어댑터
- z/VM VSwitch 설정. 권장 설정의 경우 OSA 링크 통합을 사용합니다.

z/VM 게스트 가상 머신 용 디스크 스토리지

- FICON 연결 디스크 스토리지 (DASD). 이는 z/VM 미니 디스크, 풀팩 미니 디스크 또는 전용 DASD가 포함될 수 있으며, 모두 기본 CDL로 포맷되어야 합니다. Red Hat Enterprise Linux CoreOS (RHCOS) 설치에 필요한 최소 DASD 크기에 도달하려면 확장 주소 볼륨 (EAV)이 필요합니다. 가능한 경우 HyperPAV 및 고성능 FICON (zHPF)을 사용하여 최적의 성능을 보장합니다.
- FCP 연결 디스크 스토리지

스토리지 / 메인 메모리

- OpenShift Container Platform 컨트롤 플레인 시스템용 16GB
- OpenShift Container Platform 컴퓨팅 머신 용 8GB
- 임시 OpenShift Container Platform 부트스트랩 머신 용 16GB

12.2.3.5. 인증서 서명 요청 관리

사용자가 프로비저닝하는 인프라를 사용하는 경우 자동 시스템 관리 기능으로 인해 클러스터의 액세스가 제한되므로 설치한 후 클러스터 인증서 서명 요청(CSR)을 승인하는 메커니즘을 제공해야 합니다. kube-controller-manager는 kubelet 클라이언트 CSR만 승인합니다. machine-approver는 올바른 시스템에서 발행한 요청인지 확인할 수 없기 때문에 kubelet 자격 증명을 사용하여 요청하는 서비스 인증서의 유효성을 보장할 수 없습니다. kubelet 서빙 인증서 요청의 유효성을 확인하고 요청을 승인하는 방법을 결정하여 구현해야 합니다.

추가 리소스

- IBM 문서에서 Bridging a HiperSockets LAN with a z/VM Virtual Switch를 참조하십시오.
- 성능 최적화를 위해 Scaling HyperPAV alias devices on Linux guests on z/VM을 참조하십시오.
- LPAR 가중치 관리 및 자격을 보려면 LPAR 성능의 주제를 참조하십시오.
- IBM Z 및 LinuxONE 환경에 대한 권장 호스트 사례

12.2.3.6. 사용자 프로비저닝 인프라에 대한 네트워킹 요구사항

모든 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템이 부팅 중에 Ignition 구성 파일을 가져오려면 initramfs에 네트워킹을 구성해야 합니다.

초기 부팅 과정에서 시스템에 Ignition 설정 파일을 다운로드하는 데 필요한 네트워크 연결을 구축하기 위해 HTTP 또는 HTTPS 서버가 있어야 합니다.

머신은 고정 IP 주소로 설정됩니다. DHCP 서버가 필요하지 않습니다. 시스템에 영구 IP 주소와 호스

트 이름이 있는지 확인합니다.

Kubernetes API 서버가 클러스터 시스템의 노드 이름을 확인할 수 있어야 합니다. **API** 서버와 작업자 노드가 서로 다른 영역에 있는 경우, **API** 서버가 노드 이름을 확인할 수 있도록 기본 **DNS** 검색 영역을 설정할 수 있습니다. 노드 개체와 모든 **DNS** 요청에서 항상 정규화된 도메인 이름으로 호스트를 가리키는 것도 지원되는 방법입니다

12.2.3.6.1. 네트워크 연결 요구사항

OpenShift Container Platform 클러스터 구성 요소가 통신할 수 있도록 시스템 간 네트워크 연결을 구성해야 합니다. 각 시스템에서 클러스터에 있는 다른 모든 시스템의 호스트 이름을 확인할 수 있어야 합니다.

이 섹션에서는 필요한 포트에 대해 자세히 설명합니다.

중요

연결된 **OpenShift Container Platform** 환경에서 모든 노드는 플랫폼 컨테이너의 이미지를 가져오고 **Red Hat**에 원격 측정 데이터를 제공하기 위해 인터넷에 액세스할 수 있어야 합니다.

표 **12.3.** 모든 시스템 간 통신에 사용되는 포트

프로토콜	포트	설명
ICMP	해당 없음	네트워크 연결성 테스트
TCP	**1936**	메트릭
	9000-**9999**	**9100**-**9101** 포트의 노드 내보내기 및 **9099** 포트의 Cluster Version Operator를 포함한 호스트 수준 서비스.
	10250-**10259**	Kubernetes에서 예약하는 기본 포트
	10256	openshift-sdn
UDP	**4789**	VXLAN 및 Geneve
	6081	VXLAN 및 Geneve
	9000-**9999**	**9100**-**9101** 포트의 노드 내보내기를 포함한 호스트 수준 서비스.

프로토콜	포트	설명
	500	IPsec IKE 패킷
	4500	IPsec NAT-T 패킷
TCP/UDP	**30000-32767**	Kubernetes 노드 포트
ESP	해당 없음	IPsec Encapsulating Security Payload (ESP)

표 12.4. 모든 시스템과 컨트롤 플레인 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**6443**	Kubernetes API

표 12.5. 컨트롤 플레인 머신 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**2379-2380**	etcd 서버 및 피어 포트

사용자 프로비저닝 인프라에 대한 NTP 구성

OpenShift Container Platform 클러스터는 기본적으로 공용 **NTP(Network Time Protocol)** 서버를 사용하도록 구성되어 있습니다. 로컬 엔터프라이즈 **NTP** 서버를 사용하거나 클러스터가 연결이 끊긴 네트워크에 배포되는 경우 특정 시간 서버를 사용하도록 클러스터를 구성할 수 있습니다. 자세한 내용은 ***chrony* 타임 서비스 설정** 문서를 참조하십시오.

추가 리소스

- **chrony 타임 서비스 설정**

12.2.3.7. 사용자 프로비저닝 DNS 요구사항

OpenShift Container Platform 배포의 경우 다음 구성 요소에 **DNS** 이름을 확인해야 합니다.

- **Kubernetes API**
- **OpenShift Container Platform** 애플리케이션 와일드카드

- 부트스트랩, 컨트롤 플레인 및 컴퓨팅 시스템

Kubernetes API, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템에 대한 역방향 DNS 확인이 필요합니다.

DNS A/AAAA 또는 CNAME 레코드는 이름 확인에 사용되며 PTR 레코드는 역방향 이름 확인에 사용됩니다. RHCOS (Red Hat Enterprise Linux CoreOS)는 DHCP에서 호스트 이름을 제공하지 않는 한 모든 노드의 호스트 이름을 설정할 때 역방향 레코드를 사용하기 때문에 역방향 레코드가 중요합니다. 또한 역방향 레코드는 OpenShift Container Platform이 작동하는 데 필요한 인증서 서명 요청 (CSR)을 생성하는 데 사용됩니다.

사용자가 프로비저닝한 OpenShift Container Platform 클러스터에 대해 다음 DNS 레코드가 필요하며 설치 전에 있어야 합니다. 각 레코드에서 <cluster_name>은 클러스터 이름이고 <base_domain>은 install-config.yaml 파일에서 지정하는 기반 도메인입니다. 전체 DNS 레코드는 <component>.<cluster_name>.<base_domain> 형식입니다.

표 12.6. 필수 DNS 레코드

구성 요소	레코드	설명
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API 로드 밸런서를 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.
	api-int.<cluster_name>.<base_domain>.	내부적으로 API 로드 밸런서를 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.  **중요** API 서버는 Kubernetes에 기록된 호스트 이름으로 작업자 노드를 확인할 수 있어야 합니다. API 서버가 노드 이름을 확인할 수 없는 경우 프록시된 API 호출이 실패할 수 있으며 pod에서 로그를 검색할 수 없습니다.

구성 요소	레코드	설명
라우트	***.apps.<cluster_name>. <base_domain>.**	애플리케이션 인그레스 로드 밸런서를 참조하는 와일드카드 DNS A/AAA 또는 CNAME 레코드입니다. 애플리케이션 인그레스 로드 밸런서는 Ingress 컨트롤러 Pod를 실행하는 머신을 대상으로 합니다. Ingress 컨트롤러 Pod는 기본적으로 컴퓨팅 머신에서 실행됩니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다. 예를 들어 **console-openshift-console.apps. <cluster_name>.<base_domain>**은 OpenShift Container Platform 콘솔의 와일드카드 경로로 사용됩니다.
부트스트랩 시스템	**bootstrap.<cluster_name>. <base_domain>.**	부트스트랩 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.
컨트롤 플레인 머신	**<master><n>. <cluster_name>. <base_domain>.**	컨트롤 플레인 노드의 각 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.
컴퓨팅 머신	**<worker><n>. <cluster_name>. <base_domain>.**	작업자 노드의 각 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.

참고

OpenShift Container Platform 4.4 이상에서는 DNS 구성에서 etcd 호스트 및 SRV 레코드를 지정할 필요가 없습니다.

작은 정보

dig 명령을 사용하여 이름과 역방향 이름을 확인할 수 있습니다. 자세한 검증 단계는 *사용자 프로비저닝 인프라의 DNS 확인* 섹션을 참조하십시오.

12.2.3.7.1. 사용자 프로비저닝 클러스터의 DNS 구성 예

이 섹션에서는 사용자 프로비저닝 인프라에 OpenShift Container Platform을 배포하기 위한 DNS 요구 사항을 충족하는 A 및 PTR 레코드 구성 샘플을 제공합니다. 샘플은 하나의 DNS 솔루션을 선택하기 위한 조언을 제공하기 위한 것이 아닙니다.

이 예제에서 클러스터 이름은 **ocp4**이고 기본 도메인은 **example.com**입니다.

사용자 프로비저닝 클러스터의 **DNS A** 레코드 구성 예

다음 **BIND** 영역 파일의 예제에서는 사용자가 프로비저닝한 클러스터의 이름 확인을 위한 샘플 **A** 레코드를 보여줍니다.

예 12.1. 샘플 DNS 영역 데이터베이스

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
master0.ocp4.example.com. IN A 192.168.1.97 5
master1.ocp4.example.com. IN A 192.168.1.98 6
master2.ocp4.example.com. IN A 192.168.1.99 7
;
worker0.ocp4.example.com. IN A 192.168.1.11 8
worker1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

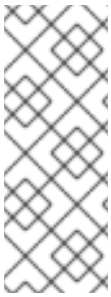
1

Kubernetes API의 이름 확인을 제공합니다. 레코드는 **API** 로드 밸런서의 **IP** 주소를 나타냅니다.

2

3

와일드카드 경로의 이름 확인을 제공합니다. 레코드는 애플리케이션 인그레스 로드 밸런서의 IP 주소를 나타냅니다. 애플리케이션 인그레스 로드 밸런서는 Ingress 컨트롤러 Pod를 실행하는 머신을 대상으로 합니다. Ingress 컨트롤러 Pod는 기본적으로 컴퓨팅 머신에서 실행됩니다.



참고

이 예제에서는 Kubernetes API 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 API 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

4

부트스트랩 시스템의 이름 확인을 제공합니다.

5 6 7

컨트롤 플레인 시스템의 이름 확인을 제공합니다.

8 9

컴퓨팅 시스템의 이름 확인을 제공합니다.

사용자 프로비저닝 클러스터의 DNS PTR 레코드 구성 예

다음 예제 BIND 영역 파일은 사용자 프로비저닝 클러스터의 역방향 이름 확인을 위한 샘플 PTR 레코드를 보여줍니다.

예 12.2. 역방향 레코드의 샘플 DNS 영역 데이터베이스

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
```

```
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR master0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR master1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR master2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com. 8
;
;EOF
```

1

Kubernetes API의 역방향 **DNS** 확인을 제공합니다. **PTR** 레코드는 **API** 로드 밸런서의 레코드 이름을 참조합니다.

2

Kubernetes API의 역방향 **DNS** 확인을 제공합니다. **PTR** 레코드는 **API** 로드 밸런서의 레코드 이름을 참조하며 내부 클러스터 통신에 사용됩니다.

3

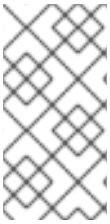
부트스트랩 시스템의 역방향 **DNS** 확인을 제공합니다.

4 5 6

컨트롤 플레인 시스템의 역방향 **DNS** 확인을 제공합니다.

7 8

컴퓨팅 시스템의 역방향 **DNS** 확인을 제공합니다.



참고

OpenShift Container Platform 애플리케이션 와일드카드에는 **PTR** 레코드가 필요하지 않습니다.

12.2.3.8. 사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구사항

OpenShift Container Platform을 설치하기 전에 **API** 및 애플리케이션 인그레스 로드 밸런싱 인프라를 프로비저닝해야 합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 **API** 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

참고

RHEL(Red Hat Enterprise Linux) 인스턴스를 사용하여 API 및 애플리케이션 인그레스 로드 밸런서를 배포하려면 RHEL 서브스크립션을 별도로 구매해야 합니다.

로드 밸런서 인프라는 다음 요구 사항을 충족해야 합니다.

1. API 로드 밸런서: 플랫폼과 상호 작용하고 플랫폼을 구성할 수 있도록 사용자(인간과 시스템 모두)에게 공통 끝점을 제공합니다. 다음 조건을 설정합니다.

   - Layer 4 로드 밸런싱 전용입니다. 이를 Raw TCP, SSL Passthrough 또는 SSL Bridge 모드라고 합니다. SSL Bridge 모드를 사용하는 경우, API 경로에 대해 SNI(Server Name Indication, 서버 이름 표시)를 활성화해야 합니다.

   - 스테이트리스 로드 밸런싱 알고리즘입니다. 옵션은 로드 밸런서 구현에 따라 달라집니다.

참고

API 로드 밸런서가 제대로 작동하기 위해 세션 지속성이 필요하지 않습니다.

로드 밸런서의 전면과 후면 모두에서 다음 포트를 구성하십시오.

표 12.7. API 로드 밸런서

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
6443	부트스트랩 및 컨트롤 플레인. 부트스트랩 시스템이 클러스터 컨트롤 플레인을 초기화한 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다. API 서버 상태 검사 프로브에 대한 **/readyz** 끝점을 구성해야 합니다.	X	X	Kubernetes API 서버
22623	부트스트랩 및 컨트롤 플레인. 부트스트랩 시스템이 클러스터 컨트롤 플레인을 초기화한 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다.	X		시스템 구성 서버

참고

API 서버가 /readyz 엔드포인트를 해제하는 시점부터 풀에서 API 서버 인스턴스가 제거되는 시점까지 시간이 30초를 넘지 않도록 로드 밸런서를 구성해야 합니다. /readyz가 오류를 반환하거나 정상 상태가 된 후 정해진 시간 안에 끝점이 제거 또는 추가되어야 합니다. 5초 또는 10초의 프로빙 주기(두 번의 성공적인 요청은 정상 상태, 세 번의 요청은 비정상 상태)는 충분한 테스트를 거친 값입니다.

2. 애플리케이션 인그레스 로드 밸런서: 클러스터 외부에서 유입되는 애플리케이션 트래픽에 대한 인그래스 포인트를 제공합니다. 다음 조건을 설정합니다.

- Layer 4 로드 밸런싱 전용입니다. 이를 Raw TCP, SSL Passthrough 또는 SSL Bridge 모드라고 합니다. SSL Bridge 모드를 사용하는 경우 인그레스 경로에 대해 SNI(Server Name Indication, 서버 이름 표시)를 활성화해야 합니다.

- 사용 가능한 옵션과 플랫폼에서 호스팅되는 애플리케이션 유형에 따라 연결 기반 또는 세션 기반 지속성이 권장됩니다.

작은 정보

애플리케이션 인그레스 로드 밸런서에서 클라이언트의 실제 IP 주소를 확인할 수 있는 경우 소스 IP 기반 세션 지속성을 활성화하면 엔드 투 엔드 TLS 암호화를 사용하는 애플리케이션의 성능을 향상시킬 수 있습니다.

로드 밸런서의 전면과 후면 모두에서 다음 포트를 구성하십시오.

표 12.8. 애플리케이션 인그레스 로드 밸런서

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
443	기본적으로 인그레스 컨트롤러 pod, 컴퓨팅 또는 작업자를 실행하는 시스템입니다.	X	X	HTTPS 트래픽
80	기본적으로 인그레스 컨트롤러 pod, 컴퓨팅 또는 작업자를 실행하는 시스템입니다.	X	X	HTTP 트래픽

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
1936	기본적으로 Ingress 컨트롤러 Pod를 실행하는 작업자 노드입니다. 수신 상태 점검 프로브에 대해 **/healthz/ready** 끝점을 구성해야 합니다.	X	X	HTTP 트래픽

참고

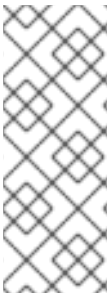
컴퓨팅 노드가 0인 3-노드 클러스터를 배포하는 경우 Ingress 컨트롤러 Pod는 컨트롤 플레인 노드에서 실행됩니다. 3-노드 클러스터 배포에서 HTTP 및 HTTPS 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다.

참고

인그레스 라우터에 대한 작업 구성이 OpenShift Container Platform 클러스터에 필요합니다. 컨트롤 플레인 초기화 후 인그레스 라우터를 설정해야 합니다.

12.2.3.8.1. 사용자 프로비저닝 클러스터의 로드 밸런서 구성 예

이 섹션에서는 사용자 프로비저닝 클러스터의 로드 밸런싱 요구 사항을 충족하는 API 및 애플리케이션 수신 로드 밸런서 구성 예를 제공합니다. 샘플은 HAProxy 로드 밸런서에 대한 **/etc/haproxy/haproxy.cfg** 구성입니다. 이 예제에서는 하나의 로드 밸런싱 솔루션을 선택하기 위한 조언을 제공하는 것을 목적으로 하지 않습니다.



참고

이 예제에서는 Kubernetes API 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 API 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

예 12.3. API 및 애플리케이션 인그레스 로드 밸런서 구성 샘플

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
```

```
    option                  redispatch
    retries                 3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s
    maxconn                 3000
  frontend stats
    bind *:1936
    mode            http
    log             global
    maxconn 10
    stats enable
    stats hide-version
    stats refresh 30s
    stats show-node
    stats show-desc Stats for ocp4 cluster (1)
    stats auth admin:ocp4
    stats uri /stats
  listen api-server-6443 (2)
    bind *:6443
    mode tcp
    server bootstrap bootstrap.ocp4.example.com:6443 check inter 1s backup (3)
    server master0 master0.ocp4.example.com:6443 check inter 1s
    server master1 master1.ocp4.example.com:6443 check inter 1s
    server master2 master2.ocp4.example.com:6443 check inter 1s
  listen machine-config-server-22623 (4)
    bind *:22623
    mode tcp
    server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup (5)
    server master0 master0.ocp4.example.com:22623 check inter 1s
    server master1 master1.ocp4.example.com:22623 check inter 1s
    server master2 master2.ocp4.example.com:22623 check inter 1s
  listen ingress-router-443 (6)
    bind *:443
    mode tcp
    balance source
    server worker0 worker0.ocp4.example.com:443 check inter 1s
    server worker1 worker1.ocp4.example.com:443 check inter 1s
  listen ingress-router-80 (7)
    bind *:80
    mode tcp
    balance source
    server worker0 worker0.ocp4.example.com:80 check inter 1s
    server worker1 worker1.ocp4.example.com:80 check inter 1s
```

(1)

이 예에서 클러스터 이름은 **ocp4** 입니다.

(2)

3 5

부트스트랩 항목은 **OpenShift Container Platform** 클러스터 설치 전에 있어야 하며 부트스트랩 프로세스가 완료된 후 제거해야 합니다.

4

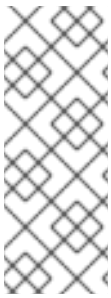
포트 **22623**은 머신 구성 서버 트래픽을 처리하고 컨트롤 플레인 시스템을 가리킵니다.

6

포트 **443**은 **HTTPS** 트래픽을 처리하고 **Ingress** 컨트롤러 **Pod**를 실행하는 시스템을 가리킵니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

7

포트 **80**은 **HTTP** 트래픽을 처리하고 **Ingress** 컨트롤러 **Pod**를 실행하는 머신을 가리킵니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

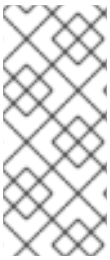


참고

컴퓨팅 노드가 **0**인 **3-**노드 클러스터를 배포하는 경우 **Ingress** 컨트롤러 **Pod**는 컨트롤 플레인 노드에서 실행됩니다. **3-**노드 클러스터 배포에서 **HTTP** 및 **HTTPS** 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다.

작은 정보

HAProxy를 로드 밸런서로 사용하는 경우 **HAProxy** 노드에서 **netstat -nltupe**를 실행하여 **haproxy** 프로세스가 포트 **6443, 22623, 443** 및 **80**에서 수신 대기 중인지 확인할 수 있습니다.



참고

HAProxy를 로드 밸런서로 사용하고 **SELinux**가 **enforcing**으로 설정된 경우 **HAProxy** 서비스가 **setsebool -P haproxy_connect_any=1**을 실행하여 구성된 **TCP** 포트에 바인딩할 수 있는지 확인해야 합니다.

12.2.4. 사용자 프로비저닝 인프라 준비

사용자 프로비저닝 인프라에 **OpenShift Container Platform**을 설치하기 전에 기본 인프라를 준비해야 합니다.

이 섹션에서는 **OpenShift Container Platform** 설치를 준비하기 위해 클러스터 인프라를 설정하는 데 필요한 높은 수준의 단계에 대해 자세히 설명합니다. 여기에는 클러스터 노드에 대한 **IP** 네트워킹 및 네트워크 연결 구성, **Ignition** 파일의 웹 서버 준비, 방화벽을 통해 필요한 포트 활성화, 필수 **DNS** 및 로드 밸런싱 인프라 설정 등이 포함됩니다.

준비 후 클러스터 인프라는 *사용자 프로비저닝 인프라가 있는 클러스터의 요구 사항* 섹션에 설명된 요구 사항을 충족해야 합니다.

사전 요구 사항

- OpenShift Container Platform 4.x 테스트된 통합 페이지를 검토했습니다.
- *사용자 프로비저닝 인프라가 있는 클러스터의 요구 사항* 섹션에 자세히 설명된 인프라 요구 사항을 검토했습니다.

프로세스

1. 고정 **IP** 주소를 설정합니다.
2. 클러스터 노드에 **Ignition** 파일을 제공하기 위해 **HTTP** 또는 **HTTPS** 서버를 설정합니다.
3. 네트워크 인프라가 클러스터 구성 요소 간 필수 네트워크 연결을 제공하는지 확인합니다. 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 인프라 섹션의 네트워킹 요구 사항* 섹션을 참조하십시오.
4. **OpenShift Container Platform** 클러스터 구성 요소가 통신하는 데 필요한 포트를 활성화하도록 방화벽을 구성합니다. 필요한 포트에 대한 자세한 내용은 *사용자 프로비저닝 인프라 섹션의 네트워킹 요구 사항* 섹션을 참조하십시오.
5. 클러스터에 필요한 **DNS** 인프라를 설정합니다.
    a. **Kubernetes API**, 애플리케이션 와일드카드, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템의 **DNS** 이름 확인을 구성합니다.

b.

Kubernetes API, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템에 대한 역방향 DNS 확인을 구성합니다.

OpenShift Container Platform DNS 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 DNS 요구 사항* 섹션을 참조하십시오.

6.

DNS 구성을 확인합니다.

a.

설치 노드에서 Kubernetes API의 레코드 이름, 와일드카드 경로 및 클러스터 노드에 대해 DNS 조회를 실행합니다. 응답의 IP 주소가 올바른 구성 요소에 해당하는지 확인합니다.

b.

설치 노드에서 로드 밸런서 및 클러스터 노드의 IP 주소에 대해 역방향 DNS 조회를 실행합니다. 응답의 레코드 이름이 올바른 구성 요소에 해당하는지 확인합니다.

자세한 DNS 검증 단계는 *사용자 프로비저닝 인프라에 대한 DNS 확인* 섹션을 참조하십시오.

7.

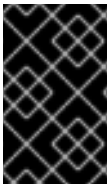
필요한 API 및 애플리케이션 수신 로드 밸런싱 인프라를 프로비저닝합니다. 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구 사항* 섹션을 참조하십시오.

참고

일부 로드 밸런싱 솔루션에는 로드 밸런싱을 초기화하기 전에 클러스터 노드의 DNS 이름을 확인해야 합니다.

12.2.5. 사용자 프로비저닝 인프라에 대한 DNS 확인 검증

사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치하기 전에 DNS 구성을 확인할 수 있습니다.



중요

클러스터를 설치하기 전에 이 섹션에 설명된 검증 단계를 성공해야 합니다.

사전 요구 사항

- 사용자 프로비저닝 인프라에 필요한 DNS 레코드를 구성했습니다.

프로세스

1. 설치 노드에서 Kubernetes API의 레코드 이름, 와일드카드 경로 및 클러스터 노드에 대해 DNS 조회를 실행합니다. 응답에 포함된 IP 주소가 올바른 구성 요소에 해당하는지 확인합니다.

   a. Kubernetes API 레코드 이름을 조회합니다. 결과가 API 로드 밸런서의 IP 주소를 가리키는지 확인합니다.

   ```
   $ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
   ```

   1 <nameserver_ip>를 네임서버의 IP 주소로, <cluster_name>을 클러스터 이름으로, <base_domain>을 기본 도메인 이름으로 바꿉니다.

   출력 예

   ```
   api.ocp4.example.com.  0 IN A 192.168.1.5
   ```

   b. Kubernetes 내부 API 레코드 이름을 조회합니다. 결과가 API 로드 밸런서의 IP 주소를 가리키는지 확인합니다.

   ```
   $ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
   ```

   출력 예

   ```
   api-int.ocp4.example.com.  0 IN A 192.168.1.5
   ```

   c.

예제 *.apps.<cluster_name>.<base_domain>을 테스트합니다. DNS 와일드카드를 조회합니다. 모든 애플리케이션 와일드카드 조회는 애플리케이션 인그레스 로드 밸런서의 IP 주소로 확인되어야 합니다.

```
$ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.
<base_domain>
```

출력 예

```
random.apps.ocp4.example.com.  0 IN A 192.168.1.5
```

참고

예제 출력에서는 Kubernetes API 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 API 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

random 항목을 다른 와일드카드 값으로 교체할 수 있습니다. 예를 들어 OpenShift Container Platform 콘솔의 경로를 쿼리할 수 있습니다.

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

출력 예

```
console-openshift-console.apps.ocp4.example.com. 0 IN A 192.168.1.5
```

d.

부트스트랩 DNS 레코드 이름에 대해 조회를 실행합니다. 결과가 부트스트랩 노드의 IP 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.
<base_domain>
```

출력 예

```
bootstrap.ocp4.example.com.  0 IN A 192.168.1.96
```

e. 이 방법을 사용하여 컨트롤 플레인 및 컴퓨팅 노드의 DNS 레코드 이름에 대해 조회를 수행합니다. 결과가 각 노드의 IP 주소에 해당하는지 확인합니다.

2. 설치 노드에서 로드 밸런서 및 클러스터 노드의 IP 주소에 대해 역방향 DNS 조회를 실행합니다. 응답에 포함된 레코드 이름이 올바른 구성 요소에 해당하는지 확인합니다.

a. API 로드 밸런서의 IP 주소에 대해 역방향 조회를 수행합니다. 응답에 Kubernetes API 및 Kubernetes 내부 API의 레코드 이름이 포함되어 있는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

출력 예

```
5.1.168.192.in-addr.arpa. 0 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 0 IN PTR api.ocp4.example.com. 2
```

1

2

Kubernetes API의 레코드 이름을 제공합니다.

참고

OpenShift Container Platform 애플리케이션 와일드카드에는 PTR 레코드가 필요하지 않습니다. 애플리케이션 인그레스 로드 밸런서의 IP 주소에 대한 역방향 DNS 확인에는 유효성 검사 단계가 필요하지 않습니다.

b.

부트스트랩 노드의 IP 주소에 대해 역방향 조회를 수행합니다. 결과가 부트스트랩 노드의 DNS 레코드 이름을 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

출력 예

```
96.1.168.192.in-addr.arpa. 0 IN PTR bootstrap.ocp4.example.com.
```

c.

이 방법을 사용하여 컨트롤 플레인 및 컴퓨팅 노드의 IP 주소에 대해 역방향 조회를 수행합니다. 결과가 각 노드의 DNS 레코드 이름과 일치하는지 확인합니다.

12.2.6. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 core 사용자의 ~/.ssh/authorized_keys 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 core로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. ./openshift-install gather 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: ~/.ssh/id_ed25519)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 '~/.ssh 디렉터리에 있는지 확인하십시오.

참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 ed25519 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 rsa 또는 ecdsa 알고리즘을 사용하는 키를 생성합니다.

2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 ~/.ssh/id_ed25519.pub 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 ./openshift-install gather 명령을 사용하려는 경우 필요합니다.

참고

일부 배포에서는 ~/.ssh/id_rsa 및 ~/.ssh/id_dsa와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a.

ssh-agent 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4.

ssh-agent에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1

SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: ~/.ssh/id_ed25519).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

12.2.7. 설치 프로그램 받기

OpenShift Container Platform을 설치하기 전에 프로비저닝 머신에 설치 파일을 다운로드하십시오.

사전 요구 사항

- Linux를 실행하는 머신(예: 로컬 디스크 공간이 500MB인 Red Hat Enterprise Linux 8)이 있습니다.

프로세스

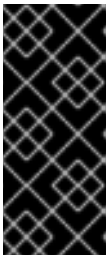
1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.

2. 인프라 공급자를 선택합니다.

3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.

중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.



중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 OpenShift Container Platform 설치 제거 절차를 완료해야 합니다.

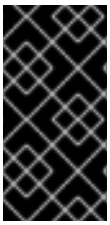
4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

12.2.8. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift CLI(oc)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 oc를 설치할 수 있습니다.



중요

이전 버전의 oc를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 oc를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 Linux에서 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

절차

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.
2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. OpenShift v4.9 Linux Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. oc 바이너리를 PATH에 있는 디렉터리에 배치합니다.

PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치

다음 절차에 따라 Windows에 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

절차

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.
2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. OpenShift v4.9 Windows Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.
4. ZIP 프로그램으로 아카이브의 압축을 풉니다.
5. oc 바이너리를 PATH에 있는 디렉터리로 이동합니다.

   PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

   ```
   C:\> path
   ```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치

다음 절차에 따라 macOS에서 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

절차

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.
2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. OpenShift v4.9 MacOSX Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 해제하고 압축을 풉니다.
5. oc 바이너리 PATH의 디렉터리로 이동합니다.

   PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

   ```
   $ echo $PATH
   ```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
$ oc <command>
```

12.2.9. 수동으로 설치 구성 파일 만들기

OpenShift Container Platform을 사용자가 프로비저닝한 설치의 경우 설치 구성 파일을 수동으로 생성합니다.

사전 요구 사항

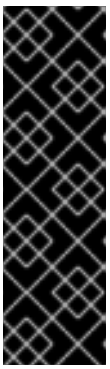
- 로컬 시스템에 설치 프로그램에 제공할 SSH 공개 키가 있습니다. 키는 디버깅 및 재해 복구를 위해 클러스터 노드에 대한 SSH 인증에 사용됩니다.

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿이 있습니다.

절차

1. 필요한 설치 자산을 저장할 설치 디렉터리를 만듭니다.

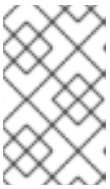
```
$ mkdir <installation_directory>
```



중요

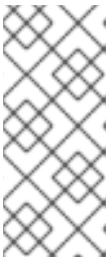
디렉터리를 만들어야 합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

2. 샘플 install-config.yaml 파일 템플릿을 사용자 지정하여 <installation_directory>에 저장합니다.



참고

이 설정 파일의 이름을 install-config.yaml로 지정해야 합니다.



참고

일부 플랫폼 유형의 경우 대체로 ./openshift-install create install-config --dir <installation_directory> 를 실행하여 install-config.yaml 파일을 생성할 수 있습니다. 프롬프트에서 클러스터 구성에 대한 세부 정보를 제공할 수 있습니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 install-config.yaml 파일을 백업합니다.

중요

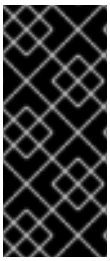
install-config.yaml 파일은 설치 과정의 다음 단계에서 사용됩니다. 이 시점에서 이를 백업해야 합니다.

12.2.9.1. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 환경에 대한 세부 정보를 설명하는 사용자 지정 **install-config.yaml** 설치 구성 파일을 제공합니다.

참고

설치한 후에는 install-config.yaml 파일에서 이러한 매개변수를 수정할 수 없습니다.



중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

12.2.9.1.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 12.9. 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체

매개변수	설명	값
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	소문자, 하이픈(**-**), 마침표(**.**)로 구성되는 문자열(예: **dev**)입니다.
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	{
		"auths":{
		"cloud.openshift.com":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		},
		"quay.io":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		}
		}
		}

12.2.9.1.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 12.10. 네트워크 매개변수

매개변수	설명	값

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체 **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.
networking.clusterNetwork	Pod의 IP 주소 블록입니다. 기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된 **cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.

매개변수	설명	값
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다. 여러 IP 커널 인수를 지정하는 경우 **machineNetwork.cidr** 값은 기본 네트워크의 CIDR이어야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16** **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

12.2.9.1.3. 선택적 구성 매개변수

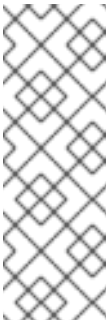
선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 12.11. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.

매개변수	설명	값
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **s390x** (기본값)입니다.	문자열
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은**3**입니다.
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **s390x** (기본값)입니다.	문자열

매개변수	설명	값
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 기본값은 **External**입니다. 이 필드를 **Internal** 로 설정하는 것은 클라우드 이외의 플랫폼에서는 지원되지 않습니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

12.2.9.2. IBM Z의 샘플 install-config.yaml 파일

install-config.yaml 파일을 사용자 지정하여 **OpenShift Container Platform** 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.

```
apiVersion: v1
baseDomain: example.com 1
compute: 2
- hyperthreading: Enabled 3
  name: worker
  replicas: 0 4
  architecture : s390x
controlPlane: 5
  hyperthreading: Enabled 6
  name: master
  replicas: 3 7
  architecture : s390x
metadata:
  name: test 8
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 9
    hostPrefix: 23 10
  networkType: OpenShiftSDN
  serviceNetwork: 11
  - 172.30.0.0/16
platform:
  none: {} 12
fips: false 13
pullSecret: '{"auths": ...}' 14
sshKey: 'ssh-ed25519 AAAA...' 15
```

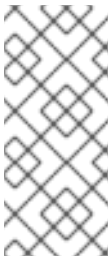
1

2 5

controlPlane 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

3 6

동시 멀티스레딩(**SMT**) 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 **SMT**가 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. **SMT**를 비활성화하는 경우 모든 클러스터 머신에서 이를 비활성화해야 합니다. 여기에는 컨트롤 플레인과 컴퓨팅 머신이 모두 포함됩니다.



참고

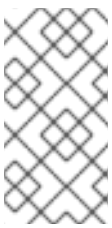
SMT(동시 멀티 스레딩)는 기본적으로 활성화되어 있습니다. **OpenShift Container Platform** 노드에서 **SMT**를 사용할 수 없는 경우 **hyperthreading** 매개변수가 적용되지 않습니다.

중요

OpenShift Container Platform 노드 또는 **install-config.yaml** 파일에서 **hyperthreading**을 비활성화하는 경우 용량 계획에서 머신 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.

4

사용자 프로비저닝 인프라에 **OpenShift Container Platform**을 설치할 때 이 값을 **0**으로 설정해야 합니다. 설치 프로그램에서 제공하는 설치에서 매개 변수는 클러스터가 생성 및 관리하는 컴퓨팅 머신 수를 제어합니다. 사용자 프로비저닝 설치에서는 클러스터 설치를 완료하기 전에 컴퓨팅 시스템을 수동으로 배포해야 합니다.



참고

3-노드 클러스터를 설치하는 경우 **RHCOS(Red Hat Enterprise Linux CoreOS)** 시스템을 설치할 때 컴퓨팅 머신을 배포하지 마십시오.

7

클러스터에 추가하는 컨트롤 플레인 시스템의 수입니다. 클러스터에서 이 값을 클러스터의 **etcd** 끝점 수로 사용하므로 이 값은 배포하는 컨트롤 플레인 시스템의 수와 일치해야 합니다.

8

DNS 레코드에 지정한 클러스터 이름입니다.

9

Pod IP 주소가 할당되는 IP 주소 블록입니다. 이 블록은 기존 물리적 네트워크와 중복되지 않아야합니다. 이러한 IP 주소는 Pod 네트워크에 사용됩니다. 외부 네트워크에서 Pod에 액세스해야 하는 경우, 트래픽을 관리하도록 로드 밸런서와 라우터를 설정해야 합니다.

참고

클래스 E CIDR 범위는 향후 사용을 위해 예약되어 있습니다. 클래스 E CIDR 범위를 사용하려면 네트워킹 환경에서 클래스 E CIDR 범위 내에서 IP 주소를 수락하는지 확인해야 합니다.

10

개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 hostPrefix를 23으로 설정하면 지정된 cidr 이외 /23 서브넷이 각 노드에 할당되어 510(2^(32 - 23) - 2) Pod IP 주소가 허용됩니다. 외부 네트워크에서 노드에 액세스해야 하는 경우 트래픽을 관리하도록 로드 밸런서와 라우터를 구성합니다.

11

서비스 IP 주소에 사용할 IP 주소 풀입니다. IP 주소 풀은 하나만 입력할 수 있습니다. 이 블록은 기존 물리적 네트워크와 중복되지 않아야합니다. 외부 네트워크에서 서비스에 액세스해야 하는 경우, 트래픽을 관리하도록 로드 밸런서와 라우터를 구성합니다.

12

플랫폼을 none으로 설정해야 합니다. IBM Z 인프라에 대한 추가 플랫폼 구성 변수는 지정할 수 없습니다.

중요

플랫폼 유형으로 설치된 클러스터는 Machine API로 컴퓨팅 머신 관리와 같은 일부 기능을 사용할 수 없습니다. 이 제한은 클러스터에 연결된 컴퓨팅 시스템이 일반적으로 기능을 지원하는 플랫폼에 설치된 경우에도 적용됩니다. 이 매개변수는 설치 후 변경할 수 없습니다.

13

FIPS 모드 활성화 또는 비활성화 여부입니다. 기본적으로 FIPS 모드는 비활성화됩니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화

모듈을 대신 사용합니다.

중요

FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 **OpenShift Container Platform** 배포에서만 지원됩니다.

14

Red Hat OpenShift Cluster Manager의 풀 시크릿. 이 풀 시크릿을 사용하면 **OpenShift Container Platform** 구성 요소에 대한 컨테이너 이미지를 제공하는 **Quay.io**를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

15

RHCOS(Red Hat Enterprise Linux CoreOS)의 **core** 사용자에 대한 **SSH** 공용 키입니다.

참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 **OpenShift Container Platform** 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 **SSH** 키를 지정합니다.

12.2.9.3. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 **HTTP** 또는 **HTTPS** 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 **OpenShift Container Platform** 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 **API**에 대한 호출을 포함하여 모든 클러스터 발신(**Egress**) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.

참고

Proxy 오브젝트의 status.noProxy 필드는 설치 구성에 있는 networking.machineNetwork[].cidr, networking.clusterNetwork[].cidr, networking.serviceNetwork[] 필드의 값으로 채워집니다.

Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 Red Hat OpenStack Platform (RHOSP)에 설치하는 경우 Proxy 오브젝트 status.noProxy 필드도 인스턴스 메타데이터 끝점(169.254.169.254)로 채워집니다.

절차

1. install-config.yaml 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

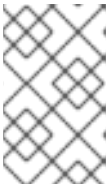
```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1 클러스터 외부에서 HTTP 연결을 구축하는 데 사용할 프록시 URL입니다. URL 스키마는 http여야 합니다.

2 클러스터 외부에서 HTTPS 연결을 구축하는 데 사용할 프록시 URL입니다.

3 대상 도메인 이름, IP 주소 또는 프록시에서 제외할 기타 네트워크 CIDR로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 .을 입력합니다. 예를 들어, .y.com은 x.y.com과 일치하지만 y.com은 일치하지 않습니다. *를 사용하여 모든 대상에 대해 프록시를 바이패스합니다.

4

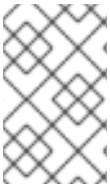


참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 **OpenShift Container Platform**을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 **nil**이 됩니다.



참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

12.2.9.4. 3개의 노드 클러스터 구성

필요한 경우 세 개의 컨트롤 플레인 시스템으로 구성된 베어 메탈 클러스터에 제로 컴퓨팅 머신을 배포할 수 있습니다. 이를 통해 클러스터 관리자와 개발자들이 테스트, 개발, 프로덕션에 사용할 수 있는 소형화되고 리소스 효율이 높은 클러스터를 제공합니다.

3-노드 **OpenShift Container Platform** 환경에서 세 개의 컨트롤 플레인 머신을 예약할 수 있습니다. 즉, 애플리케이션 워크로드가 해당 플랫폼에서 실행되도록 예약됩니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.

절차

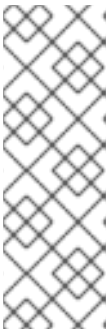
- **install-config.yaml** 파일에서 다음 **compute** 스탠자에 표시된 대로 컴퓨팅 복제본 수가 **0**으로 설정되어 있는지 확인합니다.

```
compute:
- name: worker
  platform: {}
```

```
replicas: 0
```

참고

배포 중인 컴퓨팅 머신 수에 관계없이 사용자 프로비저닝 인프라에 **OpenShift Container Platform**을 설치할 때 컴퓨팅 머신의 **replicas** 매개변수 값을 **0**으로 설정해야 합니다. 설치 프로그램에서 제공하는 설치에서 매개 변수는 클러스터가 생성 및 관리하는 컴퓨팅 머신 수를 제어합니다. 이 설정은 컴퓨팅 시스템이 수동으로 배포되는 사용자 프로비저닝 설치에는 적용되지 않습니다.



참고

컨트롤 플레인 노드의 기본 리소스는 **vCPU 6**개와 **21GB**입니다. **3**개의 컨트롤 플레인 노드의 경우 메모리 + **vCPU**는 최소 **5-**노드 클러스터와 동등합니다. **SMT2**가 활성화된 **IFL 3**개와 함께 각각 **120GB** 디스크에 설치된 **3**개의 노드를 백업해야 합니다. 테스트된 최소 설정은 각 컨트롤 플레인 노드에 대해 **120GB** 디스크에서 **3**개의 **vCPU**와 **10GB**입니다.

3-노드 클러스터 설치의 경우 다음 단계를 따르십시오.

- 컴퓨팅 노드가 **0**인 **3-**노드 클러스터를 배포하는 경우 **Ingress** 컨트롤러 **Pod**는 컨트롤 플레인 노드에서 실행됩니다. **3-**노드 클러스터 배포에서 **HTTP** 및 **HTTPS** 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다. 자세한 내용은 *사용자 프로비저닝 인프라 섹션에 대한 로드 밸런싱 요구 사항* 섹션을 참조하십시오.

- 다음 절차에서 **Kubernetes** 매니페스트 파일을 생성할 때 **<installation_directory>/manifests/cluster-scheduler-02-config.yml** 파일의 **mastersSchedulable** 매개변수가 **true**로 설정되어 있는지 확인합니다. 이렇게 하면 애플리케이션 워크로드를 컨트롤 플레인 노드에서 실행할 수 있습니다.

- **RHCOS(Red Hat Enterprise Linux CoreOS)** 시스템을 생성할 때 컴퓨팅 노드를 배포하지 마십시오.

12.2.10. CNO(Cluster Network Operator) 구성

클러스터 네트워크의 구성은 **CNO(Cluster Network Operator)** 구성의 일부로 지정되며 **cluster**라는 이름의 **CR**(사용자 정의 리소스) 오브젝트에 저장됩니다. **CR**은 **operator.openshift.io API** 그룹에서 **Network API**의 필드를 지정합니다.

CNO 구성은 Network.config.openshift.io API 그룹의 Network API에서 클러스터 설치 중에 다음 필드를 상속하며 이러한 필드는 변경할 수 없습니다.

clusterNetwork

Pod IP 주소가 할당되는 IP 주소 풀입니다.

serviceNetwork

서비스를 위한 IP 주소 풀입니다.

defaultNetwork.type

OpenShift SDN 또는 OVN-Kubernetes와 같은 클러스터 네트워크 공급자입니다.

cluster라는 CNO 오브젝트에서 defaultNetwork 오브젝트의 필드를 설정하여 클러스터의 클러스터 네트워크 공급자 구성을 지정할 수 있습니다.

12.2.10.1. CNO(Cluster Network Operator) 구성 오브젝트

CNO(Cluster Network Operator)의 필드는 다음 표에 설명되어 있습니다.

표 12.12. CNO(Cluster Network Operator) 구성 오브젝트

필드	유형	설명
metadata.name	**string**	CNO 개체 이름입니다. 이 이름은 항상 **cluster**입니다.

spec.clusterNetwork	**array**	Pod IP 주소가 할당되는 IP 주소 블록과 클러스터의 각 개별 노드에 할당된 서브넷 접두사 길이를 지정하는 목록입니다. 예를 들면 다음과 같습니다.
		`spec:`
		`  clusterNetwork:`
		`  - cidr: 10.128.0.0/19`
		`  hostPrefix: 23`
		`  - cidr: 10.128.32.0/19`
		`  hostPrefix: 23`
		매니페스트를 생성하기 전에 **install-config.yaml** 파일에서만 이 필드를 사용자 지정할 수 있습니다. 값은 매니페스트 파일에서 읽기 전용입니다.

필드	유형	설명

spec.serviceNetwork	**array**	서비스의 IP 주소 블록입니다. OpenShift SDN 및 OVN-Kubernetes CNI(Container Network Interface) 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다. 예를 들면 다음과 같습니다.
		`spec:`
		`  serviceNetwork:`
		`  - 172.30.0.0/14`
		매니페스트를 생성하기 전에 **install-config.yaml** 파일에서만 이 필드를 사용자 지정할 수 있습니다. 값은 매니페스트 파일에서 읽기 전용입니다.

spec.defaultNetwork	**object**	클러스터 네트워크의 CNI(Container Network Interface) 클러스터 네트워크 공급자를 구성합니다.
spec.kubeProxyConfig	**object**	이 개체의 필드는 kube-proxy 구성을 지정합니다. OVN-Kubernetes 클러스터 네트워크 공급자를 사용하는 경우 kube-proxy 구성이 적용되지 않습니다.

defaultNetwork 오브젝트 구성

defaultNetwork 오브젝트의 값은 다음 표에 정의되어 있습니다.

표 12.13. defaultNetwork 오브젝트

필드	유형	설명
type	**string**	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, 클러스터 네트워크 공급자가 설치 중에 선택됩니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다. **참고** OpenShift Container Platform은 기본적으로 OpenShift SDN CNI(Container Network Interface) 클러스터 네트워크 공급자를 사용합니다.
openshiftSDNConfig	**object**	이 오브젝트는 OpenShift SDN 클러스터 네트워크 공급자에만 유효합니다.
ovnKubernetesConfig	**object**	이 오브젝트는 OVN-Kubernetes 클러스터 네트워크 공급자에만 유효합니다.

OpenShift SDN CNI 네트워크 공급자에 대한 구성

다음 표에서는 OpenShift SDN Container Network Interface (CNI) 클러스터 네트워크 공급자의 구성 필드를 설명합니다.

표 12.14. openshiftSDNConfig 오브젝트

필드	유형	설명
mode	**string**	OpenShift SDN의 네트워크 격리 모드를 구성합니다. 기본값은 **NetworkPolicy**입니다. **Multitenant** 및 **Subnet** 값은 OpenShift Container Platform 3.x 버전과의 역호환할 수 있지만 권장되지는 않습니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
mtu	**integer**	VXLAN 오버레이 네트워크의 최대 전송 단위(MTU)입니다. 이는 기본 네트워크 인터페이스의 MTU를 기준으로 자동 탐지됩니다. 일반적으로 감지된 MTU를 재정의할 필요는 없습니다. 자동 감지 값이 예상 밖인 경우 노드의 기본 네트워크 인터페이스의 MTU가 올바른지 확인합니다. 이 옵션을 사용하여 노드의 기본 네트워크 인터페이스의 MTU 값을 변경할 수 없습니다. 클러스터에 다른 노드에 대한 다른 MTU 값이 필요한 경우 이 값을 클러스터의 가장 낮은 MTU 값보다 **50** 미만으로 설정해야 합니다. 예를 들어 클러스터의 일부 노드에는 **9001**의 MTU가 있고 일부에는 **1500**의 MTU가 있는 경우 이 값을 **1450**으로 설정해야 합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
vxlanPort	**integer**	모든 VXLAN 패킷에 사용할 포트입니다. 기본값은 **4789**입니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다. 기존 노드가 다른 VXLAN 네트워크에 속하는 가상 환경에서 실행 중인 경우에는 기본값을 변경해야 할 수도 있습니다. 예를 들어 VMware NSX-T 위에서 OpenShift SDN 오버레이를 실행할 때 두 SDN이 동일한 기본 VXLAN 포트 번호를 사용하므로 VXLAN의 대체 포트를 선택해야 합니다. AWS(Amazon Web Services)에서는 포트 **9000**과 포트 **9999** 사이에서 VXLAN의 대체 포트를 선택할 수 있습니다.

OpenShift SDN 구성 예

```
defaultNetwork:
  type: OpenShiftSDN
  openshiftSDNConfig:
    mode: NetworkPolicy
    mtu: 1450
    vxlanPort: 4789
```

OVN-Kubernetes CNI 클러스터 네트워크 공급자에 대한 구성

다음 표에서는 OVN-Kubernetes CNI 클러스터 네트워크 공급자의 구성 필드를 설명합니다.

표 12.15. ovnKubernetesConfig object

필드	유형	설명
mtu	**integer**	Geneve(Generic Network Virtualization Encapsulation) 오버레이 네트워크의 MTU(최대 전송 단위)입니다. 이는 기본 네트워크 인터페이스의 MTU를 기준으로 자동 탐지됩니다. 일반적으로 감지된 MTU를 재정의할 필요는 없습니다. 자동 감지 값이 예상 밖인 경우 노드의 기본 네트워크 인터페이스의 MTU가 올바른지 확인합니다. 이 옵션을 사용하여 노드의 기본 네트워크 인터페이스의 MTU 값을 변경할 수 없습니다. 클러스터에 다른 노드에 대한 다른 MTU 값이 필요한 경우, 이 값을 클러스터의 가장 낮은 MTU 값보다 **100** 미만으로 설정해야 합니다. 예를 들어, 클러스터의 일부 노드에 **9001**의 MTU가 있고 일부에는 **1500**의 MTU가 있는 경우 이 값을**1400**으로 설정해야 합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
genevePort	**integer**	모든 Geneve 패킷에 사용할 포트입니다. 기본값은 **6081**입니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
policyAuditConfig	**object**	네트워크 정책 감사 로깅을 사용자 정의할 구성 오브젝트를 지정합니다. 설정되지 않으면 기본값 감사 로그 설정이 사용됩니다.

표 12.16. policyAuditConfig object

필드	유형	설명
rateLimit	integer	노드당 1초마다 생성할 최대 메시지 수입니다. 기본값은 초당 **20** 개의 메시지입니다.
maxFileSize	integer	감사 로그의 최대 크기(바이트)입니다. 기본값은 **50000000 또는 50** MB입니다.

필드	유형	설명
대상	string	다음 추가 감사 로그 대상 중 하나입니다. **libc** 호스트에서 journald 프로세스의 libc **syslog()** 함수입니다. **udp:<host>:<port>** syslog 서버입니다. **<host>:<port>**를 syslog 서버의 호스트 및 포트로 바꿉니다. **unix:<file>** **<file>**로 지정된 Unix Domain Socket 파일입니다. **null** 감사 로그를 추가 대상으로 보내지 마십시오.
syslogFacility	string	RFC5424에 정의된 **kern**과 같은 syslog 기능입니다. 기본값은 **local0**입니다.

OVN-Kubernetes 구성 예

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
```

kubeProxyConfig 오브젝트 구성

kubeProxyConfig 오브젝트의 값은 다음 표에 정의되어 있습니다.

표 12.17. kubeProxyConfig object

필드	유형	설명
iptablesSyncPeriod	**string**	**iptables** 규칙의 새로 고침 간격입니다. 기본값은 **30s**입니다. 유효 접미사로 **s**, **m**, **h**가 있으며, 자세한 설명은 Go **time** 패키지 문서를 참조하십시오.  **참고** OpenShift Container Platform 4.3 이상에서는 성능이 개선되어 더 이상 **iptablesSyncPeriod** 매개변수를 조정할 필요가 없습니다.

필드	유형	설명
proxyArguments.iptables-min-sync-period	**array**	**iptables** 규칙을 새로 고치기 전 최소 기간입니다. 이 필드를 통해 새로 고침 간격이 너무 짧지 않도록 조정할 수 있습니다. 유효 접미사로 **s**, **m**, **h**가 있으며, 자세한 설명은 Go **time** 패키지를 참조하십시오. 기본값은 다음과 같습니다. kubeProxyConfig: proxyArguments: iptables-min-sync-period: - 0s

12.2.11. Kubernetes 매니페스트 및 Ignition 설정 파일 생성

일부 클러스터 정의 파일을 수정하고 클러스터 시스템을 수동으로 시작해야 하므로 클러스터가 시스템을 구성하는 데 필요한 **Kubernetes** 매니페스트 및 **Ignition** 구성 파일을 사용자가 생성해야 합니다.

설치 구성 파일은 **Kubernetes** 매니페스트로 변환됩니다. 매니페스트는 나중에 클러스터 머신을 구성하는 데 사용되는 **Ignition** 구성 파일로 래핑됩니다.

중요

- **OpenShift Container Platform** 설치 프로그램에서 생성하는 **Ignition** 구성 파일에 **24**시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 **24**시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 **kubelet** 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(**CSR**)을 수동으로 승인해야 합니다. 자세한 내용은 *만료된 컨트롤 플레인 인증서에서 복구* 문서를 참조하십시오.
- 클러스터를 설치한 후 **24**시간에서 **22**시간까지의 인증서가 교체되기 때문에 생성된 후 **12**시간 이내에 **Ignition** 구성 파일을 사용하는 것이 좋습니다. **12**시간 이내에 **Ignition** 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

참고

매니페스트 및 **Ignition** 파일을 생성하는 설치 프로그램은 아키텍처에 따라 다르며 클라이언트 이미지 미러에서 얻을 수 있습니다. 설치 프로그램의 **Linux** 버전은 **s390x**에서만 실행됩니다. 이 설치 프로그램은 **Mac OS** 버전으로도 사용할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램을 가져오셨습니다.
- install-config.yaml 설치 구성 파일을 생성하셨습니다.

프로세스

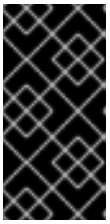
1. OpenShift Container Platform 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터에 대한 Kubernetes 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 <installation_directory>는 사용자가 만든 install-config.yaml 파일이 포함된 설치 디렉터리를 지정합니다.

주의

3 노드 클러스터를 실행 중이면 다음 단계를 건너 뛰어 컨트롤 플레인 노드 일정을 계획할 수 있도록 하십시오.



중요

예약할 수 없는 기본에서 컨트롤 플레인 노드를 구성하면 추가 서브스크립션이 필요합니다. 이는 컨트롤 플레인 노드가 작업자 노드가 되기 때문입니다.

2. <installation_directory>/manifests/cluster-scheduler-02-config.yml Kubernetes 매니페스트 파일의 mastersSchedulable 매개변수가 false로 설정되어 있는지 확인합니다. 이 설정으로 인해 컨트롤 플레인 머신에서 포드가 예약되지 않습니다.

   a. <installation_directory>/manifests/cluster-scheduler-02-config.yml 파일을 엽니다.

b. **mastersSchedulable** 매개변수를 찾아서 값을 **False**로 설정되어 있는지 확인합니다.

c. 파일을 저장하고 종료합니다.

3. **Ignition** 구성 파일을 생성하려면 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행합니다.

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>**는 동일한 설치 디렉터리를 지정합니다.

설치 디렉터리의 부트스트랩, 컨트롤 플레인 및 컴퓨팅 노드에 대한 **Ignition** 구성 파일이 생성됩니다. **kubeadmin-password** 및 **kubeconfig** 파일은 **./<installation_directory>/auth** 디렉터리에 생성됩니다.

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

12.2.12. RHCOS 설치 및 OpenShift Container Platform 부트스트랩 프로세스 시작

프로비저닝하는 **IBM Z** 인프라에 **OpenShift Container Platform**을 설치하려면 **z/VM** 게스트 가상 머신에 **RHCOS(Red Hat Enterprise Linux CoreOS)**를 설치해야 합니다. **RHCOS**를 설치할 때 설치 중인 머신 유형에 대해 **OpenShift Container Platform** 설치 프로그램에서 생성한 **Ignition** 구성 파일을 제공해야 합니다. 적합한 네트워킹, **DNS** 및 로드 밸런싱 인프라를 구성한 경우 **RHCOS z/VM** 게스트 가상 머신이 재부팅된 후 **OpenShift Container Platform** 부트스트랩 프로세스가 자동으로 시작됩니다.

머신을 생성하려면 다음 단계를 완료하십시오.

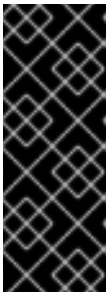
사전 요구 사항

- 생성한 머신에 액세스할 수 있는 프로비저닝 머신에서 실행 중인 **HTTP** 또는 **HTTPS** 서버.

프로세스

1. 프로비저닝 머신에서 **Linux**에 로그인합니다.

2. RHCOS 이미지 미러 에서 **RHCOS(Red Hat Enterprise Linux CoreOS)** 커널, **initramfs** 및 **rootfs** 파일을 가져옵니다.

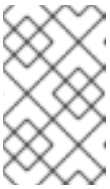


중요

RHCOS 이미지는 **OpenShift Container Platform** 릴리스에 따라 변경되지 않을 수 있습니다. 설치하는 **OpenShift Container Platform** 버전과 같거나 그 이하의 버전 중 가장 최신 버전의 이미지를 다운로드해야 합니다. 이 프로세스에는 아래 설명된 적절한 **kernel, initramfs** 및 **rootfs** 아티팩트만 사용하십시오.

OpenShift Container Platform 버전 번호가 파일 이름에 포함됩니다. 다음 예와 유사합니다.

- **kernel: rhcos-<version>-live-kernel-<architecture>**
- **initramfs: rhcos-<version>-live-initramfs.<architecture>.img**
- **rootfs: rhcos-<version>-live-rootfs.<architecture>.img**



참고

rootfs 이미지는 **FCP** 및 **DASD**에 대해 동일합니다.

3. 매개 변수 파일을 생성합니다. 다음 매개 변수는 특정 가상 머신에 지정해야 합니다.

- **ip=**에 다음 **7** 개의 항목을 지정하십시오.

  i. 컴퓨터의 **IP** 주소

ii. 빈 문자열

iii. 게이트웨이

iv. 넷 마스크

v. **hostname.domainname** 형식의 시스템 호스트 및 도메인 이름. **RHCOS**가 설정하도록 하려면 이 값을 생략하십시오.

vi. 네트워크 인터페이스 이름. **RHCOS**가 설정하도록 하려면 이 값을 생략하십시오.

vii. 고정 **IP** 주소를 사용하는 경우 **none**을 지정합니다.

- **coreos.inst.ignition_url=**의 경우 시스템 역할의 **Ignition** 파일을 지정합니다. **bootstrap.ign, master.ign** 또는 **worker.ign**을 사용하십시오. **HTTP** 및 **HTTPS** 프로토콜만 지원됩니다.

- **coreos.live.rootfs_url=**의 경우 부팅 중인 커널 및 **initramfs**와 일치하는 **rootfs** 아티팩트를 지정합니다. **HTTP** 및 **HTTPS** 프로토콜만 지원됩니다.

- **DASD** 유형 디스크에 설치하려면 다음 작업을 완료합니다.

i. **coreos.inst.install_dev=**의 경우 **dasda**를 지정합니다.

ii. **rd.dasd=**의 경우 **RHCOS**를 설치할 **DASD**를 지정합니다.

iii. 변경되지 않은 다른 모든 매개변수는 그대로 두십시오.

부트스트랩 시스템의 매개 변수 파일 예 **bootstrap-0.parm**

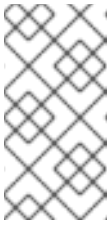
```
rd.neednet=1 \
console=ttysclp0 \
```

```
coreos.inst.install_dev=dasda \
coreos.live.rootfs_url=http://cl1.provide.example.com:8080/assets/rhcos-live-rootfs.s390x.img \
coreos.inst.ignition_url=http://cl1.provide.example.com:8080/ignition/bootstrap.ign \
ip=172.18.78.2::172.18.78.1:255.255.255.0:::none nameserver=172.18.78.1 \
rd.znet=qeth,0.0.bdf0,0.0.bdf1,0.0.bdf2,layer2=1,portno=0 \
zfcp.allow_lun_scan=0 \
rd.dasd=0.0.3490
```

매개 변수 파일의 모든 옵션을 한 줄로 작성하고 줄 바꿈 문자가 없는지 확인합니다.

- FCP 유형 디스크에 설치하려면 다음 작업을 완료합니다.

i. RHCOS를 설치할 FCP 디스크를 지정하려면 rd.zfcp=<adapter>,<wwpn>,<lun>을 사용합니다. 멀티패스의 경우 추가 경로마다 이 단계를 반복합니다.



참고

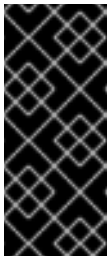
여러 경로를 사용하여 설치할 때 나중에 문제가 발생할 수 있으므로 설치 후에 직접 멀티패스를 활성화해야 합니다.

ii. 설치 장치를 coreos.inst.install_dev=sda로 설정합니다.

참고

NPIV로 추가 LUN을 구성하는 경우 FCP에는 zfcp.allow_lun_scan=0이 필요합니다. 예를 들어 CSI 드라이버를 사용하므로 zfcp.allow_lun_scan=1 을 활성화해야 하는 경우, 각 노드가 다른 노드의 부팅 파티션에 액세스할 수 없도록 NPIV를 구성해야 합니다.

iii. 변경되지 않은 다른 모든 매개변수는 그대로 두십시오.



중요

멀티패스를 완전히 활성화하려면 추가 설치 후 단계가 필요합니다. 자세한 내용은 *설치 후 머신 구성 작업*의 "RHCOS에서 커널 인수를 사용하여 다중 경로 활성화"를 참조하십시오.

다음은 다중 경로가 있는 작업자 노드의 예제 매개변수 파일 **worker-1.parm**입니다.

```
rd.neednet=1 \
console=ttysclp0 \
coreos.inst.install_dev=sda \
coreos.live.rootfs_url=http://cl1.provide.example.com:8080/assets/rhcos-live-rootfs.s390x.img \
coreos.inst.ignition_url=http://cl1.provide.example.com:8080/ignition/worker.ign \
ip=172.18.78.2::172.18.78.1:255.255.255.0:::none nameserver=172.18.78.1 \
rd.znet=qeth,0.0.bdf0,0.0.bdf1,0.0.bdf2,layer2=1,portno=0 \
zfcp.allow_lun_scan=0 \
rd.zfcp=0.0.1987,0x50050763070bc5e3,0x4008400B00000000 \
rd.zfcp=0.0.19C7,0x50050763070bc5e3,0x4008400B00000000 \
rd.zfcp=0.0.1987,0x50050763071bc5e3,0x4008400B00000000 \
rd.zfcp=0.0.19C7,0x50050763071bc5e3,0x4008400B00000000
```

매개 변수 파일의 모든 옵션을 한 줄로 작성하고 줄 바꿈 문자가 없는지 확인합니다.

4. **initramfs**, 커널, 매개 변수 파일 및 **RHCOS** 이미지를 **z/VM**에 전송합니다 (예: **FTP** 사용). **FTP**를 사용하여 파일을 전송하고 가상 리더에서 부팅하는 방법에 대한 자세한 내용은 **Z/VM**에서 설치를 참조하십시오.

5. 부트스트랩 노드가 될 **z/VM** 게스트 가상 머신의 가상 리더에 파일 **punch**를 실행합니다.

IBM 문서의 **PUNCH**를 참조하십시오.

작은 정보

CP PUNCH 명령을 사용하거나 **Linux**를 사용하는 경우 **vmur** 명령을 사용하여 두 개의 **z/VM** 게스트 가상 머신간에 파일을 전송할 수 있습니다.

6. 부트스트랩 시스템에서 **CMS**에 로그인합니다.

7. 리더에서 부트스트랩 머신에 대해 **IPL**을 수행합니다.

```
$ ipl c
```

IBM 문서의 **IPL** 을 참조하십시오.

8. 클러스터의 다른 컴퓨터에 대해 이 프로세스를 반복합니다.

12.2.12.1. 고급 RHCOS 설치 참조

여기서는 **RHCOS(Red Hat Enterprise Linux CoreOS)** 수동 설치 프로세스를 수정하는 데 사용할 수 있는 네트워킹 구성 및 기타 고급 옵션에 대해 설명합니다. 다음 표에서는 **RHCOS** 라이브 설치 프로그램 및 **coreos-installer** 명령과 함께 사용할 수있는 커널 인수 및 명령 줄 옵션에 대해 설명합니다.

12.2.12.1.1. ISO 설치를 위한 네트워킹 및 본딩 옵션

ISO 이미지에서 **RHCOS**를 설치하는 경우, 해당 이미지를 부팅할 때 수동으로 커널 인수를 추가하여 노드의 네트워킹을 구성할 수 있습니다. 네트워킹 인수를 지정하지 않으면 **RHCOS**에서 **Ignition** 구성 파일을 가져오는 데 네트워킹이 필요함을 감지하면 **initramfs**에서 **DHCP**가 활성화됩니다.

중요

네트워킹 인수를 수동으로 추가할 때 **initramfs**에서 네트워크를 가져오려면 **rd.neednet=1** 커널 인수도 추가해야 합니다.

다음 표는 **ISO** 설치를 위해 **RHCOS(Red Hat Enterprise Linux CoreOS)** 노드의 네트워킹 및 본딩 구성 예를 보여줍니다. 예제에서는 **ip=, nameserver=, bond=** 커널 인수를 사용하는 방법을 설명합니다.

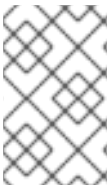
참고

커널 인수를 추가할 때 순서가 중요합니다: **ip=, nameserver=** 및 **bond=** 입니다.

이는 시스템 부팅 중에 **dracut** 툴로 전달되는 네트워킹 옵션입니다. **dracut**에서 지원하는 네트워킹

옵션에 대한 자세한 내용은 **dracut.cmdline** 메뉴얼 페이지를 참조하십시오.

다음 정보는 **ISO** 설치를 위해 **RHCOS** 노드에서 네트워킹을 구성하는 예제를 제공합니다. 예제에서는 **ip=** 및 **nameserver=** 커널 인수를 사용하는 방법을 설명합니다.



참고

커널 인수를 추가할 때 순서가 중요합니다(**ip =** 및 **nameserver= **).

이는 시스템 부팅 중에 **dracut** 툴로 전달되는 네트워킹 옵션입니다. **dracut**에서 지원하는 네트워킹 옵션에 대한 자세한 내용은 **dracut.cmdline** 메뉴얼 페이지를 참조하십시오.

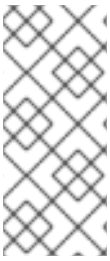
다음 예제는 **ISO** 설치를 위한 네트워킹 옵션입니다.

DHCP 또는 고정 **IP** 주소 구성

IP 주소를 구성하려면 **DHCP(ip=dhcp)**를 사용하거나 개별 고정 **IP** 주소**(ip=<host_ip>)**를 설정합니다. 정적 **IP**를 설정하는 경우 각 노드에서 **DNS** 서버 **IP** 주소 **(nameserver=<dns_ip>)**를 확인합니다. 다음 예제에서는 다음을 설정합니다.

- 노드의 **IP** 주소는 **10.10.10.2**로 설정됩니다.
- 게이트웨이 주소는 **10.10.10.254**로 설정됩니다.
- 넷마스크는 **255.255.255.0**으로
- **core0.example.com**에 대한 호스트 이름
- **4.4.4.41**의 **DNS** 서버 주소
- **auto-configuration** 값을 **none**으로 설정합니다. **IP** 네트워킹이 정적으로 구성되면 자동 구성이 필요하지 않습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
nameserver=4.4.4.41
```



참고

DHCP를 사용하여 RHCOS 시스템의 IP 주소 지정을 구성하는 경우 시스템은 DHCP를 통해 DNS 서버 정보도 가져옵니다. DHCP 기반 배포의 경우 DHCP 서버 구성을 통해 RHCOS 노드에서 사용할 DNS 서버 주소를 정의할 수 있습니다.

고정 호스트 이름 없이 IP 주소 구성

정적 호스트 이름을 할당하지 않고 IP 주소를 구성할 수 있습니다. 사용자가 정적 호스트 이름을 설정하지 않으면 역방향 DNS 조회에 의해 자동으로 선택됩니다. 정적 호스트 이름이 없는 IP 주소를 구성하려면 다음 예제를 참조합니다.

- 노드의 IP 주소는 10.10.10.2로 설정됩니다.
- 게이트웨이 주소는 10.10.10.254로 설정됩니다.
- 넷마스크는 255.255.255.0으로
- 4.4.4.41의 DNS 서버 주소
- auto-configuration 값을 none으로 설정합니다. IP 네트워킹이 정적으로 구성되면 자동 구성이 필요하지 않습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0::enp1s0:none
nameserver=4.4.4.41
```

여러 네트워크 인터페이스 지정

여러 ip= 항목을 설정하여 여러 네트워크 인터페이스를 지정할 수 있습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

기본 게이트웨이 및 경로 구성

선택 사항: rd.route= 값을 설정하여 추가 네트워크에 대한 경로를 구성할 수 있습니다.

참고

하나 이상의 네트워크를 구성할 때 하나의 기본 게이트웨이가 필요합니다. 추가 네트워크 게이트웨이가 기본 네트워크 게이트웨이와 다른 경우 기본 게이트웨이가 기본 네트워크 게이트웨이어야 합니다.

- 다음 명령을 실행하여 기본 게이트웨이를 구성합니다.

```
ip=::10.10.10.254::::
```

- 다음 명령을 입력하여 추가 네트워크의 경로를 구성합니다.

```
rd.route=20.20.20.0/24:20.20.20.254:enp2s0
```

단일 인터페이스에서 DHCP 비활성화

두 개 이상의 네트워크 인터페이스가 있고 하나의 인터페이스만 사용되는 경우와 같이 단일 인터페이스에서 DHCP를 비활성화할 수 있습니다. 이 예에서 enp1s0 인터페이스에는 정적 네트워킹 구성이 있으며 enp2s0 용으로 DHCP가 사용되지 않습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=::::core0.example.com:enp2s0:none
```

DHCP 및 고정 IP 구성 결합

다음과 같이 시스템의 DHCP 및 고정 IP 구성을 여러 네트워크 인터페이스와 결합할 수 있습니다.

```
ip=enp1s0:dhcp
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

개별 인터페이스에서 VLAN 구성

선택 사항: vlan= 매개변수를 사용하여 개별 인터페이스에서 VLAN을 구성할 수 있습니다.

- 네트워크 인터페이스에서 VLAN을 구성하고 고정 IP 주소를 사용하려면 다음 명령을 실행합니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0.100:none
vlan=enp2s0.100:enp2s0
```

- 네트워크 인터페이스에서 VLAN을 구성하고 DHCP를 사용하려면 다음 명령을 실행합니다.

```
ip=enp2s0.100:dhcp
vlan=enp2s0.100:enp2s0
```

여러 DNS 서버 제공

각 서버에 nameserver= 항목을 추가하여 여러 DNS 서버를 제공할 수 있습니다. 예를 들면 다음과 같습니다.

```
nameserver=1.1.1.1
nameserver=8.8.8.8
```

단일 인터페이스에 여러 네트워크 인터페이스 본딩

선택 사항: bond= 옵션을 사용하여 여러 네트워크 인터페이스를 단일 인터페이스에 결합할 수 있습니다. 다음 예제를 참조하십시오.

- 결합된 인터페이스를 구성하는 구문: bond = name [: network_interfaces] [: options]

  *name*은 결합하는 기기 이름(bond0)이고 *network_interfaces*는 쉼표로 구분되는 물리적(이더넷) 인터페이스 목록(em1, em2)이며, *options*은 쉼표로 구분되는 결합 옵션 목록입니다. 사용 가능한 옵션을 보려면 modinfo bonding을 입력하십시오.

- bond=를 사용하여 결합된 인터페이스를 생성할 때 IP 주소가 할당되는 방법과 결합된 인터페이스에 대한 기타 정보를 지정해야 합니다.

- DHCP를 사용하도록 결합된 인터페이스를 구성하려면 bond의 IP 주소를 dhcp로 설정하십시오. 예를 들면 다음과 같습니다.

  ```
  bond=bond0:em1,em2:mode=active-backup
  ip=bond0:dhcp
  ```

- 고정 IP 주소를 사용하도록 결합된 인터페이스를 구성하려면 원하는 특정 IP 주소 및 관련 정보를 입력합니다. 예를 들면 다음과 같습니다.

```
bond=bond0:em1,em2:mode=active-backup,fail_over_mac=1
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
```

공유 OSA/RoCE 카드가 사용될 때 문제를 방지하기 위해 항상 active-backup 모드에서 옵션 fail_over_mac=1 을 설정합니다.

단일 인터페이스에 여러 네트워크 인터페이스 본딩

선택 사항: vlan= 매개변수를 사용하고 DHCP를 사용하도록 결합된 인터페이스에서 VLAN을 구성할 수 있습니다. 예를 들면 다음과 같습니다.

```
ip=bond0.100:dhcp
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

다음 예제를 사용하여 VLAN을 사용하여 결합된 인터페이스를 구성하고 고정 IP 주소를 사용합니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0.100:none
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

네트워크 팀 구성 사용

선택 사항: team= 매개변수를 사용하여 네트워크 팀을 결합의 대안으로 사용할 수 있습니다.

- 팀 인터페이스를 구성하는 구문은 team=name[:network_interfaces]입니다.

  name 은 팀 장치 이름(team0)이고 *network_interfaces* 는 쉼표로 구분된 실제(이더넷) 인터페이스 목록(em1, em2)을 나타냅니다.

RHCOS가 향후 RHEL 버전으로 전환되면 티밍이 더 이상 사용되지 않을 예정입니다. 자세한 내용은 이 Red Hat 지식베이스 문서를 참조하십시오.

다음 예제를 사용하여 네트워크 팀을 구성합니다.

```
team=team0:em1,em2
ip=team0:dhcp
```

12.2.13. 부트스트랩 프로세스가 완료될 때까지 대기 중

OpenShift Container Platform 부트스트랩 프로세스는 클러스터 노드가 먼저 디스크에 설치된 영구 RHCOS 환경으로 부팅된 후에 시작됩니다. Ignition 구성 파일을 통해 제공되는 구성 정보는 부트스트랩 프로세스를 초기화하고 머신에 OpenShift Container Platform을 설치하는 데 사용됩니다. 부트스트랩 프로세스가 완료될 때까지 기다려야 합니다.

사전 요구 사항

- 클러스터에 대한 Ignition 구성 파일이 생성되어 있습니다.
- 적합한 네트워크, DNS 및 로드 밸런싱 인프라가 구성되어 있습니다.
- 설치 프로그램을 받아서 클러스터의 Ignition 구성 파일을 생성했습니다.
- 클러스터 머신에 RHCOS를 설치하고 OpenShift Container Platform 설치 프로그램에서 생성된 Ignition 구성 파일을 제공했습니다.
- 사용자 시스템에서 직접 인터넷에 액세스하거나 HTTP 또는 HTTPS 프록시를 사용할 수 있습니다.

절차

1. 부트스트랩 프로세스를 모니터링합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

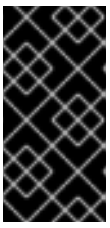
2 다른 설치 세부 사항을 보려면 info 대신 warn, debug 또는 error를 지정합니다.

출력 예

```
INFO Waiting up to 30m0s for the Kubernetes API at
https://api.test.example.com:6443...
INFO API v1.22.1 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

이 명령은 **Kubernetes API** 서버가 컨트롤 플레인 시스템에서 부트스트랩되었다는 신호를 보낼 때 성공합니다.

2. 부트스트랩 프로세스가 완료된 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다.



중요

이 시점에 로드 밸런서에서 부트스트랩 시스템을 제거해야 합니다. 부트스트랩 머신 자체를 제거하거나 다시 포맷할 수도 있습니다.

12.2.14. CLI를 사용하여 클러스터에 로그인

클러스터 **kubeconfig** 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. **kubeconfig** 파일에는 **CLI**에서 올바른 클러스터 및 **API** 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 **OpenShift Container Platform** 설치 과정에서 생성됩니다.

사전 요구 사항

- **OpenShift Container Platform** 클러스터를 배포했습니다.
- **oc CLI**를 설치했습니다.

절차

1. **kubeadmin** 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 **oc** 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

12.2.15. 머신의 인증서 서명 요청 승인

클러스터에 시스템을 추가하면 추가한 시스템별로 보류 중인 인증서 서명 요청(CSR)이 두 개씩 생성됩니다. 이러한 CSR이 승인되었는지 확인해야 하며, 필요한 경우 이를 직접 승인해야 합니다. 클라이언트 요청을 먼저 승인한 다음 서버 요청을 승인해야 합니다.

사전 요구 사항

- 클러스터에 시스템을 추가했습니다.

절차

1. 클러스터가 시스템을 인식하는지 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0  Ready    master  63m  v1.22.1
master-1  Ready    master  63m  v1.22.1
master-2  Ready    master  64m  v1.22.1
```

출력에 생성된 모든 시스템이 나열됩니다.

참고

이전 출력에는 일부 **CSR**이 승인될 때까지 컴퓨팅 노드(작업자 노드라고도 함)가 포함되지 않을 수 있습니다.

2.

보류 중인 **CSR**을 검토하고 클러스터에 추가한 각 시스템에 대해 **Pending** 또는 **Approved** 상태의 클라이언트 및 서버 요청이 표시되는지 확인합니다.

```
$ oc get csr
```

출력 예

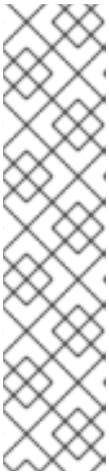
```
NAME        AGE   REQUESTOR                                   CONDITION
csr-mddf5   20m   system:node:master-01.example.com   Approved,Issued
csr-z5rln   16m   system:node:worker-21.example.com   Approved,Issued
```

3.

CSR이 승인되지 않은 경우, 추가된 시스템에 대한 모든 보류 중인 **CSR**이 **Pending** 상태로 전환된 후 클러스터 시스템의 **CSR**을 승인합니다.

참고

CSR은 교체 주기가 자동으로 만료되므로 클러스터에 시스템을 추가한 후 **1** 시간 이내에 **CSR**을 승인하십시오. 한 시간 내에 승인하지 않으면 인증서가 교체되고 각 노드에 대해 두 개 이상의 인증서가 표시됩니다. 이러한 인증서를 모두 승인해야 합니다. 클라이언트 **CSR**이 승인되면 **Kubelet**은 인증서에 대한 보조 **CSR**을 생성하므로 수동 승인이 필요합니다. 그러면 **Kubelet**에서 동일한 매개변수를 사용하여 새 인증서를 요청하는 경우 인증서 갱신 요청은 **machine-approver**에 의해 자동으로 승인됩니다.



참고

베어 메탈 및 기타 사용자 프로비저닝 인프라와 같이 머신 **API**를 사용하도록 활성화되지 않는 플랫폼에서 실행되는 클러스터의 경우 **CSR(Kubelet service Certificate Request)**을 자동으로 승인하는 방법을 구현해야 합니다. 요청이 승인되지 않으면 **API** 서버가 **kubelet**에 연결될 때 서비스 인증서가 필요하므로 **oc exec, oc rsh, oc logs** 명령을 성공적으로 수행할 수 없습니다. **Kubelet** 엔드 포인트에 연결하는 모든 작업을 수행하려면 이 인증서 승인이 필요합니다. 이 방법은 새 **CSR**을 감시하고 **CSR**이 **system:node** 또는 **system:admin** 그룹의 **node-bootstrapper** 서비스 계정에 의해 제출되었는지 확인하고 노드의 **ID**를 확인합니다.

- 개별적으로 승인하려면 유효한 **CSR** 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1

<csr_name>은 현재 **CSR** 목록에 있는 **CSR**의 이름입니다.

- 보류 중인 **CSR**을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

참고

일부 **Operator**는 일부 **CSR**이 승인될 때까지 사용할 수 없습니다.

4. 이제 클라이언트 요청이 승인되었으므로 클러스터에 추가한 각 머신의 서버 요청을 검토해야 합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE    REQUESTOR                                                   CONDITION
csr-bfd72   5m26s  system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
```

```
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 나머지 CSR이 승인되지 않고 Pending 상태인 경우 클러스터 머신의 CSR을 승인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 <csr_name>은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs oc adm certificate approve
```

6. 모든 클라이언트 및 서버 CSR이 승인된 후 머신은 Ready 상태가 됩니다. 다음 명령을 실행하여 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0  Ready    master  73m  v1.22.1
master-1  Ready    master  73m  v1.22.1
master-2  Ready    master  74m  v1.22.1
worker-0  Ready    worker  11m  v1.22.1
worker-1  Ready    worker  11m  v1.22.1
```

참고

머신이 **Ready** 상태로 전환하는 데 서버 **CSR**의 승인 후 몇 분이 걸릴 수 있습니다.

추가 정보

- **CSR**에 대한 자세한 내용은 인증서 서명 요청을 참조하십시오.

12.2.16. Operator의 초기 설정

컨트롤 플레인이 초기화된 후 일부 **Operator**를 즉시 구성하여 모두 사용 가능하도록 해야 합니다.

사전 요구 사항

- 컨트롤 플레인이 초기화되어 있습니다.

프로세스

1. 클러스터 구성 요소가 온라인 상태인지 확인합니다.

```
$ watch -n5 oc get clusteroperators
```

출력 예

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.9.0     True        False         False      19m
baremetal                                  4.9.0     True        False         False      37m
cloud-credential                           4.9.0     True        False         False      40m
cluster-autoscaler                         4.9.0     True        False         False      37m
config-operator                            4.9.0     True        False         False      38m
console                                    4.9.0     True        False         False      26m
csi-snapshot-controller                    4.9.0     True        False         False      37m
dns                                        4.9.0     True        False         False      37m
etcd                                       4.9.0     True        False         False      36m
image-registry                             4.9.0     True        False         False      31m
ingress                                    4.9.0     True        False         False      30m
insights                                   4.9.0     True        False         False      31m
kube-apiserver                             4.9.0     True        False         False      26m
kube-controller-manager                    4.9.0     True        False         False      36m
kube-scheduler                             4.9.0     True        False         False      36m
```

```
kube-storage-version-migrator              4.9.0     True        False         False      37m
machine-api                                4.9.0     True        False         False      29m
machine-approver                           4.9.0     True        False         False      37m
machine-config                             4.9.0     True        False         False      36m
marketplace                                4.9.0     True        False         False      37m
monitoring                                 4.9.0     True        False         False      29m
network                                    4.9.0     True        False         False      38m
node-tuning                                4.9.0     True        False         False      37m
openshift-apiserver                        4.9.0     True        False         False      32m
openshift-controller-manager               4.9.0     True        False         False      30m
openshift-samples                          4.9.0     True        False         False      32m
operator-lifecycle-manager                 4.9.0     True        False         False      37m
operator-lifecycle-manager-catalog         4.9.0     True        False         False      37m
operator-lifecycle-manager-packageserver   4.9.0     True        False         False      32m
service-ca                                 4.9.0     True        False         False      38m
storage                                    4.9.0     True        False         False      37m
```

2. 사용할 수 없는 **Operator**를 구성합니다.

12.2.16.1. 이미지 레지스트리 스토리지 구성

기본 스토리지를 제공하지 않는 플랫폼에서는 처음에 **Image Registry Operator**를 사용할 수 없습니다. 설치한 후에 스토리지를 사용하도록 레지스트리를 구성하여 **Registry Operator**를 사용 가능하도록 만들어야 합니다.

프로덕션 클러스터에 필요한 영구 볼륨을 구성하는 과정의 지침이 표시됩니다. 해당하는 경우, 프로덕션 환경 외 클러스터에서만 사용할 수 있는 저장 위치로서 빈 디렉터리를 구성하는 과정의 지침이 표시됩니다.

업그레이드 중에 **Recreate** 롤아웃 전략을 사용하여 이미지 레지스트리의 블록 스토리지 유형 사용을 허용하기 위한 추가 지침이 제공됩니다.

12.2.16.1.1. IBM Z용 레지스트리 스토리지 구성

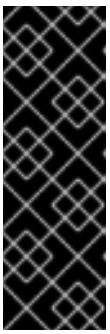
클러스터 관리자는 설치한 후 스토리지를 사용하도록 레지스트리를 구성해야 합니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

- IBM Z에 클러스터가 있습니다.

- Red Hat OpenShift Container Storage와 같이 클러스터용 영구 스토리지 프로비저닝이 있습니다.



중요

OpenShift Container Platform은 복제본이 하나만 있는 경우 이미지 레지스트리 스토리지에 대한 ReadWriteOnce 액세스를 지원합니다. ReadWriteOnce 액세스에서는 레지스트리가 Recreate 롤아웃 전략을 사용해야 합니다. 두 개 이상의 복제본으로 고 가용성을 지원하는 이미지 레지스트리를 배포하려면 ReadWriteMany 액세스가 필요합니다.

- "100Gi" 용량이 필요합니다.

프로세스

1. 스토리지를 사용하도록 레지스트리를 구성하기 위해 configs.imageregistry/cluster 리소스에서 spec.storage.pvc를 변경합니다.

참고

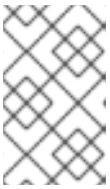
공유 스토리지를 사용할 때 보안 설정을 확인하여 외부에서의 액세스를 방지합니다.

2. 레지스트리 pod가 없는지 확인합니다.

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

출력 예

```
No resourses found in openshift-image-registry namespace
```



참고

출력에 레지스트리 **Pod**가 있는 경우 이 절차를 계속할 필요가 없습니다.

3. 레지스트리 구성을 확인합니다.

```
$ oc edit configs.imageregistry.operator.openshift.io
```

출력 예

```
storage:
  pvc:
    claim:
```

image-registry-storage PVC의 자동 생성을 허용하도록 **claim** 필드를 비워 둡니다.

4. **clusteroperator** 상태를 확인합니다.

```
$ oc get clusteroperator image-registry
```

출력 예

```
NAME              VERSION   AVAILABLE   PROGRESSING   DEGRADED   SINCE   MESSAGE
image-registry    4.9       True        False         False      6h50m
```

5. 이미지를 빌드 및 푸시할 수 있도록 레지스트리의 관리가 설정되어 있는지 확인하십시오.

- 다음을 실행합니다.

```
$ oc edit configs.imageregistry/cluster
```

다음으로 라인을 변경하십시오.

```
managementState: Removed
```

다음으로 변경

```
managementState: Managed
```

12.2.16.1.2. 프로덕션 환경 외 클러스터에서 이미지 레지스트리의 스토리지 구성

이미지 레지스트리 Operator에 대한 스토리지를 구성해야 합니다. 프로덕션 환경 외 클러스터의 경우, 이미지 레지스트리를 빈 디렉터리로 설정할 수 있습니다. 이렇게 하는 경우 레지스트리를 다시 시작하면 모든 이미지가 손실됩니다.

프로세스

- 이미지 레지스트리 스토리지를 빈 디렉터리로 설정하려면 다음을 수행하십시오.

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch
'{"spec":{"storage":{"emptyDir":{}}}}'
```

주의

프로덕션 환경 외 클러스터에 대해서만 이 옵션을 구성하십시오.

Image Registry Operator가 구성 요소를 초기화하기 전에 이 명령을 실행하면 oc patch 명령이 실패하며 다음 오류가 발생합니다.

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster"
not found
```

몇 분 후에 명령을 다시 실행하십시오.

12.2.17. 사용자 프로비저닝 인프라에 설치 완료

Operator 구성을 완료한 후 제공하는 인프라에 클러스터 설치를 완료할 수 있습니다.

사전 요구 사항

- 컨트롤 플레인이 초기화되어 있습니다.
- 초기 Operator 구성을 완료해야 합니다.

프로세스

1. 다음 명령을 사용하여 모든 클러스터 구성 요소가 온라인 상태인지 확인합니다.

```
$ watch -n5 oc get clusteroperators
```

출력 예

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.9.0     True        False         False      19m
baremetal                                  4.9.0     True        False         False      37m
cloud-credential                           4.9.0     True        False         False      40m
cluster-autoscaler                         4.9.0     True        False         False      37m
config-operator                            4.9.0     True        False         False      38m
console                                    4.9.0     True        False         False      26m
csi-snapshot-controller                    4.9.0     True        False         False      37m
dns                                        4.9.0     True        False         False      37m
etcd                                       4.9.0     True        False         False      36m
image-registry                             4.9.0     True        False         False      31m
ingress                                    4.9.0     True        False         False      30m
insights                                   4.9.0     True        False         False      31m
kube-apiserver                             4.9.0     True        False         False      26m
kube-controller-manager                    4.9.0     True        False         False      36m
kube-scheduler                             4.9.0     True        False         False      36m
kube-storage-version-migrator              4.9.0     True        False         False      37m
machine-api                                4.9.0     True        False         False      29m
machine-approver                           4.9.0     True        False         False      37m
machine-config                             4.9.0     True        False         False      36m
marketplace                                4.9.0     True        False         False      37m
monitoring                                 4.9.0     True        False         False      29m
network                                    4.9.0     True        False         False      38m
node-tuning                                4.9.0     True        False         False      37m
openshift-apiserver                        4.9.0     True        False         False      32m
```

```
openshift-controller-manager             4.9.0     True        False         False      30m
openshift-samples                        4.9.0     True        False         False      32m
operator-lifecycle-manager               4.9.0     True        False         False      37m
operator-lifecycle-manager-catalog       4.9.0     True        False         False      37m
operator-lifecycle-manager-packageserver 4.9.0     True        False         False      32m
service-ca                               4.9.0     True        False         False      38m
storage                                  4.9.0     True        False         False      37m
```

또는 다음 명령은 모든 클러스터를 사용할 수 있을 때 알립니다. 또한 인증 정보를 검색하고 표시합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

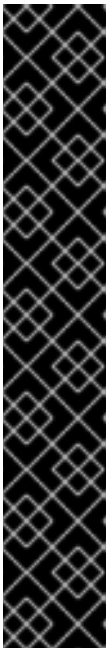
1

<installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

출력 예

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator가 Kubernetes API 서버에서 OpenShift Container Platform 클러스터 배포를 완료하면 명령이 성공합니다.



중요

- 설치 프로그램에서 생성하는 **Ignition** 구성 파일에 **24**시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 **24**시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 **kubelet** 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(**CSR**)을 수동으로 승인해야 합니다. 자세한 내용은 *만료된 컨트롤 플레인 인증서에서 복구* 문서를 참조하십시오.

- 클러스터를 설치한 후 **24**시간에서 **22**시간까지의 인증서가 교체되기 때문에 생성된 후 **12**시간 이내에 **Ignition** 구성 파일을 사용하는 것이 좋습니다. **12**시간 이내에 **Ignition** 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

2. **Kubernetes API** 서버가 **Pod**와 통신하고 있는지 확인합니다.

   a. 모든 **Pod** 목록을 보려면 다음 명령을 사용하십시오.

   ```
   $ oc get pods --all-namespaces
   ```

   출력 예

   ```
   NAMESPACE                       NAME                                        READY   STATUS
   RESTARTS   AGE
   openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8
   1/1     Running   1          9m
   openshift-apiserver             apiserver-67b9g                             1/1     Running   0
   3m
   openshift-apiserver             apiserver-ljcmx                             1/1     Running   0
   1m
   openshift-apiserver             apiserver-z25h4                             1/1     Running   0
   2m
   openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8
   1/1     Running   0          5m
   ...
   ```

   b. 다음 명령을 사용하여 이전 명령의 출력에 나열된 **Pod**의 로그를 표시합니다.

   -

```
$ oc logs <pod_name> -n <namespace> 1
```

1

이전 명령의 출력에 표시된 대로 **Pod** 이름과 네임스페이스를 지정합니다.

Pod 로그가 표시되면 **Kubernetes API** 서버는 클러스터 시스템과 통신할 수 있습니다.

3. **FCP(Fibre Channel Protocol)**를 사용하는 설치에는 다중 경로를 활성화하기 위해 추가 단계가 필요합니다. 설치 중에 멀티패스를 활성화하지 마십시오.

자세한 내용은 *설치 후 머신 구성 작업* 설명서에서 **"RHCOS에서 커널 인수를 사용하여 멀티패스 활성화"**를 참조하십시오.

12.2.18. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 **Telemetry** 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 **Telemetry**가 자동으로 실행되고 **OpenShift Cluster Manager** 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 **OpenShift Cluster Manager**를 사용하여 자동으로 또는 **OpenShift Cluster Manager**를 사용하여 수동으로 유지 관리되는지 확인한 후 **subscription watch**를 사용하여 계정 또는 다중 클러스터 수준에서 **OpenShift Container Platform** 서브스크립션을 추적합니다.

추가 리소스

- **Telemetry** 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

12.2.19. 디버깅 정보 수집

IBM Z에서 **OpenShift Container Platform** 설치와 관련된 특정 문제를 해결하고 디버깅하는데 도움이 될 수 있는 디버깅 정보를 수집할 수 있습니다.

사전 요구 사항

- **oc CLI** 도구가 설치되어 있어야 합니다.

절차

1. 클러스터에 로그인합니다.

```
$ oc login -u <username>
```

2. 하드웨어 정보를 수집하는 노드에서 디버깅 컨테이너를 시작합니다.

```
$ oc debug node/<nodename>
```

3. **/host** 파일 시스템으로 변경하고 **toolbox**를 시작합니다.

```
$ chroot /host
$ toolbox
```

4. **dbginfo** 데이터를 수집합니다.

```
$ dbginfo.sh
```

5. 다음으로 **scp**를 사용하여 데이터를 검색할 수 있습니다.

추가 리소스

- SSH 없이 OpenShift 4 노드 내에서 SOSREPORT를 생성하는 방법을 참조하십시오.

12.2.20. 다음 단계

- RHCOS에서 커널 인수를 사용하여 멀티패스 활성화
- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.

12.3. 네트워크가 제한된 환경에서 IBM Z 및 LINUXONE에 Z/VM으로 클러스터 설치

OpenShift Container Platform 버전 4.9에서는 네트워크가 제한된 환경에서 사용자가 제공하는 IBM Z 및 LinuxONE 인프라에 클러스터를 설치할 수 있습니다.

참고

이 문서는 IBM Z에 대해서만 설명하지만 여기에 있는 모든 정보는 LinuxONE에도 적용됩니다.

중요

베어 메탈 이외의 플랫폼의 경우 추가 고려 사항이 있습니다. OpenShift Container Platform 클러스터를 설치하기 전에 guidelines for deploying OpenShift Container Platform on non-tested platforms에 있는 내용을 확인하십시오.

12.3.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.
- 네트워크가 제한된 환경에서 설치 미러 레지스트리를 생성하고 OpenShift Container Platform 버전의 imageContentSources 데이터를 가져옵니다.
- 설치 프로세스를 시작하기 전에 기존 설치 파일을 이동하거나 제거해야 합니다. 이렇게 하면 설치 프로세스 중에 필요한 설치 파일이 생성되고 업데이트됩니다.

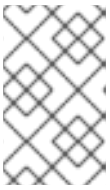
  

  중요

  설치 미디어에 액세스할 수있는 시스템에서 설치 프로세스를 수행해야 합니다.

- 클러스터 용 NFS를 사용하여 영구 스토리지를 프로비저닝합니다. 프라이빗 이미지 레지스트리를 배포하려면 스토리지에서 ReadWriteMany 액세스 모드를 제공해야 합니다.
- 방화벽을 사용하며 Telemetry 서비스를 사용할 예정인 경우 클러스터가 액세스해야 하는 사

이트를 허용하도록 방화벽을 구성해야 합니다.



참고

프록시를 구성하는 경우에도 해당 사이트 목록을 검토하십시오.

12.3.2. 네트워크가 제한된 환경에서의 설치 정보

OpenShift Container Platform 4.9에서는 소프트웨어 구성 요소를 받기 위한 인터넷 접속이 필요하지 않은 설치를 수행할 수 있습니다. 제한된 네트워크 설치는 클러스터를 설치하는 클라우드 플랫폼에 따라 설치 관리자 프로비저닝 인프라 또는 사용자 프로비저닝 인프라를 사용하여 완료할 수 있습니다.

클라우드 플랫폼에 제한된 네트워크 설치를 수행하는 방법을 선택해도 클라우드 API에 액세스는 가능해야 합니다. Amazon Web Service의 Route 53 DNS 및 IAM 서비스와 같은 일부 클라우드 기능에는 인터넷 액세스가 필요합니다. 사용 중인 네트워크에 따라 베어메탈 하드웨어 또는 VMware vSphere에 설치하기 위해 필요한 인터넷 액세스가 줄어들 수 있습니다.

제한된 네트워크 설치를 완료하려면 OpenShift Container Platform 레지스트리의 내용을 미러링하고 설치 미디어를 포함할 레지스트리를 생성해야 합니다. 인터넷과 폐쇄 네트워크에 모두 액세스하거나 제한 사항을 따르는 다른 방법을 통해 미러 호스트에 레지스트리를 생성할 수 있습니다.

중요

사용자 프로비저닝 설치의 구성이 복잡하므로 사용자 프로비저닝 인프라를 사용하여 제한된 네트워크 설치를 시도하기 전에 표준 사용자 프로비저닝 인프라 설치를 완료하는 것이 좋습니다. 이 테스트 설치를 완료하면 제한된 네트워크에 설치하는 동안 발생할 수 있는 문제를 보다 쉽게 파악 및 해결할 수 있습니다.

12.3.2.1. 추가 제한

제한된 네트워크의 클러스터에는 다음과 같은 추가 제한이 있습니다.

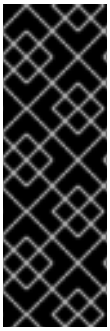
- ClusterVersion 상태에 사용 가능한 업데이트를 검색할 수 없음 오류가 포함되어 있습니다.
- 기본적으로 필요한 이미지 스트림 태그에 액세스할 수 없기 때문에 개발자 카탈로그의 내용을 사용할 수 없습니다.

12.3.3. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하기 위해 필요한 이미지를 받으려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 **Telemetry** 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.



중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

12.3.4. 사용자 프로비저닝 인프라를 포함한 클러스터의 시스템 요구사항

사용자 프로비저닝 인프라가 포함된 클러스터의 경우, 필요한 모든 시스템을 배포해야 합니다.

이 섹션에서는 사용자 프로비저닝 인프라에 **OpenShift Container Platform**을 배포해야 하는 요구 사항에 대해 설명합니다.

12.3.4.1. 클러스터 설치에 필요한 시스템

최소 **OpenShift Container Platform** 클러스터에 다음과 같은 호스트가 필요합니다.

표 **12.18.** 최소 필수 호스트

호스트	설명
임시 부트스트랩 시스템 한 개	컨트롤 플레인 시스템 세 개에 OpenShift Container Platform 클러스터를 배포하기 위한 부트스트랩 시스템이 클러스터에 필요합니다. 클러스터를 설치한 후 부트스트랩 시스템을 제거할 수 있습니다.
컨트롤 플레인 시스템 세 개	컨트롤 플레인 시스템은 컨트롤 플레인을 구성하는 Kubernetes 및 OpenShift Container Platform 서비스를 실행합니다.
두 개 이상의 컴퓨팅 시스템(작업자 시스템이라고도 함).	OpenShift Container Platform 사용자가 요청한 워크로드는 컴퓨팅 머신에서 실행됩니다.

중요

클러스터의 고가용성을 개선하려면 컨트롤 플레인 시스템을 두 개 이상의 물리적 시스템의 서로 다른 z/VM 인스턴스에 배포하십시오.

부트스트랩, 컨트롤 플레인 시스템은 운영 체제로 RHCOS (Red Hat Enterprise Linux CoreOS)를 사용해야 합니다. 그러나 컴퓨팅 머신은 RHCOS(Red Hat Enterprise Linux CoreOS), RHEL (Red Hat Enterprise Linux) 7.9 또는 RHEL 8.4 중에서 선택할 수 있습니다.

RHCOS는 RHEL 8(Red Hat Enterprise Linux)을 기반으로하며 모든 하드웨어 인증 및 요구사항을 모두 이어받습니다. Red Hat Enterprise Linux 기술 기능 및 제한을 참조하십시오.

12.3.4.2. 클러스터 설치를 위한 최소 리소스 요구 사항

각 클러스터 시스템이 다음과 같은 최소 요구사항을 충족해야 합니다.

표 12.19. 최소 리소스 요구사항

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS
부트스트랩	RHCOS	4	16GB	100GB	해당 없음
컨트롤 플레인	RHCOS	4	16GB	100GB	해당 없음
컴퓨팅	RHCOS	2	8GB	100GB	해당 없음

1. SMT-2가 활성화된 경우 하나의 물리적 코어(IFL)는 두 개의 논리 코어(스레드)를 제공합니

다. 하이퍼바이저는 두 개 이상의 **vCPU**를 제공할 수 있습니다.

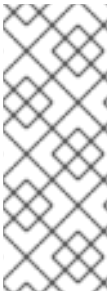
12.3.4.3. 최소 IBM Z 시스템 환경

다음 **IBM** 하드웨어에 **OpenShift Container Platform** 버전 **4.9**를 설치할 수 있습니다.

- **IBM z15**(모든 모델), **IBM z14**(모든 모델), **IBM z13** 및 **IBM z13s**
- **LinuxONE**, 모든 버전

하드웨어 요구 사항

- 각 클러스터에 대해 **SMT2**를 사용할 수 있는 **6**개의 **IFL**과 동등합니다.
- **LoadBalancer** 서비스에 연결하고 클러스터 외부 트래픽에 대한 데이터를 제공하는 네트워크 연결은 하나 이상 있습니다.



참고

전용 또는 공유 **IFL**을 사용하여 충분한 컴퓨팅 리소스를 할당할 수 있습니다. 리소스 공유는 **IBM Z**의 주요 강점 중 하나입니다. 그러나 각 하이퍼바이저 계층에서 용량을 올바르게 조정하고 모든 **OpenShift Container Platform** 클러스터에 충분한 리소스를 확인해야 합니다.

중요

클러스터의 전반적인 성능에 영향을 미칠 수 있으므로 **OpenShift Container Platform** 클러스터를 설정하는 데 사용되는 **LPAR**은 충분한 컴퓨팅 용량을 제공해야 합니다. 이 컨텍스트에서 하이퍼바이저 수준의 **LPAR** 가중치 관리, 권한 부여 및 **CPU** 공유는 중요한 역할을 합니다.

운영 체제 요구 사항

- **z/VM 7.1** 이상의 인스턴스 **1**개

z/VM 인스턴스에서 다음을 설정합니다.

- OpenShift Container Platform 컨트롤 플레인 시스템 용 게스트 가상 머신 3대
- OpenShift Container Platform 컴퓨팅 시스템 용 게스트 가상 머신 2 대
- 임시 OpenShift Container Platform 부트스트랩 머신 용 게스트 가상 머신 1 대

IBM Z 네트워크 연결 요구 사항

IBM Z의 z/VM에 설치하려면 레이어 2 모드의 단일 z/VM 가상 NIC가 필요합니다. 또한 다음이 필요합니다.

- 직접 연결된 OSA 또는 RoCE 네트워크 어댑터
- z/VM VSwitch 설정. 권장 설정의 경우 OSA 링크 통합을 사용합니다.

z/VM 게스트 가상 머신 용 디스크 스토리지

- FICON 연결 디스크 스토리지 (DASD). 이는 z/VM 미니 디스크, 풀팩 미니 디스크 또는 전용 DASD가 포함될 수 있으며, 모두 기본 CDL로 포맷되어야 합니다. Red Hat Enterprise Linux CoreOS (RHCOS) 설치에 필요한 최소 DASD 크기에 도달하려면 확장 주소 볼륨 (EAV)이 필요합니다. 사용 가능한 경우 HyperPAV를 사용하여 최적의 성능을 보장합니다.
- FCP 연결 디스크 스토리지

스토리지 / 메인 메모리

- OpenShift Container Platform 컨트롤 플레인 시스템용 16GB
- OpenShift Container Platform 컴퓨팅 머신 용 8GB
- 임시 OpenShift Container Platform 부트스트랩 머신 용 16GB

12.3.4.4. 권장되는 IBM Z 시스템 환경

하드웨어 요구 사항

- 각 클러스터에 대해 SMT2가 활성화된 6개의 IFL과 동등한 3개의 LPARS가 있습니다.

- LoadBalancer 서비스에 연결하고 클러스터 외부의 트래픽에 대한 데이터를 제공하는 두 개의 네트워크 연결

- Hipersockets - 하나의 장치에 직접 연결되거나 z/VM 게스트에 투명하도록 하나의 z/VM VSWITCH와 브리징하여 노드에 연결됩니다. HiperSockets을 노드에 직접 연결하려면 RHEL 8 게스트를 통해 외부 네트워크의 게이트웨이를 설정하여 HiperSockets 네트워크에 연결해야 합니다.

운영 체제 요구 사항

- 고가용성을 위해 z/VM 7.1 이상의 인스턴스 2개 또는 3개

z/VM 인스턴스에서 다음을 설정합니다.

- OpenShift Container Platform 컨트롤 플레인 시스템용 게스트 가상 머신 3개(z/VM 인스턴스당 1개)

- z/VM 인스턴스에 분산된 OpenShift Container Platform 컴퓨팅 머신용 게스트 가상 머신 6개 이상

- 임시 OpenShift Container Platform 부트스트랩 시스템용 게스트 가상 머신 1개

- 오버 커밋된 환경에서 통합 구성 요소를 사용하려면 CP 명령 SET SHARE 를 사용하여 컨트롤 플레인의 우선 순위를 높입니다. 인프라 노드가 있는 경우 동일한 작업을 수행합니다. IBM 문서의 SET SHARE를 참조하십시오.

IBM Z 네트워크 연결 요구 사항

IBM Z의 z/VM에 설치하려면 레이어 2 모드의 단일 z/VM 가상 NIC가 필요합니다. 또한 다음이 필요합니다.

- 직접 연결된 OSA 또는 RoCE 네트워크 어댑터

- z/VM VSwitch 설정. 권장 설정의 경우 OSA 링크 통합을 사용합니다.

z/VM 게스트 가상 머신 용 디스크 스토리지

- FICON 연결 디스크 스토리지 (DASD). 이는 z/VM 미니 디스크, 풀팩 미니 디스크 또는 전용 DASD가 포함될 수 있으며, 모두 기본 CDL로 포맷되어야 합니다. Red Hat Enterprise Linux CoreOS (RHCOS) 설치에 필요한 최소 DASD 크기에 도달하려면 확장 주소 볼륨 (EAV)이 필요합니다. 가능한 경우 HyperPAV 및 고성능 FICON (zHPF)을 사용하여 최적의 성능을 보장합니다.

- FCP 연결 디스크 스토리지

스토리지 / 메인 메모리

- OpenShift Container Platform 컨트롤 플레인 시스템용 16GB

- OpenShift Container Platform 컴퓨팅 머신 용 8GB

- 임시 OpenShift Container Platform 부트스트랩 머신 용 16GB

12.3.4.5. 인증서 서명 요청 관리

사용자가 프로비저닝하는 인프라를 사용하는 경우 자동 시스템 관리 기능으로 인해 클러스터의 액세스가 제한되므로 설치한 후 클러스터 인증서 서명 요청(CSR)을 승인하는 메커니즘을 제공해야 합니다. kube-controller-manager는 kubelet 클라이언트 CSR만 승인합니다. machine-approver는 올바른 시스템에서 발행한 요청인지 확인할 수 없기 때문에 kubelet 자격 증명을 사용하여 요청하는 서비스 인증서의 유효성을 보장할 수 없습니다. kubelet 서빙 인증서 요청의 유효성을 확인하고 요청을 승인하는 방법을 결정하여 구현해야 합니다.

추가 리소스

- IBM 문서에서 Bridging a HiperSockets LAN with a z/VM Virtual Switch를 참조하십시오.

- 성능 최적화를 위해 Scaling HyperPAV alias devices on Linux guests on z/VM을 참조하십시오.

- LPAR 가중치 관리 및 자격을 보려면 LPAR 성능의 주제를 참조하십시오.

- IBM Z 및 LinuxONE 환경에 대한 권장 호스트 사례

12.3.4.6. 사용자 프로비저닝 인프라에 대한 네트워킹 요구사항

모든 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템이 부팅 중에 Ignition 구성 파일을 가져오려면 initramfs에 네트워킹을 구성해야 합니다.

초기 부팅 과정에서 시스템에 필요한 부팅 옵션을 제공하여 DHCP 서버를 통해 설정하거나 정적으로 설정하는 IP 주소 구성이 필요합니다. 네트워크 연결이 설정된 후 시스템은 HTTP 또는 HTTPS 서버에서 Ignition 구성 파일을 다운로드합니다. 그런 다음 Ignition 구성 파일을 사용하여 각 머신의 정확한 상태를 설정합니다. Machine Config Operator는 설치 후 새 인증서 또는 키 적용과 같은 머신에 대한 추가 변경을 완료합니다.

클러스터 시스템의 장기적인 관리를 위해 DHCP 서버를 사용하는 것이 좋습니다. DHCP 서버가 클러스터 시스템에 영구 IP 주소, DNS 서버 정보 및 호스트 이름을 제공하도록 구성되었는지 확인합니다.

참고

사용자 프로비저닝 인프라에 DHCP 서비스를 사용할 수 없는 경우 RHCOS 설치 시 노드에 IP 네트워킹 구성과 DNS 서버의 주소를 대신 제공할 수 있습니다. ISO 이미지에서 설치하는 경우 부팅 인수로 전달할 수 있습니다. 고정 IP 프로비저닝 *및 고급 네트워킹 옵션에 대한 자세한 내용은 RHCOS 설치 및 OpenShift Container Platform 부트스트랩 프로세스 시작* 섹션을 참조하십시오.

Kubernetes API 서버가 클러스터 시스템의 노드 이름을 확인할 수 있어야 합니다. API 서버와 작업자 노드가 서로 다른 영역에 있는 경우, API 서버가 노드 이름을 확인할 수 있도록 기본 DNS 검색 영역을 설정할 수 있습니다. 노드 개체와 모든 DNS 요청에서 항상 정규화된 도메인 이름으로 호스트를 가리키는 것도 지원되는 방법입니다

12.3.4.6.1. DHCP를 통해 클러스터 노드의 호스트 이름 설정

RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 호스트 이름은 NetworkManager를 통해 설정됩니다. 기본적으로 시스템은 DHCP를 통해 호스트 이름을 가져옵니다. DHCP에서 호스트 이름을 제공하지 않거나, 커널 인수를 통해 정적으로 설정하거나 다른 방법을 통해 설정하면 역방향 DNS 조회를 통해 얻을 수 있습니다. 역방향 DNS 조회는 노드에서 네트워크를 초기화한 후 수행되며 확인하는 데 시간이 걸릴 수 있습니다. 다른 시스템 서비스는 이 보다 먼저 시작하여 호스트 이름을 localhost 등으로 감지할 수 있습니다. DHCP를 사용하여 각 클러스터 노드의 호스트 이름을 제공하여 이 문제를 방지할 수 있습니다.

또한 DHCP를 통해 호스트 이름을 설정하면 DNS 분할 수평 구현 환경에서 수동으로 DNS 레코드 이

름 구성 오류를 무시할 수 있습니다.

12.3.4.6.2. 네트워크 연결 요구사항

OpenShift Container Platform 클러스터 구성 요소가 통신할 수 있도록 시스템 간 네트워크 연결을 구성해야 합니다. 각 시스템에서 클러스터에 있는 다른 모든 시스템의 호스트 이름을 확인할 수 있어야 합니다.

이 섹션에서는 필요한 포트에 대해 자세히 설명합니다.

표 12.20. 모든 시스템 간 통신에 사용되는 포트

프로토콜	포트	설명
ICMP	해당 없음	네트워크 연결성 테스트
TCP	**1936**	메트릭
	9000-9999	**9100-9101** 포트의 노드 내보내기 및 **9099** 포트의 Cluster Version Operator를 포함한 호스트 수준 서비스.
	10250-10259	Kubernetes에서 예약하는 기본 포트
	10256	openshift-sdn
UDP	**4789**	VXLAN 및 Geneve
	6081	VXLAN 및 Geneve
	9000-9999	**9100-9101** 포트의 노드 내보내기를 포함한 호스트 수준 서비스.
	500	IPsec IKE 패킷
	4500	IPsec NAT-T 패킷
TCP/UDP	**30000-32767**	Kubernetes 노드 포트
ESP	해당 없음	IPsec Encapsulating Security Payload (ESP)

표 12.21. 모든 시스템과 컨트롤 플레인 간 통신에 사용되는 포트

프로토콜	포트	설명

프로토콜	포트	설명
TCP	**6443**	Kubernetes API

표 **12.22.** 컨트롤 플레인 머신 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**2379-2380**	etcd 서버 및 피어 포트

사용자 프로비저닝 인프라에 대한 **NTP** 구성

OpenShift Container Platform 클러스터는 기본적으로 공용 **NTP(Network Time Protocol)** 서버를 사용하도록 구성되어 있습니다. 로컬 엔터프라이즈 **NTP** 서버를 사용하거나 클러스터가 연결이 끊긴 네트워크에 배포되는 경우 특정 시간 서버를 사용하도록 클러스터를 구성할 수 있습니다. 자세한 내용은 ***chrony** 타임 서비스 설정* 문서를 참조하십시오.

추가 리소스

- chrony 타임 서비스 설정

12.3.4.7. 사용자 프로비저닝 **DNS** 요구사항

OpenShift Container Platform 배포의 경우 다음 구성 요소에 **DNS** 이름을 확인해야 합니다.

- **Kubernetes API**
- **OpenShift Container Platform** 애플리케이션 와일드카드
- 부트스트랩, 컨트롤 플레인 및 컴퓨팅 시스템

Kubernetes API, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템에 대한 역방향 **DNS** 확인이 필요합니다.

DNS A/AAAA 또는 **CNAME** 레코드는 이름 확인에 사용되며 **PTR** 레코드는 역방향 이름 확인에 사용됩니다. **RHCOS (Red Hat Enterprise Linux CoreOS)**는 **DHCP**에서 호스트 이름을 제공하지 않는 한 모든 노드의 호스트 이름을 설정할 때 역방향 레코드를 사용하기 때문에 역방향 레코드가 중요합니다. 또한

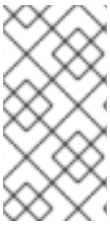
역방향 레코드는 OpenShift Container Platform이 작동하는 데 필요한 인증서 서명 요청 (CSR)을 생성하는 데 사용됩니다.

사용자가 프로비저닝한 OpenShift Container Platform 클러스터에 대해 다음 DNS 레코드가 필요하며 설치 전에 있어야 합니다. 각 레코드에서 <cluster_name>은 클러스터 이름이고 <base_domain>은 install-config.yaml 파일에서 지정하는 기반 도메인입니다. 전체 DNS 레코드는 <component>.<cluster_name>.<base_domain> 형식입니다.

표 12.23. 필수 DNS 레코드

구성 요소	레코드	설명
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API 로드 밸런서를 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.
	api-int.<cluster_name>.<base_domain>.	내부적으로 API 로드 밸런서를 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.  **중요** API 서버는 Kubernetes에 기록된 호스트 이름으로 작업자 노드를 확인할 수 있어야 합니다. API 서버가 노드 이름을 확인할 수 없는 경우 프록시된 API 호출이 실패할 수 있으며 pod에서 로그를 검색할 수 없습니다.
라우트	***.apps.<cluster_name>.<base_domain>.**	애플리케이션 인그레스 로드 밸런서를 참조하는 와일드카드 DNS A/AAA 또는 CNAME 레코드입니다. 애플리케이션 인그레스 로드 밸런서는 Ingress 컨트롤러 Pod를 실행하는 머신을 대상으로 합니다. Ingress 컨트롤러 Pod는 기본적으로 컴퓨팅 머신에서 실행됩니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다. 예를 들어 **console-openshift-console.apps.<cluster_name>.<base_domain>**은 OpenShift Container Platform 콘솔의 와일드카드 경로로 사용됩니다.
부트스트랩 시스템	**bootstrap.<cluster_name>.<base_domain>.**	부트스트랩 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.
컨트롤 플레인 머신	**<master><n>.<cluster_name>.<base_domain>.**	컨트롤 플레인 노드의 각 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.

구성 요소	레코드	설명
컴퓨팅 머신	**<worker><n>. <cluster_name>. <base_domain>.**	작업자 노드의 각 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.



참고

OpenShift Container Platform 4.4 이상에서는 **DNS** 구성에서 **etcd** 호스트 및 **SRV** 레코드를 지정할 필요가 없습니다.

작은 정보

dig 명령을 사용하여 이름과 역방향 이름을 확인할 수 있습니다. 자세한 검증 단계는 *사용자 프로비저닝 인프라의 **DNS** 확인* 섹션을 참조하십시오.

12.3.4.7.1. 사용자 프로비저닝 클러스터의 DNS 구성 예

이 섹션에서는 사용자 프로비저닝 인프라에 **OpenShift Container Platform**을 배포하기 위한 **DNS** 요구 사항을 충족하는 **A** 및 **PTR** 레코드 구성 샘플을 제공합니다. 샘플은 하나의 **DNS** 솔루션을 선택하기 위한 조언을 제공하기 위한 것이 아닙니다.

이 예제에서 클러스터 이름은 **ocp4**이고 기본 도메인은 **example.com**입니다.

사용자 프로비저닝 클러스터의 **DNS A** 레코드 구성 예

다음 **BIND** 영역 파일의 예제에서는 사용자가 프로비저닝한 클러스터의 이름 확인을 위한 샘플 **A** 레코드를 보여줍니다.

예 12.4. 샘플 DNS 영역 데이터베이스

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
```

```
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
master0.ocp4.example.com. IN A 192.168.1.97 5
master1.ocp4.example.com. IN A 192.168.1.98 6
master2.ocp4.example.com. IN A 192.168.1.99 7
;
worker0.ocp4.example.com. IN A 192.168.1.11 8
worker1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1

Kubernetes API의 이름 확인을 제공합니다. 레코드는 **API** 로드 밸런서의 **IP** 주소를 나타냅니다.

2

Kubernetes API의 이름 확인을 제공합니다. 레코드는 **API** 로드 밸런서의 **IP** 주소를 참조하며 내부 클러스터 통신에 사용됩니다.

3

와일드카드 경로의 이름 확인을 제공합니다. 레코드는 애플리케이션 인그레스 로드 밸런서의 **IP** 주소를 나타냅니다. 애플리케이션 인그레스 로드 밸런서는 **Ingress** 컨트롤러 **Pod**를 실행하는 머신을 대상으로 합니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

참고

이 예제에서는 **Kubernetes API** 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 **API** 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

4

5 6 7

컨트롤 플레인 시스템의 이름 확인을 제공합니다.

8 9

컴퓨팅 시스템의 이름 확인을 제공합니다.

사용자 프로비저닝 클러스터의 **DNS PTR** 레코드 구성 예

다음 예제 **BIND** 영역 파일은 사용자 프로비저닝 클러스터의 역방향 이름 확인을 위한 샘플 **PTR** 레코드를 보여줍니다.

예 12.5. 역방향 레코드의 샘플 DNS 영역 데이터베이스

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR master0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR master1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR master2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com. 8
;
;EOF
```

1

Kubernetes API의 역방향 **DNS** 확인을 제공합니다. **PTR** 레코드는 **API** 로드 밸런서의 레코드 이름을 참조합니다.

2

3

부트스트랩 시스템의 역방향 **DNS** 확인을 제공합니다.

4 5 6

컨트롤 플레인 시스템의 역방향 **DNS** 확인을 제공합니다.

7 8

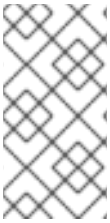
컴퓨팅 시스템의 역방향 **DNS** 확인을 제공합니다.

참고

OpenShift Container Platform 애플리케이션 와일드카드에는 **PTR** 레코드가 필요하지 않습니다.

12.3.4.8. 사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구사항

OpenShift Container Platform을 설치하기 전에 **API** 및 애플리케이션 인그레스 로드 밸런싱 인프라를 프로비저닝해야 합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 **API** 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.



참고

RHEL(Red Hat Enterprise Linux) 인스턴스를 사용하여 **API** 및 애플리케이션 인그레스 로드 밸런서를 배포하려면 **RHEL** 서브스크립션을 별도로 구매해야 합니다.

로드 밸런서 인프라는 다음 요구 사항을 충족해야 합니다.

1. **API** 로드 밸런서: 플랫폼과 상호 작용하고 플랫폼을 구성할 수 있도록 사용자(인간과 시스템 모두)에게 공통 끝점을 제공합니다. 다음 조건을 설정합니다.

   - **Layer 4** 로드 밸런싱 전용입니다. 이를 **Raw TCP, SSL Passthrough** 또는 **SSL Bridge** 모드라고 합니다. **SSL Bridge** 모드를 사용하는 경우, **API** 경로에 대해 **SNI(Server Name Indication,** 서버 이름 표시)를 활성화해야 합니다.

   -

스테이트리스 로드 밸런싱 알고리즘입니다. 옵션은 로드 밸런서 구현에 따라 달라집니다.

참고

API 로드 밸런서가 제대로 작동하기 위해 세션 지속성이 필요하지 않습니다.

로드 밸런서의 전면과 후면 모두에서 다음 포트를 구성하십시오.

표 12.24. API 로드 밸런서

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
6443	부트스트랩 및 컨트롤 플레인. 부트스트랩 시스템이 클러스터 컨트롤 플레인을 초기화한 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다. API 서버 상태 검사 프로브에 대한 **/readyz** 끝점을 구성해야 합니다.	X	X	Kubernetes API 서버
22623	부트스트랩 및 컨트롤 플레인. 부트스트랩 시스템이 클러스터 컨트롤 플레인을 초기화한 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다.	X		시스템 구성 서버

참고

API 서버가 /readyz 엔드포인트를 해제하는 시점부터 풀에서 API 서버 인스턴스가 제거되는 시점까지 시간이 30초를 넘지 않도록 로드 밸런서를 구성해야 합니다. /readyz가 오류를 반환하거나 정상 상태가 된 후 정해진 시간 안에 끝점이 제거 또는 추가되어야 합니다. 5초 또는 10초의 프로빙 주기(두 번의 성공적인 요청은 정상 상태, 세 번의 요청은 비정상 상태)는 충분한 테스트를 거친 값입니다.

2. 애플리케이션 인그레스 로드 밸런서: 클러스터 외부에서 유입되는 애플리케이션 트래픽에 대한 인그래스 포인트를 제공합니다. 다음 조건을 설정합니다.

- Layer 4 로드 밸런싱 전용입니다. 이를 Raw TCP, SSL Passthrough 또는 SSL Bridge 모드라고 합니다. SSL Bridge 모드를 사용하는 경우 인그레스 경로에 대해 SNI(Server Name Indication, 서버 이름 표시)를 활성화해야 합니다.

- 사용 가능한 옵션과 플랫폼에서 호스팅되는 애플리케이션 유형에 따라 연결 기반 또는

세션 기반 지속성이 권장됩니다.

작은 정보

애플리케이션 인그레스 로드 밸런서에서 클라이언트의 실제 IP 주소를 확인할 수 있는 경우 소스 IP 기반 세션 지속성을 활성화하면 엔드 투 엔드 TLS 암호화를 사용하는 애플리케이션의 성능을 향상시킬 수 있습니다.

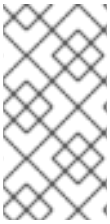
로드 밸런서의 전면과 후면 모두에서 다음 포트를 구성하십시오.

표 12.25. 애플리케이션 인그레스 로드 밸런서

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
443	기본적으로 인그레스 컨트롤러 pod, 컴퓨팅 또는 작업자를 실행하는 시스템입니다.	X	X	HTTPS 트래픽
80	기본적으로 인그레스 컨트롤러 pod, 컴퓨팅 또는 작업자를 실행하는 시스템입니다.	X	X	HTTP 트래픽
1936	기본적으로 Ingress 컨트롤러 Pod를 실행하는 작업자 노드입니다. 수신 상태 점검 프로브에 대해 **/healthz/ready** 끝점을 구성해야 합니다.	X	X	HTTP 트래픽

참고

컴퓨팅 노드가 0인 3-노드 클러스터를 배포하는 경우 Ingress 컨트롤러 Pod는 컨트롤 플레인 노드에서 실행됩니다. 3-노드 클러스터 배포에서 HTTP 및 HTTPS 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다.



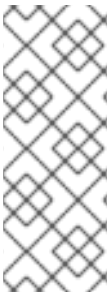
참고

인그레스 라우터에 대한 작업 구성이 OpenShift Container Platform 클러스터에 필요합니다. 컨트롤 플레인 초기화 후 인그레스 라우터를 설정해야 합니다.

12.3.4.8.1. 사용자 프로비저닝 클러스터의 로드 밸런서 구성 예

이 섹션에서는 사용자 프로비저닝 클러스터의 로드 밸런싱 요구 사항을 충족하는 API 및 애플리케이션 수신 로드 밸런서 구성 예를 제공합니다. 샘플은 HAProxy 로드 밸런서에 대한

/etc/haproxy/haproxy.cfg 구성입니다. 이 예제에서는 하나의 로드 밸런싱 솔루션을 선택하기 위한 조언을 제공하는 것을 목적으로 하지 않습니다.



참고

이 예제에서는 Kubernetes API 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 API 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

예 12.6. API 및 애플리케이션 인그레스 로드 밸런서 구성 샘플

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
frontend stats
  bind *:1936
  mode            http
  log             global
  maxconn 10
  stats enable
  stats hide-version
  stats refresh 30s
  stats show-node
  stats show-desc Stats for ocp4 cluster 1
  stats auth admin:ocp4
  stats uri /stats
listen api-server-6443 2
  bind *:6443
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:6443 check inter 1s backup 3
  server master0 master0.ocp4.example.com:6443 check inter 1s
  server master1 master1.ocp4.example.com:6443 check inter 1s
  server master2 master2.ocp4.example.com:6443 check inter 1s
```

```
listen machine-config-server-22623 4
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup 5
  server master0 master0.ocp4.example.com:22623 check inter 1s
  server master1 master1.ocp4.example.com:22623 check inter 1s
  server master2 master2.ocp4.example.com:22623 check inter 1s
listen ingress-router-443 6
  bind *:443
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:443 check inter 1s
  server worker1 worker1.ocp4.example.com:443 check inter 1s
listen ingress-router-80 7
  bind *:80
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:80 check inter 1s
  server worker1 worker1.ocp4.example.com:80 check inter 1s
```

1

이 예에서 클러스터 이름은 **ocp4** 입니다.

2

포트 **6443**은 **Kubernetes API** 트래픽을 처리하고 컨트롤 플레인 시스템을 가리킵니다.

3 5

부트스트랩 항목은 **OpenShift Container Platform** 클러스터 설치 전에 있어야 하며 부트스트랩 프로세스가 완료된 후 제거해야 합니다.

4

포트 **22623**은 머신 구성 서버 트래픽을 처리하고 컨트롤 플레인 시스템을 가리킵니다.

6

포트 **443**은 **HTTPS** 트래픽을 처리하고 **Ingress** 컨트롤러 **Pod**를 실행하는 시스템을 가리킵니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

7

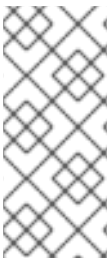
포트 **80**은 **HTTP** 트래픽을 처리하고 **Ingress** 컨트롤러 **Pod**를 실행하는 머신을 가리킵니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

참고

컴퓨팅 노드가 0인 3-노드 클러스터를 배포하는 경우 Ingress 컨트롤러 Pod는 컨트롤 플레인 노드에서 실행됩니다. 3-노드 클러스터 배포에서 HTTP 및 HTTPS 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다.

작은 정보

HAProxy를 로드 밸런서로 사용하는 경우 HAProxy 노드에서 netstat -nltupe를 실행하여 haproxy 프로세스가 포트 6443, 22623, 443 및 80에서 수신 대기 중인지 확인할 수 있습니다.



참고

HAProxy를 로드 밸런서로 사용하고 SELinux가 enforcing으로 설정된 경우 HAProxy 서비스가 setsebool -P haproxy_connect_any=1을 실행하여 구성된 TCP 포트에 바인딩할 수 있는지 확인해야 합니다.

12.3.5. 사용자 프로비저닝 인프라 준비

사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치하기 전에 기본 인프라를 준비해야 합니다.

이 섹션에서는 OpenShift Container Platform 설치를 준비하기 위해 클러스터 인프라를 설정하는 데 필요한 높은 수준의 단계에 대해 자세히 설명합니다. 여기에는 클러스터 노드에 대한 IP 네트워킹 및 네트워크 연결 구성, Ignition 파일의 웹 서버 준비, 방화벽을 통해 필요한 포트 활성화, 필수 DNS 및 로드 밸런싱 인프라 설정 등이 포함됩니다.

준비 후 클러스터 인프라는 *사용자 프로비저닝 인프라가 있는 클러스터의 요구 사항* 섹션에 설명된 요구 사항을 충족해야 합니다.

사전 요구 사항

- OpenShift Container Platform 4.x 테스트된 통합 페이지를 검토했습니다.
- *사용자 프로비저닝 인프라가 있는 클러스터의 요구 사항* 섹션에 자세히 설명된 인프라 요구 사항을 검토했습니다.

프로세스

1. 고정 IP 주소를 설정합니다.

2. 클러스터 노드에 Ignition 파일을 제공하기 위해 HTTP 또는 HTTPS 서버를 설정합니다.

3. 네트워크 인프라가 클러스터 구성 요소 간 필수 네트워크 연결을 제공하는지 확인합니다. 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 인프라 섹션의 네트워킹 요구 사항* 섹션을 참조하십시오.

4. OpenShift Container Platform 클러스터 구성 요소가 통신하는 데 필요한 포트를 활성화하도록 방화벽을 구성합니다. 필요한 포트에 대한 자세한 내용은 *사용자 프로비저닝 인프라 섹션의 네트워킹 요구 사항* 섹션을 참조하십시오.

5. 클러스터에 필요한 DNS 인프라를 설정합니다.

   a. Kubernetes API, 애플리케이션 와일드카드, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템의 DNS 이름 확인을 구성합니다.

   b. Kubernetes API, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템에 대한 역방향 DNS 확인을 구성합니다.

   OpenShift Container Platform DNS 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 DNS 요구 사항* 섹션을 참조하십시오.

6. DNS 구성을 확인합니다.

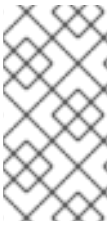
   a. 설치 노드에서 Kubernetes API의 레코드 이름, 와일드카드 경로 및 클러스터 노드에 대해 DNS 조회를 실행합니다. 응답의 IP 주소가 올바른 구성 요소에 해당하는지 확인합니다.

   b. 설치 노드에서 로드 밸런서 및 클러스터 노드의 IP 주소에 대해 역방향 DNS 조회를 실행합니다. 응답의 레코드 이름이 올바른 구성 요소에 해당하는지 확인합니다.

   자세한 DNS 검증 단계는 *사용자 프로비저닝 인프라에 대한 DNS 확인* 섹션을 참조하십

시오.

7. 필요한 API 및 애플리케이션 수신 로드 밸런싱 인프라를 프로비저닝합니다. 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구 사항* 섹션을 참조하십시오.



참고

일부 로드 밸런싱 솔루션에는 로드 밸런싱을 초기화하기 전에 클러스터 노드의 DNS 이름을 확인해야 합니다.

12.3.6. 사용자 프로비저닝 인프라에 대한 DNS 확인 검증

사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치하기 전에 DNS 구성을 확인할 수 있습니다.

중요

클러스터를 설치하기 전에 이 섹션에 설명된 검증 단계를 성공해야 합니다.

사전 요구 사항

- 사용자 프로비저닝 인프라에 필요한 DNS 레코드를 구성했습니다.

프로세스

1. 설치 노드에서 Kubernetes API의 레코드 이름, 와일드카드 경로 및 클러스터 노드에 대해 DNS 조회를 실행합니다. 응답에 포함된 IP 주소가 올바른 구성 요소에 해당하는지 확인합니다.

   a. Kubernetes API 레코드 이름을 조회합니다. 결과가 API 로드 밸런서의 IP 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
```

1 <nameserver_ip>를 네임서버의 IP 주소로, <cluster_name>을 클러스터 이름으로, <base_domain>을 기본 도메인 이름으로 바꿉니다.

출력 예

```
api.ocp4.example.com.  0 IN A 192.168.1.5
```

b.

Kubernetes 내부 API 레코드 이름을 조회합니다. 결과가 API 로드 밸런서의 IP 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
```

출력 예

```
api-int.ocp4.example.com.  0 IN A 192.168.1.5
```

c.

예제 *.apps.<cluster_name>.<base_domain>을 테스트합니다. DNS 와일드카드를 조회합니다. 모든 애플리케이션 와일드카드 조회는 애플리케이션 인그레스 로드 밸런서의 IP 주소로 확인되어야 합니다.

```
$ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.
<base_domain>
```

출력 예

```
random.apps.ocp4.example.com.  0 IN A 192.168.1.5
```

참고

예제 출력에서는 **Kubernetes API** 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 **API** 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

random 항목을 다른 와일드카드 값으로 교체할 수 있습니다. 예를 들어 **OpenShift Container Platform** 콘솔의 경로를 쿼리할 수 있습니다.

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

출력 예

```
console-openshift-console.apps.ocp4.example.com. 0 IN A 192.168.1.5
```

d. 부트스트랩 **DNS** 레코드 이름에 대해 조회를 실행합니다. 결과가 부트스트랩 노드의 **IP** 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.
<base_domain>
```

출력 예

```
bootstrap.ocp4.example.com.  0 IN A 192.168.1.96
```

e. 이 방법을 사용하여 컨트롤 플레인 및 컴퓨팅 노드의 **DNS** 레코드 이름에 대해 조회를 수행합니다. 결과가 각 노드의 **IP** 주소에 해당하는지 확인합니다.

2. 설치 노드에서 로드 밸런서 및 클러스터 노드의 **IP** 주소에 대해 역방향 **DNS** 조회를 실행합니다. 응답에 포함된 레코드 이름이 올바른 구성 요소에 해당하는지 확인합니다.

a.

API 로드 밸런서의 IP 주소에 대해 역방향 조회를 수행합니다. 응답에 Kubernetes API 및 Kubernetes 내부 API의 레코드 이름이 포함되어 있는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

출력 예

```
5.1.168.192.in-addr.arpa. 0 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 0 IN PTR api.ocp4.example.com. 2
```

1

Kubernetes 내부 API의 레코드 이름을 제공합니다.

2

Kubernetes API의 레코드 이름을 제공합니다.

참고

OpenShift Container Platform 애플리케이션 와일드카드에는 PTR 레코드가 필요하지 않습니다. 애플리케이션 인그레스 로드 밸런서의 IP 주소에 대한 역방향 DNS 확인에는 유효성 검사 단계가 필요하지 않습니다.

b.

부트스트랩 노드의 IP 주소에 대해 역방향 조회를 수행합니다. 결과가 부트스트랩 노드의 DNS 레코드 이름을 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

출력 예

```
96.1.168.192.in-addr.arpa. 0 IN PTR bootstrap.ocp4.example.com.
```

c.

이 방법을 사용하여 컨트롤 플레인 및 컴퓨팅 노드의 IP 주소에 대해 역방향 조회를 수행합니다. 결과가 각 노드의 DNS 레코드 이름과 일치하는지 확인합니다.

12.3.7. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 core 사용자의 ~/.ssh/authorized_keys 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 core로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. ./openshift-install gather 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

프로세스

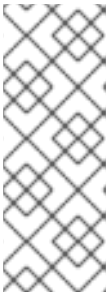
1.

로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1

새 SSH 키의 경로 및 파일 이름(예: ~/.ssh/id_ed25519)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 '~/.ssh 디렉터리에 있는지 확인하십시오.



참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 ed25519 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 rsa 또는 ecdsa 알고리즘을 사용하는 키를 생성합니다.

2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 ~/.ssh/id_ed25519.pub 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 ./openshift-install gather 명령을 사용하려는 경우 필요합니다.

참고

일부 배포에서는 ~/.ssh/id_rsa 및 ~/.ssh/id_dsa와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. ssh-agent 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. ssh-agent에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: ~/.ssh/id_ed25519).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

12.3.8. 수동으로 설치 구성 파일 만들기

OpenShift Container Platform을 사용자가 프로비저닝한 설치의 경우 설치 구성 파일을 수동으로 생성합니다.

사전 요구 사항

- 로컬 시스템에 설치 프로그램에 제공할 SSH 공개 키가 있습니다. 키는 디버깅 및 재해 복구를 위해 클러스터 노드에 대한 SSH 인증에 사용됩니다.
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿이 있습니다.

프로세스

1. 필요한 설치 자산을 저장할 설치 디렉터리를 만듭니다.

```
$ mkdir <installation_directory>
```

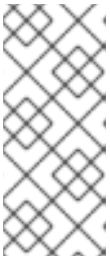
중요

디렉터리를 만들어야 합니다. 부트스트랩 **X.509** 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 **OpenShift Container Platform** 버전에서 설치 파일을 복사할 때는 주의하십시오.

2. 샘플 **install-config.yaml** 파일 템플릿을 사용자 지정하여 **<installation_directory>**에 저장합니다.

참고

이 설정 파일의 이름을 **install-config.yaml**로 지정해야 합니다.



참고

일부 플랫폼 유형의 경우 대체로 **./openshift-install create install-config --dir <installation_directory>** 를 실행하여 **install-config.yaml** 파일을 생성할 수 있습니다. 프롬프트에서 클러스터 구성에 대한 세부 정보를 제공할 수 있습니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.

중요

install-config.yaml 파일은 설치 과정의 다음 단계에서 사용됩니다. 이 시점에서 이를 백업해야 합니다.

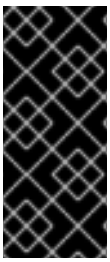
12.3.8.1. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 환경에 대한 세부 정보를 설명하는 사용자

지정 **install-config.yaml** 설치 구성 파일을 제공합니다.

참고

설치한 후에는 **install-config.yaml** 파일에서 이러한 매개변수를 수정할 수 없습니다.



중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

12.3.8.1.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 12.26. 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체

매개변수	설명	값
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	소문자, 하이픈(**-**), 마침표(**.**)로 구성되는 문자열(예: **dev**)입니다.
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

12.3.8.1.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 12.27. 네트워크 매개변수

매개변수	설명	값

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체 **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.
networking.clusterNetwork	Pod의 IP 주소 블록입니다. 기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된**cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16

매개변수	설명	값
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다. 여러 IP 커널 인수를 지정하는 경우 **machineNetwork.cidr** 값은 기본 네트워크의 CIDR이어야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16** **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

12.3.8.1.3. 선택적 구성 매개변수

선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 12.28. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **s390x** (기본값)입니다.	문자열

매개변수	설명	값
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은 **3**입니다.
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **s390x** (기본값)입니다.	문자열

매개변수	설명	값
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 기본값은 **External**입니다. 이 필드를 **Internal** 로 설정하는 것은 클라우드 이외의 플랫폼에서는 지원되지 않습니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

12.3.8.2. IBM Z의 샘플 install-config.yaml 파일

install-config.yaml 파일을 사용자 지정하여 **OpenShift Container Platform** 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.

```
apiVersion: v1
baseDomain: example.com 1
compute: 2
- hyperthreading: Enabled 3
  name: worker
  replicas: 0 4
  architecture : s390x
controlPlane: 5
  hyperthreading: Enabled 6
  name: master
  replicas: 3 7
  architecture : s390x
metadata:
  name: test 8
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 9
    hostPrefix: 23 10
  networkType: OpenShiftSDN
  serviceNetwork: 11
  - 172.30.0.0/16
platform:
  none: {} 12
fips: false 13
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email":
"you@example.com"}}}' 14
sshKey: 'ssh-ed25519 AAAA...' 15
additionalTrustBundle: | 16
```

```
  -----BEGIN CERTIFICATE-----
  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
imageContentSources: 17
- mirrors:
  - <local_repository>/ocp4/openshift4
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_repository>/ocp4/openshift4
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

1

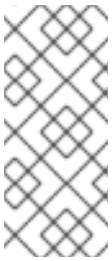
클러스터의 기본 도메인입니다. 모든 **DNS** 레코드는 이 기본 도메인의 하위 도메인이어야 하며 클러스터 이름을 포함해야 합니다.

2 5

controlPlane 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

3 6

동시 멀티스레딩(**SMT**) 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 **SMT**가 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. **SMT**를 비활성화하는 경우 모든 클러스터 머신에서 이를 비활성화해야 합니다. 여기에는 컨트롤 플레인과 컴퓨팅 머신이 모두 포함됩니다.



참고

SMT(동시 멀티 스레딩)는 기본적으로 활성화되어 있습니다. **OpenShift Container Platform** 노드에서 **SMT**를 사용할 수 없는 경우 **hyperthreading** 매개변수가 적용되지 않습니다.

중요

OpenShift Container Platform 노드 또는 **install-config.yaml** 파일에서 **hyperthreading**을 비활성화하는 경우 용량 계획에서 머신 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.

4

사용자 프로비저닝 인프라에 **OpenShift Container Platform**을 설치할 때 이 값을 **0**으로 설정해야 합니다. 설치 프로그램에서 제공하는 설치에서 매개 변수는 클러스터가 생성 및 관리하는 컴퓨팅 머신 수를 제어합니다. 사용자 프로비저닝 설치에서는 클러스터 설치를 완료하기 전에 컴퓨팅 시스템을 수동으로 배포해야 합니다.

참고

3-노드 클러스터를 설치하는 경우 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템을 설치할 때 컴퓨팅 머신을 배포하지 마십시오.

7

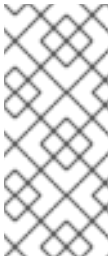
클러스터에 추가하는 컨트롤 플레인 시스템의 수입니다. 클러스터에서 이 값을 클러스터의 etcd 끝점 수로 사용하므로 이 값은 배포하는 컨트롤 플레인 시스템의 수와 일치해야 합니다.

8

DNS 레코드에 지정한 클러스터 이름입니다.

9

Pod IP 주소가 할당되는 IP 주소 블록입니다. 이 블록은 기존 물리적 네트워크와 중복되지 않아야합니다. 이러한 IP 주소는 Pod 네트워크에 사용됩니다. 외부 네트워크에서 Pod에 액세스해야 하는 경우, 트래픽을 관리하도록 로드 밸런서와 라우터를 설정해야 합니다.



참고

클래스 E CIDR 범위는 향후 사용을 위해 예약되어 있습니다. 클래스 E CIDR 범위를 사용하려면 네트워킹 환경에서 클래스 E CIDR 범위 내에서 IP 주소를 수락하는지 확인해야 합니다.

10

개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 hostPrefix를 23으로 설정하면 지정된 cidr 이외 /23 서브넷이 각 노드에 할당되어 510(2^(32 - 23) - 2) Pod IP 주소가 허용됩니다. 외부 네트워크에서 노드에 액세스해야 하는 경우 트래픽을 관리하도록 로드 밸런서와 라우터를 구성합니다.

11

서비스 IP 주소에 사용할 IP 주소 풀입니다. IP 주소 풀은 하나만 입력할 수 있습니다. 이 블록은 기존 물리적 네트워크와 중복되지 않아야합니다. 외부 네트워크에서 서비스에 액세스해야 하는 경우, 트래픽을 관리하도록 로드 밸런서와 라우터를 구성합니다.

12

플랫폼을 none으로 설정해야 합니다. IBM Z 인프라에 대한 추가 플랫폼 구성 변수는 지정할 수 없습니다.

중요

플랫폼 유형으로 설치된 클러스터는 **Machine API**로 컴퓨팅 머신 관리와 같은 일부 기능을 사용할 수 없습니다. 이 제한은 클러스터에 연결된 컴퓨팅 시스템이 일반적으로 기능을 지원하는 플랫폼에 설치된 경우에도 적용됩니다. 이 매개변수는 설치 후 변경할 수 없습니다.

13

FIPS 모드 활성화 또는 비활성화 여부입니다. 기본적으로 **FIPS** 모드는 비활성화됩니다. **FIPS** 모드가 활성화되면 **OpenShift Container Platform**이 실행되는 **RHCOS(Red Hat Enterprise Linux CoreOS)** 시스템에서 기본 **Kubernetes** 암호화 제품군은 우회하고 **RHCOS**와 함께 제공되는 암호화 모듈을 대신 사용합니다.

중요

FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 **OpenShift Container Platform** 배포에서만 지원됩니다.

14

<local_registry>는 미러 레지스트리가 해당 내용을 제공하는 데 사용하는 레지스트리 도메인 이름과 포트(선택사항)를 지정합니다. 예: **registry.example.com** 또는 **registry.example.com:5000 <credentials>**는 미러 레지스트리의 **base64** 인코딩 사용자 이름과 암호를 지정합니다.

15

RHCOS(Red Hat Enterprise Linux CoreOS)의 **core** 사용자에 대한 **SSH** 공용 키입니다.

참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 **OpenShift Container Platform** 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 **SSH** 키를 지정합니다.

16

additionalTrustBundle 매개변수와 값을 추가합니다. 값은 미러 레지스트리에 사용한 인증서 파일의 내용이어야 하며, 신뢰할 수 있는 기존 인증 기관 또는 미러 레지스트리에 대해 생성한 자체 서명 인증서일 수 있습니다.

17

명령 출력에서 **imageContentSources** 섹션을 제공하여 리포지토리를 미러링하십시오.

12.3.8.3. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 **HTTP** 또는 **HTTPS** 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 **OpenShift Container Platform** 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 **API**에 대한 호출을 포함하여 모든 클러스터 발신(**Egress**) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.

참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), **Google Cloud Platform (GCP)**, **Microsoft Azure** 및 **Red Hat OpenStack Platform (RHOSP)**에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
```

```
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1

클러스터 외부에서 **HTTP** 연결을 구축하는 데 사용할 프록시 **URL**입니다. **URL** 스키마는 **http**여야 합니다.

2

클러스터 외부에서 **HTTPS** 연결을 구축하는 데 사용할 프록시 **URL**입니다.

3

대상 도메인 이름, **IP** 주소 또는 프록시에서 제외할 기타 네트워크 **CIDR**로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. ***를 사용하여 모든 대상에 대해 프록시를 바이패스합니다.

4

이 값을 제공하면 설치 프로그램에서 추가 **CA** 인증서를 보유할 **openshift -config** 네임스페이스에 **user-ca- bundle** 이라는 구성 맵을 생성합니다. **additionalTrustBundle** 및 하나 이상의 프록시 설정을 제공하는 경우 프록시 오브젝트는 **trustedCA** 필드의 **user-ca-bundle** 구성 맵을 참조하도록 구성됩니다. 그러면 **Cluster Network Operator**에서 **trustedCA** 매개변수에 대해 지정된 콘텐츠를 **RHCOS** 신뢰 번들과 병합하는 **trusted-ca-bundle** 구성 맵을 생성합니다. 프록시의 **ID** 인증서를 **RHCOS** 트러스트 번들에 있는 기관에서 서명하지 않은 경우 **additionalTrustBundle** 필드가 있어야 합니다.

참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 **OpenShift Container Platform**을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 **nil**이 됩니다.

참고

cluster라는 Proxy 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

12.3.8.4. 3개의 노드 클러스터 구성

필요한 경우 세 개의 컨트롤 플레인 시스템으로 구성된 베어 메탈 클러스터에 제로 컴퓨팅 머신을 배포할 수 있습니다. 이를 통해 클러스터 관리자와 개발자들이 테스트, 개발, 프로덕션에 사용할 수 있는 소형화되고 리소스 효율이 높은 클러스터를 제공합니다.

3-노드 OpenShift Container Platform 환경에서 세 개의 컨트롤 플레인 머신을 예약할 수 있습니다. 즉, 애플리케이션 워크로드가 해당 플랫폼에서 실행되도록 예약됩니다.

사전 요구 사항

- 기존 install-config.yaml 파일이 있습니다.

절차

- install-config.yaml 파일에서 다음 compute 스탠자에 표시된 대로 컴퓨팅 복제본 수가 0으로 설정되어 있는지 확인합니다.

```
compute:
- name: worker
  platform: {}
  replicas: 0
```

참고

배포 중인 컴퓨팅 머신 수에 관계없이 사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치할 때 컴퓨팅 머신의 replicas 매개변수 값을 0으로 설정해야 합니다. 설치 프로그램에서 제공하는 설치에서 매개 변수는 클러스터가 생성 및 관리하는 컴퓨팅 머신 수를 제어합니다. 이 설정은 컴퓨팅 시스템이 수동으로 배포되는 사용자 프로비저닝 설치에는 적용되지 않습니다.

참고

컨트롤 플레인 노드의 기본 리소스는 vCPU 6개와 21GB입니다. 3개의 컨트롤 플레인 노드의 경우 메모리 + vCPU는 최소 5-노드 클러스터와 동등합니다. SMT2가 활성화된 IFL 3개와 함께 각각 120GB 디스크에 설치된 3개의 노드를 백업해야 합니다. 테스트된 최소 설정은 각 컨트롤 플레인 노드에 대해 120GB 디스크에서 3개의 vCPU와 10GB입니다.

3-노드 클러스터 설치의 경우 다음 단계를 따르십시오.

- 컴퓨팅 노드가 0인 3-노드 클러스터를 배포하는 경우 Ingress 컨트롤러 Pod는 컨트롤 플레인 노드에서 실행됩니다. 3-노드 클러스터 배포에서 HTTP 및 HTTPS 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다. 자세한 내용은 *사용자 프로비저닝 인프라 섹션에 대한 로드 밸런싱 요구 사항* 섹션을 참조하십시오.

- 다음 절차에서 Kubernetes 매니페스트 파일을 생성할 때 <installation_directory>/manifests/cluster-scheduler-02-config.yml 파일의 mastersSchedulable 매개변수가 true로 설정되어 있는지 확인합니다. 이렇게 하면 애플리케이션 워크로드를 컨트롤 플레인 노드에서 실행할 수 있습니다.

- RHCOS(Red Hat Enterprise Linux CoreOS) 시스템을 생성할 때 컴퓨팅 노드를 배포하지 마십시오.

12.3.9. CNO(Cluster Network Operator) 구성

클러스터 네트워크의 구성은 CNO(Cluster Network Operator) 구성의 일부로 지정되며 cluster라는 이름의 CR(사용자 정의 리소스) 오브젝트에 저장됩니다. CR은 operator.openshift.io API 그룹에서 Network API의 필드를 지정합니다.

CNO 구성은 Network.config.openshift.io API 그룹의 Network API에서 클러스터 설치 중에 다음 필드를 상속하며 이러한 필드는 변경할 수 없습니다.

clusterNetwork

Pod IP 주소가 할당되는 IP 주소 풀입니다.

serviceNetwork

서비스를 위한 IP 주소 풀입니다.

defaultNetwork.type

OpenShift SDN 또는 OVN-Kubernetes와 같은 클러스터 네트워크 공급자입니다.

cluster라는 CNO 오브젝트에서 defaultNetwork 오브젝트의 필드를 설정하여 클러스터의 클러스터 네트워크 공급자 구성을 지정할 수 있습니다.

12.3.9.1. CNO(Cluster Network Operator) 구성 오브젝트

CNO(Cluster Network Operator)의 필드는 다음 표에 설명되어 있습니다.

표 12.29. CNO(Cluster Network Operator) 구성 오브젝트

필드	유형	설명
metadata.name	**string**	CNO 개체 이름입니다. 이 이름은 항상 **cluster**입니다.

spec.clusterNetwork	**array**	Pod IP 주소가 할당되는 IP 주소 블록과 클러스터의 각 개별 노드에 할당된 서브넷 접두사 길이를 지정하는 목록입니다. 예를 들면 다음과 같습니다.
		`spec:`
		`  clusterNetwork:`
		`  - cidr: 10.128.0.0/19`
		`  hostPrefix: 23`
		`  - cidr: 10.128.32.0/19`
		`  hostPrefix: 23`
		매니페스트를 생성하기 전에 **install-config.yaml** 파일에서만 이 필드를 사용자 지정할 수 있습니다. 값은 매니페스트 파일에서 읽기 전용입니다.

spec.serviceNetwork	**array**	서비스의 IP 주소 블록입니다. OpenShift SDN 및 OVN-Kubernetes CNI(Container Network Interface) 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다. 예를 들면 다음과 같습니다.
		`spec:`
		`  serviceNetwork:`
		`  - 172.30.0.0/14`
		매니페스트를 생성하기 전에 **install-config.yaml** 파일에서만 이 필드를 사용자 지정할 수 있습니다. 값은 매니페스트 파일에서 읽기 전용입니다.

spec.defaultNetwork	**object**	클러스터 네트워크의 CNI(Container Network Interface) 클러스터 네트워크 공급자를 구성합니다.

필드	유형	설명
spec.kubeProxyConfig	**object**	이 개체의 필드는 kube-proxy 구성을 지정합니다. OVN-Kubernetes 클러스터 네트워크 공급자를 사용하는 경우 kube-proxy 구성이 적용되지 않습니다.

defaultNetwork 오브젝트 구성

defaultNetwork 오브젝트의 값은 다음 표에 정의되어 있습니다.

표 12.30. defaultNetwork 오브젝트

필드	유형	설명
type	**string**	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, 클러스터 네트워크 공급자가 설치 중에 선택됩니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다. **참고** OpenShift Container Platform은 기본적으로 OpenShift SDN CNI(Container Network Interface) 클러스터 네트워크 공급자를 사용합니다.
openshiftSDNConfig	**object**	이 오브젝트는 OpenShift SDN 클러스터 네트워크 공급자에만 유효합니다.
ovnKubernetesConfig	**object**	이 오브젝트는 OVN-Kubernetes 클러스터 네트워크 공급자에만 유효합니다.

OpenShift SDN CNI 네트워크 공급자에 대한 구성

다음 표에서는 OpenShift SDN Container Network Interface (CNI) 클러스터 네트워크 공급자의 구성 필드를 설명합니다.

표 12.31. openshiftSDNConfig 오브젝트

필드	유형	설명

필드	유형	설명
mode	**string**	OpenShift SDN의 네트워크 격리 모드를 구성합니다. 기본값은 **NetworkPolicy**입니다. **Multitenant** 및 **Subnet** 값은 OpenShift Container Platform 3.x 버전과의 역호환할 수 있지만 권장되지는 않습니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
mtu	**integer**	VXLAN 오버레이 네트워크의 최대 전송 단위(MTU)입니다. 이는 기본 네트워크 인터페이스의 MTU를 기준으로 자동 탐지됩니다. 일반적으로 감지된 MTU를 재정의할 필요는 없습니다. 자동 감지 값이 예상 밖인 경우 노드의 기본 네트워크 인터페이스의 MTU가 올바른지 확인합니다. 이 옵션을 사용하여 노드의 기본 네트워크 인터페이스의 MTU 값을 변경할 수 없습니다. 클러스터에 다른 노드에 대한 다른 MTU 값이 필요한 경우 이 값을 클러스터의 가장 낮은 MTU 값보다 **50** 미만으로 설정해야 합니다. 예를 들어 클러스터의 일부 노드에는 **9001**의 MTU가 있고 일부에는 **1500**의 MTU가 있는 경우 이 값을 **1450**으로 설정해야 합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
vxlanPort	**integer**	모든 VXLAN 패킷에 사용할 포트입니다. 기본값은 **4789**입니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다. 기존 노드가 다른 VXLAN 네트워크에 속하는 가상 환경에서 실행 중인 경우에는 기본값을 변경해야 할 수도 있습니다. 예를 들어 VMware NSX-T 위에서 OpenShift SDN 오버레이를 실행할 때 두 SDN이 동일한 기본 VXLAN 포트 번호를 사용하므로 VXLAN의 대체 포트를 선택해야 합니다. AWS(Amazon Web Services)에서는 포트 **9000**과 포트 **9999** 사이에서 VXLAN의 대체 포트를 선택할 수 있습니다.

OpenShift SDN 구성 예

```
defaultNetwork:
  type: OpenShiftSDN
  openshiftSDNConfig:
    mode: NetworkPolicy
    mtu: 1450
    vxlanPort: 4789
```

OVN-Kubernetes CNI 클러스터 네트워크 공급자에 대한 구성

다음 표에서는 **OVN-Kubernetes CNI** 클러스터 네트워크 공급자의 구성 필드를 설명합니다.

표 12.32. ovnKubernetesConfig object

필드	유형	설명
mtu	**integer**	Geneve(Generic Network Virtualization Encapsulation) 오버레이 네트워크의 MTU(최대 전송 단위)입니다. 이는 기본 네트워크 인터페이스의 MTU를 기준으로 자동 탐지됩니다. 일반적으로 감지된 MTU를 재정의할 필요는 없습니다. 자동 감지 값이 예상 밖인 경우 노드의 기본 네트워크 인터페이스의 MTU가 올바른지 확인합니다. 이 옵션을 사용하여 노드의 기본 네트워크 인터페이스의 MTU 값을 변경할 수 없습니다. 클러스터에 다른 노드에 대한 다른 MTU 값이 필요한 경우, 이 값을 클러스터의 가장 낮은 MTU 값보다 **100** 미만으로 설정해야 합니다. 예를 들어, 클러스터의 일부 노드에 **9001**의 MTU가 있고 일부에는 **1500**의 MTU가 있는 경우 이 값을 **1400**으로 설정해야 합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
genevePort	**integer**	모든 Geneve 패킷에 사용할 포트입니다. 기본값은 **6081**입니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
policyAuditConfig	**object**	네트워크 정책 감사 로깅을 사용자 정의할 구성 오브젝트를 지정합니다. 설정되지 않으면 기본값 감사 로그 설정이 사용됩니다.

표 12.33. policyAuditConfig object

필드	유형	설명
rateLimit	integer	노드당 1초마다 생성할 최대 메시지 수입니다. 기본값은 초당 **20** 개의 메시지입니다.
maxFileSize	integer	감사 로그의 최대 크기(바이트)입니다. 기본값은 **50000000** 또는 **50** MB입니다.
대상	string	다음 추가 감사 로그 대상 중 하나입니다. **libc** 호스트에서 journald 프로세스의 libc **syslog()** 함수입니다. **udp:<host>:<port>** syslog 서버입니다. **<host>:<port>**를 syslog 서버의 호스트 및 포트로 바꿉니다. **unix:<file>** **<file>**로 지정된 Unix Domain Socket 파일입니다. **null** 감사 로그를 추가 대상으로 보내지 마십시오.

필드	유형	설명
syslogFacility	string	RFC5424에 정의된 **kern**과 같은 syslog 기능입니다. 기본값은 **local0**입니다.

OVN-Kubernetes 구성 예

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
```

kubeProxyConfig 오브젝트 구성

kubeProxyConfig 오브젝트의 값은 다음 표에 정의되어 있습니다.

표 12.34. kubeProxyConfig object

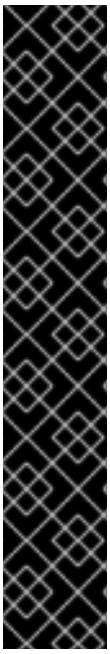
필드	유형	설명
iptablesSyncPeriod	**string**	**iptables** 규칙의 새로 고침 간격입니다. 기본값은 **30s**입니다. 유효 접미사로 **s**, **m**, **h**가 있으며, 자세한 설명은 Go **time** 패키지 문서를 참조하십시오.  **참고** OpenShift Container Platform 4.3 이상에서는 성능이 개선되어 더 이상 **iptablesSyncPeriod** 매개변수를 조정할 필요가 없습니다.

proxyArguments.iptables-min-sync-period	**array**	**iptables** 규칙을 새로 고치기 전 최소 기간입니다. 이 필드를 통해 새로 고침 간격이 너무 짧지 않도록 조정할 수 있습니다. 유효 접미사로 **s**, **m**, **h**가 있으며, 자세한 설명은 Go **time** 패키지를 참조하십시오. 기본값은 다음과 같습니다.
		
		`kubeProxyConfig:`
		`  proxyArguments:`
		`  iptables-min-sync-period:`
		`  - 0s`

12.3.10. Kubernetes 매니페스트 및 Ignition 설정 파일 생성

일부 클러스터 정의 파일을 수정하고 클러스터 시스템을 수동으로 시작해야 하므로 클러스터가 시스템을 구성하는 데 필요한 **Kubernetes** 매니페스트 및 **Ignition** 구성 파일을 사용자가 생성해야 합니다.

설치 구성 파일은 **Kubernetes** 매니페스트로 변환됩니다. 매니페스트는 나중에 클러스터 머신을 구성하는 데 사용되는 **Ignition** 구성 파일로 래핑됩니다.



중요

- **OpenShift Container Platform** 설치 프로그램에서 생성하는 **Ignition** 구성 파일에 **24**시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 **24**시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 **kubelet** 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(**CSR**)을 수동으로 승인해야 합니다. 자세한 내용은 *만료된 컨트롤 플레인 인증서에서 복구* 문서를 참조하십시오.

- 클러스터를 설치한 후 **24**시간에서 **22**시간까지의 인증서가 교체되기 때문에 생성된 후 **12**시간 이내에 **Ignition** 구성 파일을 사용하는 것이 좋습니다. **12**시간 이내에 **Ignition** 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

참고

매니페스트 및 **Ignition** 파일을 생성하는 설치 프로그램은 아키텍처에 따라 다르며 클라이언트 이미지 미러에서 얻을 수 있습니다. 설치 프로그램의 **Linux** 버전은 **s390x**에서만 실행됩니다. 이 설치 프로그램은 **Mac OS** 버전으로도 사용할 수 있습니다.

사전 요구 사항

- **OpenShift Container Platform** 설치 프로그램을 가져오셨습니다. 제한된 네트워크 설치의 경우, 해당 파일은 미러 호스트에 있습니다.

- **install-config.yaml** 설치 구성 파일을 생성하셨습니다.

프로세스

1. **OpenShift Container Platform** 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터에 대한 **Kubernetes** 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1

<installation_directory>는 사용자가 만든 install-config.yaml 파일이 포함된 설치 디렉터리를 지정합니다.

주의

3 노드 클러스터를 실행 중이면 다음 단계를 건너 뛰어 컨트롤 플레인 노드 일정을 계획할 수 있도록 하십시오.

중요

예약할 수 없는 기본에서 컨트롤 플레인 노드를 구성하면 추가 서브스크립션이 필요합니다. 이는 컨트롤 플레인 노드가 작업자 노드가 되기 때문입니다.

2. <installation_directory>/manifests/cluster-scheduler-02-config.yml Kubernetes 매니페스트 파일의 mastersSchedulable 매개변수가 false로 설정되어 있는지 확인합니다. 이 설정으로 인해 컨트롤 플레인 머신에서 포드가 예약되지 않습니다.

   a. <installation_directory>/manifests/cluster-scheduler-02-config.yml 파일을 엽니다.

   b. mastersSchedulable 매개변수를 찾아서 값을 False로 설정되어 있는지 확인합니다.

   c. 파일을 저장하고 종료합니다.

3. Ignition 구성 파일을 생성하려면 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행합니다.

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1

<installation_directory>는 동일한 설치 디렉터리를 지정합니다.

설치 디렉터리의 부트스트랩, 컨트롤 플레인 및 컴퓨팅 노드에 대한 Ignition 구성 파일이 생성됩니다. kubeadmin-password 및 kubeconfig 파일은 ./<installation_directory>/auth 디렉터리에 생성됩니다.

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

12.3.11. RHCOS 설치 및 OpenShift Container Platform 부트스트랩 프로세스 시작

프로비저닝하는 IBM Z 인프라에 OpenShift Container Platform을 설치하려면 z/VM 게스트 가상 머신에 RHCOS(Red Hat Enterprise Linux CoreOS)를 설치해야 합니다. RHCOS를 설치할 때 설치 중인 머신 유형에 대해 OpenShift Container Platform 설치 프로그램에서 생성한 Ignition 구성 파일을 제공해야 합니다. 적합한 네트워킹, DNS 및 로드 밸런싱 인프라를 구성한 경우 RHCOS z/VM 게스트 가상 머신이 재부팅된 후 OpenShift Container Platform 부트스트랩 프로세스가 자동으로 시작됩니다.

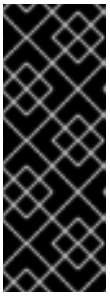
머신을 생성하려면 다음 단계를 완료하십시오.

사전 요구 사항

- 생성한 머신에 액세스할 수 있는 프로비저닝 머신에서 실행 중인 HTTP 또는 HTTPS 서버.

프로세스

1. 프로비저닝 머신에서 Linux에 로그인합니다.
2. RHCOS 이미지 미러 에서 RHCOS(Red Hat Enterprise Linux CoreOS) 커널, initramfs 및 rootfs 파일을 가져옵니다.

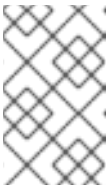


중요

RHCOS 이미지는 OpenShift Container Platform 릴리스에 따라 변경되지 않을 수 있습니다. 설치하는 OpenShift Container Platform 버전과 같거나 그 이하의 버전 중 가장 최신 버전의 이미지를 다운로드해야 합니다. 이 프로세스에는 아래 설명된 적절한 kernel, initramfs 및 rootfs 아티팩트만 사용하십시오.

OpenShift Container Platform 버전 번호가 파일 이름에 포함됩니다. 다음 예와 유사합니다.

- kernel: rhcos-<version>-live-kernel-<architecture>
- initramfs: rhcos-<version>-live-initramfs.<architecture>.img
- rootfs: rhcos-<version>-live-rootfs.<architecture>.img



참고

rootfs 이미지는 FCP 및 DASD에 대해 동일합니다.

3. 매개 변수 파일을 생성합니다. 다음 매개 변수는 특정 가상 머신에 지정해야 합니다.

- ip=에 다음 7 개의 항목을 지정하십시오.
  i. 컴퓨터의 IP 주소
  ii. 빈 문자열
  iii. 게이트웨이
  iv. 넷 마스크

v. **hostname.domainname 형식의 시스템 호스트 및 도메인 이름. RHCOS가 설정하도록 하려면 이 값을 생략하십시오.**

vi. **네트워크 인터페이스 이름. RHCOS가 설정하도록 하려면 이 값을 생략하십시오.**

vii. **고정 IP 주소를 사용하는 경우 none을 지정합니다.**

- **coreos.inst.ignition_url=의 경우 시스템 역할의 Ignition 파일을 지정합니다. bootstrap.ign, master.ign 또는 worker.ign을 사용하십시오. HTTP 및 HTTPS 프로토콜만 지원됩니다.**

- **coreos.live.rootfs_url=의 경우 부팅 중인 커널 및 initramfs와 일치하는 rootfs 아티팩트를 지정합니다. HTTP 및 HTTPS 프로토콜만 지원됩니다.**

- **DASD 유형 디스크에 설치하려면 다음 작업을 완료합니다.**

  i. **coreos.inst.install_dev=의 경우 dasda를 지정합니다.**

  ii. **rd.dasd=의 경우 RHCOS를 설치할 DASD를 지정합니다.**

  iii. **변경되지 않은 다른 모든 매개변수는 그대로 두십시오.**

  부트스트랩 시스템의 매개 변수 파일 예 bootstrap-0.parm

```
rd.neednet=1 \
console=ttysclp0 \
coreos.inst.install_dev=dasda \
coreos.live.rootfs_url=http://cl1.provide.example.com:8080/assets/rhcos-live-
rootfs.s390x.img \
coreos.inst.ignition_url=http://cl1.provide.example.com:8080/ignition/bootstra
p.ign \
ip=172.18.78.2::172.18.78.1:255.255.255.0:::none nameserver=172.18.78.1 \
rd.znet=qeth,0.0.bdf0,0.0.bdf1,0.0.bdf2,layer2=1,portno=0 \
zfcp.allow_lun_scan=0 \
rd.dasd=0.0.3490
```

매개 변수 파일의 모든 옵션을 한 줄로 작성하고 줄 바꿈 문자가 없는지 확인합니다.

- FCP 유형 디스크에 설치하려면 다음 작업을 완료합니다.

  i. RHCOS를 설치할 FCP 디스크를 지정하려면 rd.zfcp=<adapter>,<wwpn>,<lun>을 사용합니다. 멀티패스의 경우 추가 경로마다 이 단계를 반복합니다.

  참고

  여러 경로를 사용하여 설치할 때 나중에 문제가 발생할 수 있으므로 설치 후에 직접 멀티패스를 활성화해야 합니다.

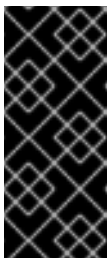
  ii. 설치 장치를 coreos.inst.install_dev=sda로 설정합니다.

  참고

  NPIV로 추가 LUN을 구성하는 경우 FCP에는 zfcp.allow_lun_scan=0이 필요합니다. 예를 들어 CSI 드라이버를 사용하므로 zfcp.allow_lun_scan=1 을 활성화해야 하는 경우, 각 노드가 다른 노드의 부팅 파티션에 액세스할 수 없도록 NPIV를 구성해야 합니다.

  iii. 변경되지 않은 다른 모든 매개변수는 그대로 두십시오.

  중요

  멀티패스를 완전히 활성화하려면 추가 설치 후 단계가 필요합니다. 자세한 내용은 *설치 후 머신 구성 작업*의 "RHCOS에서 커널 인수를 사용하여 다중 경로 활성화"를 참조하십시오.

다음은 다중 경로가 있는 작업자 노드의 예제 매개변수 파일 worker-1.parm입니다.

```
rd.neednet=1 \
console=ttysclp0 \
coreos.inst.install_dev=sda \
coreos.live.rootfs_url=http://cl1.provide.example.com:8080/assets/rhcos-live-
```

```
rootfs.s390x.img \
coreos.inst.ignition_url=http://cl1.provide.example.com:8080/ignition/worker.ign \
ip=172.18.78.2::172.18.78.1:255.255.255.0:::none nameserver=172.18.78.1 \
rd.znet=qeth,0.0.bdf0,0.0.bdf1,0.0.bdf2,layer2=1,portno=0 \
zfcp.allow_lun_scan=0 \
rd.zfcp=0.0.1987,0x50050763070bc5e3,0x4008400B00000000 \
rd.zfcp=0.0.19C7,0x50050763070bc5e3,0x4008400B00000000 \
rd.zfcp=0.0.1987,0x50050763071bc5e3,0x4008400B00000000 \
rd.zfcp=0.0.19C7,0x50050763071bc5e3,0x4008400B00000000
```

매개 변수 파일의 모든 옵션을 한 줄로 작성하고 줄 바꿈 문자가 없는지 확인합니다.

4. **initramfs**, 커널, 매개 변수 파일 및 **RHCOS** 이미지를 **z/VM**에 전송합니다 (예: **FTP** 사용). **FTP**를 사용하여 파일을 전송하고 가상 리더에서 부팅하는 방법에 대한 자세한 내용은 **Z/VM**에서 설치를 참조하십시오.

5. 부트스트랩 노드가 될 **z/VM** 게스트 가상 머신의 가상 리더에 파일 **punch**를 실행합니다.

   IBM 문서의 **PUNCH**를 참조하십시오.

   작은 정보

   CP PUNCH 명령을 사용하거나 **Linux**를 사용하는 경우 **vmur** 명령을 사용하여 두 개의 **z/VM** 게스트 가상 머신간에 파일을 전송할 수 있습니다.

6. 부트스트랩 시스템에서 **CMS**에 로그인합니다.

7. 리더에서 부트스트랩 머신에 대해 **IPL**을 수행합니다.

   ```
   $ ipl c
   ```

   IBM 문서의 **IPL** 을 참조하십시오.

8. 클러스터의 다른 컴퓨터에 대해 이 프로세스를 반복합니다.

12.3.11.1. 고급 RHCOS 설치 참조

여기서는 **RHCOS(Red Hat Enterprise Linux CoreOS)** 수동 설치 프로세스를 수정하는 데 사용할 수 있는 네트워킹 구성 및 기타 고급 옵션에 대해 설명합니다. 다음 표에서는 **RHCOS** 라이브 설치 프로그램 및 **coreos-installer** 명령과 함께 사용할 수있는 커널 인수 및 명령 줄 옵션에 대해 설명합니다.

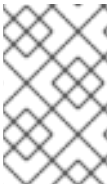
12.3.11.1.1. ISO 설치를 위한 네트워킹 및 본딩 옵션

ISO 이미지에서 **RHCOS**를 설치하는 경우, 해당 이미지를 부팅할 때 수동으로 커널 인수를 추가하여 노드의 네트워킹을 구성할 수 있습니다. 네트워킹 인수를 지정하지 않으면 **RHCOS**에서 **Ignition** 구성 파일을 가져오는 데 네트워킹이 필요함을 감지하면 **initramfs**에서 **DHCP**가 활성화됩니다.

중요

네트워킹 인수를 수동으로 추가할 때 **initramfs**에서 네트워크를 가져오려면 **rd.neednet=1** 커널 인수도 추가해야 합니다.

다음 표는 **ISO** 설치를 위해 **RHCOS(Red Hat Enterprise Linux CoreOS)** 노드의 네트워킹 및 본딩 구성 예를 보여줍니다. 예제에서는 **ip=, nameserver=, bond=** 커널 인수를 사용하는 방법을 설명합니다.

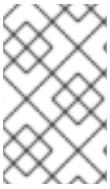


참고

커널 인수를 추가할 때 순서가 중요합니다: **ip=, nameserver=** 및 **bond=** 입니다.

이는 시스템 부팅 중에 **dracut** 툴로 전달되는 네트워킹 옵션입니다. **dracut**에서 지원하는 네트워킹 옵션에 대한 자세한 내용은 **dracut.cmdline** 메뉴얼 페이지를 참조하십시오.

다음 정보는 **ISO** 설치를 위해 **RHCOS** 노드에서 네트워킹을 구성하는 예제를 제공합니다. 예제에서는 **ip=** 및 **nameserver=** 커널 인수를 사용하는 방법을 설명합니다.



참고

커널 인수를 추가할 때 순서가 중요합니다(**ip =** 및 **nameserver=**).

이는 시스템 부팅 중에 **dracut** 툴로 전달되는 네트워킹 옵션입니다. **dracut**에서 지원하는 네트워킹 옵션에 대한 자세한 내용은 **dracut.cmdline** 메뉴얼 페이지를 참조하십시오.

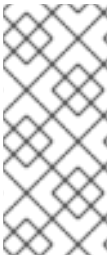
다음 예제는 ISO 설치를 위한 네트워킹 옵션입니다.

DHCP 또는 고정 IP 주소 구성

IP 주소를 구성하려면 DHCP(ip=dhcp)를 사용하거나 개별 고정 IP 주소(ip=<host_ip>)를 설정합니다. 정적 IP를 설정하는 경우 각 노드에서 DNS 서버 IP 주소 (nameserver=<dns_ip>)를 확인합니다. 다음 예제에서는 다음을 설정합니다.

- 노드의 IP 주소는 10.10.10.2로 설정됩니다.
- 게이트웨이 주소는 10.10.10.254로 설정됩니다.
- 넷마스크는 255.255.255.0으로
- core0.example.com에 대한 호스트 이름
- 4.4.4.41의 DNS 서버 주소
- auto-configuration 값을 none으로 설정합니다. IP 네트워킹이 정적으로 구성되면 자동 구성이 필요하지 않습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
nameserver=4.4.4.41
```



참고

DHCP를 사용하여 RHCOS 시스템의 IP 주소 지정을 구성하는 경우 시스템은 DHCP를 통해 DNS 서버 정보도 가져옵니다. DHCP 기반 배포의 경우 DHCP 서버 구성을 통해 RHCOS 노드에서 사용할 DNS 서버 주소를 정의할 수 있습니다.

고정 호스트 이름 없이 IP 주소 구성

정적 호스트 이름을 할당하지 않고 IP 주소를 구성할 수 있습니다. 사용자가 정적 호스트 이름을 설정하지 않으면 역방향 DNS 조회에 의해 자동으로 선택됩니다. 정적 호스트 이름이 없는 IP 주소를 구성하려면 다음 예제를 참조합니다.

- 노드의 IP 주소는 10.10.10.2로 설정됩니다.
- 게이트웨이 주소는 10.10.10.254로 설정됩니다.
- 넷마스크는 255.255.255.0으로
- 4.4.4.41의 DNS 서버 주소
- auto-configuration 값을 none으로 설정합니다. IP 네트워킹이 정적으로 구성되면 자동 구성이 필요하지 않습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0::enp1s0:none
nameserver=4.4.4.41
```

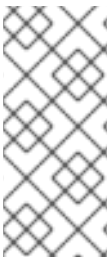
여러 네트워크 인터페이스 지정

여러 ip= 항목을 설정하여 여러 네트워크 인터페이스를 지정할 수 있습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

기본 게이트웨이 및 경로 구성

선택 사항: rd.route= 값을 설정하여 추가 네트워크에 대한 경로를 구성할 수 있습니다.



참고

하나 이상의 네트워크를 구성할 때 하나의 기본 게이트웨이가 필요합니다. 추가 네트워크 게이트웨이가 기본 네트워크 게이트웨이와 다른 경우 기본 게이트웨이가 기본 네트워크 게이트웨이어야 합니다.

- 다음 명령을 실행하여 기본 게이트웨이를 구성합니다.

```
ip=::10.10.10.254::::
```

- 다음 명령을 입력하여 추가 네트워크의 경로를 구성합니다.

-

```
rd.route=20.20.20.0/24:20.20.20.254:enp2s0
```

단일 인터페이스에서 DHCP 비활성화

두 개 이상의 네트워크 인터페이스가 있고 하나의 인터페이스만 사용되는 경우와 같이 단일 인터페이스에서 DHCP를 비활성화할 수 있습니다. 이 예에서 enp1s0 인터페이스에는 정적 네트워킹 구성이 있으며 enp2s0 용으로 DHCP가 사용되지 않습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=::::core0.example.com:enp2s0:none
```

DHCP 및 고정 IP 구성 결합

다음과 같이 시스템의 DHCP 및 고정 IP 구성을 여러 네트워크 인터페이스와 결합할 수 있습니다.

```
ip=enp1s0:dhcp
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

개별 인터페이스에서 VLAN 구성

선택 사항: vlan= 매개변수를 사용하여 개별 인터페이스에서 VLAN을 구성할 수 있습니다.

- 네트워크 인터페이스에서 VLAN을 구성하고 고정 IP 주소를 사용하려면 다음 명령을 실행합니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0.100:none
vlan=enp2s0.100:enp2s0
```

- 네트워크 인터페이스에서 VLAN을 구성하고 DHCP를 사용하려면 다음 명령을 실행합니다.

```
ip=enp2s0.100:dhcp
vlan=enp2s0.100:enp2s0
```

여러 DNS 서버 제공

각 서버에 nameserver= 항목을 추가하여 여러 DNS 서버를 제공할 수 있습니다. 예를 들면 다음과 같습니다.

```
nameserver=1.1.1.1
nameserver=8.8.8.8
```

단일 인터페이스에 여러 네트워크 인터페이스 본딩

선택 사항: bond= 옵션을 사용하여 여러 네트워크 인터페이스를 단일 인터페이스에 결합할 수 있습니다. 다음 예제를 참조하십시오.

- 결합된 인터페이스를 구성하는 구문: **bond = name [: network_interfaces] [: options]**

  name은 결합하는 기기 이름(**bond0**)이고 ***network_interfaces***는 쉼표로 구분되는 물리적(이더넷) 인터페이스 목록(**em1, em2**)이며, ***options***은 쉼표로 구분되는 결합 옵션 목록입니다. 사용 가능한 옵션을 보려면 **modinfo bonding**을 입력하십시오.

- **bond=**를 사용하여 결합된 인터페이스를 생성할 때 **IP** 주소가 할당되는 방법과 결합된 인터페이스에 대한 기타 정보를 지정해야 합니다.

- **DHCP**를 사용하도록 결합된 인터페이스를 구성하려면 **bond**의 **IP** 주소를 **dhcp**로 설정하십시오. 예를 들면 다음과 같습니다.

```
bond=bond0:em1,em2:mode=active-backup
ip=bond0:dhcp
```

- 고정 **IP** 주소를 사용하도록 결합된 인터페이스를 구성하려면 원하는 특정 **IP** 주소 및 관련 정보를 입력합니다. 예를 들면 다음과 같습니다.

```
bond=bond0:em1,em2:mode=active-backup,fail_over_mac=1
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
```

공유 **OSA/RoCE** 카드가 사용될 때 문제를 방지하기 위해 항상 **active-backup** 모드에서 옵션 **fail_over_mac=1** 을 설정합니다.

단일 인터페이스에 여러 네트워크 인터페이스 본딩

선택 사항: **vlan=** 매개변수를 사용하고 **DHCP**를 사용하도록 결합된 인터페이스에서 **VLAN**을 구성할 수 있습니다. 예를 들면 다음과 같습니다.

```
ip=bond0.100:dhcp
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

다음 예제를 사용하여 **VLAN**을 사용하여 결합된 인터페이스를 구성하고 고정 **IP** 주소를 사용합니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0.100:none
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

네트워크 팀 구성 사용

선택 사항: team= 매개변수를 사용하여 네트워크 팀을 결합의 대안으로 사용할 수 있습니다.

- 팀 인터페이스를 구성하는 구문은 team=name[:network_interfaces]입니다.

  name 은 팀 장치 이름(team0)이고 *network_interfaces* 는 쉼표로 구분된 실제(이더넷) 인터페이스 목록(em1, em2)을 나타냅니다.

RHCOS가 향후 RHEL 버전으로 전환되면 티밍이 더 이상 사용되지 않을 예정입니다. 자세한 내용은 이 Red Hat 지식베이스 문서를 참조하십시오.

다음 예제를 사용하여 네트워크 팀을 구성합니다.

```
team=team0:em1,em2
ip=team0:dhcp
```

12.3.12. 부트스트랩 프로세스가 완료될 때까지 대기 중

OpenShift Container Platform 부트스트랩 프로세스는 클러스터 노드가 먼저 디스크에 설치된 영구 RHCOS 환경으로 부팅된 후에 시작됩니다. Ignition 구성 파일을 통해 제공되는 구성 정보는 부트스트랩 프로세스를 초기화하고 머신에 OpenShift Container Platform을 설치하는 데 사용됩니다. 부트스트랩 프로세스가 완료될 때까지 기다려야 합니다.

사전 요구 사항

- 클러스터에 대한 Ignition 구성 파일이 생성되어 있습니다.
- 적합한 네트워크, DNS 및 로드 밸런싱 인프라가 구성되어 있습니다.
- 설치 프로그램을 받아서 클러스터의 Ignition 구성 파일을 생성했습니다.
- 클러스터 머신에 RHCOS를 설치하고 OpenShift Container Platform 설치 프로그램에서 생성된 Ignition 구성 파일을 제공했습니다.

프로세스

1. 부트스트랩 프로세스를 모니터링합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
  --log-level=info 2
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2 다른 설치 세부 사항을 보려면 info 대신 warn, debug 또는 error를 지정합니다.

출력 예

```
INFO Waiting up to 30m0s for the Kubernetes API at
https://api.test.example.com:6443...
INFO API v1.22.1 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

이 명령은 Kubernetes API 서버가 컨트롤 플레인 시스템에서 부트스트랩되었다는 신호를 보낼 때 성공합니다.

2. 부트스트랩 프로세스가 완료된 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다.

중요

이 시점에 로드 밸런서에서 부트스트랩 시스템을 제거해야 합니다. 부트스트랩 머신 자체를 제거하거나 다시 포맷할 수도 있습니다.

12.3.13. CLI를 사용하여 클러스터에 로그인

클러스터 kubeconfig 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. kubeconfig 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러

스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- oc CLI를 설치했습니다.

프로세스

1. kubeadmin 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 oc 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

12.3.14. 머신의 인증서 서명 요청 승인

클러스터에 시스템을 추가하면 추가한 시스템별로 보류 중인 인증서 서명 요청(CSR)이 두 개씩 생성됩니다. 이러한 CSR이 승인되었는지 확인해야 하며, 필요한 경우 이를 직접 승인해야 합니다. 클라이언트 요청을 먼저 승인한 다음 서버 요청을 승인해야 합니다.

사전 요구 사항

- 클러스터에 시스템을 추가했습니다.

프로세스

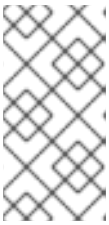
1. 클러스터가 시스템을 인식하는지 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0  Ready    master  63m  v1.22.1
master-1  Ready    master  63m  v1.22.1
master-2  Ready    master  64m  v1.22.1
```

출력에 생성된 모든 시스템이 나열됩니다.



참고

이전 출력에는 일부 **CSR**이 승인될 때까지 컴퓨팅 노드(작업자 노드라고도 함)가 포함되지 않을 수 있습니다.

2. 보류 중인 **CSR**을 검토하고 클러스터에 추가한 각 시스템에 대해 **Pending** 또는 **Approved** 상태의 클라이언트 및 서버 요청이 표시되는지 확인합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE    REQUESTOR                                                   CONDITION
csr-8b2br   15m    system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
```

```
csr-8vnps   15m    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

예에서는 두 시스템이 클러스터에 참여하고 있습니다. 목록에는 승인된 CSR이 더 많이 나타날 수도 있습니다.

3. CSR이 승인되지 않은 경우, 추가된 시스템에 대한 모든 보류 중인 CSR이 Pending 상태로 전환된 후 클러스터 시스템의 CSR을 승인합니다.

참고

CSR은 교체 주기가 자동으로 만료되므로 클러스터에 시스템을 추가한 후 1시간 이내에 CSR을 승인하십시오. 한 시간 내에 승인하지 않으면 인증서가 교체되고 각 노드에 대해 두 개 이상의 인증서가 표시됩니다. 이러한 인증서를 모두 승인해야 합니다. 클라이언트 CSR이 승인되면 Kubelet은 인증서에 대한 보조 CSR을 생성하므로 수동 승인이 필요합니다. 그러면 Kubelet에서 동일한 매개변수를 사용하여 새 인증서를 요청하는 경우 인증서 갱신 요청은 machine-approver에 의해 자동으로 승인됩니다.

참고

베어 메탈 및 기타 사용자 프로비저닝 인프라와 같이 머신 API를 사용하도록 활성화되지 않는 플랫폼에서 실행되는 클러스터의 경우 CSR(Kubelet service Certificate Request)을 자동으로 승인하는 방법을 구현해야 합니다. 요청이 승인되지 않으면 API 서버가 kubelet에 연결될 때 서비스 인증서가 필요하므로 oc exec, oc rsh, oc logs 명령을 성공적으로 수행할 수 없습니다. Kubelet 엔드 포인트에 연결하는 모든 작업을 수행하려면 이 인증서 승인이 필요합니다. 이 방법은 새 CSR을 감시하고 CSR이 system:node 또는 system:admin 그룹의 node-bootstrapper 서비스 계정에 의해 제출되었는지 확인하고 노드의 ID를 확인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 <csr_name>은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

참고

일부 Operator는 일부 CSR이 승인될 때까지 사용할 수 없습니다.

4. 이제 클라이언트 요청이 승인되었으므로 클러스터에 추가한 각 머신의 서버 요청을 검토해야 합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 나머지 CSR이 승인되지 않고 Pending 상태인 경우 클러스터 머신의 CSR을 승인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1

<csr_name>은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

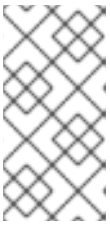
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs oc adm certificate approve
```

6. 모든 클라이언트 및 서버 CSR이 승인된 후 머신은 Ready 상태가 됩니다. 다음 명령을 실행하여 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0  Ready    master  73m  v1.22.1
master-1  Ready    master  73m  v1.22.1
master-2  Ready    master  74m  v1.22.1
worker-0  Ready    worker  11m  v1.22.1
worker-1  Ready    worker  11m  v1.22.1
```



참고

머신이 Ready 상태로 전환하는 데 서버 CSR의 승인 후 몇 분이 걸릴 수 있습니다.

추가 정보

- CSR에 대한 자세한 내용은 인증서 서명 요청을 참조하십시오.

12.3.15. Operator의 초기 설정

컨트롤 플레인이 초기화된 후 일부 Operator를 즉시 구성하여 모두 사용 가능하도록 해야 합니다.

사전 요구 사항

- 컨트롤 플레인이 초기화되어 있습니다.

프로세스

1. 클러스터 구성 요소가 온라인 상태인지 확인합니다.

```
$ watch -n5 oc get clusteroperators
```

출력 예

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.9.0     True        False         False      19m
baremetal                                  4.9.0     True        False         False      37m
cloud-credential                           4.9.0     True        False         False      40m
cluster-autoscaler                         4.9.0     True        False         False      37m
config-operator                            4.9.0     True        False         False      38m
console                                    4.9.0     True        False         False      26m
csi-snapshot-controller                    4.9.0     True        False         False      37m
dns                                        4.9.0     True        False         False      37m
etcd                                       4.9.0     True        False         False      36m
image-registry                             4.9.0     True        False         False      31m
ingress                                    4.9.0     True        False         False      30m
insights                                   4.9.0     True        False         False      31m
kube-apiserver                             4.9.0     True        False         False      26m
kube-controller-manager                    4.9.0     True        False         False      36m
kube-scheduler                             4.9.0     True        False         False      36m
kube-storage-version-migrator              4.9.0     True        False         False      37m
machine-api                                4.9.0     True        False         False      29m
machine-approver                           4.9.0     True        False         False      37m
machine-config                             4.9.0     True        False         False      36m
marketplace                                4.9.0     True        False         False      37m
monitoring                                 4.9.0     True        False         False      29m
network                                    4.9.0     True        False         False      38m
node-tuning                                4.9.0     True        False         False      37m
openshift-apiserver                        4.9.0     True        False         False      32m
openshift-controller-manager               4.9.0     True        False         False      30m
openshift-samples                          4.9.0     True        False         False      32m
operator-lifecycle-manager                 4.9.0     True        False         False      37m
operator-lifecycle-manager-catalog         4.9.0     True        False         False      37m
operator-lifecycle-manager-packageserver   4.9.0     True        False         False      32m
service-ca                                 4.9.0     True        False         False      38m
storage                                    4.9.0     True        False         False      37m
```

2. 사용할 수 없는 **Operator**를 구성합니다.

12.3.15.1. 기본 OperatorHub 소스 비활성화

Red Hat 및 커뮤니티 프로젝트에서 제공하는 콘텐츠를 소싱하는 Operator 카탈로그는 OpenShift Container Platform을 설치하는 동안 기본적으로 OperatorHub용으로 구성됩니다. 제한된 네트워크 환경에서는 클러스터 관리자로서 기본 카탈로그를 비활성화해야 합니다.

프로세스

- OperatorHub 오브젝트에 disableAllDefaultSources: true를 추가하여 기본 카탈로그의 소스를 비활성화합니다.

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

작은 정보

또는 웹 콘솔을 사용하여 카탈로그 소스를 관리할 수 있습니다. 관리 → 클러스터 설정 → 구성 → OperatorHub 페이지에서 개별 소스를 생성, 삭제, 비활성화 및 활성화할 수 있는 소스 탭을 클릭합니다.

12.3.15.2. 이미지 레지스트리 스토리지 구성

기본 스토리지를 제공하지 않는 플랫폼에서는 처음에 Image Registry Operator를 사용할 수 없습니다. 설치한 후에 스토리지를 사용하도록 레지스트리를 구성하여 Registry Operator를 사용 가능하도록 만들어야 합니다.

프로덕션 클러스터에 필요한 영구 볼륨을 구성하는 과정의 지침이 표시됩니다. 해당하는 경우, 프로덕션 환경 외 클러스터에서만 사용할 수 있는 저장 위치로서 빈 디렉터리를 구성하는 과정의 지침이 표시됩니다.

업그레이드 중에 Recreate 롤아웃 전략을 사용하여 이미지 레지스트리의 블록 스토리지 유형 사용을 허용하기 위한 추가 지침이 제공됩니다.

12.3.15.2.1. IBM Z용 레지스트리 스토리지 구성

클러스터 관리자는 설치한 후 스토리지를 사용하도록 레지스트리를 구성해야 합니다.

사전 요구 사항

- cluster-admin 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- IBM Z에 클러스터가 있습니다.
- Red Hat OpenShift Container Storage와 같이 클러스터용 영구 스토리지 프로비저닝이 있습니다.

중요

OpenShift Container Platform은 복제본이 하나만 있는 경우 이미지 레지스트리 스토리지에 대한 ReadWriteOnce 액세스를 지원합니다. ReadWriteOnce 액세스에서는 레지스트리가 Recreate 롤아웃 전략을 사용해야 합니다. 두 개 이상의 복제본으로 고 가용성을 지원하는 이미지 레지스트리를 배포하려면 ReadWriteMany 액세스가 필요합니다.

- "100Gi" 용량이 필요합니다.

프로세스

1. 스토리지를 사용하도록 레지스트리를 구성하기 위해 configs.imageregistry/cluster 리소스에서 spec.storage.pvc를 변경합니다.

참고

공유 스토리지를 사용할 때 보안 설정을 확인하여 외부에서의 액세스를 방지합니다.

2. 레지스트리 pod가 없는지 확인합니다.

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

출력 예

```
No resourses found in openshift-image-registry namespace
```

참고

출력에 레지스트리 Pod가 있는 경우 이 절차를 계속할 필요가 없습니다.

3. 레지스트리 구성을 확인합니다.

```
$ oc edit configs.imageregistry.operator.openshift.io
```

출력 예

```
storage:
  pvc:
    claim:
```

image-registry-storage PVC의 자동 생성을 허용하도록 claim 필드를 비워 둡니다.

4. clusteroperator 상태를 확인합니다.

```
$ oc get clusteroperator image-registry
```

출력 예

```
NAME              VERSION       AVAILABLE   PROGRESSING   DEGRADED   SINCE
MESSAGE
image-registry    4.9           True        False         False      6h50m
```

5. 이미지를 빌드 및 푸시할 수 있도록 레지스트리의 관리가 설정되어 있는지 확인하십시오.

- 다음을 실행합니다.

```
$ oc edit configs.imageregistry/cluster
```

다음으로 라인을 변경하십시오.

```
managementState: Removed
```

다음으로 변경

```
managementState: Managed
```

12.3.15.2.2. 프로덕션 환경 외 클러스터에서 이미지 레지스트리의 스토리지 구성

이미지 레지스트리 **Operator**에 대한 스토리지를 구성해야 합니다. 프로덕션 환경 외 클러스터의 경우, 이미지 레지스트리를 빈 디렉터리로 설정할 수 있습니다. 이렇게 하는 경우 레지스트리를 다시 시작하면 모든 이미지가 손실됩니다.

프로세스

- 이미지 레지스트리 스토리지를 빈 디렉터리로 설정하려면 다음을 수행하십시오.

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch
'{"spec":{"storage":{"emptyDir":{}}}}'
```

주의

프로덕션 환경 외 클러스터에 대해서만 이 옵션을 구성하십시오.

Image Registry Operator가 구성 요소를 초기화하기 전에 이 명령을 실행하면 **oc patch** 명령이 실패하며 다음 오류가 발생합니다.

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster"
not found
```

몇 분 후에 명령을 다시 실행하십시오.

12.3.16. 사용자 프로비저닝 인프라에 설치 완료

Operator 구성을 완료한 후 제공하는 인프라에 클러스터 설치를 완료할 수 있습니다.

사전 요구 사항

- 컨트롤 플레인이 초기화되어 있습니다.
- 초기 Operator 구성을 완료해야 합니다.

프로세스

1. 다음 명령을 사용하여 모든 클러스터 구성 요소가 온라인 상태인지 확인합니다.

```
$ watch -n5 oc get clusteroperators
```

출력 예

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.9.0     True        False         False      19m
baremetal                                  4.9.0     True        False         False      37m
cloud-credential                           4.9.0     True        False         False      40m
cluster-autoscaler                         4.9.0     True        False         False      37m
config-operator                            4.9.0     True        False         False      38m
console                                    4.9.0     True        False         False      26m
csi-snapshot-controller                    4.9.0     True        False         False      37m
dns                                        4.9.0     True        False         False      37m
etcd                                       4.9.0     True        False         False      36m
image-registry                             4.9.0     True        False         False      31m
ingress                                    4.9.0     True        False         False      30m
insights                                   4.9.0     True        False         False      31m
kube-apiserver                             4.9.0     True        False         False      26m
kube-controller-manager                    4.9.0     True        False         False      36m
kube-scheduler                             4.9.0     True        False         False      36m
kube-storage-version-migrator              4.9.0     True        False         False      37m
```

```
machine-api                                4.9.0     True        False         False      29m
machine-approver                           4.9.0     True        False         False      37m
machine-config                             4.9.0     True        False         False      36m
marketplace                                4.9.0     True        False         False      37m
monitoring                                 4.9.0     True        False         False      29m
network                                    4.9.0     True        False         False      38m
node-tuning                                4.9.0     True        False         False      37m
openshift-apiserver                        4.9.0     True        False         False      32m
openshift-controller-manager               4.9.0     True        False         False      30m
openshift-samples                          4.9.0     True        False         False      32m
operator-lifecycle-manager                 4.9.0     True        False         False      37m
operator-lifecycle-manager-catalog         4.9.0     True        False         False      37m
operator-lifecycle-manager-packageserver   4.9.0     True        False         False      32m
service-ca                                 4.9.0     True        False         False      38m
storage                                    4.9.0     True        False         False      37m
```

또는 다음 명령은 모든 클러스터를 사용할 수 있을 때 알립니다. 또한 인증 정보를 검색하고 표시합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1

<installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

출력 예

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator가 Kubernetes API 서버에서 OpenShift Container Platform 클러스터 배포를 완료하면 명령이 성공합니다.

중요

- 설치 프로그램에서 생성하는 **Ignition** 구성 파일에 **24**시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 **24**시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 **kubelet** 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(**CSR**)을 수동으로 승인해야 합니다. 자세한 내용은 *만료된 컨트롤 플레인 인증서에서 복구* 문서를 참조하십시오.

- 클러스터를 설치한 후 **24**시간에서 **22**시간까지의 인증서가 교체되기 때문에 생성된 후 **12**시간 이내에 **Ignition** 구성 파일을 사용하는 것이 좋습니다. **12**시간 이내에 **Ignition** 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

2. **Kubernetes API** 서버가 **Pod**와 통신하고 있는지 확인합니다.

   a. 모든 **Pod** 목록을 보려면 다음 명령을 사용하십시오.

   ```
   $ oc get pods --all-namespaces
   ```

   출력 예

   ```
   NAMESPACE                  NAME                                        READY   STATUS
   RESTARTS   AGE
   openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8
   1/1    Running    1        9m
   openshift-apiserver        apiserver-67b9g                              1/1     Running    0
   3m
   openshift-apiserver        apiserver-ljcmx                              1/1     Running    0
   1m
   openshift-apiserver        apiserver-z25h4                              1/1     Running    0
   2m
   openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8
   1/1    Running    0        5m
   ...
   ```

   b. 다음 명령을 사용하여 이전 명령의 출력에 나열된 **Pod**의 로그를 표시합니다.

```
$ oc logs <pod_name> -n <namespace> 1
```

1

이전 명령의 출력에 표시된 대로 **Pod** 이름과 네임스페이스를 지정합니다.

Pod 로그가 표시되면 **Kubernetes API** 서버는 클러스터 시스템과 통신할 수 있습니다.

3. **FCP(Fibre Channel Protocol)**를 사용하는 설치에는 다중 경로를 활성화하기 위해 추가 단계가 필요합니다. 설치 중에 멀티패스를 활성화하지 마십시오.

   자세한 내용은 *설치 후 머신 구성 작업* 설명서에서 **"RHCOS에서 커널 인수를 사용하여 멀티패스 활성화"**를 참조하십시오.

4. 클러스터 등록 페이지에서 클러스터를 등록합니다.

12.3.17. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 **Telemetry** 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 **Telemetry**가 자동으로 실행되고 **OpenShift Cluster Manager** 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 **OpenShift Cluster Manager**를 사용하여 자동으로 또는 **OpenShift Cluster Manager**를 사용하여 수동으로 유지 관리되는지 확인한 후 **subscription watch**를 사용하여 계정 또는 다중 클러스터 수준에서 **OpenShift Container Platform** 서브스크립션을 추적합니다.

추가 리소스

- **Telemetry** 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

12.3.18. 디버깅 정보 수집

IBM Z에서 **OpenShift Container Platform** 설치와 관련된 특정 문제를 해결하고 디버깅하는데 도움이 될 수 있는 디버깅 정보를 수집할 수 있습니다.

사전 요구 사항

- **oc CLI** 도구가 설치되어 있어야 합니다.

절차

1. 클러스터에 로그인합니다.

```
$ oc login -u <username>
```

2. 하드웨어 정보를 수집하는 노드에서 디버깅 컨테이너를 시작합니다.

```
$ oc debug node/<nodename>
```

3. **/host** 파일 시스템으로 변경하고 **toolbox**를 시작합니다.

```
$ chroot /host
$ toolbox
```

4. **dbginfo** 데이터를 수집합니다.

```
$ dbginfo.sh
```

5. 다음으로 **scp**를 사용하여 데이터를 검색할 수 있습니다.

추가 리소스

- **How to generate SOSREPORT within OpenShift Container Platform version 4 nodes without SSH** 참조.

12.3.19. 다음 단계

- 클러스터를 사용자 지정합니다.
- 클러스터를 설치하는 데 사용한 미러 레지스트리에 신뢰할 수 있는 **CA**가 있는 경우 추가 신뢰 저장소를 구성하여 클러스터에 추가합니다.

13장. IBM Z 및 LINUXONE에 RHEL KVM으로 설치

13.1. IBM Z 및 LINUXONE에 RHEL KVM으로 클러스터 설치 준비

13.1.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.

13.1.2. IBM Z 또는 Linux ONE에 RHEL KVM을 사용하여 OpenShift Container Platform을 설치할 방법 선택

다음 방법 중 하나를 사용하여 프로비저닝하는 IBM Z 또는 LinuxONE 인프라에 RHEL KVM으로 클러스터를 설치할 수 있습니다.

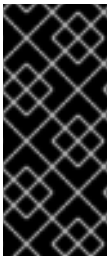
- IBM Z 및 Linux ONE에 RHEL KVM으로 클러스터 설치: 프로비저닝하는 IBM Z 또는 Linux ONE 인프라에 KVM으로 OpenShift Container Platform을 설치할 수 있습니다.
- 네트워크가 제한된 환경에서 IBM Z 및 LinuxONE에 RHEL KVM으로 클러스터 설치: 설치 릴리스 콘텐츠의 내부 미러를 사용하여 RHEL KVM을 사용하여 IBM Z 또는 LinuxONE 인프라에 OpenShift Container Platform을 설치할 수 있습니다. 이 방법을 사용하여 소프트웨어 구성 요소를 받기 위해 활성 인터넷 연결이 필요하지 않은 클러스터를 설치할 수 있습니다. 이 설치 방법을 사용하여 클러스터가 외부 콘텐츠에 대해 조직의 제어 조건을 충족하는 컨테이너 이미지만 사용하도록 할 수 있습니다.

13.2. IBM Z 및 LINUXONE에 RHEL KVM으로 클러스터 설치

OpenShift Container Platform 버전 4.9에서는 사용자가 제공하는 IBM Z 또는 LinuxONE 인프라에 클러스터를 설치할 수 있습니다.

참고

이 문서는 IBM Z에 대해서만 설명하지만 여기에 있는 모든 정보는 LinuxONE에도 적용됩니다.



중요

베어 메탈 이외의 플랫폼의 경우 추가 고려 사항이 있습니다. **OpenShift Container Platform** 클러스터를 설치하기 전에 **guidelines for deploying OpenShift Container Platform on non-tested platforms**에 있는 내용을 확인하십시오.

13.2.1. 사전 요구 사항

- **OpenShift Container Platform** 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.
- 설치 프로세스를 시작하기 전에 설치 디렉터리를 정리해야 합니다. 이렇게 하면 설치 프로세스 중에 필요한 설치 파일이 생성되고 업데이트됩니다.
- 클러스터 용 **NFS**를 사용하여 영구 스토리지를 프로비저닝합니다. 프라이빗 이미지 레지스트리를 배포하려면 **ReadWriteMany** 액세스 권한을 사용하여 영구 스토리지를 설정해야 합니다.
- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성해야 합니다.
- **LPAR(Logical partition)**에서 호스팅되고 **RHEL 8.4** 이상을 기반으로 하는 **RHEL KVM**(커널 가상 머신) 시스템을 프로비저닝합니다.

참고

프록시를 구성하는 경우에도 해당 사이트 목록을 검토하십시오.

13.2.2. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

-

OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.

- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.

중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

13.2.3. 사용자 프로비저닝 인프라를 포함한 클러스터의 시스템 요구사항

사용자 프로비저닝 인프라가 포함된 클러스터의 경우, 필요한 모든 시스템을 배포해야 합니다.

RHEL 8.4 이상을 기반으로 하는 하나 이상의 KVM 호스트 머신입니다. 각 RHEL KVM 호스트 머신에는 libvirt가 설치되어 실행되어야 합니다. 가상 머신은 각 RHEL KVM 호스트 머신에서 프로비저닝됩니다.

13.2.3.1. 필요한 시스템

최소 OpenShift Container Platform 클러스터에 다음과 같은 호스트가 필요합니다.

표 13.1. 최소 필수 호스트

호스트	설명
임시 부트스트랩 시스템 한 개	컨트롤 플레인 시스템 세 개에 OpenShift Container Platform 클러스터를 배포하기 위한 부트스트랩 시스템이 클러스터에 필요합니다. 클러스터를 설치한 후 부트스트랩 시스템을 제거할 수 있습니다.
컨트롤 플레인 시스템 세 개	컨트롤 플레인 시스템은 컨트롤 플레인을 구성하는 Kubernetes 및 OpenShift Container Platform 서비스를 실행합니다.

호스트	설명
두 개 이상의 컴퓨팅 시스템(작업자 시스템이라고도 함).	OpenShift Container Platform 사용자가 요청한 워크로드는 컴퓨팅 머신에서 실행됩니다.

중요

클러스터의 고가용성을 개선하려면 컨트롤 플레인 시스템을 두 개 이상의 물리적 시스템의 서로 다른 RHEL 인스턴스에 배포하십시오.

부트스트랩, 컨트롤 플레인 및 컴퓨팅 시스템은 운영 체제로 RHCOS (Red Hat Enterprise Linux CoreOS)를 사용해야 합니다.

Red Hat Enterprise Linux 기술 기능 및 제한을 참조하십시오.

13.2.3.2. 네트워크 연결 요구사항

OpenShift Container Platform 설치 프로그램은 Ignition 파일을 생성합니다. 이 파일은 모든 RHCOS(Red Hat Enterprise Linux CoreOS) 가상 머신에 필요합니다. OpenShift Container Platform의 자동화된 설치는 부트스트랩 머신에서 수행합니다. 각 노드에서 OpenShift Container Platform 설치를 시작하고 Kubernetes 클러스터를 시작한 다음 완료합니다. 이 부트스트랩 중에 가상 머신에 DHCP(Dynamic Host Configuration Protocol) 서버 또는 고정 IP 주소를 통해 기존 네트워크 연결이 있어야 합니다.

13.2.3.3. IBM Z 네트워크 연결 요구 사항

RHEL KVM 아래의 IBM Z에 설치하려면 다음이 필요합니다.

- OSA 또는 RoCE 네트워크 어댑터로 구성된 RHEL KVM 호스트.
- libvirt 또는 MacVTap에서 브릿지 네트워킹을 사용하여 네트워크를 게스트에 연결하도록 구성된 RHEL KVM 호스트.

가상 네트워크 연결 유형을 참조하십시오.

13.2.3.4. 호스트 머신 리소스 요구사항

환경의 RHEL KVM 호스트는 다음 요구사항을 충족해야 OpenShift Container Platform 환경에 계획하는 가상 머신을 호스팅해야 합니다. 가상화 시작하기를 참조하십시오.

다음 IBM 하드웨어에 OpenShift Container Platform 버전 4.9를 설치할 수 있습니다.

- IBM z15(모든 모델), IBM z14(모든 모델), IBM z13 및 IBM z13s
- LinuxONE, 모든 버전

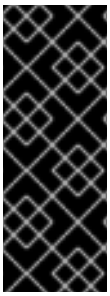
13.2.3.5. 최소 IBM Z 시스템 환경

하드웨어 요구 사항

- 각 클러스터에 대해 SMT2가 활성화된 6개의 IFL과 동일합니다.
- LoadBalancer 서비스에 연결하고 클러스터 외부 트래픽에 대한 데이터를 제공하는 네트워크 연결은 하나 이상 있습니다.

참고

전용 또는 공유 IFL을 사용하여 충분한 컴퓨팅 리소스를 할당할 수 있습니다. 리소스 공유는 IBM Z의 주요 강점 중 하나입니다. 그러나 각 하이퍼바이저 계층에서 용량을 올바르게 조정하고 모든 OpenShift Container Platform 클러스터에 충분한 리소스를 확인해야 합니다.



중요

클러스터의 전반적인 성능에 영향을 미칠 수 있으므로 OpenShift Container Platform 클러스터를 설정하는 데 사용되는 LPAR은 충분한 컴퓨팅 용량을 제공해야 합니다. 이 컨텍스트에서 하이퍼바이저 수준의 LPAR 가중치 관리, 권한 부여 및 CPU 공유는 중요한 역할을 합니다.

운영 체제 요구 사항

- libvirt에서 관리하는 KVM을 사용하여 RHEL 8.4 이상을 실행하는 하나의 LPAR

RHEL KVM 호스트에서 다음을 설정합니다.

- OpenShift Container Platform 컨트롤 플레인 시스템 용 게스트 가상 머신 세 개
- OpenShift Container Platform 컴퓨팅 머신 용 게스트 가상 머신 두 개
- 임시 OpenShift Container Platform 부트스트랩 시스템용 게스트 가상 머신 1개

13.2.3.6. 최소 리소스 요구사항

각 클러스터 가상 머신은 다음과 같은 최소 요구사항을 충족해야 합니다.

가상 머신	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS
부트스트랩	RHCOS	4	16GB	100GB	해당 없음
컨트롤 플레인	RHCOS	4	16GB	100GB	해당 없음
컴퓨팅	RHCOS	2	8GB	100GB	해당 없음

1. SMT-2가 활성화된 경우 하나의 물리적 코어(IFL)는 두 개의 논리 코어(스레드)를 제공합니다. 하이퍼바이저는 두 개 이상의 vCPU를 제공할 수 있습니다.

13.2.3.7. 권장되는 IBM Z 시스템 환경

하드웨어 요구 사항

- 각 클러스터에 대해 SMT2가 활성화된 6개의 IFL과 동등한 3개의 LPARS가 있습니다.
- LoadBalancer 서비스에 연결하고 클러스터 외부의 트래픽에 대한 데이터를 제공하는 두 개의 네트워크 연결

운영 체제 요구 사항

- 고가용성을 위해 libvirt에서 관리하는 KVM을 사용하여 RHEL 8.4 이상을 실행하는 두 개의 LPAR.

RHEL KVM 호스트에서 다음을 설정합니다.

- RHEL KVM 호스트 머신에 분산된 OpenShift Container Platform 컨트롤 플레인 시스템용 게스트 가상 머신 세 개
- RHEL KVM 호스트 머신에 분산된 OpenShift Container Platform 컴퓨팅 머신용 게스트 가상 머신 6개 이상
- 임시 OpenShift Container Platform 부트스트랩 시스템용 게스트 가상 머신 1개
- 오버 커밋된 환경에서 통합 구성 요소를 사용하려면 cpu_shares 를 사용하여 컨트롤 플레인의 우선 순위를 높입니다. 인프라 노드가 있는 경우 동일한 작업을 수행합니다. IBM 문서의 schedinfo 를 참조하십시오.

13.2.3.8. 권장되는 리소스 요구사항.

각 클러스터 가상 머신의 권장되는 요구 사항은 다음과 같습니다.

가상 머신	운영 체제	vCPU	가상 RAM	스토리지
부트스트랩	RHCOS	4	16GB	120GB
컨트롤 플레인	RHCOS	8	16GB	120GB
컴퓨팅	RHCOS	6	8GB	120GB

13.2.3.9. 인증서 서명 요청 관리

사용자가 프로비저닝하는 인프라를 사용하는 경우 자동 시스템 관리 기능으로 인해 클러스터의 액세스가 제한되므로 설치한 후 클러스터 인증서 서명 요청(CSR)을 승인하는 메커니즘을 제공해야 합니다. kube-controller-manager는 kubelet 클라이언트 CSR만 승인합니다. machine-approver는 올바른 시스템에서 발행한 요청인지 확인할 수 없기 때문에 kubelet 자격 증명을 사용하여 요청하는 서비스 인증서의 유효성을 보장할 수 없습니다. kubelet 서빙 인증서 요청의 유효성을 확인하고 요청을 승인하는 방법을 결정하여 구현해야 합니다.

추가 리소스

- IBM Z 및 LinuxONE 환경에 대한 권장 호스트 사례

13.2.3.10. 사용자 프로비저닝 인프라에 대한 네트워킹 요구사항

모든 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템이 부팅 중에 Ignition 구성 파일을 가져오려면 initramfs에 네트워킹을 구성해야 합니다.

초기 부팅 과정에서 시스템에 필요한 부팅 옵션을 제공하여 DHCP 서버를 통해 설정하거나 정적으로 설정하는 IP 주소 구성이 필요합니다. 네트워크 연결이 설정된 후 시스템은 HTTP 또는 HTTPS 서버에서 Ignition 구성 파일을 다운로드합니다. 그런 다음 Ignition 구성 파일을 사용하여 각 머신의 정확한 상태를 설정합니다. Machine Config Operator는 설치 후 새 인증서 또는 키 적용과 같은 머신에 대한 추가 변경을 완료합니다.

클러스터 시스템의 장기적인 관리를 위해 DHCP 서버를 사용하는 것이 좋습니다. DHCP 서버가 클러스터 시스템에 영구 IP 주소, DNS 서버 정보 및 호스트 이름을 제공하도록 구성되었는지 확인합니다.

참고

사용자 프로비저닝 인프라에 DHCP 서비스를 사용할 수 없는 경우 RHCOS 설치 시 노드에 IP 네트워킹 구성과 DNS 서버의 주소를 대신 제공할 수 있습니다. ISO 이미지에서 설치하는 경우 부팅 인수로 전달할 수 있습니다. 고정 IP 프로비저닝 *및 고급 네트워킹 옵션에 대한 자세한 내용은 **RHCOS** 설치 및 **OpenShift Container Platform** 부트스트랩 프로세스 시작* 섹션을 참조하십시오.

Kubernetes API 서버가 클러스터 시스템의 노드 이름을 확인할 수 있어야 합니다. API 서버와 작업자 노드가 서로 다른 영역에 있는 경우, API 서버가 노드 이름을 확인할 수 있도록 기본 DNS 검색 영역을 설정할 수 있습니다. 노드 개체와 모든 DNS 요청에서 항상 정규화된 도메인 이름으로 호스트를 가리키는 것도 지원되는 방법입니다

13.2.3.10.1. DHCP를 통해 클러스터 노드의 호스트 이름 설정

RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 호스트 이름은 NetworkManager를 통해 설정됩니다. 기본적으로 시스템은 DHCP를 통해 호스트 이름을 가져옵니다. DHCP에서 호스트 이름을 제공하지 않거나, 커널 인수를 통해 정적으로 설정하거나 다른 방법을 통해 설정하면 역방향 DNS 조회를 통해 얻을 수 있습니다. 역방향 DNS 조회는 노드에서 네트워크를 초기화한 후 수행되며 확인하는 데 시간이 걸릴 수 있습니다. 다른 시스템 서비스는 이 보다 먼저 시작하여 호스트 이름을 localhost 등으로 감지할 수 있습니다. DHCP를 사용하여 각 클러스터 노드의 호스트 이름을 제공하여 이 문제를 방지할 수 있습니다.

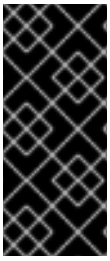
또한 DHCP를 통해 호스트 이름을 설정하면 DNS 분할 수평 구현 환경에서 수동으로 DNS 레코드 이

름 구성 오류를 무시할 수 있습니다.

13.2.3.10.2. 네트워크 연결 요구사항

OpenShift Container Platform 클러스터 구성 요소가 통신할 수 있도록 시스템 간 네트워크 연결을 구성해야 합니다. 각 시스템에서 클러스터에 있는 다른 모든 시스템의 호스트 이름을 확인할 수 있어야 합니다.

이 섹션에서는 필요한 포트에 대해 자세히 설명합니다.



중요

연결된 **OpenShift Container Platform** 환경에서 모든 노드는 플랫폼 컨테이너의 이미지를 가져오고 **Red Hat**에 원격 측정 데이터를 제공하기 위해 인터넷에 액세스할 수 있어야 합니다.

참고

libvirt 또는 **MacVTap**에서 브릿지 네트워킹을 사용하여 가상 머신에 네트워크를 연결하도록 **RHEL KVM** 호스트를 구성해야 합니다. 가상 머신은 **RHEL KVM** 호스트에 연결된 네트워크에 액세스할 수 있어야 합니다. **KVM** 내에서 네트워크 주소 변환(예: 네트워크 주소 변환)은 지원되지 않습니다.

표 13.2. 모든 시스템 간 통신에 사용되는 포트

프로토콜	포트	설명
ICMP	해당 없음	네트워크 연결성 테스트
TCP	**1936**	메트릭
	9000-**9999**	**9100**-**9101** 포트의 노드 내보내기 및 **9099** 포트의 Cluster Version Operator를 포함한 호스트 수준 서비스.
	10250-**10259**	Kubernetes에서 예약하는 기본 포트
	10256	openshift-sdn
UDP	**4789**	VXLAN 및 Geneve
	6081	VXLAN 및 Geneve

프로토콜	포트	설명
	9000-9999	**9100-9101** 포트의 노드 내보내기를 포함한 호스트 수준 서비스.
	500	IPsec IKE 패킷
	4500	IPsec NAT-T 패킷
TCP/UDP	**30000-32767**	Kubernetes 노드 포트
ESP	해당 없음	IPsec Encapsulating Security Payload (ESP)

표 13.3. 모든 시스템과 컨트롤 플레인 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**6443**	Kubernetes API

표 13.4. 컨트롤 플레인 머신 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**2379-2380**	etcd 서버 및 피어 포트

사용자 프로비저닝 인프라에 대한 NTP 구성

OpenShift Container Platform 클러스터는 기본적으로 공용 **NTP(Network Time Protocol)** 서버를 사용하도록 구성되어 있습니다. 로컬 엔터프라이즈 **NTP** 서버를 사용하거나 클러스터가 연결이 끊긴 네트워크에 배포되는 경우 특정 시간 서버를 사용하도록 클러스터를 구성할 수 있습니다. 자세한 내용은 ***chrony 타임 서비스 설정*** 문서를 참조하십시오.

DHCP 서버가 **NTP** 서버 정보를 제공하는 경우 **RHCOS(Red Hat Enterprise Linux CoreOS)** 시스템의 **chrony** 타임 서비스에서 정보를 읽고 **NTP** 서버와 클럭을 동기화할 수 있습니다.

추가 리소스

- chrony 타임 서비스 설정

13.2.3.11. 사용자 프로비저닝 DNS 요구사항

OpenShift Container Platform 배포의 경우 다음 구성 요소에 **DNS** 이름을 확인해야 합니다.

- **Kubernetes API**
- **OpenShift Container Platform** 애플리케이션 와일드카드
- 부트스트랩, 컨트롤 플레인 및 컴퓨팅 시스템

Kubernetes API, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템에 대한 역방향 **DNS** 확인이 필요합니다.

DNS A/AAAA 또는 **CNAME** 레코드는 이름 확인에 사용되며 **PTR** 레코드는 역방향 이름 확인에 사용됩니다. **RHCOS (Red Hat Enterprise Linux CoreOS)**는 **DHCP**에서 호스트 이름을 제공하지 않는 한 모든 노드의 호스트 이름을 설정할 때 역방향 레코드를 사용하기 때문에 역방향 레코드가 중요합니다. 또한 역방향 레코드는 **OpenShift Container Platform**이 작동하는 데 필요한 인증서 서명 요청 **(CSR)**을 생성하는 데 사용됩니다.

사용자가 프로비저닝한 **OpenShift Container Platform** 클러스터에 대해 다음 **DNS** 레코드가 필요하며 설치 전에 있어야 합니다. 각 레코드에서 **<cluster_name>**은 클러스터 이름이고 **<base_domain>**은 **install-config.yaml** 파일에서 지정하는 기반 도메인입니다. 전체 **DNS** 레코드는 **<component>.<cluster_name>.<base_domain>** 형식입니다.

표 **13.5.** 필수 **DNS** 레코드

구성 요소	레코드	설명
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API 로드 밸런서를 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.

구성 요소	레코드	설명
	api-int.<cluster_name>. <base_domain>.	내부적으로 API 로드 밸런서를 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.  **중요** API 서버는 Kubernetes에 기록된 호스트 이름으로 작업자 노드를 확인할 수 있어야 합니다. API 서버가 노드 이름을 확인할 수 없는 경우 프록시된 API 호출이 실패할 수 있으며 pod에서 로그를 검색할 수 없습니다.
라우트	***.apps.<cluster_name>. <base_domain>.**	애플리케이션 인그레스 로드 밸런서를 참조하는 와일드카드 DNS A/AAA 또는 CNAME 레코드입니다. 애플리케이션 인그레스 로드 밸런서는 Ingress 컨트롤러 Pod를 실행하는 머신을 대상으로 합니다. Ingress 컨트롤러 Pod는 기본적으로 컴퓨팅 머신에서 실행됩니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다. 예를 들어 **console-openshift-console.apps. <cluster_name>.<base_domain>**은 OpenShift Container Platform 콘솔의 와일드카드 경로로 사용됩니다.
부트스트랩 시스템	**bootstrap.<cluster_name>. <base_domain>.**	부트스트랩 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.
컨트롤 플레인 머신	**<master><n>. <cluster_name>. <base_domain>.**	컨트롤 플레인 노드의 각 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.
컴퓨팅 머신	**<worker><n>. <cluster_name>. <base_domain>.**	작업자 노드의 각 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.

참고

OpenShift Container Platform 4.4 이상에서는 DNS 구성에서 etcd 호스트 및 SRV 레코드를 지정할 필요가 없습니다.

작은 정보

dig 명령을 사용하여 이름과 역방향 이름을 확인할 수 있습니다. 자세한 검증 단계는 *사용자 프로비저닝 인프라의 DNS 확인* 섹션을 참조하십시오.

13.2.3.11.1. 사용자 프로비저닝 클러스터의 DNS 구성 예

이 섹션에서는 사용자 프로비저닝 인프라에 OpenShift Container Platform을 배포하기 위한 DNS 요구 사항을 충족하는 A 및 PTR 레코드 구성 샘플을 제공합니다. 샘플은 하나의 DNS 솔루션을 선택하기 위한 조언을 제공하기 위한 것이 아닙니다.

이 예제에서 클러스터 이름은 ocp4이고 기본 도메인은 example.com입니다.

사용자 프로비저닝 클러스터의 DNS A 레코드 구성 예

다음 BIND 영역 파일의 예제에서는 사용자가 프로비저닝한 클러스터의 이름 확인을 위한 샘플 A 레코드를 보여줍니다.

예 13.1. 샘플 DNS 영역 데이터베이스

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
master0.ocp4.example.com. IN A 192.168.1.97 5
master1.ocp4.example.com. IN A 192.168.1.98 6
master2.ocp4.example.com. IN A 192.168.1.99 7
```

```
;
worker0.ocp4.example.com. IN A 192.168.1.11 8
worker1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1

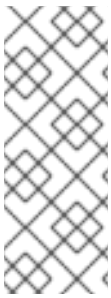
Kubernetes API의 이름 확인을 제공합니다. 레코드는 **API** 로드 밸런서의 **IP** 주소를 나타냅니다.

2

Kubernetes API의 이름 확인을 제공합니다. 레코드는 **API** 로드 밸런서의 **IP** 주소를 참조하며 내부 클러스터 통신에 사용됩니다.

3

와일드카드 경로의 이름 확인을 제공합니다. 레코드는 애플리케이션 인그레스 로드 밸런서의 **IP** 주소를 나타냅니다. 애플리케이션 인그레스 로드 밸런서는 **Ingress** 컨트롤러 **Pod**를 실행하는 머신을 대상으로 합니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.



참고

이 예제에서는 **Kubernetes API** 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 **API** 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

4

부트스트랩 시스템의 이름 확인을 제공합니다.

5 6 7

컨트롤 플레인 시스템의 이름 확인을 제공합니다.

8 9

컴퓨팅 시스템의 이름 확인을 제공합니다.

사용자 프로비저닝 클러스터의 **DNS PTR** 레코드 구성 예

다음 예제 BIND 영역 파일은 사용자 프로비저닝 클러스터의 역방향 이름 확인을 위한 샘플 PTR 레코드를 보여줍니다.

예 13.2. 역방향 레코드의 샘플 DNS 영역 데이터베이스

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR master0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR master1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR master2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com. 8
;
;EOF
```

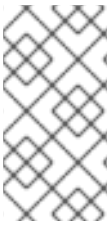
1 Kubernetes API의 역방향 DNS 확인을 제공합니다. PTR 레코드는 API 로드 밸런서의 레코드 이름을 참조합니다.

2 Kubernetes API의 역방향 DNS 확인을 제공합니다. PTR 레코드는 API 로드 밸런서의 레코드 이름을 참조하며 내부 클러스터 통신에 사용됩니다.

3 부트스트랩 시스템의 역방향 DNS 확인을 제공합니다.

4 5 6 컨트롤 플레인 시스템의 역방향 DNS 확인을 제공합니다.

7 8 컴퓨팅 시스템의 역방향 DNS 확인을 제공합니다.



참고

OpenShift Container Platform 애플리케이션 와일드카드에는 PTR 레코드가 필요하지 않습니다.

13.2.3.12. 사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구사항

OpenShift Container Platform을 설치하기 전에 API 및 애플리케이션 인그레스 로드 밸런싱 인프라를 프로비저닝해야 합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 API 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

참고

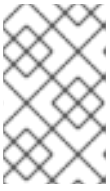
RHEL(Red Hat Enterprise Linux) 인스턴스를 사용하여 API 및 애플리케이션 인그레스 로드 밸런서를 배포하려면 RHEL 서브스크립션을 별도로 구매해야 합니다.

로드 밸런서 인프라는 다음 요구 사항을 충족해야 합니다.

1. API 로드 밸런서: 플랫폼과 상호 작용하고 플랫폼을 구성할 수 있도록 사용자(인간과 시스템 모두)에게 공통 끝점을 제공합니다. 다음 조건을 설정합니다.

   - Layer 4 로드 밸런싱 전용입니다. 이를 Raw TCP, SSL Passthrough 또는 SSL Bridge 모드라고 합니다. SSL Bridge 모드를 사용하는 경우, API 경로에 대해 SNI(Server Name Indication, 서버 이름 표시)를 활성화해야 합니다.

   - 스테이트리스 로드 밸런싱 알고리즘입니다. 옵션은 로드 밸런서 구현에 따라 달라집니다.

   참고

   API 로드 밸런서가 제대로 작동하기 위해 세션 지속성이 필요하지 않습니다.

   로드 밸런서의 전면과 후면 모두에서 다음 포트를 구성하십시오.

표 13.6. API 로드 밸런서

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
6443	부트스트랩 및 컨트롤 플레인. 부트스트랩 시스템이 클러스터 컨트롤 플레인을 초기화한 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다. API 서버 상태 검사 프로브에 대한 **/readyz** 끝점을 구성해야 합니다.	X	X	Kubernetes API 서버
22623	부트스트랩 및 컨트롤 플레인. 부트스트랩 시스템이 클러스터 컨트롤 플레인을 초기화한 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다.	X		시스템 구성 서버

참고

API 서버가 /readyz 엔드포인트를 해제하는 시점부터 풀에서 API 서버 인스턴스가 제거되는 시점까지 시간이 30초를 넘지 않도록 로드 밸런서를 구성해야 합니다. /readyz가 오류를 반환하거나 정상 상태가 된 후 정해진 시간 안에 끝점이 제거 또는 추가되어야 합니다. 5초 또는 10초의 프로빙 주기(두 번의 성공적인 요청은 정상 상태, 세 번의 요청은 비정상 상태)는 충분한 테스트를 거친 값입니다.

2. **애플리케이션 인그레스 로드 밸런서: 클러스터 외부에서 유입되는 애플리케이션 트래픽에 대한 인그래스 포인트를 제공합니다. 다음 조건을 설정합니다.**

- **Layer 4 로드 밸런싱 전용입니다. 이를 Raw TCP, SSL Passthrough 또는 SSL Bridge 모드라고 합니다. SSL Bridge 모드를 사용하는 경우 인그레스 경로에 대해 SNI(Server Name Indication, 서버 이름 표시)를 활성화해야 합니다.**

- **사용 가능한 옵션과 플랫폼에서 호스팅되는 애플리케이션 유형에 따라 연결 기반 또는 세션 기반 지속성이 권장됩니다.**

작은 정보

애플리케이션 인그레스 로드 밸런서에서 클라이언트의 실제 IP 주소를 확인할 수 있는 경우 소스 IP 기반 세션 지속성을 활성화하면 엔드 투 엔드 TLS 암호화를 사용하는 애플리케이션의 성능을 향상시킬 수 있습니다.

로드 밸런서의 전면과 후면 모두에서 다음 포트를 구성하십시오.

표 13.7. 애플리케이션 인그레스 로드 밸런서

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
443	기본적으로 인그레스 컨트롤러 pod, 컴퓨팅 또는 작업자를 실행하는 시스템입니다.	X	X	HTTPS 트래픽
80	기본적으로 인그레스 컨트롤러 pod, 컴퓨팅 또는 작업자를 실행하는 시스템입니다.	X	X	HTTP 트래픽
1936	기본적으로 Ingress 컨트롤러 Pod를 실행하는 작업자 노드입니다. 수신 상태 점검 프로브에 대해 **/healthz/ready** 끝점을 구성해야 합니다.	X	X	HTTP 트래픽

참고

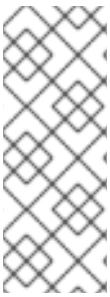
컴퓨팅 노드가 0인 3-노드 클러스터를 배포하는 경우 Ingress 컨트롤러 Pod는 컨트롤 플레인 노드에서 실행됩니다. 3-노드 클러스터 배포에서 HTTP 및 HTTPS 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다.

참고

인그레스 라우터에 대한 작업 구성이 OpenShift Container Platform 클러스터에 필요합니다. 컨트롤 플레인 초기화 후 인그레스 라우터를 설정해야 합니다.

13.2.3.12.1. 사용자 프로비저닝 클러스터의 로드 밸런서 구성 예

이 섹션에서는 사용자 프로비저닝 클러스터의 로드 밸런싱 요구 사항을 충족하는 API 및 애플리케이션 수신 로드 밸런서 구성 예를 제공합니다. 샘플은 HAProxy 로드 밸런서에 대한 /etc/haproxy/haproxy.cfg 구성입니다. 이 예제에서는 하나의 로드 밸런싱 솔루션을 선택하기 위한 조언을 제공하는 것을 목적으로 하지 않습니다.



참고

이 예제에서는 Kubernetes API 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 API 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

예 13.3. API 및 애플리케이션 인그레스 로드 밸런서 구성 샘플

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
frontend stats
  bind *:1936
  mode            http
  log             global
  maxconn 10
  stats enable
  stats hide-version
  stats refresh 30s
  stats show-node
  stats show-desc Stats for ocp4 cluster (1)
  stats auth admin:ocp4
  stats uri /stats
listen api-server-6443 (2)
  bind *:6443
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:6443 check inter 1s backup (3)
  server master0 master0.ocp4.example.com:6443 check inter 1s
  server master1 master1.ocp4.example.com:6443 check inter 1s
  server master2 master2.ocp4.example.com:6443 check inter 1s
listen machine-config-server-22623 (4)
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup (5)
  server master0 master0.ocp4.example.com:22623 check inter 1s
  server master1 master1.ocp4.example.com:22623 check inter 1s
  server master2 master2.ocp4.example.com:22623 check inter 1s
listen ingress-router-443 (6)
  bind *:443
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:443 check inter 1s
  server worker1 worker1.ocp4.example.com:443 check inter 1s
listen ingress-router-80 (7)
  bind *:80
  mode tcp
```

```
balance source
server worker0 worker0.ocp4.example.com:80 check inter 1s
server worker1 worker1.ocp4.example.com:80 check inter 1s
```

1 이 예에서 클러스터 이름은 **ocp4** 입니다.

2 포트 **6443**은 **Kubernetes API** 트래픽을 처리하고 컨트롤 플레인 시스템을 가리킵니다.

3 5 부트스트랩 항목은 **OpenShift Container Platform** 클러스터 설치 전에 있어야 하며 부트스트랩 프로세스가 완료된 후 제거해야 합니다.

4 포트 **22623**은 머신 구성 서버 트래픽을 처리하고 컨트롤 플레인 시스템을 가리킵니다.

6 포트 **443**은 **HTTPS** 트래픽을 처리하고 **Ingress** 컨트롤러 **Pod**를 실행하는 시스템을 가리킵니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

7 포트 **80**은 **HTTP** 트래픽을 처리하고 **Ingress** 컨트롤러 **Pod**를 실행하는 머신을 가리킵니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

참고

컴퓨팅 노드가 **0**인 **3-**노드 클러스터를 배포하는 경우 **Ingress** 컨트롤러 **Pod**는 컨트롤 플레인 노드에서 실행됩니다. **3-**노드 클러스터 배포에서 **HTTP** 및 **HTTPS** 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다.

작은 정보

HAProxy를 로드 밸런서로 사용하는 경우 **HAProxy** 노드에서 **netstat -nltupe**를 실행하여 **haproxy** 프로세스가 포트 **6443, 22623, 443** 및 **80**에서 수신 대기 중인지 확인할 수 있습니다.

참고

HAProxy를 로드 밸런서로 사용하고 SELinux가 enforcing으로 설정된 경우 HAProxy 서비스가 setsebool -P haproxy_connect_any=1을 실행하여 구성된 TCP 포트에 바인딩할 수 있는지 확인해야 합니다.

13.2.4. 사용자 프로비저닝 인프라 준비

사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치하기 전에 기본 인프라를 준비해야 합니다.

이 섹션에서는 OpenShift Container Platform 설치를 준비하기 위해 클러스터 인프라를 설정하는 데 필요한 높은 수준의 단계에 대해 자세히 설명합니다. 여기에는 클러스터 노드에 대한 IP 네트워킹 및 네트워크 연결 구성, 방화벽을 통해 필요한 포트 활성화, 필수 DNS 및 로드 밸런싱 인프라를 설정하는 작업이 포함됩니다.

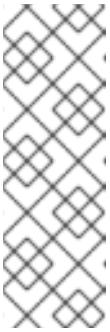
준비 후 클러스터 인프라는 *사용자 프로비저닝 인프라가 있는 클러스터의 요구 사항* 섹션에 설명된 요구 사항을 충족해야 합니다.

사전 요구 사항

- OpenShift Container Platform 4.x 테스트된 통합 페이지를 검토했습니다.
- *사용자 프로비저닝 인프라가 있는 클러스터의 요구 사항* 섹션에 자세히 설명된 인프라 요구 사항을 검토했습니다.

프로세스

1. DHCP를 사용하여 클러스터 노드에 IP 네트워킹 구성을 제공하는 경우 DHCP 서비스를 구성합니다.
   a. 노드의 영구 IP 주소를 DHCP 서버 구성에 추가합니다. 구성에서 관련 네트워크 인터페이스의 MAC 주소를 각 노드의 의도한 IP 주소와 일치시킵니다.
   b. DHCP를 사용하여 클러스터 시스템의 IP 주소 지정을 구성하는 경우 시스템은 DHCP를 통해 DNS 서버 정보도 가져옵니다. DHCP 서버 구성을 통해 클러스터 노드에서 사용하는 영구 DNS 서버 주소를 정의합니다.

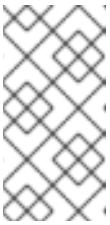


참고

DHCP 서비스를 사용하지 않는 경우 **RHCOS** 설치 시 **IP** 네트워킹 구성과 **DNS** 서버의 주소를 노드에 제공해야 합니다. **ISO** 이미지에서 설치하는 경우 부팅 인수로 전달할 수 있습니다. 고정 **IP** 프로비저닝 *및 고급 네트워킹 옵션에 대한 자세한 내용은* ***RHCOS*** *설치 및* ***OpenShift Container Platform*** *부트스트랩 프로세스 시작* 섹션을 참조하십시오.

c.

DHCP 서버 구성에 클러스터 노드의 호스트 이름을 정의합니다. 호스트 이름 고려 사항에 대한 자세한 내용은 ***DHCP****를 통해 클러스터 노드 호스트 이름 설정* 섹션을 참조하십시오.



참고

DHCP 서비스를 사용하지 않는 경우 클러스터 노드는 역방향 **DNS** 조회를 통해 호스트 이름을 가져옵니다.

2.

Red Hat Enterprise Linux CoreOS (RHCOS)의 빠른 설치 또는 **RHCOS(Red Hat Enterprise Linux CoreOS)**의 전체 설치를 수행하려면 선택합니다. 전체 설치의 경우 **Ignition** 파일을 제공하고 클러스터 노드에 이미지를 설치하기 위해 **HTTP** 또는 **HTTPS** 서버를 설정해야 합니다. 빠른 설치의 경우 **HTTP** 또는 **HTTPS** 서버가 필요하지 않지만 **DHCP** 서버가 필요합니다. **"빠른 설치: RHCOS (Red Hat Enterprise Linux CoreOS) 머신 생성 섹션" 및 "전체 설치: Red Hat Enterprise Linux CoreOS (RHCOS) 머신 생성" 섹션을 참조하십시오.**

3.

네트워크 인프라가 클러스터 구성 요소 간 필수 네트워크 연결을 제공하는지 확인합니다. 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 인프라 섹션의 네트워킹 요구 사항* 섹션을 참조하십시오.

4.

OpenShift Container Platform 클러스터 구성 요소가 통신하는 데 필요한 포트를 활성화하도록 방화벽을 구성합니다. 필요한 포트에 대한 자세한 내용은 *사용자 프로비저닝 인프라 섹션의 네트워킹 요구 사항* 섹션을 참조하십시오.

5.

클러스터에 필요한 **DNS** 인프라를 설정합니다.

a.

Kubernetes API, 애플리케이션 와일드카드, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템의 DNS 이름 확인을 구성합니다.

b.

Kubernetes API, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템에 대한 역방향 DNS 확인을 구성합니다.

OpenShift Container Platform DNS 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 DNS 요구 사항* 섹션을 참조하십시오.

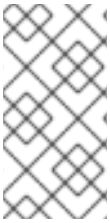
6. **DNS** 구성을 확인합니다.

   a. 설치 노드에서 **Kubernetes API**의 레코드 이름, 와일드카드 경로 및 클러스터 노드에 대해 **DNS** 조회를 실행합니다. 응답의 **IP** 주소가 올바른 구성 요소에 해당하는지 확인합니다.

   b. 설치 노드에서 로드 밸런서 및 클러스터 노드의 **IP** 주소에 대해 역방향 **DNS** 조회를 실행합니다. 응답의 레코드 이름이 올바른 구성 요소에 해당하는지 확인합니다.

   자세한 **DNS** 검증 단계는 *사용자 프로비저닝 인프라에 대한 DNS 확인* 섹션을 참조하십시오.

7. 필요한 **API** 및 애플리케이션 수신 로드 밸런싱 인프라를 프로비저닝합니다. 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구 사항* 섹션을 참조하십시오.

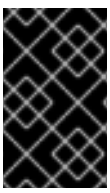


참고

일부 로드 밸런싱 솔루션에는 로드 밸런싱을 초기화하기 전에 클러스터 노드의 **DNS** 이름을 확인해야 합니다.

13.2.5. 사용자 프로비저닝 인프라에 대한 DNS 확인 검증

사용자 프로비저닝 인프라에 **OpenShift Container Platform**을 설치하기 전에 **DNS** 구성을 확인할 수 있습니다.



중요

클러스터를 설치하기 전에 이 섹션에 설명된 검증 단계를 성공해야 합니다.

사전 요구 사항

- 사용자 프로비저닝 인프라에 필요한 **DNS** 레코드를 구성했습니다.

프로세스

1. 설치 노드에서 Kubernetes API의 레코드 이름, 와일드카드 경로 및 클러스터 노드에 대해 DNS 조회를 실행합니다. 응답에 포함된 IP 주소가 올바른 구성 요소에 해당하는지 확인합니다.

   a. Kubernetes API 레코드 이름을 조회합니다. 결과가 API 로드 밸런서의 IP 주소를 가리키는지 확인합니다.

   ```
   $ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
   ```

   1 <nameserver_ip>를 네임서버의 IP 주소로, <cluster_name>을 클러스터 이름으로, <base_domain>을 기본 도메인 이름으로 바꿉니다.

   출력 예

   ```
   api.ocp4.example.com.  0 IN A 192.168.1.5
   ```

   b. Kubernetes 내부 API 레코드 이름을 조회합니다. 결과가 API 로드 밸런서의 IP 주소를 가리키는지 확인합니다.

   ```
   $ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
   ```

   출력 예

   ```
   api-int.ocp4.example.com.  0 IN A 192.168.1.5
   ```

   c. 예제 *.apps.<cluster_name>.<base_domain>을 테스트합니다. DNS 와일드카드를 조회합니다. 모든 애플리케이션 와일드카드 조회는 애플리케이션 인그레스 로드 밸런서의 IP 주소로 확인되어야 합니다.

```
$ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.
<base_domain>
```

출력 예

```
random.apps.ocp4.example.com.  0 IN A 192.168.1.5
```

참고

예제 출력에서는 **Kubernetes API** 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 **API** 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

random 항목을 다른 와일드카드 값으로 교체할 수 있습니다. 예를 들어 **OpenShift Container Platform** 콘솔의 경로를 쿼리할 수 있습니다.

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

출력 예

```
console-openshift-console.apps.ocp4.example.com. 0 IN A 192.168.1.5
```

d. 부트스트랩 **DNS** 레코드 이름에 대해 조회를 실행합니다. 결과가 부트스트랩 노드의 **IP** 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.
<base_domain>
```

출력 예

```
bootstrap.ocp4.example.com.  0 IN A 192.168.1.96
```

e. 이 방법을 사용하여 컨트롤 플레인 및 컴퓨팅 노드의 DNS 레코드 이름에 대해 조회를 수행합니다. 결과가 각 노드의 IP 주소에 해당하는지 확인합니다.

2. 설치 노드에서 로드 밸런서 및 클러스터 노드의 IP 주소에 대해 역방향 DNS 조회를 실행합니다. 응답에 포함된 레코드 이름이 올바른 구성 요소에 해당하는지 확인합니다.

a. API 로드 밸런서의 IP 주소에 대해 역방향 조회를 수행합니다. 응답에 Kubernetes API 및 Kubernetes 내부 API의 레코드 이름이 포함되어 있는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

출력 예

```
5.1.168.192.in-addr.arpa. 0 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 0 IN PTR api.ocp4.example.com. 2
```

1 Kubernetes 내부 API의 레코드 이름을 제공합니다.

2 Kubernetes API의 레코드 이름을 제공합니다.

참고

OpenShift Container Platform 애플리케이션 와일드카드에는 PTR 레코드가 필요하지 않습니다. 애플리케이션 인그레스 로드 밸런서의 IP 주소에 대한 역방향 DNS 확인에는 유효성 검사 단계가 필요하지 않습니다.

b.

부트스트랩 노드의 IP 주소에 대해 역방향 조회를 수행합니다. 결과가 부트스트랩 노드의 DNS 레코드 이름을 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

출력 예

```
96.1.168.192.in-addr.arpa. 0 IN PTR bootstrap.ocp4.example.com.
```

c.

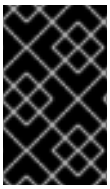
이 방법을 사용하여 컨트롤 플레인 및 컴퓨팅 노드의 IP 주소에 대해 역방향 조회를 수행합니다. 결과가 각 노드의 DNS 레코드 이름과 일치하는지 확인합니다.

13.2.6. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 core 사용자의 ~/.ssh/authorized_keys 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 core로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. ./openshift-install gather 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.



중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

프로세스

1.

로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1

새 SSH 키의 경로 및 파일 이름(예: ~/.ssh/id_ed25519)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 '~/.ssh 디렉터리에 있는지 확인하십시오.

참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 ed25519 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 rsa 또는 ecdsa 알고리즘을 사용하는 키를 생성합니다.

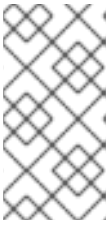
2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 ~/.ssh/id_ed25519.pub 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 ./openshift-install gather 명령을 사용하려는 경우 필요합니다.



참고

일부 배포에서는 ~/.ssh/id_rsa 및 ~/.ssh/id_dsa와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. ssh-agent 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. ssh-agent에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: ~/.ssh/id_ed25519).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

13.2.7. 설치 프로그램 받기

OpenShift Container Platform을 설치하기 전에 프로비저닝 머신에 설치 파일을 다운로드하십시오.

사전 요구 사항

-

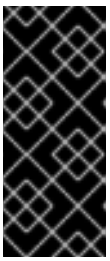
Linux를 실행하는 머신(예: 로컬 디스크 공간이 500MB인 Red Hat Enterprise Linux 8)이 있습니다.

절차

1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.

2. 인프라 공급자를 선택합니다.

3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.



중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.

중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 OpenShift Container Platform 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

13.2.8. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift CLI(oc)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 oc를 설치할 수 있습니다.

중요

이전 버전의 oc를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 oc를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 Linux에서 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

절차

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.
2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. OpenShift v4.9 Linux Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. oc 바이너리를 PATH에 있는 디렉터리에 배치합니다.

   PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치

다음 절차에 따라 Windows에 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

절차

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.

2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.

3. OpenShift v4.9 Windows Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.

4. ZIP 프로그램으로 아카이브의 압축을 풉니다.

5. oc 바이너리를 PATH에 있는 디렉터리로 이동합니다.

   PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

   ```
   C:\> path
   ```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치

다음 절차에 따라 macOS에서 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

절차

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.

2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.

3. OpenShift v4.9 MacOSX Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.

4. 아카이브의 압축을 해제하고 압축을 풉니다.

5. **oc** 바이너리 **PATH**의 디렉터리로 이동합니다.

   PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

13.2.9. 수동으로 설치 구성 파일 만들기

OpenShift Container Platform을 사용자가 프로비저닝한 설치의 경우 설치 구성 파일을 수동으로 생성합니다.

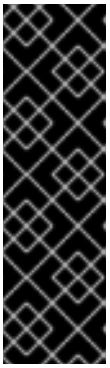
사전 요구 사항

- 로컬 시스템에 설치 프로그램에 제공할 **SSH** 공개 키가 있습니다. 키는 디버깅 및 재해 복구를 위해 클러스터 노드에 대한 **SSH** 인증에 사용됩니다.
- **OpenShift Container Platform** 설치 프로그램과 클러스터의 풀 시크릿이 있습니다.

절차

1. 필요한 설치 자산을 저장할 설치 디렉터리를 만듭니다.

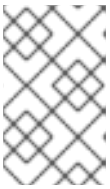
```
$ mkdir <installation_directory>
```



중요

디렉터리를 만들어야 합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

2. 샘플 install-config.yaml 파일 템플릿을 사용자 지정하여 <installation_directory>에 저장합니다.



참고

이 설정 파일의 이름을 install-config.yaml로 지정해야 합니다.

참고

일부 플랫폼 유형의 경우 대체로 ./openshift-install create install-config --dir <installation_directory> 를 실행하여 install-config.yaml 파일을 생성할 수 있습니다. 프롬프트에서 클러스터 구성에 대한 세부 정보를 제공할 수 있습니다.

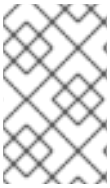
3. 여러 클러스터를 설치하는 데 사용할 수 있도록 install-config.yaml 파일을 백업합니다.

중요

install-config.yaml 파일은 설치 과정의 다음 단계에서 사용됩니다. 이 시점에서 이를 백업해야 합니다.

13.2.9.1. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 환경에 대한 세부 정보를 설명하는 사용자 지정 install-config.yaml 설치 구성 파일을 제공합니다.



참고

설치한 후에는 install-config.yaml 파일에서 이러한 매개변수를 수정할 수 없습니다.

중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

13.2.9.1.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 13.8. 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	소문자, 하이픈(**-**), 마침표(**.**)로 구성되는 문자열(예: **dev**)입니다.

매개변수	설명	값
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

13.2.9.1.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 13.9. 네트워크 매개변수

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체 **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.

매개변수	설명	값
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.
networking.clusterNetwork	Pod의 IP 주소 블록입니다. 기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된 **cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다. 여러 IP 커널 인수를 지정하는 경우 **machineNetwork.cidr** 값은 기본 네트워크의 CIDR이어야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16

매개변수	설명	값
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16** **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

13.2.9.1.3. 선택적 구성 매개변수

선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 13.10. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **s390x** (기본값)입니다.	문자열

매개변수	설명	값
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은 **3**입니다.
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **s390x** (기본값)입니다.	문자열

매개변수	설명	값
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 기본값은 **External**입니다. 이 필드를 **Internal** 로 설정하는 것은 클라우드 이외의 플랫폼에서는 지원되지 않습니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

13.2.9.2. IBM Z의 샘플 install-config.yaml 파일

install-config.yaml 파일을 사용자 지정하여 **OpenShift Container Platform** 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.

```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 0 (4)
  architecture : s390x
controlPlane: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3 (7)
  architecture : s390x
metadata:
  name: test (8)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 (9)
    hostPrefix: 23 (10)
  networkType: OpenShiftSDN
  serviceNetwork: (11)
  - 172.30.0.0/16
platform:
  none: {} (12)
fips: false (13)
pullSecret: '{"auths": ...}' (14)
sshKey: 'ssh-ed25519 AAAA...' (15)
```

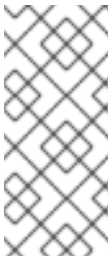
(1)

2 5

controlPlane 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

3 6

동시 멀티스레딩(**SMT**) 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 **SMT**가 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. **SMT**를 비활성화하는 경우 모든 클러스터 머신에서 이를 비활성화해야 합니다. 여기에는 컨트롤 플레인과 컴퓨팅 머신이 모두 포함됩니다.



참고

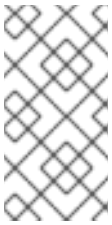
SMT(동시 멀티 스레딩)는 기본적으로 활성화되어 있습니다. **OpenShift Container Platform** 노드에서 **SMT**를 사용할 수 없는 경우 **hyperthreading** 매개변수가 적용되지 않습니다.

중요

OpenShift Container Platform 노드 또는 **install-config.yaml** 파일에서 **hyperthreading**을 비활성화하는 경우 용량 계획에서 머신 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.

4

사용자 프로비저닝 인프라에 **OpenShift Container Platform**을 설치할 때 이 값을 **0**으로 설정해야 합니다. 설치 프로그램에서 제공하는 설치에서 매개 변수는 클러스터가 생성 및 관리하는 컴퓨팅 머신 수를 제어합니다. 사용자 프로비저닝 설치에서는 클러스터 설치를 완료하기 전에 컴퓨팅 시스템을 수동으로 배포해야 합니다.



참고

3-노드 클러스터를 설치하는 경우 **RHCOS(Red Hat Enterprise Linux CoreOS)** 시스템을 설치할 때 컴퓨팅 머신을 배포하지 마십시오.

7

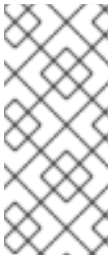
클러스터에 추가하는 컨트롤 플레인 시스템의 수입니다. 클러스터에서 이 값을 클러스터의 **etcd** 끝점 수로 사용하므로 이 값은 배포하는 컨트롤 플레인 시스템의 수와 일치해야 합니다.

8

DNS 레코드에 지정한 클러스터 이름입니다.

9

Pod IP 주소가 할당되는 IP 주소 블록입니다. 이 블록은 기존 물리적 네트워크와 중복되지 않아야합니다. 이러한 IP 주소는 Pod 네트워크에 사용됩니다. 외부 네트워크에서 Pod에 액세스해야 하는 경우, 트래픽을 관리하도록 로드 밸런서와 라우터를 설정해야 합니다.



참고

클래스 E CIDR 범위는 향후 사용을 위해 예약되어 있습니다. 클래스 E CIDR 범위를 사용하려면 네트워킹 환경에서 클래스 E CIDR 범위 내에서 IP 주소를 수락하는지 확인해야 합니다.

10

개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 hostPrefix를 23으로 설정하면 지정된 cidr 이외 /23 서브넷이 각 노드에 할당되어 510(2^(32 - 23) - 2) Pod IP 주소가 허용됩니다. 외부 네트워크에서 노드에 액세스해야 하는 경우 트래픽을 관리하도록 로드 밸런서와 라우터를 구성합니다.

11

서비스 IP 주소에 사용할 IP 주소 풀입니다. IP 주소 풀은 하나만 입력할 수 있습니다. 이 블록은 기존 물리적 네트워크와 중복되지 않아야합니다. 외부 네트워크에서 서비스에 액세스해야 하는 경우, 트래픽을 관리하도록 로드 밸런서와 라우터를 구성합니다.

12

플랫폼을 none으로 설정해야 합니다. IBM Z 인프라에 대한 추가 플랫폼 구성 변수는 지정할 수 없습니다.

중요

플랫폼 유형으로 설치된 클러스터는 Machine API로 컴퓨팅 머신 관리와 같은 일부 기능을 사용할 수 없습니다. 이 제한은 클러스터에 연결된 컴퓨팅 시스템이 일반적으로 기능을 지원하는 플랫폼에 설치된 경우에도 적용됩니다. 이 매개변수는 설치 후 변경할 수 없습니다.

13

FIPS 모드 활성화 또는 비활성화 여부입니다. 기본적으로 FIPS 모드는 비활성화됩니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화

모듈을 대신 사용합니다.

중요

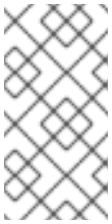
FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 **OpenShift Container Platform** 배포에서만 지원됩니다.

14

Red Hat OpenShift Cluster Manager의 풀 시크릿. 이 풀 시크릿을 사용하면 **OpenShift Container Platform** 구성 요소에 대한 컨테이너 이미지를 제공하는 **Quay.io**를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

15

RHCOS(Red Hat Enterprise Linux CoreOS)의 **core** 사용자에 대한 **SSH** 공용 키입니다.



참고

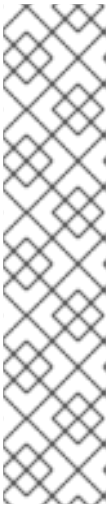
설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 **OpenShift Container Platform** 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 **SSH** 키를 지정합니다.

13.2.9.3. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 **HTTP** 또는 **HTTPS** 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 **OpenShift Container Platform** 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 **API**에 대한 호출을 포함하여 모든 클러스터 발신(**Egress**) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.



참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 Red Hat OpenStack Platform (RHOSP)에 설치하는 경우 Proxy 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. 클러스터 외부에서 **HTTP** 연결을 구축하는 데 사용할 프록시 **URL**입니다. **URL** 스키마는 **http**여야 합니다.

2. 클러스터 외부에서 **HTTPS** 연결을 구축하는 데 사용할 프록시 **URL**입니다.

3. 대상 도메인 이름, **IP** 주소 또는 프록시에서 제외할 기타 네트워크 **CIDR**로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. ***를 사용하여 모든 대상에 대해 프록시를 바이패스합니다.

4.

참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 **OpenShift Container Platform**을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 **nil**이 됩니다.

참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

13.2.9.4. 3개의 노드 클러스터 구성

필요한 경우 세 개의 컨트롤 플레인 시스템으로 구성된 베어 메탈 클러스터에 제로 컴퓨팅 머신을 배포할 수 있습니다. 이를 통해 클러스터 관리자와 개발자들이 테스트, 개발, 프로덕션에 사용할 수 있는 소형화되고 리소스 효율이 높은 클러스터를 제공합니다.

3-노드 **OpenShift Container Platform** 환경에서 세 개의 컨트롤 플레인 머신을 예약할 수 있습니다. 즉, 애플리케이션 워크로드가 해당 플랫폼에서 실행되도록 예약됩니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.

절차

- **install-config.yaml** 파일에서 다음 **compute** 스탠자에 표시된 대로 컴퓨팅 복제본 수가 **0**으로 설정되어 있는지 확인합니다.

```
compute:
- name: worker
  platform: {}
```

```
replicas: 0
```

참고

배포 중인 컴퓨팅 머신 수에 관계없이 사용자 프로비저닝 인프라에 **OpenShift Container Platform**을 설치할 때 컴퓨팅 머신의 **replicas** 매개변수 값을 **0**으로 설정해야 합니다. 설치 프로그램에서 제공하는 설치에서 매개 변수는 클러스터가 생성 및 관리하는 컴퓨팅 머신 수를 제어합니다. 이 설정은 컴퓨팅 시스템이 수동으로 배포되는 사용자 프로비저닝 설치에는 적용되지 않습니다.

참고

컨트롤 플레인 노드의 기본 리소스는 **vCPU 6**개와 **21GB**입니다. **3**개의 컨트롤 플레인 노드의 경우 메모리 **+ vCPU**는 최소 **5-**노드 클러스터와 동등합니다. **SMT2**가 활성화된 **IFL 3**개와 함께 각각 **120GB** 디스크에 설치된 **3**개의 노드를 백업해야 합니다. 테스트된 최소 설정은 각 컨트롤 플레인 노드에 대해 **120GB** 디스크에서 **3**개의 **vCPU**와 **10GB**입니다.

3-노드 클러스터 설치의 경우 다음 단계를 따르십시오.

- 컴퓨팅 노드가 **0**인 **3-**노드 클러스터를 배포하는 경우 **Ingress** 컨트롤러 **Pod**는 컨트롤 플레인 노드에서 실행됩니다. **3-**노드 클러스터 배포에서 **HTTP** 및 **HTTPS** 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다. 자세한 내용은 *사용자 프로비저닝 인프라 섹션에 대한 로드 밸런싱 요구 사항* 섹션을 참조하십시오.

- 다음 절차에서 **Kubernetes** 매니페스트 파일을 생성할 때 **<installation_directory>/manifests/cluster-scheduler-02-config.yml** 파일의 **mastersSchedulable** 매개변수가 **true**로 설정되어 있는지 확인합니다. 이렇게 하면 애플리케이션 워크로드를 컨트롤 플레인 노드에서 실행할 수 있습니다.

- **RHCOS(Red Hat Enterprise Linux CoreOS)** 시스템을 생성할 때 컴퓨팅 노드를 배포하지 마십시오.

13.2.10. CNO(Cluster Network Operator) 구성

클러스터 네트워크의 구성은 **CNO(Cluster Network Operator)** 구성의 일부로 지정되며 **cluster**라는 이름의 **CR**(사용자 정의 리소스) 오브젝트에 저장됩니다. **CR**은 **operator.openshift.io API** 그룹에서 **Network API**의 필드를 지정합니다.

CNO 구성은 Network.config.openshift.io API 그룹의 Network API에서 클러스터 설치 중에 다음 필드를 상속하며 이러한 필드는 변경할 수 없습니다.

clusterNetwork

Pod IP 주소가 할당되는 IP 주소 풀입니다.

serviceNetwork

서비스를 위한 IP 주소 풀입니다.

defaultNetwork.type

OpenShift SDN 또는 OVN-Kubernetes와 같은 클러스터 네트워크 공급자입니다.

cluster라는 CNO 오브젝트에서 defaultNetwork 오브젝트의 필드를 설정하여 클러스터의 클러스터 네트워크 공급자 구성을 지정할 수 있습니다.

13.2.10.1. CNO(Cluster Network Operator) 구성 오브젝트

CNO(Cluster Network Operator)의 필드는 다음 표에 설명되어 있습니다.

표 13.11. CNO(Cluster Network Operator) 구성 오브젝트

필드	유형	설명
metadata.name	**string**	CNO 개체 이름입니다. 이 이름은 항상 **cluster**입니다.
spec.clusterNetwork	**array**	Pod IP 주소가 할당되는 IP 주소 블록과 클러스터의 각 개별 노드에 할당된 서브넷 접두사 길이를 지정하는 목록입니다. 예를 들면 다음과 같습니다. spec: clusterNetwork: - cidr: 10.128.0.0/19 hostPrefix: 23 - cidr: 10.128.32.0/19 hostPrefix: 23 매니페스트를 생성하기 전에 **install-config.yaml** 파일에서만 이 필드를 사용자 지정할 수 있습니다. 값은 매니페스트 파일에서 읽기 전용입니다.

필드	유형	설명
spec.serviceNetwork	**array**	서비스의 IP 주소 블록입니다. OpenShift SDN 및 OVN-Kubernetes CNI(Container Network Interface) 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다. 예를 들면 다음과 같습니다. spec: serviceNetwork: - 172.30.0.0/14 매니페스트를 생성하기 전에 **install-config.yaml** 파일에서만 이 필드를 사용자 지정할 수 있습니다. 값은 매니페스트 파일에서 읽기 전용입니다.
spec.defaultNetwork	**object**	클러스터 네트워크의 CNI(Container Network Interface) 클러스터 네트워크 공급자를 구성합니다.
spec.kubeProxyConfig	**object**	이 개체의 필드는 kube-proxy 구성을 지정합니다. OVN-Kubernetes 클러스터 네트워크 공급자를 사용하는 경우 kube-proxy 구성이 적용되지 않습니다.

defaultNetwork 오브젝트 구성

defaultNetwork 오브젝트의 값은 다음 표에 정의되어 있습니다.

표 13.12. defaultNetwork 오브젝트

필드	유형	설명
type	**string**	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, 클러스터 네트워크 공급자가 설치 중에 선택됩니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다. **참고** OpenShift Container Platform은 기본적으로 OpenShift SDN CNI(Container Network Interface) 클러스터 네트워크 공급자를 사용합니다.
openshiftSDNConfig	**object**	이 오브젝트는 OpenShift SDN 클러스터 네트워크 공급자에만 유효합니다.
ovnKubernetesConfig	**object**	이 오브젝트는 OVN-Kubernetes 클러스터 네트워크 공급자에만 유효합니다.

OpenShift SDN CNI 네트워크 공급자에 대한 구성

다음 표에서는 **OpenShift SDN Container Network Interface (CNI)** 클러스터 네트워크 공급자의 구성 필드를 설명합니다.

표 13.13. openshiftSDNConfig 오브젝트

필드	유형	설명
mode	**string**	OpenShift SDN의 네트워크 격리 모드를 구성합니다. 기본값은 **NetworkPolicy**입니다. **Multitenant** 및 **Subnet** 값은 OpenShift Container Platform 3.x 버전과의 역호환할 수 있지만 권장되지는 않습니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
mtu	**integer**	VXLAN 오버레이 네트워크의 최대 전송 단위(MTU)입니다. 이는 기본 네트워크 인터페이스의 MTU를 기준으로 자동 탐지됩니다. 일반적으로 감지된 MTU를 재정의할 필요는 없습니다. 자동 감지 값이 예상 밖인 경우 노드의 기본 네트워크 인터페이스의 MTU가 올바른지 확인합니다. 이 옵션을 사용하여 노드의 기본 네트워크 인터페이스의 MTU 값을 변경할 수 없습니다. 클러스터에 다른 노드에 대한 다른 MTU 값이 필요한 경우 이 값을 클러스터의 가장 낮은 MTU 값보다 **50** 미만으로 설정해야 합니다. 예를 들어 클러스터의 일부 노드에는 **9001**의 MTU가 있고 일부에는 **1500**의 MTU가 있는 경우 이 값을 **1450**으로 설정해야 합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
vxlanPort	**integer**	모든 VXLAN 패킷에 사용할 포트입니다. 기본값은 **4789**입니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다. 기존 노드가 다른 VXLAN 네트워크에 속하는 가상 환경에서 실행 중인 경우에는 기본값을 변경해야 할 수도 있습니다. 예를 들어 VMware NSX-T 위에서 OpenShift SDN 오버레이를 실행할 때 두 SDN이 동일한 기본 VXLAN 포트 번호를 사용하므로 VXLAN의 대체 포트를 선택해야 합니다. AWS(Amazon Web Services)에서는 포트 **9000**과 포트 **9999** 사이에서 VXLAN의 대체 포트를 선택할 수 있습니다.

OpenShift SDN 구성 예

```
defaultNetwork:
  type: OpenShiftSDN
  openshiftSDNConfig:
    mode: NetworkPolicy
    mtu: 1450
    vxlanPort: 4789
```

OVN-Kubernetes CNI 클러스터 네트워크 공급자에 대한 구성

다음 표에서는 OVN-Kubernetes CNI 클러스터 네트워크 공급자의 구성 필드를 설명합니다.

표 13.14. ovnKubernetesConfig object

필드	유형	설명
mtu	**integer**	Geneve(Generic Network Virtualization Encapsulation) 오버레이 네트워크의 MTU(최대 전송 단위)입니다. 이는 기본 네트워크 인터페이스의 MTU를 기준으로 자동 탐지됩니다. 일반적으로 감지된 MTU를 재정의할 필요는 없습니다. 자동 감지 값이 예상 밖인 경우 노드의 기본 네트워크 인터페이스의 MTU가 올바른지 확인합니다. 이 옵션을 사용하여 노드의 기본 네트워크 인터페이스의 MTU 값을 변경할 수 없습니다. 클러스터에 다른 노드에 대한 다른 MTU 값이 필요한 경우, 이 값을 클러스터의 가장 낮은 MTU 값보다 **100** 미만으로 설정해야 합니다. 예를 들어, 클러스터의 일부 노드에 **9001**의 MTU가 있고 일부에는 **1500**의 MTU가 있는 경우 이 값을**1400**으로 설정해야 합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
genevePort	**integer**	모든 Geneve 패킷에 사용할 포트입니다. 기본값은 **6081**입니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
policyAuditConfig	**object**	네트워크 정책 감사 로깅을 사용자 정의할 구성 오브젝트를 지정합니다. 설정되지 않으면 기본값 감사 로그 설정이 사용됩니다.

표 13.15. policyAuditConfig object

필드	유형	설명
rateLimit	integer	노드당 1초마다 생성할 최대 메시지 수입니다. 기본값은 초당 **20** 개의 메시지입니다.
maxFileSize	integer	감사 로그의 최대 크기(바이트)입니다. 기본값은 **50000000 또는 50** MB입니다.

필드	유형	설명
대상	string	다음 추가 감사 로그 대상 중 하나입니다. **libc** 호스트에서 journald 프로세스의 libc **syslog()** 함수입니다. **udp:<host>:<port>** syslog 서버입니다. **<host>:<port>**를 syslog 서버의 호스트 및 포트로 바꿉니다. **unix:<file>** **<file>**로 지정된 Unix Domain Socket 파일입니다. **null** 감사 로그를 추가 대상으로 보내지 마십시오.
syslogFacility	string	RFC5424에 정의된 **kern**과 같은 syslog 기능입니다. 기본값은 **local0**입니다.

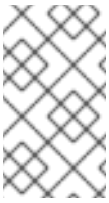
OVN-Kubernetes 구성 예

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
```

kubeProxyConfig 오브젝트 구성

kubeProxyConfig 오브젝트의 값은 다음 표에 정의되어 있습니다.

표 13.16. kubeProxyConfig object

필드	유형	설명
iptablesSyncPeriod	**string**	**iptables** 규칙의 새로 고침 간격입니다. 기본값은 **30s**입니다. 유효 접미사로 **s**, **m**, **h**가 있으며, 자세한 설명은 Go **time** 패키지 문서를 참조하십시오.  **참고** OpenShift Container Platform 4.3 이상에서는 성능이 개선되어 더 이상 **iptablesSyncPeriod** 매개변수를 조정할 필요가 없습니다.

필드	유형	설명
proxyArguments.iptables-min-sync-period	**array**	**iptables** 규칙을 새로 고치기 전 최소 기간입니다. 이 필드를 통해 새로 고침 간격이 너무 짧지 않도록 조정할 수 있습니다. 유효 접미사로 **s**, **m**, **h**가 있으며, 자세한 설명은 Go **time** 패키지를 참조하십시오. 기본값은 다음과 같습니다. kubeProxyConfig: proxyArguments: iptables-min-sync-period: - 0s

13.2.11. Kubernetes 매니페스트 및 Ignition 설정 파일 생성

일부 클러스터 정의 파일을 수정하고 클러스터 시스템을 수동으로 시작해야 하므로 클러스터가 시스템을 구성하는 데 필요한 **Kubernetes** 매니페스트 및 **Ignition** 구성 파일을 사용자가 생성해야 합니다.

설치 구성 파일은 **Kubernetes** 매니페스트로 변환됩니다. 매니페스트는 나중에 클러스터 머신을 구성하는 데 사용되는 **Ignition** 구성 파일로 래핑됩니다.

중요

- **OpenShift Container Platform** 설치 프로그램에서 생성하는 **Ignition** 구성 파일에 **24**시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 **24**시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 **kubelet** 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(**CSR**)을 수동으로 승인해야 합니다. 자세한 내용은 *만료된 컨트롤 플레인 인증서에서 복구* 문서를 참조하십시오.
- 클러스터를 설치한 후 **24**시간에서 **22**시간까지의 인증서가 교체되기 때문에 생성된 후 **12**시간 이내에 **Ignition** 구성 파일을 사용하는 것이 좋습니다. **12**시간 이내에 **Ignition** 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

참고

매니페스트 및 **Ignition** 파일을 생성하는 설치 프로그램은 아키텍처에 따라 다르며 클라이언트 이미지 미러에서 얻을 수 있습니다. 설치 프로그램의 **Linux** 버전은 **s390x**에서만 실행됩니다. 이 설치 프로그램은 **Mac OS** 버전으로도 사용할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램을 가져오셨습니다.
- install-config.yaml 설치 구성 파일을 생성하셨습니다.

프로세스

1. OpenShift Container Platform 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터에 대한 Kubernetes 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 <installation_directory>는 사용자가 만든 install-config.yaml 파일이 포함된 설치 디렉터리를 지정합니다.

주의

3 노드 클러스터를 실행 중이면 다음 단계를 건너 뛰어 컨트롤 플레인 노드 일정을 계획할 수 있도록 하십시오.

중요

예약할 수 없는 기본에서 컨트롤 플레인 노드를 구성하면 추가 서브스크립션이 필요합니다. 이는 컨트롤 플레인 노드가 작업자 노드가 되기 때문입니다.

2. <installation_directory>/manifests/cluster-scheduler-02-config.yml Kubernetes 매니페스트 파일의 mastersSchedulable 매개변수가 false로 설정되어 있는지 확인합니다. 이 설정으로 인해 컨트롤 플레인 머신에서 포드가 예약되지 않습니다.

a. <installation_directory>/manifests/cluster-scheduler-02-config.yml 파일을 엽니다.

b. **mastersSchedulable** 매개변수를 찾아서 값을 **False**로 설정되어 있는지 확인합니다.

c. 파일을 저장하고 종료합니다.

3. **Ignition** 구성 파일을 생성하려면 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행합니다.

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>**는 동일한 설치 디렉터리를 지정합니다.

설치 디렉터리의 부트스트랩, 컨트롤 플레인 및 컴퓨팅 노드에 대한 **Ignition** 구성 파일이 생성됩니다. **kubeadmin-password** 및 **kubeconfig** 파일은 **./<installation_directory>/auth** 디렉터리에 생성됩니다.

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

13.2.12. RHCOS 설치 및 OpenShift Container Platform 부트스트랩 프로세스 시작

프로비저닝하는 **IBM Z** 인프라에 **OpenShift Container Platform**을 설치하려면 **RHCOS(Red Hat Enterprise Linux Core OS)**를 **RHEL(Red Hat Enterprise Linux)** 게스트 가상 시스템으로 설치해야 합니다. **RHCOS**를 설치할 때 설치 중인 머신 유형에 대해 **OpenShift Container Platform** 설치 프로그램에서 생성한 **Ignition** 구성 파일을 제공해야 합니다. 적합한 네트워킹, **DNS** 및 로드 밸런싱 인프라를 구성한 경우 **RHCOS** 머신이 재부팅된 후 **OpenShift Container Platform** 부트스트랩 프로세스가 자동으로 시작됩니다.

사전 패키징된 **QEMU COW2(QCOW2)** 디스크 이미지를 사용하는 **RHCOS**의 빠른 트랙 설치를 수행할 수 있습니다. 또는 새 **QCOW2** 디스크 이미지에서 전체 설치를 수행할 수도 있습니다.

13.2.12.1. 사전 패키징된 QCOW2 디스크 이미지를 사용한 빠른 설치

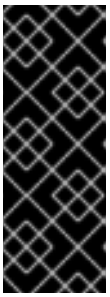
RHCOS(Red Hat Enterprise Linux CoreOS)의 빠른 설치에서 머신을 생성하고 사전 패키징된 RHCOS(Red Hat Enterprise Linux CoreOS) QEMU COW2(QCOW2) 디스크 이미지를 가져오려면 다음 단계를 완료합니다.

사전 요구 사항

- 이 절차에서 RHEL KVM 호스트라고 하는 KVM을 사용하여 RHEL 8.4를 실행하는 하나 이상의 LPAR.
- RHEL KVM 호스트에 KVM/QEMU 하이퍼바이저가 설치되어 있어야 합니다.
- 노드의 호스트 이름 및 역방향 조회를 수행할 수 있는 DNS(Domain name server)입니다.
- IP 주소를 제공하는 DHCP 서버입니다.

프로세스

1. Red Hat Customer Portal의 제품 다운로드 페이지 또는 RHCOS 이미지 미러 페이지에서 RHEL QCOW2(QEMU copy-on-write) 디스크 이미지 파일을 받습니다.

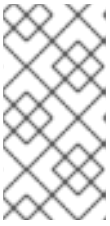


중요

RHCOS 이미지는 OpenShift Container Platform 릴리스에 따라 변경되지 않을 수 있습니다. 설치하는 OpenShift Container Platform 버전과 같거나 그 이하의 버전 중 가장 최신 버전의 이미지를 다운로드해야 합니다. 다음 프로세스에 설명된 적절한 RHCOS QCOW2 이미지만 사용합니다.

2. QCOW2 디스크 이미지 및 Ignition 파일을 RHEL KVM 호스트의 공통 디렉터리에 다운로드합니다.

예: /var/lib/libvirt/images



참고

Ignition 파일은 OpenShift Container Platform 설치 프로그램에서 생성됩니다.

3. 각 KVM 게스트 노드에 대해 QCOW2 디스크 이미지 백업 파일을 사용하여 새 디스크 이미지를 생성합니다.

```
$ qemu-img create -f qcow2 -F qcow2 -b /var/lib/libvirt/images/{source_rhcos_qemu}
/var/lib/libvirt/images/{vmname}.qcow2 {size}
```

4. Ignition 파일 및 새 디스크 이미지를 사용하여 새 KVM 게스트 노드를 생성합니다.

```
$ virt-install --noautoconsole \
  --connect qemu:///system \
  --name {vn_name} \
  --memory {memory} \
  --vcpus {vcpus} \
  --disk {disk} \
  --import \
  --network network={network},mac={mac} \
  --disk path=
{ign_file},format=raw,readonly=on,serial=ignition,startup_policy=optional
```

13.2.12.2. 새 QCOW2 디스크 이미지에 전체 설치

새 QCOW2(QCOW2) 디스크 이미지에 전체 설치로 머신을 생성하려면 다음 단계를 완료합니다.

사전 요구 사항

- 이 절차에서 RHEL KVM 호스트라고 하는 KVM을 사용하여 RHEL 8.4를 실행하는 하나 이상의 LPAR.
- RHEL KVM 호스트에 KVM/QEMU 하이퍼바이저가 설치되어 있어야 합니다.
- 노드의 호스트 이름 및 역방향 조회를 수행할 수 있는 DNS(Domain name server)입니다.
- HTTP 또는 HTTPS 서버가 설정됩니다.

프로세스

1. Red Hat Customer Portal의 Product Downloads 페이지 또는 RHCOS image mirror 페이지에서RHEL kernel, initramfs, rootfs 파일을 받으십시오.

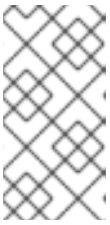
중요

RHCOS 이미지는 OpenShift Container Platform 릴리스에 따라 변경되지 않을 수 있습니다. 설치하는 OpenShift Container Platform 버전과 같거나 그 이하의 버전 중 가장 최신 버전의 이미지를 다운로드해야 합니다. 다음 프로세스에 설명된 적절한 RHCOS QCOW2 이미지만 사용합니다.

OpenShift Container Platform 버전 번호가 파일 이름에 포함됩니다. 다음 예와 유사합니다.

- kernel: rhcos-<version>-live-kernel-<architecture>
- initramfs: rhcos-<version>-live-initramfs.<architecture>.img
- rootfs: rhcos-<version>-live-rootfs.<architecture>.img

2. virt-install을 시작하기 전에 다운로드한 RHEL 라이브 커널, initramfs 및 rootfs 및 Ignition 파일을 HTTP 또는 HTTPS 서버로 이동합니다.



참고

Ignition 파일은 OpenShift Container Platform 설치 프로그램에서 생성됩니다.

3. RHEL 커널, initramfs 및 Ignition 파일, 새 디스크 이미지, 수정된 매개 변수 인수를 사용하여 새 KVM 게스트 노드를 만듭니다.

- --location에 대해 HTTP 또는 HTTPS 서버의 kernel/initrd 위치를 지정합니다.
- coreos.inst.ignition_url=의 경우 시스템 역할의 Ignition 파일을 지정합니다. bootstrap.ign, master.ign 또는 worker.ign을 사용하십시오. HTTP 및 HTTPS 프로토콜만 지원됩니다.
- coreos.live.rootfs_url=의 경우 부팅 중인 커널 및 initramfs와 일치하는 rootfs 아티

팩트를 지정합니다. **HTTP** 및 **HTTPS** 프로토콜만 지원됩니다.

```
$ virt-install \
  --connect qemu:///system \
  --name {vn_name} \
  --vcpus {vcpus} \
  --memory {memory_mb} \
  --disk {vn_name}.qcow2,size={image_size| default(10,true)} \
  --network network={virt_network_parm} \
  --boot hd \
  --location {media_location},kernel={rhcos_kernel},initrd={rhcos_initrd} \
  --extra-args "rd.neednet=1 coreos.inst=yes coreos.inst.install_dev=vda
coreos.live.rootfs_url={rhcos_liveos} ip={ip}::{default_gateway}:
{subnet_mask_length}:{vn_name}:enc1:none:{MTU} nameserver={dns}
coreos.inst.ignition_url={rhcos_ign}" \
  --noautoconsole \
  --wait
```

13.2.12.3. 고급 RHCOS 설치 참조

여기서는 **RHCOS(Red Hat Enterprise Linux CoreOS)** 수동 설치 프로세스를 수정하는 데 사용할 수 있는 네트워킹 구성 및 기타 고급 옵션에 대해 설명합니다. 다음 표에서는 **RHCOS** 라이브 설치 프로그램 및 **coreos-installer** 명령과 함께 사용할 수있는 커널 인수 및 명령 줄 옵션에 대해 설명합니다.

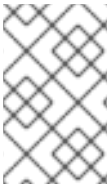
13.2.12.3.1. ISO 설치를 위한 네트워킹 옵션

ISO 이미지에서 **RHCOS**를 설치하는 경우, 해당 이미지를 부팅할 때 수동으로 커널 인수를 추가하여 노드의 네트워킹을 구성할 수 있습니다. 네트워킹 인수를 지정하지 않으면 **RHCOS**에서 **Ignition** 구성 파일을 가져오는 데 네트워킹이 필요함을 감지하면 **initramfs**에서 **DHCP**가 활성화됩니다.

중요

네트워킹 인수를 수동으로 추가할 때 **initramfs**에서 네트워크를 가져오려면 **rd.neednet=1** 커널 인수도 추가해야 합니다.

다음 정보는 **ISO** 설치를 위해 **RHCOS** 노드에서 네트워킹을 구성하는 예제를 제공합니다. 예제에서는 **ip=** 및 **nameserver=** 커널 인수를 사용하는 방법을 설명합니다.



참고

커널 인수를 추가할 때 순서가 중요합니다(**ip =** 및 **nameserver= **).

이는 시스템 부팅 중에 **dracut** 툴로 전달되는 네트워킹 옵션입니다. **dracut**에서 지원하는 네트워킹 옵션에 대한 자세한 내용은 **dracut.cmdline** 메뉴얼 페이지를 참조하십시오.

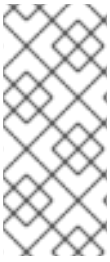
다음 예제는 **ISO** 설치를 위한 네트워킹 옵션입니다.

DHCP 또는 고정 **IP** 주소 구성

IP 주소를 구성하려면 **DHCP(ip=dhcp)**를 사용하거나 개별 고정 **IP** 주소**(ip=<host_ip>)**를 설정합니다. 정적 **IP**를 설정하는 경우 각 노드에서 **DNS** 서버 **IP** 주소 **(nameserver=<dns_ip>)**를 확인합니다. 다음 예제에서는 다음을 설정합니다.

- 노드의 **IP** 주소는 **10.10.10.2**로 설정됩니다.
- 게이트웨이 주소는 **10.10.10.254**로 설정됩니다.
- 넷마스크는 **255.255.255.0**으로
- **core0.example.com**에 대한 호스트 이름
- **4.4.4.41**의 **DNS** 서버 주소
- **auto-configuration** 값을 **none**으로 설정합니다. **IP** 네트워킹이 정적으로 구성되면 자동 구성이 필요하지 않습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
nameserver=4.4.4.41
```



참고

DHCP를 사용하여 **RHCOS** 시스템의 **IP** 주소 지정을 구성하는 경우 시스템은 **DHCP**를 통해 **DNS** 서버 정보도 가져옵니다. **DHCP** 기반 배포의 경우 **DHCP** 서버 구성을 통해 **RHCOS** 노드에서 사용할 **DNS** 서버 주소를 정의할 수 있습니다.

고정 호스트 이름 없이 **IP** 주소 구성

정적 호스트 이름을 할당하지 않고 **IP** 주소를 구성할 수 있습니다. 사용자가 정적 호스트 이름을 설정

하지 않으면 역방향 DNS 조회에 의해 자동으로 선택됩니다. 정적 호스트 이름이 없는 IP 주소를 구성하려면 다음 예제를 참조합니다.

- 노드의 IP 주소는 10.10.10.2로 설정됩니다.
- 게이트웨이 주소는 10.10.10.254로 설정됩니다.
- 넷마스크는 255.255.255.0으로
- 4.4.4.41의 DNS 서버 주소
- auto-configuration 값을 none으로 설정합니다. IP 네트워킹이 정적으로 구성되면 자동 구성이 필요하지 않습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0::enp1s0:none
nameserver=4.4.4.41
```

여러 네트워크 인터페이스 지정

여러 ip= 항목을 설정하여 여러 네트워크 인터페이스를 지정할 수 있습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

기본 게이트웨이 및 경로 구성

선택 사항: rd.route= 값을 설정하여 추가 네트워크에 대한 경로를 구성할 수 있습니다.

참고

하나 이상의 네트워크를 구성할 때 하나의 기본 게이트웨이가 필요합니다. 추가 네트워크 게이트웨이가 기본 네트워크 게이트웨이와 다른 경우 기본 게이트웨이가 기본 네트워크 게이트웨이어야 합니다.

- 다음 명령을 실행하여 기본 게이트웨이를 구성합니다.

```
ip=::10.10.10.254::::
```

- 다음 명령을 입력하여 추가 네트워크의 경로를 구성합니다.

```
rd.route=20.20.20.0/24:20.20.20.254:enp2s0
```

단일 인터페이스에서 DHCP 비활성화

두 개 이상의 네트워크 인터페이스가 있고 하나의 인터페이스만 사용되는 경우와 같이 단일 인터페이스에서 DHCP를 비활성화할 수 있습니다. 이 예에서 enp1s0 인터페이스에는 정적 네트워킹 구성이 있으며 enp2s0 용으로 DHCP가 사용되지 않습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=::::core0.example.com:enp2s0:none
```

DHCP 및 고정 IP 구성 결합

다음과 같이 시스템의 DHCP 및 고정 IP 구성을 여러 네트워크 인터페이스와 결합할 수 있습니다.

```
ip=enp1s0:dhcp
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

개별 인터페이스에서 VLAN 구성

선택 사항: vlan= 매개변수를 사용하여 개별 인터페이스에서 VLAN을 구성할 수 있습니다.

- 네트워크 인터페이스에서 VLAN을 구성하고 고정 IP 주소를 사용하려면 다음 명령을 실행합니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0.100:none
vlan=enp2s0.100:enp2s0
```

- 네트워크 인터페이스에서 VLAN을 구성하고 DHCP를 사용하려면 다음 명령을 실행합니다.

```
ip=enp2s0.100:dhcp
vlan=enp2s0.100:enp2s0
```

여러 DNS 서버 제공

각 서버에 nameserver= 항목을 추가하여 여러 DNS 서버를 제공할 수 있습니다. 예를 들면 다음과 같습니다.

```
nameserver=1.1.1.1
nameserver=8.8.8.8
```

13.2.13. 부트스트랩 프로세스가 완료될 때까지 대기 중

OpenShift Container Platform 부트스트랩 프로세스는 클러스터 노드가 먼저 디스크에 설치된 영구 **RHCOS** 환경으로 부팅된 후에 시작됩니다. **Ignition** 구성 파일을 통해 제공되는 구성 정보는 부트스트랩 프로세스를 초기화하고 머신에 **OpenShift Container Platform**을 설치하는 데 사용됩니다. 부트스트랩 프로세스가 완료될 때까지 기다려야 합니다.

사전 요구 사항

- 클러스터에 대한 **Ignition** 구성 파일이 생성되어 있습니다.
- 적합한 네트워크, **DNS** 및 로드 밸런싱 인프라가 구성되어 있습니다.
- 설치 프로그램을 받아서 클러스터의 **Ignition** 구성 파일을 생성했습니다.
- 클러스터 머신에 **RHCOS**를 설치하고 **OpenShift Container Platform** 설치 프로그램에서 생성된 **Ignition** 구성 파일을 제공했습니다.
- 사용자 시스템에서 직접 인터넷에 액세스하거나 **HTTP** 또는 **HTTPS** 프록시를 사용할 수 있습니다.

프로세스

1. 부트스트랩 프로세스를 모니터링합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

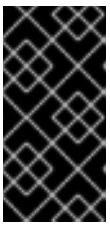
1. **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.
2. 다른 설치 세부 사항을 보려면 **info** 대신 **warn, debug** 또는 **error**를 지정합니다.

출력 예

```
INFO Waiting up to 30m0s for the Kubernetes API at
https://api.test.example.com:6443...
INFO API v1.22.1 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

이 명령은 **Kubernetes API** 서버가 컨트롤 플레인 시스템에서 부트스트랩되었다는 신호를 보낼 때 성공합니다.

2. 부트스트랩 프로세스가 완료된 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다.



중요

이 시점에 로드 밸런서에서 부트스트랩 시스템을 제거해야 합니다. 부트스트랩 머신 자체를 제거하거나 다시 포맷할 수도 있습니다.

13.2.14. CLI를 사용하여 클러스터에 로그인

클러스터 **kubeconfig** 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. **kubeconfig** 파일에는 **CLI**에서 올바른 클러스터 및 **API** 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 **OpenShift Container Platform** 설치 과정에서 생성됩니다.

사전 요구 사항

- **OpenShift Container Platform** 클러스터를 배포했습니다.
- **oc CLI**를 설치했습니다.

프로세스

1. **kubeadmin** 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1

<installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 oc 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

13.2.15. 머신의 인증서 서명 요청 승인

클러스터에 시스템을 추가하면 추가한 시스템별로 보류 중인 인증서 서명 요청(CSR)이 두 개씩 생성됩니다. 이러한 CSR이 승인되었는지 확인해야 하며, 필요한 경우 이를 직접 승인해야 합니다. 클라이언트 요청을 먼저 승인한 다음 서버 요청을 승인해야 합니다.

사전 요구 사항

- 클러스터에 시스템을 추가했습니다.

프로세스

1. 클러스터가 시스템을 인식하는지 확인합니다.

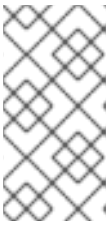
```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0 Ready    master  63m  v1.22.1
```

```
master-1  Ready    master  63m  v1.22.1
master-2  Ready    master  64m  v1.22.1
```

출력에 생성된 모든 시스템이 나열됩니다.



참고

이전 출력에는 일부 CSR이 승인될 때까지 컴퓨팅 노드(작업자 노드라고도 함)가 포함되지 않을 수 있습니다.

2. 보류 중인 CSR을 검토하고 클러스터에 추가한 각 시스템에 대해 Pending 또는 Approved 상태의 클라이언트 및 서버 요청이 표시되는지 확인합니다.

```
$ oc get csr
```

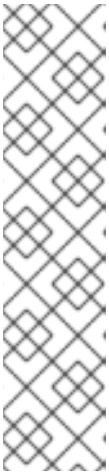
출력 예

```
NAME        AGE   REQUESTOR                                   CONDITION
csr-mddf5   20m   system:node:master-01.example.com   Approved,Issued
csr-z5rln   16m   system:node:worker-21.example.com   Approved,Issued
```

3. CSR이 승인되지 않은 경우, 추가된 시스템에 대한 모든 보류 중인 CSR이 Pending 상태로 전환된 후 클러스터 시스템의 CSR을 승인합니다.

참고

CSR은 교체 주기가 자동으로 만료되므로 클러스터에 시스템을 추가한 후 1 시간 이내에 CSR을 승인하십시오. 한 시간 내에 승인하지 않으면 인증서가 교체되고 각 노드에 대해 두 개 이상의 인증서가 표시됩니다. 이러한 인증서를 모두 승인해야 합니다. 클라이언트 CSR이 승인되면 Kubelet은 인증서에 대한 보조 CSR을 생성하므로 수동 승인이 필요합니다. 그러면 Kubelet에서 동일한 매개변수를 사용하여 새 인증서를 요청하는 경우 인증서 갱신 요청은 machine-approver에 의해 자동으로 승인됩니다.



참고

베어 메탈 및 기타 사용자 프로비저닝 인프라와 같이 머신 **API**를 사용하도록 활성화되지 않는 플랫폼에서 실행되는 클러스터의 경우 **CSR(Kubelet service Certificate Request)**을 자동으로 승인하는 방법을 구현해야 합니다. 요청이 승인되지 않으면 **API** 서버가 **kubelet**에 연결될 때 서비스 인증서가 필요하므로 **oc exec, oc rsh, oc logs** 명령을 성공적으로 수행할 수 없습니다. **Kubelet** 엔드 포인트에 연결하는 모든 작업을 수행하려면 이 인증서 승인이 필요합니다. 이 방법은 새 **CSR**을 감시하고 **CSR**이 **system:node** 또는 **system:admin** 그룹의 **node-bootstrapper** 서비스 계정에 의해 제출되었는지 확인하고 노드의 **ID**를 확인합니다.

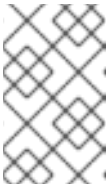
- 개별적으로 승인하려면 유효한 **CSR** 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>**은 현재 **CSR** 목록에 있는 **CSR**의 이름입니다.

- 보류 중인 **CSR**을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```



참고

일부 **Operator**는 일부 **CSR**이 승인될 때까지 사용할 수 없습니다.

4. 이제 클라이언트 요청이 승인되었으므로 클러스터에 추가한 각 머신의 서버 요청을 검토해야 합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
```

```
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 나머지 CSR이 승인되지 않고 Pending 상태인 경우 클러스터 머신의 CSR을 승인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 <csr_name>은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs oc adm certificate approve
```

6. 모든 클라이언트 및 서버 CSR이 승인된 후 머신은 Ready 상태가 됩니다. 다음 명령을 실행하여 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0  Ready    master  73m  v1.22.1
master-1  Ready    master  73m  v1.22.1
master-2  Ready    master  74m  v1.22.1
worker-0  Ready    worker  11m  v1.22.1
worker-1  Ready    worker  11m  v1.22.1
```

참고

머신이 **Ready** 상태로 전환하는 데 서버 **CSR**의 승인 후 몇 분이 걸릴 수 있습니다.

추가 정보

- **CSR**에 대한 자세한 내용은 인증서 서명 요청을 참조하십시오.

13.2.16. Operator의 초기 설정

컨트롤 플레인이 초기화된 후 일부 **Operator**를 즉시 구성하여 모두 사용 가능하도록 해야 합니다.

사전 요구 사항

- 컨트롤 플레인이 초기화되어 있습니다.

프로세스

1. 클러스터 구성 요소가 온라인 상태인지 확인합니다.

```
$ watch -n5 oc get clusteroperators
```

출력 예

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.9.0     True        False         False      19m
baremetal                                  4.9.0     True        False         False      37m
cloud-credential                           4.9.0     True        False         False      40m
cluster-autoscaler                         4.9.0     True        False         False      37m
config-operator                            4.9.0     True        False         False      38m
console                                    4.9.0     True        False         False      26m
csi-snapshot-controller                    4.9.0     True        False         False      37m
dns                                        4.9.0     True        False         False      37m
etcd                                       4.9.0     True        False         False      36m
image-registry                             4.9.0     True        False         False      31m
ingress                                    4.9.0     True        False         False      30m
insights                                   4.9.0     True        False         False      31m
kube-apiserver                             4.9.0     True        False         False      26m
kube-controller-manager                    4.9.0     True        False         False      36m
kube-scheduler                             4.9.0     True        False         False      36m
```

```
kube-storage-version-migrator              4.9.0     True        False         False      37m
machine-api                                4.9.0     True        False         False      29m
machine-approver                           4.9.0     True        False         False      37m
machine-config                             4.9.0     True        False         False      36m
marketplace                                4.9.0     True        False         False      37m
monitoring                                 4.9.0     True        False         False      29m
network                                    4.9.0     True        False         False      38m
node-tuning                                4.9.0     True        False         False      37m
openshift-apiserver                        4.9.0     True        False         False      32m
openshift-controller-manager               4.9.0     True        False         False      30m
openshift-samples                          4.9.0     True        False         False      32m
operator-lifecycle-manager                 4.9.0     True        False         False      37m
operator-lifecycle-manager-catalog         4.9.0     True        False         False      37m
operator-lifecycle-manager-packageserver   4.9.0     True        False         False      32m
service-ca                                 4.9.0     True        False         False      38m
storage                                    4.9.0     True        False         False      37m
```

2. 사용할 수 없는 **Operator**를 구성합니다.

13.2.16.1. 이미지 레지스트리 스토리지 구성

기본 스토리지를 제공하지 않는 플랫폼에서는 처음에 **Image Registry Operator**를 사용할 수 없습니다. 설치한 후에 스토리지를 사용하도록 레지스트리를 구성하여 **Registry Operator**를 사용 가능하도록 만들어야 합니다.

프로덕션 클러스터에 필요한 영구 볼륨을 구성하는 과정의 지침이 표시됩니다. 해당하는 경우, 프로덕션 환경 외 클러스터에서만 사용할 수 있는 저장 위치로서 빈 디렉터리를 구성하는 과정의 지침이 표시됩니다.

업그레이드 중에 **Recreate** 롤아웃 전략을 사용하여 이미지 레지스트리의 블록 스토리지 유형 사용을 허용하기 위한 추가 지침이 제공됩니다.

13.2.16.1.1. IBM Z용 레지스트리 스토리지 구성

클러스터 관리자는 설치한 후 스토리지를 사용하도록 레지스트리를 구성해야 합니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

- IBM Z에 클러스터가 있습니다.

- Red Hat OpenShift Container Storage와 같이 클러스터용 영구 스토리지 프로비저닝이 있습니다.

중요

OpenShift Container Platform은 복제본이 하나만 있는 경우 이미지 레지스트리 스토리지에 대한 ReadWriteOnce 액세스를 지원합니다. ReadWriteOnce 액세스에서는 레지스트리가 Recreate 롤아웃 전략을 사용해야 합니다. 두 개 이상의 복제본으로 고 가용성을 지원하는 이미지 레지스트리를 배포하려면 ReadWriteMany 액세스가 필요합니다.

- "100Gi" 용량이 필요합니다.

프로세스

1. 스토리지를 사용하도록 레지스트리를 구성하기 위해 configs.imageregistry/cluster 리소스에서 spec.storage.pvc를 변경합니다.

참고

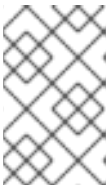
공유 스토리지를 사용할 때 보안 설정을 확인하여 외부에서의 액세스를 방지합니다.

2. 레지스트리 pod가 없는지 확인합니다.

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

출력 예

```
No resourses found in openshift-image-registry namespace
```



참고

출력에 레지스트리 **Pod**가 있는 경우 이 절차를 계속할 필요가 없습니다.

3. 레지스트리 구성을 확인합니다.

```
$ oc edit configs.imageregistry.operator.openshift.io
```

출력 예

```
storage:
  pvc:
    claim:
```

image-registry-storage PVC의 자동 생성을 허용하도록 **claim** 필드를 비워 둡니다.

4. **clusteroperator** 상태를 확인합니다.

```
$ oc get clusteroperator image-registry
```

출력 예

```
NAME              VERSION   AVAILABLE   PROGRESSING   DEGRADED   SINCE   MESSAGE
image-registry    4.9       True        False         False      6h50m
```

5. 이미지를 빌드 및 푸시할 수 있도록 레지스트리의 관리가 설정되어 있는지 확인하십시오.

- 다음을 실행합니다.

```
$ oc edit configs.imageregistry/cluster
```

다음으로 라인을 변경하십시오.

```
managementState: Removed
```

다음으로 변경

```
managementState: Managed
```

13.2.16.1.2. 프로덕션 환경 외 클러스터에서 이미지 레지스트리의 스토리지 구성

이미지 레지스트리 Operator에 대한 스토리지를 구성해야 합니다. 프로덕션 환경 외 클러스터의 경우, 이미지 레지스트리를 빈 디렉터리로 설정할 수 있습니다. 이렇게 하는 경우 레지스트리를 다시 시작하면 모든 이미지가 손실됩니다.

프로세스

- 이미지 레지스트리 스토리지를 빈 디렉터리로 설정하려면 다음을 수행하십시오.

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch
'{"spec":{"storage":{"emptyDir":{}}}}'
```

주의

프로덕션 환경 외 클러스터에 대해서만 이 옵션을 구성하십시오.

Image Registry Operator가 구성 요소를 초기화하기 전에 이 명령을 실행하면 oc patch 명령이 실패하며 다음 오류가 발생합니다.

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster"
not found
```

몇 분 후에 명령을 다시 실행하십시오.

13.2.17. 사용자 프로비저닝 인프라에 설치 완료

Operator 구성을 완료한 후 제공하는 인프라에 클러스터 설치를 완료할 수 있습니다.

사전 요구 사항

- 컨트롤 플레인이 초기화되어 있습니다.
- 초기 Operator 구성을 완료해야 합니다.

프로세스

1. 다음 명령을 사용하여 모든 클러스터 구성 요소가 온라인 상태인지 확인합니다.

```
$ watch -n5 oc get clusteroperators
```

출력 예

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.9.0     True        False         False      19m
baremetal                                  4.9.0     True        False         False      37m
cloud-credential                           4.9.0     True        False         False      40m
cluster-autoscaler                         4.9.0     True        False         False      37m
config-operator                            4.9.0     True        False         False      38m
console                                    4.9.0     True        False         False      26m
csi-snapshot-controller                    4.9.0     True        False         False      37m
dns                                        4.9.0     True        False         False      37m
etcd                                       4.9.0     True        False         False      36m
image-registry                             4.9.0     True        False         False      31m
ingress                                    4.9.0     True        False         False      30m
insights                                   4.9.0     True        False         False      31m
kube-apiserver                             4.9.0     True        False         False      26m
kube-controller-manager                    4.9.0     True        False         False      36m
kube-scheduler                             4.9.0     True        False         False      36m
kube-storage-version-migrator              4.9.0     True        False         False      37m
machine-api                                4.9.0     True        False         False      29m
machine-approver                           4.9.0     True        False         False      37m
machine-config                             4.9.0     True        False         False      36m
marketplace                                4.9.0     True        False         False      37m
monitoring                                 4.9.0     True        False         False      29m
network                                    4.9.0     True        False         False      38m
node-tuning                                4.9.0     True        False         False      37m
openshift-apiserver                        4.9.0     True        False         False      32m
```

```
openshift-controller-manager             4.9.0     True        False         False      30m
openshift-samples                        4.9.0     True        False         False      32m
operator-lifecycle-manager               4.9.0     True        False         False      37m
operator-lifecycle-manager-catalog       4.9.0     True        False         False      37m
operator-lifecycle-manager-packageserver 4.9.0     True        False         False      32m
service-ca                               4.9.0     True        False         False      38m
storage                                  4.9.0     True        False         False      37m
```

또는 다음 명령은 모든 클러스터를 사용할 수 있을 때 알립니다. 또한 인증 정보를 검색하고 표시합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

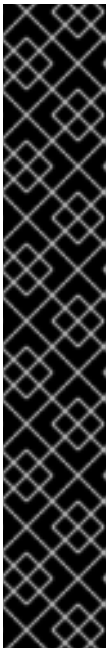
1

<installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

출력 예

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator가 Kubernetes API 서버에서 OpenShift Container Platform 클러스터 배포를 완료하면 명령이 성공합니다.



중요

- 설치 프로그램에서 생성하는 **Ignition** 구성 파일에 **24**시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 **24**시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 **kubelet** 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(**CSR**)을 수동으로 승인해야 합니다. 자세한 내용은 *만료된 컨트롤 플레인 인증서에서 복구* 문서를 참조하십시오.

- 클러스터를 설치한 후 **24**시간에서 **22**시간까지의 인증서가 교체되기 때문에 생성된 후 **12**시간 이내에 **Ignition** 구성 파일을 사용하는 것이 좋습니다. **12**시간 이내에 **Ignition** 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

2. **Kubernetes API** 서버가 **Pod**와 통신하고 있는지 확인합니다.

   a. 모든 **Pod** 목록을 보려면 다음 명령을 사용하십시오.

   ```
   $ oc get pods --all-namespaces
   ```

   출력 예

   ```
   NAMESPACE                       NAME                                        READY   STATUS
   RESTARTS   AGE
   openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8
   1/1     Running   1          9m
   openshift-apiserver             apiserver-67b9g                             1/1     Running   0
   3m
   openshift-apiserver             apiserver-ljcmx                             1/1     Running   0
   1m
   openshift-apiserver             apiserver-z25h4                             1/1     Running   0
   2m
   openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8
   1/1     Running   0          5m
   ...
   ```

   b. 다음 명령을 사용하여 이전 명령의 출력에 나열된 **Pod**의 로그를 표시합니다.

   -

```
$ oc logs <pod_name> -n <namespace> 1
```

1

이전 명령의 출력에 표시된 대로 **Pod** 이름과 네임스페이스를 지정합니다.

Pod 로그가 표시되면 **Kubernetes API** 서버는 클러스터 시스템과 통신할 수 있습니다.

3. **FCP(Fibre Channel Protocol)**를 사용하는 설치에는 다중 경로를 활성화하기 위해 추가 단계가 필요합니다. 설치 중에 멀티패스를 활성화하지 마십시오.

자세한 내용은 *설치 후 머신 구성 작업* 설명서에서 **"RHCOS에서 커널 인수를 사용하여 멀티패스 활성화"**를 참조하십시오.

13.2.18. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 **Telemetry** 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 **Telemetry**가 자동으로 실행되고 **OpenShift Cluster Manager** 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 **OpenShift Cluster Manager**를 사용하여 자동으로 또는 **OpenShift Cluster Manager**를 사용하여 수동으로 유지 관리되는지 확인한 후 **subscription watch**를 사용하여 계정 또는 다중 클러스터 수준에서 **OpenShift Container Platform** 서브스크립션을 추적합니다.

추가 리소스

- **Telemetry** 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

13.2.19. 디버깅 정보 수집

IBM Z에서 **OpenShift Container Platform** 설치와 관련된 특정 문제를 해결하고 디버깅하는데 도움이 될 수 있는 디버깅 정보를 수집할 수 있습니다.

사전 요구 사항

- **oc CLI** 도구가 설치되어 있어야 합니다.

프로세스

1. 클러스터에 로그인합니다.

```
$ oc login -u <username>
```

2. 하드웨어 정보를 수집하는 노드에서 디버깅 컨테이너를 시작합니다.

```
$ oc debug node/<nodename>
```

3. **/host** 파일 시스템으로 변경하고 **toolbox**를 시작합니다.

```
$ chroot /host
$ toolbox
```

4. **dbginfo** 데이터를 수집합니다.

```
$ dbginfo.sh
```

5. 다음으로 **scp**를 사용하여 데이터를 검색할 수 있습니다.

추가 리소스

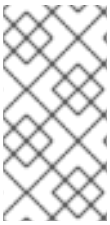
- SSH 없이 OpenShift 4 노드 내에서 SOSREPORT를 생성하는 방법을 참조하십시오.

13.2.20. 다음 단계

- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.

13.3. 네트워크가 제한된 환경에서 IBM Z 및 LINUXONE에 RHEL KVM으로 클러스터 설치

OpenShift Container Platform 버전 **4.9**에서는 네트워크가 제한된 환경에서 사용자가 제공하는 **IBM Z** 및 **LinuxONE** 인프라에 클러스터를 설치할 수 있습니다.



참고

이 문서는 **IBM Z**에 대해서만 설명하지만 여기에 있는 모든 정보는 **LinuxONE**에도 적용됩니다.

중요

베어 메탈 이외의 플랫폼의 경우 추가 고려 사항이 있습니다. **OpenShift Container Platform** 클러스터를 설치하기 전에 **guidelines for deploying OpenShift Container Platform on non-tested platforms**에 있는 내용을 확인하십시오.

13.3.1. 사전 요구 사항

- **OpenShift Container Platform** 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.
- 미러 호스트에 레지스트리를 생성하고 사용 중인 **OpenShift Container Platform** 버전의 **imageContentSources** 데이터를 가져옵니다.
- 설치 프로세스를 시작하기 전에 기존 설치 파일을 이동하거나 제거해야 합니다. 이렇게 하면 설치 프로세스 중에 필요한 설치 파일이 생성되고 업데이트됩니다.

  중요

  설치 미디어에 액세스할 수있는 시스템에서 설치 프로세스를 수행해야 합니다.
- 클러스터 용 **NFS**를 사용하여 영구 스토리지를 프로비저닝합니다. 프라이빗 이미지 레지스트리를 배포하려면 **ReadWriteMany** 액세스 권한을 사용하여 영구 스토리지를 설정해야 합니다.
- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성해야 합니다.
- **LPAR(Logical partition)**에서 호스팅되고 **RHEL 8.4** 이상을 기반으로 하는 **RHEL KVM**(커널 가상 머신) 시스템을 프로비저닝합니다.

참고

프록시를 구성하는 경우에도 해당 사이트 목록을 검토하십시오.

13.3.2. 네트워크가 제한된 환경에서의 설치 정보

OpenShift Container Platform 4.9에서는 소프트웨어 구성 요소를 받기 위한 인터넷 접속이 필요하지 않은 설치를 수행할 수 있습니다. 제한된 네트워크 설치는 클러스터를 설치하는 클라우드 플랫폼에 따라 설치 관리자 프로비저닝 인프라 또는 사용자 프로비저닝 인프라를 사용하여 완료할 수 있습니다.

클라우드 플랫폼에 제한된 네트워크 설치를 수행하는 방법을 선택해도 클라우드 API에 액세스는 가능해야 합니다. Amazon Web Service의 Route 53 DNS 및 IAM 서비스와 같은 일부 클라우드 기능에는 인터넷 액세스가 필요합니다. 사용 중인 네트워크에 따라 베어메탈 하드웨어 또는 VMware vSphere에 설치하기 위해 필요한 인터넷 액세스가 줄어들 수 있습니다.

제한된 네트워크 설치를 완료하려면 OpenShift Container Platform 레지스트리의 내용을 미러링하고 설치 미디어를 포함할 레지스트리를 생성해야 합니다. 인터넷과 폐쇄 네트워크에 모두 액세스하거나 제한 사항을 따르는 다른 방법을 통해 미러 호스트에 레지스트리를 생성할 수 있습니다.

중요

사용자 프로비저닝 설치의 구성이 복잡하므로 사용자 프로비저닝 인프라를 사용하여 제한된 네트워크 설치를 시도하기 전에 표준 사용자 프로비저닝 인프라 설치를 완료하는 것이 좋습니다. 이 테스트 설치를 완료하면 제한된 네트워크에 설치하는 동안 발생할 수 있는 문제를 보다 쉽게 파악 및 해결할 수 있습니다.

13.3.2.1. 추가 제한

제한된 네트워크의 클러스터에는 다음과 같은 추가 제한이 있습니다.

- ClusterVersion 상태에 사용 가능한 업데이트를 검색할 수 없음 오류가 포함되어 있습니다.
- 기본적으로 필요한 이미지 스트림 태그에 액세스할 수 없기 때문에 개발자 카탈로그의 내용을 사용할 수 없습니다.

13.3.3. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하기 위해 필요한 이미지를 받으려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.

중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

13.3.4. 사용자 프로비저닝 인프라를 포함한 클러스터의 시스템 요구사항

사용자 프로비저닝 인프라가 포함된 클러스터의 경우, 필요한 모든 시스템을 배포해야 합니다.

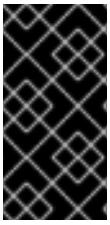
RHEL 8.4 이상을 기반으로 하는 하나 이상의 KVM 호스트 머신입니다. 각 RHEL KVM 호스트 머신에는 libvirt가 설치되어 실행되어야 합니다. 가상 머신은 각 RHEL KVM 호스트 머신에서 프로비저닝됩니다.

13.3.4.1. 필요한 시스템

최소 OpenShift Container Platform 클러스터에 다음과 같은 호스트가 필요합니다.

표 13.17. 최소 필수 호스트

호스트	설명
임시 부트스트랩 시스템 한 개	컨트롤 플레인 시스템 세 개에 OpenShift Container Platform 클러스터를 배포하기 위한 부트스트랩 시스템이 클러스터에 필요합니다. 클러스터를 설치한 후 부트스트랩 시스템을 제거할 수 있습니다.
컨트롤 플레인 시스템 세 개	컨트롤 플레인 시스템은 컨트롤 플레인을 구성하는 Kubernetes 및 OpenShift Container Platform 서비스를 실행합니다.
두 개 이상의 컴퓨팅 시스템(작업자 시스템이라고도 함).	OpenShift Container Platform 사용자가 요청한 워크로드는 컴퓨팅 머신에서 실행됩니다.



중요

클러스터의 고가용성을 개선하려면 컨트롤 플레인 시스템을 두 개 이상의 물리적 시스템의 서로 다른 RHEL 인스턴스에 배포하십시오.

부트스트랩, 컨트롤 플레인 및 컴퓨팅 시스템은 운영 체제로 RHCOS (Red Hat Enterprise Linux CoreOS)를 사용해야 합니다.

Red Hat Enterprise Linux 기술 기능 및 제한을 참조하십시오.

13.3.4.2. 네트워크 연결 요구사항

OpenShift Container Platform 설치 프로그램은 Ignition 파일을 생성합니다. 이 파일은 모든 RHCOS(Red Hat Enterprise Linux CoreOS) 가상 머신에 필요합니다. OpenShift Container Platform의 자동화된 설치는 부트스트랩 머신에서 수행합니다. 각 노드에서 OpenShift Container Platform 설치를 시작하고 Kubernetes 클러스터를 시작한 다음 완료합니다. 이 부트스트랩 중에 가상 머신에 DHCP(Dynamic Host Configuration Protocol) 서버 또는 고정 IP 주소를 통해 기존 네트워크 연결이 있어야 합니다.

13.3.4.3. IBM Z 네트워크 연결 요구 사항

RHEL KVM 아래의 IBM Z에 설치하려면 다음이 필요합니다.

- OSA 또는 RoCE 네트워크 어댑터로 구성된 RHEL KVM 호스트.
-

libvirt 또는 MacVTap에서 브릿지 네트워킹을 사용하여 네트워크를 게스트에 연결하도록 구성된 RHEL KVM 호스트.

가상 네트워크 연결 유형을 참조하십시오.

13.3.4.4. 호스트 머신 리소스 요구사항

환경의 RHEL KVM 호스트는 다음 요구사항을 충족해야 OpenShift Container Platform 환경에 계획하는 가상 머신을 호스팅해야 합니다. 가상화 시작하기를 참조하십시오.

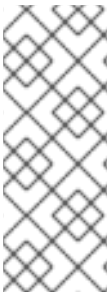
다음 IBM 하드웨어에 OpenShift Container Platform 버전 4.9를 설치할 수 있습니다.

- IBM z15(모든 모델), IBM z14(모든 모델), IBM z13 및 IBM z13s
- LinuxONE, 모든 버전

13.3.4.5. 최소 IBM Z 시스템 환경

하드웨어 요구 사항

- 각 클러스터에 대해 SMT2가 활성화된 6개의 IFL과 동일합니다.
- LoadBalancer 서비스에 연결하고 클러스터 외부 트래픽에 대한 데이터를 제공하는 네트워크 연결은 하나 이상 있습니다.



참고

전용 또는 공유 IFL을 사용하여 충분한 컴퓨팅 리소스를 할당할 수 있습니다. 리소스 공유는 IBM Z의 주요 강점 중 하나입니다. 그러나 각 하이퍼바이저 계층에서 용량을 올바르게 조정하고 모든 OpenShift Container Platform 클러스터에 충분한 리소스를 확인해야 합니다.

중요

클러스터의 전반적인 성능에 영향을 미칠 수 있으므로 **OpenShift Container Platform** 클러스터를 설정하는 데 사용되는 **LPAR**은 충분한 컴퓨팅 용량을 제공해야 합니다. 이 컨텍스트에서 하이퍼바이저 수준의 **LPAR** 가중치 관리, 권한 부여 및 **CPU** 공유는 중요한 역할을 합니다.

운영 체제 요구 사항

- **libvirt**에서 관리하는 **KVM**을 사용하여 **RHEL 8.4** 이상을 실행하는 하나의 **LPAR**

RHEL KVM 호스트에서 다음을 설정합니다.

- **OpenShift Container Platform** 컨트롤 플레인 시스템 용 게스트 가상 머신 세 개
- **OpenShift Container Platform** 컴퓨팅 머신 용 게스트 가상 머신 두 개
- 임시 **OpenShift Container Platform** 부트스트랩 시스템용 게스트 가상 머신 **1**개

13.3.4.6. 최소 리소스 요구사항

각 클러스터 가상 머신은 다음과 같은 최소 요구사항을 충족해야 합니다.

가상 머신	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS
부트스트랩	RHCOS	4	16GB	100GB	해당 없음
컨트롤 플레인	RHCOS	4	16GB	100GB	해당 없음
컴퓨팅	RHCOS	2	8GB	100GB	해당 없음

1. **SMT-2**가 활성화된 경우 하나의 물리적 코어(**IFL**)는 두 개의 논리 코어(스레드)를 제공합니다. 하이퍼바이저는 두 개 이상의 **vCPU**를 제공할 수 있습니다.

13.3.4.7. 권장되는 IBM Z 시스템 환경

하드웨어 요구 사항

- 각 클러스터에 대해 SMT2가 활성화된 6개의 IFL과 동등한 3개의 LPARS가 있습니다.

- LoadBalancer 서비스에 연결하고 클러스터 외부의 트래픽에 대한 데이터를 제공하는 두 개의 네트워크 연결

운영 체제 요구 사항

- 고가용성을 위해 libvirt에서 관리하는 KVM을 사용하여 RHEL 8.4 이상을 실행하는 두 개의 LPAR.

RHEL KVM 호스트에서 다음을 설정합니다.

- RHEL KVM 호스트 머신에 분산된 OpenShift Container Platform 컨트롤 플레인 시스템용 게스트 가상 머신 세 개

- RHEL KVM 호스트 머신에 분산된 OpenShift Container Platform 컴퓨팅 머신용 게스트 가상 머신 6개 이상

- 임시 OpenShift Container Platform 부트스트랩 시스템용 게스트 가상 머신 1개

- 오버 커밋된 환경에서 통합 구성 요소를 사용하려면 cpu_shares 를 사용하여 컨트롤 플레인의 우선 순위를 높입니다. 인프라 노드가 있는 경우 동일한 작업을 수행합니다. IBM 문서의 schedinfo 를 참조하십시오.

13.3.4.8. 권장되는 리소스 요구사항.

각 클러스터 가상 머신의 권장되는 요구 사항은 다음과 같습니다.

가상 머신	운영 체제	vCPU	가상 RAM	스토리지
부트스트랩	RHCOS	4	16GB	120GB
컨트롤 플레인	RHCOS	8	16GB	120GB
컴퓨팅	RHCOS	6	8GB	120GB

13.3.4.9. 인증서 서명 요청 관리

사용자가 프로비저닝하는 인프라를 사용하는 경우 자동 시스템 관리 기능으로 인해 클러스터의 액세스가 제한되므로 설치한 후 클러스터 인증서 서명 요청(CSR)을 승인하는 메커니즘을 제공해야 합니다. kube-controller-manager는 kubelet 클라이언트 CSR만 승인합니다. machine-approver는 올바른 시스템에서 발행한 요청인지 확인할 수 없기 때문에 kubelet 자격 증명을 사용하여 요청하는 서비스 인증서의 유효성을 보장할 수 없습니다. kubelet 서빙 인증서 요청의 유효성을 확인하고 요청을 승인하는 방법을 결정하여 구현해야 합니다.

추가 리소스

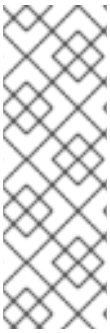
- IBM Z 및 LinuxONE 환경에 대한 권장 호스트 사례

13.3.4.10. 사용자 프로비저닝 인프라에 대한 네트워킹 요구사항

모든 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템이 부팅 중에 Ignition 구성 파일을 가져오려면 initramfs에 네트워킹을 구성해야 합니다.

초기 부팅 과정에서 시스템에 필요한 부팅 옵션을 제공하여 DHCP 서버를 통해 설정하거나 정적으로 설정하는 IP 주소 구성이 필요합니다. 네트워크 연결이 설정된 후 시스템은 HTTP 또는 HTTPS 서버에서 Ignition 구성 파일을 다운로드합니다. 그런 다음 Ignition 구성 파일을 사용하여 각 머신의 정확한 상태를 설정합니다. Machine Config Operator는 설치 후 새 인증서 또는 키 적용과 같은 머신에 대한 추가 변경을 완료합니다.

클러스터 시스템의 장기적인 관리를 위해 DHCP 서버를 사용하는 것이 좋습니다. DHCP 서버가 클러스터 시스템에 영구 IP 주소, DNS 서버 정보 및 호스트 이름을 제공하도록 구성되었는지 확인합니다.



참고

사용자 프로비저닝 인프라에 DHCP 서비스를 사용할 수 없는 경우 RHCOS 설치 시 노드에 IP 네트워킹 구성과 DNS 서버의 주소를 대신 제공할 수 있습니다. ISO 이미지에서 설치하는 경우 부팅 인수로 전달할 수 있습니다. 고정 IP 프로비저닝 *및 고급 네트워킹 옵션에 대한 자세한 내용은 RHCOS 설치 및 OpenShift Container Platform 부트스트랩 프로세스 시작* 섹션을 참조하십시오.

Kubernetes API 서버가 클러스터 시스템의 노드 이름을 확인할 수 있어야 합니다. API 서버와 작업자 노드가 서로 다른 영역에 있는 경우, API 서버가 노드 이름을 확인할 수 있도록 기본 DNS 검색 영역을 설정할 수 있습니다. 노드 개체와 모든 DNS 요청에서 항상 정규화된 도메인 이름으로 호스트를 가리키는 것도 지원되는 방법입니다

13.3.4.10.1. DHCP를 통해 클러스터 노드의 호스트 이름 설정

RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 호스트 이름은 NetworkManager를 통해 설정됩니다. 기본적으로 시스템은 DHCP를 통해 호스트 이름을 가져옵니다. DHCP에서 호스트 이름을 제공하지 않거나, 커널 인수를 통해 정적으로 설정하거나 다른 방법을 통해 설정하면 역방향 DNS 조회를 통해 얻을 수 있습니다. 역방향 DNS 조회는 노드에서 네트워크를 초기화한 후 수행되며 확인하는 데 시간이 걸릴 수 있습니다. 다른 시스템 서비스는 이 보다 먼저 시작하여 호스트 이름을 localhost 등으로 감지할 수 있습니다. DHCP를 사용하여 각 클러스터 노드의 호스트 이름을 제공하여 이 문제를 방지할 수 있습니다.

또한 DHCP를 통해 호스트 이름을 설정하면 DNS 분할 수평 구현 환경에서 수동으로 DNS 레코드 이름 구성 오류를 무시할 수 있습니다.

13.3.4.10.2. 네트워크 연결 요구사항

OpenShift Container Platform 클러스터 구성 요소가 통신할 수 있도록 시스템 간 네트워크 연결을 구성해야 합니다. 각 시스템에서 클러스터에 있는 다른 모든 시스템의 호스트 이름을 확인할 수 있어야 합니다.

이 섹션에서는 필요한 포트에 대해 자세히 설명합니다.

표 13.18. 모든 시스템 간 통신에 사용되는 포트

프로토콜	포트	설명
ICMP	해당 없음	네트워크 연결성 테스트
TCP	**1936**	메트릭
	9000-**9999**	**9100**-**9101** 포트의 노드 내보내기 및 **9099** 포트의 Cluster Version Operator를 포함한 호스트 수준 서비스.
	10250-**10259**	Kubernetes에서 예약하는 기본 포트
	10256	openshift-sdn
UDP	**4789**	VXLAN 및 Geneve
	6081	VXLAN 및 Geneve
	9000-**9999**	**9100**-**9101** 포트의 노드 내보내기를 포함한 호스트 수준 서비스.
	500	IPsec IKE 패킷

프로토콜	포트	설명
	4500	IPsec NAT-T 패킷
TCP/UDP	**30000-32767**	Kubernetes 노드 포트
ESP	해당 없음	IPsec Encapsulating Security Payload (ESP)

표 13.19. 모든 시스템과 컨트롤 플레인 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**6443**	Kubernetes API

표 13.20. 컨트롤 플레인 머신 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**2379-2380**	etcd 서버 및 피어 포트

사용자 프로비저닝 인프라에 대한 NTP 구성

OpenShift Container Platform 클러스터는 기본적으로 공용 **NTP(Network Time Protocol)** 서버를 사용하도록 구성되어 있습니다. 로컬 엔터프라이즈 **NTP** 서버를 사용하거나 클러스터가 연결이 끊긴 네트워크에 배포되는 경우 특정 시간 서버를 사용하도록 클러스터를 구성할 수 있습니다. 자세한 내용은 ***chrony 타임 서비스 설정*** 문서를 참조하십시오.

DHCP 서버가 **NTP** 서버 정보를 제공하는 경우 **RHCOS(Red Hat Enterprise Linux CoreOS)** 시스템의 **chrony** 타임 서비스에서 정보를 읽고 **NTP** 서버와 클럭을 동기화할 수 있습니다.

추가 리소스

- **chrony 타임 서비스 설정**

13.3.4.11. 사용자 프로비저닝 DNS 요구사항

OpenShift Container Platform 배포의 경우 다음 구성 요소에 **DNS** 이름을 확인해야 합니다.

- **Kubernetes API**

- **OpenShift Container Platform 애플리케이션 와일드카드**

- **부트스트랩, 컨트롤 플레인 및 컴퓨팅 시스템**

Kubernetes API, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템에 대한 역방향 DNS 확인이 필요합니다.

DNS A/AAAA 또는 CNAME 레코드는 이름 확인에 사용되며 PTR 레코드는 역방향 이름 확인에 사용됩니다. RHCOS (Red Hat Enterprise Linux CoreOS)는 DHCP에서 호스트 이름을 제공하지 않는 한 모든 노드의 호스트 이름을 설정할 때 역방향 레코드를 사용하기 때문에 역방향 레코드가 중요합니다. 또한 역방향 레코드는 OpenShift Container Platform이 작동하는 데 필요한 인증서 서명 요청 (CSR)을 생성하는 데 사용됩니다.

사용자가 프로비저닝한 OpenShift Container Platform 클러스터에 대해 다음 DNS 레코드가 필요하며 설치 전에 있어야 합니다. 각 레코드에서 <cluster_name>은 클러스터 이름이고 <base_domain>은 install-config.yaml 파일에서 지정하는 기반 도메인입니다. 전체 DNS 레코드는 <component>.<cluster_name>.<base_domain> 형식입니다.

표 13.21. 필수 DNS 레코드

구성 요소	레코드	설명
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API 로드 밸런서를 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.
	api-int.<cluster_name>.<base_domain>.	내부적으로 API 로드 밸런서를 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.  **중요** API 서버는 Kubernetes에 기록된 호스트 이름으로 작업자 노드를 확인할 수 있어야 합니다. API 서버가 노드 이름을 확인할 수 없는 경우 프록시된 API 호출이 실패할 수 있으며 pod에서 로그를 검색할 수 없습니다.

구성 요소	레코드	설명
라우트	***.apps.<cluster_name>. <base_domain>.**	애플리케이션 인그레스 로드 밸런서를 참조하는 와일드카드 DNS A/AAA 또는 CNAME 레코드입니다. 애플리케이션 인그레스 로드 밸런서는 Ingress 컨트롤러 Pod를 실행하는 머신을 대상으로 합니다. Ingress 컨트롤러 Pod는 기본적으로 컴퓨팅 머신에서 실행됩니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다. 예를 들어 **console-openshift-console.apps. <cluster_name>.<base_domain>**은 OpenShift Container Platform 콘솔의 와일드카드 경로로 사용됩니다.
부트스트랩 시스템	**bootstrap.<cluster_name>. <base_domain>.**	부트스트랩 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.
컨트롤 플레인 머신	**<master><n>. <cluster_name>. <base_domain>.**	컨트롤 플레인 노드의 각 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.
컴퓨팅 머신	**<worker><n>. <cluster_name>. <base_domain>.**	작업자 노드의 각 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.

참고

OpenShift Container Platform 4.4 이상에서는 DNS 구성에서 etcd 호스트 및 SRV 레코드를 지정할 필요가 없습니다.

작은 정보

dig 명령을 사용하여 이름과 역방향 이름을 확인할 수 있습니다. 자세한 검증 단계는 *사용자 프로비저닝 인프라의 DNS 확인* 섹션을 참조하십시오.

13.3.4.11.1. 사용자 프로비저닝 클러스터의 DNS 구성 예

이 섹션에서는 사용자 프로비저닝 인프라에 OpenShift Container Platform을 배포하기 위한 DNS 요구 사항을 충족하는 A 및 PTR 레코드 구성 샘플을 제공합니다. 샘플은 하나의 DNS 솔루션을 선택하기 위한 조언을 제공하기 위한 것이 아닙니다.

이 예제에서 클러스터 이름은 **ocp4**이고 기본 도메인은 **example.com**입니다.

사용자 프로비저닝 클러스터의 **DNS A** 레코드 구성 예

다음 **BIND** 영역 파일의 예제에서는 사용자가 프로비저닝한 클러스터의 이름 확인을 위한 샘플 **A** 레코드를 보여줍니다.

예 13.4. 샘플 DNS 영역 데이터베이스

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
master0.ocp4.example.com. IN A 192.168.1.97 5
master1.ocp4.example.com. IN A 192.168.1.98 6
master2.ocp4.example.com. IN A 192.168.1.99 7
;
worker0.ocp4.example.com. IN A 192.168.1.11 8
worker1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

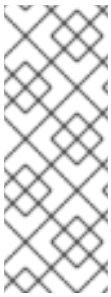
1

Kubernetes API의 이름 확인을 제공합니다. 레코드는 **API** 로드 밸런서의 **IP** 주소를 나타냅니다.

2

3

와일드카드 경로의 이름 확인을 제공합니다. 레코드는 애플리케이션 인그레스 로드 밸런서의 IP 주소를 나타냅니다. 애플리케이션 인그레스 로드 밸런서는 Ingress 컨트롤러 Pod를 실행하는 머신을 대상으로 합니다. Ingress 컨트롤러 Pod는 기본적으로 컴퓨팅 머신에서 실행됩니다.



참고

이 예제에서는 Kubernetes API 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 API 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

4

부트스트랩 시스템의 이름 확인을 제공합니다.

5 6 7

컨트롤 플레인 시스템의 이름 확인을 제공합니다.

8 9

컴퓨팅 시스템의 이름 확인을 제공합니다.

사용자 프로비저닝 클러스터의 DNS PTR 레코드 구성 예

다음 예제 BIND 영역 파일은 사용자 프로비저닝 클러스터의 역방향 이름 확인을 위한 샘플 PTR 레코드를 보여줍니다.

예 13.5. 역방향 레코드의 샘플 DNS 영역 데이터베이스

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
```

```
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR master0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR master1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR master2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com. 8
;
;EOF
```

1

Kubernetes API의 역방향 DNS 확인을 제공합니다. PTR 레코드는 API 로드 밸런서의 레코드 이름을 참조합니다.

2

Kubernetes API의 역방향 DNS 확인을 제공합니다. PTR 레코드는 API 로드 밸런서의 레코드 이름을 참조하며 내부 클러스터 통신에 사용됩니다.

3

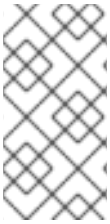
부트스트랩 시스템의 역방향 DNS 확인을 제공합니다.

4 5 6

컨트롤 플레인 시스템의 역방향 DNS 확인을 제공합니다.

7 8

컴퓨팅 시스템의 역방향 DNS 확인을 제공합니다.



참고

OpenShift Container Platform 애플리케이션 와일드카드에는 PTR 레코드가 필요하지 않습니다.

13.3.4.12. 사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구사항

OpenShift Container Platform을 설치하기 전에 API 및 애플리케이션 인그레스 로드 밸런싱 인프라를 프로비저닝해야 합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 API 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

참고

RHEL(Red Hat Enterprise Linux) 인스턴스를 사용하여 API 및 애플리케이션 인그레스 로드 밸런서를 배포하려면 RHEL 서브스크립션을 별도로 구매해야 합니다.

로드 밸런서 인프라는 다음 요구 사항을 충족해야 합니다.

1. API 로드 밸런서: 플랫폼과 상호 작용하고 플랫폼을 구성할 수 있도록 사용자(인간과 시스템 모두)에게 공통 끝점을 제공합니다. 다음 조건을 설정합니다.

   - Layer 4 로드 밸런싱 전용입니다. 이를 Raw TCP, SSL Passthrough 또는 SSL Bridge 모드라고 합니다. SSL Bridge 모드를 사용하는 경우, API 경로에 대해 SNI(Server Name Indication, 서버 이름 표시)를 활성화해야 합니다.

   - 스테이트리스 로드 밸런싱 알고리즘입니다. 옵션은 로드 밸런서 구현에 따라 달라집니다.

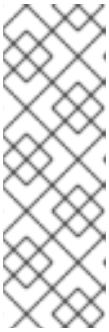
참고

API 로드 밸런서가 제대로 작동하기 위해 세션 지속성이 필요하지 않습니다.

로드 밸런서의 전면과 후면 모두에서 다음 포트를 구성하십시오.

표 13.22. API 로드 밸런서

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
6443	부트스트랩 및 컨트롤 플레인. 부트스트랩 시스템이 클러스터 컨트롤 플레인을 초기화한 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다. API 서버 상태 검사 프로브에 대한 **/readyz** 끝점을 구성해야 합니다.	X	X	Kubernetes API 서버
22623	부트스트랩 및 컨트롤 플레인. 부트스트랩 시스템이 클러스터 컨트롤 플레인을 초기화한 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다.	X		시스템 구성 서버



참고

API 서버가 /readyz 엔드포인트를 해제하는 시점부터 풀에서 API 서버 인스턴스가 제거되는 시점까지 시간이 30초를 넘지 않도록 로드 밸런서를 구성해야 합니다. /readyz가 오류를 반환하거나 정상 상태가 된 후 정해진 시간 안에 끝점이 제거 또는 추가되어야 합니다. 5초 또는 10초의 프로빙 주기(두 번의 성공적인 요청은 정상 상태, 세 번의 요청은 비정상 상태)는 충분한 테스트를 거친 값입니다.

2. 애플리케이션 인그레스 로드 밸런서: 클러스터 외부에서 유입되는 애플리케이션 트래픽에 대한 인그래스 포인트를 제공합니다. 다음 조건을 설정합니다.

- Layer 4 로드 밸런싱 전용입니다. 이를 Raw TCP, SSL Passthrough 또는 SSL Bridge 모드라고 합니다. SSL Bridge 모드를 사용하는 경우 인그레스 경로에 대해 SNI(Server Name Indication, 서버 이름 표시)를 활성화해야 합니다.

- 사용 가능한 옵션과 플랫폼에서 호스팅되는 애플리케이션 유형에 따라 연결 기반 또는 세션 기반 지속성이 권장됩니다.

작은 정보

애플리케이션 인그레스 로드 밸런서에서 클라이언트의 실제 IP 주소를 확인할 수 있는 경우 소스 IP 기반 세션 지속성을 활성화하면 엔드 투 엔드 TLS 암호화를 사용하는 애플리케이션의 성능을 향상시킬 수 있습니다.

로드 밸런서의 전면과 후면 모두에서 다음 포트를 구성하십시오.

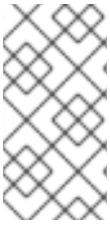
표 13.23. 애플리케이션 인그레스 로드 밸런서

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
443	기본적으로 인그레스 컨트롤러 pod, 컴퓨팅 또는 작업자를 실행하는 시스템입니다.	X	X	HTTPS 트래픽
80	기본적으로 인그레스 컨트롤러 pod, 컴퓨팅 또는 작업자를 실행하는 시스템입니다.	X	X	HTTP 트래픽

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
1936	기본적으로 Ingress 컨트롤러 Pod를 실행하는 작업자 노드입니다. 수신 상태 점검 프로브에 대해 **/healthz/ready** 끝점을 구성해야 합니다.	X	X	HTTP 트래픽

참고

컴퓨팅 노드가 0인 3-노드 클러스터를 배포하는 경우 Ingress 컨트롤러 Pod는 컨트롤 플레인 노드에서 실행됩니다. 3-노드 클러스터 배포에서 HTTP 및 HTTPS 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다.

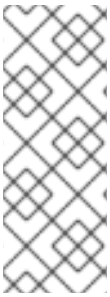


참고

인그레스 라우터에 대한 작업 구성이 OpenShift Container Platform 클러스터에 필요합니다. 컨트롤 플레인 초기화 후 인그레스 라우터를 설정해야 합니다.

13.3.4.12.1. 사용자 프로비저닝 클러스터의 로드 밸런서 구성 예

이 섹션에서는 사용자 프로비저닝 클러스터의 로드 밸런싱 요구 사항을 충족하는 API 및 애플리케이션 수신 로드 밸런서 구성 예를 제공합니다. 샘플은 HAProxy 로드 밸런서에 대한 **/etc/haproxy/haproxy.cfg** 구성입니다. 이 예제에서는 하나의 로드 밸런싱 솔루션을 선택하기 위한 조언을 제공하는 것을 목적으로 하지 않습니다.



참고

이 예제에서는 Kubernetes API 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 API 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

예 13.6. API 및 애플리케이션 인그레스 로드 밸런서 구성 샘플

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
```

```
    option                  redispatch
    retries                 3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s
    maxconn                 3000
  frontend stats
    bind *:1936
    mode            http
    log             global
    maxconn 10
    stats enable
    stats hide-version
    stats refresh 30s
    stats show-node
    stats show-desc Stats for ocp4 cluster (1)
    stats auth admin:ocp4
    stats uri /stats
  listen api-server-6443 (2)
    bind *:6443
    mode tcp
    server bootstrap bootstrap.ocp4.example.com:6443 check inter 1s backup (3)
    server master0 master0.ocp4.example.com:6443 check inter 1s
    server master1 master1.ocp4.example.com:6443 check inter 1s
    server master2 master2.ocp4.example.com:6443 check inter 1s
  listen machine-config-server-22623 (4)
    bind *:22623
    mode tcp
    server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup (5)
    server master0 master0.ocp4.example.com:22623 check inter 1s
    server master1 master1.ocp4.example.com:22623 check inter 1s
    server master2 master2.ocp4.example.com:22623 check inter 1s
  listen ingress-router-443 (6)
    bind *:443
    mode tcp
    balance source
    server worker0 worker0.ocp4.example.com:443 check inter 1s
    server worker1 worker1.ocp4.example.com:443 check inter 1s
  listen ingress-router-80 (7)
    bind *:80
    mode tcp
    balance source
    server worker0 worker0.ocp4.example.com:80 check inter 1s
    server worker1 worker1.ocp4.example.com:80 check inter 1s
```

(1) 이 예에서 클러스터 이름은 **ocp4** 입니다.

(2)

3 5

부트스트랩 항목은 **OpenShift Container Platform** 클러스터 설치 전에 있어야 하며 부트스트랩 프로세스가 완료된 후 제거해야 합니다.

4

포트 **22623**은 머신 구성 서버 트래픽을 처리하고 컨트롤 플레인 시스템을 가리킵니다.

6

포트 **443**은 **HTTPS** 트래픽을 처리하고 **Ingress** 컨트롤러 **Pod**를 실행하는 시스템을 가리킵니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

7

포트 **80**은 **HTTP** 트래픽을 처리하고 **Ingress** 컨트롤러 **Pod**를 실행하는 머신을 가리킵니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

참고

컴퓨팅 노드가 **0**인 **3-**노드 클러스터를 배포하는 경우 **Ingress** 컨트롤러 **Pod**는 컨트롤 플레인 노드에서 실행됩니다. **3-**노드 클러스터 배포에서 **HTTP** 및 **HTTPS** 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다.

작은 정보

HAProxy를 로드 밸런서로 사용하는 경우 **HAProxy** 노드에서 **netstat -nltupe**를 실행하여 **haproxy** 프로세스가 포트 **6443, 22623, 443** 및 **80**에서 수신 대기 중인지 확인할 수 있습니다.

참고

HAProxy를 로드 밸런서로 사용하고 **SELinux**가 **enforcing**으로 설정된 경우 **HAProxy** 서비스가 **setsebool -P haproxy_connect_any=1**을 실행하여 구성된 **TCP** 포트에 바인딩할 수 있는지 확인해야 합니다.

13.3.5. 사용자 프로비저닝 인프라 준비

사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치하기 전에 기본 인프라를 준비해야 합니다.

이 섹션에서는 OpenShift Container Platform 설치를 준비하기 위해 클러스터 인프라를 설정하는 데 필요한 높은 수준의 단계에 대해 자세히 설명합니다. 여기에는 클러스터 노드에 대한 IP 네트워킹 및 네트워크 연결 구성, 방화벽을 통해 필요한 포트 활성화, 필수 DNS 및 로드 밸런싱 인프라를 설정하는 작업이 포함됩니다.

준비 후 클러스터 인프라는 *사용자 프로비저닝 인프라가 있는 클러스터의 요구 사항* 섹션에 설명된 요구 사항을 충족해야 합니다.

사전 요구 사항

- OpenShift Container Platform 4.x 테스트된 통합 페이지를 검토했습니다.
- *사용자 프로비저닝 인프라가 있는 클러스터의 요구 사항* 섹션에 자세히 설명된 인프라 요구 사항을 검토했습니다.

프로세스

1. DHCP를 사용하여 클러스터 노드에 IP 네트워킹 구성을 제공하는 경우 DHCP 서비스를 구성합니다.
   a. 노드의 영구 IP 주소를 DHCP 서버 구성에 추가합니다. 구성에서 관련 네트워크 인터페이스의 MAC 주소를 각 노드의 의도한 IP 주소와 일치시킵니다.
   b. DHCP를 사용하여 클러스터 시스템의 IP 주소 지정을 구성하는 경우 시스템은 DHCP를 통해 DNS 서버 정보도 가져옵니다. DHCP 서버 구성을 통해 클러스터 노드에서 사용하는 영구 DNS 서버 주소를 정의합니다.

참고

DHCP 서비스를 사용하지 않는 경우 RHCOS 설치 시 IP 네트워킹 구성과 DNS 서버의 주소를 노드에 제공해야 합니다. ISO 이미지에서 설치하는 경우 부팅 인수로 전달할 수 있습니다. 고정 IP 프로비저닝 *및 고급 네트워킹 옵션에 대한 자세한 내용은 RHCOS 설치 및 OpenShift Container Platform 부트스트랩 프로세스 시작* 섹션을 참조하십시오.

c.

DHCP 서버 구성에 클러스터 노드의 호스트 이름을 정의합니다. 호스트 이름 고려 사항에 대한 자세한 내용은 ***DHCP****를 통해 클러스터 노드 호스트 이름 설정* 섹션을 참조하십시오.

참고

DHCP 서비스를 사용하지 않는 경우 클러스터 노드는 역방향 **DNS** 조회를 통해 호스트 이름을 가져옵니다.

2.

Red Hat Enterprise Linux CoreOS (RHCOS)의 빠른 설치 또는 **RHCOS(Red Hat Enterprise Linux CoreOS)**의 전체 설치를 수행하려면 선택합니다. 전체 설치의 경우 **Ignition** 파일을 제공하고 클러스터 노드에 이미지를 설치하기 위해 **HTTP** 또는 **HTTPS** 서버를 설정해야 합니다. 빠른 설치의 경우 **HTTP** 또는 **HTTPS** 서버가 필요하지 않지만 **DHCP** 서버가 필요합니다. **"빠른 설치: RHCOS (Red Hat Enterprise Linux CoreOS)** 머신 생성 섹션**"** 및 **"전체 설치: Red Hat Enterprise Linux CoreOS (RHCOS)** 머신 생성**"** 섹션을 참조하십시오.

3.

네트워크 인프라가 클러스터 구성 요소 간 필수 네트워크 연결을 제공하는지 확인합니다. 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 인프라 섹션의 네트워킹 요구 사항* 섹션을 참조하십시오.

4.

OpenShift Container Platform 클러스터 구성 요소가 통신하는 데 필요한 포트를 활성화하도록 방화벽을 구성합니다. 필요한 포트에 대한 자세한 내용은 *사용자 프로비저닝 인프라 섹션의 네트워킹 요구 사항* 섹션을 참조하십시오.

5.

클러스터에 필요한 **DNS** 인프라를 설정합니다.

a.

Kubernetes API, 애플리케이션 와일드카드, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템의 **DNS** 이름 확인을 구성합니다.

b.

Kubernetes API, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템에 대한 역방향 **DNS** 확인을 구성합니다.

OpenShift Container Platform DNS 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 **DNS** 요구 사항* 섹션을 참조하십시오.

6.

DNS 구성을 확인합니다.

a.

설치 노드에서 **Kubernetes API**의 레코드 이름, 와일드카드 경로 및 클러스터 노드에 대해 **DNS** 조회를 실행합니다. 응답의 **IP** 주소가 올바른 구성 요소에 해당하는지 확인합니다.

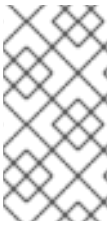
b.

설치 노드에서 로드 밸런서 및 클러스터 노드의 **IP** 주소에 대해 역방향 **DNS** 조회를 실행합니다. 응답의 레코드 이름이 올바른 구성 요소에 해당하는지 확인합니다.

자세한 **DNS** 검증 단계는 *사용자 프로비저닝 인프라에 대한 DNS 확인* 섹션을 참조하십시오.

7.

필요한 **API** 및 애플리케이션 수신 로드 밸런싱 인프라를 프로비저닝합니다. 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구 사항* 섹션을 참조하십시오.

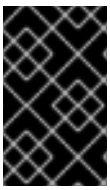


참고

일부 로드 밸런싱 솔루션에는 로드 밸런싱을 초기화하기 전에 클러스터 노드의 **DNS** 이름을 확인해야 합니다.

13.3.6. 사용자 프로비저닝 인프라에 대한 DNS 확인 검증

사용자 프로비저닝 인프라에 **OpenShift Container Platform**을 설치하기 전에 **DNS** 구성을 확인할 수 있습니다.



중요

클러스터를 설치하기 전에 이 섹션에 설명된 검증 단계를 성공해야 합니다.

사전 요구 사항

- 사용자 프로비저닝 인프라에 필요한 **DNS** 레코드를 구성했습니다.

프로세스

1.

설치 노드에서 **Kubernetes API**의 레코드 이름, 와일드카드 경로 및 클러스터 노드에 대해 **DNS** 조회를 실행합니다. 응답에 포함된 **IP** 주소가 올바른 구성 요소에 해당하는지 확인합니다.

a.

Kubernetes API 레코드 이름을 조회합니다. 결과가 **API** 로드 밸런서의 **IP** 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
```

1

<nameserver_ip>를 네임서버의 IP 주소로, <cluster_name>을 클러스터 이름으로, <base_domain>을 기본 도메인 이름으로 바꿉니다.

출력 예

```
api.ocp4.example.com.  0 IN A 192.168.1.5
```

b.

Kubernetes 내부 API 레코드 이름을 조회합니다. 결과가 API 로드 밸런서의 IP 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
```

출력 예

```
api-int.ocp4.example.com.  0 IN A 192.168.1.5
```

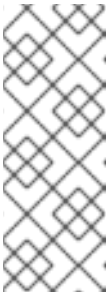
c.

예제 *.apps.<cluster_name>.<base_domain>을 테스트합니다. DNS 와일드카드를 조회합니다. 모든 애플리케이션 와일드카드 조회는 애플리케이션 인그레스 로드 밸런서의 IP 주소로 확인되어야 합니다.

```
$ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.
<base_domain>
```

출력 예

```
random.apps.ocp4.example.com.  0 IN A 192.168.1.5
```



참고

예제 출력에서는 Kubernetes API 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 API 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

random 항목을 다른 와일드카드 값으로 교체할 수 있습니다. 예를 들어 OpenShift Container Platform 콘솔의 경로를 쿼리할 수 있습니다.

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

출력 예

```
console-openshift-console.apps.ocp4.example.com. 0 IN A 192.168.1.5
```

d.

부트스트랩 DNS 레코드 이름에 대해 조회를 실행합니다. 결과가 부트스트랩 노드의 IP 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.
<base_domain>
```

출력 예

```
bootstrap.ocp4.example.com.  0 IN A 192.168.1.96
```

e.

이 방법을 사용하여 컨트롤 플레인 및 컴퓨팅 노드의 DNS 레코드 이름에 대해 조회를 수행합니다. 결과가 각 노드의 IP 주소에 해당하는지 확인합니다.

2. 설치 노드에서 로드 밸런서 및 클러스터 노드의 IP 주소에 대해 역방향 DNS 조회를 실행합니다. 응답에 포함된 레코드 이름이 올바른 구성 요소에 해당하는지 확인합니다.

a. API 로드 밸런서의 IP 주소에 대해 역방향 조회를 수행합니다. 응답에 Kubernetes API 및 Kubernetes 내부 API의 레코드 이름이 포함되어 있는지 확인합니다.

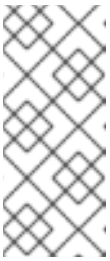
```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

출력 예

```
5.1.168.192.in-addr.arpa. 0 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 0 IN PTR api.ocp4.example.com. 2
```

1 Kubernetes 내부 API의 레코드 이름을 제공합니다.

2 Kubernetes API의 레코드 이름을 제공합니다.



참고

OpenShift Container Platform 애플리케이션 와일드카드에는 PTR 레코드가 필요하지 않습니다. 애플리케이션 인그레스 로드 밸런서의 IP 주소에 대한 역방향 DNS 확인에는 유효성 검사 단계가 필요하지 않습니다.

b. 부트스트랩 노드의 IP 주소에 대해 역방향 조회를 수행합니다. 결과가 부트스트랩 노드의 DNS 레코드 이름을 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

출력 예

```
96.1.168.192.in-addr.arpa. 0 IN PTR bootstrap.ocp4.example.com.
```

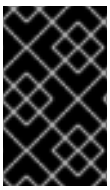
c. 이 방법을 사용하여 컨트롤 플레인 및 컴퓨팅 노드의 IP 주소에 대해 역방향 조회를 수행합니다. 결과가 각 노드의 DNS 레코드 이름과 일치하는지 확인합니다.

13.3.7. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 core 사용자의 ~/.ssh/authorized_keys 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 core로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. ./openshift-install gather 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.



중요

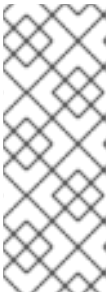
재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: ~/.ssh/id_ed25519)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 '~/.ssh 디렉터리에 있는지 확인하십시오.



참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 ed25519 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 rsa 또는 ecdsa 알고리즘을 사용하는 키를 생성합니다.

2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 ~/.ssh/id_ed25519.pub 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 ./openshift-install gather 명령을 사용하려는 경우 필요합니다.

참고

일부 배포에서는 ~/.ssh/id_rsa 및 ~/.ssh/id_dsa와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. ssh-agent 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. ssh-agent에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: ~/.ssh/id_ed25519).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

13.3.8. 수동으로 설치 구성 파일 만들기

OpenShift Container Platform을 사용자가 프로비저닝한 설치의 경우 설치 구성 파일을 수동으로 생성합니다.

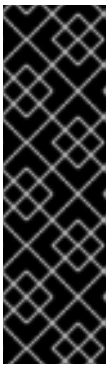
사전 요구 사항

- 로컬 시스템에 설치 프로그램에 제공할 SSH 공개 키가 있습니다. 키는 디버깅 및 재해 복구를 위해 클러스터 노드에 대한 SSH 인증에 사용됩니다.
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿이 있습니다.

프로세스

1. 필요한 설치 자산을 저장할 설치 디렉터리를 만듭니다.

```
$ mkdir <installation_directory>
```



중요

디렉터리를 만들어야 합니다. 부트스트랩 **X.509** 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 **OpenShift Container Platform** 버전에서 설치 파일을 복사할 때는 주의하십시오.

2. 샘플 **install-config.yaml** 파일 템플릿을 사용자 지정하여 **<installation_directory>**에 저장합니다.

참고

이 설정 파일의 이름을 **install-config.yaml**로 지정해야 합니다.

참고

일부 플랫폼 유형의 경우 대체로 **./openshift-install create install-config --dir <installation_directory>** 를 실행하여 **install-config.yaml** 파일을 생성할 수 있습니다. 프롬프트에서 클러스터 구성에 대한 세부 정보를 제공할 수 있습니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.

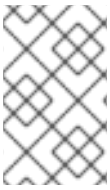
중요

install-config.yaml 파일은 설치 과정의 다음 단계에서 사용됩니다. 이 시점에서 이를 백업해야 합니다.

13.3.8.1. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 환경에 대한 세부 정보를 설명하는 사용자

지정 **install-config.yaml** 설치 구성 파일을 제공합니다.



참고

설치한 후에는 **install-config.yaml** 파일에서 이러한 매개변수를 수정할 수 없습니다.

중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

13.3.8.1.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 13.24. 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체

매개변수	설명	값
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	소문자, 하이픈(**-**), 마침표(**.**)로 구성되는 문자열(예: **dev**)입니다.
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

13.3.8.1.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 13.25. 네트워크 매개변수

매개변수	설명	값

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체 **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.
networking.clusterNetwork	Pod의 IP 주소 블록입니다. 기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된 **cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16

매개변수	설명	값
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다. 여러 IP 커널 인수를 지정하는 경우 **machineNetwork.cidr** 값은 기본 네트워크의 CIDR이어야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16** **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

13.3.8.1.3. 선택적 구성 매개변수

선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 13.26. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **s390x** (기본값)입니다.	문자열

매개변수	설명	값
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은 **3**입니다.
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **s390x** (기본값)입니다.	문자열

매개변수	설명	값
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 기본값은 **External**입니다. 이 필드를 **Internal** 로 설정하는 것은 클라우드 이외의 플랫폼에서는 지원되지 않습니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

13.3.8.2. IBM Z의 샘플 install-config.yaml 파일

install-config.yaml 파일을 사용자 지정하여 **OpenShift Container Platform** 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.

```
apiVersion: v1
baseDomain: example.com 1
compute: 2
- hyperthreading: Enabled 3
  name: worker
  replicas: 0 4
  architecture : s390x
controlPlane: 5
  hyperthreading: Enabled 6
  name: master
  replicas: 3 7
  architecture : s390x
metadata:
  name: test 8
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 9
    hostPrefix: 23 10
  networkType: OpenShiftSDN
  serviceNetwork: 11
  - 172.30.0.0/16
platform:
  none: {} 12
fips: false 13
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email":
"you@example.com"}}}' 14
sshKey: 'ssh-ed25519 AAAA...' 15
additionalTrustBundle: | 16
```

```
  -----BEGIN CERTIFICATE-----
  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
imageContentSources: 17
- mirrors:
  - <local_repository>/ocp4/openshift4
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_repository>/ocp4/openshift4
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

1

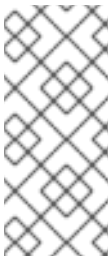
클러스터의 기본 도메인입니다. 모든 **DNS** 레코드는 이 기본 도메인의 하위 도메인이어야 하며 클러스터 이름을 포함해야 합니다.

2 5

controlPlane 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

3 6

동시 멀티스레딩(**SMT**) 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 **SMT**가 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. **SMT**를 비활성화하는 경우 모든 클러스터 머신에서 이를 비활성화해야 합니다. 여기에는 컨트롤 플레인과 컴퓨팅 머신이 모두 포함됩니다.



참고

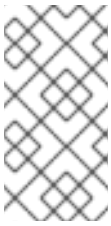
SMT(동시 멀티 스레딩)는 기본적으로 활성화되어 있습니다. **OpenShift Container Platform** 노드에서 **SMT**를 사용할 수 없는 경우 **hyperthreading** 매개변수가 적용되지 않습니다.

중요

OpenShift Container Platform 노드 또는 **install-config.yaml** 파일에서 **hyperthreading**을 비활성화하는 경우 용량 계획에서 머신 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.

4

사용자 프로비저닝 인프라에 **OpenShift Container Platform**을 설치할 때 이 값을 **0**으로 설정해야 합니다. 설치 프로그램에서 제공하는 설치에서 매개 변수는 클러스터가 생성 및 관리하는 컴퓨팅 머신 수를 제어합니다. 사용자 프로비저닝 설치에서는 클러스터 설치를 완료하기 전에 컴퓨팅 시스템을 수동으로 배포해야 합니다.



참고

3-노드 클러스터를 설치하는 경우 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템을 설치할 때 컴퓨팅 머신을 배포하지 마십시오.

7

클러스터에 추가하는 컨트롤 플레인 시스템의 수입니다. 클러스터에서 이 값을 클러스터의 etcd 끝점 수로 사용하므로 이 값은 배포하는 컨트롤 플레인 시스템의 수와 일치해야 합니다.

8

DNS 레코드에 지정한 클러스터 이름입니다.

9

Pod IP 주소가 할당되는 IP 주소 블록입니다. 이 블록은 기존 물리적 네트워크와 중복되지 않아야합니다. 이러한 IP 주소는 Pod 네트워크에 사용됩니다. 외부 네트워크에서 Pod에 액세스해야 하는 경우, 트래픽을 관리하도록 로드 밸런서와 라우터를 설정해야 합니다.

참고

클래스 E CIDR 범위는 향후 사용을 위해 예약되어 있습니다. 클래스 E CIDR 범위를 사용하려면 네트워킹 환경에서 클래스 E CIDR 범위 내에서 IP 주소를 수락하는지 확인해야 합니다.

10

개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 hostPrefix를 23으로 설정하면 지정된 cidr 이외 /23 서브넷이 각 노드에 할당되어 510(2^(32 - 23) - 2) Pod IP 주소가 허용됩니다. 외부 네트워크에서 노드에 액세스해야 하는 경우 트래픽을 관리하도록 로드 밸런서와 라우터를 구성합니다.

11

서비스 IP 주소에 사용할 IP 주소 풀입니다. IP 주소 풀은 하나만 입력할 수 있습니다. 이 블록은 기존 물리적 네트워크와 중복되지 않아야합니다. 외부 네트워크에서 서비스에 액세스해야 하는 경우, 트래픽을 관리하도록 로드 밸런서와 라우터를 구성합니다.

12

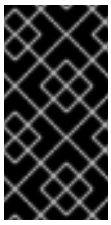
플랫폼을 none으로 설정해야 합니다. IBM Z 인프라에 대한 추가 플랫폼 구성 변수는 지정할 수 없습니다.

중요

플랫폼 유형으로 설치된 클러스터는 **Machine API**로 컴퓨팅 머신 관리와 같은 일부 기능을 사용할 수 없습니다. 이 제한은 클러스터에 연결된 컴퓨팅 시스템이 일반적으로 기능을 지원하는 플랫폼에 설치된 경우에도 적용됩니다. 이 매개변수는 설치 후 변경할 수 없습니다.

13

FIPS 모드 활성화 또는 비활성화 여부입니다. 기본적으로 **FIPS** 모드는 비활성화됩니다. **FIPS** 모드가 활성화되면 **OpenShift Container Platform**이 실행되는 **RHCOS(Red Hat Enterprise Linux CoreOS)** 시스템에서 기본 **Kubernetes** 암호화 제품군은 우회하고 **RHCOS**와 함께 제공되는 암호화 모듈을 대신 사용합니다.



중요

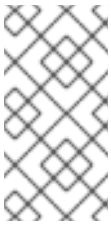
FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 **OpenShift Container Platform** 배포에서만 지원됩니다.

14

<local_registry>는 미러 레지스트리가 해당 내용을 제공하는 데 사용하는 레지스트리 도메인 이름과 포트(선택사항)를 지정합니다. 예: **registry.example.com** 또는 **registry.example.com:5000 <credentials>**는 미러 레지스트리의 **base64** 인코딩 사용자 이름과 암호를 지정합니다.

15

RHCOS(Red Hat Enterprise Linux CoreOS)의 **core** 사용자에 대한 **SSH** 공용 키입니다.



참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 **OpenShift Container Platform** 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 **SSH** 키를 지정합니다.

16

additionalTrustBundle 매개변수와 값을 추가합니다. 값은 미러 레지스트리에 사용한 인증서 파일의 내용이어야 하며, 신뢰할 수 있는 기존 인증 기관 또는 미러 레지스트리에 대해 생성한 자체 서명 인증서일 수 있습니다.

17

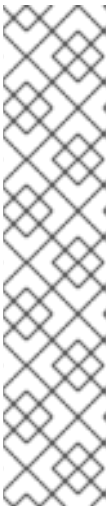
명령 출력에서 **imageContentSources** 섹션을 제공하여 리포지토리를 미러링하십시오.

13.3.8.3. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 **HTTP** 또는 **HTTPS** 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 **OpenShift Container Platform** 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 **API**에 대한 호출을 포함하여 모든 클러스터 발신(**Egress**) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.



참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), **Google Cloud Platform (GCP)**, **Microsoft Azure** 및 **Red Hat OpenStack Platform (RHOSP)**에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
```

```
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1

클러스터 외부에서 **HTTP** 연결을 구축하는 데 사용할 프록시 **URL**입니다. **URL** 스키마는 **http**여야 합니다.

2

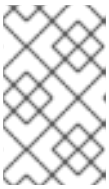
클러스터 외부에서 **HTTPS** 연결을 구축하는 데 사용할 프록시 **URL**입니다.

3

대상 도메인 이름, **IP** 주소 또는 프록시에서 제외할 기타 네트워크 **CIDR**로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. ***를 사용하여 모든 대상에 대해 프록시를 바이패스합니다.

4

이 값을 제공하면 설치 프로그램에서 추가 **CA** 인증서를 보유할 **openshift -config** 네임스페이스에 **user-ca- bundle** 이라는 구성 맵을 생성합니다. **additionalTrustBundle** 및 하나 이상의 프록시 설정을 제공하는 경우 프록시 오브젝트는 **trustedCA** 필드의 **user-ca-bundle** 구성 맵을 참조하도록 구성됩니다. 그러면 **Cluster Network Operator**에서 **trustedCA** 매개변수에 대해 지정된 콘텐츠를 **RHCOS** 신뢰 번들과 병합하는 **trusted-ca-bundle** 구성 맵을 생성합니다. 프록시의 **ID** 인증서를 **RHCOS** 트러스트 번들에 있는 기관에서 서명하지 않은 경우 **additionalTrustBundle** 필드가 있어야 합니다.



참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 **OpenShift Container Platform**을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 **nil**이 됩니다.

참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

13.3.8.4. 3개의 노드 클러스터 구성

필요한 경우 세 개의 컨트롤 플레인 시스템으로 구성된 베어 메탈 클러스터에 제로 컴퓨팅 머신을 배포할 수 있습니다. 이를 통해 클러스터 관리자와 개발자들이 테스트, 개발, 프로덕션에 사용할 수 있는 소형화되고 리소스 효율이 높은 클러스터를 제공합니다.

3-노드 **OpenShift Container Platform** 환경에서 세 개의 컨트롤 플레인 머신을 예약할 수 있습니다. 즉, 애플리케이션 워크로드가 해당 플랫폼에서 실행되도록 예약됩니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.

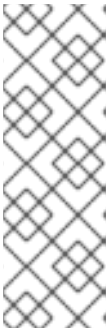
절차

- **install-config.yaml** 파일에서 다음 **compute** 스탠자에 표시된 대로 컴퓨팅 복제본 수가 **0**으로 설정되어 있는지 확인합니다.

```
compute:
- name: worker
  platform: {}
  replicas: 0
```

참고

배포 중인 컴퓨팅 머신 수에 관계없이 사용자 프로비저닝 인프라에 **OpenShift Container Platform**을 설치할 때 컴퓨팅 머신의 **replicas** 매개변수 값을 **0**으로 설정해야 합니다. 설치 프로그램에서 제공하는 설치에서 매개 변수는 클러스터가 생성 및 관리하는 컴퓨팅 머신 수를 제어합니다. 이 설정은 컴퓨팅 시스템이 수동으로 배포되는 사용자 프로비저닝 설치에는 적용되지 않습니다.



참고

컨트롤 플레인 노드의 기본 리소스는 **vCPU 6**개와 **21GB**입니다. **3**개의 컨트롤 플레인 노드의 경우 메모리 **+ vCPU**는 최소 **5-**노드 클러스터와 동등합니다. **SMT2**가 활성화된 **IFL 3**개와 함께 각각 **120GB** 디스크에 설치된 **3**개의 노드를 백업해야 합니다. 테스트된 최소 설정은 각 컨트롤 플레인 노드에 대해 **120GB** 디스크에서 **3**개의 **vCPU**와 **10GB**입니다.

3-노드 클러스터 설치의 경우 다음 단계를 따르십시오.

- 컴퓨팅 노드가 **0**인 **3-**노드 클러스터를 배포하는 경우 **Ingress** 컨트롤러 **Pod**는 컨트롤 플레인 노드에서 실행됩니다. **3-**노드 클러스터 배포에서 **HTTP** 및 **HTTPS** 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다. 자세한 내용은 *사용자 프로비저닝 인프라 섹션에 대한 로드 밸런싱 요구 사항* 섹션을 참조하십시오.

- 다음 절차에서 **Kubernetes** 매니페스트 파일을 생성할 때 **<installation_directory>/manifests/cluster-scheduler-02-config.yml** 파일의 **mastersSchedulable** 매개변수가 **true**로 설정되어 있는지 확인합니다. 이렇게 하면 애플리케이션 워크로드를 컨트롤 플레인 노드에서 실행할 수 있습니다.

- **RHCOS(Red Hat Enterprise Linux CoreOS)** 시스템을 생성할 때 컴퓨팅 노드를 배포하지 마십시오.

13.3.9. CNO(Cluster Network Operator) 구성

클러스터 네트워크의 구성은 **CNO(Cluster Network Operator)** 구성의 일부로 지정되며 **cluster**라는 이름의 **CR**(사용자 정의 리소스) 오브젝트에 저장됩니다. **CR**은 **operator.openshift.io API** 그룹에서 **Network API**의 필드를 지정합니다.

CNO 구성은 **Network.config.openshift.io API** 그룹의 **Network API**에서 클러스터 설치 중에 다음 필드를 상속하며 이러한 필드는 변경할 수 없습니다.

clusterNetwork

Pod IP 주소가 할당되는 **IP** 주소 풀입니다.

serviceNetwork

서비스를 위한 **IP** 주소 풀입니다.

defaultNetwork.type

OpenShift SDN 또는 OVN-Kubernetes와 같은 클러스터 네트워크 공급자입니다.

cluster라는 CNO 오브젝트에서 defaultNetwork 오브젝트의 필드를 설정하여 클러스터의 클러스터 네트워크 공급자 구성을 지정할 수 있습니다.

13.3.9.1. CNO(Cluster Network Operator) 구성 오브젝트

CNO(Cluster Network Operator)의 필드는 다음 표에 설명되어 있습니다.

표 13.27. CNO(Cluster Network Operator) 구성 오브젝트

필드	유형	설명
metadata.name	**string**	CNO 개체 이름입니다. 이 이름은 항상 **cluster**입니다.

spec.clusterNetwork	**array**	Pod IP 주소가 할당되는 IP 주소 블록과 클러스터의 각 개별 노드에 할당된 서브넷 접두사 길이를 지정하는 목록입니다. 예를 들면 다음과 같습니다.
		`spec:`
		`  clusterNetwork:`
		`  - cidr: 10.128.0.0/19`
		`  hostPrefix: 23`
		`  - cidr: 10.128.32.0/19`
		`  hostPrefix: 23`
		매니페스트를 생성하기 전에 **install-config.yaml** 파일에서만 이 필드를 사용자 지정할 수 있습니다. 값은 매니페스트 파일에서 읽기 전용입니다.

spec.serviceNetwork	**array**	서비스의 IP 주소 블록입니다. OpenShift SDN 및 OVN-Kubernetes CNI(Container Network Interface) 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다. 예를 들면 다음과 같습니다.
		`spec:`
		`  serviceNetwork:`
		`  - 172.30.0.0/14`
		매니페스트를 생성하기 전에 **install-config.yaml** 파일에서만 이 필드를 사용자 지정할 수 있습니다. 값은 매니페스트 파일에서 읽기 전용입니다.

spec.defaultNetwork	**object**	클러스터 네트워크의 CNI(Container Network Interface) 클러스터 네트워크 공급자를 구성합니다.

필드	유형	설명
spec.kubeProxyConfig	**object**	이 개체의 필드는 kube-proxy 구성을 지정합니다. OVN-Kubernetes 클러스터 네트워크 공급자를 사용하는 경우 kube-proxy 구성이 적용되지 않습니다.

defaultNetwork 오브젝트 구성

defaultNetwork 오브젝트의 값은 다음 표에 정의되어 있습니다.

표 13.28. defaultNetwork 오브젝트

필드	유형	설명
type	**string**	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, 클러스터 네트워크 공급자가 설치 중에 선택됩니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다. **참고** OpenShift Container Platform은 기본적으로 OpenShift SDN CNI(Container Network Interface) 클러스터 네트워크 공급자를 사용합니다.
openshiftSDNConfig	**object**	이 오브젝트는 OpenShift SDN 클러스터 네트워크 공급자에만 유효합니다.
ovnKubernetesConfig	**object**	이 오브젝트는 OVN-Kubernetes 클러스터 네트워크 공급자에만 유효합니다.

OpenShift SDN CNI 네트워크 공급자에 대한 구성

다음 표에서는 OpenShift SDN Container Network Interface (CNI) 클러스터 네트워크 공급자의 구성 필드를 설명합니다.

표 13.29. openshiftSDNConfig 오브젝트

필드	유형	설명

필드	유형	설명
mode	**string**	OpenShift SDN의 네트워크 격리 모드를 구성합니다. 기본값은 **NetworkPolicy**입니다. **Multitenant** 및 **Subnet** 값은 OpenShift Container Platform 3.x 버전과의 역호환할 수 있지만 권장되지는 않습니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
mtu	**integer**	VXLAN 오버레이 네트워크의 최대 전송 단위(MTU)입니다. 이는 기본 네트워크 인터페이스의 MTU를 기준으로 자동 탐지됩니다. 일반적으로 감지된 MTU를 재정의할 필요는 없습니다. 자동 감지 값이 예상 밖인 경우 노드의 기본 네트워크 인터페이스의 MTU가 올바른지 확인합니다. 이 옵션을 사용하여 노드의 기본 네트워크 인터페이스의 MTU 값을 변경할 수 없습니다. 클러스터에 다른 노드에 대한 다른 MTU 값이 필요한 경우 이 값을 클러스터의 가장 낮은 MTU 값보다 **50** 미만으로 설정해야 합니다. 예를 들어 클러스터의 일부 노드에는 **9001**의 MTU가 있고 일부에는 **1500**의 MTU가 있는 경우 이 값을 **1450**으로 설정해야 합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
vxlanPort	**integer**	모든 VXLAN 패킷에 사용할 포트입니다. 기본값은 **4789**입니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다. 기존 노드가 다른 VXLAN 네트워크에 속하는 가상 환경에서 실행 중인 경우에는 기본값을 변경해야 할 수도 있습니다. 예를 들어 VMware NSX-T 위에서 OpenShift SDN 오버레이를 실행할 때 두 SDN이 동일한 기본 VXLAN 포트 번호를 사용하므로 VXLAN의 대체 포트를 선택해야 합니다. AWS(Amazon Web Services)에서는 포트 **9000**과 포트 **9999** 사이에서 VXLAN의 대체 포트를 선택할 수 있습니다.

OpenShift SDN 구성 예

```
defaultNetwork:
  type: OpenShiftSDN
  openshiftSDNConfig:
    mode: NetworkPolicy
    mtu: 1450
    vxlanPort: 4789
```

OVN-Kubernetes CNI 클러스터 네트워크 공급자에 대한 구성

다음 표에서는 **OVN-Kubernetes CNI** 클러스터 네트워크 공급자의 구성 필드를 설명합니다.

표 13.30. ovnKubernetesConfig object

필드	유형	설명
mtu	**integer**	Geneve(Generic Network Virtualization Encapsulation) 오버레이 네트워크의 MTU(최대 전송 단위)입니다. 이는 기본 네트워크 인터페이스의 MTU를 기준으로 자동 탐지됩니다. 일반적으로 감지된 MTU를 재정의할 필요는 없습니다. 자동 감지 값이 예상 밖인 경우 노드의 기본 네트워크 인터페이스의 MTU가 올바른지 확인합니다. 이 옵션을 사용하여 노드의 기본 네트워크 인터페이스의 MTU 값을 변경할 수 없습니다. 클러스터에 다른 노드에 대한 다른 MTU 값이 필요한 경우, 이 값을 클러스터의 가장 낮은 MTU 값보다 **100** 미만으로 설정해야 합니다. 예를 들어, 클러스터의 일부 노드에 **9001**의 MTU가 있고 일부에는 **1500**의 MTU가 있는 경우 이 값을**1400**으로 설정해야 합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
genevePort	**integer**	모든 Geneve 패킷에 사용할 포트입니다. 기본값은 **6081**입니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
policyAuditConfig	**object**	네트워크 정책 감사 로깅을 사용자 정의할 구성 오브젝트를 지정합니다. 설정되지 않으면 기본값 감사 로그 설정이 사용됩니다.

표 13.31. policyAuditConfig object

필드	유형	설명
rateLimit	integer	노드당 1초마다 생성할 최대 메시지 수입니다. 기본값은 초당 **20** 개의 메시지입니다.
maxFileSize	integer	감사 로그의 최대 크기(바이트)입니다. 기본값은 **50000000 또는 50** MB입니다.
대상	string	다음 추가 감사 로그 대상 중 하나입니다. **libc** 호스트에서 journald 프로세스의 libc **syslog()** 함수입니다. **udp:<host>:<port>** syslog 서버입니다. **<host>:<port>**를 syslog 서버의 호스트 및 포트로 바꿉니다. **unix:<file>** **<file>**로 지정된 Unix Domain Socket 파일입니다. **null** 감사 로그를 추가 대상으로 보내지 마십시오.

필드	유형	설명
syslogFacility	string	RFC5424에 정의된 **kern**과 같은 syslog 기능입니다. 기본값은 **local0**입니다.

OVN-Kubernetes 구성 예

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
```

kubeProxyConfig 오브젝트 구성

kubeProxyConfig 오브젝트의 값은 다음 표에 정의되어 있습니다.

표 13.32. kubeProxyConfig object

필드	유형	설명
iptablesSyncPeriod	**string**	**iptables** 규칙의 새로 고침 간격입니다. 기본값은 **30s**입니다. 유효 접미사로 **s**, **m**, **h**가 있으며, 자세한 설명은 Go **time** 패키지 문서를 참조하십시오. **참고** OpenShift Container Platform 4.3 이상에서는 성능이 개선되어 더 이상 **iptablesSyncPeriod** 매개변수를 조정할 필요가 없습니다.

proxyArguments.iptables-min-sync-period	**array**	**iptables** 규칙을 새로 고치기 전 최소 기간입니다. 이 필드를 통해 새로 고침 간격이 너무 짧지 않도록 조정할 수 있습니다. 유효 접미사로 **s**, **m**, **h**가 있으며, 자세한 설명은 Go **time** 패키지를 참조하십시오. 기본값은 다음과 같습니다.
		
		`kubeProxyConfig:`
		`  proxyArguments:`
		`  iptables-min-sync-period:`
		`  - 0s`

13.3.10. Kubernetes 매니페스트 및 Ignition 설정 파일 생성

일부 클러스터 정의 파일을 수정하고 클러스터 시스템을 수동으로 시작해야 하므로 클러스터가 시스템을 구성하는 데 필요한 Kubernetes 매니페스트 및 Ignition 구성 파일을 사용자가 생성해야 합니다.

설치 구성 파일은 Kubernetes 매니페스트로 변환됩니다. 매니페스트는 나중에 클러스터 머신을 구성하는 데 사용되는 Ignition 구성 파일로 래핑됩니다.

중요

- OpenShift Container Platform 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 node-bootstrapper 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *만료된 컨트롤 플레인 인증서에서 복구* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

참고

매니페스트 및 Ignition 파일을 생성하는 설치 프로그램은 아키텍처에 따라 다르며 클라이언트 이미지 미러에서 얻을 수 있습니다. 설치 프로그램의 Linux 버전은 s390x에서만 실행됩니다. 이 설치 프로그램은 Mac OS 버전으로도 사용할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램을 가져오셨습니다. 제한된 네트워크 설치의 경우, 해당 파일은 미러 호스트에 있습니다.
- install-config.yaml 설치 구성 파일을 생성하셨습니다.

프로세스

1. OpenShift Container Platform 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터에 대한 Kubernetes 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1

<installation_directory>는 사용자가 만든 install-config.yaml 파일이 포함된 설치 디렉터리를 지정합니다.

주의

3 노드 클러스터를 실행 중이면 다음 단계를 건너 뛰어 컨트롤 플레인 노드 일정을 계획할 수 있도록 하십시오.

중요

예약할 수 없는 기본에서 컨트롤 플레인 노드를 구성하면 추가 서브스크립션이 필요합니다. 이는 컨트롤 플레인 노드가 작업자 노드가 되기 때문입니다.

2. <installation_directory>/manifests/cluster-scheduler-02-config.yml Kubernetes 매니페스트 파일의 mastersSchedulable 매개변수가 false로 설정되어 있는지 확인합니다. 이 설정으로 인해 컨트롤 플레인 머신에서 포드가 예약되지 않습니다.

   a. <installation_directory>/manifests/cluster-scheduler-02-config.yml 파일을 엽니다.

   b. mastersSchedulable 매개변수를 찾아서 값을 False로 설정되어 있는지 확인합니다.

   c. 파일을 저장하고 종료합니다.

3. Ignition 구성 파일을 생성하려면 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행합니다.

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1

<installation_directory>는 동일한 설치 디렉터리를 지정합니다.

설치 디렉터리의 부트스트랩, 컨트롤 플레인 및 컴퓨팅 노드에 대한 Ignition 구성 파일이 생성됩니다. kubeadmin-password 및 kubeconfig 파일은 ./<installation_directory>/auth 디렉터리에 생성됩니다.

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

13.3.11. RHCOS 설치 및 OpenShift Container Platform 부트스트랩 프로세스 시작

프로비저닝하는 IBM Z 인프라에 OpenShift Container Platform을 설치하려면 RHCOS(Red Hat Enterprise Linux Core OS)를 RHEL(Red Hat Enterprise Linux) 게스트 가상 시스템으로 설치해야 합니다. RHCOS를 설치할 때 설치 중인 머신 유형에 대해 OpenShift Container Platform 설치 프로그램에서 생성한 Ignition 구성 파일을 제공해야 합니다. 적합한 네트워킹, DNS 및 로드 밸런싱 인프라를 구성한 경우 RHCOS 머신이 재부팅된 후 OpenShift Container Platform 부트스트랩 프로세스가 자동으로 시작됩니다.

사전 패키징된 QEMU COW2(QCOW2) 디스크 이미지를 사용하는 RHCOS의 빠른 트랙 설치를 수행할 수 있습니다. 또는 새 QCOW2 디스크 이미지에서 전체 설치를 수행할 수도 있습니다.

13.3.11.1. 사전 패키징된 QCOW2 디스크 이미지를 사용한 빠른 설치

RHCOS(Red Hat Enterprise Linux CoreOS)의 빠른 설치에서 머신을 생성하고 사전 패키징된 RHCOS(Red Hat Enterprise Linux CoreOS) QEMU COW2(QCOW2) 디스크 이미지를 가져오려면 다음 단계를 완료합니다.

사전 요구 사항

- 이 절차에서 RHEL KVM 호스트라고 하는 KVM을 사용하여 RHEL 8.4를 실행하는 하나 이상의 LPAR.
- RHEL KVM 호스트에 KVM/QEMU 하이퍼바이저가 설치되어 있어야 합니다.

- 노드의 호스트 이름 및 역방향 조회를 수행할 수 있는 **DNS(Domain name server)**입니다.
- **IP** 주소를 제공하는 **DHCP** 서버입니다.

프로세스

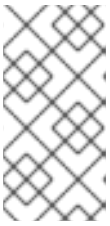
1. **Red Hat Customer Portal**의 제품 다운로드 페이지 또는 **RHCOS** 이미지 미러 페이지에서 **RHEL QCOW2(QEMU copy-on-write)** 디스크 이미지 파일을 받습니다.

중요

RHCOS 이미지는 **OpenShift Container Platform** 릴리스에 따라 변경되지 않을 수 있습니다. 설치하는 **OpenShift Container Platform** 버전과 같거나 그 이하의 버전 중 가장 최신 버전의 이미지를 다운로드해야 합니다. 다음 프로세스에 설명된 적절한 **RHCOS QCOW2** 이미지만 사용합니다.

2. **QCOW2** 디스크 이미지 및 **Ignition** 파일을 **RHEL KVM** 호스트의 공통 디렉터리에 다운로드합니다.

예: /var/lib/libvirt/images



참고

Ignition 파일은 **OpenShift Container Platform** 설치 프로그램에서 생성됩니다.

3. 각 **KVM** 게스트 노드에 대해 **QCOW2** 디스크 이미지 백업 파일을 사용하여 새 디스크 이미지를 생성합니다.

```
$ qemu-img create -f qcow2 -F qcow2 -b /var/lib/libvirt/images/{source_rhcos_qemu}
/var/lib/libvirt/images/{vmname}.qcow2 {size}
```

4. **Ignition** 파일 및 새 디스크 이미지를 사용하여 새 **KVM** 게스트 노드를 생성합니다.

```
$ virt-install --noautoconsole \
  --connect qemu:///system \
  --name {vn_name} \
```

```
    --memory {memory} \
    --vcpus {vcpus} \
    --disk {disk} \
    --import \
    --network network={network},mac={mac} \
    --disk path=
{ign_file},format=raw,readonly=on,serial=ignition,startup_policy=optional
```

13.3.11.2. 새 QCOW2 디스크 이미지에 전체 설치

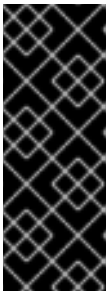
새 QCOW2(QCOW2) 디스크 이미지에 전체 설치로 머신을 생성하려면 다음 단계를 완료합니다.

사전 요구 사항

- 이 절차에서 RHEL KVM 호스트라고 하는 KVM을 사용하여 RHEL 8.4를 실행하는 하나 이상의 LPAR.
- RHEL KVM 호스트에 KVM/QEMU 하이퍼바이저가 설치되어 있어야 합니다.
- 노드의 호스트 이름 및 역방향 조회를 수행할 수 있는 DNS(Domain name server)입니다.
- HTTP 또는 HTTPS 서버가 설정됩니다.

프로세스

1. Red Hat Customer Portal의 Product Downloads 페이지 또는 RHCOS image mirror 페이지에서RHEL kernel, initramfs, rootfs 파일을 받으십시오.



중요

RHCOS 이미지는 OpenShift Container Platform 릴리스에 따라 변경되지 않을 수 있습니다. 설치하는 OpenShift Container Platform 버전과 같거나 그 이하의 버전 중 가장 최신 버전의 이미지를 다운로드해야 합니다. 다음 프로세스에 설명된 적절한 RHCOS QCOW2 이미지만 사용합니다.

OpenShift Container Platform 버전 번호가 파일 이름에 포함됩니다. 다음 예와 유사합니다.

- kernel: rhcos-<version>-live-kernel-<architecture>
- initramfs: rhcos-<version>-live-initramfs.<architecture>.img
- rootfs: rhcos-<version>-live-rootfs.<architecture>.img

2. virt-install을 시작하기 전에 다운로드한 RHEL 라이브 커널, initramfs 및 rootfs 및 Ignition 파일을 HTTP 또는 HTTPS 서버로 이동합니다.

참고

Ignition 파일은 OpenShift Container Platform 설치 프로그램에서 생성됩니다.

3. RHEL 커널, initramfs 및 Ignition 파일, 새 디스크 이미지, 수정된 매개 변수 인수를 사용하여 새 KVM 게스트 노드를 만듭니다.

- --location에 대해 HTTP 또는 HTTPS 서버의 kernel/initrd 위치를 지정합니다.
- coreos.inst.ignition_url=의 경우 시스템 역할의 Ignition 파일을 지정합니다. bootstrap.ign, master.ign 또는 worker.ign을 사용하십시오. HTTP 및 HTTPS 프로토콜만 지원됩니다.
- coreos.live.rootfs_url=의 경우 부팅 중인 커널 및 initramfs와 일치하는 rootfs 아티팩트를 지정합니다. HTTP 및 HTTPS 프로토콜만 지원됩니다.

```
$ virt-install \
  --connect qemu:///system \
  --name {vn_name} \
  --vcpus {vcpus} \
  --memory {memory_mb} \
  --disk {vn_name}.qcow2,size={image_size| default(10,true)} \
  --network network={virt_network_parm} \
  --boot hd \
  --location {media_location},kernel={rhcos_kernel},initrd={rhcos_initrd} \
  --extra-args "rd.neednet=1 coreos.inst=yes coreos.inst.install_dev=vda
coreos.live.rootfs_url={rhcos_liveos} ip={ip}::{default_gateway}:
{subnet_mask_length}:{vn_name}:enc1:none:{MTU} nameserver={dns}
```

```
coreos.inst.ignition_url={rhcos_ign}" \
  --noautoconsole \
  --wait
```

13.3.11.3. 고급 RHCOS 설치 참조

여기서는 RHCOS(Red Hat Enterprise Linux CoreOS) 수동 설치 프로세스를 수정하는 데 사용할 수 있는 네트워킹 구성 및 기타 고급 옵션에 대해 설명합니다. 다음 표에서는 RHCOS 라이브 설치 프로그램 및 coreos-installer 명령과 함께 사용할 수있는 커널 인수 및 명령 줄 옵션에 대해 설명합니다.

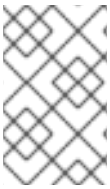
13.3.11.3.1. ISO 설치를 위한 네트워킹 옵션

ISO 이미지에서 RHCOS를 설치하는 경우, 해당 이미지를 부팅할 때 수동으로 커널 인수를 추가하여 노드의 네트워킹을 구성할 수 있습니다. 네트워킹 인수를 지정하지 않으면 RHCOS에서 Ignition 구성 파일을 가져오는 데 네트워킹이 필요함을 감지하면 initramfs에서 DHCP가 활성화됩니다.

중요

네트워킹 인수를 수동으로 추가할 때 initramfs에서 네트워크를 가져오려면 rd.neednet=1 커널 인수도 추가해야 합니다.

다음 정보는 ISO 설치를 위해 RHCOS 노드에서 네트워킹을 구성하는 예제를 제공합니다. 예제에서는 ip= 및 nameserver= 커널 인수를 사용하는 방법을 설명합니다.



참고

커널 인수를 추가할 때 순서가 중요합니다(ip = 및 nameserver=).

이는 시스템 부팅 중에 dracut 툴로 전달되는 네트워킹 옵션입니다. dracut에서 지원하는 네트워킹 옵션에 대한 자세한 내용은 dracut.cmdline 메뉴얼 페이지를 참조하십시오.

다음 예제는 ISO 설치를 위한 네트워킹 옵션입니다.

DHCP 또는 고정 IP 주소 구성

IP 주소를 구성하려면 DHCP(ip=dhcp)를 사용하거나 개별 고정 IP 주소(ip=<host_ip>)를 설정합니다. 정적 IP를 설정하는 경우 각 노드에서 DNS 서버 IP 주소 (nameserver=<dns_ip>)를 확인합니다. 다음 예제에서는 다음을 설정합니다.

- 노드의 IP 주소는 10.10.10.2로 설정됩니다.
- 게이트웨이 주소는 10.10.10.254로 설정됩니다.
- 넷마스크는 255.255.255.0으로
- core0.example.com에 대한 호스트 이름
- 4.4.4.41의 DNS 서버 주소
- auto-configuration 값을 none으로 설정합니다. IP 네트워킹이 정적으로 구성되면 자동 구성이 필요하지 않습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
nameserver=4.4.4.41
```

참고

DHCP를 사용하여 RHCOS 시스템의 IP 주소 지정을 구성하는 경우 시스템은 DHCP를 통해 DNS 서버 정보도 가져옵니다. DHCP 기반 배포의 경우 DHCP 서버 구성을 통해 RHCOS 노드에서 사용할 DNS 서버 주소를 정의할 수 있습니다.

고정 호스트 이름 없이 IP 주소 구성

정적 호스트 이름을 할당하지 않고 IP 주소를 구성할 수 있습니다. 사용자가 정적 호스트 이름을 설정하지 않으면 역방향 DNS 조회에 의해 자동으로 선택됩니다. 정적 호스트 이름이 없는 IP 주소를 구성하려면 다음 예제를 참조합니다.

- 노드의 IP 주소는 10.10.10.2로 설정됩니다.
- 게이트웨이 주소는 10.10.10.254로 설정됩니다.
- 넷마스크는 255.255.255.0으로

- 4.4.4.41의 DNS 서버 주소

- auto-configuration 값을 none으로 설정합니다. IP 네트워킹이 정적으로 구성되면 자동 구성이 필요하지 않습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0::enp1s0:none
nameserver=4.4.4.41
```

여러 네트워크 인터페이스 지정

여러 ip= 항목을 설정하여 여러 네트워크 인터페이스를 지정할 수 있습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

기본 게이트웨이 및 경로 구성

선택 사항: rd.route= 값을 설정하여 추가 네트워크에 대한 경로를 구성할 수 있습니다.

참고

하나 이상의 네트워크를 구성할 때 하나의 기본 게이트웨이가 필요합니다. 추가 네트워크 게이트웨이가 기본 네트워크 게이트웨이와 다른 경우 기본 게이트웨이가 기본 네트워크 게이트웨이어야 합니다.

- 다음 명령을 실행하여 기본 게이트웨이를 구성합니다.

```
ip=::10.10.10.254::::
```

- 다음 명령을 입력하여 추가 네트워크의 경로를 구성합니다.

```
rd.route=20.20.20.0/24:20.20.20.254:enp2s0
```

단일 인터페이스에서 DHCP 비활성화

두 개 이상의 네트워크 인터페이스가 있고 하나의 인터페이스만 사용되는 경우와 같이 단일 인터페이스에서 DHCP를 비활성화할 수 있습니다. 이 예에서 enp1s0 인터페이스에는 정적 네트워킹 구성이 있으며 enp2s0 용으로 DHCP가 사용되지 않습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=::::core0.example.com:enp2s0:none
```

DHCP 및 고정 IP 구성 결합

다음과 같이 시스템의 DHCP 및 고정 IP 구성을 여러 네트워크 인터페이스와 결합할 수 있습니다.

```
ip=enp1s0:dhcp
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

개별 인터페이스에서 VLAN 구성

선택 사항: vlan= 매개변수를 사용하여 개별 인터페이스에서 VLAN을 구성할 수 있습니다.

- 네트워크 인터페이스에서 VLAN을 구성하고 고정 IP 주소를 사용하려면 다음 명령을 실행합니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0.100:none
vlan=enp2s0.100:enp2s0
```

- 네트워크 인터페이스에서 VLAN을 구성하고 DHCP를 사용하려면 다음 명령을 실행합니다.

```
ip=enp2s0.100:dhcp
vlan=enp2s0.100:enp2s0
```

여러 DNS 서버 제공

각 서버에 nameserver= 항목을 추가하여 여러 DNS 서버를 제공할 수 있습니다. 예를 들면 다음과 같습니다.

```
nameserver=1.1.1.1
nameserver=8.8.8.8
```

13.3.12. 부트스트랩 프로세스가 완료될 때까지 대기 중

OpenShift Container Platform 부트스트랩 프로세스는 클러스터 노드가 먼저 디스크에 설치된 영구 RHCOS 환경으로 부팅된 후에 시작됩니다. Ignition 구성 파일을 통해 제공되는 구성 정보는 부트스트랩 프로세스를 초기화하고 머신에 OpenShift Container Platform을 설치하는 데 사용됩니다. 부트스트랩 프로세스가 완료될 때까지 기다려야 합니다.

사전 요구 사항

- 클러스터에 대한 Ignition 구성 파일이 생성되어 있습니다.
- 적합한 네트워크, DNS 및 로드 밸런싱 인프라가 구성되어 있습니다.
- 설치 프로그램을 받아서 클러스터의 Ignition 구성 파일을 생성했습니다.
- 클러스터 머신에 RHCOS를 설치하고 OpenShift Container Platform 설치 프로그램에서 생성된 Ignition 구성 파일을 제공했습니다.

프로세스

1. 부트스트랩 프로세스를 모니터링합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

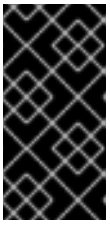
2 다른 설치 세부 사항을 보려면 info 대신 warn, debug 또는 error를 지정합니다.

출력 예

```
INFO Waiting up to 30m0s for the Kubernetes API at
https://api.test.example.com:6443...
INFO API v1.22.1 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

이 명령은 Kubernetes API 서버가 컨트롤 플레인 시스템에서 부트스트랩되었다는 신호를 보낼 때 성공합니다.

2. 부트스트랩 프로세스가 완료된 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다.



중요

이 시점에 로드 밸런서에서 부트스트랩 시스템을 제거해야 합니다. 부트스트랩 머신 자체를 제거하거나 다시 포맷할 수도 있습니다.

13.3.13. CLI를 사용하여 클러스터에 로그인

클러스터 kubeconfig 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. kubeconfig 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- oc CLI를 설치했습니다.

프로세스

1. kubeadmin 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 oc 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

13.3.14. 머신의 인증서 서명 요청 승인

클러스터에 시스템을 추가하면 추가한 시스템별로 보류 중인 인증서 서명 요청(CSR)이 두 개씩 생성됩니다. 이러한 CSR이 승인되었는지 확인해야 하며, 필요한 경우 이를 직접 승인해야 합니다. 클라이언트 요청을 먼저 승인한 다음 서버 요청을 승인해야 합니다.

사전 요구 사항

- 클러스터에 시스템을 추가했습니다.

절차

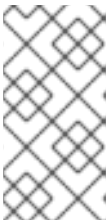
1. 클러스터가 시스템을 인식하는지 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0  Ready    master  63m  v1.22.1
master-1  Ready    master  63m  v1.22.1
master-2  Ready    master  64m  v1.22.1
```

출력에 생성된 모든 시스템이 나열됩니다.



참고

이전 출력에는 일부 CSR이 승인될 때까지 컴퓨팅 노드(작업자 노드라고도 함)가 포함되지 않을 수 있습니다.

2. 보류 중인 **CSR**을 검토하고 클러스터에 추가한 각 시스템에 대해 **Pending** 또는 **Approved** 상태의 클라이언트 및 서버 요청이 표시되는지 확인합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE    REQUESTOR                                                   CONDITION
csr-8b2br   15m    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

예에서는 두 시스템이 클러스터에 참여하고 있습니다. 목록에는 승인된 **CSR**이 더 많이 나타날 수도 있습니다.

3. **CSR**이 승인되지 않은 경우, 추가된 시스템에 대한 모든 보류 중인 **CSR**이 **Pending** 상태로 전환된 후 클러스터 시스템의 **CSR**을 승인합니다.

참고

CSR은 교체 주기가 자동으로 만료되므로 클러스터에 시스템을 추가한 후 **1** 시간 이내에 **CSR**을 승인하십시오. 한 시간 내에 승인하지 않으면 인증서가 교체되고 각 노드에 대해 두 개 이상의 인증서가 표시됩니다. 이러한 인증서를 모두 승인해야 합니다. 클라이언트 **CSR**이 승인되면 **Kubelet**은 인증서에 대한 보조 **CSR**을 생성하므로 수동 승인이 필요합니다. 그러면 **Kubelet**에서 동일한 매개변수를 사용하여 새 인증서를 요청하는 경우 인증서 갱신 요청은 **machine-approver**에 의해 자동으로 승인됩니다.

참고

베어 메탈 및 기타 사용자 프로비저닝 인프라와 같이 머신 **API**를 사용하도록 활성화되지 않는 플랫폼에서 실행되는 클러스터의 경우 **CSR(Kubelet service Certificate Request)**을 자동으로 승인하는 방법을 구현해야 합니다. 요청이 승인되지 않으면 **API** 서버가 **kubelet**에 연결될 때 서비스 인증서가 필요하므로 **oc exec, oc rsh, oc logs** 명령을 성공적으로 수행할 수 없습니다. **Kubelet** 엔드 포인트에 연결하는 모든 작업을 수행하려면 이 인증서 승인이 필요합니다. 이 방법은 새 **CSR**을 감시하고 **CSR**이 **system:node** 또는 **system:admin** 그룹의 **node-bootstrapper** 서비스 계정에 의해 제출되었는지 확인하고 노드의 **ID**를 확인합니다.

- 개별적으로 승인하려면 유효한 **CSR** 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1

<csr_name>은 현재 **CSR** 목록에 있는 **CSR**의 이름입니다.

- 보류 중인 **CSR**을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

참고

일부 **Operator**는 일부 **CSR**이 승인될 때까지 사용할 수 없습니다.

4. 이제 클라이언트 요청이 승인되었으므로 클러스터에 추가한 각 머신의 서버 요청을 검토해야 합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE    REQUESTOR                                                   CONDITION
csr-bfd72   5m26s  system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
```

```
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 나머지 CSR이 승인되지 않고 Pending 상태인 경우 클러스터 머신의 CSR을 승인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 <csr_name>은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs oc adm certificate approve
```

6. 모든 클라이언트 및 서버 CSR이 승인된 후 머신은 Ready 상태가 됩니다. 다음 명령을 실행하여 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0  Ready    master  73m  v1.22.1
master-1  Ready    master  73m  v1.22.1
master-2  Ready    master  74m  v1.22.1
worker-0  Ready    worker  11m  v1.22.1
worker-1  Ready    worker  11m  v1.22.1
```

참고

머신이 **Ready** 상태로 전환하는 데 서버 **CSR**의 승인 후 몇 분이 걸릴 수 있습니다.

추가 정보

- **CSR**에 대한 자세한 내용은 인증서 서명 요청을 참조하십시오.

13.3.15. Operator의 초기 설정

컨트롤 플레인이 초기화된 후 일부 **Operator**를 즉시 구성하여 모두 사용 가능하도록 해야 합니다.

사전 요구 사항

- 컨트롤 플레인이 초기화되어 있습니다.

프로세스

1. 클러스터 구성 요소가 온라인 상태인지 확인합니다.

```
$ watch -n5 oc get clusteroperators
```

출력 예

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.9.0     True        False         False      19m
baremetal                                  4.9.0     True        False         False      37m
cloud-credential                           4.9.0     True        False         False      40m
cluster-autoscaler                         4.9.0     True        False         False      37m
config-operator                            4.9.0     True        False         False      38m
console                                    4.9.0     True        False         False      26m
csi-snapshot-controller                    4.9.0     True        False         False      37m
dns                                        4.9.0     True        False         False      37m
etcd                                       4.9.0     True        False         False      36m
image-registry                             4.9.0     True        False         False      31m
ingress                                    4.9.0     True        False         False      30m
insights                                   4.9.0     True        False         False      31m
kube-apiserver                             4.9.0     True        False         False      26m
kube-controller-manager                    4.9.0     True        False         False      36m
kube-scheduler                             4.9.0     True        False         False      36m
```

```
kube-storage-version-migrator              4.9.0     True        False         False      37m
machine-api                                4.9.0     True        False         False      29m
machine-approver                           4.9.0     True        False         False      37m
machine-config                             4.9.0     True        False         False      36m
marketplace                                4.9.0     True        False         False      37m
monitoring                                 4.9.0     True        False         False      29m
network                                    4.9.0     True        False         False      38m
node-tuning                                4.9.0     True        False         False      37m
openshift-apiserver                        4.9.0     True        False         False      32m
openshift-controller-manager               4.9.0     True        False         False      30m
openshift-samples                          4.9.0     True        False         False      32m
operator-lifecycle-manager                 4.9.0     True        False         False      37m
operator-lifecycle-manager-catalog         4.9.0     True        False         False      37m
operator-lifecycle-manager-packageserver   4.9.0     True        False         False      32m
service-ca                                 4.9.0     True        False         False      38m
storage                                    4.9.0     True        False         False      37m
```

2. 사용할 수 없는 **Operator**를 구성합니다.

13.3.15.1. 기본 OperatorHub 소스 비활성화

Red Hat 및 커뮤니티 프로젝트에서 제공하는 콘텐츠를 소싱하는 **Operator** 카탈로그는 **OpenShift Container Platform**을 설치하는 동안 기본적으로 **OperatorHub**용으로 구성됩니다. 제한된 네트워크 환경에서는 클러스터 관리자로서 기본 카탈로그를 비활성화해야 합니다.

프로세스

- **OperatorHub** 오브젝트에 **disableAllDefaultSources: true**를 추가하여 기본 카탈로그의 소스를 비활성화합니다.

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

작은 정보

또는 웹 콘솔을 사용하여 카탈로그 소스를 관리할 수 있습니다. 관리 → 클러스터 설정 → 구성 → **OperatorHub** 페이지에서 개별 소스를 생성, 삭제, 비활성화 및 활성화할 수 있는 소스 탭을 클릭합니다.

13.3.15.2. 이미지 레지스트리 스토리지 구성

기본 스토리지를 제공하지 않는 플랫폼에서는 처음에 **Image Registry Operator**를 사용할 수 없습니

다. 설치한 후에 스토리지를 사용하도록 레지스트리를 구성하여 **Registry Operator**를 사용 가능하도록 만들어야 합니다.

프로덕션 클러스터에 필요한 영구 볼륨을 구성하는 과정의 지침이 표시됩니다. 해당하는 경우, 프로덕션 환경 외 클러스터에서만 사용할 수 있는 저장 위치로서 빈 디렉터리를 구성하는 과정의 지침이 표시됩니다.

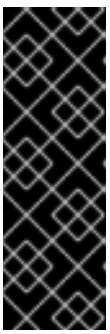
업그레이드 중에 **Recreate** 롤아웃 전략을 사용하여 이미지 레지스트리의 블록 스토리지 유형 사용을 허용하기 위한 추가 지침이 제공됩니다.

13.3.15.2.1. IBM Z용 레지스트리 스토리지 구성

클러스터 관리자는 설치한 후 스토리지를 사용하도록 레지스트리를 구성해야 합니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- **IBM Z**에 클러스터가 있습니다.
- **Red Hat OpenShift Container Storage**와 같이 클러스터용 영구 스토리지 프로비저닝이 있습니다.



중요

OpenShift Container Platform은 복제본이 하나만 있는 경우 이미지 레지스트리 스토리지에 대한 **ReadWriteOnce** 액세스를 지원합니다. **ReadWriteOnce** 액세스에서는 레지스트리가 **Recreate** 롤아웃 전략을 사용해야 합니다. 두 개 이상의 복제본으로 고 가용성을 지원하는 이미지 레지스트리를 배포하려면 **ReadWriteMany** 액세스가 필요합니다.

- **"100Gi"** 용량이 필요합니다.

프로세스

1. 스토리지를 사용하도록 레지스트리를 구성하기 위해 **configs.imageregistry/cluster** 리소스에서 **spec.storage.pvc**를 변경합니다.

참고

공유 스토리지를 사용할 때 보안 설정을 확인하여 외부에서의 액세스를 방지합니다.

2. 레지스트리 **pod**가 없는지 확인합니다.

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

출력 예

```
No resourses found in openshift-image-registry namespace
```

참고

출력에 레지스트리 **Pod**가 있는 경우 이 절차를 계속할 필요가 없습니다.

3. 레지스트리 구성을 확인합니다.

```
$ oc edit configs.imageregistry.operator.openshift.io
```

출력 예

```
storage:
  pvc:
    claim:
```

image-registry-storage PVC의 자동 생성을 허용하도록 **claim** 필드를 비워 둡니다.

4. **clusteroperator** 상태를 확인합니다.

```
$ oc get clusteroperator image-registry
```

출력 예

```
NAME              VERSION   AVAILABLE   PROGRESSING   DEGRADED   SINCE   MESSAGE
image-registry    4.9       True        False         False      6h50m
```

5. 이미지를 빌드 및 푸시할 수 있도록 레지스트리의 관리가 설정되어 있는지 확인하십시오.

- 다음을 실행합니다.

```
$ oc edit configs.imageregistry/cluster
```

다음으로 라인을 변경하십시오.

```
managementState: Removed
```

다음으로 변경

```
managementState: Managed
```

13.3.15.2.2. 프로덕션 환경 외 클러스터에서 이미지 레지스트리의 스토리지 구성

이미지 레지스트리 **Operator**에 대한 스토리지를 구성해야 합니다. 프로덕션 환경 외 클러스터의 경우, 이미지 레지스트리를 빈 디렉터리로 설정할 수 있습니다. 이렇게 하는 경우 레지스트리를 다시 시작하면 모든 이미지가 손실됩니다.

프로세스

- 이미지 레지스트리 스토리지를 빈 디렉터리로 설정하려면 다음을 수행하십시오.

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch
'{"spec":{"storage":{"emptyDir":{}}}}'
```

주의

프로덕션 환경 외 클러스터에 대해서만 이 옵션을 구성하십시오.

Image Registry Operator가 구성 요소를 초기화하기 전에 이 명령을 실행하면 oc patch 명령이 실패하며 다음 오류가 발생합니다.

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster"
not found
```

몇 분 후에 명령을 다시 실행하십시오.

13.3.16. 사용자 프로비저닝 인프라에 설치 완료

Operator 구성을 완료한 후 제공하는 인프라에 클러스터 설치를 완료할 수 있습니다.

사전 요구 사항

- 컨트롤 플레인이 초기화되어 있습니다.
- 초기 Operator 구성을 완료해야 합니다.

프로세스

1. 다음 명령을 사용하여 모든 클러스터 구성 요소가 온라인 상태인지 확인합니다.

```
$ watch -n5 oc get clusteroperators
```

출력 예

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.9.0     True        False         False      19m
baremetal                                  4.9.0     True        False         False      37m
cloud-credential                           4.9.0     True        False         False      40m
cluster-autoscaler                         4.9.0     True        False         False      37m
config-operator                            4.9.0     True        False         False      38m
console                                    4.9.0     True        False         False      26m
csi-snapshot-controller                    4.9.0     True        False         False      37m
dns                                        4.9.0     True        False         False      37m
etcd                                       4.9.0     True        False         False      36m
image-registry                             4.9.0     True        False         False      31m
ingress                                    4.9.0     True        False         False      30m
insights                                   4.9.0     True        False         False      31m
kube-apiserver                             4.9.0     True        False         False      26m
kube-controller-manager                    4.9.0     True        False         False      36m
kube-scheduler                             4.9.0     True        False         False      36m
kube-storage-version-migrator              4.9.0     True        False         False      37m
machine-api                                4.9.0     True        False         False      29m
machine-approver                           4.9.0     True        False         False      37m
machine-config                             4.9.0     True        False         False      36m
marketplace                                4.9.0     True        False         False      37m
monitoring                                 4.9.0     True        False         False      29m
network                                    4.9.0     True        False         False      38m
node-tuning                                4.9.0     True        False         False      37m
openshift-apiserver                        4.9.0     True        False         False      32m
openshift-controller-manager               4.9.0     True        False         False      30m
openshift-samples                          4.9.0     True        False         False      32m
operator-lifecycle-manager                 4.9.0     True        False         False      37m
operator-lifecycle-manager-catalog         4.9.0     True        False         False      37m
operator-lifecycle-manager-packageserver   4.9.0     True        False         False      32m
service-ca                                 4.9.0     True        False         False      38m
storage                                    4.9.0     True        False         False      37m
```

또는 다음 명령은 모든 클러스터를 사용할 수 있을 때 알립니다. 또한 인증 정보를 검색하고 표시합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1

<installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

출력 예

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator가 Kubernetes API 서버에서 OpenShift Container Platform 클러스터 배포를 완료하면 명령이 성공합니다.

중요

- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 node-bootstrapper 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *만료된 컨트롤 플레인 인증서에서 복구* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

2. Kubernetes API 서버가 Pod와 통신하고 있는지 확인합니다.

   a. 모든 Pod 목록을 보려면 다음 명령을 사용하십시오.

```
$ oc get pods --all-namespaces
```

출력 예

```
NAMESPACE                       NAME                                        READY   STATUS
RESTARTS   AGE
openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8
1/1     Running   1          9m
openshift-apiserver             apiserver-67b9g                             1/1     Running   0
```

```
3m
openshift-apiserver            apiserver-ljcmx                         1/1     Running     0
1m
openshift-apiserver            apiserver-z25h4                         1/1     Running     0
2m
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8
1/1     Running     0          5m
...
```

b. 다음 명령을 사용하여 이전 명령의 출력에 나열된 Pod의 로그를 표시합니다.

```
$ oc logs <pod_name> -n <namespace> 1
```

1 이전 명령의 출력에 표시된 대로 Pod 이름과 네임스페이스를 지정합니다.

Pod 로그가 표시되면 Kubernetes API 서버는 클러스터 시스템과 통신할 수 있습니다.

3. FCP(Fibre Channel Protocol)를 사용하는 설치에는 다중 경로를 활성화하기 위해 추가 단계가 필요합니다. 설치 중에 멀티패스를 활성화하지 마십시오.

자세한 내용은 *설치 후 머신 구성 작업* 설명서에서 "RHCOS에서 커널 인수를 사용하여 멀티패스 활성화"를 참조하십시오.

4. 클러스터 등록 페이지에서 클러스터를 등록합니다.

13.3.17. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription

watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

13.3.18. 디버깅 정보 수집

IBM Z에서 OpenShift Container Platform 설치와 관련된 특정 문제를 해결하고 디버깅하는데 도움이 될 수 있는 디버깅 정보를 수집할 수 있습니다.

사전 요구 사항

- oc CLI 도구가 설치되어 있어야 합니다.

프로세스

1. 클러스터에 로그인합니다.

```
$ oc login -u <username>
```

2. 하드웨어 정보를 수집하는 노드에서 디버깅 컨테이너를 시작합니다.

```
$ oc debug node/<nodename>
```

3. /host 파일 시스템으로 변경하고 toolbox를 시작합니다.

```
$ chroot /host
$ toolbox
```

4. dbginfo 데이터를 수집합니다.

```
$ dbginfo.sh
```

5. 다음으로 scp를 사용하여 데이터를 검색할 수 있습니다.

추가 리소스

- How to generate SOSREPORT within OpenShift Container Platform version 4 nodes without SSH 참조.

13.3.19. 다음 단계

- 클러스터를 사용자 지정합니다.

- 클러스터를 설치하는 데 사용한 미러 레지스트리에 신뢰할 수 있는 CA가 있는 경우 추가 신뢰 저장소를 구성하여 클러스터에 추가합니다.

14장. IBM POWER에 설치

14.1. IBM POWER에 설치 준비

14.1.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.

14.1.2. IBM Power에 OpenShift Container Platform을 설치할 방법 선택

다음 방법 중 하나를 사용하여 프로비저닝하는 IBM Power 인프라에 클러스터를 설치할 수 있습니다.

- IBM Power에 클러스터 설치: 프로비저닝하는 IBM Power 인프라에 OpenShift Container Platform을 설치할 수 있습니다.
- 네트워크가 제한된 환경의 IBM Power에 클러스터 설치: 설치 릴리스 콘텐츠의 내부 미러를 사용하여 제한되거나 연결이 끊긴 네트워크에서 프로비저닝하는 IBM Power 인프라에 OpenShift Container Platform을 설치할 수 있습니다. 이 방법을 사용하여 소프트웨어 구성 요소를 받기 위해 활성 인터넷 연결이 필요하지 않은 클러스터를 설치할 수 있습니다. 이 설치 방법을 사용하여 클러스터가 외부 콘텐츠에 대해 조직의 제어 조건을 충족하는 컨테이너 이미지만 사용하도록 할 수 있습니다.

14.2. IBM POWER에 클러스터 설치

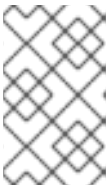
OpenShift Container Platform 버전 4.9에서는 사용자가 제공하는 IBM Power 인프라에 클러스터를 설치할 수 있습니다.

중요

베어 메탈 이외의 플랫폼의 경우 추가 고려 사항이 있습니다. OpenShift Container Platform 클러스터를 설치하기 전에 guidelines for deploying OpenShift Container Platform on non-tested platforms에 있는 내용을 확인하십시오.

14.2.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.
- 설치 프로세스를 시작하기 전에 설치 디렉터리를 정리해야 합니다. 이렇게 하면 설치 프로세스 중에 필요한 설치 파일이 생성되고 업데이트됩니다.
- 클러스터 용 NFS를 사용하여 영구 스토리지를 프로비저닝합니다. 프라이빗 이미지 레지스트리를 배포하려면 스토리지에서 **ReadWriteMany** 액세스 모드를 제공해야 합니다.
- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성해야 합니다.



참고

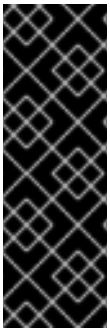
프록시를 구성하는 경우에도 해당 사이트 목록을 검토하십시오.

14.2.2. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 **Telemetry** 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.



중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

14.2.3. 사용자 프로비저닝 인프라를 포함한 클러스터의 시스템 요구사항

사용자 프로비저닝 인프라가 포함된 클러스터의 경우, 필요한 모든 시스템을 배포해야 합니다.

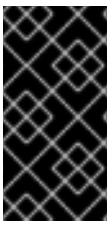
이 섹션에서는 사용자 프로비저닝 인프라에 **OpenShift Container Platform**을 배포해야 하는 요구 사항에 대해 설명합니다.

14.2.3.1. 클러스터 설치에 필요한 시스템

최소 **OpenShift Container Platform** 클러스터에 다음과 같은 호스트가 필요합니다.

표 14.1. 최소 필수 호스트

호스트	설명
임시 부트스트랩 시스템 한 개	컨트롤 플레인 시스템 세 개에 OpenShift Container Platform 클러스터를 배포하기 위한 부트스트랩 시스템이 클러스터에 필요합니다. 클러스터를 설치한 후 부트스트랩 시스템을 제거할 수 있습니다.
컨트롤 플레인 시스템 세 개	컨트롤 플레인 시스템은 컨트롤 플레인을 구성하는 Kubernetes 및 OpenShift Container Platform 서비스를 실행합니다.
두 개 이상의 컴퓨팅 시스템(작업자 시스템이라고도 함).	OpenShift Container Platform 사용자가 요청한 워크로드는 컴퓨팅 머신에서 실행됩니다.



중요

클러스터의 고가용성을 유지하려면 이러한 클러스터 시스템에 대해 별도의 물리적 호스트를 사용하십시오.

부트스트랩, 컨트롤 플레인 시스템은 운영 체제로 **RHCOS (Red Hat Enterprise Linux CoreOS)**를 사용해야 합니다. 그러나 컴퓨팅 머신은 **RHCOS(Red Hat Enterprise Linux CoreOS), RHEL (Red Hat**

Enterprise Linux) 7.9 또는 RHEL 8.4 중에서 선택할 수 있습니다.

RHCOS는 RHEL 8(Red Hat Enterprise Linux)을 기반으로하며 모든 하드웨어 인증 및 요구사항을 모두 이어받습니다. Red Hat Enterprise Linux 기술 기능 및 제한을 참조하십시오.

14.2.3.2. 클러스터 설치를 위한 최소 리소스 요구 사항

각 클러스터 시스템이 다음과 같은 최소 요구사항을 충족해야 합니다.

표 14.2. 최소 리소스 요구사항

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
부트스트랩	RHCOS	2	16GB	100GB	300
컨트롤 플레인	RHCOS	2	16GB	100GB	300
컴퓨팅	RHCOS	2	8GB	100GB	300

1. SMT(동시 멀티스레딩) 또는 하이퍼 스레딩이 활성화되지 않은 경우 하나의 vCPU는 하나의 물리적 코어와 동일합니다. 활성화하면 다음과 같은 공식을 사용하여 해당 비율을 계산합니다. (코어 당 스레드 수 × 코어 수) × 소켓 수 = vCPU 수

2. OpenShift Container Platform 및 Kubernetes는 디스크 성능에 민감하며 특히 10ms p99 fsync 기간이 필요한 컨트롤 플레인 노드의 etcd에 더 빠른 스토리지가 권장됩니다. 많은 클라우드 플랫폼에서 스토리지 크기와 IOPS를 함께 확장되므로 충분한 성능을 얻으려면 스토리지 볼륨을 과도하게 할당해야 할 수 있습니다.

3. 사용자가 프로비저닝한 모든 설치와 마찬가지로 클러스터에서 RHEL 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. RHEL 7 컴퓨팅 머신 사용은 더 이상 사용되지 않으며 향후 OpenShift Container Platform 4 릴리스에서 제거될 예정입니다.

14.2.3.3. 최소 IBM Power 요구사항

다음 IBM 하드웨어에 OpenShift Container Platform 버전 4.9를 설치할 수 있습니다.

- IBM Power8, Power9 또는 Power10 프로세서 기반 시스템

하드웨어 요구 사항

- 여러 PowerVM 서버에서 6개의 IBM Power 베어 메탈 서버 또는 6개의 LPAR

운영 체제 요구 사항

- IBM Power8, Power9 또는 Power10 프로세서 기반 시스템의 인스턴스 하나

IBM Power 인스턴스에서 다음을 설정합니다.

- OpenShift Container Platform 컨트롤 플레인 시스템 용 게스트 가상 머신 세 개
- OpenShift Container Platform 컴퓨팅 머신 용 게스트 가상 머신 두 개
- 임시 OpenShift Container Platform 부트스트랩 시스템용 게스트 가상 머신 1개

IBM Power 게스트 가상 머신용 디스크 스토리지

- vSCSI, NPIV(N-Port ID Virtualization) 또는 SSP(공장 스토리지 풀)를 사용하여 가상 I/O 서버에서 프로비저닝한 스토리지

PowerVM 게스트 가상 머신용 네트워크

- Shared Ethernet Adapter를 사용하여 가상 I/O 서버에서 가상화
- IBM vNIC를 사용하여 가상 I/O 서버에서 가상화

스토리지 / 메인 메모리

- OpenShift Container Platform 컨트롤 플레인 시스템용 100 GB / 16 GB
- OpenShift Container Platform 컴퓨팅 머신용 100GB/8GB

- 임시 OpenShift Container Platform 부트스트랩 머신 용 100GB / 16GB

14.2.3.4. 권장되는 IBM Power 시스템 요구 사항

하드웨어 요구 사항

- 여러 PowerVM 서버에서 6개의 IBM Power 베어 메탈 서버 또는 6개의 LPAR

운영 체제 요구 사항

- IBM Power8, Power9 또는 Power10 프로세서 기반 시스템의 인스턴스 하나

IBM Power 인스턴스에서 다음을 설정합니다.

- OpenShift Container Platform 컨트롤 플레인 시스템 용 게스트 가상 머신 세 개
- OpenShift Container Platform 컴퓨팅 머신 용 게스트 가상 머신 두 개
- 임시 OpenShift Container Platform 부트스트랩 시스템용 게스트 가상 머신 1개

IBM Power 게스트 가상 머신용 디스크 스토리지

- vSCSI, NPIV(N-Port ID Virtualization) 또는 SSP(공장 스토리지 풀)를 사용하여 가상 I/O 서버에서 프로비저닝한 스토리지

PowerVM 게스트 가상 머신용 네트워크

- Shared Ethernet Adapter를 사용하여 가상 I/O 서버에서 가상화
- IBM vNIC를 사용하여 가상 I/O 서버에서 가상화

스토리지 / 메인 메모리

- OpenShift Container Platform 컨트롤 플레인 시스템용 120GB/32GB

- **OpenShift Container Platform** 컴퓨팅 머신용 **120GB/32GB**

- 임시 **OpenShift Container Platform** 부트스트랩 머신용 **120GB/16GB**

14.2.3.5. 인증서 서명 요청 관리

사용자가 프로비저닝하는 인프라를 사용하는 경우 자동 시스템 관리 기능으로 인해 클러스터의 액세스가 제한되므로 설치한 후 클러스터 인증서 서명 요청(**CSR**)을 승인하는 메커니즘을 제공해야 합니다. **kube-controller-manager**는 **kubelet** 클라이언트 **CSR**만 승인합니다. **machine-approver**는 올바른 시스템에서 발행한 요청인지 확인할 수 없기 때문에 **kubelet** 자격 증명을 사용하여 요청하는 서비스 인증서의 유효성을 보장할 수 없습니다. **kubelet** 서빙 인증서 요청의 유효성을 확인하고 요청을 승인하는 방법을 결정하여 구현해야 합니다.

14.2.3.6. 사용자 프로비저닝 인프라에 대한 네트워킹 요구사항

모든 **RHCOS(Red Hat Enterprise Linux CoreOS)** 시스템이 부팅 중에 **Ignition** 구성 파일을 가져오려면 **initramfs**에 네트워킹을 구성해야 합니다.

초기 부팅 과정에서 시스템에 필요한 부팅 옵션을 제공하여 **DHCP** 서버를 통해 설정하거나 정적으로 설정하는 **IP** 주소 구성이 필요합니다. 네트워크 연결이 설정된 후 시스템은 **HTTP** 또는 **HTTPS** 서버에서 **Ignition** 구성 파일을 다운로드합니다. 그런 다음 **Ignition** 구성 파일을 사용하여 각 머신의 정확한 상태를 설정합니다. **Machine Config Operator**는 설치 후 새 인증서 또는 키 적용과 같은 머신에 대한 추가 변경을 완료합니다.

클러스터 시스템의 장기적인 관리를 위해 **DHCP** 서버를 사용하는 것이 좋습니다. **DHCP** 서버가 클러스터 시스템에 영구 **IP** 주소, **DNS** 서버 정보 및 호스트 이름을 제공하도록 구성되었는지 확인합니다.

참고

사용자 프로비저닝 인프라에 **DHCP** 서비스를 사용할 수 없는 경우 **RHCOS** 설치 시 노드에 **IP** 네트워킹 구성과 **DNS** 서버의 주소를 대신 제공할 수 있습니다. **ISO** 이미지에서 설치하는 경우 부팅 인수로 전달할 수 있습니다. 고정 **IP** 프로비저닝 *및 고급 네트워킹 옵션에 대한 자세한 내용은 **RHCOS** 설치 및 **OpenShift Container Platform** 부트스트랩 프로세스 시작* 섹션을 참조하십시오.

Kubernetes API 서버가 클러스터 시스템의 노드 이름을 확인할 수 있어야 합니다. **API** 서버와 작업자 노드가 서로 다른 영역에 있는 경우, **API** 서버가 노드 이름을 확인할 수 있도록 기본 **DNS** 검색 영역을 설정할 수 있습니다. 노드 개체와 모든 **DNS** 요청에서 항상 정규화된 도메인 이름으로 호스트를 가리키는 것도 지원되는 방법입니다

14.2.3.6.1. DHCP를 통해 클러스터 노드의 호스트 이름 설정

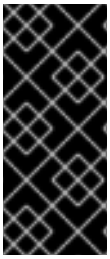
RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 호스트 이름은 NetworkManager를 통해 설정됩니다. 기본적으로 시스템은 DHCP를 통해 호스트 이름을 가져옵니다. DHCP에서 호스트 이름을 제공하지 않거나, 커널 인수를 통해 정적으로 설정하거나 다른 방법을 통해 설정하면 역방향 DNS 조회를 통해 얻을 수 있습니다. 역방향 DNS 조회는 노드에서 네트워크를 초기화한 후 수행되며 확인하는 데 시간이 걸릴 수 있습니다. 다른 시스템 서비스는 이 보다 먼저 시작하여 호스트 이름을 localhost 등으로 감지할 수 있습니다. DHCP를 사용하여 각 클러스터 노드의 호스트 이름을 제공하여 이 문제를 방지할 수 있습니다.

또한 DHCP를 통해 호스트 이름을 설정하면 DNS 분할 수평 구현 환경에서 수동으로 DNS 레코드 이름 구성 오류를 무시할 수 있습니다.

14.2.3.6.2. 네트워크 연결 요구사항

OpenShift Container Platform 클러스터 구성 요소가 통신할 수 있도록 시스템 간 네트워크 연결을 구성해야 합니다. 각 시스템에서 클러스터에 있는 다른 모든 시스템의 호스트 이름을 확인할 수 있어야 합니다.

이 섹션에서는 필요한 포트에 대해 자세히 설명합니다.



중요

연결된 OpenShift Container Platform 환경에서 모든 노드는 플랫폼 컨테이너의 이미지를 가져오고 Red Hat에 원격 측정 데이터를 제공하기 위해 인터넷에 액세스할 수 있어야 합니다.

표 14.3. 모든 시스템 간 통신에 사용되는 포트

프로토콜	포트	설명
ICMP	해당 없음	네트워크 연결성 테스트
TCP	**1936**	메트릭
	9000-**9999**	**9100**-**9101** 포트의 노드 내보내기 및 **9099** 포트의 Cluster Version Operator를 포함한 호스트 수준 서비스.
	10250-**10259**	Kubernetes에서 예약하는 기본 포트
	10256	openshift-sdn

프로토콜	포트	설명
UDP	**4789**	VXLAN 및 Geneve
	6081	VXLAN 및 Geneve
	9000-9999	**9100-9101** 포트의 노드 내보내기를 포함한 호스트 수준 서비스.
	500	IPsec IKE 패킷
	4500	IPsec NAT-T 패킷
TCP/UDP	**30000-32767**	Kubernetes 노드 포트
ESP	해당 없음	IPsec Encapsulating Security Payload (ESP)

표 14.4. 모든 시스템과 컨트롤 플레인 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**6443**	Kubernetes API

표 14.5. 컨트롤 플레인 머신 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**2379-2380**	etcd 서버 및 피어 포트

사용자 프로비저닝 인프라에 대한 NTP 구성

OpenShift Container Platform 클러스터는 기본적으로 공용 **NTP(Network Time Protocol)** 서버를 사용하도록 구성되어 있습니다. 로컬 엔터프라이즈 **NTP** 서버를 사용하거나 클러스터가 연결이 끊긴 네트워크에 배포되는 경우 특정 시간 서버를 사용하도록 클러스터를 구성할 수 있습니다. 자세한 내용은 ***chrony* 타임 서비스 설정** 문서를 참조하십시오.

DHCP 서버가 **NTP** 서버 정보를 제공하는 경우 **RHCOS(Red Hat Enterprise Linux CoreOS)** 시스템의 **chrony** 타임 서비스에서 정보를 읽고 **NTP** 서버와 클럭을 동기화할 수 있습니다.

추가 리소스

- **chrony 타임 서비스 설정**

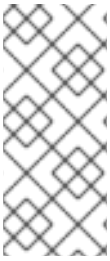
14.2.3.7. 사용자 프로비저닝 DNS 요구사항

OpenShift Container Platform 배포의 경우 다음 구성 요소에 DNS 이름을 확인해야 합니다.

- Kubernetes API
- OpenShift Container Platform 애플리케이션 와일드카드
- 부트스트랩, 컨트롤 플레인 및 컴퓨팅 시스템

Kubernetes API, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템에 대한 역방향 DNS 확인이 필요합니다.

DNS A/AAAA 또는 CNAME 레코드는 이름 확인에 사용되며 PTR 레코드는 역방향 이름 확인에 사용됩니다. RHCOS (Red Hat Enterprise Linux CoreOS)는 DHCP에서 호스트 이름을 제공하지 않는 한 모든 노드의 호스트 이름을 설정할 때 역방향 레코드를 사용하기 때문에 역방향 레코드가 중요합니다. 또한 역방향 레코드는 OpenShift Container Platform이 작동하는 데 필요한 인증서 서명 요청 (CSR)을 생성하는 데 사용됩니다.



참고

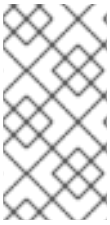
DHCP 서버를 사용하여 각 클러스터 노드에 호스트 이름을 제공하는 것이 좋습니다. 자세한 내용은 *사용자 프로비저닝 인프라 섹션에 대한 DHCP 권장 사항* 섹션을 참조하십시오.

사용자가 프로비저닝한 OpenShift Container Platform 클러스터에 대해 다음 DNS 레코드가 필요하며 설치 전에 있어야 합니다. 각 레코드에서 <cluster_name>은 클러스터 이름이고 <base_domain>은 install-config.yaml 파일에서 지정하는 기반 도메인입니다. 전체 DNS 레코드는 <component>.<cluster_name>.<base_domain> 형식입니다.

표 14.6. 필수 DNS 레코드

구성 요소	레코드	설명
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API 로드 밸런서를 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.

구성 요소	레코드	설명
	api-int.<cluster_name>.<base_domain>.	내부적으로 API 로드 밸런서를 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.  **중요** API 서버는 Kubernetes에 기록된 호스트 이름으로 작업자 노드를 확인할 수 있어야 합니다. API 서버가 노드 이름을 확인할 수 없는 경우 프록시된 API 호출이 실패할 수 있으며 pod에서 로그를 검색할 수 없습니다.
라우트	***.apps.<cluster_name>.<base_domain>.**	애플리케이션 인그레스 로드 밸런서를 참조하는 와일드카드 DNS A/AAA 또는 CNAME 레코드입니다. 애플리케이션 인그레스 로드 밸런서는 Ingress 컨트롤러 Pod를 실행하는 머신을 대상으로 합니다. Ingress 컨트롤러 Pod는 기본적으로 컴퓨팅 머신에서 실행됩니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다. 예를 들어 **console-openshift-console.apps.<cluster_name>.<base_domain>**은 OpenShift Container Platform 콘솔의 와일드카드 경로로 사용됩니다.
부트스트랩 시스템	**bootstrap.<cluster_name>.<base_domain>.**	부트스트랩 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.
컨트롤 플레인 머신	**<master><n>.<cluster_name>.<base_domain>.**	컨트롤 플레인 노드의 각 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.
컴퓨팅 머신	**<worker><n>.<cluster_name>.<base_domain>.**	작업자 노드의 각 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.



참고

OpenShift Container Platform 4.4 이상에서는 DNS 구성에서 etcd 호스트 및 SRV 레코드를 지정할 필요가 없습니다.

작은 정보

dig 명령을 사용하여 이름과 역방향 이름을 확인할 수 있습니다. 자세한 검증 단계는 *사용자 프로비저닝 인프라의 DNS 확인* 섹션을 참조하십시오.

14.2.3.7.1. 사용자 프로비저닝 클러스터의 DNS 구성 예

이 섹션에서는 사용자 프로비저닝 인프라에 OpenShift Container Platform을 배포하기 위한 DNS 요구 사항을 충족하는 A 및 PTR 레코드 구성 샘플을 제공합니다. 샘플은 하나의 DNS 솔루션을 선택하기 위한 조언을 제공하기 위한 것이 아닙니다.

이 예제에서 클러스터 이름은 ocp4이고 기본 도메인은 example.com입니다.

사용자 프로비저닝 클러스터의 DNS A 레코드 구성 예

다음 BIND 영역 파일의 예제에서는 사용자가 프로비저닝한 클러스터의 이름 확인을 위한 샘플 A 레코드를 보여줍니다.

예 14.1. 샘플 DNS 영역 데이터베이스

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
master0.ocp4.example.com. IN A 192.168.1.97 5
master1.ocp4.example.com. IN A 192.168.1.98 6
master2.ocp4.example.com. IN A 192.168.1.99 7
```

```
;
worker0.ocp4.example.com. IN A 192.168.1.11 8
worker1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1

Kubernetes API의 이름 확인을 제공합니다. 레코드는 **API** 로드 밸런서의 **IP** 주소를 나타냅니다.

2

Kubernetes API의 이름 확인을 제공합니다. 레코드는 **API** 로드 밸런서의 **IP** 주소를 참조하며 내부 클러스터 통신에 사용됩니다.

3

와일드카드 경로의 이름 확인을 제공합니다. 레코드는 애플리케이션 인그레스 로드 밸런서의 **IP** 주소를 나타냅니다. 애플리케이션 인그레스 로드 밸런서는 **Ingress** 컨트롤러 **Pod**를 실행하는 머신을 대상으로 합니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

참고

이 예제에서는 **Kubernetes API** 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 **API** 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

4

부트스트랩 시스템의 이름 확인을 제공합니다.

5 6 7

컨트롤 플레인 시스템의 이름 확인을 제공합니다.

8 9

컴퓨팅 시스템의 이름 확인을 제공합니다.

사용자 프로비저닝 클러스터의 **DNS PTR** 레코드 구성 예

다음 예제 BIND 영역 파일은 사용자 프로비저닝 클러스터의 역방향 이름 확인을 위한 샘플 PTR 레코드를 보여줍니다.

예 14.2. 역방향 레코드의 샘플 DNS 영역 데이터베이스

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR master0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR master1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR master2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com. 8
;
;EOF
```

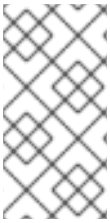
1 Kubernetes API의 역방향 DNS 확인을 제공합니다. PTR 레코드는 API 로드 밸런서의 레코드 이름을 참조합니다.

2 Kubernetes API의 역방향 DNS 확인을 제공합니다. PTR 레코드는 API 로드 밸런서의 레코드 이름을 참조하며 내부 클러스터 통신에 사용됩니다.

3 부트스트랩 시스템의 역방향 DNS 확인을 제공합니다.

4 5 6 컨트롤 플레인 시스템의 역방향 DNS 확인을 제공합니다.

7 8 컴퓨팅 시스템의 역방향 DNS 확인을 제공합니다.

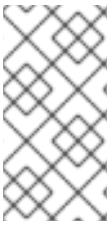


참고

OpenShift Container Platform 애플리케이션 와일드카드에는 **PTR** 레코드가 필요하지 않습니다.

14.2.3.8. 사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구사항

OpenShift Container Platform을 설치하기 전에 **API** 및 애플리케이션 인그레스 로드 밸런싱 인프라를 프로비저닝해야 합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 **API** 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.



참고

RHEL(Red Hat Enterprise Linux) 인스턴스를 사용하여 **API** 및 애플리케이션 인그레스 로드 밸런서를 배포하려면 **RHEL** 서브스크립션을 별도로 구매해야 합니다.

로드 밸런서 인프라는 다음 요구 사항을 충족해야 합니다.

1. **API** 로드 밸런서**:** 플랫폼과 상호 작용하고 플랫폼을 구성할 수 있도록 사용자(인간과 시스템 모두)에게 공통 끝점을 제공합니다. 다음 조건을 설정합니다.

   - **Layer 4** 로드 밸런싱 전용입니다. 이를 **Raw TCP, SSL Passthrough** 또는 **SSL Bridge** 모드라고 합니다. **SSL Bridge** 모드를 사용하는 경우**, API** 경로에 대해 **SNI(Server Name Indication,** 서버 이름 표시)를 활성화해야 합니다.
   - 스테이트리스 로드 밸런싱 알고리즘입니다. 옵션은 로드 밸런서 구현에 따라 달라집니다.

   참고

   API 로드 밸런서가 제대로 작동하기 위해 세션 지속성이 필요하지 않습니다.

   로드 밸런서의 전면과 후면 모두에서 다음 포트를 구성하십시오.

표 14.7. API 로드 밸런서

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
6443	부트스트랩 및 컨트롤 플레인. 부트스트랩 시스템이 클러스터 컨트롤 플레인을 초기화한 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다. API 서버 상태 검사 프로브에 대한 **/readyz** 끝점을 구성해야 합니다.	X	X	Kubernetes API 서버
22623	부트스트랩 및 컨트롤 플레인. 부트스트랩 시스템이 클러스터 컨트롤 플레인을 초기화한 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다.	X		시스템 구성 서버

참고

API 서버가 /readyz 엔드포인트를 해제하는 시점부터 풀에서 API 서버 인스턴스가 제거되는 시점까지 시간이 30초를 넘지 않도록 로드 밸런서를 구성해야 합니다. /readyz가 오류를 반환하거나 정상 상태가 된 후 정해진 시간 안에 끝점이 제거 또는 추가되어야 합니다. 5초 또는 10초의 프로빙 주기(두 번의 성공적인 요청은 정상 상태, 세 번의 요청은 비정상 상태)는 충분한 테스트를 거친 값입니다.

2. **애플리케이션 인그레스 로드 밸런서: 클러스터 외부에서 유입되는 애플리케이션 트래픽에 대한 인그래스 포인트를 제공합니다. 다음 조건을 설정합니다.**

- **Layer 4 로드 밸런싱 전용입니다. 이를 Raw TCP, SSL Passthrough 또는 SSL Bridge 모드라고 합니다. SSL Bridge 모드를 사용하는 경우 인그레스 경로에 대해 SNI(Server Name Indication, 서버 이름 표시)를 활성화해야 합니다.**

- **사용 가능한 옵션과 플랫폼에서 호스팅되는 애플리케이션 유형에 따라 연결 기반 또는 세션 기반 지속성이 권장됩니다.**

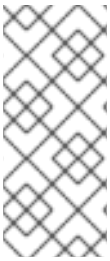
작은 정보

애플리케이션 인그레스 로드 밸런서에서 클라이언트의 실제 IP 주소를 확인할 수 있는 경우 소스 IP 기반 세션 지속성을 활성화하면 엔드 투 엔드 TLS 암호화를 사용하는 애플리케이션의 성능을 향상시킬 수 있습니다.

로드 밸런서의 전면과 후면 모두에서 다음 포트를 구성하십시오.

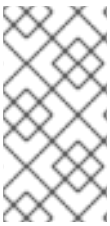
표 14.8. 애플리케이션 인그레스 로드 밸런서

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
443	기본적으로 인그레스 컨트롤러 pod, 컴퓨팅 또는 작업자를 실행하는 시스템입니다.	X	X	HTTPS 트래픽
80	기본적으로 인그레스 컨트롤러 pod, 컴퓨팅 또는 작업자를 실행하는 시스템입니다.	X	X	HTTP 트래픽
1936	기본적으로 Ingress 컨트롤러 Pod를 실행하는 작업자 노드입니다. 수신 상태 점검 프로브에 대해 **/healthz/ready** 끝점을 구성해야 합니다.	X	X	HTTP 트래픽



참고

컴퓨팅 노드가 0인 3-노드 클러스터를 배포하는 경우 Ingress 컨트롤러 Pod는 컨트롤 플레인 노드에서 실행됩니다. 3-노드 클러스터 배포에서 HTTP 및 HTTPS 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다.



참고

인그레스 라우터에 대한 작업 구성이 OpenShift Container Platform 클러스터에 필요합니다. 컨트롤 플레인 초기화 후 인그레스 라우터를 설정해야 합니다.

14.2.3.8.1. 사용자 프로비저닝 클러스터의 로드 밸런서 구성 예

이 섹션에서는 사용자 프로비저닝 클러스터의 로드 밸런싱 요구 사항을 충족하는 API 및 애플리케이션 수신 로드 밸런서 구성 예를 제공합니다. 샘플은 HAProxy 로드 밸런서에 대한 /etc/haproxy/haproxy.cfg 구성입니다. 이 예제에서는 하나의 로드 밸런싱 솔루션을 선택하기 위한 조언을 제공하는 것을 목적으로 하지 않습니다.

참고

이 예제에서는 Kubernetes API 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 API 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

예 14.3. API 및 애플리케이션 인그레스 로드 밸런서 구성 샘플

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
frontend stats
  bind *:1936
  mode            http
  log             global
  maxconn 10
  stats enable
  stats hide-version
  stats refresh 30s
  stats show-node
  stats show-desc Stats for ocp4 cluster (1)
  stats auth admin:ocp4
  stats uri /stats
listen api-server-6443 (2)
  bind *:6443
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:6443 check inter 1s backup (3)
  server master0 master0.ocp4.example.com:6443 check inter 1s
  server master1 master1.ocp4.example.com:6443 check inter 1s
  server master2 master2.ocp4.example.com:6443 check inter 1s
listen machine-config-server-22623 (4)
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup (5)
  server master0 master0.ocp4.example.com:22623 check inter 1s
  server master1 master1.ocp4.example.com:22623 check inter 1s
  server master2 master2.ocp4.example.com:22623 check inter 1s
listen ingress-router-443 (6)
  bind *:443
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:443 check inter 1s
  server worker1 worker1.ocp4.example.com:443 check inter 1s
listen ingress-router-80 (7)
  bind *:80
  mode tcp
```

```
balance source
server worker0 worker0.ocp4.example.com:80 check inter 1s
server worker1 worker1.ocp4.example.com:80 check inter 1s
```

1 이 예에서 클러스터 이름은 **ocp4** 입니다.

2 포트 **6443**은 **Kubernetes API** 트래픽을 처리하고 컨트롤 플레인 시스템을 가리킵니다.

3 5 부트스트랩 항목은 **OpenShift Container Platform** 클러스터 설치 전에 있어야 하며 부트스트랩 프로세스가 완료된 후 제거해야 합니다.

4 포트 **22623**은 머신 구성 서버 트래픽을 처리하고 컨트롤 플레인 시스템을 가리킵니다.

6 포트 **443**은 **HTTPS** 트래픽을 처리하고 **Ingress** 컨트롤러 **Pod**를 실행하는 시스템을 가리킵니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

7 포트 **80**은 **HTTP** 트래픽을 처리하고 **Ingress** 컨트롤러 **Pod**를 실행하는 머신을 가리킵니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

참고

컴퓨팅 노드가 **0**인 **3-**노드 클러스터를 배포하는 경우 **Ingress** 컨트롤러 **Pod**는 컨트롤 플레인 노드에서 실행됩니다. **3-**노드 클러스터 배포에서 **HTTP** 및 **HTTPS** 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다.

작은 정보

HAProxy를 로드 밸런서로 사용하는 경우 **HAProxy** 노드에서 **netstat -nltupe**를 실행하여 **haproxy** 프로세스가 포트 **6443, 22623, 443** 및 **80**에서 수신 대기 중인지 확인할 수 있습니다.

참고

HAProxy를 로드 밸런서로 사용하고 SELinux가 enforcing으로 설정된 경우 HAProxy 서비스가 setsebool -P haproxy_connect_any=1을 실행하여 구성된 TCP 포트에 바인딩할 수 있는지 확인해야 합니다.

14.2.4. 사용자 프로비저닝 인프라 준비

사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치하기 전에 기본 인프라를 준비해야 합니다.

이 섹션에서는 OpenShift Container Platform 설치를 준비하기 위해 클러스터 인프라를 설정하는 데 필요한 높은 수준의 단계에 대해 자세히 설명합니다. 여기에는 클러스터 노드에 대한 IP 네트워킹 및 네트워크 연결 구성, 방화벽을 통해 필요한 포트 활성화, 필수 DNS 및 로드 밸런싱 인프라를 설정하는 작업이 포함됩니다.

준비 후 클러스터 인프라는 *사용자 프로비저닝 인프라가 있는 클러스터의 요구 사항* 섹션에 설명된 요구 사항을 충족해야 합니다.

사전 요구 사항

- OpenShift Container Platform 4.x 테스트된 통합 페이지를 검토했습니다.
- *사용자 프로비저닝 인프라가 있는 클러스터의 요구 사항* 섹션에 자세히 설명된 인프라 요구 사항을 검토했습니다.

프로세스

1. DHCP를 사용하여 클러스터 노드에 IP 네트워킹 구성을 제공하는 경우 DHCP 서비스를 구성합니다.
   a. 노드의 영구 IP 주소를 DHCP 서버 구성에 추가합니다. 구성에서 관련 네트워크 인터페이스의 MAC 주소를 각 노드의 의도한 IP 주소와 일치시킵니다.
   b. DHCP를 사용하여 클러스터 시스템의 IP 주소 지정을 구성하는 경우 시스템은 DHCP를 통해 DNS 서버 정보도 가져옵니다. DHCP 서버 구성을 통해 클러스터 노드에서 사용하는 영구 DNS 서버 주소를 정의합니다.

참고

DHCP 서비스를 사용하지 않는 경우 **RHCOS** 설치 시 **IP** 네트워킹 구성과 **DNS** 서버의 주소를 노드에 제공해야 합니다. **ISO** 이미지에서 설치하는 경우 부팅 인수로 전달할 수 있습니다. 고정 **IP** 프로비저닝 *및 고급 네트워킹 옵션에 대한 자세한 내용은 **RHCOS** 설치 및 **OpenShift Container Platform** 부트스트랩 프로세스 시작* 섹션을 참조하십시오.

c.

DHCP 서버 구성에 클러스터 노드의 호스트 이름을 정의합니다. 호스트 이름 고려 사항에 대한 자세한 내용은 ***DHCP**를 통해 클러스터 노드 호스트 이름 설정* 섹션을 참조하십시오.

참고

DHCP 서비스를 사용하지 않는 경우 클러스터 노드는 역방향 **DNS** 조회를 통해 호스트 이름을 가져옵니다.

2.

네트워크 인프라가 클러스터 구성 요소 간 필수 네트워크 연결을 제공하는지 확인합니다. 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 인프라 섹션의 네트워킹 요구 사항* 섹션을 참조하십시오.

3.

OpenShift Container Platform 클러스터 구성 요소가 통신하는 데 필요한 포트를 활성화하도록 방화벽을 구성합니다. 필요한 포트에 대한 자세한 내용은 *사용자 프로비저닝 인프라 섹션의 네트워킹 요구 사항* 섹션을 참조하십시오.

4.

클러스터에 필요한 **DNS** 인프라를 설정합니다.

a.

Kubernetes API, 애플리케이션 와일드카드, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템의 **DNS** 이름 확인을 구성합니다.

b.

Kubernetes API, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템에 대한 역방향 **DNS** 확인을 구성합니다.

OpenShift Container Platform DNS 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 **DNS** 요구 사항* 섹션을 참조하십시오.

5.

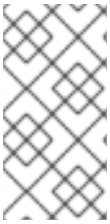
DNS 구성을 확인합니다.

a.
설치 노드에서 **Kubernetes API**의 레코드 이름, 와일드카드 경로 및 클러스터 노드에 대해 **DNS** 조회를 실행합니다. 응답의 **IP** 주소가 올바른 구성 요소에 해당하는지 확인합니다.

b.
설치 노드에서 로드 밸런서 및 클러스터 노드의 **IP** 주소에 대해 역방향 **DNS** 조회를 실행합니다. 응답의 레코드 이름이 올바른 구성 요소에 해당하는지 확인합니다.

자세한 **DNS** 검증 단계는 *사용자 프로비저닝 인프라에 대한 DNS 확인* 섹션을 참조하십시오.

6.
필요한 **API** 및 애플리케이션 수신 로드 밸런싱 인프라를 프로비저닝합니다. 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구 사항* 섹션을 참조하십시오.



참고

일부 로드 밸런싱 솔루션에는 로드 밸런싱을 초기화하기 전에 클러스터 노드의 **DNS** 이름을 확인해야 합니다.

14.2.5. 사용자 프로비저닝 인프라에 대한 DNS 확인 검증

사용자 프로비저닝 인프라에 **OpenShift Container Platform**을 설치하기 전에 **DNS** 구성을 확인할 수 있습니다.

중요

클러스터를 설치하기 전에 이 섹션에 설명된 검증 단계를 성공해야 합니다.

사전 요구 사항

- 사용자 프로비저닝 인프라에 필요한 **DNS** 레코드를 구성했습니다.

프로세스

1.
설치 노드에서 **Kubernetes API**의 레코드 이름, 와일드카드 경로 및 클러스터 노드에 대해 **DNS** 조회를 실행합니다. 응답에 포함된 **IP** 주소가 올바른 구성 요소에 해당하는지 확인합니다.

a.
Kubernetes API 레코드 이름을 조회합니다. 결과가 **API** 로드 밸런서의 **IP** 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
```

1

<nameserver_ip>를 네임서버의 IP 주소로, <cluster_name>을 클러스터 이름으로, <base_domain>을 기본 도메인 이름으로 바꿉니다.

출력 예

```
api.ocp4.example.com.  0 IN A 192.168.1.5
```

b.

Kubernetes 내부 API 레코드 이름을 조회합니다. 결과가 API 로드 밸런서의 IP 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
```

출력 예

```
api-int.ocp4.example.com.  0 IN A 192.168.1.5
```

c.

예제 *.apps.<cluster_name>.<base_domain>을 테스트합니다. DNS 와일드카드를 조회합니다. 모든 애플리케이션 와일드카드 조회는 애플리케이션 인그레스 로드 밸런서의 IP 주소로 확인되어야 합니다.

```
$ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.
<base_domain>
```

출력 예

```
random.apps.ocp4.example.com.  0 IN A 192.168.1.5
```

참고

예제 출력에서는 **Kubernetes API** 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 **API** 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

random 항목을 다른 와일드카드 값으로 교체할 수 있습니다. 예를 들어 **OpenShift Container Platform** 콘솔의 경로를 쿼리할 수 있습니다.

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

출력 예

```
console-openshift-console.apps.ocp4.example.com. 0 IN A 192.168.1.5
```

d.

부트스트랩 **DNS** 레코드 이름에 대해 조회를 실행합니다. 결과가 부트스트랩 노드의 **IP** 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.
<base_domain>
```

출력 예

```
bootstrap.ocp4.example.com.  0 IN A 192.168.1.96
```

e.

이 방법을 사용하여 컨트롤 플레인 및 컴퓨팅 노드의 **DNS** 레코드 이름에 대해 조회를 수행합니다. 결과가 각 노드의 **IP** 주소에 해당하는지 확인합니다.

2. 설치 노드에서 로드 밸런서 및 클러스터 노드의 IP 주소에 대해 역방향 DNS 조회를 실행합니다. 응답에 포함된 레코드 이름이 올바른 구성 요소에 해당하는지 확인합니다.

a. API 로드 밸런서의 IP 주소에 대해 역방향 조회를 수행합니다. 응답에 Kubernetes API 및 Kubernetes 내부 API의 레코드 이름이 포함되어 있는지 확인합니다.

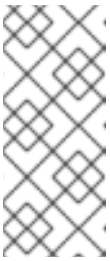
```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

출력 예

```
5.1.168.192.in-addr.arpa. 0 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 0 IN PTR api.ocp4.example.com. 2
```

1 Kubernetes 내부 API의 레코드 이름을 제공합니다.

2 Kubernetes API의 레코드 이름을 제공합니다.



참고

OpenShift Container Platform 애플리케이션 와일드카드에는 PTR 레코드가 필요하지 않습니다. 애플리케이션 인그레스 로드 밸런서의 IP 주소에 대한 역방향 DNS 확인에는 유효성 검사 단계가 필요하지 않습니다.

b. 부트스트랩 노드의 IP 주소에 대해 역방향 조회를 수행합니다. 결과가 부트스트랩 노드의 DNS 레코드 이름을 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

출력 예

```
96.1.168.192.in-addr.arpa. 0 IN PTR bootstrap.ocp4.example.com.
```

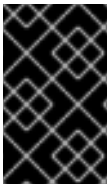
c. 이 방법을 사용하여 컨트롤 플레인 및 컴퓨팅 노드의 IP 주소에 대해 역방향 조회를 수행합니다. 결과가 각 노드의 DNS 레코드 이름과 일치하는지 확인합니다.

14.2.6. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 core 사용자의 ~/.ssh/authorized_keys 목록에 추가되어 암호 없는 인증을 활성화합니다.

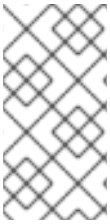
키가 노드에 전달되면 키 쌍을 사용하여 사용자 core로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. ./openshift-install gather 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.



중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.



참고

AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1

새 SSH 키의 경로 및 파일 이름(예: ~/.ssh/id_ed25519)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 '~/.ssh 디렉터리에 있는지 확인하십시오.

참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 ed25519 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 rsa 또는 ecdsa 알고리즘을 사용하는 키를 생성합니다.

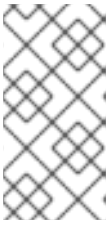
2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 ~/.ssh/id_ed25519.pub 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 ./openshift-install gather 명령을 사용하려는 경우 필요합니다.



참고

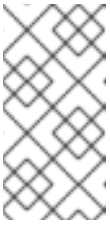
일부 배포에서는 ~/.ssh/id_rsa 및 ~/.ssh/id_dsa와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. ssh-agent 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```



참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. ssh-agent에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: ~/.ssh/id_ed25519).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

14.2.7. 설치 프로그램 받기

OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

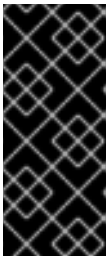
- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

프로세스

1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.

2. 인프라 공급자를 선택합니다.

3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.



중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.

중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 OpenShift Container Platform 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

14.2.8. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift CLI(oc)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 oc를 설치할 수 있습니다.

중요

이전 버전의 oc를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 oc를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 Linux에서 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

절차

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.
2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. OpenShift v4.9 Linux Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. oc 바이너리를 PATH에 있는 디렉터리에 배치합니다.

   PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치

다음 절차에 따라 Windows에 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

절차

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.
2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. OpenShift v4.9 Windows Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.
4. ZIP 프로그램으로 아카이브의 압축을 풉니다.
5. oc 바이너리를 PATH에 있는 디렉터리로 이동합니다.

   PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

   ```
   C:\> path
   ```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치

다음 절차에 따라 macOS에서 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

절차

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.
2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. OpenShift v4.9 MacOSX Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.

4. 아카이브의 압축을 해제하고 압축을 풉니다.

5. oc 바이너리 PATH의 디렉터리로 이동합니다.

   PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
$ oc <command>
```

14.2.9. 수동으로 설치 구성 파일 만들기

OpenShift Container Platform을 사용자가 프로비저닝한 설치의 경우 설치 구성 파일을 수동으로 생성합니다.

사전 요구 사항

- 로컬 시스템에 설치 프로그램에 제공할 SSH 공개 키가 있습니다. 키는 디버깅 및 재해 복구를 위해 클러스터 노드에 대한 SSH 인증에 사용됩니다.
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿이 있습니다.

절차

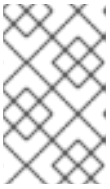
1. 필요한 설치 자산을 저장할 설치 디렉터리를 만듭니다.

```
$ mkdir <installation_directory>
```

중요

디렉터리를 만들어야 합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

2. 샘플 install-config.yaml 파일 템플릿을 사용자 지정하여 <installation_directory>에 저장합니다.



참고

이 설정 파일의 이름을 install-config.yaml로 지정해야 합니다.

참고

일부 플랫폼 유형의 경우 대체로 ./openshift-install create install-config --dir <installation_directory> 를 실행하여 install-config.yaml 파일을 생성할 수 있습니다. 프롬프트에서 클러스터 구성에 대한 세부 정보를 제공할 수 있습니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 install-config.yaml 파일을 백업합니다.

중요

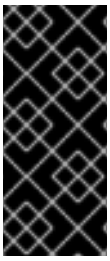
install-config.yaml 파일은 설치 과정의 다음 단계에서 사용됩니다. 이 시점에서 이를 백업해야 합니다.

14.2.9.1. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 환경에 대한 세부 정보를 설명하는 사용자 지정 install-config.yaml 설치 구성 파일을 제공합니다.

참고

설치한 후에는 install-config.yaml 파일에서 이러한 매개변수를 수정할 수 없습니다.



중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

14.2.9.1.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 14.9. 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	소문자, 하이픈(**-**), 마침표(**.**)로 구성되는 문자열(예: **dev**)입니다.

매개변수	설명	값
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

14.2.9.1.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 14.10. 네트워크 매개변수

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체 **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.

매개변수	설명	값
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.
networking.clusterNetwork	Pod의 IP 주소 블록입니다. 기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된 **cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다. 여러 IP 커널 인수를 지정하는 경우 **machineNetwork.cidr** 값은 기본 네트워크의 CIDR이어야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16

매개변수	설명	값
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16**  **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

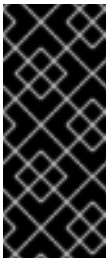
14.2.9.1.3. 선택적 구성 매개변수

선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 14.11. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **ppc64le** (기본값)입니다.	문자열

매개변수	설명	값
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은 **3**입니다.
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **ppc64le** (기본값)입니다.	문자열

매개변수	설명	값
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 기본값은 **External**입니다. 이 필드를 **Internal** 로 설정하는 것은 클라우드 이외의 플랫폼에서는 지원되지 않습니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

14.2.9.2. IBM Power의 샘플 install-config.yaml 파일

install-config.yaml 파일을 사용자 지정하여 OpenShift Container Platform 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.

```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 0 (4)
  architecture : ppc64le
controlPlane: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3 (7)
  architecture : ppc64le
metadata:
  name: test (8)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 (9)
    hostPrefix: 23 (10)
  networkType: OpenShiftSDN
  serviceNetwork: (11)
  - 172.30.0.0/16
platform:
  none: {} (12)
fips: false (13)
pullSecret: '{"auths": ...}' (14)
sshKey: 'ssh-ed25519 AAAA...' (15)
```

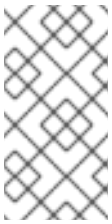
(1)

2 5

controlPlane 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

3 6

동시 멀티스레딩(**SMT**) 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 **SMT**가 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. **SMT**를 비활성화하는 경우 모든 클러스터 머신에서 이를 비활성화해야 합니다. 여기에는 컨트롤 플레인과 컴퓨팅 머신이 모두 포함됩니다.



참고

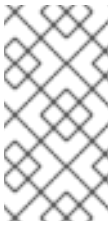
SMT(동시 멀티 스레딩)는 기본적으로 활성화되어 있습니다. **BIOS** 설정에서 **SMT**를 활성화하지 않으면 **hyperthreading** 매개변수가 적용되지 않습니다.

중요

BIOS에서든 **install-config.yaml** 파일에서든 **hyperthreading**을 비활성화한 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.

4

사용자 프로비저닝 인프라에 **OpenShift Container Platform**을 설치할 때 이 값을 **0**으로 설정해야 합니다. 설치 프로그램에서 제공하는 설치에서 매개 변수는 클러스터가 생성 및 관리하는 컴퓨팅 머신 수를 제어합니다. 사용자 프로비저닝 설치에서는 클러스터 설치를 완료하기 전에 컴퓨팅 시스템을 수동으로 배포해야 합니다.



참고

3-노드 클러스터를 설치하는 경우 **RHCOS(Red Hat Enterprise Linux CoreOS)** 시스템을 설치할 때 컴퓨팅 머신을 배포하지 마십시오.

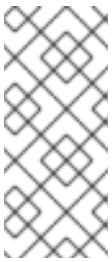
7

클러스터에 추가하는 컨트롤 플레인 시스템의 수입니다. 클러스터에서 이 값을 클러스터의 **etcd** 끝점 수로 사용하므로 이 값은 배포하는 컨트롤 플레인 시스템의 수와 일치해야 합니다.

8

9

Pod IP 주소가 할당되는 IP 주소 블록입니다. 이 블록은 기존 물리적 네트워크와 중복되지 않아야합니다. 이러한 IP 주소는 Pod 네트워크에 사용됩니다. 외부 네트워크에서 Pod에 액세스해야 하는 경우, 트래픽을 관리하도록 로드 밸런서와 라우터를 설정해야 합니다.



참고

클래스 E CIDR 범위는 향후 사용을 위해 예약되어 있습니다. 클래스 E CIDR 범위를 사용하려면 네트워킹 환경에서 클래스 E CIDR 범위 내에서 IP 주소를 수락하는지 확인해야 합니다.

10

개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 hostPrefix를 23으로 설정하면 지정된 cidr 이외 /23 서브넷이 각 노드에 할당되어 510(2^(32 - 23) - 2) Pod IP 주소가 허용됩니다. 외부 네트워크에서 노드에 액세스해야 하는 경우 트래픽을 관리하도록 로드 밸런서와 라우터를 구성합니다.

11

서비스 IP 주소에 사용할 IP 주소 풀입니다. IP 주소 풀은 하나만 입력할 수 있습니다. 이 블록은 기존 물리적 네트워크와 중복되지 않아야합니다. 외부 네트워크에서 서비스에 액세스해야 하는 경우, 트래픽을 관리하도록 로드 밸런서와 라우터를 구성합니다.

12

플랫폼을 none으로 설정해야 합니다. IBM Power 인프라에 대한 추가 플랫폼 구성 변수는 지정할 수 없습니다.

중요

플랫폼 유형으로 설치된 클러스터는 Machine API로 컴퓨팅 머신 관리와 같은 일부 기능을 사용할 수 없습니다. 이 제한은 클러스터에 연결된 컴퓨팅 시스템이 일반적으로 기능을 지원하는 플랫폼에 설치된 경우에도 적용됩니다. 이 매개변수는 설치 후 변경할 수 없습니다.

13

FIPS 모드 활성화 또는 비활성화 여부입니다. 기본적으로 FIPS 모드는 비활성화됩니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.

중요

FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 **OpenShift Container Platform** 배포에서만 지원됩니다.

14

Red Hat OpenShift Cluster Manager의 풀 시크릿. 이 풀 시크릿을 사용하면 **OpenShift Container Platform** 구성 요소에 대한 컨테이너 이미지를 제공하는 **Quay.io**를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

15

RHCOS(Red Hat Enterprise Linux CoreOS)의 **core** 사용자에 대한 **SSH** 공용 키입니다.

참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 **OpenShift Container Platform** 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 **SSH** 키를 지정합니다.

14.2.9.3. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 **HTTP** 또는 **HTTPS** 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 **OpenShift Container Platform** 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 **API**에 대한 호출을 포함하여 모든 클러스터 발신(**Egress**) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.

참고

Proxy 오브젝트의 status.noProxy 필드는 설치 구성에 있는 networking.machineNetwork[].cidr, networking.clusterNetwork[].cidr, networking.serviceNetwork[] 필드의 값으로 채워집니다.

Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 Red Hat OpenStack Platform (RHOSP)에 설치하는 경우 Proxy 오브젝트 status.noProxy 필드도 인스턴스 메타데이터 끝점(169.254.169.254)로 채워집니다.

절차

1. install-config.yaml 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

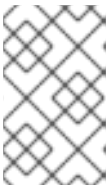
```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1 클러스터 외부에서 HTTP 연결을 구축하는 데 사용할 프록시 URL입니다. URL 스키마는 http여야 합니다.

2 클러스터 외부에서 HTTPS 연결을 구축하는 데 사용할 프록시 URL입니다.

3 대상 도메인 이름, IP 주소 또는 프록시에서 제외할 기타 네트워크 CIDR로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 .을 입력합니다. 예를 들어, .y.com은 x.y.com과 일치하지만 y.com은 일치하지 않습니다. *를 사용하여 모든 대상에 대해 프록시를 바이패스합니다.

4

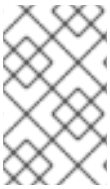


참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 **OpenShift Container Platform**을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 **nil**이 됩니다.



참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

14.2.9.4. 3개의 노드 클러스터 구성

필요한 경우 세 개의 컨트롤 플레인 시스템으로 구성된 베어 메탈 클러스터에 제로 컴퓨팅 머신을 배포할 수 있습니다. 이를 통해 클러스터 관리자와 개발자들이 테스트, 개발, 프로덕션에 사용할 수 있는 소형화되고 리소스 효율이 높은 클러스터를 제공합니다.

3-노드 **OpenShift Container Platform** 환경에서 세 개의 컨트롤 플레인 머신을 예약할 수 있습니다. 즉, 애플리케이션 워크로드가 해당 플랫폼에서 실행되도록 예약됩니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.

절차

- **install-config.yaml** 파일에서 다음 **compute** 스탠자에 표시된 대로 컴퓨팅 복제본 수가 **0**으로 설정되어 있는지 확인합니다.

```
compute:
- name: worker
  platform: {}
```

```
replicas: 0
```

참고

배포 중인 컴퓨팅 머신 수에 관계없이 사용자 프로비저닝 인프라에 **OpenShift Container Platform**을 설치할 때 컴퓨팅 머신의 **replicas** 매개변수 값을 **0**으로 설정해야 합니다. 설치 프로그램에서 제공하는 설치에서 매개 변수는 클러스터가 생성 및 관리하는 컴퓨팅 머신 수를 제어합니다. 이 설정은 컴퓨팅 시스템이 수동으로 배포되는 사용자 프로비저닝 설치에는 적용되지 않습니다.

3-노드 클러스터 설치의 경우 다음 단계를 따르십시오.

- 컴퓨팅 노드가 **0**인 **3-**노드 클러스터를 배포하는 경우 **Ingress** 컨트롤러 **Pod**는 컨트롤 플레인 노드에서 실행됩니다. **3-**노드 클러스터 배포에서 **HTTP** 및 **HTTPS** 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다. 자세한 내용은 *사용자 프로비저닝 인프라 섹션에 대한 로드 밸런싱 요구 사항* 섹션을 참조하십시오.

- 다음 절차에서 **Kubernetes** 매니페스트 파일을 생성할 때 **<installation_directory>/manifests/cluster-scheduler-02-config.yml** 파일의 **mastersSchedulable** 매개변수가 **true**로 설정되어 있는지 확인합니다. 이렇게 하면 애플리케이션 워크로드를 컨트롤 플레인 노드에서 실행할 수 있습니다.

- **RHCOS(Red Hat Enterprise Linux CoreOS)** 시스템을 생성할 때 컴퓨팅 노드를 배포하지 마십시오.

14.2.10. CNO(Cluster Network Operator) 구성

클러스터 네트워크의 구성은 **CNO(Cluster Network Operator)** 구성의 일부로 지정되며 **cluster**라는 이름의 **CR(**사용자 정의 리소스**)** 오브젝트에 저장됩니다. **CR**은 **operator.openshift.io API** 그룹에서 **Network API**의 필드를 지정합니다.

CNO 구성은 **Network.config.openshift.io API** 그룹의 **Network API**에서 클러스터 설치 중에 다음 필드를 상속하며 이러한 필드는 변경할 수 없습니다.

clusterNetwork

Pod IP 주소가 할당되는 **IP** 주소 풀입니다.

serviceNetwork

서비스를 위한 IP 주소 풀입니다.

defaultNetwork.type

OpenShift SDN 또는 OVN-Kubernetes와 같은 클러스터 네트워크 공급자입니다.

cluster라는 CNO 오브젝트에서 defaultNetwork 오브젝트의 필드를 설정하여 클러스터의 클러스터 네트워크 공급자 구성을 지정할 수 있습니다.

14.2.10.1. CNO(Cluster Network Operator) 구성 오브젝트

CNO(Cluster Network Operator)의 필드는 다음 표에 설명되어 있습니다.

표 14.12. CNO(Cluster Network Operator) 구성 오브젝트

필드	유형	설명
metadata.name	**string**	CNO 개체 이름입니다. 이 이름은 항상 **cluster**입니다.

spec.clusterNetwork	**array**	Pod IP 주소가 할당되는 IP 주소 블록과 클러스터의 각 개별 노드에 할당된 서브넷 접두사 길이를 지정하는 목록입니다. 예를 들면 다음과 같습니다.
		`spec:`
		`  clusterNetwork:`
		`  - cidr: 10.128.0.0/19`
		`  hostPrefix: 23`
		`  - cidr: 10.128.32.0/19`
		`  hostPrefix: 23`
		매니페스트를 생성하기 전에 **install-config.yaml** 파일에서만 이 필드를 사용자 지정할 수 있습니다. 값은 매니페스트 파일에서 읽기 전용입니다.

spec.serviceNetwork	**array**	서비스의 IP 주소 블록입니다. OpenShift SDN 및 OVN-Kubernetes CNI(Container Network Interface) 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다. 예를 들면 다음과 같습니다.
		`spec:`
		`  serviceNetwork:`
		`  - 172.30.0.0/14`
		매니페스트를 생성하기 전에 **install-config.yaml** 파일에서만 이 필드를 사용자 지정할 수 있습니다. 값은 매니페스트 파일에서 읽기 전용입니다.

필드	유형	설명
spec.defaultNetwork	**object**	클러스터 네트워크의 CNI(Container Network Interface) 클러스터 네트워크 공급자를 구성합니다.
spec.kubeProxyConfig	**object**	이 개체의 필드는 kube-proxy 구성을 지정합니다. OVN-Kubernetes 클러스터 네트워크 공급자를 사용하는 경우 kube-proxy 구성이 적용되지 않습니다.

defaultNetwork 오브젝트 구성

defaultNetwork 오브젝트의 값은 다음 표에 정의되어 있습니다.

표 14.13. defaultNetwork 오브젝트

필드	유형	설명
type	**string**	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, 클러스터 네트워크 공급자가 설치 중에 선택됩니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다. **참고** OpenShift Container Platform은 기본적으로 OpenShift SDN CNI(Container Network Interface) 클러스터 네트워크 공급자를 사용합니다.
openshiftSDNConfig	**object**	이 오브젝트는 OpenShift SDN 클러스터 네트워크 공급자에만 유효합니다.
ovnKubernetesConfig	**object**	이 오브젝트는 OVN-Kubernetes 클러스터 네트워크 공급자에만 유효합니다.

OpenShift SDN CNI 네트워크 공급자에 대한 구성

다음 표에서는 OpenShift SDN Container Network Interface (CNI) 클러스터 네트워크 공급자의 구성 필드를 설명합니다.

표 14.14. openshiftSDNConfig 오브젝트

필드	유형	설명

필드	유형	설명
mode	**string**	OpenShift SDN의 네트워크 격리 모드를 구성합니다. 기본값은 **NetworkPolicy**입니다. **Multitenant** 및 **Subnet** 값은 OpenShift Container Platform 3.x 버전과의 역호환할 수 있지만 권장되지는 않습니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
mtu	**integer**	VXLAN 오버레이 네트워크의 최대 전송 단위(MTU)입니다. 이는 기본 네트워크 인터페이스의 MTU를 기준으로 자동 탐지됩니다. 일반적으로 감지된 MTU를 재정의할 필요는 없습니다. 자동 감지 값이 예상 밖인 경우 노드의 기본 네트워크 인터페이스의 MTU가 올바른지 확인합니다. 이 옵션을 사용하여 노드의 기본 네트워크 인터페이스의 MTU 값을 변경할 수 없습니다. 클러스터에 다른 노드에 대한 다른 MTU 값이 필요한 경우 이 값을 클러스터의 가장 낮은 MTU 값보다 **50** 미만으로 설정해야 합니다. 예를 들어 클러스터의 일부 노드에는 **9001**의 MTU가 있고 일부에는 **1500**의 MTU가 있는 경우 이 값을 **1450**으로 설정해야 합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
vxlanPort	**integer**	모든 VXLAN 패킷에 사용할 포트입니다. 기본값은 **4789**입니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다. 기존 노드가 다른 VXLAN 네트워크에 속하는 가상 환경에서 실행 중인 경우에는 기본값을 변경해야 할 수도 있습니다. 예를 들어 VMware NSX-T 위에서 OpenShift SDN 오버레이를 실행할 때 두 SDN이 동일한 기본 VXLAN 포트 번호를 사용하므로 VXLAN의 대체 포트를 선택해야 합니다. AWS(Amazon Web Services)에서는 포트 **9000**과 포트 **9999** 사이에서 VXLAN의 대체 포트를 선택할 수 있습니다.

OpenShift SDN 구성 예

```
defaultNetwork:
  type: OpenShiftSDN
  openshiftSDNConfig:
    mode: NetworkPolicy
    mtu: 1450
    vxlanPort: 4789
```

OVN-Kubernetes CNI 클러스터 네트워크 공급자에 대한 구성

다음 표에서는 **OVN-Kubernetes CNI** 클러스터 네트워크 공급자의 구성 필드를 설명합니다.

표 14.15. ovnKubernetesConfig object

필드	유형	설명
mtu	**integer**	Geneve(Generic Network Virtualization Encapsulation) 오버레이 네트워크의 MTU(최대 전송 단위)입니다. 이는 기본 네트워크 인터페이스의 MTU를 기준으로 자동 탐지됩니다. 일반적으로 감지된 MTU를 재정의할 필요는 없습니다. 자동 감지 값이 예상 밖인 경우 노드의 기본 네트워크 인터페이스의 MTU가 올바른지 확인합니다. 이 옵션을 사용하여 노드의 기본 네트워크 인터페이스의 MTU 값을 변경할 수 없습니다. 클러스터에 다른 노드에 대한 다른 MTU 값이 필요한 경우, 이 값을 클러스터의 가장 낮은 MTU 값보다 **100** 미만으로 설정해야 합니다. 예를 들어, 클러스터의 일부 노드에 **9001**의 MTU가 있고 일부에는 **1500**의 MTU가 있는 경우 이 값을 **1400**으로 설정해야 합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
genevePort	**integer**	모든 Geneve 패킷에 사용할 포트입니다. 기본값은 **6081**입니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
ipsecConfig	**object**	IPsec 암호화를 활성화하려면 빈 오브젝트를 지정합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
policyAuditConfig	**object**	네트워크 정책 감사 로깅을 사용자 정의할 구성 오브젝트를 지정합니다. 설정되지 않으면 기본값 감사 로그 설정이 사용됩니다.

표 14.16. policyAuditConfig object

필드	유형	설명
rateLimit	integer	노드당 1초마다 생성할 최대 메시지 수입니다. 기본값은 초당 **20** 개의 메시지입니다.
maxFileSize	integer	감사 로그의 최대 크기(바이트)입니다. 기본값은 **50000000** 또는 **50** MB입니다.

필드	유형	설명
대상	string	다음 추가 감사 로그 대상 중 하나입니다. **libc** 호스트에서 journald 프로세스의 libc **syslog()** 함수입니다. **udp:<host>:<port>** syslog 서버입니다. **<host>:<port>**를 syslog 서버의 호스트 및 포트로 바꿉니다. **unix:<file>** **<file>**로 지정된 Unix Domain Socket 파일입니다. **null** 감사 로그를 추가 대상으로 보내지 마십시오.
syslogFacility	string	RFC5424에 정의된 **kern**과 같은 syslog 기능입니다. 기본값은 **local0**입니다.

OVN-Kubernetes 구성 예

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
    ipsecConfig: {}
```

kubeProxyConfig 오브젝트 구성

kubeProxyConfig 오브젝트의 값은 다음 표에 정의되어 있습니다.

표 14.17. kubeProxyConfig object

필드	유형	설명

필드	유형	설명
iptablesSyncPeriod	**string**	**iptables** 규칙의 새로 고침 간격입니다. 기본값은 **30s**입니다. 유효 접미사로 **s**, **m**, **h**가 있으며, 자세한 설명은 Go **time** 패키지 문서를 참조하십시오. **참고** OpenShift Container Platform 4.3 이상에서는 성능이 개선되어 더 이상 **iptablesSyncPeriod** 매개변수를 조정할 필요가 없습니다.

proxyArguments.iptables-min-sync-period	**array**	**iptables** 규칙을 새로 고치기 전 최소 기간입니다. 이 필드를 통해 새로 고침 간격이 너무 짧지 않도록 조정할 수 있습니다. 유효 접미사로 **s**, **m**, **h**가 있으며, 자세한 설명은 Go **time** 패키지를 참조하십시오. 기본값은 다음과 같습니다.
		
		`kubeProxyConfig:`
		`  proxyArguments:`
		`  iptables-min-sync-period:`
		`  - 0s`

14.2.11. Kubernetes 매니페스트 및 Ignition 설정 파일 생성

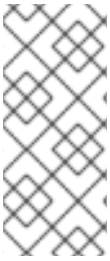
일부 클러스터 정의 파일을 수정하고 클러스터 시스템을 수동으로 시작해야 하므로 클러스터가 시스템을 구성하는 데 필요한 **Kubernetes** 매니페스트 및 **Ignition** 구성 파일을 사용자가 생성해야 합니다.

설치 구성 파일은 **Kubernetes** 매니페스트로 변환됩니다. 매니페스트는 나중에 클러스터 머신을 구성하는 데 사용되는 **Ignition** 구성 파일로 래핑됩니다.

중요

- **OpenShift Container Platform** 설치 프로그램에서 생성하는 **Ignition** 구성 파일에 **24**시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 **24**시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 **kubelet** 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(**CSR**)을 수동으로 승인해야 합니다. 자세한 내용은 *만료된 컨트롤 플레인 인증서에서 복구* 문서를 참조하십시오.

- 클러스터를 설치한 후 **24**시간에서 **22**시간까지의 인증서가 교체되기 때문에 생성된 후 **12**시간 이내에 **Ignition** 구성 파일을 사용하는 것이 좋습니다. **12**시간 이내에 **Ignition** 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.



참고

매니페스트 및 **Ignition** 파일을 생성하는 설치 프로그램은 아키텍처에 따라 다르며 클라이언트 이미지 미러에서 얻을 수 있습니다. 설치 프로그램의 **Linux** 버전은 **ppc64le**에서만 실행됩니다. 이 설치 프로그램은 **Mac OS** 버전으로도 사용할 수 있습니다.

사전 요구 사항

- **OpenShift Container Platform** 설치 프로그램을 가져오셨습니다.
- **install-config.yaml** 설치 구성 파일을 생성하셨습니다.

프로세스

1. **OpenShift Container Platform** 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터에 대한 **Kubernetes** 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1. **<installation_directory>**는 사용자가 만든 **install-config.yaml** 파일이 포함된 설치 디렉터리를 지정합니다.

주의

3 노드 클러스터를 실행 중이면 다음 단계를 건너 뛰어 컨트롤 플레인 노드 일정을 계획할 수 있도록 하십시오.

중요

예약할 수 없는 기본에서 컨트롤 플레인 노드를 구성하면 추가 서브스크립션이 필요합니다. 이는 컨트롤 플레인 노드가 작업자 노드가 되기 때문입니다.

2. <installation_directory>/manifests/cluster-scheduler-02-config.yml Kubernetes 매니페스트 파일의 mastersSchedulable 매개변수가 false로 설정되어 있는지 확인합니다. 이 설정으로 인해 컨트롤 플레인 머신에서 포드가 예약되지 않습니다.

   a. <installation_directory>/manifests/cluster-scheduler-02-config.yml 파일을 엽니다.

   b. mastersSchedulable 매개변수를 찾아서 값을 False로 설정되어 있는지 확인합니다.

   c. 파일을 저장하고 종료합니다.

3. Ignition 구성 파일을 생성하려면 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행합니다.

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 <installation_directory>는 동일한 설치 디렉터리를 지정합니다.

설치 디렉터리의 부트스트랩, 컨트롤 플레인 및 컴퓨팅 노드에 대한 Ignition 구성 파일이 생성됩니다. kubeadmin-password 및 kubeconfig 파일은 ./<installation_directory>/auth 디렉터리에 생성됩니다.

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

14.2.12. RHCOS 설치 및 OpenShift Container Platform 부트스트랩 프로세스 시작

프로비저닝하는 IBM Power 인프라에 OpenShift Container Platform을 설치하려면 머신에 RHCOS(Red Hat Enterprise Linux CoreOS)를 설치해야 합니다. RHCOS를 설치할 때 설치 중인 머신 유형에 대해 OpenShift Container Platform 설치 프로그램에서 생성한 Ignition 구성 파일을 제공해야 합니다. 적합한 네트워킹, DNS 및 로드 밸런싱 인프라를 구성한 경우 RHCOS 머신이 재부팅된 후 OpenShift Container Platform 부트스트랩 프로세스가 자동으로 시작됩니다.

단계에 따라 ISO 이미지 또는 네트워크 PXE 부팅을 사용하여 시스템에 RHCOS를 설치합니다.

14.2.12.1. ISO 이미지를 사용하여 RHCOS 설치

ISO 이미지를 사용하여 시스템에 RHCOS을 설치할 수 있습니다.

사전 요구 사항

- 클러스터에 대한 Ignition 구성 파일이 생성되어 있습니다.
- 적합한 네트워크, DNS 및 로드 밸런싱 인프라가 구성되어 있습니다.
- 사용자 컴퓨터에서 액세스할 수 있고 사용자가 생성한 시스템에서 액세스할 수 있는 HTTP 서버에 대한 액세스 권한을 확보합니다.
- 네트워킹 및 디스크 파티션과 같은 기능을 구성하는 다양한 방법에 대해서는 *고급 RHCOS 설치 구성*섹션을 살펴보십시오.

프로세스

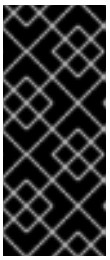
1. 각 Ignition 구성 파일에 대해 SHA512 다이제스트를 가져옵니다. 예를 들어 Linux를 실행하는 시스템에서 다음을 사용하여 bootstrap.ign Ignition 구성 파일의 SHA512 다이제스트를 가져

올 수 있습니다.

```
$ sha512sum <installation_directory>/bootstrap.ign
```

다이제스트는 클러스터 노드에서 Ignition 구성 파일의 신뢰성을 검증하기 위해 이후 단계에서 coreos-installer에 제공됩니다.

2. 설치 프로그램에서 생성한 부트스트랩, 컨트롤 플레인, 컴퓨팅 노드 Ignition 구성 파일을 HTTP 서버에 업로드합니다. 해당 파일의 URL을 기록해 둡니다.



중요

HTTP 서버에 저장하기 전에 Ignition 구성에서 구성 설정을 추가하거나 변경할 수 있습니다. 설치를 마친 후 클러스터에 컴퓨팅 시스템을 더 추가하려면 Ignition 구성 파일을 삭제하지 마십시오.

3. 설치 호스트에서 Ignition 구성 파일을 URL에서 사용할 수 있는지 확인합니다. 다음 예에서는 부트스트랩 노드에 대한 Ignition 구성 파일을 가져옵니다.

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

출력 예

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-
rsa...
```

명령에서 bootstrap.ign을 master.ign 또는 worker.ign으로 교체하여 컨트롤 플레인 및 컴퓨팅 노드의 Ignition 구성 파일도 사용할 수 있는지 확인합니다.

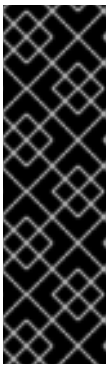
4. RHCOS 이미지 미러 이미지 미러 페이지에서 원하는 운영 체제 인스턴스 설치 방법에 필요한 RHCOS 이미지를가져올 수 있지만 RHCOS 이미지 의 올바른 버전을 가져오는 권장 방법은

openshift-install 명령 출력에서 있습니다.

```
$ openshift-install coreos print-stream-json | grep '\.iso[^.]'
```

출력 예

```
"location": "<url>/art/storage/releases/rhcos-4.9-aarch64/<release>/aarch64/rhcos-
<release>-live.aarch64.iso",
"location": "<url>/art/storage/releases/rhcos-4.9-ppc64le/<release>/ppc64le/rhcos-
<release>-live.ppc64le.iso",
"location": "<url>/art/storage/releases/rhcos-4.9-s390x/<release>/s390x/rhcos-
<release>-live.s390x.iso",
"location": "<url>/art/storage/releases/rhcos-4.9/<release>/x86_64/rhcos-<release>-
live.x86_64.iso",
```



중요

RHCOS 이미지는 OpenShift Container Platform 릴리스에 따라 변경되지 않을 수 있습니다. 설치하는 OpenShift Container Platform 버전과 같거나 그 이하의 버전 중 가장 최신 버전의 이미지를 다운로드해야 합니다. 사용 가능한 경우 OpenShift Container Platform 버전과 일치하는 이미지 버전을 사용합니다. 이 프로세스에는 ISO 이미지만 사용하십시오. 이 설치 유형에서는 RHCOS qcow2 이미지가 지원되지 않습니다.

ISO 파일 이름은 다음 예와 유사합니다.

```
rhcos-<version>-live.<architecture>.iso
```

5. ISO를 사용하여 RHCOS 설치를 시작합니다. 다음 설치 옵션 중 하나를 사용합니다.

- ISO 이미지를 디스크에 굽고 직접 부팅합니다.
- LOM(Lightweight-out Management) 인터페이스를 사용하여 ISO 리디렉션을 사용합니다.

6. 옵션을 지정하거나 라이브 부팅 시퀀스를 중단하지 않고 **RHCOS ISO** 이미지를 부팅합니다. 설치 프로그램이 **RHCOS** 라이브 환경에서 쉘 프롬프트로 부팅될 때까지 기다립니다.

참고

커널 인수를 추가하기 위해 **RHCOS** 설치 부팅 프로세스를 중단할 수 있습니다. 하지만 이 **ISO** 프로세스에서는 커널 인수를 추가하지 않고 다음 단계에 설명된 대로 **coreos-installer** 명령을 사용해야 합니다.

7. **coreos-installer** 명령을 실행하고 설치 요구 사항을 충족하는 옵션을 지정합니다. 최소한 노드 유형에 대한 **Ignition** 구성 파일과 설치할 장치를 가리키는 **URL**을 지정해야 합니다.

```
$ sudo coreos-installer install --ignition-url=http://<HTTP_server>/<node_type>.ign
<device> --ignition-hash=sha512-<digest> 1 2
```

1 1

core 사용자에게 설치를 수행하는 데 필요한 **root** 권한이 없으므로 **sudo**를 사용하여 **coreos-installer** 명령을 실행해야 합니다.

2

클러스터 노드에서 **Ignition** 구성 파일을 **HTTP URL**을 통해 가져오려면 **--ignition-hash** 옵션이 필요합니다. **<digest>**는 이전 단계에서 얻은 **Ignition** 구성 파일 **SHA512** 다이제스트입니다.

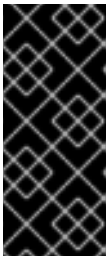
참고

TLS를 사용하는 **HTTPS** 서버를 통해 **Ignition** 구성 파일을 제공하려는 경우 **coreos-installer**를 실행하기 전에 내부 인증 기관**(CA)**을 시스템 신뢰 저장소에 추가할 수 있습니다.

다음 예제에서는 **/dev/sda** 장치에 부트스트랩 노드 설치를 초기화합니다. 부트스트랩 노드의 **Ignition** 구성 파일은 **IP** 주소 **192.168.1.2**가 있는 **HTTP** 웹 서버에서 가져옵니다.

```
$ sudo coreos-installer install --ignition-
url=http://192.168.1.2:80/installation_directory/bootstrap.ign /dev/sda --ignition-
hash=sha512-
a5a2d43879223273c9b60af66b44202a1d1248fc01cf156c46d4a79f552b6bad47bc8cc78dd
f0116e80c59d2ea9e32ba53bc807afbca581aa059311def2c3e3b
```

8. 머신 콘솔에서 RHCOS 설치 진행률을 모니터링합니다.



중요

OpenShift Container Platform 설치를 시작하기 전에 각 노드에서 성공적으로 설치되었는지 확인합니다. 설치 프로세스를 관찰하면 발생할 수 있는 RHCOS 설치 문제의 원인을 파악하는 데 도움이 될 수 있습니다.

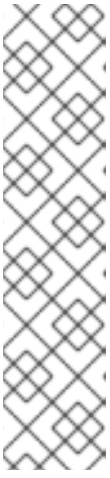
9. RHCOS를 설치한 후 시스템을 재부팅해야 합니다. 시스템이 재부팅되는 동안 지정한 Ignition 구성 파일이 적용됩니다.

10. 계속해서 클러스터에 대한 나머지 시스템을 모두 생성합니다.

중요

이때 부트스트랩 및 컨트롤 플레인 시스템을 생성해야 합니다. 컨트롤 플레인 시스템이 예약 가능하지 않은 경우 OpenShift Container Platform을 설치하기 전에 두 개 이상의 컴퓨팅 시스템도 생성합니다.

적합한 네트워킹, DNS 및 로드 밸런싱 인프라가 있는 경우 RHCOS 노드가 재부팅된 후 OpenShift Container Platform 부트스트랩 프로세스가 자동으로 시작됩니다.



참고

RHCOS 노드에는 core 사용자의 기본 암호가 포함되지어 있지 않습니다. install_config.yaml 파일에 지정한 공개 키에 대한 액세스 권한이 있는 사용자로 ssh core@<node>.<cluster_name>.<base_domain>을 실행하여 노드에 액세스할 수 있습니다. RHCOS를 실행하는 OpenShift Container Platform 4 클러스터 노드는 변경할 수 없으며 Operator를 통해 클러스터 변경 사항을 적용합니다. SSH를 사용하여 클러스터 노드에 액세스하는 것은 권장되지 않습니다. 그러나 설치 문제를 조사할 때 OpenShift Container Platform API를 사용할 수 없거나 kubelet이 대상 노드에서 제대로 작동하지 않는 경우 디버깅 또는 재해 복구에 SSH 액세스가 필요할 수 있습니다.

14.2.12.1.1. 고급 RHCOS 설치 참조

여기서는 RHCOS(Red Hat Enterprise Linux CoreOS) 수동 설치 프로세스를 수정하는 데 사용할 수 있는 네트워킹 구성 및 기타 고급 옵션에 대해 설명합니다. 다음 표에서는 RHCOS 라이브 설치 프로그램

및 **coreos-installer** 명령과 함께 사용할 수있는 커널 인수 및 명령 줄 옵션에 대해 설명합니다.

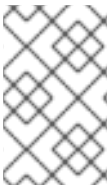
14.2.12.1.1.1. ISO 설치를 위한 네트워킹 및 본딩 옵션

ISO 이미지에서 **RHCOS**를 설치하는 경우, 해당 이미지를 부팅할 때 수동으로 커널 인수를 추가하여 노드의 네트워킹을 구성할 수 있습니다. 네트워킹 인수를 지정하지 않으면 **RHCOS**에서 **Ignition** 구성 파일을 가져오는 데 네트워킹이 필요함을 감지하면 **initramfs**에서 **DHCP**가 활성화됩니다.

중요

네트워킹 인수를 수동으로 추가할 때 **initramfs**에서 네트워크를 가져오려면 **rd.neednet=1** 커널 인수도 추가해야 합니다.

다음 표는 **ISO** 설치를 위해 **RHCOS(Red Hat Enterprise Linux CoreOS)** 노드의 네트워킹 및 본딩 구성 예를 보여줍니다. 예제에서는 **ip=, nameserver=, bond=** 커널 인수를 사용하는 방법을 설명합니다.



참고

커널 인수를 추가할 때 순서가 중요합니다: **ip=, nameserver=** 및 **bond=** 입니다.

이는 시스템 부팅 중에 **dracut** 툴로 전달되는 네트워킹 옵션입니다. **dracut**에서 지원하는 네트워킹 옵션에 대한 자세한 내용은 **dracut.cmdline** 메뉴얼 페이지를 참조하십시오.

다음 정보는 **ISO** 설치를 위해 **RHCOS** 노드에서 네트워킹을 구성하는 예제를 제공합니다. 예제에서는 **ip=** 및 **nameserver=** 커널 인수를 사용하는 방법을 설명합니다.

참고

커널 인수를 추가할 때 순서가 중요합니다(**ip =** 및 **nameserver= **).

이는 시스템 부팅 중에 **dracut** 툴로 전달되는 네트워킹 옵션입니다. **dracut**에서 지원하는 네트워킹 옵션에 대한 자세한 내용은 **dracut.cmdline** 메뉴얼 페이지를 참조하십시오.

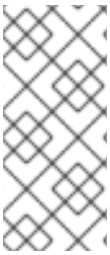
다음 예제는 **ISO** 설치를 위한 네트워킹 옵션입니다.

DHCP 또는 고정 IP 주소 구성

IP 주소를 구성하려면 DHCP(ip=dhcp)를 사용하거나 개별 고정 IP 주소(ip=<host_ip>)를 설정합니다. 정적 IP를 설정하는 경우 각 노드에서 DNS 서버 IP 주소 (nameserver=<dns_ip>)를 확인합니다. 다음 예제에서는 다음을 설정합니다.

- **노드의 IP 주소는 10.10.10.2로 설정됩니다.**
- **게이트웨이 주소는 10.10.10.254로 설정됩니다.**
- **넷마스크는 255.255.255.0으로**
- **core0.example.com에 대한 호스트 이름**
- **4.4.4.41의 DNS 서버 주소**
- **auto-configuration 값을 none으로 설정합니다. IP 네트워킹이 정적으로 구성되면 자동 구성이 필요하지 않습니다.**

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
nameserver=4.4.4.41
```



참고

DHCP를 사용하여 RHCOS 시스템의 IP 주소 지정을 구성하는 경우 시스템은 DHCP를 통해 DNS 서버 정보도 가져옵니다. DHCP 기반 배포의 경우 DHCP 서버 구성을 통해 RHCOS 노드에서 사용할 DNS 서버 주소를 정의할 수 있습니다.

고정 호스트 이름 없이 IP 주소 구성

정적 호스트 이름을 할당하지 않고 IP 주소를 구성할 수 있습니다. 사용자가 정적 호스트 이름을 설정하지 않으면 역방향 DNS 조회에 의해 자동으로 선택됩니다. 정적 호스트 이름이 없는 IP 주소를 구성하려면 다음 예제를 참조합니다.

- **노드의 IP 주소는 10.10.10.2로 설정됩니다.**

- 게이트웨이 주소는 **10.10.10.254**로 설정됩니다.
- 넷마스크는 **255.255.255.0**으로
- **4.4.4.41**의 **DNS** 서버 주소
- **auto-configuration** 값을 **none**으로 설정합니다. **IP** 네트워킹이 정적으로 구성되면 자동 구성이 필요하지 않습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0::enp1s0:none
nameserver=4.4.4.41
```

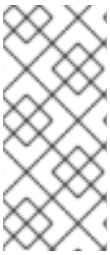
여러 네트워크 인터페이스 지정

여러 **ip=** 항목을 설정하여 여러 네트워크 인터페이스를 지정할 수 있습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

기본 게이트웨이 및 경로 구성

선택 사항: **rd.route=** 값을 설정하여 추가 네트워크에 대한 경로를 구성할 수 있습니다.



참고

하나 이상의 네트워크를 구성할 때 하나의 기본 게이트웨이가 필요합니다. 추가 네트워크 게이트웨이가 기본 네트워크 게이트웨이와 다른 경우 기본 게이트웨이가 기본 네트워크 게이트웨이어야 합니다.

- 다음 명령을 실행하여 기본 게이트웨이를 구성합니다.

```
ip=::10.10.10.254::::
```

- 다음 명령을 입력하여 추가 네트워크의 경로를 구성합니다.

```
rd.route=20.20.20.0/24:20.20.20.254:enp2s0
```

단일 인터페이스에서 **DHCP** 비활성화

두 개 이상의 네트워크 인터페이스가 있고 하나의 인터페이스만 사용되는 경우와 같이 단일 인터페이스에서 **DHCP**를 비활성화할 수 있습니다. 이 예에서 **enp1s0** 인터페이스에는 정적 네트워킹 구성이 있으며 **enp2s0** 용으로 **DHCP**가 사용되지 않습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=::::core0.example.com:enp2s0:none
```

DHCP 및 고정 **IP** 구성 결합

다음과 같이 시스템의 **DHCP** 및 고정 **IP** 구성을 여러 네트워크 인터페이스와 결합할 수 있습니다.

```
ip=enp1s0:dhcp
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

개별 인터페이스에서 **VLAN** 구성

선택 사항: **vlan=** 매개변수를 사용하여 개별 인터페이스에서 **VLAN**을 구성할 수 있습니다.

- 네트워크 인터페이스에서 **VLAN**을 구성하고 고정 **IP** 주소를 사용하려면 다음 명령을 실행합니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0.100:none
vlan=enp2s0.100:enp2s0
```

- 네트워크 인터페이스에서 **VLAN**을 구성하고 **DHCP**를 사용하려면 다음 명령을 실행합니다.

```
ip=enp2s0.100:dhcp
vlan=enp2s0.100:enp2s0
```

여러 **DNS** 서버 제공

각 서버에 **nameserver=** 항목을 추가하여 여러 **DNS** 서버를 제공할 수 있습니다. 예를 들면 다음과 같습니다.

```
nameserver=1.1.1.1
nameserver=8.8.8.8
```

단일 인터페이스에 여러 네트워크 인터페이스 본딩

선택 사항: **bond=** 옵션을 사용하여 여러 네트워크 인터페이스를 단일 인터페이스에 결합할 수 있습니다. 다음 예제를 참조하십시오.

- 결합된 인터페이스를 구성하는 구문: **bond = name [: network_interfaces] [: options]**

  name은 결합하는 기기 이름(**bond0**)이고 ***network_interfaces***는 쉼표로 구분되는 물리적(이더넷) 인터페이스 목록(**em1, em2**)이며, ***options***은 쉼표로 구분되는 결합 옵션 목록입니다. 사용 가능한 옵션을 보려면 **modinfo bonding**을 입력하십시오.

- **bond=**를 사용하여 결합된 인터페이스를 생성할 때 **IP** 주소가 할당되는 방법과 결합된 인터페이스에 대한 기타 정보를 지정해야 합니다.

- **DHCP**를 사용하도록 결합된 인터페이스를 구성하려면 **bond**의 **IP** 주소를 **dhcp**로 설정하십시오. 예를 들면 다음과 같습니다.

```
bond=bond0:em1,em2:mode=active-backup
ip=bond0:dhcp
```

- 고정 **IP** 주소를 사용하도록 결합된 인터페이스를 구성하려면 원하는 특정 **IP** 주소 및 관련 정보를 입력합니다. 예를 들면 다음과 같습니다.

```
bond=bond0:em1,em2:mode=active-backup
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
```

단일 인터페이스에 여러 네트워크 인터페이스 본딩

선택 사항: **vlan=** 매개변수를 사용하고 **DHCP**를 사용하도록 결합된 인터페이스에서 **VLAN**을 구성할 수 있습니다. 예를 들면 다음과 같습니다.

```
ip=bond0.100:dhcp
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

다음 예제를 사용하여 **VLAN**을 사용하여 결합된 인터페이스를 구성하고 고정 **IP** 주소를 사용합니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0.100:none
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

네트워크 팀 구성 사용

선택 사항: **team=** 매개변수를 사용하여 네트워크 팀을 결합의 대안으로 사용할 수 있습니다.

- 팀 인터페이스를 구성하는 구문은 team=name[:network_interfaces]입니다.

  name 은 팀 장치 이름(team0)이고 *network_interfaces* 는 쉼표로 구분된 실제(이더넷) 인터페이스 목록(em1, em2)을 나타냅니다.

RHCOS가 향후 RHEL 버전으로 전환되면 티밍이 더 이상 사용되지 않을 예정입니다. 자세한 내용은 이 Red Hat 지식베이스 문서를 참조하십시오.

다음 예제를 사용하여 네트워크 팀을 구성합니다.

```
team=team0:em1,em2
ip=team0:dhcp
```

14.2.12.2. PXE 부팅을 사용하여 RHCOS 설치

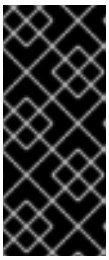
PXE 부팅을 사용하여 시스템에 RHCOS를 설치할 수 있습니다.

사전 요구 사항

- 클러스터에 대한 Ignition 구성 파일이 생성되어 있습니다.
- 적합한 네트워크, DNS 및 로드 밸런싱 인프라가 구성되어 있습니다.
- 적합한 PXE 인프라가 구성되어 있습니다.
- 사용자 컴퓨터에서 액세스할 수 있고 사용자가 생성한 시스템에서 액세스할 수 있는 HTTP 서버에 대한 액세스 권한을 확보합니다.
- 네트워킹 및 디스크 파티션과 같은 기능을 구성하는 다양한 방법에 대해서는 *고급 RHCOS 설치 구성*섹션을 살펴보십시오.

프로세스

1. 설치 프로그램에서 생성한 부트스트랩, 컨트롤 플레인, 컴퓨팅 노드 Ignition 구성 파일을 HTTP 서버에 업로드합니다. 해당 파일의 URL을 기록해 둡니다.



중요

HTTP 서버에 저장하기 전에 Ignition 구성에서 구성 설정을 추가하거나 변경할 수 있습니다. 설치를 마친 후 클러스터에 컴퓨팅 시스템을 더 추가하려면 Ignition 구성 파일을 삭제하지 마십시오.

2. 설치 호스트에서 Ignition 구성 파일을 URL에서 사용할 수 있는지 확인합니다. 다음 예에서는 부트스트랩 노드에 대한 Ignition 구성 파일을 가져옵니다.

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

출력 예

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-
rsa...
```

명령에서 bootstrap.ign을 master.ign 또는 worker.ign으로 교체하여 컨트롤 플레인 및 컴퓨팅 노드의 Ignition 구성 파일도 사용할 수 있는지 확인합니다.

3. RHCOS 커널,initramfs 및 rootfs 파일을 RHCOS 이미지 미러페이지에서 원하는 운영 체제 인스턴스 설치 방법에 사용할 수 있지만 RHCOS 파일의 올바른 버전을 얻는 것이 openshift-install 명령 출력에서 제공됩니다.

```
$ openshift-install coreos print-stream-json | grep -Eo '"https.*(kernel-
|initramfs.|rootfs.)\w+(\.img)?"'
```

출력 예

```
"<url>/art/storage/releases/rhcos-4.9-aarch64/<release>/aarch64/rhcos-<release>-live-
kernel-aarch64"
"<url>/art/storage/releases/rhcos-4.9-aarch64/<release>/aarch64/rhcos-<release>-live-
```

```
initramfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.9-aarch64/<release>/aarch64/rhcos-<release>-live-
rootfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.9-ppc64le/49.84.202110081256-0/ppc64le/rhcos-
<release>-live-kernel-ppc64le"
"<url>/art/storage/releases/rhcos-4.9-ppc64le/<release>/ppc64le/rhcos-<release>-live-
initramfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.9-ppc64le/<release>/ppc64le/rhcos-<release>-live-
rootfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.9-s390x/<release>/s390x/rhcos-<release>-live-
kernel-s390x"
"<url>/art/storage/releases/rhcos-4.9-s390x/<release>/s390x/rhcos-<release>-live-
initramfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.9-s390x/<release>/s390x/rhcos-<release>-live-
rootfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.9/<release>/x86_64/rhcos-<release>-live-kernel-
x86_64"
"<url>/art/storage/releases/rhcos-4.9/<release>/x86_64/rhcos-<release>-live-
initramfs.x86_64.img"
"<url>/art/storage/releases/rhcos-4.9/<release>/x86_64/rhcos-<release>-live-
rootfs.x86_64.img"
```

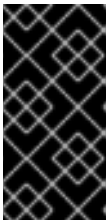
중요

OpenShift Container Platform의 모든 릴리스에서 **RHCOS** 아티팩트가 변경되지 않을 수도 있습니다. 설치하는 **OpenShift Container Platform** 버전과 같거나 그 이하의 버전 중 가장 최신 버전의 이미지를 다운로드해야 합니다. 이 프로시저에는 아래 설명된 적절한 **kernel, initramfs** 및 **rootfs** 아티팩트만 사용하십시오. 이 설치 유형에서는 **RHCOS QCOW2** 이미지가 지원되지 않습니다.

OpenShift Container Platform 버전 번호가 파일 이름에 포함됩니다. 다음 예와 유사합니다.

- **kernel: rhcos-<version>-live-kernel-<architecture>**
- **initramfs: rhcos-<version>-live-initramfs.<architecture>.img**
- **rootfs: rhcos-<version>-live-rootfs.<architecture>.img**

4. rootfs, kernel 및 initramfs 파일을 HTTP 서버에 업로드합니다.



중요

설치를 마친 후 클러스터에 컴퓨팅 시스템을 더 추가하려면 Ignition 구성 파일을 삭제하지 마십시오.

5. RHCOS가 설치된 후 시스템이 로컬 디스크에서 부팅되도록 네트워크 부팅 인프라를 구성하십시오.

6. RHCOS 이미지에 대한 PXE 설치를 구성하고 설치를 시작하십시오.

사용 환경에 대한 다음 예시 메뉴 항목을 수정하고, 이미지 및 Ignition 파일에 적절히 접근할 수 있는지 확인하십시오.

```
DEFAULT pxeboot
TIMEOUT 20
PROMPT 0
LABEL pxeboot
    KERNEL http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> (1)
    APPEND initrd=http://<HTTP_server>/rhcos-<version>-live-initramfs.<architecture>.img
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.<architecture>.img
coreos.inst.install_dev=/dev/sda coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign
(2) (3)
```

(1) (1)

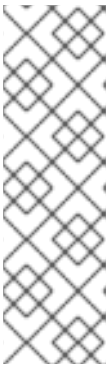
HTTP 서버에 업로드한 라이브 kernel 파일의 위치를 지정합니다. URL은 HTTP, TFTP 또는 FTP여야 합니다. HTTPS와 NFS는 지원되지 않습니다.

(2)

NIC를 여러 개 사용하는 경우 ip 옵션에 단일 인터페이스를 지정합니다. 예를 들어, eno1라는 NIC에서 DHCP를 사용하려면 ip=eno1:dhcp를 설정하십시오.

(3)

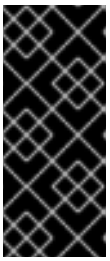
HTTP 서버에 업로드한 RHCOS 파일의 위치를 지정합니다. initrd 매개변수 값은 initramfs 파일의 위치, coreos.live.rootfs_url 매개변수 값은 rootfs 파일의 위치, coreos.inst.ignition_url 매개변수 값은 부트스트랩 Ignition 구성 파일의 위치입니다. APPEND 줄에 커널 인수를 더 추가하여 네트워킹 또는 기타 부팅 옵션도 구성할 수 있습니다.



참고

이 구성은 그래픽 콘솔이 있는 머신에서 직렬 콘솔 액세스를 활성화하지 않습니다. 다른 콘솔을 구성하려면 **APPEND** 행에 하나 이상의 **console=** 인수를 추가합니다. 예를 들어 **console=tty0 console=ttyS0**을 추가하여 첫 번째 PC 직렬 포트를 기본 콘솔로 설정하고 그래픽 콘솔을 보조 콘솔로 설정합니다. 자세한 내용은 Red Hat Enterprise Linux에서 직렬 터미널 및/또는 콘솔 설정 방법을 참조하십시오.

7. 머신 콘솔에서 **RHCOS** 설치 진행률을 모니터링합니다.

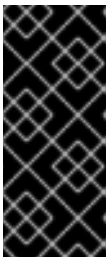


중요

OpenShift Container Platform 설치를 시작하기 전에 각 노드에서 성공적으로 설치되었는지 확인합니다. 설치 프로세스를 관찰하면 발생할 수 있는 **RHCOS** 설치 문제의 원인을 파악하는 데 도움이 될 수 있습니다.

8. **RHCOS**가 설치되면 시스템을 재부팅합니다. 시스템이 재부팅되는 동안 지정한 **Ignition** 구성 파일이 적용됩니다.

9. 클러스터용 시스템 생성을 계속합니다.



중요

이때 부트스트랩 및 컨트롤 플레인 시스템을 생성해야 합니다. 컨트롤 플레인 시스템이 예약 가능하지 않은 경우 클러스터를 설치하기 전에 두 개 이상의 컴퓨팅 시스템도 생성합니다.

적합한 네트워킹, **DNS** 및 로드 밸런싱 인프라가 있는 경우 **RHCOS** 노드가 재부팅된 후 **OpenShift Container Platform** 부트스트랩 프로세스가 자동으로 시작됩니다.

참고

RHCOS 노드에는 core 사용자의 기본 암호가 포함되지어 있지 않습니다. install_config.yaml 파일에 지정한 공개 키에 대한 액세스 권한이 있는 사용자로 ssh core@<node>.<cluster_name>.<base_domain>을 실행하여 노드에 액세스할 수 있습니다. RHCOS를 실행하는 OpenShift Container Platform 4 클러스터 노드는 변경할 수 없으며 Operator를 통해 클러스터 변경 사항을 적용합니다. SSH를 사용하여 클러스터 노드에 액세스하는 것은 권장되지 않습니다. 그러나 설치 문제를 조사할 때 OpenShift Container Platform API를 사용할 수 없거나 kubelet이 대상 노드에서 제대로 작동하지 않는 경우 디버깅 또는 재해 복구에 SSH 액세스가 필요할 수 있습니다.

14.2.12.3. RHCOS에서 커널 인수로 다중 경로 활성화

OpenShift Container Platform 4.9 이상에서는 설치 중에 프로비저닝된 노드에 대해 멀티패스를 활성화할 수 있습니다. RHCOS는 기본 디스크에서 멀티패스를 지원합니다. 멀티패스는 호스트 가용성을 향상시키기 위해 하드웨어 장애에 대한 복원력 강화의 추가 이점을 제공합니다.

초기 클러스터 생성 중에 모든 마스터 또는 작업자 노드에 커널 인수를 추가할 수 있습니다. 마스터 노드 또는 작업자 노드에 커널 매개 변수를 추가하기 위해 MachineConfig 객체를 생성하고 해당 객체를 클러스터 설정 중에 Ignition에서 사용하는 매니페스트 파일 세트에 삽입할 수 있습니다.

프로세스

1. 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터에 대한 Kubernetes 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory>
```

2. 커널 매개 변수를 작업자 또는 컨트롤 플레인 노드에 추가할지 여부를 결정합니다.

- 머신 구성 파일을 생성합니다. 예를 들어 클러스터에 master 레이블을 추가하고 멀티패스 커널 인수를 식별하도록 지시하는 99-master-kargs-mpath.yaml을 생성합니다.

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: "master"
  name: 99-master-kargs-mpath
spec:
```

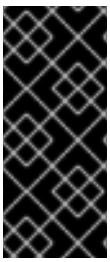
```
  kernelArguments:
    - 'rd.multipath=default'
    - 'root=/dev/disk/by-label/dm-mpath-root'
```

3. 작업자 노드에서 멀티패스 설치 후 활성화하려면 다음을 수행합니다.

- 머신 구성 파일을 생성합니다. 예를 들어 클러스터에 **worker** 레이블을 추가하고 멀티패스 커널 인수를 식별하도록 지시하는 **99-worker-kargs-mpath.yaml** 을 생성합니다.

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: "worker"
  name: 99-worker-kargs-mpath
spec:
  kernelArguments:
    - 'rd.multipath=default'
    - 'root=/dev/disk/by-label/dm-mpath-root'
```

이제 계속해서 클러스터를 만들 수 있습니다.



중요

멀티패스를 완전히 활성화하려면 추가 설치 후 단계가 필요합니다. 자세한 내용은 *설치 후 머신 구성 작업*의 **"RHCOS에서 커널 인수를 사용하여 다중 경로 활성화"**를 참조하십시오.

MPIO 오류가 발생하는 경우 **bootlist** 명령을 사용하여 대체 논리 장치 이름으로 부팅 장치 목록을 업데이트합니다. 명령은 부팅 목록을 표시하고 시스템을 일반 모드로 부팅할 때 에 대해 가능한 부팅 장치를 지정합니다.

a. 부팅 목록을 표시하고 시스템이 일반 모드로 부팅되는 경우 가능한 부팅 장치를 지정하도록 다음 명령을 입력합니다.

```
$ bootlist -m normal -o
sda
```

b. 일반 모드의 부팅 목록을 업데이트하고 대체 장치 이름을 추가하려면 다음 명령을 입력합니다.

-

```
$ bootlist -m normal -o /dev/sdc /dev/sdd /dev/sde
sdc
sdd
sde
```

원래 부팅 디스크 경로가 다운되면 노드가 일반 부팅 장치 목록에 등록된 대체 장치에서 재부팅됩니다.

14.2.13. 부트스트랩 프로세스가 완료될 때까지 대기 중

OpenShift Container Platform 부트스트랩 프로세스는 클러스터 노드가 먼저 디스크에 설치된 영구 RHCOS 환경으로 부팅된 후에 시작됩니다. Ignition 구성 파일을 통해 제공되는 구성 정보는 부트스트랩 프로세스를 초기화하고 머신에 OpenShift Container Platform을 설치하는 데 사용됩니다. 부트스트랩 프로세스가 완료될 때까지 기다려야 합니다.

사전 요구 사항

- 클러스터에 대한 Ignition 구성 파일이 생성되어 있습니다.
- 적합한 네트워크, DNS 및 로드 밸런싱 인프라가 구성되어 있습니다.
- 설치 프로그램을 받아서 클러스터의 Ignition 구성 파일을 생성했습니다.
- 클러스터 머신에 RHCOS를 설치하고 OpenShift Container Platform 설치 프로그램에서 생성된 Ignition 구성 파일을 제공했습니다.
- 사용자 시스템에서 직접 인터넷에 액세스하거나 HTTP 또는 HTTPS 프록시를 사용할 수 있습니다.

절차

1. 부트스트랩 프로세스를 모니터링합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
  --log-level=info 2
```

1

2

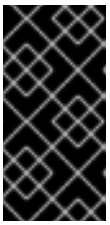
다른 설치 세부 사항을 보려면 **info** 대신 **warn, debug** 또는 **error**를 지정합니다.

출력 예

```
INFO Waiting up to 30m0s for the Kubernetes API at
https://api.test.example.com:6443...
INFO API v1.22.1 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

이 명령은 **Kubernetes API** 서버가 컨트롤 플레인 시스템에서 부트스트랩되었다는 신호를 보낼 때 성공합니다.

2. 부트스트랩 프로세스가 완료된 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다.



중요

이 시점에 로드 밸런서에서 부트스트랩 시스템을 제거해야 합니다. 부트스트랩 머신 자체를 제거하거나 다시 포맷할 수도 있습니다.

14.2.14. CLI를 사용하여 클러스터에 로그인

클러스터 **kubeconfig** 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. **kubeconfig** 파일에는 **CLI**에서 올바른 클러스터 및 **API** 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 **OpenShift Container Platform** 설치 과정에서 생성됩니다.

사전 요구 사항

- **OpenShift Container Platform** 클러스터를 배포했습니다.

- oc CLI를 설치했습니다.

절차

1. kubeadmin 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 oc 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

14.2.15. 머신의 인증서 서명 요청 승인

클러스터에 시스템을 추가하면 추가한 시스템별로 보류 중인 인증서 서명 요청(CSR)이 두 개씩 생성됩니다. 이러한 CSR이 승인되었는지 확인해야 하며, 필요한 경우 이를 직접 승인해야 합니다. 클라이언트 요청을 먼저 승인한 다음 서버 요청을 승인해야 합니다.

사전 요구 사항

- 클러스터에 시스템을 추가했습니다.

절차

1. 클러스터가 시스템을 인식하는지 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0  Ready    master  63m  v1.22.1
master-1  Ready    master  63m  v1.22.1
master-2  Ready    master  64m  v1.22.1
```

출력에 생성된 모든 시스템이 나열됩니다.

참고

이전 출력에는 일부 **CSR**이 승인될 때까지 컴퓨팅 노드(작업자 노드라고도 함)가 포함되지 않을 수 있습니다.

2. 보류 중인 **CSR**을 검토하고 클러스터에 추가한 각 시스템에 대해 **Pending** 또는 **Approved** 상태의 클라이언트 및 서버 요청이 표시되는지 확인합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE    REQUESTOR                                                   CONDITION
csr-8b2br   15m    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

예에서는 두 시스템이 클러스터에 참여하고 있습니다. 목록에는 승인된 **CSR**이 더 많이 나타날 수도 있습니다.

3. CSR이 승인되지 않은 경우, 추가된 시스템에 대한 모든 보류 중인 CSR이 Pending 상태로 전환된 후 클러스터 시스템의 CSR을 승인합니다.

참고

CSR은 교체 주기가 자동으로 만료되므로 클러스터에 시스템을 추가한 후 1 시간 이내에 CSR을 승인하십시오. 한 시간 내에 승인하지 않으면 인증서가 교체되고 각 노드에 대해 두 개 이상의 인증서가 표시됩니다. 이러한 인증서를 모두 승인해야 합니다. 클라이언트 CSR이 승인되면 Kubelet은 인증서에 대한 보조 CSR을 생성하므로 수동 승인이 필요합니다. 그러면 Kubelet에서 동일한 매개변수를 사용하여 새 인증서를 요청하는 경우 인증서 갱신 요청은 machine-approver에 의해 자동으로 승인됩니다.

참고

베어 메탈 및 기타 사용자 프로비저닝 인프라와 같이 머신 API를 사용하도록 활성화되지 않는 플랫폼에서 실행되는 클러스터의 경우 CSR(Kubelet service Certificate Request)을 자동으로 승인하는 방법을 구현해야 합니다. 요청이 승인되지 않으면 API 서버가 kubelet에 연결될 때 서비스 인증서가 필요하므로 oc exec, oc rsh, oc logs 명령을 성공적으로 수행할 수 없습니다. Kubelet 엔드 포인트에 연결하는 모든 작업을 수행하려면 이 인증서 승인이 필요합니다. 이 방법은 새 CSR을 감시하고 CSR이 system:node 또는 system:admin 그룹의 node-bootstrapper 서비스 계정에 의해 제출되었는지 확인하고 노드의 ID를 확인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 <csr_name>은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

참고

일부 Operator는 일부 CSR이 승인될 때까지 사용할 수 없습니다.

4. 이제 클라이언트 요청이 승인되었으므로 클러스터에 추가한 각 머신의 서버 요청을 검토해야 합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal        Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal        Pending
...
```

5. 나머지 CSR이 승인되지 않고 Pending 상태인 경우 클러스터 머신의 CSR을 승인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 <csr_name>은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

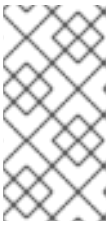
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs oc adm certificate approve
```

6. 모든 클라이언트 및 서버 CSR이 승인된 후 머신은 Ready 상태가 됩니다. 다음 명령을 실행하여 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0  Ready    master  73m  v1.22.1
master-1  Ready    master  73m  v1.22.1
master-2  Ready    master  74m  v1.22.1
worker-0  Ready    worker  11m  v1.22.1
worker-1  Ready    worker  11m  v1.22.1
```



참고

머신이 **Ready** 상태로 전환하는 데 서버 **CSR**의 승인 후 몇 분이 걸릴 수 있습니다.

추가 정보

- **CSR**에 대한 자세한 내용은 인증서 서명 요청을 참조하십시오.

14.2.16. Operator의 초기 설정

컨트롤 플레인이 초기화된 후 일부 **Operator**를 즉시 구성하여 모두 사용 가능하도록 해야 합니다.

사전 요구 사항

- 컨트롤 플레인이 초기화되어 있습니다.

프로세스

1. 클러스터 구성 요소가 온라인 상태인지 확인합니다.

```
$ watch -n5 oc get clusteroperators
```

출력 예

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.9.0     True        False         False      19m
baremetal                                  4.9.0     True        False         False      37m
cloud-credential                           4.9.0     True        False         False      40m
cluster-autoscaler                         4.9.0     True        False         False      37m
config-operator                            4.9.0     True        False         False      38m
console                                    4.9.0     True        False         False      26m
csi-snapshot-controller                    4.9.0     True        False         False      37m
dns                                        4.9.0     True        False         False      37m
etcd                                       4.9.0     True        False         False      36m
image-registry                             4.9.0     True        False         False      31m
ingress                                    4.9.0     True        False         False      30m
insights                                   4.9.0     True        False         False      31m
kube-apiserver                             4.9.0     True        False         False      26m
kube-controller-manager                    4.9.0     True        False         False      36m
kube-scheduler                             4.9.0     True        False         False      36m
kube-storage-version-migrator              4.9.0     True        False         False      37m
machine-api                                4.9.0     True        False         False      29m
machine-approver                           4.9.0     True        False         False      37m
machine-config                             4.9.0     True        False         False      36m
marketplace                                4.9.0     True        False         False      37m
monitoring                                 4.9.0     True        False         False      29m
network                                    4.9.0     True        False         False      38m
node-tuning                                4.9.0     True        False         False      37m
openshift-apiserver                        4.9.0     True        False         False      32m
openshift-controller-manager               4.9.0     True        False         False      30m
openshift-samples                          4.9.0     True        False         False      32m
operator-lifecycle-manager                 4.9.0     True        False         False      37m
operator-lifecycle-manager-catalog         4.9.0     True        False         False      37m
operator-lifecycle-manager-packageserver   4.9.0     True        False         False      32m
service-ca                                 4.9.0     True        False         False      38m
storage                                    4.9.0     True        False         False      37m
```

2. 사용할 수 없는 **Operator**를 구성합니다.

14.2.16.1. 이미지 레지스트리 스토리지 구성

기본 스토리지를 제공하지 않는 플랫폼에서는 처음에 **Image Registry Operator**를 사용할 수 없습니다. 설치한 후에 스토리지를 사용하도록 레지스트리를 구성하여 **Registry Operator**를 사용 가능하도록 만들어야 합니다.

프로덕션 클러스터에 필요한 영구 볼륨을 구성하는 과정의 지침이 표시됩니다. 해당하는 경우, 프로덕션 환경 외 클러스터에서만 사용할 수 있는 저장 위치로서 빈 디렉터리를 구성하는 과정의 지침이 표시됩니다.

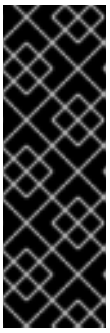
업그레이드 중에 **Recreate** 롤아웃 전략을 사용하여 이미지 레지스트리의 블록 스토리지 유형 사용을 허용하기 위한 추가 지침이 제공됩니다.

14.2.16.1.1. IBM Power 용 레지스트리 스토리지 구성

클러스터 관리자는 설치한 후 스토리지를 사용하도록 레지스트리를 구성해야 합니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- **IBM Power**에 클러스터가 있습니다.
- **Red Hat OpenShift Container Storage**와 같이 클러스터용 영구 스토리지 프로비저닝이 있습니다.



중요

OpenShift Container Platform은 복제본이 하나만 있는 경우 이미지 레지스트리 스토리지에 대한 **ReadWriteOnce** 액세스를 지원합니다. **ReadWriteOnce** 액세스에서는 레지스트리가 **Recreate** 롤아웃 전략을 사용해야 합니다. 두 개 이상의 복제본으로 고 가용성을 지원하는 이미지 레지스트리를 배포하려면 **ReadWriteMany** 액세스가 필요합니다.

- **"100Gi"** 용량이 필요합니다.

프로세스

1. 스토리지를 사용하도록 레지스트리를 구성하기 위해 **configs.imageregistry/cluster** 리소스에서 **spec.storage.pvc**를 변경합니다.

참고

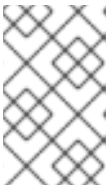
공유 스토리지를 사용할 때 보안 설정을 확인하여 외부에서의 액세스를 방지합니다.

2. 레지스트리 **pod**가 없는지 확인합니다.

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

출력 예

```
No resourses found in openshift-image-registry namespace
```



참고

출력에 레지스트리 **Pod**가 있는 경우 이 절차를 계속할 필요가 없습니다.

3. 레지스트리 구성을 확인합니다.

```
$ oc edit configs.imageregistry.operator.openshift.io
```

출력 예

```
storage:
  pvc:
    claim:
```

image-registry-storage PVC의 자동 생성을 허용하도록 **claim** 필드를 비워 둡니다.

4. clusteroperator 상태를 확인합니다.

```
$ oc get clusteroperator image-registry
```

출력 예

```
NAME             VERSION       AVAILABLE   PROGRESSING   DEGRADED   SINCE
MESSAGE
image-registry   4.9           True        False         False      6h50m
```

5. 이미지를 빌드 및 푸시할 수 있도록 레지스트리의 관리가 설정되어 있는지 확인하십시오.

- 다음을 실행합니다.

```
$ oc edit configs.imageregistry/cluster
```

다음으로 라인을 변경하십시오.

```
managementState: Removed
```

다음으로 변경

```
managementState: Managed
```

14.2.16.1.2. 프로덕션 환경 외 클러스터에서 이미지 레지스트리의 스토리지 구성

이미지 레지스트리 **Operator**에 대한 스토리지를 구성해야 합니다. 프로덕션 환경 외 클러스터의 경우, 이미지 레지스트리를 빈 디렉터리로 설정할 수 있습니다. 이렇게 하는 경우 레지스트리를 다시 시작하면 모든 이미지가 손실됩니다.

프로세스

- 이미지 레지스트리 스토리지를 빈 디렉터리로 설정하려면 다음을 수행하십시오.

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch
'{"spec":{"storage":{"emptyDir":{}}}}'
```

주의

프로덕션 환경 외 클러스터에 대해서만 이 옵션을 구성하십시오.

Image Registry Operator가 구성 요소를 초기화하기 전에 이 명령을 실행하면 oc patch 명령이 실패하며 다음 오류가 발생합니다.

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster"
not found
```

몇 분 후에 명령을 다시 실행하십시오.

14.2.17. 사용자 프로비저닝 인프라에 설치 완료

Operator 구성을 완료한 후 제공하는 인프라에 클러스터 설치를 완료할 수 있습니다.

사전 요구 사항

- 컨트롤 플레인이 초기화되어 있습니다.
- 초기 Operator 구성을 완료해야 합니다.

프로세스

1. 다음 명령을 사용하여 모든 클러스터 구성 요소가 온라인 상태인지 확인합니다.

```
$ watch -n5 oc get clusteroperators
```

출력 예

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.9.0     True        False         False      19m
baremetal                                  4.9.0     True        False         False      37m
cloud-credential                           4.9.0     True        False         False      40m
cluster-autoscaler                         4.9.0     True        False         False      37m
config-operator                            4.9.0     True        False         False      38m
console                                    4.9.0     True        False         False      26m
csi-snapshot-controller                    4.9.0     True        False         False      37m
dns                                        4.9.0     True        False         False      37m
etcd                                       4.9.0     True        False         False      36m
image-registry                             4.9.0     True        False         False      31m
ingress                                    4.9.0     True        False         False      30m
insights                                   4.9.0     True        False         False      31m
kube-apiserver                             4.9.0     True        False         False      26m
kube-controller-manager                    4.9.0     True        False         False      36m
kube-scheduler                             4.9.0     True        False         False      36m
kube-storage-version-migrator              4.9.0     True        False         False      37m
machine-api                                4.9.0     True        False         False      29m
machine-approver                           4.9.0     True        False         False      37m
machine-config                             4.9.0     True        False         False      36m
marketplace                                4.9.0     True        False         False      37m
monitoring                                 4.9.0     True        False         False      29m
network                                    4.9.0     True        False         False      38m
node-tuning                                4.9.0     True        False         False      37m
openshift-apiserver                        4.9.0     True        False         False      32m
openshift-controller-manager               4.9.0     True        False         False      30m
openshift-samples                          4.9.0     True        False         False      32m
operator-lifecycle-manager                 4.9.0     True        False         False      37m
operator-lifecycle-manager-catalog         4.9.0     True        False         False      37m
operator-lifecycle-manager-packageserver   4.9.0     True        False         False      32m
service-ca                                 4.9.0     True        False         False      38m
storage                                    4.9.0     True        False         False      37m
```

또는 다음 명령은 모든 클러스터를 사용할 수 있을 때 알립니다. 또한 인증 정보를 검색하고 표시합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1

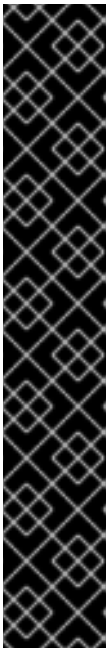
<installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

출력 예

-

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator가 Kubernetes API 서버에서 OpenShift Container Platform 클러스터 배포를 완료하면 명령이 성공합니다.



중요

- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 node-bootstrapper 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *만료된 컨트롤 플레인 인증서에서 복구* 문서를 참조하십시오.

- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

2. Kubernetes API 서버가 Pod와 통신하고 있는지 확인합니다.

   a. 모든 Pod 목록을 보려면 다음 명령을 사용하십시오.

```
$ oc get pods --all-namespaces
```

출력 예

```
NAMESPACE                  NAME                                        READY   STATUS
RESTARTS   AGE
openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8
1/1    Running    1        9m
openshift-apiserver        apiserver-67b9g                             1/1     Running    0
3m
openshift-apiserver        apiserver-ljcmx                             1/1     Running    0
1m
openshift-apiserver        apiserver-z25h4                             1/1     Running    0
2m
```

```
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8
1/1     Running     0          5m
...
```

b. 다음 명령을 사용하여 이전 명령의 출력에 나열된 **Pod**의 로그를 표시합니다.

```
$ oc logs <pod_name> -n <namespace> 1
```

1 이전 명령의 출력에 표시된 대로 **Pod** 이름과 네임스페이스를 지정합니다.

Pod 로그가 표시되면 **Kubernetes API** 서버는 클러스터 시스템과 통신할 수 있습니다.

3. 다중 경로를 활성화하기 위해 추가 단계가 필요합니다. 설치 중에 멀티패스를 활성화하지 마십시오.

자세한 내용은 *설치 후 머신 구성 작업* 설명서에서 **"RHCOS에서 커널 인수를 사용하여 멀티패스 활성화"**를 참조하십시오.

14.2.18. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 **Telemetry** 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 **Telemetry**가 자동으로 실행되고 **OpenShift Cluster Manager** 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 **OpenShift Cluster Manager**를 사용하여 자동으로 또는 **OpenShift Cluster Manager**를 사용하여 수동으로 유지 관리되는지 확인한 후 **subscription watch**를 사용하여 계정 또는 다중 클러스터 수준에서 **OpenShift Container Platform** 서브스크립션을 추적합니다.

추가 리소스

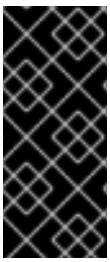
- **Telemetry** 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

14.2.19. 다음 단계

- RHCOS에서 커널 인수를 사용하여 멀티패스 활성화
- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.

14.3. 네트워크가 제한된 환경에서 IBM POWER에 클러스터 설치

OpenShift Container Platform 버전 4.9에서는 네트워크가 제한된 환경에서 사용자가 제공하는 IBM Power 인프라에 클러스터를 설치할 수 있습니다.



중요

베어 메탈 이외의 플랫폼의 경우 추가 고려 사항이 있습니다. OpenShift Container Platform 클러스터를 설치하기 전에 guidelines for deploying OpenShift Container Platform on non-tested platforms에 있는 내용을 확인하십시오.

14.3.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.
- 네트워크가 제한된 환경에서 설치 미러 레지스트리를 생성하고 OpenShift Container Platform 버전의 imageContentSources 데이터를 가져옵니다.
- 설치 프로세스를 시작하기 전에 기존 설치 파일을 이동하거나 제거해야 합니다. 이렇게 하면 설치 프로세스 중에 필요한 설치 파일이 생성되고 업데이트됩니다.

중요

설치 미디어에 액세스할 수있는 시스템에서 설치 프로세스를 수행해야 합니다.

- 클러스터용 영구 스토리지를 프로비저닝합니다. 프라이빗 이미지 레지스트리를 배포하려면 스토리지에서 **ReadWriteMany** 액세스 모드를 제공해야 합니다.
- 방화벽을 사용하며 **Telemetry** 서비스를 사용할 예정인 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성해야 합니다.

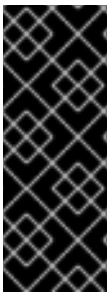
참고

프록시를 구성하는 경우에도 해당 사이트 목록을 검토하십시오.

14.3.2. 네트워크가 제한된 환경에서의 설치 정보

OpenShift Container Platform 4.9에서는 소프트웨어 구성 요소를 받기 위한 인터넷 접속이 필요하지 않은 설치를 수행할 수 있습니다. 제한된 네트워크 설치는 클러스터를 설치하는 클라우드 플랫폼에 따라 설치 관리자 프로비저닝 인프라 또는 사용자 프로비저닝 인프라를 사용하여 완료할 수 있습니다.

제한된 네트워크 설치를 완료하려면 **OpenShift Container Platform** 레지스트리의 내용을 미러링하고 설치 미디어를 포함할 레지스트리를 생성해야 합니다. 인터넷과 폐쇄 네트워크에 모두 액세스하거나 제한 사항을 따르는 다른 방법을 통해 미러 호스트에 레지스트리를 생성할 수 있습니다.



중요

사용자 프로비저닝 설치의 구성이 복잡하므로 사용자 프로비저닝 인프라를 사용하여 제한된 네트워크 설치를 시도하기 전에 표준 사용자 프로비저닝 인프라 설치를 완료하는 것이 좋습니다. 이 테스트 설치를 완료하면 제한된 네트워크에 설치하는 동안 발생할 수 있는 문제를 보다 쉽게 파악 및 해결할 수 있습니다.

14.3.2.1. 추가 제한

제한된 네트워크의 클러스터에는 다음과 같은 추가 제한이 있습니다.

- ClusterVersion 상태에 사용 가능한 업데이트를 검색할 수 없음 오류가 포함되어 있습니다.

- 기본적으로 필요한 이미지 스트림 태그에 액세스할 수 없기 때문에 개발자 카탈로그의 내용을 사용할 수 없습니다.

14.3.3. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하기 위해 필요한 이미지를 받으려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.

- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.

- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.

중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

14.3.4. 사용자 프로비저닝 인프라를 포함한 클러스터의 시스템 요구사항

사용자 프로비저닝 인프라가 포함된 클러스터의 경우, 필요한 모든 시스템을 배포해야 합니다.

이 섹션에서는 사용자 프로비저닝 인프라에 OpenShift Container Platform을 배포해야 하는 요구 사항에 대해 설명합니다.

14.3.4.1. 클러스터 설치에 필요한 시스템

최소 OpenShift Container Platform 클러스터에 다음과 같은 호스트가 필요합니다.

표 14.18. 최소 필수 호스트

호스트	설명
임시 부트스트랩 시스템 한 개	컨트롤 플레인 시스템 세 개에 OpenShift Container Platform 클러스터를 배포하기 위한 부트스트랩 시스템이 클러스터에 필요합니다. 클러스터를 설치한 후 부트스트랩 시스템을 제거할 수 있습니다.
컨트롤 플레인 시스템 세 개	컨트롤 플레인 시스템은 컨트롤 플레인을 구성하는 Kubernetes 및 OpenShift Container Platform 서비스를 실행합니다.
두 개 이상의 컴퓨팅 시스템(작업자 시스템이라고도 함).	OpenShift Container Platform 사용자가 요청한 워크로드는 컴퓨팅 머신에서 실행됩니다.

중요

클러스터의 고가용성을 유지하려면 이러한 클러스터 시스템에 대해 별도의 물리적 호스트를 사용하십시오.

부트스트랩, 컨트롤 플레인 시스템은 운영 체제로 RHCOS (Red Hat Enterprise Linux CoreOS)를 사용해야 합니다. 그러나 컴퓨팅 머신은 RHCOS(Red Hat Enterprise Linux CoreOS), RHEL (Red Hat Enterprise Linux) 7.9 또는 RHEL 8.4 중에서 선택할 수 있습니다.

RHCOS는 RHEL 8(Red Hat Enterprise Linux)을 기반으로하며 모든 하드웨어 인증 및 요구사항을 모두 이어받습니다. Red Hat Enterprise Linux 기술 기능 및 제한을 참조하십시오.

14.3.4.2. 클러스터 설치를 위한 최소 리소스 요구 사항

각 클러스터 시스템이 다음과 같은 최소 요구사항을 충족해야 합니다.

표 14.19. 최소 리소스 요구사항

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
부트스트랩	RHCOS	2	16GB	100GB	300

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
컨트롤 플레인	RHCOS	2	16GB	100GB	300
컴퓨팅	RHCOS	2	8GB	100GB	300

1. **SMT(동시 멀티스레딩) 또는 하이퍼 스레딩이 활성화되지 않은 경우 하나의 vCPU는 하나의 물리적 코어와 동일합니다. 활성화하면 다음과 같은 공식을 사용하여 해당 비율을 계산합니다. (코어 당 스레드 수 × 코어 수) × 소켓 수 = vCPU 수**

2. **OpenShift Container Platform 및 Kubernetes는 디스크 성능에 민감하며 특히 10ms p99 fsync 기간이 필요한 컨트롤 플레인 노드의 etcd에 더 빠른 스토리지가 권장됩니다. 많은 클라우드 플랫폼에서 스토리지 크기와 IOPS를 함께 확장되므로 충분한 성능을 얻으려면 스토리지 볼륨을 과도하게 할당해야 할 수 있습니다.**

3. **사용자가 프로비저닝한 모든 설치와 마찬가지로 클러스터에서 RHEL 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. RHEL 7 컴퓨팅 머신 사용은 더 이상 사용되지 않으며 향후 OpenShift Container Platform 4 릴리스에서 제거될 예정입니다.**

14.3.4.3. 최소 IBM Power 요구사항

다음 IBM 하드웨어에 OpenShift Container Platform 버전 4.9를 설치할 수 있습니다.

- **IBM Power8, Power9 또는 Power10 프로세서 기반 시스템**

하드웨어 요구 사항

- **여러 PowerVM 서버에서 6개의 IBM Power 베어 메탈 서버 또는 6개의 LPAR**

운영 체제 요구 사항

- **IBM Power8, Power9 또는 Power10 프로세서 기반 시스템의 인스턴스 하나**

IBM Power 인스턴스에서 다음을 설정합니다.

- OpenShift Container Platform 컨트롤 플레인 시스템 용 게스트 가상 머신 세 개
- OpenShift Container Platform 컴퓨팅 머신 용 게스트 가상 머신 두 개
- 임시 OpenShift Container Platform 부트스트랩 시스템용 게스트 가상 머신 1개

IBM Power 게스트 가상 머신용 디스크 스토리지

- vSCSI, NPIV(N-Port ID Virtualization) 또는 SSP(공장 스토리지 풀)를 사용하여 가상 I/O 서버에서 프로비저닝한 스토리지

PowerVM 게스트 가상 머신용 네트워크

- Shared Ethernet Adapter를 사용하여 가상 I/O 서버에서 가상화
- IBM vNIC를 사용하여 가상 I/O 서버에서 가상화

스토리지 / 메인 메모리

- OpenShift Container Platform 컨트롤 플레인 시스템용 100 GB / 16 GB
- OpenShift Container Platform 컴퓨팅 머신용 100GB/8GB
- 임시 OpenShift Container Platform 부트스트랩 머신 용 100GB / 16GB

14.3.4.4. 권장되는 IBM Power 시스템 요구 사항

하드웨어 요구 사항

- 여러 PowerVM 서버에서 6개의 IBM Power 베어 메탈 서버 또는 6개의 LPAR

운영 체제 요구 사항

- IBM Power8, Power9 또는 Power10 프로세서 기반 시스템의 인스턴스 하나

IBM Power 인스턴스에서 다음을 설정합니다.

- OpenShift Container Platform 컨트롤 플레인 시스템 용 게스트 가상 머신 세 개
- OpenShift Container Platform 컴퓨팅 머신 용 게스트 가상 머신 두 개
- 임시 OpenShift Container Platform 부트스트랩 시스템용 게스트 가상 머신 1개

IBM Power 게스트 가상 머신용 디스크 스토리지

- vSCSI, NPIV(N-Port ID Virtualization) 또는 SSP(공장 스토리지 풀)를 사용하여 가상 I/O 서버에서 프로비저닝한 스토리지

PowerVM 게스트 가상 머신용 네트워크

- Shared Ethernet Adapter를 사용하여 가상 I/O 서버에서 가상화
- IBM vNIC를 사용하여 가상 I/O 서버에서 가상화

스토리지 / 메인 메모리

- OpenShift Container Platform 컨트롤 플레인 시스템용 120GB/32GB
- OpenShift Container Platform 컴퓨팅 머신용 120GB/32GB
- 임시 OpenShift Container Platform 부트스트랩 머신용 120GB/16GB

14.3.4.5. 인증서 서명 요청 관리

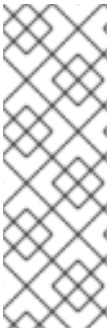
사용자가 프로비저닝하는 인프라를 사용하는 경우 자동 시스템 관리 기능으로 인해 클러스터의 액세스가 제한되므로 설치한 후 클러스터 인증서 서명 요청(CSR)을 승인하는 메커니즘을 제공해야 합니다. kube-controller-manager는 kubelet 클라이언트 CSR만 승인합니다. machine-approver는 올바른 시스템에서 발행한 요청인지 확인할 수 없기 때문에 kubelet 자격 증명을 사용하여 요청하는 서비스 인증서의 유효성을 보장할 수 없습니다. kubelet 서빙 인증서 요청의 유효성을 확인하고 요청을 승인하는 방법을 결정하여 구현해야 합니다.

14.3.4.6. 사용자 프로비저닝 인프라에 대한 네트워킹 요구사항

모든 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템이 부팅 중에 Ignition 구성 파일을 가져오려면 initramfs에 네트워킹을 구성해야 합니다.

초기 부팅 과정에서 시스템에 필요한 부팅 옵션을 제공하여 DHCP 서버를 통해 설정하거나 정적으로 설정하는 IP 주소 구성이 필요합니다. 네트워크 연결이 설정된 후 시스템은 HTTP 또는 HTTPS 서버에서 Ignition 구성 파일을 다운로드합니다. 그런 다음 Ignition 구성 파일을 사용하여 각 머신의 정확한 상태를 설정합니다. Machine Config Operator는 설치 후 새 인증서 또는 키 적용과 같은 머신에 대한 추가 변경을 완료합니다.

클러스터 시스템의 장기적인 관리를 위해 DHCP 서버를 사용하는 것이 좋습니다. DHCP 서버가 클러스터 시스템에 영구 IP 주소, DNS 서버 정보 및 호스트 이름을 제공하도록 구성되었는지 확인합니다.



참고

사용자 프로비저닝 인프라에 DHCP 서비스를 사용할 수 없는 경우 RHCOS 설치 시 노드에 IP 네트워킹 구성과 DNS 서버의 주소를 대신 제공할 수 있습니다. ISO 이미지에서 설치하는 경우 부팅 인수로 전달할 수 있습니다. 고정 IP 프로비저닝 *및 고급 네트워킹 옵션에 대한 자세한 내용은 **RHCOS 설치 및 OpenShift Container Platform 부트스트랩 프로세스 시작* 섹션을 참조하십시오.

Kubernetes API 서버가 클러스터 시스템의 노드 이름을 확인할 수 있어야 합니다. API 서버와 작업자 노드가 서로 다른 영역에 있는 경우, API 서버가 노드 이름을 확인할 수 있도록 기본 DNS 검색 영역을 설정할 수 있습니다. 노드 개체와 모든 DNS 요청에서 항상 정규화된 도메인 이름으로 호스트를 가리키는 것도 지원되는 방법입니다

14.3.4.6.1. DHCP를 통해 클러스터 노드의 호스트 이름 설정

RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 호스트 이름은 NetworkManager를 통해 설정됩니다. 기본적으로 시스템은 DHCP를 통해 호스트 이름을 가져옵니다. DHCP에서 호스트 이름을 제공하지 않거나, 커널 인수를 통해 정적으로 설정하거나 다른 방법을 통해 설정하면 역방향 DNS 조회를 통해 얻을 수 있습니다. 역방향 DNS 조회는 노드에서 네트워크를 초기화한 후 수행되며 확인하는 데 시간이 걸릴 수 있습니다. 다른 시스템 서비스는 이 보다 먼저 시작하여 호스트 이름을 localhost 등으로 감지할 수 있습니다. DHCP를 사용하여 각 클러스터 노드의 호스트 이름을 제공하여 이 문제를 방지할 수 있습니다.

또한 DHCP를 통해 호스트 이름을 설정하면 DNS 분할 수평 구현 환경에서 수동으로 DNS 레코드 이름 구성 오류를 무시할 수 있습니다.

14.3.4.6.2. 네트워크 연결 요구사항

OpenShift Container Platform 클러스터 구성 요소가 통신할 수 있도록 시스템 간 네트워크 연결을 구성해야 합니다. 각 시스템에서 클러스터에 있는 다른 모든 시스템의 호스트 이름을 확인할 수 있어야 합니다.

이 섹션에서는 필요한 포트에 대해 자세히 설명합니다.

표 14.20. 모든 시스템 간 통신에 사용되는 포트

프로토콜	포트	설명
ICMP	해당 없음	네트워크 연결성 테스트
TCP	**1936**	메트릭
	9000-9999	**9100-9101** 포트의 노드 내보내기 및 **9099** 포트의 Cluster Version Operator를 포함한 호스트 수준 서비스.
	10250-10259	Kubernetes에서 예약하는 기본 포트
	10256	openshift-sdn
UDP	**4789**	VXLAN 및 Geneve
	6081	VXLAN 및 Geneve
	9000-9999	**9100-9101** 포트의 노드 내보내기를 포함한 호스트 수준 서비스.
	500	IPsec IKE 패킷
	4500	IPsec NAT-T 패킷
TCP/UDP	**30000-32767**	Kubernetes 노드 포트
ESP	해당 없음	IPsec Encapsulating Security Payload (ESP)

표 14.21. 모든 시스템과 컨트롤 플레인 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**6443**	Kubernetes API

표 14.22. 컨트롤 플레인 머신 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**2379-2380**	etcd 서버 및 피어 포트

사용자 프로비저닝 인프라에 대한 NTP 구성

OpenShift Container Platform 클러스터는 기본적으로 공용 **NTP(Network Time Protocol)** 서버를 사용하도록 구성되어 있습니다. 로컬 엔터프라이즈 **NTP** 서버를 사용하거나 클러스터가 연결이 끊긴 네트워크에 배포되는 경우 특정 시간 서버를 사용하도록 클러스터를 구성할 수 있습니다. 자세한 내용은 *chrony 타임 서비스 설정* 문서를 참조하십시오.

DHCP 서버가 **NTP** 서버 정보를 제공하는 경우 **RHCOS(Red Hat Enterprise Linux CoreOS)** 시스템의 **chrony** 타임 서비스에서 정보를 읽고 **NTP** 서버와 클럭을 동기화할 수 있습니다.

추가 리소스

- chrony 타임 서비스 설정

14.3.4.7. 사용자 프로비저닝 DNS 요구사항

OpenShift Container Platform 배포의 경우 다음 구성 요소에 **DNS** 이름을 확인해야 합니다.

- **Kubernetes API**
- **OpenShift Container Platform** 애플리케이션 와일드카드
- 부트스트랩, 컨트롤 플레인 및 컴퓨팅 시스템

Kubernetes API, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템에 대한 역방향 **DNS** 확인이 필요합니다.

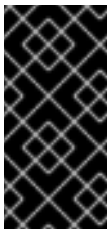
DNS A/AAAA 또는 **CNAME** 레코드는 이름 확인에 사용되며 **PTR** 레코드는 역방향 이름 확인에 사용됩니다. **RHCOS (Red Hat Enterprise Linux CoreOS)**는 **DHCP**에서 호스트 이름을 제공하지 않는 한 모든 노드의 호스트 이름을 설정할 때 역방향 레코드를 사용하기 때문에 역방향 레코드가 중요합니다. 또한 역방향 레코드는 **OpenShift Container Platform**이 작동하는 데 필요한 인증서 서명 요청 **(CSR)**을 생성하는 데 사용됩니다.

참고

DHCP 서버를 사용하여 각 클러스터 노드에 호스트 이름을 제공하는 것이 좋습니다. 자세한 내용은 *사용자 프로비저닝 인프라 섹션에 대한 DHCP 권장 사항* 섹션을 참조하십시오.

사용자가 프로비저닝한 OpenShift Container Platform 클러스터에 대해 다음 DNS 레코드가 필요하며 설치 전에 있어야 합니다. 각 레코드에서 <cluster_name>은 클러스터 이름이고 <base_domain>은 install-config.yaml 파일에서 지정하는 기반 도메인입니다. 전체 DNS 레코드는 <component>.<cluster_name>.<base_domain> 형식입니다.

표 14.23. 필수 DNS 레코드

구성 요소	레코드	설명
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API 로드 밸런서를 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.
	api-int.<cluster_name>.<base_domain>.	내부적으로 API 로드 밸런서를 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.  **중요** API 서버는 Kubernetes에 기록된 호스트 이름으로 작업자 노드를 확인할 수 있어야 합니다. API 서버가 노드 이름을 확인할 수 없는 경우 프록시된 API 호출이 실패할 수 있으며 pod에서 로그를 검색할 수 없습니다.
라우트	***.apps.<cluster_name>.<base_domain>.**	애플리케이션 인그레스 로드 밸런서를 참조하는 와일드카드 DNS A/AAA 또는 CNAME 레코드입니다. 애플리케이션 인그레스 로드 밸런서는 Ingress 컨트롤러 Pod를 실행하는 머신을 대상으로 합니다. Ingress 컨트롤러 Pod는 기본적으로 컴퓨팅 머신에서 실행됩니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다. 예를 들어 **console-openshift-console.apps.<cluster_name>.<base_domain>**은 OpenShift Container Platform 콘솔의 와일드카드 경로로 사용됩니다.
부트스트랩 시스템	**bootstrap.<cluster_name>.<base_domain>.**	부트스트랩 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.

구성 요소	레코드	설명
컨트롤 플레인 머신	**<master><n>. <cluster_name>. <base_domain>.**	컨트롤 플레인 노드의 각 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.
컴퓨팅 머신	**<worker><n>. <cluster_name>. <base_domain>.**	작업자 노드의 각 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.

참고

OpenShift Container Platform 4.4 이상에서는 DNS 구성에서 etcd 호스트 및 SRV 레코드를 지정할 필요가 없습니다.

작은 정보

dig 명령을 사용하여 이름과 역방향 이름을 확인할 수 있습니다. 자세한 검증 단계는 *사용자 프로비저닝 인프라의 DNS 확인* 섹션을 참조하십시오.

14.3.4.7.1. 사용자 프로비저닝 클러스터의 DNS 구성 예

이 섹션에서는 사용자 프로비저닝 인프라에 OpenShift Container Platform을 배포하기 위한 DNS 요구 사항을 충족하는 A 및 PTR 레코드 구성 샘플을 제공합니다. 샘플은 하나의 DNS 솔루션을 선택하기 위한 조언을 제공하기 위한 것이 아닙니다.

이 예제에서 클러스터 이름은 ocp4이고 기본 도메인은 example.com입니다.

사용자 프로비저닝 클러스터의 DNS A 레코드 구성 예

다음 BIND 영역 파일의 예제에서는 사용자가 프로비저닝한 클러스터의 이름 확인을 위한 샘플 A 레코드를 보여줍니다.

예 14.4. 샘플 DNS 영역 데이터베이스

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
```

```
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
master0.ocp4.example.com. IN A 192.168.1.97 5
master1.ocp4.example.com. IN A 192.168.1.98 6
master2.ocp4.example.com. IN A 192.168.1.99 7
;
worker0.ocp4.example.com. IN A 192.168.1.11 8
worker1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1

Kubernetes API의 이름 확인을 제공합니다. 레코드는 **API** 로드 밸런서의 **IP** 주소를 나타냅니다.

2

Kubernetes API의 이름 확인을 제공합니다. 레코드는 **API** 로드 밸런서의 **IP** 주소를 참조하며 내부 클러스터 통신에 사용됩니다.

3

와일드카드 경로의 이름 확인을 제공합니다. 레코드는 애플리케이션 인그레스 로드 밸런서의 **IP** 주소를 나타냅니다. 애플리케이션 인그레스 로드 밸런서는 **Ingress** 컨트롤러 **Pod**를 실행하는 머신을 대상으로 합니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

참고

이 예제에서는 **Kubernetes API** 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 **API** 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

4

부트스트랩 시스템의 이름 확인을 제공합니다.

5 6 7

컨트롤 플레인 시스템의 이름 확인을 제공합니다.

8 9

컴퓨팅 시스템의 이름 확인을 제공합니다.

사용자 프로비저닝 클러스터의 **DNS PTR** 레코드 구성 예

다음 예제 **BIND** 영역 파일은 사용자 프로비저닝 클러스터의 역방향 이름 확인을 위한 샘플 **PTR** 레코드를 보여줍니다.

예 14.5. 역방향 레코드의 샘플 DNS 영역 데이터베이스

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR master0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR master1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR master2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 7
```

```
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com. 8
;
;EOF
```

1

Kubernetes API의 역방향 DNS 확인을 제공합니다. PTR 레코드는 API 로드 밸런서의 레코드 이름을 참조합니다.

2

Kubernetes API의 역방향 DNS 확인을 제공합니다. PTR 레코드는 API 로드 밸런서의 레코드 이름을 참조하며 내부 클러스터 통신에 사용됩니다.

3

부트스트랩 시스템의 역방향 DNS 확인을 제공합니다.

4 5 6

컨트롤 플레인 시스템의 역방향 DNS 확인을 제공합니다.

7 8

컴퓨팅 시스템의 역방향 DNS 확인을 제공합니다.

참고

OpenShift Container Platform 애플리케이션 와일드카드에는 PTR 레코드가 필요하지 않습니다.

14.3.4.8. 사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구사항

OpenShift Container Platform을 설치하기 전에 API 및 애플리케이션 인그레스 로드 밸런싱 인프라를 프로비저닝해야 합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 API 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

참고

RHEL(Red Hat Enterprise Linux) 인스턴스를 사용하여 API 및 애플리케이션 인그레스 로드 밸런서를 배포하려면 RHEL 서브스크립션을 별도로 구매해야 합니다.

로드 밸런서 인프라는 다음 요구 사항을 충족해야 합니다.

1. **API 로드 밸런서: 플랫폼과 상호 작용하고 플랫폼을 구성할 수 있도록 사용자(인간과 시스템 모두)에게 공통 끝점을 제공합니다. 다음 조건을 설정합니다.**

   - **Layer 4 로드 밸런싱 전용입니다. 이를 Raw TCP, SSL Passthrough 또는 SSL Bridge 모드라고 합니다. SSL Bridge 모드를 사용하는 경우, API 경로에 대해 SNI(Server Name Indication, 서버 이름 표시)를 활성화해야 합니다.**

   - **스테이트리스 로드 밸런싱 알고리즘입니다. 옵션은 로드 밸런서 구현에 따라 달라집니다.**

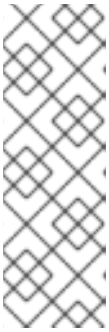
참고

API 로드 밸런서가 제대로 작동하기 위해 세션 지속성이 필요하지 않습니다.

로드 밸런서의 전면과 후면 모두에서 다음 포트를 구성하십시오.

표 14.24. API 로드 밸런서

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
6443	부트스트랩 및 컨트롤 플레인. 부트스트랩 시스템이 클러스터 컨트롤 플레인을 초기화한 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다. API 서버 상태 검사 프로브에 대한 **/readyz** 끝점을 구성해야 합니다.	X	X	Kubernetes API 서버
22623	부트스트랩 및 컨트롤 플레인. 부트스트랩 시스템이 클러스터 컨트롤 플레인을 초기화한 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다.	X		시스템 구성 서버



참고

API 서버가 /readyz 엔드포인트를 해제하는 시점부터 풀에서 API 서버 인스턴스가 제거되는 시점까지 시간이 30초를 넘지 않도록 로드 밸런서를 구성해야 합니다. /readyz가 오류를 반환하거나 정상 상태가 된 후 정해진 시간 안에 끝점이 제거 또는 추가되어야 합니다. 5초 또는 10초의 프로빙 주기(두 번의 성공적인 요청은 정상 상태, 세 번의 요청은 비정상 상태)는 충분한 테스트를 거친 값입니다.

2. 애플리케이션 인그레스 로드 밸런서: 클러스터 외부에서 유입되는 애플리케이션 트래픽에 대한 인그래스 포인트를 제공합니다. 다음 조건을 설정합니다.

- Layer 4 로드 밸런싱 전용입니다. 이를 Raw TCP, SSL Passthrough 또는 SSL Bridge 모드라고 합니다. SSL Bridge 모드를 사용하는 경우 인그레스 경로에 대해 SNI(Server Name Indication, 서버 이름 표시)를 활성화해야 합니다.

- 사용 가능한 옵션과 플랫폼에서 호스팅되는 애플리케이션 유형에 따라 연결 기반 또는 세션 기반 지속성이 권장됩니다.

작은 정보

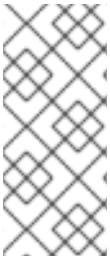
애플리케이션 인그레스 로드 밸런서에서 클라이언트의 실제 IP 주소를 확인할 수 있는 경우 소스 IP 기반 세션 지속성을 활성화하면 엔드 투 엔드 TLS 암호화를 사용하는 애플리케이션의 성능을 향상시킬 수 있습니다.

로드 밸런서의 전면과 후면 모두에서 다음 포트를 구성하십시오.

표 14.25. 애플리케이션 인그레스 로드 밸런서

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
443	기본적으로 인그레스 컨트롤러 pod, 컴퓨팅 또는 작업자를 실행하는 시스템입니다.	X	X	HTTPS 트래픽
80	기본적으로 인그레스 컨트롤러 pod, 컴퓨팅 또는 작업자를 실행하는 시스템입니다.	X	X	HTTP 트래픽

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
1936	기본적으로 Ingress 컨트롤러 Pod를 실행하는 작업자 노드입니다. 수신 상태 점검 프로브에 대해 **/healthz/ready** 끝점을 구성해야 합니다.	X	X	HTTP 트래픽



참고

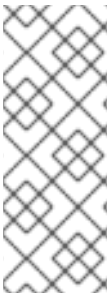
컴퓨팅 노드가 0인 3-노드 클러스터를 배포하는 경우 Ingress 컨트롤러 Pod는 컨트롤 플레인 노드에서 실행됩니다. 3-노드 클러스터 배포에서 HTTP 및 HTTPS 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다.

참고

인그레스 라우터에 대한 작업 구성이 OpenShift Container Platform 클러스터에 필요합니다. 컨트롤 플레인 초기화 후 인그레스 라우터를 설정해야 합니다.

14.3.4.8.1. 사용자 프로비저닝 클러스터의 로드 밸런서 구성 예

이 섹션에서는 사용자 프로비저닝 클러스터의 로드 밸런싱 요구 사항을 충족하는 API 및 애플리케이션 수신 로드 밸런서 구성 예를 제공합니다. 샘플은 HAProxy 로드 밸런서에 대한 /etc/haproxy/haproxy.cfg 구성입니다. 이 예제에서는 하나의 로드 밸런싱 솔루션을 선택하기 위한 조언을 제공하는 것을 목적으로 하지 않습니다.



참고

이 예제에서는 Kubernetes API 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 API 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

예 14.6. API 및 애플리케이션 인그레스 로드 밸런서 구성 샘플

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
```

```
    option                  redispatch
    retries                 3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s
    maxconn                 3000
  frontend stats
    bind *:1936
    mode            http
    log             global
    maxconn 10
    stats enable
    stats hide-version
    stats refresh 30s
    stats show-node
    stats show-desc Stats for ocp4 cluster (1)
    stats auth admin:ocp4
    stats uri /stats
  listen api-server-6443 (2)
    bind *:6443
    mode tcp
    server bootstrap bootstrap.ocp4.example.com:6443 check inter 1s backup (3)
    server master0 master0.ocp4.example.com:6443 check inter 1s
    server master1 master1.ocp4.example.com:6443 check inter 1s
    server master2 master2.ocp4.example.com:6443 check inter 1s
  listen machine-config-server-22623 (4)
    bind *:22623
    mode tcp
    server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup (5)
    server master0 master0.ocp4.example.com:22623 check inter 1s
    server master1 master1.ocp4.example.com:22623 check inter 1s
    server master2 master2.ocp4.example.com:22623 check inter 1s
  listen ingress-router-443 (6)
    bind *:443
    mode tcp
    balance source
    server worker0 worker0.ocp4.example.com:443 check inter 1s
    server worker1 worker1.ocp4.example.com:443 check inter 1s
  listen ingress-router-80 (7)
    bind *:80
    mode tcp
    balance source
    server worker0 worker0.ocp4.example.com:80 check inter 1s
    server worker1 worker1.ocp4.example.com:80 check inter 1s
```

(1)

이 예에서 클러스터 이름은 **ocp4** 입니다.

(2)

3 5

부트스트랩 항목은 **OpenShift Container Platform** 클러스터 설치 전에 있어야 하며 부트스트랩 프로세스가 완료된 후 제거해야 합니다.

4

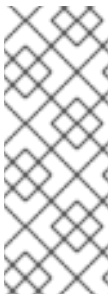
포트 **22623**은 머신 구성 서버 트래픽을 처리하고 컨트롤 플레인 시스템을 가리킵니다.

6

포트 **443**은 **HTTPS** 트래픽을 처리하고 **Ingress** 컨트롤러 **Pod**를 실행하는 시스템을 가리킵니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

7

포트 **80**은 **HTTP** 트래픽을 처리하고 **Ingress** 컨트롤러 **Pod**를 실행하는 머신을 가리킵니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

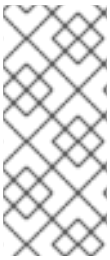


참고

컴퓨팅 노드가 **0**인 **3-**노드 클러스터를 배포하는 경우 **Ingress** 컨트롤러 **Pod**는 컨트롤 플레인 노드에서 실행됩니다. **3-**노드 클러스터 배포에서 **HTTP** 및 **HTTPS** 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다.

작은 정보

HAProxy를 로드 밸런서로 사용하는 경우 **HAProxy** 노드에서 **netstat -nltupe**를 실행하여 **haproxy** 프로세스가 포트 **6443, 22623, 443** 및 **80**에서 수신 대기 중인지 확인할 수 있습니다.



참고

HAProxy를 로드 밸런서로 사용하고 **SELinux**가 **enforcing**으로 설정된 경우 **HAProxy** 서비스가 **setsebool -P haproxy_connect_any=1**을 실행하여 구성된 **TCP** 포트에 바인딩할 수 있는지 확인해야 합니다.

14.3.5. 사용자 프로비저닝 인프라 준비

사용자 프로비저닝 인프라에 **OpenShift Container Platform**을 설치하기 전에 기본 인프라를 준비해야 합니다.

이 섹션에서는 **OpenShift Container Platform** 설치를 준비하기 위해 클러스터 인프라를 설정하는 데 필요한 높은 수준의 단계에 대해 자세히 설명합니다. 여기에는 클러스터 노드에 대한 **IP** 네트워킹 및 네트워크 연결 구성, 방화벽을 통해 필요한 포트 활성화, 필수 **DNS** 및 로드 밸런싱 인프라를 설정하는 작업이 포함됩니다.

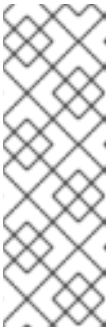
준비 후 클러스터 인프라는 *사용자 프로비저닝 인프라가 있는 클러스터의 요구 사항* 섹션에 설명된 요구 사항을 충족해야 합니다.

사전 요구 사항

- OpenShift Container Platform 4.x 테스트된 통합 페이지를 검토했습니다.
- *사용자 프로비저닝 인프라가 있는 클러스터의 요구 사항* 섹션에 자세히 설명된 인프라 요구 사항을 검토했습니다.

프로세스

1. **DHCP**를 사용하여 클러스터 노드에 **IP** 네트워킹 구성을 제공하는 경우 **DHCP** 서비스를 구성합니다.
   a. 노드의 영구 **IP** 주소를 **DHCP** 서버 구성에 추가합니다. 구성에서 관련 네트워크 인터페이스의 **MAC** 주소를 각 노드의 의도한 **IP** 주소와 일치시킵니다.
   b. **DHCP**를 사용하여 클러스터 시스템의 **IP** 주소 지정을 구성하는 경우 시스템은 **DHCP**를 통해 **DNS** 서버 정보도 가져옵니다. **DHCP** 서버 구성을 통해 클러스터 노드에서 사용하는 영구 **DNS** 서버 주소를 정의합니다.



참고

DHCP 서비스를 사용하지 않는 경우 **RHCOS** 설치 시 **IP** 네트워킹 구성과 **DNS** 서버의 주소를 노드에 제공해야 합니다. **ISO** 이미지에서 설치하는 경우 부팅 인수로 전달할 수 있습니다. 고정 **IP** 프로비저닝 *및 고급 네트워킹 옵션에 대한 자세한 내용은 **RHCOS** 설치 및 **OpenShift Container Platform** 부트스트랩 프로세스 시작* 섹션을 참조하십시오.

c.

DHCP 서버 구성에 클러스터 노드의 호스트 이름을 정의합니다. 호스트 이름 고려 사항에 대한 자세한 내용은 ***DHCP****를 통해 클러스터 노드 호스트 이름 설정* 섹션을 참조하십시오.

참고

DHCP 서비스를 사용하지 않는 경우 클러스터 노드는 역방향 **DNS** 조회를 통해 호스트 이름을 가져옵니다.

2.

네트워크 인프라가 클러스터 구성 요소 간 필수 네트워크 연결을 제공하는지 확인합니다. 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 인프라 섹션의 네트워킹 요구 사항* 섹션을 참조하십시오.

3.

OpenShift Container Platform 클러스터 구성 요소가 통신하는 데 필요한 포트를 활성화하도록 방화벽을 구성합니다. 필요한 포트에 대한 자세한 내용은 *사용자 프로비저닝 인프라 섹션의 네트워킹 요구 사항* 섹션을 참조하십시오.

4.

클러스터에 필요한 **DNS** 인프라를 설정합니다.

a.

Kubernetes API, 애플리케이션 와일드카드, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템의 **DNS** 이름 확인을 구성합니다.

b.

Kubernetes API, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템에 대한 역방향 **DNS** 확인을 구성합니다.

OpenShift Container Platform DNS 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 **DNS** 요구 사항* 섹션을 참조하십시오.

5.

DNS 구성을 확인합니다.

a.

설치 노드에서 **Kubernetes API**의 레코드 이름, 와일드카드 경로 및 클러스터 노드에 대해 **DNS** 조회를 실행합니다. 응답의 **IP** 주소가 올바른 구성 요소에 해당하는지 확인합니다.

b.

설치 노드에서 로드 밸런서 및 클러스터 노드의 **IP** 주소에 대해 역방향 **DNS** 조회를 실행합니다. 응답의 레코드 이름이 올바른 구성 요소에 해당하는지 확인합니다.

자세한 DNS 검증 단계는 *사용자 프로비저닝 인프라에 대한 DNS 확인* 섹션을 참조하십시오.

6. 필요한 API 및 애플리케이션 수신 로드 밸런싱 인프라를 프로비저닝합니다. 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구 사항* 섹션을 참조하십시오.

참고

일부 로드 밸런싱 솔루션에는 로드 밸런싱을 초기화하기 전에 클러스터 노드의 DNS 이름을 확인해야 합니다.

14.3.6. 사용자 프로비저닝 인프라에 대한 DNS 확인 검증

사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치하기 전에 DNS 구성을 확인할 수 있습니다.

중요

클러스터를 설치하기 전에 이 섹션에 설명된 검증 단계를 성공해야 합니다.

사전 요구 사항

- 사용자 프로비저닝 인프라에 필요한 DNS 레코드를 구성했습니다.

프로세스

1. 설치 노드에서 Kubernetes API의 레코드 이름, 와일드카드 경로 및 클러스터 노드에 대해 DNS 조회를 실행합니다. 응답에 포함된 IP 주소가 올바른 구성 요소에 해당하는지 확인합니다.

   a. Kubernetes API 레코드 이름을 조회합니다. 결과가 API 로드 밸런서의 IP 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
```

1 <nameserver_ip>를 네임서버의 IP 주소로, <cluster_name>을 클러스터 이름으로, <base_domain>을 기본 도메인 이름으로 바꿉니다.

출력 예

```
api.ocp4.example.com.  0 IN A 192.168.1.5
```

b.

Kubernetes 내부 API 레코드 이름을 조회합니다. 결과가 API 로드 밸런서의 IP 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
```

출력 예

```
api-int.ocp4.example.com.  0 IN A 192.168.1.5
```

c.

예제 *.apps.<cluster_name>.<base_domain>을 테스트합니다. DNS 와일드카드를 조회합니다. 모든 애플리케이션 와일드카드 조회는 애플리케이션 인그레스 로드 밸런서의 IP 주소로 확인되어야 합니다.

```
$ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.
<base_domain>
```

출력 예

```
random.apps.ocp4.example.com.  0 IN A 192.168.1.5
```

참고

예제 출력에서는 **Kubernetes API** 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 **API** 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

random 항목을 다른 와일드카드 값으로 교체할 수 있습니다. 예를 들어 **OpenShift Container Platform** 콘솔의 경로를 쿼리할 수 있습니다.

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

출력 예

```
console-openshift-console.apps.ocp4.example.com. 0 IN A 192.168.1.5
```

d. 부트스트랩 **DNS** 레코드 이름에 대해 조회를 실행합니다. 결과가 부트스트랩 노드의 **IP** 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.
<base_domain>
```

출력 예

```
bootstrap.ocp4.example.com.  0 IN A 192.168.1.96
```

e. 이 방법을 사용하여 컨트롤 플레인 및 컴퓨팅 노드의 **DNS** 레코드 이름에 대해 조회를 수행합니다. 결과가 각 노드의 **IP** 주소에 해당하는지 확인합니다.

2. 설치 노드에서 로드 밸런서 및 클러스터 노드의 **IP** 주소에 대해 역방향 **DNS** 조회를 실행합니다. 응답에 포함된 레코드 이름이 올바른 구성 요소에 해당하는지 확인합니다.

a.

API 로드 밸런서의 IP 주소에 대해 역방향 조회를 수행합니다. 응답에 Kubernetes API 및 Kubernetes 내부 API의 레코드 이름이 포함되어 있는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

출력 예

```
5.1.168.192.in-addr.arpa. 0 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 0 IN PTR api.ocp4.example.com. 2
```

1

Kubernetes 내부 API의 레코드 이름을 제공합니다.

2

Kubernetes API의 레코드 이름을 제공합니다.

참고

OpenShift Container Platform 애플리케이션 와일드카드에는 PTR 레코드가 필요하지 않습니다. 애플리케이션 인그레스 로드 밸런서의 IP 주소에 대한 역방향 DNS 확인에는 유효성 검사 단계가 필요하지 않습니다.

b.

부트스트랩 노드의 IP 주소에 대해 역방향 조회를 수행합니다. 결과가 부트스트랩 노드의 DNS 레코드 이름을 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

출력 예

```
96.1.168.192.in-addr.arpa. 0 IN PTR bootstrap.ocp4.example.com.
```

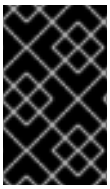
c.
이 방법을 사용하여 컨트롤 플레인 및 컴퓨팅 노드의 IP 주소에 대해 역방향 조회를 수행합니다. 결과가 각 노드의 DNS 레코드 이름과 일치하는지 확인합니다.

14.3.7. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 core 사용자의 ~/.ssh/authorized_keys 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 core로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. ./openshift-install gather 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.



중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

참고

AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

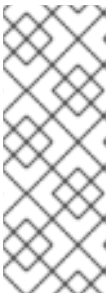
프로세스

1.
로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1

새 SSH 키의 경로 및 파일 이름(예: ~/.ssh/id_ed25519)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 '~/.ssh 디렉터리에 있는지 확인하십시오.



참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 ed25519 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 rsa 또는 ecdsa 알고리즘을 사용하는 키를 생성합니다.

2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 ~/.ssh/id_ed25519.pub 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 ./openshift-install gather 명령을 사용하려는 경우 필요합니다.

참고

일부 배포에서는 ~/.ssh/id_rsa 및 ~/.ssh/id_dsa와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. ssh-agent 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 **FIPS** 모드인 경우 **FIPS** 호환 알고리즘만 사용하여 **SSH** 키를 생성합니다. 키는 **RSA** 또는 **ECDSA**여야 합니다.

4. **ssh-agent**에 **SSH** 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 **SSH** 개인 키의 경로 및 파일 이름을 지정합니다(예: ~/.ssh/id_ed25519).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- **OpenShift Container Platform**을 설치할 때 **SSH** 공개 키를 설치 프로그램에 지정합니다.

14.3.8. 수동으로 설치 구성 파일 만들기

OpenShift Container Platform을 사용자가 프로비저닝한 설치의 경우 설치 구성 파일을 수동으로 생성합니다.

사전 요구 사항

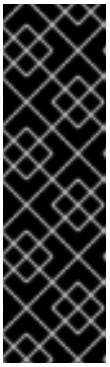
- 로컬 시스템에 설치 프로그램에 제공할 **SSH** 공개 키가 있습니다. 키는 디버깅 및 재해 복구를 위해 클러스터 노드에 대한 **SSH** 인증에 사용됩니다.

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿이 있습니다.

프로세스

1. 필요한 설치 자산을 저장할 설치 디렉터리를 만듭니다.

```
$ mkdir <installation_directory>
```



중요

디렉터리를 만들어야 합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

2. 샘플 install-config.yaml 파일 템플릿을 사용자 지정하여 <installation_directory>에 저장합니다.

참고

이 설정 파일의 이름을 install-config.yaml로 지정해야 합니다.

참고

일부 플랫폼 유형의 경우 대체로 ./openshift-install create install-config --dir <installation_directory> 를 실행하여 install-config.yaml 파일을 생성할 수 있습니다. 프롬프트에서 클러스터 구성에 대한 세부 정보를 제공할 수 있습니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 install-config.yaml 파일을 백업합니다.

중요

install-config.yaml 파일은 설치 과정의 다음 단계에서 사용됩니다. 이 시점에서 이를 백업해야 합니다.

14.3.8.1. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 환경에 대한 세부 정보를 설명하는 사용자 지정 **install-config.yaml** 설치 구성 파일을 제공합니다.

참고

설치한 후에는 install-config.yaml 파일에서 이러한 매개변수를 수정할 수 없습니다.

중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

14.3.8.1.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 14.26. 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체

매개변수	설명	값
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	소문자, 하이픈(**-**), 마침표(**.**)로 구성되는 문자열(예: **dev**)입니다.
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	{
		"auths":{
		"cloud.openshift.com":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		},
		"quay.io":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		}
		}
		}

14.3.8.1.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 14.27. 네트워크 매개변수

매개변수	설명	값

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체 **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.
networking.clusterNetwork	Pod의 IP 주소 블록입니다. 기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된**cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16

매개변수	설명	값
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다. 여러 IP 커널 인수를 지정하는 경우 **machineNetwork.cidr** 값은 기본 네트워크의 CIDR이어야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16** **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

14.3.8.1.3. 선택적 구성 매개변수

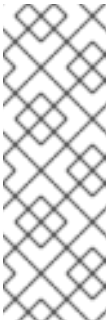
선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 14.28. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **ppc64le** (기본값)입니다.	문자열

매개변수	설명	값
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은 **3**입니다.
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **ppc64le** (기본값)입니다.	문자열

매개변수	설명	값
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 기본값은 **External**입니다. 이 필드를 **Internal** 로 설정하는 것은 클라우드 이외의 플랫폼에서는 지원되지 않습니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. 참고 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

14.3.8.2. IBM Power의 샘플 install-config.yaml 파일

install-config.yaml 파일을 사용자 지정하여 **OpenShift Container Platform** 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.

```
apiVersion: v1
baseDomain: example.com 1
compute: 2
- hyperthreading: Enabled 3
  name: worker
  replicas: 0 4
  architecture : ppc64le
controlPlane: 5
  hyperthreading: Enabled 6
  name: master
  replicas: 3 7
  architecture : ppc64le
metadata:
  name: test 8
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 9
    hostPrefix: 23 10
  networkType: OpenShiftSDN
  serviceNetwork: 11
  - 172.30.0.0/16
platform:
  none: {} 12
fips: false 13
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email":
"you@example.com"}}}' 14
sshKey: 'ssh-ed25519 AAAA...' 15
additionalTrustBundle: | 16
```

```
  -----BEGIN CERTIFICATE-----
  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
imageContentSources: 17
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

1

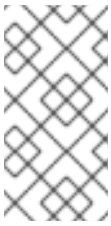
클러스터의 기본 도메인입니다. 모든 **DNS** 레코드는 이 기본 도메인의 하위 도메인이어야 하며 클러스터 이름을 포함해야 합니다.

2 5

controlPlane 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

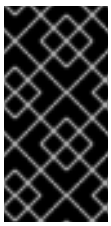
3 6

동시 멀티스레딩(**SMT**) 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 **SMT**가 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. **SMT**를 비활성화하는 경우 모든 클러스터 머신에서 이를 비활성화해야 합니다. 여기에는 컨트롤 플레인과 컴퓨팅 머신이 모두 포함됩니다.



참고

SMT(동시 멀티 스레딩)는 기본적으로 활성화되어 있습니다. **BIOS** 설정에서 **SMT**를 활성화하지 않으면 **hyperthreading** 매개변수가 적용되지 않습니다.

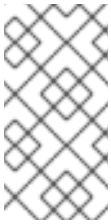


중요

BIOS에서든 **install-config.yaml** 파일에서든 **hyperthreading**을 비활성화한 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.

4

사용자 프로비저닝 인프라에 **OpenShift Container Platform**을 설치할 때 이 값을 **0**으로 설정해야 합니다. 설치 프로그램에서 제공하는 설치에서 매개 변수는 클러스터가 생성 및 관리하는 컴퓨팅 머신 수를 제어합니다. 사용자 프로비저닝 설치에서는 클러스터 설치를 완료하기 전에 컴퓨팅 시스템을 수동으로 배포해야 합니다.



참고

3-노드 클러스터를 설치하는 경우 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템을 설치할 때 컴퓨팅 머신을 배포하지 마십시오.

7

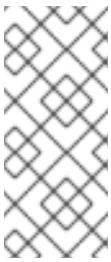
클러스터에 추가하는 컨트롤 플레인 시스템의 수입니다. 클러스터에서 이 값을 클러스터의 etcd 끝점 수로 사용하므로 이 값은 배포하는 컨트롤 플레인 시스템의 수와 일치해야 합니다.

8

DNS 레코드에 지정한 클러스터 이름입니다.

9

Pod IP 주소가 할당되는 IP 주소 블록입니다. 이 블록은 기존 물리적 네트워크와 중복되지 않아야합니다. 이러한 IP 주소는 Pod 네트워크에 사용됩니다. 외부 네트워크에서 Pod에 액세스해야 하는 경우, 트래픽을 관리하도록 로드 밸런서와 라우터를 설정해야 합니다.



참고

클래스 E CIDR 범위는 향후 사용을 위해 예약되어 있습니다. 클래스 E CIDR 범위를 사용하려면 네트워킹 환경에서 클래스 E CIDR 범위 내에서 IP 주소를 수락하는지 확인해야 합니다.

10

개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 hostPrefix를 23으로 설정하면 지정된 cidr 이외 /23 서브넷이 각 노드에 할당되어 510(2^(32 - 23) - 2) Pod IP 주소가 허용됩니다. 외부 네트워크에서 노드에 액세스해야 하는 경우 트래픽을 관리하도록 로드 밸런서와 라우터를 구성합니다.

11

서비스 IP 주소에 사용할 IP 주소 풀입니다. IP 주소 풀은 하나만 입력할 수 있습니다. 이 블록은 기존 물리적 네트워크와 중복되지 않아야합니다. 외부 네트워크에서 서비스에 액세스해야 하는 경우, 트래픽을 관리하도록 로드 밸런서와 라우터를 구성합니다.

12

플랫폼을 none으로 설정해야 합니다. IBM Power 인프라에 대한 추가 플랫폼 구성 변수는 지정할 수 없습니다.

중요

플랫폼 유형으로 설치된 클러스터는 **Machine API**로 컴퓨팅 머신 관리와 같은 일부 기능을 사용할 수 없습니다. 이 제한은 클러스터에 연결된 컴퓨팅 시스템이 일반적으로 기능을 지원하는 플랫폼에 설치된 경우에도 적용됩니다. 이 매개변수는 설치 후 변경할 수 없습니다.

13

FIPS 모드 활성화 또는 비활성화 여부입니다. 기본적으로 **FIPS** 모드는 비활성화됩니다. **FIPS** 모드가 활성화되면 **OpenShift Container Platform**이 실행되는 **RHCOS(Red Hat Enterprise Linux CoreOS)** 시스템에서 기본 **Kubernetes** 암호화 제품군은 우회하고 **RHCOS**와 함께 제공되는 암호화 모듈을 대신 사용합니다.

중요

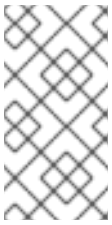
FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 **OpenShift Container Platform** 배포에서만 지원됩니다.

14

<local_registry>는 미러 레지스트리가 해당 내용을 제공하는 데 사용하는 레지스트리 도메인 이름과 포트(선택사항)를 지정합니다. 예: **registry.example.com** 또는 **registry.example.com:5000 <credentials>**는 미러 레지스트리의 **base64** 인코딩 사용자 이름과 암호를 지정합니다.

15

RHCOS(Red Hat Enterprise Linux CoreOS)의 **core** 사용자에 대한 **SSH** 공용 키입니다.



참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 **OpenShift Container Platform** 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 **SSH** 키를 지정합니다.

16

미러 레지스트리에 사용한 인증서 파일의 내용을 제공하십시오.

17

명령 출력에서 **imageContentSources** 섹션을 제공하여 리포지토리를 미러링하십시오.

14.3.8.3. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 **HTTP** 또는 **HTTPS** 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 **OpenShift Container Platform** 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 **API**에 대한 호출을 포함하여 모든 클러스터 발신(**Egress**) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.

참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), **Google Cloud Platform (GCP)**, **Microsoft Azure** 및 **Red Hat OpenStack Platform (RHOSP)**에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1

클러스터 외부에서 **HTTP** 연결을 구축하는 데 사용할 프록시 **URL**입니다. **URL** 스키마는 **http**여야 합니다.

2

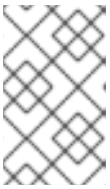
클러스터 외부에서 **HTTPS** 연결을 구축하는 데 사용할 프록시 **URL**입니다.

3

대상 도메인 이름, **IP** 주소 또는 프록시에서 제외할 기타 네트워크 **CIDR**로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. ***를 사용하여 모든 대상에 대해 프록시를 바이패스합니다.

4

이 값을 제공하면 설치 프로그램에서 추가 **CA** 인증서를 보유할 **openshift -config** 네임스페이스에 **user-ca- bundle** 이라는 구성 맵을 생성합니다. **additionalTrustBundle** 및 하나 이상의 프록시 설정을 제공하는 경우 프록시 오브젝트는 **trustedCA** 필드의 **user-ca-bundle** 구성 맵을 참조하도록 구성됩니다. 그러면 **Cluster Network Operator**에서 **trustedCA** 매개변수에 대해 지정된 콘텐츠를 **RHCOS** 신뢰 번들과 병합하는 **trusted-ca-bundle** 구성 맵을 생성합니다. 프록시의 **ID** 인증서를 **RHCOS** 트러스트 번들에 있는 기관에서 서명하지 않은 경우 **additionalTrustBundle** 필드가 있어야 합니다.



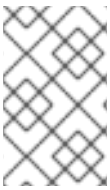
참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2.

파일을 저장해 놓고 **OpenShift Container Platform**을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 **nil**이 됩니다.



참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

14.3.8.4. 3개의 노드 클러스터 구성

필요한 경우 세 개의 컨트롤 플레인 시스템으로 구성된 베어 메탈 클러스터에 제로 컴퓨팅 머신을 배포할 수 있습니다. 이를 통해 클러스터 관리자와 개발자들이 테스트, 개발, 프로덕션에 사용할 수 있는 소형화되고 리소스 효율이 높은 클러스터를 제공합니다.

3-노드 OpenShift Container Platform 환경에서 세 개의 컨트롤 플레인 머신을 예약할 수 있습니다. 즉, 애플리케이션 워크로드가 해당 플랫폼에서 실행되도록 예약됩니다.

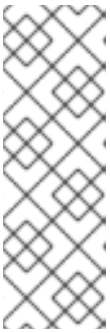
사전 요구 사항

- 기존 install-config.yaml 파일이 있습니다.

절차

- install-config.yaml 파일에서 다음 compute 스탠자에 표시된 대로 컴퓨팅 복제본 수가 0으로 설정되어 있는지 확인합니다.

```
compute:
- name: worker
  platform: {}
  replicas: 0
```



참고

배포 중인 컴퓨팅 머신 수에 관계없이 사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치할 때 컴퓨팅 머신의 replicas 매개변수 값을 0으로 설정해야 합니다. 설치 프로그램에서 제공하는 설치에서 매개 변수는 클러스터가 생성 및 관리하는 컴퓨팅 머신 수를 제어합니다. 이 설정은 컴퓨팅 시스템이 수동으로 배포되는 사용자 프로비저닝 설치에는 적용되지 않습니다.

3-노드 클러스터 설치의 경우 다음 단계를 따르십시오.

- 컴퓨팅 노드가 0인 3-노드 클러스터를 배포하는 경우 Ingress 컨트롤러 Pod는 컨트롤 플레인 노드에서 실행됩니다. 3-노드 클러스터 배포에서 HTTP 및 HTTPS 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다. 자세한 내용은 *사용자 프로비저닝 인프라 섹션에 대한 로드 밸런싱 요구 사항* 섹션을 참조하십시오.

- 다음 절차에서 Kubernetes 매니페스트 파일을 생성할 때 <installation_directory>/manifests/cluster-scheduler-02-config.yml 파일의

mastersSchedulable 매개변수가 true로 설정되어 있는지 확인합니다. 이렇게 하면 애플리케이션 워크로드를 컨트롤 플레인 노드에서 실행할 수 있습니다.

- RHCOS(Red Hat Enterprise Linux CoreOS) 시스템을 생성할 때 컴퓨팅 노드를 배포하지 마십시오.

14.3.9. CNO(Cluster Network Operator) 구성

클러스터 네트워크의 구성은 CNO(Cluster Network Operator) 구성의 일부로 지정되며 cluster라는 이름의 CR(사용자 정의 리소스) 오브젝트에 저장됩니다. CR은 operator.openshift.io API 그룹에서 Network API의 필드를 지정합니다.

CNO 구성은 Network.config.openshift.io API 그룹의 Network API에서 클러스터 설치 중에 다음 필드를 상속하며 이러한 필드는 변경할 수 없습니다.

clusterNetwork

Pod IP 주소가 할당되는 IP 주소 풀입니다.

serviceNetwork

서비스를 위한 IP 주소 풀입니다.

defaultNetwork.type

OpenShift SDN 또는 OVN-Kubernetes와 같은 클러스터 네트워크 공급자입니다.

cluster라는 CNO 오브젝트에서 defaultNetwork 오브젝트의 필드를 설정하여 클러스터의 클러스터 네트워크 공급자 구성을 지정할 수 있습니다.

14.3.9.1. CNO(Cluster Network Operator) 구성 오브젝트

CNO(Cluster Network Operator)의 필드는 다음 표에 설명되어 있습니다.

표 14.29. CNO(Cluster Network Operator) 구성 오브젝트

필드	유형	설명
metadata.name	string	CNO 개체 이름입니다. 이 이름은 항상 cluster입니다.

필드	유형	설명
spec.clusterNetwork	**array**	Pod IP 주소가 할당되는 IP 주소 블록과 클러스터의 각 개별 노드에 할당된 서브넷 접두사 길이를 지정하는 목록입니다. 예를 들면 다음과 같습니다. spec: clusterNetwork: - cidr: 10.128.0.0/19 hostPrefix: 23 - cidr: 10.128.32.0/19 hostPrefix: 23 매니페스트를 생성하기 전에 **install-config.yaml** 파일에서만 이 필드를 사용자 지정할 수 있습니다. 값은 매니페스트 파일에서 읽기 전용입니다.
spec.serviceNetwork	**array**	서비스의 IP 주소 블록입니다. OpenShift SDN 및 OVN-Kubernetes CNI(Container Network Interface) 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다. 예를 들면 다음과 같습니다. spec: serviceNetwork: - 172.30.0.0/14 매니페스트를 생성하기 전에 **install-config.yaml** 파일에서만 이 필드를 사용자 지정할 수 있습니다. 값은 매니페스트 파일에서 읽기 전용입니다.
spec.defaultNetwork	**object**	클러스터 네트워크의 CNI(Container Network Interface) 클러스터 네트워크 공급자를 구성합니다.
spec.kubeProxyConfig	**object**	이 개체의 필드는 kube-proxy 구성을 지정합니다. OVN-Kubernetes 클러스터 네트워크 공급자를 사용하는 경우 kube-proxy 구성이 적용되지 않습니다.

defaultNetwork 오브젝트 구성

defaultNetwork 오브젝트의 값은 다음 표에 정의되어 있습니다.

표 14.30. defaultNetwork 오브젝트

필드	유형	설명

필드	유형	설명
type	**string**	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, 클러스터 네트워크 공급자가 설치 중에 선택됩니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.  **참고** OpenShift Container Platform은 기본적으로 OpenShift SDN CNI(Container Network Interface) 클러스터 네트워크 공급자를 사용합니다.
openshiftSDNConfig	**object**	이 오브젝트는 OpenShift SDN 클러스터 네트워크 공급자에만 유효합니다.
ovnKubernetesConfig	**object**	이 오브젝트는 OVN-Kubernetes 클러스터 네트워크 공급자에만 유효합니다.

OpenShift SDN CNI 네트워크 공급자에 대한 구성

다음 표에서는 OpenShift SDN Container Network Interface (CNI) 클러스터 네트워크 공급자의 구성 필드를 설명합니다.

표 14.31. openshiftSDNConfig 오브젝트

필드	유형	설명
mode	**string**	OpenShift SDN의 네트워크 격리 모드를 구성합니다. 기본값은 **NetworkPolicy**입니다. **Multitenant** 및 **Subnet** 값은 OpenShift Container Platform 3.x 버전과의 역호환할 수 있지만 권장되지는 않습니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
mtu	**integer**	VXLAN 오버레이 네트워크의 최대 전송 단위(MTU)입니다. 이는 기본 네트워크 인터페이스의 MTU를 기준으로 자동 탐지됩니다. 일반적으로 감지된 MTU를 재정의할 필요는 없습니다. 자동 감지 값이 예상 밖인 경우 노드의 기본 네트워크 인터페이스의 MTU가 올바른지 확인합니다. 이 옵션을 사용하여 노드의 기본 네트워크 인터페이스의 MTU 값을 변경할 수 없습니다. 클러스터에 다른 노드에 대한 다른 MTU 값이 필요한 경우 이 값을 클러스터의 가장 낮은 MTU 값보다 **50** 미만으로 설정해야 합니다. 예를 들어 클러스터의 일부 노드에는 **9001**의 MTU가 있고 일부에는 **1500**의 MTU가 있는 경우 이 값을 **1450**으로 설정해야 합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.

필드	유형	설명
vxlanPort	**integer**	모든 VXLAN 패킷에 사용할 포트입니다. 기본값은 **4789**입니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다. 기존 노드가 다른 VXLAN 네트워크에 속하는 가상 환경에서 실행 중인 경우에는 기본값을 변경해야 할 수도 있습니다. 예를 들어 VMware NSX-T 위에서 OpenShift SDN 오버레이를 실행할 때 두 SDN이 동일한 기본 VXLAN 포트 번호를 사용하므로 VXLAN의 대체 포트를 선택해야 합니다. AWS(Amazon Web Services)에서는 포트 **9000**과 포트 **9999** 사이에서 VXLAN의 대체 포트를 선택할 수 있습니다.

OpenShift SDN 구성 예

```
defaultNetwork:
  type: OpenShiftSDN
  openshiftSDNConfig:
    mode: NetworkPolicy
    mtu: 1450
    vxlanPort: 4789
```

OVN-Kubernetes CNI 클러스터 네트워크 공급자에 대한 구성

다음 표에서는 OVN-Kubernetes CNI 클러스터 네트워크 공급자의 구성 필드를 설명합니다.

표 14.32. ovnKubernetesConfig object

필드	유형	설명

필드	유형	설명
mtu	**integer**	Geneve(Generic Network Virtualization Encapsulation) 오버레이 네트워크의 MTU(최대 전송 단위)입니다. 이는 기본 네트워크 인터페이스의 MTU를 기준으로 자동 탐지됩니다. 일반적으로 감지된 MTU를 재정의할 필요는 없습니다. 자동 감지 값이 예상 밖인 경우 노드의 기본 네트워크 인터페이스의 MTU가 올바른지 확인합니다. 이 옵션을 사용하여 노드의 기본 네트워크 인터페이스의 MTU 값을 변경할 수 없습니다. 클러스터에 다른 노드에 대한 다른 MTU 값이 필요한 경우, 이 값을 클러스터의 가장 낮은 MTU 값보다 **100** 미만으로 설정해야 합니다. 예를 들어, 클러스터의 일부 노드에 **9001**의 MTU가 있고 일부에는 **1500**의 MTU가 있는 경우 이 값을 **1400**으로 설정해야 합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
genevePort	**integer**	모든 Geneve 패킷에 사용할 포트입니다. 기본값은 **6081**입니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
ipsecConfig	**object**	IPsec 암호화를 활성화하려면 빈 오브젝트를 지정합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
policyAuditConfig	**object**	네트워크 정책 감사 로깅을 사용자 정의할 구성 오브젝트를 지정합니다. 설정되지 않으면 기본값 감사 로그 설정이 사용됩니다.

표 14.33. policyAuditConfig object

필드	유형	설명
rateLimit	integer	노드당 1초마다 생성할 최대 메시지 수입니다. 기본값은 초당 **20** 개의 메시지입니다.
maxFileSize	integer	감사 로그의 최대 크기(바이트)입니다. 기본값은 **50000000 또는 50** MB입니다.
대상	string	다음 추가 감사 로그 대상 중 하나입니다. **libc** 호스트에서 journald 프로세스의 libc **syslog()** 함수입니다. **udp:<host>:<port>** syslog 서버입니다. **<host>:<port>**를 syslog 서버의 호스트 및 포트로 바꿉니다. **unix:<file>** **<file>**로 지정된 Unix Domain Socket 파일입니다. **null** 감사 로그를 추가 대상으로 보내지 마십시오.

필드	유형	설명
syslogFacility	string	RFC5424에 정의된 **kern**과 같은 syslog 기능입니다. 기본값은 **local0**입니다.

OVN-Kubernetes 구성 예

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
    ipsecConfig: {}
```

kubeProxyConfig 오브젝트 구성

kubeProxyConfig 오브젝트의 값은 다음 표에 정의되어 있습니다.

표 14.34. kubeProxyConfig object

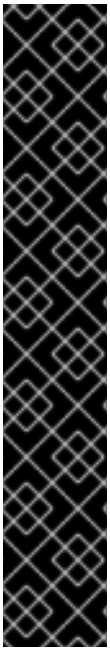
필드	유형	설명
iptablesSyncPeriod	**string**	**iptables** 규칙의 새로 고침 간격입니다. 기본값은 **30s**입니다. 유효 접미사로 **s**, **m**, **h**가 있으며, 자세한 설명은 Go **time** 패키지 문서를 참조하십시오.  **참고** OpenShift Container Platform 4.3 이상에서는 성능이 개선되어 더 이상 **iptablesSyncPeriod** 매개변수를 조정할 필요가 없습니다.

필드	유형	설명
proxyArguments.iptables-min-sync-period	**array**	**iptables** 규칙을 새로 고치기 전 최소 기간입니다. 이 필드를 통해 새로 고침 간격이 너무 짧지 않도록 조정할 수 있습니다. 유효 접미사로 **s**, **m**, **h**가 있으며, 자세한 설명은 Go **time** 패키지를 참조하십시오. 기본값은 다음과 같습니다. kubeProxyConfig: proxyArguments: iptables-min-sync-period: - 0s

14.3.10. Kubernetes 매니페스트 및 Ignition 설정 파일 생성

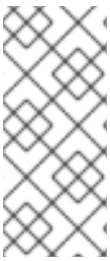
일부 클러스터 정의 파일을 수정하고 클러스터 시스템을 수동으로 시작해야 하므로 클러스터가 시스템을 구성하는 데 필요한 **Kubernetes** 매니페스트 및 **Ignition** 구성 파일을 사용자가 생성해야 합니다.

설치 구성 파일은 **Kubernetes** 매니페스트로 변환됩니다. 매니페스트는 나중에 클러스터 머신을 구성하는 데 사용되는 **Ignition** 구성 파일로 래핑됩니다.



중요

- **OpenShift Container Platform** 설치 프로그램에서 생성하는 **Ignition** 구성 파일에 **24**시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 **24**시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 **kubelet** 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 ***Recovering from expired control plane certificates*** 문서를 참조하십시오.
- 클러스터를 설치한 후 **24**시간에서 **22**시간까지의 인증서가 교체되기 때문에 생성된 후 **12**시간 이내에 **Ignition** 구성 파일을 사용하는 것이 좋습니다. **12**시간 이내에 **Ignition** 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.



참고

매니페스트 및 **Ignition** 파일을 생성하는 설치 프로그램은 아키텍처에 따라 다르며 클라이언트 이미지 미러에서 얻을 수 있습니다. 설치 프로그램의 **Linux** 버전은 **ppc64le**에서만 실행됩니다. 이 설치 프로그램은 **Mac OS** 버전으로도 사용할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램을 가져오셨습니다. 제한된 네트워크 설치의 경우, 해당 파일은 미러 호스트에 있습니다.
- install-config.yaml 설치 구성 파일을 생성하셨습니다.

프로세스

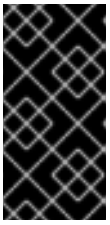
1. OpenShift Container Platform 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터에 대한 Kubernetes 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 <installation_directory>는 사용자가 만든 install-config.yaml 파일이 포함된 설치 디렉터리를 지정합니다.

주의

3 노드 클러스터를 실행 중이면 다음 단계를 건너 뛰어 컨트롤 플레인 노드 일정을 계획할 수 있도록 하십시오.



중요

예약할 수 없는 기본에서 컨트롤 플레인 노드를 구성하면 추가 서브스크립션이 필요합니다. 이는 컨트롤 플레인 노드가 작업자 노드가 되기 때문입니다.

2. <installation_directory>/manifests/cluster-scheduler-02-config.yml Kubernetes 매니페스트 파일의 mastersSchedulable 매개변수가 false로 설정되어 있는지 확인합니다. 이 설정으로 인해 컨트롤 플레인 머신에서 포드가 예약되지 않습니다.

   a. <installation_directory>/manifests/cluster-scheduler-02-config.yml 파일을 엽니다.

b. **mastersSchedulable** 매개변수를 찾아서 값을 **False**로 설정되어 있는지 확인합니다.

c. 파일을 저장하고 종료합니다.

3. **Ignition** 구성 파일을 생성하려면 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행합니다.

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>**는 동일한 설치 디렉터리를 지정합니다.

설치 디렉터리의 부트스트랩, 컨트롤 플레인 및 컴퓨팅 노드에 대한 **Ignition** 구성 파일이 생성됩니다. **kubeadmin-password** 및 **kubeconfig** 파일은 **./<installation_directory>/auth** 디렉터리에 생성됩니다.

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

14.3.11. RHCOS 설치 및 OpenShift Container Platform 부트스트랩 프로세스 시작

프로비저닝하는 **IBM Power** 인프라에 **OpenShift Container Platform**을 설치하려면 머신에 **RHCOS(Red Hat Enterprise Linux CoreOS)**를 설치해야 합니다. **RHCOS**를 설치할 때 설치 중인 머신 유형에 대해 **OpenShift Container Platform** 설치 프로그램에서 생성한 **Ignition** 구성 파일을 제공해야 합니다. 적합한 네트워킹, **DNS** 및 로드 밸런싱 인프라를 구성한 경우 **RHCOS** 머신이 재부팅된 후 **OpenShift Container Platform** 부트스트랩 프로세스가 자동으로 시작됩니다.

단계에 따라 **ISO** 이미지 또는 네트워크 **PXE** 부팅을 사용하여 시스템에 **RHCOS**를 설치합니다.

14.3.11.1. ISO 이미지를 사용하여 RHCOS 설치

ISO 이미지를 사용하여 시스템에 RHCOS을 설치할 수 있습니다.

사전 요구 사항

- 클러스터에 대한 Ignition 구성 파일이 생성되어 있습니다.
- 적합한 네트워크, DNS 및 로드 밸런싱 인프라가 구성되어 있습니다.
- 사용자 컴퓨터에서 액세스할 수 있고 사용자가 생성한 시스템에서 액세스할 수 있는 HTTP 서버에 대한 액세스 권한을 확보합니다.
- 네트워킹 및 디스크 파티션과 같은 기능을 구성하는 다양한 방법에 대해서는 *고급 RHCOS 설치 구성*섹션을 살펴보십시오.

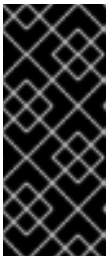
프로세스

1. 각 Ignition 구성 파일에 대해 SHA512 다이제스트를 가져옵니다. 예를 들어 Linux를 실행하는 시스템에서 다음을 사용하여 bootstrap.ign Ignition 구성 파일의 SHA512 다이제스트를 가져올 수 있습니다.

```
$ sha512sum <installation_directory>/bootstrap.ign
```

   다이제스트는 클러스터 노드에서 Ignition 구성 파일의 신뢰성을 검증하기 위해 이후 단계에서 coreos-installer에 제공됩니다.

2. 설치 프로그램에서 생성한 부트스트랩, 컨트롤 플레인, 컴퓨팅 노드 Ignition 구성 파일을 HTTP 서버에 업로드합니다. 해당 파일의 URL을 기록해 둡니다.



   중요

   HTTP 서버에 저장하기 전에 Ignition 구성에서 구성 설정을 추가하거나 변경할 수 있습니다. 설치를 마친 후 클러스터에 컴퓨팅 시스템을 더 추가하려면 Ignition 구성 파일을 삭제하지 마십시오.

3. 설치 호스트에서 Ignition 구성 파일을 URL에서 사용할 수 있는지 확인합니다. 다음 예에서는 부트스트랩 노드에 대한 Ignition 구성 파일을 가져옵니다.

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

출력 예

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-
rsa...
```

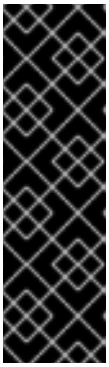
명령에서 **bootstrap.ign**을 **master.ign** 또는 **worker.ign**으로 교체하여 컨트롤 플레인 및 컴퓨팅 노드의 **Ignition** 구성 파일도 사용할 수 있는지 확인합니다.

4. **RHCOS** 이미지 미러 이미지 미러 페이지에서 원하는 운영 체제 인스턴스 설치 방법에 필요한 **RHCOS** 이미지를가져올 수 있지만 **RHCOS** 이미지 의 올바른 버전을 가져오는 권장 방법은 **openshift-install** 명령 출력에서 있습니다.

```
$ openshift-install coreos print-stream-json | grep '\.iso[^.]'
```

출력 예

```
"location": "<url>/art/storage/releases/rhcos-4.9-aarch64/<release>/aarch64/rhcos-
<release>-live.aarch64.iso",
"location": "<url>/art/storage/releases/rhcos-4.9-ppc64le/<release>/ppc64le/rhcos-
<release>-live.ppc64le.iso",
"location": "<url>/art/storage/releases/rhcos-4.9-s390x/<release>/s390x/rhcos-
<release>-live.s390x.iso",
"location": "<url>/art/storage/releases/rhcos-4.9/<release>/x86_64/rhcos-<release>-
live.x86_64.iso",
```



중요

RHCOS 이미지는 OpenShift Container Platform 릴리스에 따라 변경되지 않을 수 있습니다. 설치하는 OpenShift Container Platform 버전과 같거나 그 이하의 버전 중 가장 최신 버전의 이미지를 다운로드해야 합니다. 사용 가능한 경우 OpenShift Container Platform 버전과 일치하는 이미지 버전을 사용합니다. 이 프로세스에는 ISO 이미지만 사용하십시오. 이 설치 유형에서는 RHCOS qcow2 이미지가 지원되지 않습니다.

ISO 파일 이름은 다음 예와 유사합니다.

```
rhcos-<version>-live.<architecture>.iso
```

5. ISO를 사용하여 RHCOS 설치를 시작합니다. 다음 설치 옵션 중 하나를 사용합니다.

- ISO 이미지를 디스크에 굽고 직접 부팅합니다.
- LOM(Lightweight-out Management) 인터페이스를 사용하여 ISO 리디렉션을 사용합니다.

6. 옵션을 지정하거나 라이브 부팅 시퀀스를 중단하지 않고 RHCOS ISO 이미지를 부팅합니다. 설치 프로그램이 RHCOS 라이브 환경에서 쉘 프롬프트로 부팅될 때까지 기다립니다.

참고

커널 인수를 추가하기 위해 RHCOS 설치 부팅 프로세스를 중단할 수 있습니다. 하지만 이 ISO 프로세스에서는 커널 인수를 추가하지 않고 다음 단계에 설명된 대로 coreos-installer 명령을 사용해야 합니다.

7. coreos-installer 명령을 실행하고 설치 요구 사항을 충족하는 옵션을 지정합니다. 최소한 노드 유형에 대한 Ignition 구성 파일과 설치할 장치를 가리키는 URL을 지정해야 합니다.

```
$ sudo coreos-installer install --ignition-url=http://<HTTP_server>/<node_type>.ign
<device> --ignition-hash=sha512-<digest> 1 2
```

1 1

2

클러스터 노드에서 Ignition 구성 파일을 HTTP URL을 통해 가져오려면 --ignition-hash 옵션이 필요합니다. <digest>는 이전 단계에서 얻은 Ignition 구성 파일 SHA512 다이제스트입니다.

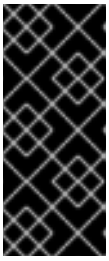
참고

TLS를 사용하는 HTTPS 서버를 통해 Ignition 구성 파일을 제공하려는 경우 coreos-installer를 실행하기 전에 내부 인증 기관(CA)을 시스템 신뢰 저장소에 추가할 수 있습니다.

다음 예제에서는 /dev/sda 장치에 부트스트랩 노드 설치를 초기화합니다. 부트스트랩 노드의 Ignition 구성 파일은 IP 주소 192.168.1.2가 있는 HTTP 웹 서버에서 가져옵니다.

```
$ sudo coreos-installer install --ignition-
url=http://192.168.1.2:80/installation_directory/bootstrap.ign /dev/sda --ignition-
hash=sha512-
a5a2d43879223273c9b60af66b44202a1d1248fc01cf156c46d4a79f552b6bad47bc8cc78dd
f0116e80c59d2ea9e32ba53bc807afbca581aa059311def2c3e3b
```

8. 머신 콘솔에서 RHCOS 설치 진행률을 모니터링합니다.



중요

OpenShift Container Platform 설치를 시작하기 전에 각 노드에서 성공적으로 설치되었는지 확인합니다. 설치 프로세스를 관찰하면 발생할 수 있는 RHCOS 설치 문제의 원인을 파악하는 데 도움이 될 수 있습니다.

9. RHCOS를 설치한 후 시스템을 재부팅해야 합니다. 시스템이 재부팅되는 동안 지정한 Ignition 구성 파일이 적용됩니다.

10. 계속해서 클러스터에 대한 나머지 시스템을 모두 생성합니다.

중요

이때 부트스트랩 및 컨트롤 플레인 시스템을 생성해야 합니다. 컨트롤 플레인 시스템이 예약 가능하지 않은 경우 **OpenShift Container Platform**을 설치하기 전에 두 개 이상의 컴퓨팅 시스템도 생성합니다.

적합한 네트워킹, **DNS** 및 로드 밸런싱 인프라가 있는 경우 **RHCOS** 노드가 재부팅된 후 **OpenShift Container Platform** 부트스트랩 프로세스가 자동으로 시작됩니다.

참고

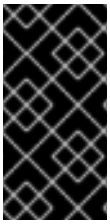
RHCOS 노드에는 **core** 사용자의 기본 암호가 포함되지어 있지 않습니다. **install_config.yaml** 파일에 지정한 공개 키에 대한 액세스 권한이 있는 사용자로 **ssh core@<node>.<cluster_name>.<base_domain>**을 실행하여 노드에 액세스할 수 있습니다. **RHCOS**를 실행하는 **OpenShift Container Platform 4** 클러스터 노드는 변경할 수 없으며 **Operator**를 통해 클러스터 변경 사항을 적용합니다. **SSH**를 사용하여 클러스터 노드에 액세스하는 것은 권장되지 않습니다. 그러나 설치 문제를 조사할 때 **OpenShift Container Platform API**를 사용할 수 없거나 **kubelet**이 대상 노드에서 제대로 작동하지 않는 경우 디버깅 또는 재해 복구에 **SSH** 액세스가 필요할 수 있습니다.

14.3.11.1.1. 고급 RHCOS 설치 참조

여기서는 **RHCOS(Red Hat Enterprise Linux CoreOS)** 수동 설치 프로세스를 수정하는 데 사용할 수 있는 네트워킹 구성 및 기타 고급 옵션에 대해 설명합니다. 다음 표에서는 **RHCOS** 라이브 설치 프로그램 및 **coreos-installer** 명령과 함께 사용할 수있는 커널 인수 및 명령 줄 옵션에 대해 설명합니다.

14.3.11.1.1.1. ISO 설치를 위한 네트워킹 및 본딩 옵션

ISO 이미지에서 **RHCOS**를 설치하는 경우, 해당 이미지를 부팅할 때 수동으로 커널 인수를 추가하여 노드의 네트워킹을 구성할 수 있습니다. 네트워킹 인수를 지정하지 않으면 **RHCOS**에서 **Ignition** 구성 파일을 가져오는 데 네트워킹이 필요함을 감지하면 **initramfs**에서 **DHCP**가 활성화됩니다.



중요

네트워킹 인수를 수동으로 추가할 때 **initramfs**에서 네트워크를 가져오려면 **rd.neednet=1** 커널 인수도 추가해야 합니다.

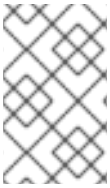
다음 표는 **ISO** 설치를 위해 **RHCOS(Red Hat Enterprise Linux CoreOS)** 노드의 네트워킹 및 본딩 구성 예를 보여줍니다. 예제에서는 **ip=, nameserver=, bond=** 커널 인수를 사용하는 방법을 설명합니다.

참고

커널 인수를 추가할 때 순서가 중요합니다: **ip=, nameserver=** 및 **bond=** 입니다.

이는 시스템 부팅 중에 **dracut** 툴로 전달되는 네트워킹 옵션입니다. **dracut**에서 지원하는 네트워킹 옵션에 대한 자세한 내용은 **dracut.cmdline** 메뉴얼 페이지를 참조하십시오.

다음 정보는 **ISO** 설치를 위해 **RHCOS** 노드에서 네트워킹을 구성하는 예제를 제공합니다. 예제에서는 **ip=** 및 **nameserver=** 커널 인수를 사용하는 방법을 설명합니다.



참고

커널 인수를 추가할 때 순서가 중요합니다(**ip =** 및 **nameserver= **).

이는 시스템 부팅 중에 **dracut** 툴로 전달되는 네트워킹 옵션입니다. **dracut**에서 지원하는 네트워킹 옵션에 대한 자세한 내용은 **dracut.cmdline** 메뉴얼 페이지를 참조하십시오.

다음 예제는 **ISO** 설치를 위한 네트워킹 옵션입니다.

DHCP 또는 고정 **IP** 주소 구성

IP 주소를 구성하려면 **DHCP(ip=dhcp)**를 사용하거나 개별 고정 **IP** 주소(**ip=<host_ip>**)를 설정합니다. 정적 **IP**를 설정하는 경우 각 노드에서 **DNS** 서버 **IP** 주소 (**nameserver=<dns_ip>**)를 확인합니다. 다음 예제에서는 다음을 설정합니다.

- 노드의 **IP** 주소는 **10.10.10.2**로 설정됩니다.
- 게이트웨이 주소는 **10.10.10.254**로 설정됩니다.
- 넷마스크는 **255.255.255.0**으로
- **core0.example.com**에 대한 호스트 이름

- 4.4.4.41의 DNS 서버 주소

- auto-configuration 값을 none으로 설정합니다. IP 네트워킹이 정적으로 구성되면 자동 구성이 필요하지 않습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
nameserver=4.4.4.41
```

참고

DHCP를 사용하여 RHCOS 시스템의 IP 주소 지정을 구성하는 경우 시스템은 DHCP를 통해 DNS 서버 정보도 가져옵니다. DHCP 기반 배포의 경우 DHCP 서버 구성을 통해 RHCOS 노드에서 사용할 DNS 서버 주소를 정의할 수 있습니다.

고정 호스트 이름 없이 IP 주소 구성

정적 호스트 이름을 할당하지 않고 IP 주소를 구성할 수 있습니다. 사용자가 정적 호스트 이름을 설정하지 않으면 역방향 DNS 조회에 의해 자동으로 선택됩니다. 정적 호스트 이름이 없는 IP 주소를 구성하려면 다음 예제를 참조합니다.

- 노드의 IP 주소는 10.10.10.2로 설정됩니다.

- 게이트웨이 주소는 10.10.10.254로 설정됩니다.

- 넷마스크는 255.255.255.0으로

- 4.4.4.41의 DNS 서버 주소

- auto-configuration 값을 none으로 설정합니다. IP 네트워킹이 정적으로 구성되면 자동 구성이 필요하지 않습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0::enp1s0:none
nameserver=4.4.4.41
```

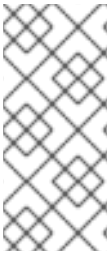
여러 네트워크 인터페이스 지정

여러 ip= 항목을 설정하여 여러 네트워크 인터페이스를 지정할 수 있습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

기본 게이트웨이 및 경로 구성

선택 사항: **rd.route=** 값을 설정하여 추가 네트워크에 대한 경로를 구성할 수 있습니다.



참고

하나 이상의 네트워크를 구성할 때 하나의 기본 게이트웨이가 필요합니다. 추가 네트워크 게이트웨이가 기본 네트워크 게이트웨이와 다른 경우 기본 게이트웨이가 기본 네트워크 게이트웨이어야 합니다.

- 다음 명령을 실행하여 기본 게이트웨이를 구성합니다.

```
ip=::10.10.10.254::::
```

- 다음 명령을 입력하여 추가 네트워크의 경로를 구성합니다.

```
rd.route=20.20.20.0/24:20.20.20.254:enp2s0
```

단일 인터페이스에서 **DHCP** 비활성화

두 개 이상의 네트워크 인터페이스가 있고 하나의 인터페이스만 사용되는 경우와 같이 단일 인터페이스에서 **DHCP**를 비활성화할 수 있습니다. 이 예에서 **enp1s0** 인터페이스에는 정적 네트워킹 구성이 있으며 **enp2s0** 용으로 **DHCP**가 사용되지 않습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=::::core0.example.com:enp2s0:none
```

DHCP 및 고정 **IP** 구성 결합

다음과 같이 시스템의 **DHCP** 및 고정 **IP** 구성을 여러 네트워크 인터페이스와 결합할 수 있습니다.

```
ip=enp1s0:dhcp
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

개별 인터페이스에서 **VLAN** 구성

선택 사항: **vlan=** 매개변수를 사용하여 개별 인터페이스에서 **VLAN**을 구성할 수 있습니다.

- 네트워크 인터페이스에서 **VLAN**을 구성하고 고정 **IP** 주소를 사용하려면 다음 명령을 실행

합니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0.100:none
vlan=enp2s0.100:enp2s0
```

- 네트워크 인터페이스에서 VLAN을 구성하고 DHCP를 사용하려면 다음 명령을 실행합니다.

```
ip=enp2s0.100:dhcp
vlan=enp2s0.100:enp2s0
```

여러 DNS 서버 제공

각 서버에 nameserver= 항목을 추가하여 여러 DNS 서버를 제공할 수 있습니다. 예를 들면 다음과 같습니다.

```
nameserver=1.1.1.1
nameserver=8.8.8.8
```

단일 인터페이스에 여러 네트워크 인터페이스 본딩

선택 사항: bond= 옵션을 사용하여 여러 네트워크 인터페이스를 단일 인터페이스에 결합할 수 있습니다. 다음 예제를 참조하십시오.

- 결합된 인터페이스를 구성하는 구문: bond = name [: network_interfaces] [: options]

*name*은 결합하는 기기 이름(bond0)이고 *network_interfaces*는 쉼표로 구분되는 물리적(이더넷) 인터페이스 목록(em1, em2)이며, *options*은 쉼표로 구분되는 결합 옵션 목록입니다. 사용 가능한 옵션을 보려면 modinfo bonding을 입력하십시오.

- bond=를 사용하여 결합된 인터페이스를 생성할 때 IP 주소가 할당되는 방법과 결합된 인터페이스에 대한 기타 정보를 지정해야 합니다.

- DHCP를 사용하도록 결합된 인터페이스를 구성하려면 bond의 IP 주소를 dhcp로 설정하십시오. 예를 들면 다음과 같습니다.

```
bond=bond0:em1,em2:mode=active-backup
ip=bond0:dhcp
```

- 고정 IP 주소를 사용하도록 결합된 인터페이스를 구성하려면 원하는 특정 IP 주소 및 관련 정보를 입력합니다. 예를 들면 다음과 같습니다.

```
bond=bond0:em1,em2:mode=active-backup
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
```

단일 인터페이스에 여러 네트워크 인터페이스 본딩

선택 사항: vlan= 매개변수를 사용하고 DHCP를 사용하도록 결합된 인터페이스에서 VLAN을 구성할 수 있습니다. 예를 들면 다음과 같습니다.

```
ip=bond0.100:dhcp
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

다음 예제를 사용하여 VLAN을 사용하여 결합된 인터페이스를 구성하고 고정 IP 주소를 사용합니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0.100:none
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

네트워크 팀 구성 사용

선택 사항: team= 매개변수를 사용하여 네트워크 팀을 결합의 대안으로 사용할 수 있습니다.

- 팀 인터페이스를 구성하는 구문은 team=name[:network_interfaces]입니다.

  name 은 팀 장치 이름(team0)이고 *network_interfaces* 는 쉼표로 구분된 실제(이더넷) 인터페이스 목록(em1, em2)을 나타냅니다.

RHCOS가 향후 RHEL 버전으로 전환되면 티밍이 더 이상 사용되지 않을 예정입니다. 자세한 내용은 이 Red Hat 지식베이스 문서를 참조하십시오.

다음 예제를 사용하여 네트워크 팀을 구성합니다.

```
team=team0:em1,em2
ip=team0:dhcp
```

14.3.11.2. PXE 부팅을 사용하여 RHCOS 설치

PXE 부팅을 사용하여 시스템에 RHCOS를 설치할 수 있습니다.

사전 요구 사항

- 클러스터에 대한 Ignition 구성 파일이 생성되어 있습니다.
- 적합한 네트워크, DNS 및 로드 밸런싱 인프라가 구성되어 있습니다.
- 적합한 PXE 인프라가 구성되어 있습니다.
- 사용자 컴퓨터에서 액세스할 수 있고 사용자가 생성한 시스템에서 액세스할 수 있는 HTTP 서버에 대한 액세스 권한을 확보합니다.
- 네트워킹 및 디스크 파티션과 같은 기능을 구성하는 다양한 방법에 대해서는 *고급 RHCOS 설치 구성*섹션을 살펴보십시오.

프로세스

1. 설치 프로그램에서 생성한 부트스트랩, 컨트롤 플레인, 컴퓨팅 노드 Ignition 구성 파일을 HTTP 서버에 업로드합니다. 해당 파일의 URL을 기록해 둡니다.

중요

HTTP 서버에 저장하기 전에 Ignition 구성에서 구성 설정을 추가하거나 변경할 수 있습니다. 설치를 마친 후 클러스터에 컴퓨팅 시스템을 더 추가하려면 Ignition 구성 파일을 삭제하지 마십시오.

2. 설치 호스트에서 Ignition 구성 파일을 URL에서 사용할 수 있는지 확인합니다. 다음 예에서는 부트스트랩 노드에 대한 Ignition 구성 파일을 가져옵니다.

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

출력 예

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-
rsa...
```

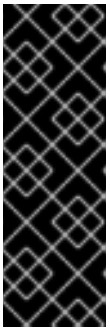
명령에서 **bootstrap.ign**을 **master.ign** 또는 **worker.ign**으로 교체하여 컨트롤 플레인 및 컴퓨팅 노드의 **Ignition** 구성 파일도 사용할 수 있는지 확인합니다.

3. **RHCOS** 커널,**initramfs** 및 **rootfs** 파일을 **RHCOS** 이미지 미러페이지에서 원하는 운영 체제 인스턴스 설치 방법에 사용할 수 있지만 **RHCOS** 파일의 올바른 버전을 얻는 것이 **openshift-install** 명령 출력에서 제공됩니다.

```
$ openshift-install coreos print-stream-json | grep -Eo '"https.*(kernel-
|initramfs.|rootfs.)\w+(\.img)?"'
```

출력 예

```
"<url>/art/storage/releases/rhcos-4.9-aarch64/<release>/aarch64/rhcos-<release>-live-
kernel-aarch64"
"<url>/art/storage/releases/rhcos-4.9-aarch64/<release>/aarch64/rhcos-<release>-live-
initramfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.9-aarch64/<release>/aarch64/rhcos-<release>-live-
rootfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.9-ppc64le/49.84.202110081256-0/ppc64le/rhcos-
<release>-live-kernel-ppc64le"
"<url>/art/storage/releases/rhcos-4.9-ppc64le/<release>/ppc64le/rhcos-<release>-live-
initramfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.9-ppc64le/<release>/ppc64le/rhcos-<release>-live-
rootfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.9-s390x/<release>/s390x/rhcos-<release>-live-
kernel-s390x"
"<url>/art/storage/releases/rhcos-4.9-s390x/<release>/s390x/rhcos-<release>-live-
initramfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.9-s390x/<release>/s390x/rhcos-<release>-live-
rootfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.9/<release>/x86_64/rhcos-<release>-live-kernel-
x86_64"
"<url>/art/storage/releases/rhcos-4.9/<release>/x86_64/rhcos-<release>-live-
initramfs.x86_64.img"
"<url>/art/storage/releases/rhcos-4.9/<release>/x86_64/rhcos-<release>-live-
rootfs.x86_64.img"
```



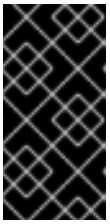
중요

OpenShift Container Platform의 모든 릴리스에서 **RHCOS** 아티팩트가 변경되지 않을 수도 있습니다. 설치하는 **OpenShift Container Platform** 버전과 같거나 그 이하의 버전 중 가장 최신 버전의 이미지를 다운로드해야 합니다. 이 프로시저에는 아래 설명된 적절한 **kernel, initramfs** 및 **rootfs** 아티팩트만 사용하십시오. 이 설치 유형에서는 **RHCOS QCOW2** 이미지가 지원되지 않습니다.

OpenShift Container Platform 버전 번호가 파일 이름에 포함됩니다. 다음 예와 유사합니다.

- **kernel: rhcos-<version>-live-kernel-<architecture>**
- **initramfs: rhcos-<version>-live-initramfs.<architecture>.img**
- **rootfs: rhcos-<version>-live-rootfs.<architecture>.img**

4. **rootfs, kernel** 및 **initramfs** 파일을 **HTTP** 서버에 업로드합니다.



중요

설치를 마친 후 클러스터에 컴퓨팅 시스템을 더 추가하려면 **Ignition** 구성 파일을 삭제하지 마십시오.

5. **RHCOS**가 설치된 후 시스템이 로컬 디스크에서 부팅되도록 네트워크 부팅 인프라를 구성하십시오.

6. **RHCOS** 이미지에 대한 **PXE** 설치를 구성하고 설치를 시작하십시오.

사용 환경에 대한 다음 예시 메뉴 항목을 수정하고, 이미지 및 **Ignition** 파일에 적절히 접근할 수 있는지 확인하십시오.

```
DEFAULT pxeboot
TIMEOUT 20
PROMPT 0
LABEL pxeboot
```

```
    KERNEL http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> (1)
    APPEND initrd=http://<HTTP_server>/rhcos-<version>-live-initramfs.<architecture>.img
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.<architecture>.img
coreos.inst.install_dev=/dev/sda coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign
(2) (3)
```

(1) (1)

HTTP 서버에 업로드한 라이브 **kernel** 파일의 위치를 지정합니다. **URL**은 **HTTP**, **TFTP** 또는 **FTP**여야 합니다. **HTTPS**와 **NFS**는 지원되지 않습니다.

(2)

NIC를 여러 개 사용하는 경우 **ip** 옵션에 단일 인터페이스를 지정합니다. 예를 들어, **eno1**라는 **NIC**에서 **DHCP**를 사용하려면 **ip=eno1:dhcp**를 설정하십시오.

(3)

HTTP 서버에 업로드한 **RHCOS** 파일의 위치를 지정합니다. **initrd** 매개변수 값은 **initramfs** 파일의 위치, **coreos.live.rootfs_url** 매개변수 값은 **rootfs** 파일의 위치, **coreos.inst.ignition_url** 매개변수 값은 부트스트랩 **Ignition** 구성 파일의 위치입니다. **APPEND** 줄에 커널 인수를 더 추가하여 네트워킹 또는 기타 부팅 옵션도 구성할 수 있습니다.

참고

이 구성은 그래픽 콘솔이 있는 머신에서 직렬 콘솔 액세스를 활성화하지 않습니다. 다른 콘솔을 구성하려면 **APPEND** 행에 하나 이상의 **console=** 인수를 추가합니다. 예를 들어 **console=tty0 console=ttyS0**을 추가하여 첫 번째 **PC** 직렬 포트를 기본 콘솔로 설정하고 그래픽 콘솔을 보조 콘솔로 설정합니다. 자세한 내용은 **Red Hat Enterprise Linux**에서 직렬 터미널 및/또는 콘솔 설정 방법을 참조하십시오.

7. 머신 콘솔에서 **RHCOS** 설치 진행률을 모니터링합니다.

중요

OpenShift Container Platform 설치를 시작하기 전에 각 노드에서 성공적으로 설치되었는지 확인합니다. 설치 프로세스를 관찰하면 발생할 수 있는 **RHCOS** 설치 문제의 원인을 파악하는 데 도움이 될 수 있습니다.

8. **RHCOS**가 설치되면 시스템을 재부팅합니다. 시스템이 재부팅되는 동안 지정한 **Ignition** 구성 파일이 적용됩니다.

9. 클러스터용 시스템 생성을 계속합니다.

중요

이때 부트스트랩 및 컨트롤 플레인 시스템을 생성해야 합니다. 컨트롤 플레인 시스템이 예약 가능하지 않은 경우 클러스터를 설치하기 전에 두 개 이상의 컴퓨팅 시스템도 생성합니다.

적합한 네트워킹, DNS 및 로드 밸런싱 인프라가 있는 경우 RHCOS 노드가 재부팅된 후 OpenShift Container Platform 부트스트랩 프로세스가 자동으로 시작됩니다.

참고

RHCOS 노드에는 core 사용자의 기본 암호가 포함되지어 있지 않습니다. install_config.yaml 파일에 지정한 공개 키에 대한 액세스 권한이 있는 사용자로 ssh core@<node>.<cluster_name>.<base_domain>을 실행하여 노드에 액세스할 수 있습니다. RHCOS를 실행하는 OpenShift Container Platform 4 클러스터 노드는 변경할 수 없으며 Operator를 통해 클러스터 변경 사항을 적용합니다. SSH를 사용하여 클러스터 노드에 액세스하는 것은 권장되지 않습니다. 그러나 설치 문제를 조사할 때 OpenShift Container Platform API를 사용할 수 없거나 kubelet이 대상 노드에서 제대로 작동하지 않는 경우 디버깅 또는 재해 복구에 SSH 액세스가 필요할 수 있습니다.

14.3.11.3. RHCOS에서 커널 인수로 다중 경로 활성화

OpenShift Container Platform 4.9 이상에서는 설치 중에 프로비저닝된 노드에 대해 멀티패스를 활성화할 수 있습니다. RHCOS는 기본 디스크에서 멀티패스를 지원합니다. 멀티패스는 호스트 가용성을 향상시키기 위해 하드웨어 장애에 대한 복원력 강화의 추가 이점을 제공합니다.

초기 클러스터 생성 중에 모든 마스터 또는 작업자 노드에 커널 인수를 추가할 수 있습니다. 마스터 노드 또는 작업자 노드에 커널 매개 변수를 추가하기 위해 MachineConfig 객체를 생성하고 해당 객체를 클러스터 설정 중에 Ignition에서 사용하는 매니페스트 파일 세트에 삽입할 수 있습니다.

프로세스

1. 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터에 대한 Kubernetes 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory>
```

2. 커널 매개 변수를 작업자 또는 컨트롤 플레인 노드에 추가할지 여부를 결정합니다.

- 머신 구성 파일을 생성합니다. 예를 들어 클러스터에 **master** 레이블을 추가하고 멀티패스 커널 인수를 식별하도록 지시하는 **99-master-kargs-mpath.yaml**을 생성합니다.

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: "master"
  name: 99-master-kargs-mpath
spec:
  kernelArguments:
    - 'rd.multipath=default'
    - 'root=/dev/disk/by-label/dm-mpath-root'
```

3. 작업자 노드에서 멀티패스 설치 후 활성화하려면 다음을 수행합니다.

- 머신 구성 파일을 생성합니다. 예를 들어 클러스터에 **worker** 레이블을 추가하고 멀티패스 커널 인수를 식별하도록 지시하는 **99-worker-kargs-mpath.yaml** 을 생성합니다.

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: "worker"
  name: 99-worker-kargs-mpath
spec:
  kernelArguments:
    - 'rd.multipath=default'
    - 'root=/dev/disk/by-label/dm-mpath-root'
```

이제 계속해서 클러스터를 만들 수 있습니다.

중요

멀티패스를 완전히 활성화하려면 추가 설치 후 단계가 필요합니다. 자세한 내용은 *설치 후 머신 구성 작업*의 **"RHCOS에서 커널 인수를 사용하여 다중 경로 활성화"**를 참조하십시오.

MPIO 오류가 발생하는 경우 bootlist 명령을 사용하여 대체 논리 장치 이름으로 부팅 장치 목록을 업데이트합니다. 명령은 부팅 목록을 표시하고 시스템을 일반 모드로 부팅할 때 에 대해 가능한 부팅 장치를 지정합니다.

a. 부팅 목록을 표시하고 시스템이 일반 모드로 부팅되는 경우 가능한 부팅 장치를 지정하도록 다음 명령을 입력합니다.

```
$ bootlist -m normal -o
sda
```

b. 일반 모드의 부팅 목록을 업데이트하고 대체 장치 이름을 추가하려면 다음 명령을 입력합니다.

```
$ bootlist -m normal -o /dev/sdc /dev/sdd /dev/sde
sdc
sdd
sde
```

원래 부팅 디스크 경로가 다운되면 노드가 일반 부팅 장치 목록에 등록된 대체 장치에서 재부팅됩니다.

14.3.12. 부트스트랩 프로세스가 완료될 때까지 대기 중

OpenShift Container Platform 부트스트랩 프로세스는 클러스터 노드가 먼저 디스크에 설치된 영구 RHCOS 환경으로 부팅된 후에 시작됩니다. Ignition 구성 파일을 통해 제공되는 구성 정보는 부트스트랩 프로세스를 초기화하고 머신에 OpenShift Container Platform을 설치하는 데 사용됩니다. 부트스트랩 프로세스가 완료될 때까지 기다려야 합니다.

사전 요구 사항

- 클러스터에 대한 Ignition 구성 파일이 생성되어 있습니다.
- 적합한 네트워크, DNS 및 로드 밸런싱 인프라가 구성되어 있습니다.
- 설치 프로그램을 받아서 클러스터의 Ignition 구성 파일을 생성했습니다.
- 클러스터 머신에 RHCOS를 설치하고 OpenShift Container Platform 설치 프로그램에서 생성된 Ignition 구성 파일을 제공했습니다.

프로세스

1. 부트스트랩 프로세스를 모니터링합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
  --log-level=info 2
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2 다른 설치 세부 사항을 보려면 info 대신 warn, debug 또는 error를 지정합니다.

출력 예

```
INFO Waiting up to 30m0s for the Kubernetes API at
https://api.test.example.com:6443...
INFO API v1.22.1 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

이 명령은 Kubernetes API 서버가 컨트롤 플레인 시스템에서 부트스트랩되었다는 신호를 보낼 때 성공합니다.

2. 부트스트랩 프로세스가 완료된 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다.

중요

이 시점에 로드 밸런서에서 부트스트랩 시스템을 제거해야 합니다. 부트스트랩 머신 자체를 제거하거나 다시 포맷할 수도 있습니다.

14.3.13. CLI를 사용하여 클러스터에 로그인

클러스터 kubeconfig 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. kubeconfig 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- oc CLI를 설치했습니다.

프로세스

1. kubeadmin 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 oc 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

14.3.14. 머신의 인증서 서명 요청 승인

클러스터에 시스템을 추가하면 추가한 시스템별로 보류 중인 인증서 서명 요청(CSR)이 두 개씩 생성됩니다. 이러한 CSR이 승인되었는지 확인해야 하며, 필요한 경우 이를 직접 승인해야 합니다. 클라이언트

요청을 먼저 승인한 다음 서버 요청을 승인해야 합니다.

사전 요구 사항

- 클러스터에 시스템을 추가했습니다.

절차

1. 클러스터가 시스템을 인식하는지 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0  Ready    master  63m  v1.22.1
master-1  Ready    master  63m  v1.22.1
master-2  Ready    master  64m  v1.22.1
```

출력에 생성된 모든 시스템이 나열됩니다.

참고

이전 출력에는 일부 **CSR**이 승인될 때까지 컴퓨팅 노드(작업자 노드라고도 함)가 포함되지 않을 수 있습니다.

2. 보류 중인 **CSR**을 검토하고 클러스터에 추가한 각 시스템에 대해 **Pending** 또는 **Approved** 상태의 클라이언트 및 서버 요청이 표시되는지 확인합니다.

```
$ oc get csr
```

출력 예

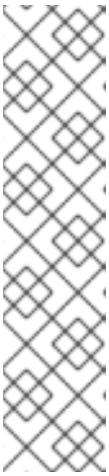
```
NAME        AGE    REQUESTOR                                                   CONDITION
csr-8b2br   15m    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

예에서는 두 시스템이 클러스터에 참여하고 있습니다. 목록에는 승인된 CSR이 더 많이 나타날 수도 있습니다.

3. CSR이 승인되지 않은 경우, 추가된 시스템에 대한 모든 보류 중인 CSR이 Pending 상태로 전환된 후 클러스터 시스템의 CSR을 승인합니다.

참고

CSR은 교체 주기가 자동으로 만료되므로 클러스터에 시스템을 추가한 후 1 시간 이내에 CSR을 승인하십시오. 한 시간 내에 승인하지 않으면 인증서가 교체되고 각 노드에 대해 두 개 이상의 인증서가 표시됩니다. 이러한 인증서를 모두 승인해야 합니다. 클라이언트 CSR이 승인되면 Kubelet은 인증서에 대한 보조 CSR을 생성하므로 수동 승인이 필요합니다. 그러면 Kubelet에서 동일한 매개변수를 사용하여 새 인증서를 요청하는 경우 인증서 갱신 요청은 machine-approver에 의해 자동으로 승인됩니다.



참고

베어 메탈 및 기타 사용자 프로비저닝 인프라와 같이 머신 API를 사용하도록 활성화되지 않는 플랫폼에서 실행되는 클러스터의 경우 CSR(Kubelet service Certificate Request)을 자동으로 승인하는 방법을 구현해야 합니다. 요청이 승인되지 않으면 API 서버가 kubelet에 연결될 때 서비스 인증서가 필요하므로 oc exec, oc rsh, oc logs 명령을 성공적으로 수행할 수 없습니다. Kubelet 엔드 포인트에 연결하는 모든 작업을 수행하려면 이 인증서 승인이 필요합니다. 이 방법은 새 CSR을 감시하고 CSR이 system:node 또는 system:admin 그룹의 node-bootstrapper 서비스 계정에 의해 제출되었는지 확인하고 노드의 ID를 확인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

참고

일부 Operator는 일부 CSR이 승인될 때까지 사용할 수 없습니다.

4. 이제 클라이언트 요청이 승인되었으므로 클러스터에 추가한 각 머신의 서버 요청을 검토해야 합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 나머지 CSR이 승인되지 않고 Pending 상태인 경우 클러스터 머신의 CSR을 승인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 <csr_name>은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs oc adm certificate approve
```

6. 모든 클라이언트 및 서버 CSR이 승인된 후 머신은 Ready 상태가 됩니다. 다음 명령을 실행하여 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0  Ready    master  73m  v1.22.1
master-1  Ready    master  73m  v1.22.1
master-2  Ready    master  74m  v1.22.1
worker-0  Ready    worker  11m  v1.22.1
worker-1  Ready    worker  11m  v1.22.1
```

참고

머신이 Ready 상태로 전환하는 데 서버 CSR의 승인 후 몇 분이 걸릴 수 있습니다.

추가 정보

- CSR에 대한 자세한 내용은 인증서 서명 요청을 참조하십시오.

14.3.15. Operator의 초기 설정

컨트롤 플레인이 초기화된 후 일부 Operator를 즉시 구성하여 모두 사용 가능하도록 해야 합니다.

사전 요구 사항

- 컨트롤 플레인이 초기화되어 있습니다.

프로세스

1. 클러스터 구성 요소가 온라인 상태인지 확인합니다.

```
$ watch -n5 oc get clusteroperators
```

출력 예

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.9.0     True        False         False      19m
baremetal                                  4.9.0     True        False         False      37m
cloud-credential                           4.9.0     True        False         False      40m
cluster-autoscaler                         4.9.0     True        False         False      37m
config-operator                            4.9.0     True        False         False      38m
console                                    4.9.0     True        False         False      26m
csi-snapshot-controller                    4.9.0     True        False         False      37m
dns                                        4.9.0     True        False         False      37m
etcd                                       4.9.0     True        False         False      36m
image-registry                             4.9.0     True        False         False      31m
ingress                                    4.9.0     True        False         False      30m
insights                                   4.9.0     True        False         False      31m
kube-apiserver                             4.9.0     True        False         False      26m
kube-controller-manager                    4.9.0     True        False         False      36m
kube-scheduler                             4.9.0     True        False         False      36m
kube-storage-version-migrator              4.9.0     True        False         False      37m
machine-api                                4.9.0     True        False         False      29m
machine-approver                           4.9.0     True        False         False      37m
machine-config                             4.9.0     True        False         False      36m
marketplace                                4.9.0     True        False         False      37m
monitoring                                 4.9.0     True        False         False      29m
network                                    4.9.0     True        False         False      38m
node-tuning                                4.9.0     True        False         False      37m
openshift-apiserver                        4.9.0     True        False         False      32m
openshift-controller-manager               4.9.0     True        False         False      30m
openshift-samples                          4.9.0     True        False         False      32m
operator-lifecycle-manager                 4.9.0     True        False         False      37m
operator-lifecycle-manager-catalog         4.9.0     True        False         False      37m
operator-lifecycle-manager-packageserver   4.9.0     True        False         False      32m
service-ca                                 4.9.0     True        False         False      38m
storage                                    4.9.0     True        False         False      37m
```

2. 사용할 수 없는 Operator를 구성합니다.

14.3.15.1. 기본 OperatorHub 소스 비활성화

Red Hat 및 커뮤니티 프로젝트에서 제공하는 콘텐츠를 소싱하는 Operator 카탈로그는 OpenShift Container Platform을 설치하는 동안 기본적으로 OperatorHub용으로 구성됩니다. 제한된 네트워크 환경에서는 클러스터 관리자로서 기본 카탈로그를 비활성화해야 합니다.

프로세스

- OperatorHub 오브젝트에 disableAllDefaultSources: true를 추가하여 기본 카탈로그의 소스를 비활성화합니다.

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

작은 정보

또는 웹 콘솔을 사용하여 카탈로그 소스를 관리할 수 있습니다. 관리 → 클러스터 설정 → 구성 → OperatorHub 페이지에서 개별 소스를 생성, 삭제, 비활성화 및 활성화할 수 있는 소스 탭을 클릭합니다.

14.3.15.2. 이미지 레지스트리 스토리지 구성

기본 스토리지를 제공하지 않는 플랫폼에서는 처음에 Image Registry Operator를 사용할 수 없습니다. 설치한 후에 스토리지를 사용하도록 레지스트리를 구성하여 Registry Operator를 사용 가능하도록 만들어야 합니다.

프로덕션 클러스터에 필요한 영구 볼륨을 구성하는 과정의 지침이 표시됩니다. 해당하는 경우, 프로덕션 환경 외 클러스터에서만 사용할 수 있는 저장 위치로서 빈 디렉터리를 구성하는 과정의 지침이 표시됩니다.

업그레이드 중에 Recreate 롤아웃 전략을 사용하여 이미지 레지스트리의 블록 스토리지 유형 사용을 허용하기 위한 추가 지침이 제공됩니다.

14.3.15.2.1. 이미지 레지스트리의 관리 상태 변경

이미지 레지스트리를 시작하려면 Image Registry Operator 구성의 managementState를 Removed에서 Managed로 변경해야 합니다.

프로세스

-

managementState Image Registry Operator 구성을 Removed에서 Managed로 변경합니다. 예를 들면 다음과 같습니다.

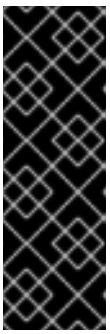
```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch
'{"spec":{"managementState":"Managed"}}'
```

14.3.15.2.2. IBM Power 용 레지스트리 스토리지 구성

클러스터 관리자는 설치한 후 스토리지를 사용하도록 레지스트리를 구성해야 합니다.

사전 요구 사항

- cluster-admin 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- IBM Power에 클러스터가 있습니다.
- Red Hat OpenShift Container Storage와 같이 클러스터용 영구 스토리지 프로비저닝이 있습니다.



중요

OpenShift Container Platform은 복제본이 하나만 있는 경우 이미지 레지스트리 스토리지에 대한 ReadWriteOnce 액세스를 지원합니다. ReadWriteOnce 액세스에서는 레지스트리가 Recreate 롤아웃 전략을 사용해야 합니다. 두 개 이상의 복제본으로 고 가용성을 지원하는 이미지 레지스트리를 배포하려면 ReadWriteMany 액세스가 필요합니다.

- "100Gi" 용량이 필요합니다.

프로세스

1. 스토리지를 사용하도록 레지스트리를 구성하기 위해 configs.imageregistry/cluster 리소스에서 spec.storage.pvc를 변경합니다.

참고

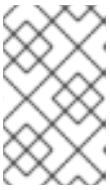
공유 스토리지를 사용할 때 보안 설정을 확인하여 외부에서의 액세스를 방지합니다.

2. 레지스트리 **pod**가 없는지 확인합니다.

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

출력 예

```
No resourses found in openshift-image-registry namespace
```



참고

출력에 레지스트리 **Pod**가 있는 경우 이 절차를 계속할 필요가 없습니다.

3. 레지스트리 구성을 확인합니다.

```
$ oc edit configs.imageregistry.operator.openshift.io
```

출력 예

```
storage:
  pvc:
    claim:
```

image-registry-storage PVC의 자동 생성을 허용하도록 **claim** 필드를 비워 둡니다.

4. **clusteroperator 상태를 확인합니다.**

```
$ oc get clusteroperator image-registry
```

출력 예

```
NAME             VERSION   AVAILABLE   PROGRESSING   DEGRADED   SINCE   MESSAGE
image-registry   4.9       True        False         False      6h50m
```

5. **이미지를 빌드 및 푸시할 수 있도록 레지스트리의 관리가 설정되어 있는지 확인하십시오.**

- **다음을 실행합니다.**

```
$ oc edit configs.imageregistry/cluster
```

다음으로 라인을 변경하십시오.

```
managementState: Removed
```

다음으로 변경

```
managementState: Managed
```

14.3.15.2.3. 프로덕션 환경 외 클러스터에서 이미지 레지스트리의 스토리지 구성

이미지 레지스트리 Operator에 대한 스토리지를 구성해야 합니다. 프로덕션 환경 외 클러스터의 경우, 이미지 레지스트리를 빈 디렉터리로 설정할 수 있습니다. 이렇게 하는 경우 레지스트리를 다시 시작하면 모든 이미지가 손실됩니다.

프로세스

- **이미지 레지스트리 스토리지를 빈 디렉터리로 설정하려면 다음을 수행하십시오.**

-

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch
'{"spec":{"storage":{"emptyDir":{}}}}'
```

주의

프로덕션 환경 외 클러스터에 대해서만 이 옵션을 구성하십시오.

Image Registry Operator가 구성 요소를 초기화하기 전에 이 명령을 실행하면 oc patch 명령이 실패하며 다음 오류가 발생합니다.

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster"
not found
```

몇 분 후에 명령을 다시 실행하십시오.

14.3.16. 사용자 프로비저닝 인프라에 설치 완료

Operator 구성을 완료한 후 제공하는 인프라에 클러스터 설치를 완료할 수 있습니다.

사전 요구 사항

- 컨트롤 플레인이 초기화되어 있습니다.
- 초기 Operator 구성을 완료해야 합니다.

프로세스

1. 다음 명령을 사용하여 모든 클러스터 구성 요소가 온라인 상태인지 확인합니다.

```
$ watch -n5 oc get clusteroperators
```

출력 예

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.9.0     True        False         False      19m
baremetal                                  4.9.0     True        False         False      37m
cloud-credential                           4.9.0     True        False         False      40m
cluster-autoscaler                         4.9.0     True        False         False      37m
config-operator                            4.9.0     True        False         False      38m
console                                    4.9.0     True        False         False      26m
csi-snapshot-controller                    4.9.0     True        False         False      37m
dns                                        4.9.0     True        False         False      37m
etcd                                       4.9.0     True        False         False      36m
image-registry                             4.9.0     True        False         False      31m
ingress                                    4.9.0     True        False         False      30m
insights                                   4.9.0     True        False         False      31m
kube-apiserver                             4.9.0     True        False         False      26m
kube-controller-manager                    4.9.0     True        False         False      36m
kube-scheduler                             4.9.0     True        False         False      36m
kube-storage-version-migrator              4.9.0     True        False         False      37m
machine-api                                4.9.0     True        False         False      29m
machine-approver                           4.9.0     True        False         False      37m
machine-config                             4.9.0     True        False         False      36m
marketplace                                4.9.0     True        False         False      37m
monitoring                                 4.9.0     True        False         False      29m
network                                    4.9.0     True        False         False      38m
node-tuning                                4.9.0     True        False         False      37m
openshift-apiserver                        4.9.0     True        False         False      32m
openshift-controller-manager               4.9.0     True        False         False      30m
openshift-samples                          4.9.0     True        False         False      32m
operator-lifecycle-manager                 4.9.0     True        False         False      37m
operator-lifecycle-manager-catalog         4.9.0     True        False         False      37m
operator-lifecycle-manager-packageserver   4.9.0     True        False         False      32m
service-ca                                 4.9.0     True        False         False      38m
storage                                    4.9.0     True        False         False      37m
```

또는 다음 명령은 모든 클러스터를 사용할 수 있을 때 알립니다. 또한 인증 정보를 검색하고 표시합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1

<installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

출력 예

-

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator가 Kubernetes API 서버에서 OpenShift Container Platform 클러스터 배포를 완료하면 명령이 성공합니다.

중요

- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 node-bootstrapper 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

2. Kubernetes API 서버가 Pod와 통신하고 있는지 확인합니다.

   a. 모든 Pod 목록을 보려면 다음 명령을 사용하십시오.

```
$ oc get pods --all-namespaces
```

출력 예

```
NAMESPACE                  NAME                                        READY   STATUS
RESTARTS   AGE
openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8
1/1    Running    1        9m
openshift-apiserver        apiserver-67b9g                             1/1     Running    0
3m
openshift-apiserver        apiserver-ljcmx                             1/1     Running    0
1m
openshift-apiserver        apiserver-z25h4                             1/1     Running    0
2m
```

```
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8
1/1     Running     0          5m
...
```

b. 다음 명령을 사용하여 이전 명령의 출력에 나열된 **Pod**의 로그를 표시합니다.

```
$ oc logs <pod_name> -n <namespace> 1
```

1 이전 명령의 출력에 표시된 대로 **Pod** 이름과 네임스페이스를 지정합니다.

Pod 로그가 표시되면 **Kubernetes API** 서버는 클러스터 시스템과 통신할 수 있습니다.

3. 다중 경로를 활성화하기 위해 추가 단계가 필요합니다. 설치 중에 멀티패스를 활성화하지 마십시오.

자세한 내용은 *설치 후 머신 구성 작업* 설명서에서 **"RHCOS에서 커널 인수를 사용하여 멀티패스 활성화"**를 참조하십시오.

4. 클러스터 등록 페이지에서 클러스터를 등록합니다.

14.3.17. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 **Telemetry** 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 **Telemetry**가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 **OpenShift Cluster Manager**를 사용하여 자동으로 또는 **OpenShift Cluster Manager**를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 **OpenShift Container Platform** 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

14.3.18. 다음 단계

- RHCOS에서 커널 인수를 사용하여 멀티패스 활성화

- 클러스터를 사용자 지정합니다.

- 클러스터를 설치하는 데 사용한 미러 레지스트리에 신뢰할 수 있는 CA가 있는 경우 추가 신뢰 저장소를 구성하여 클러스터에 추가합니다.

15장. OPENSTACK에 설치

15.1. OPENSTACK에 설치 준비

RHOSP(Red Hat OpenStack Platform)에 OpenShift Container Platform을 설치할 수 있습니다.

15.1.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.

15.1.2. OpenStack에 OpenShift Container Platform을 설치할 방법 선택

설치 관리자 프로비저닝 또는 사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치할 수 있습니다. 기본 설치 유형은 설치 프로그램이 클러스터의 기본 인프라를 프로비저닝하는 설치 관리자 프로비저닝 인프라를 사용합니다. 제공하는 인프라에 OpenShift Container Platform도 설치할 수 있습니다. 설치 프로그램에서 프로비저닝한 인프라를 사용하지 않는 경우 클러스터 리소스를 직접 관리하고 유지보수해야 합니다.

설치 관리자 프로비저닝 및 사용자 프로비저닝 설치 프로세스에 대한 자세한 내용은 설치 프로세스를 참조하십시오.

15.1.2.1. 설치 관리자 프로비저닝 인프라를 사용하여 클러스터 설치

다음 방법 중 하나를 사용하여 OpenShift Container Platform 설치 프로그램에서 프로비저닝하는 RHOSP(Red Hat OpenStack Platform) 인프라에 클러스터를 설치할 수 있습니다.

- 사용자 지정으로 OpenStack에 클러스터 설치: RHOSP에 사용자 지정 클러스터를 설치할 수 있습니다. 설치 프로그램을 통해 설치 단계에서 일부 사용자 지정을 적용할 수 있습니다. 다른 많은 사용자 정의 옵션은 설치 후사용할 수 있습니다.
- Kuryr를 사용하여 OpenStack에 클러스터 설치: Kuryr SDN을 사용하는 RHOSP(Red Hat OpenStack Platform)에 사용자 지정 클러스터를 설치할 수 있습니다. Kuryr와 OpenShift Container Platform 통합은 주로 RHOSP VM에서 실행되는 OpenShift Container Platform 클러스터를 위해 설계되었습니다. Kuryr는 OpenShift Container Platform Pod를 RHOSP SDN에 연결하여 네트워크 성능을 향상시킵니다. 또한 Pod와 RHOSP 가상 인스턴스 간의 상호 연결성을 제공합니다.

- 제한된 네트워크의 OpenStack에 클러스터 설치: 설치 릴리스 컨텐츠의 내부 미러를 만들어 RHOSP에 OpenShift Container Platform을 제한되거나 연결이 끊긴 네트워크에 설치할 수 있습니다. 이 방법을 사용하여 소프트웨어 구성 요소를 받기 위해 활성 인터넷 연결이 필요하지 않은 클러스터를 설치할 수 있습니다. 이 설치 방법을 사용하여 클러스터가 외부 콘텐츠에 대해 조직의 제어 조건을 충족하는 컨테이너 이미지만 사용하도록 할 수 있습니다.

15.1.2.2. 사용자 프로비저닝 인프라를 사용하여 클러스터 설치

다음 방법 중 하나를 사용하여 프로비저닝하는 RHOSP 인프라에 클러스터를 설치할 수 있습니다.

- 자체 인프라에서 OpenStack에 클러스터 설치: 사용자가 프로비저닝한 RHOSP 인프라에 OpenShift 컨테이너 플랫폼을 설치할 수 있습니다. 이 설치 방법을 사용하면 클러스터를 기존 인프라 및 수정 사항과 통합할 수 있습니다. 사용자 프로비저닝 인프라에 설치하려면 Nova 서버, Neutron 포트, 보안 그룹과 같은 모든 RHOSP 리소스를 생성해야 합니다. 제공된 Ansible 플레이북을 사용하여 배포 프로세스를 지원할 수 있습니다.

- 자체 인프라에서 OpenStack에 Kuryr와 함께 클러스터 설치: Kuryr SDN을 사용하는 사용자 프로비저닝 RHOSP 인프라에 OpenShift 컨테이너 플랫폼을 설치할 수 있습니다.

- 자체 SR-IOV 인프라에서 OpenStack에 클러스터 설치: 컴퓨팅 머신을 실행하기 위해 SR-IOV(Single-root input/output virtualization) 네트워크를 사용하는 사용자 프로비저닝 RHOSP 인프라에 OpenShift Container Platform을 설치할 수 있습니다.

15.2. 사용자 지정으로 OPENSTACK에 클러스터 설치

OpenShift Container Platform 버전 4.9에서는 RHOSP(Red Hat OpenStack Platform)에 사용자 지정 클러스터를 설치할 수 있습니다. 설치를 사용자 지정하려면 클러스터를 설치하기 전에 install-config.yaml에서 매개변수를 수정합니다.

15.2.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.

- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.

- OpenShift Container Platform 4.9가 OpenShift 클러스터에 지원되는 플랫폼 섹션을 사용하여 현재 RHOSP 버전과 호환되는지 확인했습니다. RHOSP 지원 매트릭스의 OpenShift

Container Platform을 확인하여 여러 버전의 플랫폼 지원을 비교할 수도 있습니다.

- RHOSP에 블록 스토리지(Cinder) 또는 개체 스토리지(Swift)와 같은 스토리지 서비스가 설치되어 있어야 합니다. 개체 스토리지는 OpenShift Container Platform 레지스트리 클러스터 배포에 권장되는 스토리지 기술입니다. 자세한 정보는 스토리지 최적화를 참조하십시오.

- RHOSP에서 메타데이터 서비스 활성화

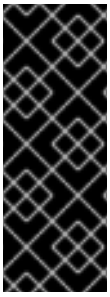
15.2.2. RHOSP에 OpenShift Container Platform을 설치하기 위한 리소스 지침

OpenShift Container Platform 설치를 지원하려면 RHOSP(Red Hat OpenStack Platform) 할당량이 다음 요구사항을 충족해야 합니다.

표 15.1. RHOSP의 기본 OpenShift Container Platform 클러스터에 권장되는 리소스

리소스 이름	값
부동 IP 주소	3
포트	15
라우터	1
서브넷	1
RAM	ECDHEGB
vCPU	22
볼륨 스토리지	275GB
인스턴스	7
보안 그룹	3
보안 그룹 규칙	60

권장 리소스보다 적은 리소스로도 클러스터가 작동할 수 있지만 성능은 보장되지 않습니다.



중요

swiftoperator 역할을 가진 사용자 계정으로 **RHOSP** 개체 스토리지(**Swift**)를 사용하고 운영하는 경우 **OpenShift Container Platform** 이미지 레지스트리의 기본 백엔드로 사용됩니다. 이 경우 볼륨 스토리지 요구사항은 **175GB**입니다. **Swift** 공간 요구사항은 이미지 레지스트리의 크기에 따라 다릅니다.

참고

기본적으로 보안 그룹 및 보안 그룹 규칙 할당량이 적을 수 있습니다. 문제가 발생하면 관리자로 **openstack quota set --secgroups 3 --secgroup-rules 60 <project>**를 실행하여 할당량을 늘립니다.

OpenShift Container Platform 배포는 컨트롤 플레인 시스템, 컴퓨팅 시스템, 부트스트랩 시스템으로 구성됩니다.

15.2.2.1. 컨트롤 플레인 머신

기본적으로 **OpenShift Container Platform** 설치 프로세스는 세 개의 컨트롤 플레인 시스템을 생성합니다.

각 시스템의 요구사항은 다음과 같습니다.

- **RHOSP** 할당량의 인스턴스
- **RHOSP** 할당량의 포트
- 최소 **16GB** 메모리 및 **vCPU 4**개가 있는 플레이버
- **RHOSP** 할당량에서 최소 **100GB** 스토리지 공간

15.2.2.2. 컴퓨팅 머신

기본적으로 **OpenShift Container Platform** 설치 프로세스는 세 개의 컴퓨팅 시스템을 생성합니다.

각 시스템의 요구사항은 다음과 같습니다.

- RHOSP 할당량의 인스턴스
- RHOSP 할당량의 포트
- 8GB 메모리 및 vCPU 2개 이상 있는 플레이버
- RHOSP 할당량에서 최소 100GB 스토리지 공간

작은 정보

컴퓨팅 시스템은 OpenShift Container Platform에서 실행하는 애플리케이션을 호스팅합니다. 최대한 많이 실행하는 것이 좋습니다.

15.2.2.3. 부트스트랩 시스템

설치하는 동안 컨트롤 플레인을 유지하기 위해 부트스트랩 시스템이 임시로 프로비저닝됩니다. 프로덕션 컨트롤 플레인이 준비되면 부트스트랩 시스템이 프로비저닝 해제됩니다.

부트스트랩 시스템의 요구사항은 다음과 같습니다.

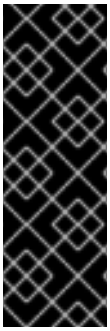
- RHOSP 할당량의 인스턴스
- RHOSP 할당량의 포트
- 최소 16GB 메모리 및 vCPU 4개가 있는 플레이버
- RHOSP 할당량에서 최소 100GB 스토리지 공간

15.2.3. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.



중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

15.2.4. RHOSP에서 Swift 활성화

Swift는 swiftoperator 역할을 가진 사용자 계정으로 운영됩니다. 설치 프로그램을 실행하려면 먼저 계정에 이 역할을 추가합니다.

중요

RHOSP(Red Hat OpenStack Platform) 개체 스토리지 서비스(일반적으로 Swift로 알려짐)를 사용할 수 있는 경우 OpenShift Container Platform이 이미지 레지스트리 스토리지로 사용합니다. 이 서비스를 사용할 수 없는 경우에는 설치 프로그램이 RHOSP 블록 스토리지 서비스(일반적으로 Cinder로 알려짐)를 사용합니다.

Swift가 있고 Swift를 사용하려면 액세스를 활성화해야 합니다. 존재하지 않거나 사용하지 않으려면 이 섹션을 건너 뛰십시오.

사전 요구 사항

- 대상 환경에 RHOSP 관리자 계정이 있습니다.
- Swift 서비스가 설치되어 있습니다.
- Ceph RGW에서 url의 계정 옵션이 활성화되어 있습니다.

프로세스

RHOSP에서 Swift를 활성화하려면:

1. RHOSP CLI의 관리자로서 Swift에 액세스할 계정에 swiftoperator 역할을 추가하십시오.

```
$ openstack role add --user <user> --project <project> swiftoperator
```

이제 RHOSP 배포에서 이미지 레지스트리에 Swift를 사용할 수 있습니다.

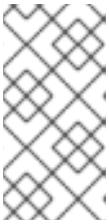
15.2.5. RHOSP에서 실행되는 클러스터에서 사용자 지정 스토리지를 사용하여 이미지 레지스트리 구성

RHOSP(Red Hat OpenStack Platform)에 클러스터를 설치한 후 레지스트리 스토리지의 특정 가용성 영역에 있는 Cinder 볼륨을 사용할 수 있습니다.

프로세스

1. 사용할 스토리지 클래스 및 가용성 영역을 지정하는 YAML 파일을 생성합니다. 예를 들면 다음과 같습니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: custom-csi-storageclass
provisioner: cinder.csi.openstack.org
volumeBindingMode: WaitForFirstConsumer
allowVolumeExpansion: true
parameters:
  availability: <availability_zone_name>
```



참고

OpenShift Container Platform은 선택한 가용성 영역이 있는지 확인하지 않습니다. 구성을 적용하기 전에 가용성 영역의 이름을 확인합니다.

2. 명령줄에서 구성을 적용합니다.

```
$ oc apply -f <storage_class_file_name>
```

출력 예

```
storageclass.storage.k8s.io/custom-csi-storageclass created
```

3. 스토리지 클래스와 openshift-image-registry 네임스페이스를 사용하는 PVC(영구 볼륨 클레임)를 지정하는 YAML 파일을 생성합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: csi-pvc-imageregistry
  namespace: openshift-image-registry (1)
  annotations:
    imageregistry.openshift.io: "true"
spec:
  accessModes:
  - ReadWriteOnce
  volumeMode: Filesystem
  resources:
    requests:
      storage: 100Gi (2)
  storageClassName: <your_custom_storage_class> (3)
```

(1) 네임스페이스 openshift-image-registry 를 입력합니다. 이 네임스페이스를 사용하면 Cluster Image Registry Operator에서 PVC를 사용할 수 있습니다.

(2) 선택 사항: 볼륨 크기를 조정합니다.

3 생성한 스토리지 클래스의 이름을 입력합니다.

4. 명령줄에서 구성을 적용합니다.

```
$ oc apply -f <pvc_file_name>
```

출력 예

```
persistentvolumeclaim/csi-pvc-imageregistry created
```

5. 이미지 레지스트리 구성의 원래 영구 볼륨 클레임을 새 클레임으로 교체합니다.

```
$ oc patch configs.imageregistry.operator.openshift.io/cluster --type 'json' -p='[{"op":
"replace", "path": "/spec/storage/pvc/claim", "value": "csi-pvc-imageregistry"}]'
```

출력 예

```
config.imageregistry.operator.openshift.io/cluster patched
```

다음 몇 분 동안 구성이 업데이트됩니다.

검증

레지스트리에서 사용자가 정의한 리소스를 사용하고 있는지 확인하려면 다음을 수행합니다.

1. PVC 클레임 값이 PVC 정의에 제공한 이름과 동일한지 확인합니다.

```
$ oc get configs.imageregistry.operator.openshift.io/cluster -o yaml
```

-

출력 예

```
...
status:
  ...
  managementState: Managed
  pvc:
    claim: csi-pvc-imageregistry
...
```

2. PVC의 상태가 Bound 인지 확인합니다.

```
$ oc get pvc -n openshift-image-registry csi-pvc-imageregistry
```

출력 예

```
NAME                    STATUS   VOLUME                                     CAPACITY   ACCESS
MODES   STORAGECLASS          AGE
csi-pvc-imageregistry   Bound    pvc-72a8f9c9-f462-11e8-b6b6-fa163e18b7b5   100Gi
RWO           custom-csi-storageclass   11m
```

15.2.6. 외부 네트워크 액세스 확인

OpenShift Container Platform 설치 프로세스에는 외부 네트워크에 액세스해야 합니다. 외부 네트워크값을 제공해야 하며 그렇지 않으면 배포가 실패합니다. 프로세스를 시작하기 전에 외부 라우터 유형의 네트워크가 RHOSP(Red Hat OpenStack Platform)에 있는지 확인합니다.

사전 요구 사항

- DHCP 에이전트가 인스턴스의 DNS 쿼리를 전달하도록 OpenStack의 네트워킹 서비스 구성

프로세스

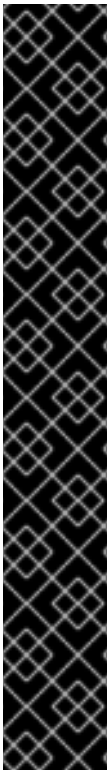
1. RHOSP CLI를 사용하여 '외부' 네트워크의 이름과 ID를 확인합니다.

```
$ openstack network list --long -c ID -c Name -c "Router Type"
```

출력 예

```
+--------------------------------------+--------------+-------------+
| ID                                   | Name         | Router Type |
+--------------------------------------+--------------+-------------+
| 148a8023-62a7-4672-b018-003462f8d7dc | public_network | External    |
+--------------------------------------+--------------+-------------+
```

외부 라우터 유형의 네트워크가 네트워크 목록에 나타납니다. 네트워크가 하나 이상 나타나지 않으면 기본 부동 IP 네트워크 생성 및 기본 공급자 네트워크 생성을 참조하십시오.



중요

외부 네트워크의 CIDR 범위가 기본 네트워크 범위 중 하나와 겹치는 경우 설치 프로세스를 시작하기 전에 install-config.yaml 파일에서 일치하는 네트워크 범위를 변경해야 합니다.

기본 네트워크 범위는 다음과 같습니다.

네트워크	범위
machineNetwork	10.0.0.0/16
serviceNetwork	172.30.0.0/16
clusterNetwork	10.128.0.0/14

주의

설치 프로그램이 이름이 같은 여러 네트워크를 발견하면 그 중 하나를 무작위로 설정합니다. 이 동작을 방지하려면 **RHOSP**에서 리소스의 고유한 이름을 만듭니다.

참고

Neutron 트렁크 서비스 플러그인이 활성화된 경우 기본적으로 트렁크 포트가 생성됩니다. 자세한 내용은 **Neutron** 트렁크 포트를 참조하십시오.

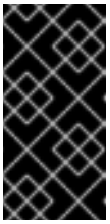
15.2.7. 설치 프로그램에 대한 매개변수 정의

OpenShift Container Platform 설치 프로그램은 **clouds.yaml** 파일을 사용합니다. 이 파일은 프로젝트 이름, 로그인 정보, 인증 서비스 **URL** 등 **RHOSP(Red Hat OpenStack Platform)** 구성 매개변수를 설명합니다.

프로세스

1. **clouds.yaml** 파일을 만듭니다.

- **RHOSP** 배포에 **Horizon** 웹 **UI**가 포함되어 있으면 그 안에 **clouds.yaml** 파일을 만듭니다.



중요

auth 필드에 암호를 추가해야 합니다. **clouds.yaml**과 별도의 파일에 비밀을 저장할 수도 있습니다.

- **RHOSP** 배포에 **Horizon** 웹 **UI**가 포함되어 있지 않거나 **Horizon**을 사용하지 않으려면 파일을 직접 만듭니다. **clouds.yaml**에 대한 자세한 내용은 **RHOSP** 문서의 구성 파일을 참조하십시오.

```
clouds:
  shiftstack:
    auth:
      auth_url: http://10.10.14.42:5000/v3
```

```
    project_name: shiftstack
    username: shiftstack_user
    password: XXX
    user_domain_name: Default
    project_domain_name: Default
  dev-env:
   region_name: RegionOne
   auth:
    username: 'devuser'
    password: XXX
    project_name: 'devonly'
    auth_url: 'https://10.10.14.22:5001/v2.0'
```

2. RHOSP 설치에서 끝점 인증을 위해 자체 서명된 CA(인증 기관) 인증서를 사용하는 경우:

   a. 인증 기관 파일을 시스템에 복사합니다.

   b. cacerts 키를 clouds.yaml 파일에 추가합니다. 값은 CA 인증서에 대한 루트가 아닌 액세스 가능한 절대 경로여야 합니다.

```
clouds:
  shiftstack:
    ...
    cacert: "/etc/pki/ca-trust/source/anchors/ca.crt.pem"
```

작은 정보

사용자 지정 CA 인증서로 설치 관리자를 실행한 후 cloud-provider-config 키맵에서 ca-cert.pem 키의 값을 편집하여 인증서를 업데이트할 수 있습니다. 명령줄에서 다음을 실행합니다.

```
$ oc edit configmap -n openshift-config cloud-provider-config
```

3. clouds.yaml 파일을 다음 위치 중 하나에 배치합니다.

   a. OS_CLIENT_CONFIG_FILE 환경 변수의 값

   b. 현재 디렉터리

c. Unix 전용 사용자 구성 디렉터리(예: ~/.config/openstack/clouds.yaml)

d. Unix 전용 사이트 구성 디렉터리(예: /etc/openstack/clouds.yaml)

설치 프로그램은 clouds.yaml을 이 순서대로 검색합니다.

15.2.8. 클라우드 공급자 옵션 설정

선택적으로 클러스터의 클라우드 공급자 구성을 편집할 수 있습니다. 클라우드 공급자 구성은 OpenShift Container Platform이 RHOSP(Red Hat OpenStack Platform)와 상호 작용하는 방법을 제어합니다.

클라우드 공급자 구성 매개변수의 전체 목록은 "OpenStack 클라우드 구성 참조 가이드" 페이지의 "OpenStack 클라우드 구성 참조 가이드" 페이지를 참조하십시오.

프로세스

1. 클러스터에 대해 이미 생성된 매니페스트 파일이 없는 경우 다음 명령을 실행하여 생성합니다.

```
$ openshift-install --dir <destination_directory> create manifests
```

2. 텍스트 편집기에서 cloud-provider 구성 매니페스트 파일을 엽니다. 예를 들면 다음과 같습니다.

```
$ vi openshift/manifests/cloud-provider-config.yaml
```

3. 클라우드 구성 사양에 따라 옵션을 수정합니다.

로드 밸런싱을 위해 Octavia를 구성하는 것은 Kuryr를 사용하지 않는 클러스터의 일반적인 사례입니다. 예를 들면 다음과 같습니다.

```
#...
[LoadBalancer]
use-octavia=true (1)
lb-provider = "amphora" (2)
floating-network-id="d3deb660-4190-40a3-91f1-37326fe6ec4a"(3)
#...
```

1

이 속성은 **Octavia** 통합을 활성화합니다.

2

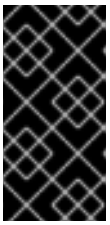
이 속성은 로드 밸런서에서 사용하는 **Octavia** 공급자를 설정합니다. **"ovn"** 또는 **"amphora"** 를 값으로 허용합니다. **OVN**을 사용하도록 선택하는 경우 **lb-method** 도 **SOURCE_IP_PORT** 로 설정해야 합니다.

3

이 속성은 클러스터에서 여러 외부 네트워크를 사용하려는 경우 필요합니다. 클라우드 프로바이더는 여기에 지정된 네트워크에 유동 **IP** 주소를 생성합니다.

중요

변경 사항을 저장하기 전에 파일이 올바르게 구성되었는지 확인합니다. 속성을 적절한 섹션에 배치하지 않으면 클러스터가 실패할 수 있습니다.



중요

Kuryr를 사용하는 설치의 경우 **Kuryr**는 관련 서비스를 처리합니다. 클라우드 공급자에서 **Octavia** 로드 밸런싱을 구성할 필요가 없습니다.

4. 변경 사항을 파일에 저장하고 설치를 진행합니다.

작은 정보

설치 프로그램을 실행한 후 클라우드 공급자 구성을 업데이트할 수 있습니다. 명령줄에서 다음을 실행합니다.

```
$ oc edit configmap -n openshift-config cloud-provider-config
```

변경 사항을 저장한 후 클러스터를 재구성하는 데 다소 시간이 걸립니다. 노드가 **SchedulingDisabled** 상태가 없는 경우 프로세스가 완료됩니다.

추가 리소스

-

클라우드 공급자 구성에 대한 자세한 내용은 OpenStack 클라우드 공급자 옵션을 참조하십시오.

15.2.9. 설치 프로그램 받기

OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

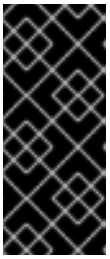
- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

프로세스

1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.

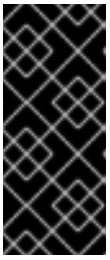
2. 인프라 공급자를 선택합니다.

3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.



중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.



중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 OpenShift Container Platform 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

15.2.10. 설치 구성 파일 만들기

RHOSP(Red Hat OpenStack Platform)에 설치하는 OpenShift Container Platform 클러스터를 사용자 지정할 수 있습니다.

사전 요구 사항

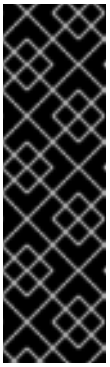
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.
- 서브스크립션 수준에서 서비스 주체 권한을 획득합니다.

프로세스

1. install-config.yaml 파일을 생성합니다.
   a. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 <installation_directory>는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.



중요

비어 있는 디렉터리를 지정합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

.

b. 화면에 나타나는 지시에 따라 클라우드에 대한 구성 세부 사항을 입력합니다.

i. 선택사항: 클러스터 시스템에 액세스하는 데 사용할 **SSH** 키를 선택합니다.

참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 **OpenShift Container Platform** 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 **SSH** 키를 지정합니다.

ii. 대상 플랫폼으로 **openstack**을 선택합니다.

iii. 클러스터 설치에 사용할 **RHOSP(Red Hat OpenStack Platform)** 외부 네트워크 이름을 지정합니다.

iv. **OpenShift API**에 대한 외부 액세스에 사용할 부동 **IP** 주소를 지정합니다.

v. 컨트롤 플레인 노드에 사용할 최소 **16GB**의 **RAM**과 컴퓨팅 노드에 **8GB RAM**이 있는 **RHOSP** 플레이버를 지정합니다.

vi. 클러스터를 배포할 기본 도메인을 선택합니다. 모든 **DNS** 레코드는 이 기본 도메인의 하위 도메인이 되므로 클러스터 이름을 포함합니다.

vii. 클러스터 이름을 입력합니다. 이름은 **14**자 이하여야 합니다.

viii. Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 붙여넣습니다.

2. **install-config.yaml** 파일을 수정합니다. 사용 가능한 매개변수에 대한 자세한 정보는 “설치 구성 매개변수” 섹션에서 확인할 수 있습니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.

중요

install-config.yaml 파일은 설치 과정에서 사용됩니다. 이 파일을 재사용하려면 지금 백업해야 합니다.

추가 리소스

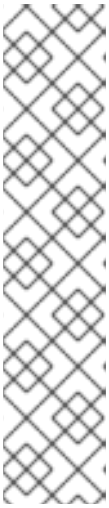
사용 가능한 매개변수에 대한 자세한 내용은 설치 구성 매개 변수 섹션 을 참조하십시오.

15.2.10.1. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 **HTTP** 또는 **HTTPS** 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 **OpenShift Container Platform** 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 **API**에 대한 호출을 포함하여 모든 클러스터 발신 **(Egress)** 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.



참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr, networking.clusterNetwork[].cidr, networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 **Red Hat OpenStack Platform (RHOSP)**에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점**(169.254.169.254)**로 채워집니다.

프로세스

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1

클러스터 외부에서 **HTTP** 연결을 구축하는 데 사용할 프록시 **URL**입니다. **URL** 스키마는 **http**여야 합니다.

2

클러스터 외부에서 **HTTPS** 연결을 구축하는 데 사용할 프록시 **URL**입니다.

3

대상 도메인 이름, **IP** 주소 또는 프록시에서 제외할 기타 네트워크 **CIDR**로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. ***를 사용하여 모든 대상에 대해 프록시를 바이패스합니다.

4

이 값을 제공하면 설치 프로그램에서 추가 **CA** 인증서를 보유할 **openshift -config** 네임스페이스에 **user-ca- bundle** 이라는 구성 맵을 생성합니다. **additionalTrustBundle** 및 하나 이상의 프록시 설정을 제공하는 경우 프록시 오브젝트는 **trustedCA** 필드의 **user-ca-bundle** 구성 맵을 참조하도록 구성됩니다. 그러면 **Cluster Network Operator**에서 **trustedCA** 매개변수에 대해 지정된 콘텐츠를 **RHCOS** 신뢰 번들과 병합하는 **trusted-ca-bundle** 구성 맵을 생성합니다. 프록시의 **ID** 인증서를 **RHCOS** 트러스트 번들에 있는 기관에서 서명하지 않은 경우 **additionalTrustBundle** 필드가 있어야 합니다.

참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 **OpenShift Container Platform**을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프

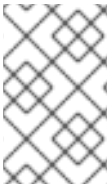
록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 **nil**이 됩니다.

참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

15.2.11. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 매개변수 값을 제공하여 클러스터를 호스팅할 클라우드 플랫폼에서 사용자 계정을 설명하고 선택사항으로 클러스터의 플랫폼을 사용자 지정합니다. **install-config.yaml** 설치 구성 파일을 생성할 때 명령줄을 통해 필요한 매개변수 값을 제공합니다. 클러스터를 사용자 지정하면 **install-config.yaml** 파일을 수정하여 플랫폼에 대한 세부 정보를 제공할 수 있습니다.



참고

설치한 후에는 **install-config.yaml** 파일에서 이러한 매개변수를 수정할 수 없습니다.

중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

15.2.11.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 **15.2.** 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열

매개변수	설명	값
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	소문자, 하이픈(**-**), 마침표(**.**)로 구성되는 문자열(예: **dev**)입니다. 문자열은 14자 이하여야 합니다.
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

15.2.11.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 15.3. 네트워크 매개변수

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체  **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.

networking.clusterNetwork	Pod의 IP 주소 블록입니다.
	기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다.
	여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.
		개체의 배열입니다. 예를 들면 다음과 같습니다.
		`networking:`
		`  clusterNetwork:`
		`  - cidr: 10.128.0.0/14`
		`    hostPrefix: 23`

networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된 **cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.

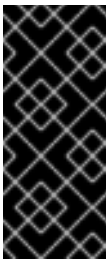
매개변수	설명	값
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16** **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

15.2.11.3. 선택적 구성 매개변수

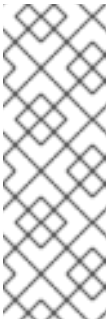
선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 15.4. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.

매개변수	설명	값
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은**3**입니다.
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열

매개변수	설명	값
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 기본값은 **External**입니다. 이 필드를 **Internal** 로 설정하는 것은 클라우드 이외의 플랫폼에서는 지원되지 않습니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

15.2.11.4. 추가 RHOSP(Red Hat OpenStack Platform) 구성 매개변수

추가 RHOSP 구성 매개변수는 다음 표에 설명되어 있습니다.

표 15.5. 추가 RHOSP 매개변수

매개변수	설명	값
compute.platform.openstack.rootVolume.size	컴퓨팅 시스템의 경우 루트 볼륨의 크기(GB)입니다. 이 값을 설정하지 않으면 시스템은 임시 스토리지를 사용합니다.	정수 (예: **30)**
compute.platform.openstack.rootVolume.type	컴퓨팅 시스템의 경우 루트 볼륨의 유형입니다.	문자열 (예: **performance)**
controlPlane.platform.openstack.rootVolume.size	컨트롤 플레인 시스템의 경우 루트 볼륨의 크기(GB)입니다. 이 값을 설정하지 않으면 시스템은 임시 스토리지를 사용합니다.	정수 (예: **30)**
controlPlane.platform.openstack.rootVolume.type	컨트롤 플레인 시스템의 경우 루트 볼륨의 유형입니다.	문자열 (예: **performance)**
platform.openstack.cloud	**clouds.yaml** 파일의 클라우드 목록에서 사용할 RHOSP 클라우드의 이름입니다.	문자열 (예: **MyCloud)**

매개변수	설명	값
platform.openstack.externalNetwork	설치에 사용할 RHOSP 외부 네트워크 이름입니다.	문자열 (예: **external)**
platform.openstack.computeFlavor	컨트롤 플레인 및 컴퓨팅 시스템에 사용할 RHOSP 버전입니다. 이 속성은 더 이상 사용되지 않습니다. 모든 머신 풀에 플레이버를 기본값으로 사용하려면 **platform.openstack.defaultMachinePlatform** 속성의 **type** 키 값으로 추가합니다. 각 머신 풀의 플레이버 값을 개별적으로 설정할 수도 있습니다.	문자열 (예: **m1.xlarge)**

15.2.11.5. 선택적 RHOSP 구성 매개변수

선택적 RHOSP 구성 매개변수는 다음 표에 설명되어 있습니다.

표 15.6. 선택적 RHOSP 매개변수

매개변수	설명	값
compute.platform.openstack.additionalNetworkIDs	컴퓨팅 시스템과 관련된 추가 네트워크입니다. 추가 네트워크에는 허용되는 주소 쌍이 생성되지 않습니다.	하나 이상의 UUID 목록을 문자열로 나타냅니다. 예: **fa806b2f-ac49-4bce-b9db-124bc64209bf**
compute.platform.openstack.additionalSecurityGroupIDs	컴퓨팅 시스템과 관련된 추가 보안 그룹입니다.	하나 이상의 UUID 목록을 문자열로 나타냅니다. 예: **7ee219f3-d2e9-48a1-96c2-e7429f1b0da7**

매개변수	설명	값
compute.platform.openstack.zones	머신을 설치할 RHOSP Compute (Nova) 가용성 영역 (AZ). 이 매개 변수가 설정되지 않은 경우 설치 프로그램은 RHOSP 관리자가 구성한 Nova의 기본 설정을 사용합니다. Kuryr를 사용하는 클러스터에서 RHOSP Octavia는 가용성 영역을 지원하지 않습니다. 로드 밸런서 및 Amphora 공급자 드라이버를 사용하는 경우 Amphora VM에 의존하는 OpenShift Container Platform 서비스는이 속성의 값에 따라 생성되지 않습니다.	문자열 목록입니다. 예: **["zone-1", "zone-2"]**.
compute.platform.openstack.rootVolume.zones	컴퓨팅 시스템의 경우 루트 볼륨을 설치할 가용성 영역입니다. 이 매개변수의 값을 설정하지 않으면 설치 프로그램에서 기본 가용성 영역을 선택합니다.	문자열 목록(예: **["zone-1", "zone-2"]**).
controlPlane.platform.openstack.additionalNetworkIDs	컨트롤 플레인 시스템과 관련된 추가 네트워크입니다. 추가 네트워크에는 허용되는 주소 쌍이 생성되지 않습니다.	하나 이상의 UUID 목록을 문자열로 나타냅니다. 예: **fa806b2f-ac49-4bce-b9db-124bc64209bf**
controlPlane.platform.openstack.additionalSecurityGroupIDs	컨트롤 플레인 시스템과 관련된 추가 보안 그룹입니다.	하나 이상의 UUID 목록을 문자열로 나타냅니다. 예: **7ee219f3-d2e9-48a1-96c2-e7429f1b0da7**
controlPlane.platform.openstack.zones	머신을 설치할 RHOSP Compute (Nova) 가용성 영역 (AZ). 이 매개 변수가 설정되지 않은 경우 설치 프로그램은 RHOSP 관리자가 구성한 Nova의 기본 설정을 사용합니다. Kuryr를 사용하는 클러스터에서 RHOSP Octavia는 가용성 영역을 지원하지 않습니다. 로드 밸런서 및 Amphora 공급자 드라이버를 사용하는 경우 Amphora VM에 의존하는 OpenShift Container Platform 서비스는이 속성의 값에 따라 생성되지 않습니다.	문자열 목록입니다. 예: **["zone-1", "zone-2"]**.

매개변수	설명	값
controlPlane.platform.openstack.rootVolume.zones	컨트롤 플레인 시스템의 경우 루트 볼륨을 설치할 가용성 영역입니다. 이 값을 설정하지 않으면 설치 프로그램에서 기본 가용성 영역을 선택합니다.	문자열 목록(예: **["zone-1", "zone-2"]**).
platform.openstack.clusterOSImage	설치 프로그램이 RHCOS 이미지를 다운로드하는 위치입니다. 네트워크가 제한된 환경에서 설치를 수행하려면이 매개 변수를 설정해야 합니다.	HTTP 또는 HTTPS URL (선택 옵션으로 SHA-256 형식의 체크섬을 사용) 예: **http://mirror.example.com/images/rhcos-43.81.201912131630.0-openstack.x86_64.qcow2.gz?sha256=ffebbd68e8a1f2a245ca19522c16c86f67f9ac8e4e0c1f0a812b068b16f7265d**. 이 값은 기존의 Glance 이미지의 이름이 될 수도 있습니다 (예: **my-rhcos)**.
platform.openstack.clusterOSImageProperties	Glance의 설치 프로그램에서 업로드된 ClusterOSImage에 추가할 속성입니다. **platform.openstack.clusterOSImage**가 기존 Glance 이미지로 설정된 경우 이 속성은 무시됩니다. 이 속성을 사용하여 노드당 26 PV의 RHOSP의 기본 PV(영구 볼륨) 제한을 초과할 수 있습니다. 제한을 초과하려면 **hw_scsi_model** 속성값을 **virtio-scsi**로, **hw_disk_bus** 값을 **scsi**로 설정합니다. 이 속성을 사용하면 **yes** 값이 있는 **hw_qemu_guest_agent** 속성을 포함하여 QEMU 게스트 에이전트를 활성화할 수도 있습니다.	키-값 문자열 쌍 목록입니다. 예를 들면 **["hw_scsi_model": "virtio-scsi", "hw_disk_bus": "scsi"]**가 있습니다.

platform.openstack.defaultMachinePlatform	기본 시스템 풀 플랫폼 구성입니다.	<pre>{
		  "type": "ml.large",
		  "rootVolume": {
		  "size": 30,
		  "type": "performance"
		  }
		}</pre>

매개변수	설명	값
platform.openstack.ingressFloatingIP	Ingress 포트와 연결할 기존 부동 IP 주소입니다. 이 속성을 사용하려면 **platform.openstack.externalNetwork** 속성도 정의해야 합니다.	IP 주소 (예: **128.0.0.1**)
platform.openstack.apiFloatingIP	로드 밸런서 API와 연결할 기존 부동 IP 주소입니다. 이 속성을 사용하려면 **platform.openstack.externalNetwork** 속성도 정의해야 합니다.	IP 주소 (예: **128.0.0.1**)
platform.openstack.externalDNS	클러스터 인스턴스가 DNS 확인에 사용하는 외부 DNS 서버의 IP 주소입니다.	IP 주소 목록을 문자열로 나타냅니다. 예: **["8.8.8.8", "192.168.1.12"]**.
platform.openstack.machinesSubnet	클러스터 노드가 사용하는 RHOSP 서브넷의 UUID입니다. 이 서브넷에 노드와 가상 IP(VIP) 포트가 생성됩니다. **networking.machineNetwork**의 첫 번째 항목은 **machinesSubnet** 값과 일치해야 합니다. 사용자 지정 서브넷에 배포하는 경우 OpenShift Container Platform 설치 관리자에 외부 DNS 서버를 지정할 수 없습니다. 대신 RHOSP의 서브넷에 DNS를 추가합니다.	문자열의 UUID입니다. 예: **fa806b2f-ac49-4bce-b9db-124bc64209bf**

15.2.11.6. RHOSP 배포의 사용자 지정 서브넷

선택한 **RHOSP(Red Hat OpenStack Platform)** 서브넷에 클러스터를 배포할 수도 있습니다. 서브넷의 **GUID**는 **install-config.yaml** 파일에서 **platform.openstack.machinesSubnet** 값으로 전달됩니다.

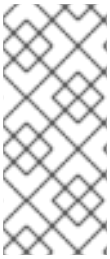
이 서브넷은 클러스터의 기본 서브넷으로 사용됩니다. 기본적으로 노드와 포트가 이 서브넷에서 생성됩니다. **platform.openstack.machinesSubnet** 속성의 값을 서브넷의 **UUID**로 설정하여 다른 **RHOSP** 서브넷에서 노드 및 포트를 생성할 수 있습니다.

사용자 지정 서브넷으로 **OpenShift Container Platform** 설치 관리자를 실행하기 전에 구성이 다음 요구 사항을 충족하는지 확인하십시오.

- platform.openstack.machinesSubnet에서 사용하는 서브넷에 DHCP가 활성화되어 있습니다.
- platform.openstack.machinesSubnet의 CIDR은 networking.machineNetwork의 CIDR과 일치합니다.
- 설치 프로그램 사용자에게 고정 IP 주소가 있는 포트를 포함하여 이 네트워크에 포트를 생성할 수 있는 권한이 있습니다.

사용자 지정 서브넷을 사용하는 클러스터에는 다음과 같은 제한 사항이 있습니다.

- 유동 IP 주소를 사용하는 클러스터를 설치하려는 경우 platform.openstack.machinesSubnet 서브넷이 externalNetwork 네트워크에 연결된 라우터에 연결되어 있어야 합니다.
- platform.openstack.machinesSubnet 값이 install-config.yaml 파일에 설정된 경우 설치 프로그램에서 RHOSP 머신의 프라이빗 네트워크 또는 서브넷을 생성하지 않습니다.
- 사용자 지정 서브넷과 동시에 platform.openstack.externalDNS 속성을 사용할 수 없습니다. 사용자 지정 서브넷을 사용하는 클러스터에 DNS를 추가하려면 RHOSP 네트워크에서 DNS를 구성합니다.



참고

기본적으로 API VIP는 x.x.x.5를 가져오고 Ingress VIP는 네트워크의 CIDR 블록에서 x.x.x.7을 가져옵니다. 이러한 기본값을 재정의하려면 DHCP 할당 풀 외부에 있는platform.openstack.apiVIP 및 platform.openstack.ingressVIP의 값을 설정합니다.

15.2.11.7. 베어 메탈 머신으로 클러스터 배포

클러스터가 베어 메탈 머신을 사용하도록 하려면 install-config.yaml 파일을 수정합니다. 클러스터는 베어 메탈에서 컨트롤 플레인 및 컴퓨팅 머신 모두를 실행하거나 컴퓨팅 머신만으로 실행할 수 있습니다.

Kuryr를 사용하는 클러스터에서 베어 메탈 컴퓨팅 머신이 지원되지 않습니다.

참고

install-config.yaml 파일에서 베어 메탈 작업자가 유동 IP 주소를 지원하는지 여부를 반영하는지 확인하십시오.

사전 요구 사항

- RHOSP Bare Metal 서비스(Ironic)가 활성화되어 RHOSP Compute API를 통해 액세스할 수 있습니다.
- 베어 메탈은 RHOSP 플레이버로 사용할 수 있습니다.
- RHOSP 네트워크는 VM 및 베어 메탈 서버 연결을 모두 지원합니다.
- 네트워크 구성이 공급자 네트워크를 사용하지 않습니다. 공급자 네트워크는 지원되지 않습니다.
- 기존 네트워크에 머신을 배포하려면 RHOSP 서브넷이 프로비저닝됩니다.
- 설치 관리자 프로비저닝 네트워크에 머신을 배포하려는 경우 RHOSP Bare Metal 서비스(Ironic)가 테넌트 네트워크에서 실행되는 PXE(Preboot eXecution Environment) 부팅 머신을 수신하고 상호 작용할 수 있습니다.
- OpenShift Container Platform 설치 프로세스의 일부로 install-config.yaml 파일을 생성하셨습니다.

절차

1. install-config.yaml 파일에서 머신의 플레이버를 편집합니다.
   a. 베어 메탈 컨트롤 플레인 머신을 사용하려면 controlPlane.platform.openstack.type 값을 베어 메탈 플레이버로 변경합니다.
   b. compute.platform.openstack.type 값을 베어 메탈 플레이버로 변경합니다.

c.

기존 네트워크에 머신을 배포하려면 **platform.openstack.machinesSubnet**의 값을 네트워크의 **RHOSP** 서브넷 **UUID**로 변경합니다. 컨트롤 플레인 및 컴퓨팅 머신은 동일한 서브넷을 사용해야 합니다.

베어 메탈의 **install-config.yaml** 예제 파일

```
controlPlane:
  platform:
    openstack:
      type: <bare_metal_control_plane_flavor> 1
...

compute:
 - architecture: amd64
   hyperthreading: Enabled
   name: worker
   platform:
     openstack:
       type: <bare_metal_compute_flavor> 2
   replicas: 3
...

platform:
  openstack:
    machinesSubnet: <subnet_UUID> 3
...
```

1

베어 메탈 컨트롤 플레인 머신이 필요한 경우 이 값을 베어 메탈 플레이버로 변경합니다.

2

컴퓨팅 머신에 사용할 베어 메탈 플레이버로 이 값을 변경합니다.

3

기존 네트워크를 사용하려는 경우 이 값을 **RHOSP** 서브넷의 **UUID**로 변경합니다.

업데이트된 **install-config.yaml** 파일을 사용하여 설치 프로세스를 완료합니다. 배포 중에 생성된 컴

퓨팅 머신은 파일에 추가한 플레이버를 사용합니다.

참고

설치 프로그램은 베어 메탈 머신이 부팅될 때까지 대기하는 동안 시간이 초과될 수 있습니다.

설치 프로그램이 시간 초과되면 설치 프로그램의 **wait-for** 명령을 사용하여 배포를 다시 시작한 다음 완료합니다. 예를 들면 다음과 같습니다.

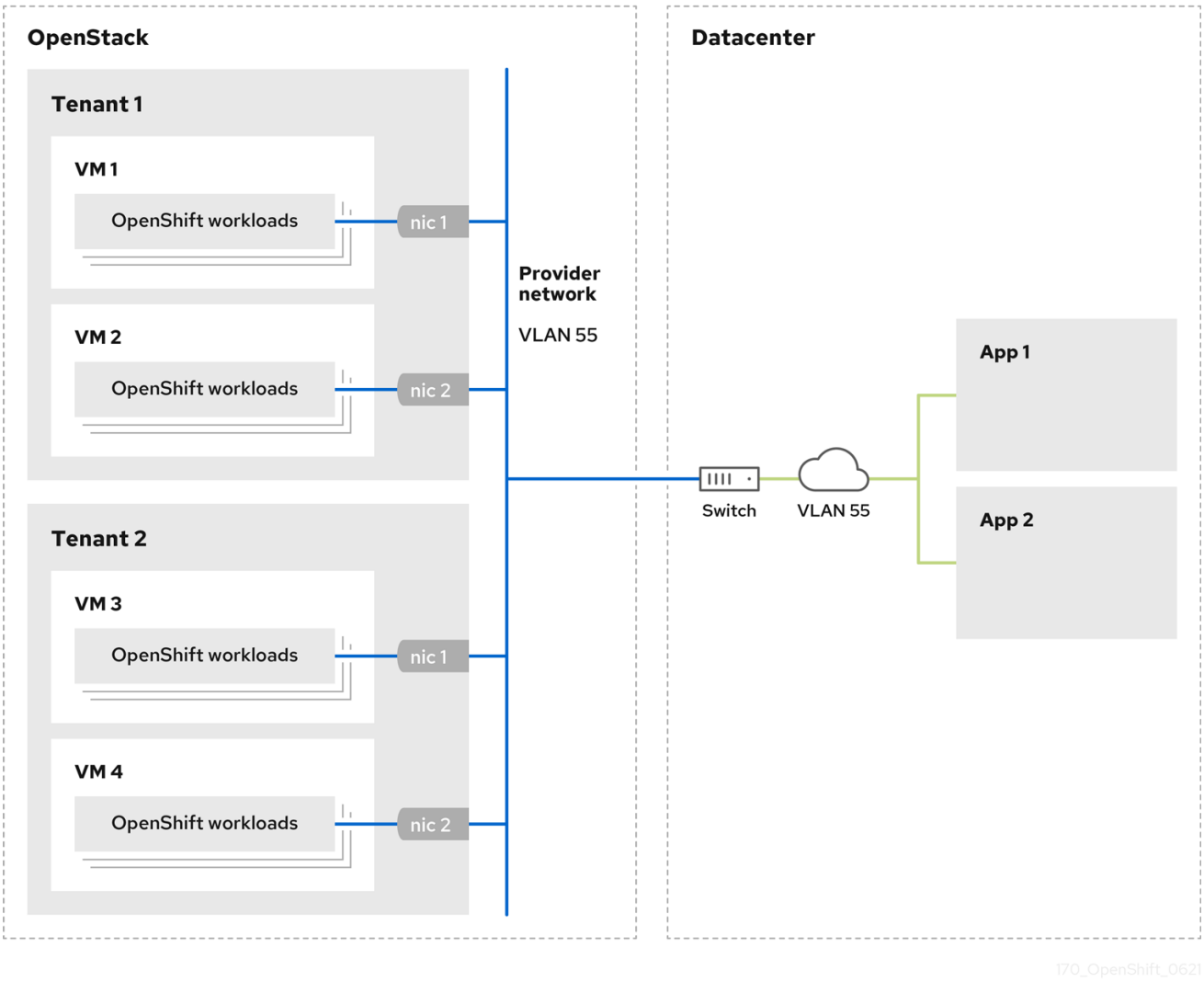
```
./openshift-install wait-for install-complete --log-level debug
```

15.2.11.8. RHOSP 공급자 네트워크에서 클러스터 배포

공급자 네트워크의 기본 네트워크 인터페이스를 사용하여 **RHOSP(Red Hat OpenStack Platform)**에 **OpenShift Container Platform** 클러스터를 배포할 수 있습니다. 공급자 네트워크는 일반적으로 프로젝트에 인터넷에 연결하는 데 사용할 수 있는 공용 네트워크에 직접 액세스할 수 있는 권한을 부여하는 데 사용됩니다. 네트워크 생성 프로세스의 일부로 프로젝트 간에 공급자 네트워크를 공유할 수도 있습니다.

RHOSP 공급자 네트워크는 데이터 센터의 기존 실제 네트워크에 직접 매핑됩니다. **RHOSP** 관리자가 이를 생성해야 합니다.

다음 예에서 **OpenShift Container Platform** 워크로드는 공급자 네트워크를 사용하여 데이터 센터에 연결됩니다.



공급자 네트워크에 설치된 **OpenShift Container Platform** 클러스터에는 테넌트 네트워크 또는 유동 **IP** 주소가 필요하지 않습니다. 설치 프로그램에서 설치하는 동안 이러한 리소스를 생성하지 않습니다.

공급자 네트워크 유형에는 **flat**(태그되지 않음) 및 **VLAN**(802.1Q 태그)이 포함됩니다.

참고

클러스터는 네트워크 유형에서 허용하는 만큼의 공급자 네트워크 연결을 지원할 수 있습니다. 예를 들어 **VLAN** 네트워크는 일반적으로 최대 **4096**개의 연결을 지원합니다.

RHOSP 설명서에서 공급자 및 테넌트 네트워크에 대해 자세히 알아볼 수 있습니다.

15.2.11.8.1. 클러스터 설치를 위한 RHOSP 공급자 네트워크 요구 사항

OpenShift Container Platform 클러스터를 설치하기 전에 RHOSP(Red Hat OpenStack Platform) 배포 및 공급자 네트워크가 다음과 같은 여러 조건을 충족해야 합니다.

- RHOSP 네트워킹 서비스(Neutron)가 활성화되어 RHOSP 네트워킹 API를 통해 액세스할 수 있습니다.
- RHOSP 네트워킹 서비스에는 포트 보안 및 허용되는 주소 쌍 확장 기능이 활성화되어 있습니다.
- 공급자 네트워크는 다른 테넌트와 공유할 수 있습니다.

작은 정보

openstack network create 명령을 --share 플래그와 함께 사용하여 공유할 수 있는 네트워크를 생성합니다.

- 클러스터를 설치하는 데 사용하는 RHOSP 프로젝트는 공급자 네트워크와 적절한 서브넷을 소유해야 합니다.

작은 정보

"openshift"라는 프로젝트의 네트워크를 만들려면 다음 명령을 입력합니다.

```
$ openstack network create --project openshift
```

"openshift"라는 프로젝트의 서브넷을 만들려면 다음 명령을 입력합니다.

```
$ openstack subnet create --project openshift
```

RHOSP에서 네트워크 생성에 대한 자세한 내용은 공급자 네트워크 설명서를 참조하십시오.

admin 사용자가 클러스터를 소유하는 경우 해당 사용자로 설치 프로그램을 실행하여 네트워크에서 포트를 생성해야 합니다.

중요

공급자 네트워크는 클러스터를 생성하는 데 사용되는 RHOSP 프로젝트에서 소유해야 합니다. RHOSP 컴퓨팅 서비스(Nova)는 해당 네트워크의 포트를 요청할 수 없습니다.

- 공급자 네트워크가 기본적으로 RHOSP 메타데이터 서비스 IP 주소 169.254.169.254에 연결할 수 있는지 확인합니다.

  RHOSP SDN 및 네트워킹 서비스 구성에 따라 서브넷을 생성할 때 경로를 제공해야 할 수 있습니다. 예를 들면 다음과 같습니다.

```
$ openstack subnet create --dhcp --host-route
destination=169.254.169.254/32,gateway=192.0.2.2 ...
```

- 선택 사항: 네트워크를 보호하려면 단일 프로젝트에 대한 네트워크 액세스를 제한하는 역할 기반 액세스 제어(RBAC) 규칙을 생성합니다.

15.2.11.8.2. 공급자 네트워크에 기본 인터페이스가 있는 클러스터 배포

RHOSP(Red Hat OpenStack Platform) 공급자 네트워크에 기본 네트워크 인터페이스가 있는 OpenShift Container Platform 클러스터를 배포할 수 있습니다.

- RHOSP(Red Hat OpenStack Platform) 배포는 "클러스터 설치를 위한 RHOSP 공급자 네트워크 요구 사항"에 설명된 대로 구성됩니다.

절차

1. 텍스트 편집기에서 install-config.yaml 파일을 엽니다.
2. platform.openstack.apiVIP 속성의 값을 API VIP의 IP 주소로 설정합니다.
3. platform.openstack.ingressVIP 속성 값을 Ingress VIP의 IP 주소로 설정합니다.
4. platform.openstack.machinesSubnet 속성 값을 공급자 네트워크 서브넷의 UUID로 설정합니다.

5. **networking.machineNetwork.cidr** 속성 값을 공급자 네트워크 서브넷의 **CIDR** 블록으로 설정합니다.

중요

platform.openstack.apiVIP 및 **platform.openstack.ingressVIP** 속성은 모두 **networking.machineNetwork.cidr** 블록에서 할당되지 않은 **IP** 주소여야 합니다.

RHOSP 공급자 네트워크를 사용하는 클러스터의 설치 구성 파일 섹션

```
...
platform:
  openstack:
    apiVIP: 192.0.2.13
    ingressVIP: 192.0.2.23
    machinesSubnet: fa806b2f-ac49-4bce-b9db-124bc64209bf
    # ...
networking:
  machineNetwork:
  - cidr: 192.0.2.0/24
```

주의

기본 네트워크 인터페이스에 공급자 네트워크를 사용하는 동안 **platform.openstack.externalNetwork** 또는 **platform.openstack.externalDNS** 매개변수를 설정할 수 없습니다.

클러스터를 배포할 때 설치 프로그램에서 **install-config.yaml** 파일을 사용하여 공급자 네트워크에 클러스터를 배포합니다.

작은 정보

공급자 네트워크를 포함한 다른 네트워크를 **platform.openstack.additionalNetworkIDs** 목록에 추가할 수 있습니다.

클러스터를 배포한 후 **Pod**를 추가 네트워크에 연결할 수 있습니다. 자세한 내용은 여러 네트워크 이해를 참조하십시오.

15.2.11.9. RHOSP용 샘플 사용자 지정 install-config.yaml 파일

이 샘플 **install-config.yaml**은 가능한 모든 **RHOSP(Red Hat OpenStack Platform)** 사용자 지정 옵션을 보여줍니다.

중요

이 샘플 파일은 참조용으로만 제공됩니다. **install-config.yaml** 파일은 설치 프로그램을 사용하여 받아야 합니다.

```
apiVersion: v1
baseDomain: example.com
controlPlane:
  name: master
  platform: {}
  replicas: 3
compute:
- name: worker
  platform:
    openstack:
      type: ml.large
  replicas: 3
metadata:
  name: example
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  serviceNetwork:
  - 172.30.0.0/16
  networkType: OpenShiftSDN
platform:
  openstack:
    cloud: mycloud
    externalNetwork: external
    computeFlavor: m1.xlarge
    apiFloatingIP: 128.0.0.1
```

```
fips: false
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```

15.2.12. 컴퓨팅 머신 유사성 설정

선택적으로 설치하는 동안 컴퓨팅 시스템의 선호도 정책을 설정할 수 있습니다. 설치 프로그램은 기본적으로 컴퓨팅 시스템의 선호도 정책을 선택하지 않습니다.

설치 후 특정 RHOSP 서버 그룹을 사용하는 머신 세트를 생성할 수도 있습니다.

참고

컨트롤 플레인 시스템은 **soft-anti-affinity** 정책을 사용하여 생성됩니다.

작은 정보

RHOSP 인스턴스 스케줄링 및 배치에 대한 자세한 내용은 RHOSP 설명서에서 확인할 수 있습니다.

사전 요구 사항

- **install-config.yaml** 파일을 생성하고 수정합니다.

절차

1. RHOSP 명령줄 인터페이스를 사용하여 컴퓨팅 시스템의 서버 그룹을 생성합니다. 예를 들면 다음과 같습니다.

```
$ openstack \
  --os-compute-api-version=2.15 \
  server group create \
  --policy anti-affinity \
  my-openshift-worker-group
```

자세한 내용은 서버 그룹 생성 명령 설명서를 참조하십시오.

2. 설치 프로그램이 포함된 디렉터리로 변경하고 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory>
```

다음과 같습니다.

installation_directory

클러스터의 **install-config.yaml** 파일이 포함된 디렉토리의 이름을 지정합니다.

3. **MachineSet** 정의 파일인 **manifests/99_openshift-cluster-api_worker-machineset-0.yaml** 을 엽니다.

4. **spec.template.spec.providerSpec.value** 속성 아래에 있는 정의에 속성 **serverGroupID** 를 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_ID>
    machine.openshift.io/cluster-api-machine-role: <node_role>
    machine.openshift.io/cluster-api-machine-type: <node_role>
  name: <infrastructure_ID>-<node_role>
  namespace: openshift-machine-api
spec:
  replicas: <number_of_replicas>
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_ID>
      machine.openshift.io/cluster-api-machineset: <infrastructure_ID>-<node_role>
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_ID>
        machine.openshift.io/cluster-api-machine-role: <node_role>
        machine.openshift.io/cluster-api-machine-type: <node_role>
        machine.openshift.io/cluster-api-machineset: <infrastructure_ID>-<node_role>
    spec:
      providerSpec:
        value:
          apiVersion: openstackproviderconfig.openshift.io/v1alpha1
          cloudName: openstack
          cloudsSecret:
            name: openstack-cloud-credentials
            namespace: openshift-machine-api
          flavor: <nova_flavor>
          image: <glance_image_name_or_location>
          serverGroupID: aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee (1)
          kind: OpenstackProviderSpec
```

```
networks:
- filter: {}
  subnets:
  - filter:
      name: <subnet_name>
      tags: openshiftClusterID=<infrastructure_ID>
securityGroups:
- filter: {}
  name: <infrastructure_ID>-<node_role>
serverMetadata:
  Name: <infrastructure_ID>-<node_role>
  openshiftClusterID: <infrastructure_ID>
tags:
- openshiftClusterID=<infrastructure_ID>
trunk: true
userDataSecret:
  name: <node_role>-user-data
availabilityZone: <optional_openstack_availability_zone>
```

1

서버 그룹의 **UUID**를 여기에 추가합니다.

5. 선택 사항: **manifests/99_openshift-cluster-api_worker-machineset-0.yaml** 파일을 백업합니다. 설치 프로그램은 클러스터를 생성할 때 **manifests/** 디렉터리를 삭제합니다.

클러스터를 설치할 때 설치 프로그램은 수정한 **MachineSet** 정의를 사용하여 **RHOSP** 서버 그룹 내에서 컴퓨팅 머신을 생성합니다.

15.2.13. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 **SSH** 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 **Ignition** 구성 파일을 통해 **RHCOS(Red Hat Enterprise Linux CoreOS)** 노드에 전달되며 노드에 대한 **SSH** 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 **core** 사용자의 **~/.ssh/authorized_keys** 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 **core**로 **RHCOS** 노드에 **SSH**로 **SSH** 연결을 수행할 수 있습니다 . **SSH**를 통해 노드에 액세스하려면 로컬 사용자의 **SSH**에서 개인 키 **ID**를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 **SSH**를 실행하려면 설치 프로세스 중에 **SSH** 공용 키를 지정해야 합니다. **./openshift-install gather** 명령에도 **SSH** 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

절차

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: ~/.ssh/id_ed25519)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 '~/.ssh 디렉터리에 있는지 확인하십시오.

참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 ed25519 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 rsa 또는 ecdsa 알고리즘을 사용하는 키를 생성합니다.

2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 ~/.ssh/id_ed25519.pub 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 ./openshift-install gather 명령을 사용하려는 경우 필요합니다.

참고

일부 배포에서는 ~/.ssh/id_rsa 및 ~/.ssh/id_dsa와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. ssh-agent 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. ssh-agent에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1

SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: ~/.ssh/id_ed25519).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

15.2.14. 환경에 대한 액세스 활성화

배포 시 모든 OpenShift Container Platform 시스템은 RHOSP(Red Hat OpenStack Platform) 테넌트 네트워크에서 생성됩니다. 따라서 대부분의 RHOSP 배포에서 직접 액세스할 수 없습니다.

설치시 부동 IP 주소 (FIP)를 사용하여 OpenShift Container Platform API 및 애플리케이션의 액세스를 설정할 수 있습니다. FIP를 구성하지 않고 설치를 완료 할 수도 있지만 설치 프로그램은 외부에서 API 또는 애플리케이션에 액세스하는 방법을 설정하지 않습니다.

15.2.14.1. 부동 IP 주소로 액세스 활성화

OpenShift Container Platform API 및 클러스터 애플리케이션에 대한 외부 액세스 용으로 유동 IP (FIP) 주소를 생성합니다.

절차

1. RHOSP(Red Hat OpenStack Platform) CLI를 사용하여 API FIP를 생성합니다.

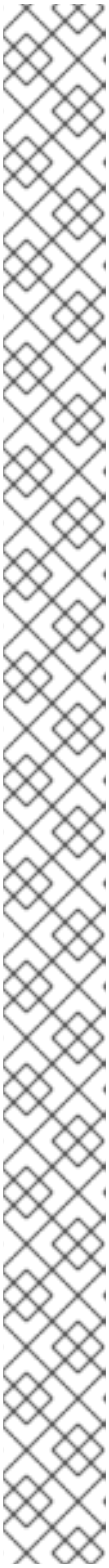
```
$ openstack floating ip create --description "API <cluster_name>.<base_domain>"
<external_network>
```

2. RHOSP(Red Hat OpenStack Platform) CLI를 사용하여 앱 또는 Ingress, FIP를 생성합니다.

```
$ openstack floating ip create --description "Ingress <cluster_name>.<base_domain>"
<external_network>
```

3. API 및 Ingress FIP의 DNS 서버에 이러한 패턴에 따라 레코드를 추가합니다.

```
api.<cluster_name>.<base_domain>.  IN  A  <API_FIP>
*.apps.<cluster_name>.<base_domain>. IN  A <apps_FIP>
```



참고

DNS 서버를 제어하지 않으면 /etc/hosts 파일에 다음과 같은 클러스터 도메인 이름을 추가하여 클러스터에 액세스할 수 있습니다.

- <api_floating_ip> api.<cluster_name>.<base_domain>
- <application_floating_ip> grafana-openshift-monitoring.apps.<cluster_name>.<base_domain>
- <application_floating_ip> prometheus-k8s-openshift-monitoring.apps.<cluster_name>.<base_domain>
- <application_floating_ip> oauth-openshift.apps.<cluster_name>.<base_domain>
- <application_floating_ip> console-openshift-console.apps.<cluster_name>.<base_domain>
- application_floating_ip integrated-oauth-server-openshift-authentication.apps.<cluster_name>.<base_domain>

/etc/hosts 파일의 클러스터 도메인 이름은 웹 콘솔과 클러스터의 모니터링 인터페이스에 로컬로 액세스할 수 있는 권한을 부여합니다. kubectl 또는 oc 에서도 사용할 수 있습니다. <application_floating_ip>를 가리키는 추가 항목을 사용하여 사용자 애플리케이션에 액세스할 수 있습니다. 이 작업을 수행하면 사용자만 API 및 애플리케이션에 액세스할 수 있어 프로덕션 배포에는 적합하지 않지만 개발 및 테스트용 설치는 가능합니다.

4. FIP를 다음 매개 변수의 값으로 install-config.yaml 파일에 추가하십시오.
   - platform.openstack.ingressFloatingIP
   - platform.openstack.apiFloatingIP

이러한 값을 사용하는 경우 **install-config.yaml** 파일에서 **platform.openstack.externalNetwork** 매개 변수 값으로 외부 네트워크를 입력해야 합니다.

작은 정보

부동 **IP** 주소를 할당하고 방화벽 구성을 업데이트하여 클러스터 외부에서 **OpenShift Container Platform** 리소스를 사용할 수 있습니다.

15.2.14.2. 유동 IP 주소없이 설치 완료

유동 **IP** 주소를 지정하지 않고도 **Red Hat OpenStack Platform (RHOSP)**에 **OpenShift Container Platform**을 설치할 수 있습니다.

install-config.yaml 파일에서 다음 매개 변수를 정의하지 마십시오.

- **platform.openstack.ingressFloatingIP**
- **platform.openstack.apiFloatingIP**

외부 네트워크를 제공 할 수 없는 경우 **platform.openstack.externalNetwork**를 비워 둘 수도 있습니다. **platform.openstack.externalNetwork** 값을 지정하지 않으면 라우터가 생성되지 않으며 추가 작업없이 설치 프로그램이 **Glance**에서 이미지를 검색하지 못합니다. 외부 연결을 직접 구성해야 합니다.

유동 **IP** 주소 또는 이름 확인 부족으로 인해 클러스터 **API**에 연결할 수 없는 시스템에서 설치 프로그램을 실행하면 설치에 실패합니다. 이러한 경우 설치 실패를 방지하기 위해 프록시 네트워크를 사용하거나 시스템과 동일한 네트워크에있는 시스템에서 설치 프로그램을 실행할 수 있습니다.

참고

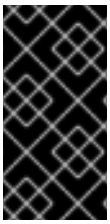
API 및 Ingress 포트의 DNS 레코드를 생성하여 이름 확인을 활성화할 수 있습니다. 예를 들면 다음과 같습니다.

```
api.<cluster_name>.<base_domain>.  IN  A  <api_port_IP>
*.apps.<cluster_name>.<base_domain>. IN  A <ingress_port_IP>
```

DNS 서버를 제어하지 않으면 /etc/hosts 파일에 레코드를 추가할 수 있습니다. 이 작업을 수행하면 사용자만 API에 액세스할 수 있어 프로덕션 배포에는 적합하지 않지만 개발 및 테스트용 설치는 가능합니다.

15.2.15. 클러스터 배포

호환되는 클라우드 플랫폼에 OpenShift Container Platform을 설치할 수 있습니다.



중요

최초 설치 과정에서 설치 프로그램의 create cluster 명령을 한 번만 실행할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

절차

1. 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터 배포를 초기화합니다.

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
  --log-level=info 2
```

1 <installation_directory> 값으로 사용자 지정한 ./install-config.yaml 파일의 위치를 지정합니다.

2 다른 설치 세부 사항을 보려면 info 대신 warn, debug 또는 error를 지정합니다.

참고

호스트에 구성된 클라우드 공급자 계정에 클러스터를 배포하기에 충분한 권한이 없는 경우, 설치 프로세스가 중단되고 누락된 권한을 알리는 메시지가 표시됩니다.

클러스터 배포가 완료되면 웹 콘솔로 연결되는 링크와 **kubeadmin** 사용자의 인증 정보가 포함된 클러스터 액세스 지침이 사용자 터미널에 표시됩니다.

출력 예

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-
ymBZj-Wt5AL"
INFO Time elapsed: 36m22s
```

참고

설치에 성공하면 클러스터 액세스 및 인증 정보도 **<installation_directory>/.openshift_install.log**로 출력됩니다.

중요

- 설치 프로그램에서 생성하는 **Ignition** 구성 파일에 **24**시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 **24**시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 **kubelet** 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(**CSR**)을 수동으로 승인해야 합니다. 자세한 내용은 ***Recovering from expired control plane certificates*** 문서를 참조하십시오.

- 클러스터를 설치한 후 **24**시간에서 **22**시간까지의 인증서가 교체되기 때문에 생성된 후 **12**시간 이내에 **Ignition** 구성 파일을 사용하는 것이 좋습니다. **12**시간 이내에 **Ignition** 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

중요

설치 프로그램에서 생성되는 파일이나 설치 프로그램을 삭제해서는 안 됩니다. 클러스터를 삭제하려면 두 가지가 모두 필요합니다.

15.2.16. 클러스터 상태 확인

설치 중 또는 설치 후 **OpenShift Container Platform** 클러스터의 상태를 확인할 수 있습니다.

절차

1. 클러스터 환경에서 관리자의 **kubeconfig** 파일을 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

kubeconfig 파일에는 **CLI**에서 올바른 클러스터 및 **API** 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다.

2. 배포 후 생성된 컨트롤 플레인 및 컴퓨팅 시스템을 확인합니다.

```
$ oc get nodes
```

3. 클러스터 버전을 확인합니다.

```
$ oc get clusterversion
```

4. Operator 상태를 확인합니다.

```
$ oc get clusteroperator
```

5. 클러스터에서 실행 중인 모든 Pod를 확인합니다.

```
$ oc get pods -A
```

15.2.17. CLI를 사용하여 클러스터에 로그인

클러스터 kubeconfig 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. kubeconfig 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- oc CLI를 설치했습니다.

절차

1. kubeadmin 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1. <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 **oc** 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

추가 리소스

- **OpenShift Container Platform** 웹 콘솔 액세스 및 이해에 대한 자세한 내용은 웹 콘솔에 액세스를 참조하십시오.

15.2.18. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 **Telemetry** 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 **Telemetry**가 자동으로 실행되고 **OpenShift Cluster Manager** 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 **OpenShift Cluster Manager**를 사용하여 자동으로 또는 **OpenShift Cluster Manager**를 사용하여 수동으로 유지 관리되는지 확인한 후 **subscription watch**를 사용하여 계정 또는 다중 클러스터 수준에서 **OpenShift Container Platform** 서브스크립션을 추적합니다.

추가 리소스

- **Telemetry** 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

15.2.19. 다음 단계

- 클러스터를 사용자 지정합니다.

- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.

- 노드 포트에 대한 외부 액세스를 활성화해야 하는 경우 노드 포트를 사용하여 Ingress 클러스터 트래픽을 구성합니다.

- 유동 IP 주소를 통한 애플리케이션 트래픽을 허용하도록 RHOSP를 구성하지 않은 경우 유동 IP 주소로 RHOSP 액세스를 구성합니다.

15.3. KURYR로 OPENSTACK에 클러스터 설치

OpenShift Container Platform 버전 4.9에서는 Kuryr SDN을 사용하는 사용자 지정 클러스터를 RHOSP(Red Hat OpenStack Platform)에 설치할 수 있습니다. 설치를 사용자 지정하려면 클러스터를 설치하기 전에 install-config.yaml에서 매개변수를 수정합니다.

15.3.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.

- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.

- OpenShift Container Platform 4.9가 OpenShift 클러스터에 지원되는 플랫폼 섹션을 사용하여 현재 RHOSP 버전과 호환되는지 확인했습니다. RHOSP 지원 매트릭스의 OpenShift Container Platform을 확인하여 여러 버전의 플랫폼 지원을 비교할 수도 있습니다.

- RHOSP에 블록 스토리지(Cinder) 또는 개체 스토리지(Swift)와 같은 스토리지 서비스가 설치되어 있어야 합니다. 개체 스토리지는 OpenShift Container Platform 레지스트리 클러스터 배포에 권장되는 스토리지 기술입니다. 자세한 정보는 스토리지 최적화를 참조하십시오.

15.3.2. Kuryr SDN 정보

Kuryr 는 Neutron 및 Octavia RHOSP(Red Hat OpenStack Platform) 서비스를 사용하여 Pod 및 서비스에 네트워킹을 제공하는 컨테이너 네트워크 인터페이스(CNI) 플러그인 솔루션입니다.

Kuryr와 OpenShift Container Platform 통합은 주로 RHOSP VM에서 실행되는 OpenShift Container Platform 클러스터를 위해 설계되었습니다. Kuryr는 OpenShift Container Platform Pod를

RHOSP SDN에 연결하여 네트워크 성능을 향상시킵니다. 또한 Pod와 RHOSP 가상 인스턴스 간의 상호 연결성을 제공합니다.

Kuryr 구성 요소는 openshift-kuryr 네임스페이스를 사용하여 OpenShift Container Platform에서 Pod로 설치됩니다.

- kuryr-controller- - master 노드에 설치된 단일 서비스 인스턴스로, OpenShift Container Platform에서 Deployment 개체로 모델링됩니다.
- kuryr- cni - 각 OpenShift Container Platform 노드에서 Kuryr를 CNI 드라이버로 설치 및 구성하는 컨테이너로, OpenShift Container Platform에서 DaemonSet 개체로 모델링됩니다.

Kuryr 컨트롤러는 OpenShift Container Platform API 서버에서 Pod, 서비스 및 네임스페이스 생성, 업데이트 및 삭제 이벤트를 감시합니다. OpenShift Container Platform API 호출을 Neutron 및 Octavia의 해당 개체에 매핑합니다. 즉 Neutron 트렁크 포트 기능을 구현하는 모든 네트워크 솔루션을 사용하여 Kuryr를 통해 OpenShift Container Platform을 지원할 수 있습니다. 여기에는 OVS(Open vSwitch), OVN(Open Virtual Network)과 같은 오픈 소스 솔루션과 Neutron 호환 상용 SDN이 포함됩니다.

Kuryr는 캡슐화된 RHOSP 테넌트 네트워크에서의 OpenShift Container Platform 배포에 권장되며 RHOSP 네트워크를 통해 캡슐화된 OpenShift Container Platform SDN을 실행하는 경우와 같은 이중 캡슐화를 피할 수 있습니다.

공급자 네트워크 또는 테넌트 VLAN을 사용하는 경우에는 이중 캡슐화를 피하기 위해 Kuryr를 사용하지 않아도 됩니다. 성능상의 이점은 무시할만한 수준입니다. 그러나 구성에 따라 두 개의 오버레이가 없도록 Kuryr을 사용하는 방법이 계속 유용할 수 있습니다.

다음 기준이 모두 해당되는 배포에는 Kuryr가 권장되지 않습니다.

- RHOSP 버전이 16 미만입니다.
- 배포 시 UDP 서비스 또는 소수의 하이퍼바이저에서 많은 수의 TCP 서비스를 사용합니다.

또는

- **ovn-octavia Octavia** 드라이버가 비활성화되었습니다.
- 배포 시 소수의 하이퍼바이저에서 많은 수의 **TCP** 서비스를 사용합니다.

15.3.3. Kuryr를 사용하는 RHOSP에 OpenShift Container Platform을 설치하기 위한 리소스 지침

Kuryr SDN을 사용하는 경우 **Pod**, 서비스, 네임스페이스, 네트워크 정책은 **RHOSP** 할당량의 리소스를 사용하며 이로 인해 최소 요구사항이 증가합니다. **Kuryr**는 또한 기본 설치에 필요한 요구사항 이외에 몇 가지 추가 요구사항을 갖습니다.

기본 클러스터의 최소 요구사항을 충족할 수 있는 할당량은 다음과 같습니다.

표 15.7. Kuryr를 사용하는 RHOSP의 기본 OpenShift Container Platform 클러스터에 권장되는 리소스

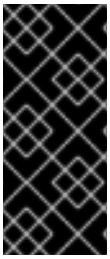
리소스 이름	값
부동 IP 주소	3 - LoadBalancer 유형의 예상 서비스 수 추가
포트	1500 - Pod당 1개 필요
라우터	1
서브넷	250 - 네임스페이스/프로젝트당 1개 필요
네트워크	250 - 네임스페이스/프로젝트당 1개 필요
RAM	112GB
vCPU	28
볼륨 스토리지	275GB
인스턴스	7
보안 그룹	250 - 서비스/NetworkPolicy당 1개 필요
보안 그룹 규칙	1000
로드 밸런서	100 - 서비스당 1개 필요
로드 밸런서 리스너	500 - 서비스 노출 포트당 1개 필요

리소스 이름	값
로드 밸런서 풀	500 - 서비스 노출 포트당 1개 필요

권장 리소스보다 적은 리소스로도 클러스터가 작동할 수 있지만 성능은 보장되지 않습니다.

중요

swiftoperator 역할을 가진 사용자 계정으로 RHOSP 개체 스토리지(Swift)를 사용하고 운영하는 경우 OpenShift Container Platform 이미지 레지스트리의 기본 백엔드로 사용됩니다. 이 경우 볼륨 스토리지 요구사항은 175GB입니다. Swift 공간 요구사항은 이미지 레지스트리의 크기에 따라 다릅니다.



중요

OVN Octavia 드라이버가 아닌 Amphora 드라이버와 함께 RHOSP(Red Hat OpenStack Platform) 버전 16을 사용하는 경우 보안 그룹은 사용자 프로젝트 대신 서비스 계정과 연결됩니다.

리소스를 설정할 때 다음 사항을 고려하십시오.

- 필요한 포트 수가 Pod 수보다 많습니다. Kuryr는 포트 풀을 사용하여 Pod가 미리 생성된 포트를 사용할 수 있도록 준비하여 Pod의 부팅 시간을 단축시킵니다.
- 각 NetworkPolicy는 RHOSP 보안 그룹에 매핑되며 NetworkPolicy 사양에 따라 하나 이상의 규칙이 보안 그룹에 추가됩니다.
- 각 서비스는 RHOSP 로드 밸런서에 매핑됩니다. 할당량에 필요한 보안 그룹 수를 추정할 때 이 요구사항을 고려하십시오.

  RHOSP 버전 15 이하 또는 ovn-octavia driver를 사용하는 경우 각 로드 밸런서에 사용자 프로젝트와 보안 그룹이 있습니다.
- 할당량은 로드 밸런서 리소스(예: VM 리소스)를 고려하지 않지만 RHOSP 배포 크기를 결정할 때 이러한 리소스를 고려해야 합니다. 기본 설치에는 50개 이상의 로드 밸런서가 있으며 클러스터가 이 로드 밸런서를 수용할 수 있어야 합니다.

OVN Octavia 드라이버가 활성화된 상태에서 RHOSP 버전 16을 사용하는 경우에는 로드 밸런서 VM이 하나만 생성됩니다. 서비스는 OVN 흐름을 통해 부하를 분산시킵니다.

OpenShift Container Platform 배포는 컨트롤 플레인 시스템, 컴퓨팅 시스템, 부트스트랩 시스템으로 구성됩니다.

Kuryr SDN을 활성화하려면 환경이 다음 요구사항을 충족해야 합니다.

- RHOSP 13+를 실행합니다.
- Octavia와 함께 Overcloud가 설치되어 있습니다.
- Neutron 트렁크 포트 확장을 사용합니다.
- ovs-hybrid 대신 ML2/OVS Neutron 드라이버를 사용하는 경우 openvswitch를 사용합니다.

15.3.3.1. 할당량 늘리기

Kuryr SDN을 사용하는 경우 Pod, 서비스, 네임스페이스 및 네트워크 정책에서 사용하는 RHOSP(Red Hat OpenStack Platform) 리소스를 충족하기 위해 할당량을 늘려야 합니다.

프로세스

- 다음 명령을 실행하여 프로젝트 할당량을 늘립니다.

```
$ sudo openstack quota set --secgroups 250 --secgroup-rules 1000 --ports 1500 --
subnets 250 --networks 250 <project>
```

15.3.3.2. Neutron 구성

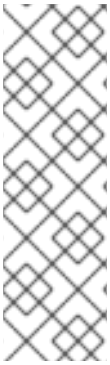
Kuryr CNI는 Neutron 트렁크 확장을 활용하여 컨테이너를 RHOSP(Red Hat OpenStack Platform) SDN에 연결하므로 Kuryr이 제대로 작동하려면 trunks 확장을 사용해야 합니다.

또한 기본 ML2/OVS Neutron 드라이버를 활용하는 경우 보안 그룹이 트렁크 서브포트에서 적용되고 Kuryr가 네트워크 정책을 올바르게 처리할 수 있도록 방화벽이 ovs_hybrid 대신 openvswitch로 설정되어야 합니다.

15.3.3.3. Octavia 구성

Kuryr SDN은 RHOSP(Red Hat OpenStack Platform)의 Octavia LBaaS를 사용하여 OpenShift Container Platform 서비스를 구현합니다. 따라서 Kuryr SDN을 사용하려면 RHOSP에서 Octavia 구성 요소를 설치하고 구성해야 합니다.

Octavia를 활성화하려면 RHOSP Overcloud 설치 중에 Octavia 서비스를 포함하거나 Overcloud가 이미 존재하는 경우 Octavia 서비스를 업그레이드해야 합니다. Octavia를 활성화하는 다음 단계는 Overcloud 새로 설치 또는 Overcloud 업데이트에 모두 적용됩니다.



참고

다음 단계는 Octavia를 처리할 때 RHOSP 배포 과정에서 필요한 주요 부분만을 다룹니다. 레지스트리 방법도 다를 수 있습니다.

이 예에서는 로컬 레지스트리 방법을 사용합니다.

프로세스

1. 로컬 레지스트리를 사용하는 경우 이미지를 레지스트리에 업로드하는 템플릿을 만듭니다. 예를 들면 다음과 같습니다.

```
(undercloud) $ openstack overcloud container image prepare \
-e /usr/share/openstack-tripleo-heat-templates/environments/services-
docker/octavia.yaml \
--namespace=registry.access.redhat.com/rhosp13 \
--push-destination=<local-ip-from-undercloud.conf>:8787 \
--prefix=openstack- \
--tag-from-label {version}-{product-version} \
--output-env-file=/home/stack/templates/overcloud_images.yaml \
--output-images-file /home/stack/local_registry_images.yaml
```

2. local_registry_images.yaml 파일에 Octavia 이미지가 포함되어 있는지 확인합니다. 예를 들면 다음과 같습니다.

```
...
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-api:13.0-43
  push_destination: <local-ip-from-undercloud.conf>:8787
```

```
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-health-
manager:13.0-45
  push_destination: <local-ip-from-undercloud.conf>:8787
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-
housekeeping:13.0-45
  push_destination: <local-ip-from-undercloud.conf>:8787
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-worker:13.0-44
  push_destination: <local-ip-from-undercloud.conf>:8787
```

참고

Octavia 컨테이너 버전은 설치된 특정 RHOSP 릴리스에 따라 다릅니다.

3. registry.redhat.io에서 Undercloud 노드로 컨테이너 이미지를 가져옵니다.

```
(undercloud) $ sudo openstack overcloud container image upload \
  --config-file  /home/stack/local_registry_images.yaml \
  --verbose
```

네트워크 및 Undercloud 디스크의 속도에 따라 다소 시간이 걸릴 수 있습니다.

4. Octavia 로드 밸런서는 OpenShift Container Platform API에 액세스하는 데 사용되므로 연결에 대한 리스너의 기본 제한 시간을 늘려야 합니다. 기본 제한 시간은 50초입니다. 다음 파일을 Overcloud 배포 명령에 전달하여 제한 시간을 20분으로 늘립니다.

```
(undercloud) $ cat octavia_timeouts.yaml
parameter_defaults:
  OctaviaTimeoutClientData: 1200000
  OctaviaTimeoutMemberData: 1200000
```

참고

RHOSP 13.0.13+에는 필요하지 않습니다.

5. Octavia로 Overcloud 환경을 설치 또는 업데이트합니다.

```
$ openstack overcloud deploy --templates \
  -e /usr/share/openstack-tripleo-heat-templates/environments/services-
docker/octavia.yaml \
  -e octavia_timeouts.yaml
```

참고

이 명령에는 **Octavia**와 관련된 파일만 포함됩니다. 해당 파일은 **RHOSP**의 특정 설치에 따라 다릅니다. 자세한 내용은 **RHOSP** 문서를 참조하십시오. **Octavia** 설치 사용자 지정에 대한 자세한 내용은 **Director**를 사용하여 **Octavia** 설치를 참조하십시오.

참고

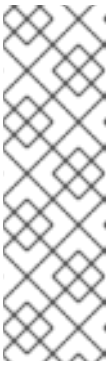
Kuryr SDN을 활용하는 경우 **Overcloud** 설치에는 **Neutron trunk** 확장이 필요합니다. 이 확장은 기본적으로 디렉터 배포에서 사용할 수 있습니다. **Neutron** 백엔드가 **ML2/OVS**인 경우 기본 **ovs-hybrid** 대신 **openvswitch** 방화벽을 사용합니다. 백엔드가 **ML2/OVN**인 경우에는 수정하지 않아도 됩니다.

6.

RHOSP 13.0.13 이전 버전에서 프로젝트를 생성한 후 프로젝트 **ID**를 **octavia.conf** 구성 파일에 추가합니다.

- 트래픽이 **Octavia** 로드 밸런서를 통과할 때와 같이 서비스 전체에 네트워크 정책을 적용하려면 **Octavia**가 사용자 프로젝트에서 **Amphora VM** 보안 그룹을 생성해야 합니다.

이 변경으로 인해 필요한 로드 밸런서 보안 그룹이 해당 프로젝트에 속하게 되며 서비스 격리를 적용하도록 업데이트될 수 있습니다.



참고

RHOSP 13.0.13 버전 이상에서는 이 작업이 필요하지 않습니다.

Octavia는 로드 밸런서 **VIP**에 대한 액세스를 제한하는 새로운 **ACL API**를 구현합니다.

a.

프로젝트 **ID** 가져오기

```
$ openstack project show <project>
```

출력 예

```
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description |                                  |
| domain_id   | default                          |
| enabled     | True                             |
| id          | PROJECT_ID                       |
| is_domain   | False                            |
| name        | *<project>*                      |
| parent_id   | default                          |
| tags        | []                               |
+-------------+----------------------------------+
```

b. 컨트롤러의 **octavia.conf**에 프로젝트 **ID**를 추가합니다.

i. **stackrc** 파일을 소싱합니다.

```
$ source stackrc  # Undercloud credentials
```

ii. **Overcloud** 컨트롤러를 나열합니다.

```
$ openstack server list
```

출력 예

```
+--------------------------------------+-------------+--------+----------------------+-------
---------+------------+
|
| ID                                   | Name        | Status | Networks
| Image         | Flavor     |
|
+--------------------------------------+-------------+--------+----------------------+-------
---------+------------+
|
| 6bef8e73-2ba5-4860-a0b1-3937f8ca7e01 | controller-0 | ACTIVE |
ctlplane=192.168.24.8 | overcloud-full | controller |
|
| dda3173a-ab26-47f8-a2dc-8473b4a67ab9 | compute-0    | ACTIVE |
ctlplane=192.168.24.6 | overcloud-full | compute    |
|
+--------------------------------------+-------------+--------+----------------------+-------
---------+------------+
```

iii.

컨트롤러에 **SSH**를 적용합니다.

```
$ ssh heat-admin@192.168.24.8
```

iv.

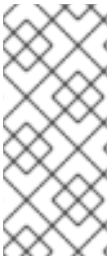
octavia.conf 파일을 편집하여 **Amphora** 보안 그룹이 사용자 계정에 있는 프로젝트 목록에 프로젝트를 추가합니다.

```
# List of project IDs that are allowed to have Load balancer security groups
# belonging to them.
amp_secgroup_allowed_projects = PROJECT_ID
```

c.

Octavia 작업자를 다시 시작하여 새 구성을 로드합니다.

```
controller-0$ sudo docker restart octavia_worker
```



참고

RHOSP 환경에 따라 **Octavia**가 **UDP** 리스너를 지원하지 않을 수 있습니다. **RHOSP** 버전 **13.0.13** 이하에서 **Kuryr SDN**을 사용하는 경우에는 **UDP** 서비스가 지원되지 않습니다. **RHOSP** 버전 **16** 이상은 **UDP**를 지원합니다.

15.3.3.3.1. Octavia OVN 드라이버

Octavia는 **Octavia API**를 통해 여러 공급자 드라이버를 지원합니다.

사용 가능한 모든 **Octavia** 공급자 드라이버를 보려면 명령줄에서 다음을 입력합니다.

```
$ openstack loadbalancer provider list
```

출력 예

```
+---------+-------------------------------------------------+
| name    | description                                     |
+---------+-------------------------------------------------+
| amphora | The Octavia Amphora driver.                     |
```

```
| octavia | Deprecated alias of the Octavia Amphora driver. |
| ovn     | Octavia OVN driver.                             |
+---------+-------------------------------------------------+
```

RHOSP 버전 16부터 Octavia OVN 공급자 드라이버(ovn)는 RHOSP 배포의 OpenShift Container Platform에서 지원됩니다.

ovn은 Octavia와 OVN이 제공하는 로드 밸런싱을 위한 통합 드라이버입니다. 기본 로드 밸런싱 기능을 지원하며 OpenFlow 규칙을 기반으로 합니다. 이 드라이버는 OVN Neutron ML2를 사용하는 배포에서 Director에 의해 Octavia에서 자동으로 활성화됩니다.

기본 드라이버는 Amphora 공급자 드라이버입니다. 그러나 ovn이 활성화되면 Kuryr는 ovn을 사용합니다.

Kuryr가 Amphora 대신 ovn을 사용하는 경우 다음과 같은 이점을 제공합니다.

- 리소스 요구사항 감소. Kuryr는 각 서비스마다 로드 밸런서 VM이 필요하지 않습니다.
- 네트워크 대기 시간 감소.
- 각 서비스마다 VM 대신 OpenFlow 규칙을 사용하므로 서비스 생성 속도가 향상됩니다.
- 로드 밸런싱 작업이 Amphora VM에 집중되지 않고 모든 노드에 분산됩니다.

RHOSP 클라우드가 버전 13에서 버전 16으로 업그레이드된 후 Octavia OVN 드라이버를 사용하도록 클러스터를 구성할 수 있습니다.

15.3.3.4. Kuryr를 사용한 설치의 알려진 제한 사항

Kuryr SDN과 함께 OpenShift Container Platform을 사용하는 경우 몇 가지 알려진 제한 사항이 있습니다.

RHOSP 일반 제한 사항

Kuryr SDN을 사용하는 **OpenShift Container Platform** 사용에는 모든 버전 및 환경에 적용되는 몇 가지 제한 사항이 있습니다.

- **NodePort** 유형의 **Service** 개체는 지원되지 않습니다.
- **OVN Octavia** 공급자 드라이버를 사용하는 클러스터는 끝점 개체의 **.spec.selector** 속성에 노드 또는 포드의 서브넷이 포함된 경우에만 **.subsets.addresses** 속성이 지정되지 않은 **Service** 개체를 지원합니다.
- 시스템이 생성된 서브넷이 라우터에 연결되어 있지 않거나 서브넷이 연결되어 있지만 라우터에 외부 게이트웨이가 설정되지 않은 경우 **Kuryr**는 **LoadBalancer**인 **Service** 개체에 대한 유동 **IP**를 생성할 수 없습니다.
- **Service** 오브젝트에 **sessionAffinity=ClientIP** 속성을 구성해도 효과가 없습니다. **Kuryr**는 이 설정을 지원하지 않습니다.

RHOSP 버전 제한 사항

Kuryr SDN을 사용하는 **OpenShift Container Platform**은 **RHOSP** 버전에 따라 몇 가지 제한 사항이 있습니다.

- 버전 **16** 미만의 **RHOSP**는 기본 **Octavia**로드 밸런서 드라이버**(Amphora)**를 사용합니다. 이 드라이버를 사용하려면 **OpenShift Container Platform** 서비스당 하나의 **Amphora** 로드 밸런서 **VM**이 배포되어야 합니다. 너무 많은 서비스를 생성하면 리소스가 부족해질 수 있습니다.

  OVN Octavia 드라이버가 비활성화된 상위 버전의 **RHOSP** 배포에도 **Amphora** 드라이버를 사용합니다. 하위 버전의 **RHOSP**와 동일한 리소스 문제가 있습니다.
- 버전 **13.0.13** 미만의 **Octavia RHOSP**는 **UDP** 리스너를 지원하지 않습니다. 따라서 **OpenShift Container Platform UDP** 서비스는 지원되지 않습니다.
- 버전 **13.0.13** 미만의 **Octavia RHOSP**는 동일한 포트에서 여러 프로토콜을 수신할 수 없습니다. **TCP, UDP** 등 다른 프로토콜에 동일한 포트를 노출하는 서비스는 지원되지 않습니다.
- **Kuryr SDN**은 서비스에서 자동 유휴 상태 해제를 지원하지 않습니다.

RHOSP 환경 제한 사항

배포 환경에 따라 **Kuryr SDN** 사용에 제한이 있습니다.

Octavia는 **UDP** 프로토콜과 다중 리스너를 지원하지 않으므로 **RHOSP** 버전이 **13.0.13** 미만이면 **Kuryr**에 따라 **Pod**가 **DNS** 확인을 위해 **TCP**를 사용해야 합니다.

Go 버전 **1.12** 이하에서는 **CGO** 지원이 비활성화된 상태로 컴파일된 애플리케이션이 **UDP**만 사용합니다. 이 경우 기본 **Go** 해결 프로그램이 **resolv.conf**의 **use-vc** 옵션을 인식하지 못합니다. 이 옵션은 **DNS** 확인을 위해 **TCP**가 적용되는지 여부를 제어합니다. 결과적으로 **DNS** 확인에 계속 **UDP**를 사용하여 실패하게 됩니다.

TCP 적용을 허용하려면 환경 변수 **CGO_ENABLED**를 **1**로 설정하거나(즉 **CGO_ENABLED=1**) 변수가 없는지 확인하여 애플리케이션을 컴파일합니다.

Go 버전 **1.13** 이상에서는 **UDP**를 사용한 **DNS** 확인이 실패하면 자동으로 **TCP**를 사용합니다.

참고

Alpine 기반 컨테이너를 포함한 **musl** 기반 컨테이너는 **use-vc** 옵션을 지원하지 않습니다.

RHOSP 업그레이드 제한 사항

RHOSP 업그레이드 프로세스에 따라 **Octavia API**가 변경될 수 있으며 로드 밸런서에 사용되는 **Amphora** 이미지로 업그레이드해야 할 수 있습니다.

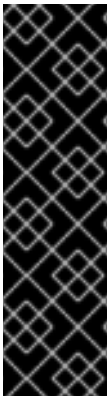
API 변경은 개별적으로 처리할 수 있습니다.

Amphora 이미지가 업그레이드되면 **RHOSP Operator**가 다음 두 가지 방법으로 기존 로드 밸런서 **VM**을 처리할 수 있습니다.

- 로드 밸런서 장애 조치를 트리거하여 각 **VM**을 업그레이드합니다.
- **VM** 업그레이드를 사용자가 처리하게 합니다.

Operator가 첫 번째 옵션을 선택하는 경우 장애 조치 중에 가동 중지 시간이 짧아질 수 있습니다.

Operator가 두 번째 옵션을 선택하는 경우 기존 로드 밸런서는 UDP 리스너와 같은 업그레이드된 Octavia API 기능을 지원하지 않습니다. 이 경우 이러한 기능을 사용하려면 사용자가 서비스를 다시 만들어야 합니다.



중요

OpenShift Container Platform이 UDP 로드 밸런싱을 지원하는 새로운 Octavia 버전을 감지하면 DNS 서비스를 자동으로 다시 생성합니다. 서비스를 다시 생성함으로써 서비스는 기본적으로 UDP 로드 밸런싱을 지원하게 됩니다.

서비스를 다시 생성하는 경우 DNS 서비스 가동이 약 1분 동안 중지됩니다.

15.3.3.5. 컨트롤 플레인 머신

기본적으로 OpenShift Container Platform 설치 프로세스는 세 개의 컨트롤 플레인 시스템을 생성합니다.

각 시스템의 요구사항은 다음과 같습니다.

- RHOSP 할당량의 인스턴스
- RHOSP 할당량의 포트
- 최소 16GB 메모리 및 vCPU 4개가 있는 플레이버
- RHOSP 할당량에서 최소 100GB 스토리지 공간

15.3.3.6. 컴퓨팅 머신

기본적으로 OpenShift Container Platform 설치 프로세스는 세 개의 컴퓨팅 시스템을 생성합니다.

각 시스템의 요구사항은 다음과 같습니다.

- RHOSP 할당량의 인스턴스
- RHOSP 할당량의 포트
- 8GB 메모리 및 vCPU 2개 이상 있는 플레이버
- RHOSP 할당량에서 최소 100GB 스토리지 공간

작은 정보

컴퓨팅 시스템은 OpenShift Container Platform에서 실행하는 애플리케이션을 호스팅합니다. 최대한 많이 실행하는 것이 좋습니다.

15.3.3.7. 부트스트랩 시스템

설치하는 동안 컨트롤 플레인을 유지하기 위해 부트스트랩 시스템이 임시로 프로비저닝됩니다. 프로덕션 컨트롤 플레인이 준비되면 부트스트랩 시스템이 프로비저닝 해제됩니다.

부트스트랩 시스템의 요구사항은 다음과 같습니다.

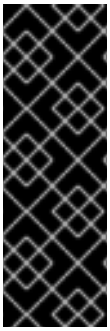
- RHOSP 할당량의 인스턴스
- RHOSP 할당량의 포트
- 최소 16GB 메모리 및 vCPU 4개가 있는 플레이버
- RHOSP 할당량에서 최소 100GB 스토리지 공간

15.3.4. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.

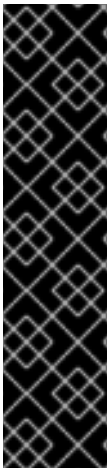


중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

15.3.5. RHOSP에서 Swift 활성화

Swift는 swiftoperator 역할을 가진 사용자 계정으로 운영됩니다. 설치 프로그램을 실행하려면 먼저 계정에 이 역할을 추가합니다.



중요

RHOSP(Red Hat OpenStack Platform) 개체 스토리지 서비스(일반적으로 Swift로 알려짐)를 사용할 수 있는 경우 OpenShift Container Platform이 이미지 레지스트리 스토리지로 사용합니다. 이 서비스를 사용할 수 없는 경우에는 설치 프로그램이 RHOSP 블록 스토리지 서비스(일반적으로 Cinder로 알려짐)를 사용합니다.

Swift가 있고 Swift를 사용하려면 액세스를 활성화해야 합니다. 존재하지 않거나 사용하지 않으려면 이 섹션을 건너 뛰십시오.

사전 요구 사항

- 대상 환경에 RHOSP 관리자 계정이 있습니다.
- Swift 서비스가 설치되어 있습니다.
- Ceph RGW에서 url의 계정 옵션이 활성화되어 있습니다.

프로세스

RHOSP에서 Swift를 활성화하려면:

1. RHOSP CLI의 관리자로서 Swift에 액세스할 계정에 swiftoperator 역할을 추가하십시오.

```
$ openstack role add --user <user> --project <project> swiftoperator
```

이제 RHOSP 배포에서 이미지 레지스트리에 Swift를 사용할 수 있습니다.

15.3.6. 외부 네트워크 액세스 확인

OpenShift Container Platform 설치 프로세스에는 외부 네트워크에 액세스해야 합니다. 외부 네트워크값을 제공해야 하며 그렇지 않으면 배포가 실패합니다. 프로세스를 시작하기 전에 외부 라우터 유형의 네트워크가 RHOSP(Red Hat OpenStack Platform)에 있는지 확인합니다.

사전 요구 사항

- DHCP 에이전트가 인스턴스의 DNS 쿼리를 전달하도록 OpenStack의 네트워킹 서비스 구성

프로세스

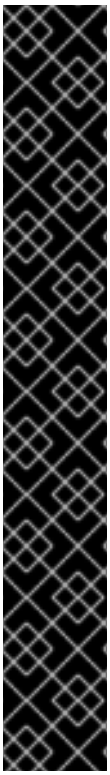
1. RHOSP CLI를 사용하여 '외부' 네트워크의 이름과 ID를 확인합니다.

```
$ openstack network list --long -c ID -c Name -c "Router Type"
```

출력 예

```
+--------------------------------------+---------------+-------------+
| ID                                   | Name          | Router Type |
+--------------------------------------+---------------+-------------+
| 148a8023-62a7-4672-b018-003462f8d7dc | public_network | External    |
+--------------------------------------+---------------+-------------+
```

외부 라우터 유형의 네트워크가 네트워크 목록에 나타납니다. 네트워크가 하나 이상 나타나지 않으면 기본 부동 IP 네트워크 생성 및 기본 공급자 네트워크 생성을 참조하십시오.



중요

외부 네트워크의 CIDR 범위가 기본 네트워크 범위 중 하나와 겹치는 경우 설치 프로세스를 시작하기 전에 install-config.yaml 파일에서 일치하는 네트워크 범위를 변경해야 합니다.

기본 네트워크 범위는 다음과 같습니다.

네트워크	범위
machineNetwork	10.0.0.0/16
serviceNetwork	172.30.0.0/16
clusterNetwork	10.128.0.0/14

주의

설치 프로그램이 이름이 같은 여러 네트워크를 발견하면 그 중 하나를 무작위로 설정합니다. 이 동작을 방지하려면 RHOSP에서 리소스의 고유한 이름을 만듭니다.

참고

Neutron 트렁크 서비스 플러그인이 활성화된 경우 기본적으로 트렁크 포트가 생성됩니다. 자세한 내용은 Neutron 트렁크 포트를 참조하십시오.

15.3.7. 설치 프로그램에 대한 매개변수 정의

OpenShift Container Platform 설치 프로그램은 **clouds.yaml** 파일을 사용합니다. 이 파일은 프로젝트 이름, 로그인 정보, 인증 서비스 **URL** 등 **RHOSP(Red Hat OpenStack Platform)** 구성 매개변수를 설명합니다.

프로세스

1. **clouds.yaml** 파일을 만듭니다.

   - **RHOSP** 배포에 **Horizon** 웹 **UI**가 포함되어 있으면 그 안에 **clouds.yaml** 파일을 만듭니다.

     중요

     auth 필드에 암호를 추가해야 합니다. **clouds.yaml**과 별도의 파일에 비밀을 저장할 수도 있습니다.

   - **RHOSP** 배포에 **Horizon** 웹 **UI**가 포함되어 있지 않거나 **Horizon**을 사용하지 않으려면 파일을 직접 만듭니다. **clouds.yaml**에 대한 자세한 내용은 **RHOSP** 문서의 구성 파일을 참조하십시오.

     ```
     clouds:
       shiftstack:
         auth:
           auth_url: http://10.10.14.42:5000/v3
           project_name: shiftstack
           username: shiftstack_user
           password: XXX
           user_domain_name: Default
           project_domain_name: Default
       dev-env:
         region_name: RegionOne
         auth:
           username: 'devuser'
           password: XXX
           project_name: 'devonly'
           auth_url: 'https://10.10.14.22:5001/v2.0'
     ```

2. **RHOSP** 설치에서 끝점 인증을 위해 자체 서명된 **CA**(인증 기관) 인증서를 사용하는 경우:

   a. 인증 기관 파일을 시스템에 복사합니다.

b.

cacerts 키를 **clouds.yaml** 파일에 추가합니다. 값은 **CA** 인증서에 대한 루트가 아닌 액세스 가능한 절대 경로여야 합니다.

```
clouds:
  shiftstack:
    ...
    cacert: "/etc/pki/ca-trust/source/anchors/ca.crt.pem"
```

작은 정보

사용자 지정 **CA** 인증서로 설치 관리자를 실행한 후 **cloud-provider-config** 키맵에서 **ca-cert.pem** 키의 값을 편집하여 인증서를 업데이트할 수 있습니다. 명령줄에서 다음을 실행합니다.

```
$ oc edit configmap -n openshift-config cloud-provider-config
```

3.

clouds.yaml 파일을 다음 위치 중 하나에 배치합니다.

a.

OS_CLIENT_CONFIG_FILE 환경 변수의 값

b.

현재 디렉터리

c.

Unix 전용 사용자 구성 디렉터리(예: **~/.config/openstack/clouds.yaml**)

d.

Unix 전용 사이트 구성 디렉터리(예: **/etc/openstack/clouds.yaml**)

설치 프로그램은 **clouds.yaml**을 이 순서대로 검색합니다.

15.3.8. 클라우드 공급자 옵션 설정

선택적으로 클러스터의 클라우드 공급자 구성을 편집할 수 있습니다. 클라우드 공급자 구성은 **OpenShift Container Platform**이 **RHOSP(Red Hat OpenStack Platform)**와 상호 작용하는 방법을 제어합니다.

클라우드 공급자 구성 매개변수의 전체 목록은 **"OpenStack** 클라우드 구성 참조 가이드**"** 페이지의

"OpenStack 클라우드 구성 참조 가이드" 페이지를 참조하십시오.

프로세스

1. 클러스터에 대해 이미 생성된 매니페스트 파일이 없는 경우 다음 명령을 실행하여 생성합니다.

```
$ openshift-install --dir <destination_directory> create manifests
```

2. 텍스트 편집기에서 cloud-provider 구성 매니페스트 파일을 엽니다. 예를 들면 다음과 같습니다.

```
$ vi openshift/manifests/cloud-provider-config.yaml
```

3. 클라우드 구성 사양에 따라 옵션을 수정합니다.

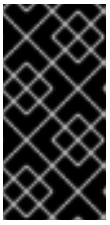
로드 밸런싱을 위해 Octavia를 구성하는 것은 Kuryr를 사용하지 않는 클러스터의 일반적인 사례입니다. 예를 들면 다음과 같습니다.

```
#...
[LoadBalancer]
use-octavia=true (1)
lb-provider = "amphora" (2)
floating-network-id="d3deb660-4190-40a3-91f1-37326fe6ec4a"(3)
#...
```

(1) 이 속성은 Octavia 통합을 활성화합니다.

(2) 이 속성은 로드 밸런서에서 사용하는 Octavia 공급자를 설정합니다. "ovn" 또는 "amphora" 를 값으로 허용합니다. OVN을 사용하도록 선택하는 경우 lb-method 도 SOURCE_IP_PORT 로 설정해야 합니다.

(3) 이 속성은 클러스터에서 여러 외부 네트워크를 사용하려는 경우 필요합니다. 클라우드 프로바이더는 여기에 지정된 네트워크에 유동 IP 주소를 생성합니다.



중요

변경 사항을 저장하기 전에 파일이 올바르게 구성되었는지 확인합니다. 속성을 적절한 섹션에 배치하지 않으면 클러스터가 실패할 수 있습니다.

중요

Kuryr를 사용하는 설치의 경우 Kuryr는 관련 서비스를 처리합니다. 클라우드 공급자에서 Octavia 로드 밸런싱을 구성할 필요가 없습니다.

4. 변경 사항을 파일에 저장하고 설치를 진행합니다.

작은 정보

설치 프로그램을 실행한 후 클라우드 공급자 구성을 업데이트할 수 있습니다. 명령줄에서 다음을 실행합니다.

```
$ oc edit configmap -n openshift-config cloud-provider-config
```

변경 사항을 저장한 후 클러스터를 재구성하는 데 다소 시간이 걸립니다. 노드가 SchedulingDisabled 상태가 없는 경우 프로세스가 완료됩니다.

추가 리소스

- 클라우드 공급자 구성에 대한 자세한 내용은 OpenStack 클라우드 공급자 옵션을 참조하십시오.

15.3.9. 설치 프로그램 받기

OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

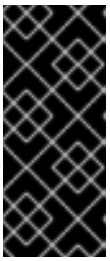
- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

프로세스

1. **OpenShift Cluster Manager** 사이트의 인프라 공급자 페이지에 액세스합니다. **Red Hat** 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.

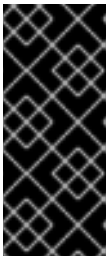
2. 인프라 공급자를 선택합니다.

3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.



중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.



중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 **OpenShift Container Platform** 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 **Linux** 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 **OpenShift Container Platform** 구성 요소에 대한 컨테이너 이미지를 제공하는 **Quay.io**를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

15.3.10. 설치 구성 파일 만들기

RHOSP(Red Hat OpenStack Platform)에 설치하는 **OpenShift Container Platform** 클러스터를 사용자 지정할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.
- 서브스크립션 수준에서 서비스 주체 권한을 획득합니다.

프로세스

1. install-config.yaml 파일을 생성합니다.

   a. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

   ```
   $ ./openshift-install create install-config --dir <installation_directory> 1
   ```

   1 <installation_directory>는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.

   중요

   비어 있는 디렉터리를 지정합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

   b. 화면에 나타나는 지시에 따라 클라우드에 대한 구성 세부 사항을 입력합니다.

      i. 선택사항: 클러스터 시스템에 액세스하는 데 사용할 SSH 키를 선택합니다.

      참고

      설치 디버깅 또는 재해 복구 수행을 원하는 프로덕션 환경 OpenShift Container Platform 클러스터의 경우 ssh-agent 프로세스에 사용할 SSH 키를 지정하십시오.

ii. 대상 플랫폼으로 **openstack**을 선택합니다.

iii. 클러스터 설치에 사용할 **RHOSP(Red Hat OpenStack Platform)** 외부 네트워크 이름을 지정합니다.

iv. **OpenShift API**에 대한 외부 액세스에 사용할 부동 **IP** 주소를 지정합니다.

v. 컨트롤 플레인 노드에 사용할 최소 **16GB**의 **RAM**과 컴퓨팅 노드에 **8GB RAM**이 있는 **RHOSP** 플레이버를 지정합니다.

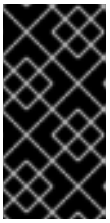
vi. 클러스터를 배포할 기본 도메인을 선택합니다. 모든 **DNS** 레코드는 이 기본 도메인의 하위 도메인이 되므로 클러스터 이름을 포함합니다.

vii. 클러스터 이름을 입력합니다. 이름은 **14**자 이하여야 합니다.

viii. Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 붙여넣습니다.

2. **install-config.yaml** 파일을 수정합니다. 사용 가능한 매개변수에 대한 자세한 정보는 “설치 구성 매개변수” 섹션에서 확인할 수 있습니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.



중요

install-config.yaml 파일은 설치 과정에서 사용됩니다. 이 파일을 재사용하려면 지금 백업해야 합니다.

15.3.10.1. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 **HTTP** 또는 **HTTPS** 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 **OpenShift Container Platform** 클러스터를 구성할 수 있습니다.

참고

Kuryr 설치의 기본값은 HTTP 프록시입니다.

사전 요구 사항

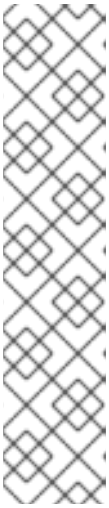
- 프록시 오브젝트를 사용하는 제한된 네트워크에 Kuryr를 설치하려면 프록시가 클러스터가 사용하는 라우터에 응답할 수 있어야 합니다. 프록시 구성에 대한 정적 경로를 추가하려면 root 사용자로 명령줄에서 다음을 입력합니다.

```
$ ip route add <cluster_network_cidr> via <installer_subnet_gateway>
```

- 제한된 서브넷에는 정의된 게이트웨이가 있고 Kuryr에서 생성하는 Router 리소스에 연결할 수 있는 게이트웨이가 있어야 합니다.

- 기존 install-config.yaml 파일이 있습니다.

- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 API에 대한 호출을 포함하여 모든 클러스터 발신(Egress) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 Proxy 오브젝트의 spec.noProxy 필드에 사이트를 추가했습니다.



참고

Proxy 오브젝트의 status.noProxy 필드는 설치 구성에 있는 networking.machineNetwork[].cidr, networking.clusterNetwork[].cidr, networking.serviceNetwork[] 필드의 값으로 채워집니다.

Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 Red Hat OpenStack Platform (RHOSP)에 설치하는 경우 Proxy 오브젝트 status.noProxy 필드도 인스턴스 메타데이터 끝점(169.254.169.254)로 채워집니다.

프로세스

1. install-config.yaml 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
```

```
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1

클러스터 외부에서 **HTTP** 연결을 구축하는 데 사용할 프록시 **URL**입니다. **URL** 스키마는 **http**여야 합니다.

2

클러스터 외부에서 **HTTPS** 연결을 구축하는 데 사용할 프록시 **URL**입니다.

3

대상 도메인 이름, **IP** 주소 또는 프록시에서 제외할 기타 네트워크 **CIDR**로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. ***를 사용하여 모든 대상에 대해 프록시를 바이패스합니다.

4

이 값을 제공하면 설치 프로그램에서 추가 **CA** 인증서를 보유할 **openshift -config** 네임스페이스에 **user-ca- bundle** 이라는 구성 맵을 생성합니다. **additionalTrustBundle** 및 하나 이상의 프록시 설정을 제공하는 경우 프록시 오브젝트는 **trustedCA** 필드의 **user-ca-bundle** 구성 맵을 참조하도록 구성됩니다. 그러면 **Cluster Network Operator**에서 **trustedCA** 매개변수에 대해 지정된 콘텐츠를 **RHCOS** 신뢰 번들과 병합하는 **trusted-ca-bundle** 구성 맵을 생성합니다. 프록시의 **ID** 인증서를 **RHCOS** 트러스트 번들에 있는 기관에서 서명하지 않은 경우 **additionalTrustBundle** 필드가 있어야 합니다.

참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 **OpenShift Container Platform**을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계

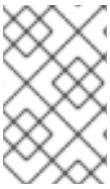
속 생성되지만 **spec**은 **nil**이 됩니다.

참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

15.3.11. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 매개변수 값을 제공하여 클러스터를 호스팅할 클라우드 플랫폼에서 사용자 계정을 설명하고 선택사항으로 클러스터의 플랫폼을 사용자 지정합니다. **install-config.yaml** 설치 구성 파일을 생성할 때 명령줄을 통해 필요한 매개변수 값을 제공합니다. 클러스터를 사용자 지정하면 **install-config.yaml** 파일을 수정하여 플랫폼에 대한 세부 정보를 제공할 수 있습니다.



참고

설치한 후에는 **install-config.yaml** 파일에서 이러한 매개변수를 수정할 수 없습니다.

중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

15.3.11.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 **15.8.** 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열

매개변수	설명	값
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	소문자, 하이픈(**-**), 마침표(**.**)로 구성되는 문자열(예: **dev**)입니다. 문자열은 14자 이하여야 합니다.
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

15.3.11.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 15.9. 네트워크 매개변수

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체  **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.

networking.clusterNetwork	Pod의 IP 주소 블록입니다.
	기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다.
	여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.
		개체의 배열입니다. 예를 들면 다음과 같습니다.
		`networking:`
		`  clusterNetwork:`
		`  - cidr: 10.128.0.0/14`
		`    hostPrefix: 23`

networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된 **cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.

매개변수	설명	값
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16** **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

15.3.11.3. 선택적 구성 매개변수

선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 15.10. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.

매개변수	설명	값
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은**3**입니다.
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열

매개변수	설명	값
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 기본값은 **External**입니다. 이 필드를 **Internal** 로 설정하는 것은 클라우드 이외의 플랫폼에서는 지원되지 않습니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

15.3.11.4. 추가 RHOSP(Red Hat OpenStack Platform) 구성 매개변수

추가 RHOSP 구성 매개변수는 다음 표에 설명되어 있습니다.

표 15.11. 추가 RHOSP 매개변수

매개변수	설명	값
compute.platform.openstack.rootVolume.size	컴퓨팅 시스템의 경우 루트 볼륨의 크기(GB)입니다. 이 값을 설정하지 않으면 시스템은 임시 스토리지를 사용합니다.	정수 (예: **30**)
compute.platform.openstack.rootVolume.type	컴퓨팅 시스템의 경우 루트 볼륨의 유형입니다.	문자열 (예: **performance**)
controlPlane.platform.openstack.rootVolume.size	컨트롤 플레인 시스템의 경우 루트 볼륨의 크기(GB)입니다. 이 값을 설정하지 않으면 시스템은 임시 스토리지를 사용합니다.	정수 (예: **30**)
controlPlane.platform.openstack.rootVolume.type	컨트롤 플레인 시스템의 경우 루트 볼륨의 유형입니다.	문자열 (예: **performance**)
platform.openstack.cloud	**clouds.yaml** 파일의 클라우드 목록에서 사용할 RHOSP 클라우드의 이름입니다.	문자열 (예: **MyCloud**)

매개변수	설명	값
platform.openstack.externalNetwork	설치에 사용할 RHOSP 외부 네트워크 이름입니다.	문자열 (예: **external)**
platform.openstack.computeFlavor	컨트롤 플레인 및 컴퓨팅 시스템에 사용할 RHOSP 버전입니다. 이 속성은 더 이상 사용되지 않습니다. 모든 머신 풀에 플레이버를 기본값으로 사용하려면 **platform.openstack.defaultMachinePlatform** 속성의 **type** 키 값으로 추가합니다. 각 머신 풀의 플레이버 값을 개별적으로 설정할 수도 있습니다.	문자열 (예: **m1.xlarge)**

15.3.11.5. 선택적 RHOSP 구성 매개변수

선택적 RHOSP 구성 매개변수는 다음 표에 설명되어 있습니다.

표 15.12. 선택적 RHOSP 매개변수

매개변수	설명	값
compute.platform.openstack.additionalNetworkIDs	컴퓨팅 시스템과 관련된 추가 네트워크입니다. 추가 네트워크에는 허용되는 주소 쌍이 생성되지 않습니다.	하나 이상의 UUID 목록을 문자열로 나타냅니다. 예: **fa806b2f-ac49-4bce-b9db-124bc64209bf**
compute.platform.openstack.additionalSecurityGroupIDs	컴퓨팅 시스템과 관련된 추가 보안 그룹입니다.	하나 이상의 UUID 목록을 문자열로 나타냅니다. 예: **7ee219f3-d2e9-48a1-96c2-e7429f1b0da7**

매개변수	설명	값
compute.platform.openstack.zones	머신을 설치할 RHOSP Compute (Nova) 가용성 영역 (AZ). 이 매개 변수가 설정되지 않은 경우 설치 프로그램은 RHOSP 관리자가 구성한 Nova의 기본 설정을 사용합니다. Kuryr를 사용하는 클러스터에서 RHOSP Octavia는 가용성 영역을 지원하지 않습니다. 로드 밸런서 및 Amphora 공급자 드라이버를 사용하는 경우 Amphora VM에 의존하는 OpenShift Container Platform 서비스는이 속성의 값에 따라 생성되지 않습니다.	문자열 목록입니다. 예: **["zone-1", "zone-2"]**.
compute.platform.openstack.rootVolume.zones	컴퓨팅 시스템의 경우 루트 볼륨을 설치할 가용성 영역입니다. 이 매개변수의 값을 설정하지 않으면 설치 프로그램에서 기본 가용성 영역을 선택합니다.	문자열 목록(예: **["zone-1", "zone-2"]**).
controlPlane.platform.openstack.additionalNetworkIDs	컨트롤 플레인 시스템과 관련된 추가 네트워크입니다. 추가 네트워크에는 허용되는 주소 쌍이 생성되지 않습니다.	하나 이상의 UUID 목록을 문자열로 나타냅니다. 예: **fa806b2f-ac49-4bce-b9db-124bc64209bf**
controlPlane.platform.openstack.additionalSecurityGroupIDs	컨트롤 플레인 시스템과 관련된 추가 보안 그룹입니다.	하나 이상의 UUID 목록을 문자열로 나타냅니다. 예: **7ee219f3-d2e9-48a1-96c2-e7429f1b0da7**
controlPlane.platform.openstack.zones	머신을 설치할 RHOSP Compute (Nova) 가용성 영역 (AZ). 이 매개 변수가 설정되지 않은 경우 설치 프로그램은 RHOSP 관리자가 구성한 Nova의 기본 설정을 사용합니다. Kuryr를 사용하는 클러스터에서 RHOSP Octavia는 가용성 영역을 지원하지 않습니다. 로드 밸런서 및 Amphora 공급자 드라이버를 사용하는 경우 Amphora VM에 의존하는 OpenShift Container Platform 서비스는이 속성의 값에 따라 생성되지 않습니다.	문자열 목록입니다. 예: **["zone-1", "zone-2"]**.

매개변수	설명	값
controlPlane.platform.openstack.rootVolume.zones	컨트롤 플레인 시스템의 경우 루트 볼륨을 설치할 가용성 영역입니다. 이 값을 설정하지 않으면 설치 프로그램에서 기본 가용성 영역을 선택합니다.	문자열 목록(예: **["zone-1", "zone-2"]**).
platform.openstack.clusterOSImage	설치 프로그램이 RHCOS 이미지를 다운로드하는 위치입니다. 네트워크가 제한된 환경에서 설치를 수행하려면이 매개 변수를 설정해야 합니다.	HTTP 또는 HTTPS URL (선택 옵션으로 SHA-256 형식의 체크섬을 사용) 예: **http://mirror.example.com/images/rhcos-43.81.201912131630.0-openstack.x86_64.qcow2.gz?sha256=ffebbd68e8a1f2a245ca19522c16c86f67f9ac8e4e0c1f0a812b068b16f7265d**. 이 값은 기존의 Glance 이미지의 이름이 될 수도 있습니다 (예: **my-rhcos**).
platform.openstack.clusterOSImageProperties	Glance의 설치 프로그램에서 업로드된 ClusterOSImage에 추가할 속성입니다. **platform.openstack.clusterOSImage**가 기존 Glance 이미지로 설정된 경우 이 속성은 무시됩니다. 이 속성을 사용하여 노드당 26 PV의 RHOSP의 기본 PV(영구 볼륨) 제한을 초과할 수 있습니다. 제한을 초과하려면 **hw_scsi_model** 속성값을 **virtio-scsi**로, **hw_disk_bus** 값을 **scsi**로 설정합니다. 이 속성을 사용하면 **yes** 값이 있는 **hw_qemu_guest_agent** 속성을 포함하여 QEMU 게스트 에이전트를 활성화할 수도 있습니다.	키-값 문자열 쌍 목록입니다. 예를 들면 **["hw_scsi_model": "virtio-scsi", "hw_disk_bus": "scsi"]**가 있습니다.

platform.openstack.defaultMachinePlatform	기본 시스템 풀 플랫폼 구성입니다.	<pre>{
		  "type": "ml.large",
		  "rootVolume": {
		  "size": 30,
		  "type": "performance"
		  }
		}</pre>

매개변수	설명	값
platform.openstack.ingressFloatingIP	Ingress 포트와 연결할 기존 부동 IP 주소입니다. 이 속성을 사용하려면 **platform.openstack.externalNetwork** 속성도 정의해야 합니다.	IP 주소 (예: **128.0.0.1**)
platform.openstack.apiFloatingIP	로드 밸런서 API와 연결할 기존 부동 IP 주소입니다. 이 속성을 사용하려면 **platform.openstack.externalNetwork** 속성도 정의해야 합니다.	IP 주소 (예: **128.0.0.1**)
platform.openstack.externalDNS	클러스터 인스턴스가 DNS 확인에 사용하는 외부 DNS 서버의 IP 주소입니다.	IP 주소 목록을 문자열로 나타냅니다. 예: **["8.8.8.8", "192.168.1.12"]**.
platform.openstack.machinesSubnet	클러스터 노드가 사용하는 RHOSP 서브넷의 UUID입니다. 이 서브넷에 노드와 가상 IP(VIP) 포트가 생성됩니다. **networking.machineNetwork**의 첫 번째 항목은 **machinesSubnet** 값과 일치해야 합니다. 사용자 지정 서브넷에 배포하는 경우 OpenShift Container Platform 설치 관리자에 외부 DNS 서버를 지정할 수 없습니다. 대신 RHOSP의 서브넷에 DNS를 추가합니다.	문자열의 UUID입니다. 예: **fa806b2f-ac49-4bce-b9db-124bc64209bf**

15.3.11.6. RHOSP 배포의 사용자 지정 서브넷

선택한 **RHOSP(Red Hat OpenStack Platform)** 서브넷에 클러스터를 배포할 수도 있습니다. 서브넷의 **GUID**는 **install-config.yaml** 파일에서 **platform.openstack.machinesSubnet** 값으로 전달됩니다.

이 서브넷은 클러스터의 기본 서브넷으로 사용됩니다. 기본적으로 노드와 포트가 이 서브넷에서 생성됩니다. **platform.openstack.machinesSubnet** 속성의 값을 서브넷의 **UUID**로 설정하여 다른 **RHOSP** 서브넷에서 노드 및 포트를 생성할 수 있습니다.

사용자 지정 서브넷으로 **OpenShift Container Platform** 설치 관리자를 실행하기 전에 구성이 다음 요구 사항을 충족하는지 확인하십시오.

- platform.openstack.machinesSubnet에서 사용하는 서브넷에 DHCP가 활성화되어 있습니다.

- platform.openstack.machinesSubnet의 CIDR은 networking.machineNetwork의 CIDR과 일치합니다.

- 설치 프로그램 사용자에게 고정 IP 주소가 있는 포트를 포함하여 이 네트워크에 포트를 생성할 수 있는 권한이 있습니다.

사용자 지정 서브넷을 사용하는 클러스터에는 다음과 같은 제한 사항이 있습니다.

- 유동 IP 주소를 사용하는 클러스터를 설치하려는 경우 platform.openstack.machinesSubnet 서브넷이 externalNetwork 네트워크에 연결된 라우터에 연결되어 있어야 합니다.

- platform.openstack.machinesSubnet 값이 install-config.yaml 파일에 설정된 경우 설치 프로그램에서 RHOSP 머신의 프라이빗 네트워크 또는 서브넷을 생성하지 않습니다.

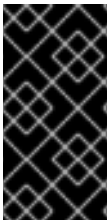
- 사용자 지정 서브넷과 동시에 platform.openstack.externalDNS 속성을 사용할 수 없습니다. 사용자 지정 서브넷을 사용하는 클러스터에 DNS를 추가하려면 RHOSP 네트워크에서 DNS를 구성합니다.

참고

기본적으로 API VIP는 x.x.x.5를 가져오고 Ingress VIP는 네트워크의 CIDR 블록에서 x.x.x.7을 가져옵니다. 이러한 기본값을 재정의하려면 DHCP 할당 풀 외부에 있는platform.openstack.apiVIP 및 platform.openstack.ingressVIP의 값을 설정합니다.

15.3.11.7. Kuryr를 사용한 RHOSP용 샘플 사용자 지정 install-config.yaml 파일

기본 OpenShift SDN 대신 Kuryr SDN으로 배포하려면 Kuryr를 원하는 networking.networkType으로 포함하도록 install-config.yaml 파일을 수정하고 기본 OpenShift Container Platform SDN 설치 단계를 진행해야 합니다. 이 샘플 install-config.yaml은 가능한 모든 RHOSP(Red Hat OpenStack Platform) 사용자 지정 옵션을 보여줍니다.



중요

이 샘플 파일은 참조용으로만 제공됩니다. **install-config.yaml** 파일은 설치 프로그램을 사용하여 받아야 합니다.

```
apiVersion: v1
baseDomain: example.com
controlPlane:
  name: master
  platform: {}
  replicas: 3
compute:
- name: worker
  platform:
    openstack:
      type: ml.large
  replicas: 3
metadata:
  name: example
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  serviceNetwork:
  - 172.30.0.0/16 (1)
  networkType: Kuryr
platform:
  openstack:
    cloud: mycloud
    externalNetwork: external
    computeFlavor: m1.xlarge
    apiFloatingIP: 128.0.0.1
    trunkSupport: true (2)
    octaviaSupport: true (3)
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```

(1)

Amphora Octavia 드라이버는 로드 밸런서당 두 개의 포트를 생성합니다. 결과적으로 설치 프로그램이 생성하는 서비스 서브넷은 **serviceNetwork** 속성 값으로 지정된 **CIDR**의 두 배 크기입니다. **IP** 주소 충돌을 방지하려면 더 큰 범위가 필요합니다.

(2) (3)

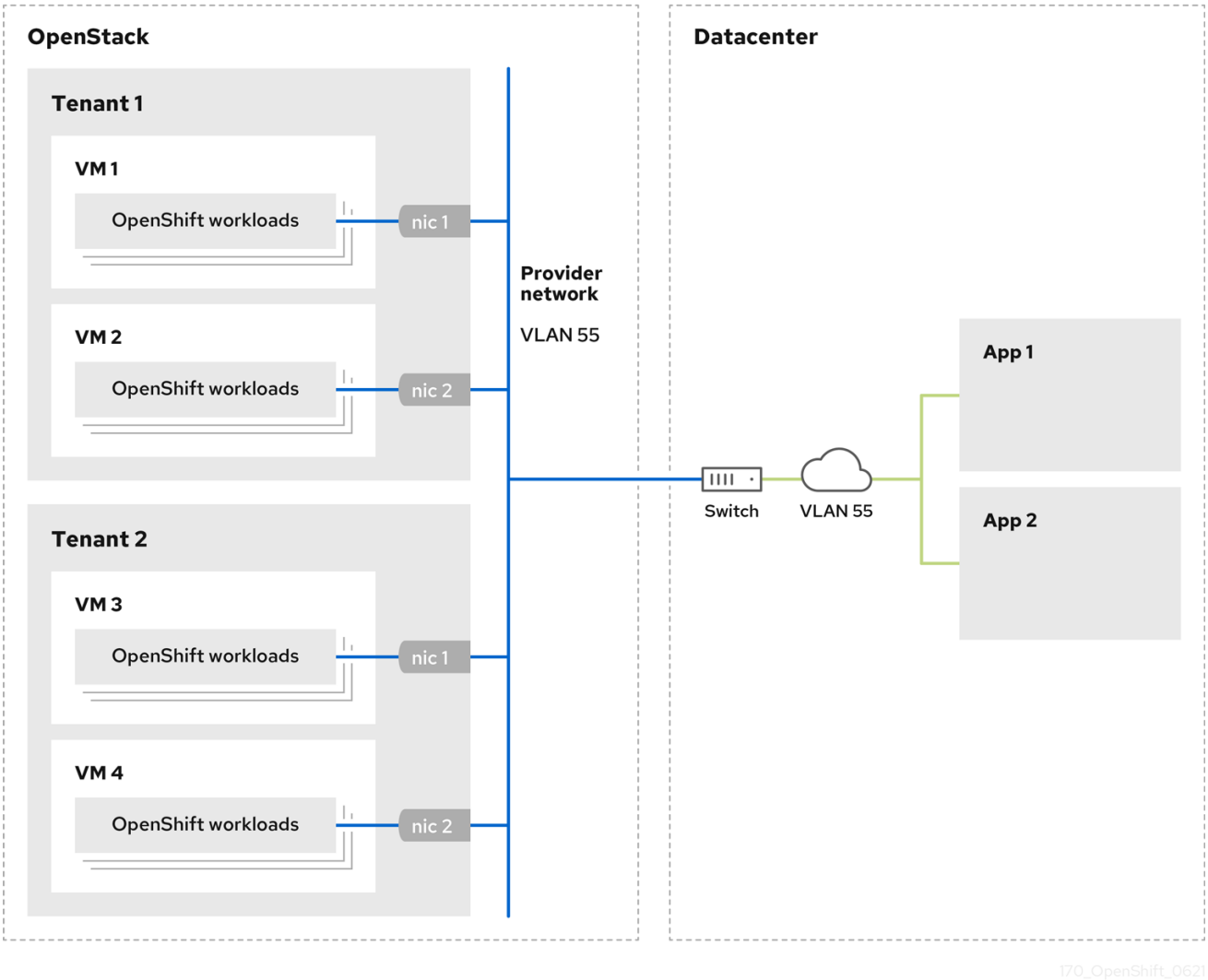
설치 관리자는 **trunkSupport**와 **octaviaSupport**를 자동으로 검색하므로 직접 설정하지 않아도 됩니다. 그러나 사용자 환경이 두 요구사항을 모두 충족하지 않으면 **Kuryr SDN**이 제대로 작동하지 않습니다. **Pod**를 **RHOSP** 네트워크에 연결하려면 트렁크가 필요하며 **OpenShift Container Platform** 서비스를 생성하려면 **Octavia**가 필요합니다.

15.3.11.8. RHOSP 공급자 네트워크에서 클러스터 배포

공급자 네트워크의 기본 네트워크 인터페이스를 사용하여 **RHOSP(Red Hat OpenStack Platform)**에 **OpenShift Container Platform** 클러스터를 배포할 수 있습니다. 공급자 네트워크는 일반적으로 프로젝트에 인터넷에 연결하는 데 사용할 수 있는 공용 네트워크에 직접 액세스할 수 있는 권한을 부여하는 데 사용됩니다. 네트워크 생성 프로세스의 일부로 프로젝트 간에 공급자 네트워크를 공유할 수도 있습니다.

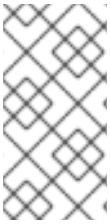
RHOSP 공급자 네트워크는 데이터 센터의 기존 실제 네트워크에 직접 매핑됩니다. **RHOSP** 관리자가 이를 생성해야 합니다.

다음 예에서 **OpenShift Container Platform** 워크로드는 공급자 네트워크를 사용하여 데이터 센터에 연결됩니다.



공급자 네트워크에 설치된 **OpenShift Container Platform** 클러스터에는 테넌트 네트워크 또는 유동 **IP** 주소가 필요하지 않습니다. 설치 프로그램에서 설치하는 동안 이러한 리소스를 생성하지 않습니다.

공급자 네트워크 유형에는 **flat**(태그되지 않음) 및 **VLAN**(802.1Q 태그)이 포함됩니다.



참고

클러스터는 네트워크 유형에서 허용하는 만큼의 공급자 네트워크 연결을 지원할 수 있습니다. 예를 들어 **VLAN** 네트워크는 일반적으로 최대 **4096**개의 연결을 지원합니다.

RHOSP 설명서에서 공급자 및 테넌트 네트워크에 대해 자세히 알아볼 수 있습니다.

15.3.11.8.1. 클러스터 설치를 위한 RHOSP 공급자 네트워크 요구 사항

OpenShift Container Platform 클러스터를 설치하기 전에 **RHOSP**(Red Hat OpenStack Platform) 배포 및 공급자 네트워크가 다음과 같은 여러 조건을 충족해야 합니다.

- RHOSP 네트워킹 서비스(Neutron)가 활성화되어 **RHOSP** 네트워킹 **API**를 통해 액세스할 수 있습니다.
- **RHOSP** 네트워킹 서비스에는 포트 보안 및 허용되는 주소 쌍 확장 기능이 활성화되어 있습니다.
- 공급자 네트워크는 다른 테넌트와 공유할 수 있습니다.

작은 정보

openstack network create 명령을 **--share** 플래그와 함께 사용하여 공유할 수 있는 네트워크를 생성합니다.

- 클러스터를 설치하는 데 사용하는 **RHOSP** 프로젝트는 공급자 네트워크와 적절한 서브넷을 소유해야 합니다.

작은 정보

"openshift"라는 프로젝트의 네트워크를 만들려면 다음 명령을 입력합니다.

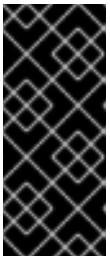
```
$ openstack network create --project openshift
```

"openshift"라는 프로젝트의 서브넷을 만들려면 다음 명령을 입력합니다.

```
$ openstack subnet create --project openshift
```

RHOSP에서 네트워크 생성에 대한 자세한 내용은 공급자 네트워크 설명서를 참조하십시오.

admin 사용자가 클러스터를 소유하는 경우 해당 사용자로 설치 프로그램을 실행하여 네트워크에서 포트를 생성해야 합니다.



중요

공급자 네트워크는 클러스터를 생성하는 데 사용되는 RHOSP 프로젝트에서 소유해야 합니다. RHOSP 컴퓨팅 서비스(Nova)는 해당 네트워크의 포트를 요청할 수 없습니다.

- 공급자 네트워크가 기본적으로 RHOSP 메타데이터 서비스 IP 주소 169.254.169.254에 연결할 수 있는지 확인합니다.

  RHOSP SDN 및 네트워킹 서비스 구성에 따라 서브넷을 생성할 때 경로를 제공해야 할 수 있습니다. 예를 들면 다음과 같습니다.

  ```
  $ openstack subnet create --dhcp --host-route
  destination=169.254.169.254/32,gateway=192.0.2.2 ...
  ```

- 선택 사항: 네트워크를 보호하려면 단일 프로젝트에 대한 네트워크 액세스를 제한하는 역할 기반 액세스 제어(RBAC) 규칙을 생성합니다.

15.3.11.8.2. 공급자 네트워크에 기본 인터페이스가 있는 클러스터 배포

RHOSP(Red Hat OpenStack Platform) 공급자 네트워크에 기본 네트워크 인터페이스가 있는 OpenShift Container Platform 클러스터를 배포할 수 있습니다.

- RHOSP(Red Hat OpenStack Platform) 배포는 "클러스터 설치를 위한 RHOSP 공급자 네트워크 요구 사항"에 설명된 대로 구성됩니다.

프로세스

1. 텍스트 편집기에서 install-config.yaml 파일을 엽니다.
2. platform.openstack.apiVIP 속성의 값을 API VIP의 IP 주소로 설정합니다.
3. platform.openstack.ingressVIP 속성 값을 Ingress VIP의 IP 주소로 설정합니다.
4. platform.openstack.machinesSubnet 속성 값을 공급자 네트워크 서브넷의 UUID로 설정합니다.
5. networking.machineNetwork.cidr 속성 값을 공급자 네트워크 서브넷의 CIDR 블록으로 설정합니다.

중요

platform.openstack.apiVIP 및 platform.openstack.ingressVIP 속성은 모두 networking.machineNetwork.cidr 블록에서 할당되지 않은 IP 주소여야 합니다.

RHOSP 공급자 네트워크를 사용하는 클러스터의 설치 구성 파일 섹션

```
...
platform:
  openstack:
    apiVIP: 192.0.2.13
    ingressVIP: 192.0.2.23
    machinesSubnet: fa806b2f-ac49-4bce-b9db-124bc64209bf
    # ...
networking:
  machineNetwork:
  - cidr: 192.0.2.0/24
```

주의

기본 네트워크 인터페이스에 공급자 네트워크를 사용하는 동안 **platform.openstack.externalNetwork** 또는 **platform.openstack.externalDNS** 매개변수를 설정할 수 없습니다.

클러스터를 배포할 때 설치 프로그램에서 **install-config.yaml** 파일을 사용하여 공급자 네트워크에 클러스터를 배포합니다.

작은 정보

공급자 네트워크를 포함한 다른 네트워크를 **platform.openstack.additionalNetworkIDs** 목록에 추가할 수 있습니다.

클러스터를 배포한 후 **Pod**를 추가 네트워크에 연결할 수 있습니다. 자세한 내용은 여러 네트워크 이해를 참조하십시오.

15.3.11.9. Kuryr 포트 풀

Kuryr 포트 풀은 **Pod** 생성을 위해 대기 중인 다수의 포트를 유지 관리합니다.

포트를 대기 상태로 유지하면 **Pod** 생성 시간이 최소화됩니다. 포트 풀이 없으면 **Kuryr**는 **Pod**를 생성하거나 삭제할 때마다 포트 생성 또는 삭제를 명시적으로 요청해야 합니다.

Kuryr가 사용하는 **Neutron** 포트는 네임스페이스에 연결된 서브넷에 생성됩니다. 이러한 **Pod** 포트도 **OpenShift Container Platform** 클러스터 노드의 기본 포트에 하위 포트로 추가됩니다.

Kuryr는 각 네임스페이스를 별도의 서브넷에 유지하므로 각 네임스페이스-작업자 쌍에 대해 별도의 포트 풀이 유지됩니다.

클러스터를 설치하기 전에 **cluster-network-03-config.yml** 매니페스트 파일에서 다음 매개변수를 설정하여 포트 풀 동작을 구성할 수 있습니다.

- enablePortPoolsPrepopulation 매개변수는 풀 사전 채우기를 제어하므로 새 호스트가 추가되거나 새 네임스페이스가 생성되는 경우와 같이 풀 생성 시 Kuryr가 풀에 포트를 추가하도록 합니다. 기본값은 false입니다.

- poolMinPorts 매개변수는 풀에 보관되는 사용 가능한 최소 포트 수입니다. 기본값은 1 입니다.

- poolMaxPorts 매개변수는 풀에 보관되는 사용 가능한 최대 포트 수입니다. 값이 0이면 해당 상한이 비활성화됩니다. 이 설정은 기본 설정입니다.

  OpenStack 포트 할당량이 낮거나 pod 네트워크에 IP 주소가 제한된 경우 이 옵션을 설정하여 불필요한 포트가 삭제되었는지 확인합니다.

- poolBatchPorts 매개 변수는 한 번에 생성할 수 있는 최대 Neutron 포트 수를 정의합니다. 기본값은 3입니다.

15.3.11.10. 설치 중에 Kuryr 포트 풀 조정

설치하는 동안 Kuryr가 RHOSP(Red Hat OpenStack Platform) Neutron 포트를 관리하여 Pod 생성 속도와 효율성을 제어하는 방법을 구성할 수 있습니다.

사전 요구 사항

- install-config.yaml 파일을 생성하고 수정합니다.

프로세스

1. 명령줄에서 매니페스트 파일을 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 <installation_directory>는 클러스터의 install-config.yaml 파일이 포함된 디렉터리의 이름을 지정합니다.

2. <installation_directory>/ manifests/ 디렉토리에 cluster-network-03-config.yml이라는

이름으로 파일을 만듭니다.

```
$ touch <installation_directory>/manifests/cluster-network-03-config.yml 1
```

1

<installation_directory>는 클러스터의 manifests / 디렉터리가 포함된 디렉터리 이름을 지정합니다.

파일이 생성되면 다음과 같이 여러 네트워크 구성 파일이 manifests/ 디렉토리에 나타납니다.

```
$ ls <installation_directory>/manifests/cluster-network-*
```

출력 예

```
cluster-network-01-crd.yml
cluster-network-02-config.yml
cluster-network-03-config.yml
```

3. 편집기에서 cluster-network-03-config.yml 파일을 열고 원하는 Cluster Network Operator 구성을 설명하는 CR(사용자 정의 리소스)을 입력합니다.

```
$ oc edit networks.operator.openshift.io cluster
```

4. 요구 사항에 맞게 설정을 편집합니다. 다음 파일은 예제로 제공됩니다.

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  serviceNetwork:
  - 172.30.0.0/16
  defaultNetwork:
    type: Kuryr
```

```
kuryrConfig:
  enablePortPoolsPrepopulation: false 1
  poolMinPorts: 1 2
  poolBatchPorts: 3 3
  poolMaxPorts: 5 4
  openstackServiceNetwork: 172.30.0.0/15 5
```

1

Kuryr가 네임스페이스를 생성하거나 클러스터에 새 노드를 추가한 후 Kuryr가 새 Neutron 포트를 생성하도록 하려면 enablePortPoolsPrepopulation 값을 true로 설정합니다. 이 설정은 Neutron 포트 할당량을 높이지만 pod를 생성하는 데 필요한 시간을 줄일 수 있습니다. 기본값은 false입니다.

2

Kuryr는 풀의 사용 가능한 포트 수가 poolMinPorts 값보다 낮은 경우 풀에 대한 새 포트를 만듭니다. 기본값은 1 입니다.

3

poolBatchPorts는 사용 가능한 포트 수가 poolMinPorts 값보다 낮은 경우 생성되는 새 포트 수를 제어합니다. 기본값은 3입니다.

4

풀에서 사용 가능한 포트 수가 poolMaxPorts 값보다 크면 Kuryr는 숫자가 해당 값과 일치할 때까지 해당 포트를 삭제합니다. 이 값을 0으로 설정하면 이 상한이 비활성화되므로 풀이 축소되지 않습니다. 기본값은 0입니다.

5

openStackServiceNetwork 매개변수는 IP 주소가 RHOSP Octavia의 LoadBalancers에 할당되는 네트워크의 CIDR 범위를 정의합니다.

이 매개변수가 Amphora 드라이버와 함께 사용되는 경우 Octavia는 각 로드 밸런서에 대해 이 네트워크에서 두 개의 IP 주소를 사용합니다. 하나는 OpenShift 및 VRRP 연결에 사용됩니다. 이러한 IP 주소는 각각 OpenShift Container Platform 및 Neutron에서 관리하므로 서로 다른 풀에서 가져와야 합니다. 따라서 openStackServiceNetwork의 값은 serviceNetwork 값의 2배 이상이어야 하며 serviceNetwork의 값은 openStackServiceNetwork에서 정의된 범위와 완전히 겹쳐야 합니다.

CNO는 이 매개변수에서 정의한 범위에서 가져온 VRRP IP 주소가 serviceNetwork 매개변수에 정의된 범위와 겹치지 않음을 확인합니다.

이 매개변수가 설정되지 않은 경우 CNO는 접두사 크기를 1로 줄임으로써 결정되는 serviceNetwork의 확장된 값을 사용합니다.

5. cluster-network-03-config.yml 파일을 저장하고 텍스트 편집기를 종료합니다.

6. 선택사항: manifests/cluster-network-03-config.yml 파일을 백업합니다. 설치 프로그램은 클러스터를 생성하는 동안 manifests/ 디렉터리를 삭제합니다.

15.3.12. 컴퓨팅 머신 유사성 설정

선택적으로 설치하는 동안 컴퓨팅 시스템의 선호도 정책을 설정할 수 있습니다. 설치 프로그램은 기본적으로 컴퓨팅 시스템의 선호도 정책을 선택하지 않습니다.

설치 후 특정 RHOSP 서버 그룹을 사용하는 머신 세트를 생성할 수도 있습니다.

참고

컨트롤 플레인 시스템은 soft-anti-affinity 정책을 사용하여 생성됩니다.

작은 정보

RHOSP 인스턴스 스케줄링 및 배치에 대한 자세한 내용은 RHOSP 설명서에서 확인할 수 있습니다.

사전 요구 사항

- install-config.yaml 파일을 생성하고 수정합니다.

프로세스

1. RHOSP 명령줄 인터페이스를 사용하여 컴퓨팅 시스템의 서버 그룹을 생성합니다. 예를 들면 다음과 같습니다.

```
$ openstack \
  --os-compute-api-version=2.15 \
  server group create \
  --policy anti-affinity \
  my-openshift-worker-group
```

자세한 내용은 서버 그룹 생성 명령 설명서를 참조하십시오.

2. 설치 프로그램이 포함된 디렉터리로 변경하고 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory>
```

다음과 같습니다.

installation_directory

클러스터의 **install-config.yaml** 파일이 포함된 디렉토리의 이름을 지정합니다.

3. **MachineSet** 정의 파일인 **manifests/99_openshift-cluster-api_worker-machineset-0.yaml** 을 엽니다.

4. **spec.template.spec.providerSpec.value** 속성 아래에 있는 정의에 속성 **serverGroupID** 를 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_ID>
    machine.openshift.io/cluster-api-machine-role: <node_role>
    machine.openshift.io/cluster-api-machine-type: <node_role>
  name: <infrastructure_ID>-<node_role>
  namespace: openshift-machine-api
spec:
  replicas: <number_of_replicas>
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_ID>
      machine.openshift.io/cluster-api-machineset: <infrastructure_ID>-<node_role>
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_ID>
        machine.openshift.io/cluster-api-machine-role: <node_role>
        machine.openshift.io/cluster-api-machine-type: <node_role>
        machine.openshift.io/cluster-api-machineset: <infrastructure_ID>-<node_role>
    spec:
      providerSpec:
        value:
          apiVersion: openstackproviderconfig.openshift.io/v1alpha1
          cloudName: openstack
```

```
      cloudsSecret:
        name: openstack-cloud-credentials
        namespace: openshift-machine-api
      flavor: <nova_flavor>
      image: <glance_image_name_or_location>
      serverGroupID: aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee (1)
      kind: OpenstackProviderSpec
      networks:
      - filter: {}
        subnets:
        - filter:
            name: <subnet_name>
            tags: openshiftClusterID=<infrastructure_ID>
      securityGroups:
      - filter: {}
        name: <infrastructure_ID>-<node_role>
      serverMetadata:
        Name: <infrastructure_ID>-<node_role>
        openshiftClusterID: <infrastructure_ID>
      tags:
      - openshiftClusterID=<infrastructure_ID>
      trunk: true
      userDataSecret:
        name: <node_role>-user-data
      availabilityZone: <optional_openstack_availability_zone>
```

(1) 서버 그룹의 UUID를 여기에 추가합니다.

5. 선택 사항: manifests/99_openshift-cluster-api_worker-machineset-0.yaml 파일을 백업합니다. 설치 프로그램은 클러스터를 생성할 때 manifests/ 디렉터리를 삭제합니다.

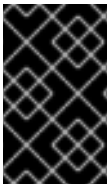
클러스터를 설치할 때 설치 프로그램은 수정한 MachineSet 정의를 사용하여 RHOSP 서버 그룹 내에서 컴퓨팅 머신을 생성합니다.

15.3.13. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 core 사용자의 ~/.ssh/authorized_keys 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 core로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. ./openshift-install gather 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.



중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: ~/.ssh/id_ed25519)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 '~/.ssh 디렉터리에 있는지 확인하십시오.

참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 ed25519 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 rsa 또는 ecdsa 알고리즘을 사용하는 키를 생성합니다.

2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 ~/.ssh/id_ed25519.pub 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 ./openshift-install

gather 명령을 사용하려는 경우 필요합니다.

참고

일부 배포에서는 ~/.ssh/id_rsa 및 ~/.ssh/id_dsa와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

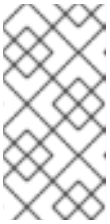
a.

ssh-agent 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```



참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4.

ssh-agent에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1

SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: ~/.ssh/id_ed25519).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

15.3.14. 환경에 대한 액세스 활성화

배포 시 모든 OpenShift Container Platform 시스템은 RHOSP(Red Hat OpenStack Platform) 테넌트 네트워크에서 생성됩니다. 따라서 대부분의 RHOSP 배포에서 직접 액세스할 수 없습니다.

설치시 부동 IP 주소 (FIP)를 사용하여 OpenShift Container Platform API 및 애플리케이션의 액세스를 설정할 수 있습니다. FIP를 구성하지 않고 설치를 완료 할 수도 있지만 설치 프로그램은 외부에서 API 또는 애플리케이션에 액세스하는 방법을 설정하지 않습니다.

15.3.14.1. 부동 IP 주소로 액세스 활성화

OpenShift Container Platform API 및 클러스터 애플리케이션에 대한 외부 액세스 용으로 유동 IP (FIP) 주소를 생성합니다.

프로세스

1. RHOSP(Red Hat OpenStack Platform) CLI를 사용하여 API FIP를 생성합니다.

```
$ openstack floating ip create --description "API <cluster_name>.<base_domain>"
<external_network>
```

2. RHOSP(Red Hat OpenStack Platform) CLI를 사용하여 앱 또는 Ingress, FIP를 생성합니다.

```
$ openstack floating ip create --description "Ingress <cluster_name>.<base_domain>"
<external_network>
```

3. API 및 Ingress FIP의 DNS 서버에 이러한 패턴에 따라 레코드를 추가합니다.

```
api.<cluster_name>.<base_domain>.  IN  A  <API_FIP>
*.apps.<cluster_name>.<base_domain>. IN  A <apps_FIP>
```

참고

DNS 서버를 제어하지 않으면 /etc/hosts 파일에 다음과 같은 클러스터 도메인 이름을 추가하여 클러스터에 액세스할 수 있습니다.

- <api_floating_ip> api.<cluster_name>.<base_domain>
- <application_floating_ip> grafana-openshift-monitoring.apps.<cluster_name>.<base_domain>
- <application_floating_ip> prometheus-k8s-openshift-monitoring.apps.<cluster_name>.<base_domain>
- <application_floating_ip> oauth-openshift.apps.<cluster_name>.<base_domain>
- <application_floating_ip> console-openshift-console.apps.<cluster_name>.<base_domain>
- application_floating_ip integrated-oauth-server-openshift-authentication.apps.<cluster_name>.<base_domain>

/etc/hosts 파일의 클러스터 도메인 이름은 웹 콘솔과 클러스터의 모니터링 인터페이스에 로컬로 액세스할 수 있는 권한을 부여합니다. kubectl 또는 oc 에서도 사용할 수 있습니다. <application_floating_ip>를 가리키는 추가 항목을 사용하여 사용자 애플리케이션에 액세스할 수 있습니다. 이 작업을 수행하면 사용자만 API 및 애플리케이션에 액세스할 수 있어 프로덕션 배포에는 적합하지 않지만 개발 및 테스트용 설치는 가능합니다.

4. FIP를 다음 매개 변수의 값으로 install-config.yaml 파일에 추가하십시오.
   - platform.openstack.ingressFloatingIP
   - platform.openstack.apiFloatingIP

이러한 값을 사용하는 경우 **install-config.yaml** 파일에서 **platform.openstack.externalNetwork** 매개 변수 값으로 외부 네트워크를 입력해야 합니다.

작은 정보

부동 **IP** 주소를 할당하고 방화벽 구성을 업데이트하여 클러스터 외부에서 **OpenShift Container Platform** 리소스를 사용할 수 있습니다.

15.3.14.2. 유동 IP 주소없이 설치 완료

유동 **IP** 주소를 지정하지 않고도 **Red Hat OpenStack Platform (RHOSP)**에 **OpenShift Container Platform**을 설치할 수 있습니다.

install-config.yaml 파일에서 다음 매개 변수를 정의하지 마십시오.

- **platform.openstack.ingressFloatingIP**
- **platform.openstack.apiFloatingIP**

외부 네트워크를 제공 할 수 없는 경우 **platform.openstack.externalNetwork**를 비워 둘 수도 있습니다. **platform.openstack.externalNetwork** 값을 지정하지 않으면 라우터가 생성되지 않으며 추가 작업없이 설치 프로그램이 **Glance**에서 이미지를 검색하지 못합니다. 외부 연결을 직접 구성해야 합니다.

유동 **IP** 주소 또는 이름 확인 부족으로 인해 클러스터 **API**에 연결할 수 없는 시스템에서 설치 프로그램을 실행하면 설치에 실패합니다. 이러한 경우 설치 실패를 방지하기 위해 프록시 네트워크를 사용하거나 시스템과 동일한 네트워크에있는 시스템에서 설치 프로그램을 실행할 수 있습니다.

참고

API 및 Ingress 포트의 DNS 레코드를 생성하여 이름 확인을 활성화할 수 있습니다. 예를 들면 다음과 같습니다.

```
api.<cluster_name>.<base_domain>.  IN  A  <api_port_IP>
*.apps.<cluster_name>.<base_domain>. IN  A <ingress_port_IP>
```

DNS 서버를 제어하지 않으면 /etc/hosts 파일에 레코드를 추가할 수 있습니다. 이 작업을 수행하면 사용자만 API에 액세스할 수 있어 프로덕션 배포에는 적합하지 않지만 개발 및 테스트용 설치는 가능합니다.

15.3.15. 클러스터 배포

호환되는 클라우드 플랫폼에 OpenShift Container Platform을 설치할 수 있습니다.

중요

최초 설치 과정에서 설치 프로그램의 create cluster 명령을 한 번만 실행할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

프로세스

1. 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터 배포를 초기화합니다.

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
  --log-level=info 2
```

1 <installation_directory> 값으로 사용자 지정한 ./install-config.yaml 파일의 위치를 지정합니다.

2 다른 설치 세부 사항을 보려면 info 대신 warn, debug 또는 error를 지정합니다.

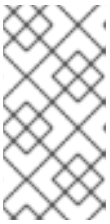
참고

호스트에 구성된 클라우드 공급자 계정에 클러스터를 배포하기에 충분한 권한이 없는 경우, 설치 프로세스가 중단되고 누락된 권한을 알리는 메시지가 표시됩니다.

클러스터 배포가 완료되면 웹 콘솔로 연결되는 링크와 **kubeadmin** 사용자의 인증 정보가 포함된 클러스터 액세스 지침이 사용자 터미널에 표시됩니다.

출력 예

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-
ymBZj-Wt5AL"
INFO Time elapsed: 36m22s
```



참고

설치에 성공하면 클러스터 액세스 및 인증 정보도 **<installation_directory>/.openshift_install.log**로 출력됩니다.

중요

- 설치 프로그램에서 생성하는 **Ignition** 구성 파일에 **24**시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 **24**시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 **kubelet** 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(**CSR**)을 수동으로 승인해야 합니다. 자세한 내용은 *만료된 컨트롤 플레인 인증서에서 복구* 문서를 참조하십시오.

- 클러스터를 설치한 후 **24**시간에서 **22**시간까지의 인증서가 교체되기 때문에 생성된 후 **12**시간 이내에 **Ignition** 구성 파일을 사용하는 것이 좋습니다. **12**시간 이내에 **Ignition** 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

중요

설치 프로그램에서 생성되는 파일이나 설치 프로그램을 삭제해서는 안 됩니다. 클러스터를 삭제하려면 두 가지가 모두 필요합니다.

15.3.16. 클러스터 상태 확인

설치 중 또는 설치 후 **OpenShift Container Platform** 클러스터의 상태를 확인할 수 있습니다.

프로세스

1. 클러스터 환경에서 관리자의 **kubeconfig** 파일을 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

kubeconfig 파일에는 **CLI**에서 올바른 클러스터 및 **API** 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다.

2. 배포 후 생성된 컨트롤 플레인 및 컴퓨팅 시스템을 확인합니다.

```
$ oc get nodes
```

3. 클러스터 버전을 확인합니다.

```
$ oc get clusterversion
```

4. Operator 상태를 확인합니다.

```
$ oc get clusteroperator
```

5. 클러스터에서 실행 중인 모든 Pod를 확인합니다.

```
$ oc get pods -A
```

15.3.17. CLI를 사용하여 클러스터에 로그인

클러스터 kubeconfig 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. kubeconfig 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- oc CLI를 설치했습니다.

프로세스

1. kubeadmin 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1. <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 **oc** 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

추가 리소스

- **OpenShift Container Platform** 웹 콘솔 액세스 및 이해에 대한 자세한 내용은 웹 콘솔에 액세스를 참조하십시오.

15.3.18. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 **Telemetry** 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 **Telemetry**가 자동으로 실행되고 **OpenShift Cluster Manager** 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 **OpenShift Cluster Manager**를 사용하여 자동으로 또는 **OpenShift Cluster Manager**를 사용하여 수동으로 유지 관리되는지 확인한 후 **subscription watch**를 사용하여 계정 또는 다중 클러스터 수준에서 **OpenShift Container Platform** 서브스크립션을 추적합니다.

추가 리소스

- **Telemetry** 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

15.3.19. 다음 단계

- 클러스터를 사용자 지정합니다.

- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.

- 노드 포트에 대한 외부 액세스를 활성화해야 하는 경우 노드 포트를 사용하여 Ingress 클러스터 트래픽을 구성합니다.

- 유동 IP 주소를 통한 애플리케이션 트래픽을 허용하도록 RHOSP를 구성하지 않은 경우 유동 IP 주소로 RHOSP 액세스를 구성합니다.

15.4. SR-IOV 연결 컴퓨팅 머신을 지원하는 OPENSTACK에 클러스터 설치

OpenShift Container Platform 버전 4.9에서는 RHOSP(Red Hat OpenStack Platform)에 SR-IOV(Single-root I/O Virtualization) 기술로 컴퓨팅 머신을 사용할 수 있는 클러스터를 설치할 수 있습니다.

15.4.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토하십시오.

  - OpenShift Container Platform 4.9가 "OpenShift 클러스터 용으로 지원되는 플랫폼" 섹션을 사용하여 현재 RHOSP 버전과 호환되는지 확인합니다. RHOSP 지원 매트릭스의 OpenShift Container Platform을 확인하여 여러 버전의 플랫폼 지원을 비교할 수도 있습니다.

- 네트워크 구성이 공급자 네트워크를 사용하지 않는지 확인합니다. 공급자 네트워크는 지원되지 않습니다.

- RHOSP에 블록 스토리지(Cinder) 또는 개체 스토리지(Swift)와 같은 스토리지 서비스가 설치되어 있어야 합니다. 개체 스토리지는 OpenShift Container Platform 레지스트리 클러스터 배포에 권장되는 스토리지 기술입니다. 자세한 정보는 스토리지 최적화를 참조하십시오.

- RHOSP에서 메타데이터 서비스 활성화

15.4.2. RHOSP에 OpenShift Container Platform을 설치하기 위한 리소스 지침

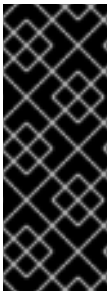
OpenShift Container Platform 설치를 지원하려면 RHOSP(Red Hat OpenStack Platform) 할당량

이 다음 요구사항을 충족해야 합니다.

표 15.13. RHOSP의 기본 OpenShift Container Platform 클러스터에 권장되는 리소스

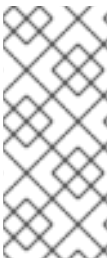
리소스 이름	값
부동 IP 주소	3
포트	15
라우터	1
서브넷	1
RAM	ECDHEGB
vCPU	22
볼륨 스토리지	275GB
인스턴스	7
보안 그룹	3
보안 그룹 규칙	60

권장 리소스보다 적은 리소스로도 클러스터가 작동할 수 있지만 성능은 보장되지 않습니다.



중요

swiftoperator 역할을 가진 사용자 계정으로 **RHOSP** 개체 스토리지(**Swift**)를 사용하고 운영하는 경우 **OpenShift Container Platform** 이미지 레지스트리의 기본 백엔드로 사용됩니다. 이 경우 볼륨 스토리지 요구사항은 **175GB**입니다. **Swift** 공간 요구사항은 이미지 레지스트리의 크기에 따라 다릅니다.



참고

기본적으로 보안 그룹 및 보안 그룹 규칙 할당량이 적을 수 있습니다. 문제가 발생하면 관리자로 **openstack quota set --secgroups 3 --secgroup-rules 60 <project>**를 실행하여 할당량을 늘립니다.

OpenShift Container Platform 배포는 컨트롤 플레인 시스템, 컴퓨팅 시스템, 부트스트랩 시스템으로

구성됩니다.

15.4.2.1. 컨트롤 플레인 머신

기본적으로 OpenShift Container Platform 설치 프로세스는 세 개의 컨트롤 플레인 시스템을 생성합니다.

각 시스템의 요구사항은 다음과 같습니다.

- RHOSP 할당량의 인스턴스
- RHOSP 할당량의 포트
- 최소 16GB 메모리 및 vCPU 4개가 있는 플레이버
- RHOSP 할당량에서 최소 100GB 스토리지 공간

15.4.2.2. 컴퓨팅 머신

기본적으로 OpenShift Container Platform 설치 프로세스는 세 개의 컴퓨팅 시스템을 생성합니다.

각 시스템의 요구사항은 다음과 같습니다.

- RHOSP 할당량의 인스턴스
- RHOSP 할당량의 포트
- 8GB 메모리 및 vCPU 2개 이상 있는 플레이버
- RHOSP 할당량에서 최소 100GB 스토리지 공간

작은 정보

컴퓨팅 시스템은 **OpenShift Container Platform**에서 실행하는 애플리케이션을 호스팅합니다. 최대한 많이 실행하는 것이 좋습니다.

또한 **SR-IOV(Single-root input/output virtualization)**를 사용하는 클러스터의 경우 **RHOSP** 컴퓨팅 노드에 대규모 페이지를 지원하는 플레이버가 필요합니다.

중요

SR-IOV 배포는 종종 전용 또는 분리된 **CPU**와 같은 성능 최적화를 사용합니다. 성능을 극대화하려면 이러한 최적화를 사용하도록 기본 **RHOSP** 배포를 구성한 다음 최적화된 인프라에서 **OpenShift Container Platform** 컴퓨팅 머신을 실행합니다.

추가 리소스

- 성능 강화를 위한 **RHOSP** 컴퓨팅 노드 구성에 대한 자세한 내용은 성능 개선을 위한 컴퓨팅 노드 구성을 참조하십시오.

15.4.2.3. 부트스트랩 시스템

설치하는 동안 컨트롤 플레인을 유지하기 위해 부트스트랩 시스템이 임시로 프로비저닝됩니다. 프로덕션 컨트롤 플레인이 준비되면 부트스트랩 시스템이 프로비저닝 해제됩니다.

부트스트랩 시스템의 요구사항은 다음과 같습니다.

- **RHOSP** 할당량의 인스턴스
- **RHOSP** 할당량의 포트
- 최소 **16GB** 메모리 및 **vCPU 4**개가 있는 플레이버
- **RHOSP** 할당량에서 최소 **100GB** 스토리지 공간

15.4.3. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

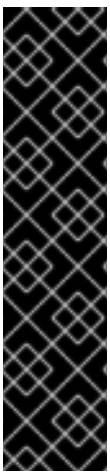
- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.

중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

15.4.4. RHOSP에서 Swift 활성화

Swift는 swiftoperator 역할을 가진 사용자 계정으로 운영됩니다. 설치 프로그램을 실행하려면 먼저 계정에 이 역할을 추가합니다.



중요

RHOSP(Red Hat OpenStack Platform) 개체 스토리지 서비스(일반적으로 Swift로 알려짐)를 사용할 수 있는 경우 OpenShift Container Platform이 이미지 레지스트리 스토리지로 사용합니다. 이 서비스를 사용할 수 없는 경우에는 설치 프로그램이 RHOSP 블록 스토리지 서비스(일반적으로 Cinder로 알려짐)를 사용합니다.

Swift가 있고 Swift를 사용하려면 액세스를 활성화해야 합니다. 존재하지 않거나 사용하지 않으려면 이 섹션을 건너 뛰십시오.

사전 요구 사항

- 대상 환경에 RHOSP 관리자 계정이 있습니다.
- Swift 서비스가 설치되어 있습니다.
- Ceph RGW에서 url의 계정 옵션이 활성화되어 있습니다.

프로세스

RHOSP에서 Swift를 활성화하려면:

1. RHOSP CLI의 관리자로서 Swift에 액세스할 계정에 swiftoperator 역할을 추가하십시오.

```
$ openstack role add --user <user> --project <project> swiftoperator
```

이제 RHOSP 배포에서 이미지 레지스트리에 Swift를 사용할 수 있습니다.

15.4.5. 외부 네트워크 액세스 확인

OpenShift Container Platform 설치 프로세스에는 외부 네트워크에 액세스해야 합니다. 외부 네트워크값을 제공해야 하며 그렇지 않으면 배포가 실패합니다. 프로세스를 시작하기 전에 외부 라우터 유형의 네트워크가 RHOSP(Red Hat OpenStack Platform)에 있는지 확인합니다.

사전 요구 사항

- DHCP 에이전트가 인스턴스의 DNS 쿼리를 전달하도록 OpenStack의 네트워킹 서비스 구성

프로세스

1. RHOSP CLI를 사용하여 '외부' 네트워크의 이름과 ID를 확인합니다.

```
$ openstack network list --long -c ID -c Name -c "Router Type"
```

출력 예

```
+--------------------------------------+----------------+-------------+
| ID                                   | Name           | Router Type |
+--------------------------------------+----------------+-------------+
| 148a8023-62a7-4672-b018-003462f8d7dc | public_network | External    |
+--------------------------------------+----------------+-------------+
```

외부 라우터 유형의 네트워크가 네트워크 목록에 나타납니다. 네트워크가 하나 이상 나타나지 않으면 기본 부동 IP 네트워크 생성 및 기본 공급자 네트워크 생성을 참조하십시오.

참고

Neutron 트렁크 서비스 플러그인이 활성화된 경우 기본적으로 트렁크 포트가 생성됩니다. 자세한 내용은 Neutron 트렁크 포트를 참조하십시오.

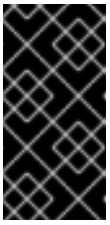
15.4.6. 설치 프로그램에 대한 매개변수 정의

OpenShift Container Platform 설치 프로그램은 clouds.yaml 파일을 사용합니다. 이 파일은 프로젝트 이름, 로그인 정보, 인증 서비스 URL 등 RHOSP(Red Hat OpenStack Platform) 구성 매개변수를 설명합니다.

프로세스

1. clouds.yaml 파일을 만듭니다.

- RHOSP 배포에 Horizon 웹 UI가 포함되어 있으면 그 안에 clouds.yaml 파일을 만듭니다.



중요

auth 필드에 암호를 추가해야 합니다. clouds.yaml과 별도의 파일에 비밀을 저장할 수도 있습니다.

- RHOSP 배포에 Horizon 웹 UI가 포함되어 있지 않거나 Horizon을 사용하지 않으려면 파일을 직접 만듭니다. clouds.yaml에 대한 자세한 내용은 RHOSP 문서의 구성 파일을 참조하십시오.

```
clouds:
  shiftstack:
    auth:
      auth_url: http://10.10.14.42:5000/v3
      project_name: shiftstack
      username: shiftstack_user
      password: XXX
      user_domain_name: Default
      project_domain_name: Default
  dev-env:
    region_name: RegionOne
    auth:
      username: 'devuser'
      password: XXX
      project_name: 'devonly'
      auth_url: 'https://10.10.14.22:5001/v2.0'
```

2. **RHOSP** 설치에서 끝점 인증을 위해 자체 서명된 **CA**(인증 기관) 인증서를 사용하는 경우:

   a. 인증 기관 파일을 시스템에 복사합니다.

   b. **cacerts** 키를 **clouds.yaml** 파일에 추가합니다. 값은 **CA** 인증서에 대한 루트가 아닌 액세스 가능한 절대 경로여야 합니다.

```
clouds:
  shiftstack:
    ...
    cacert: "/etc/pki/ca-trust/source/anchors/ca.crt.pem"
```

작은 정보

사용자 지정 **CA** 인증서로 설치 관리자를 실행한 후 **cloud-provider-config** 키맵에서 **ca-cert.pem** 키의 값을 편집하여 인증서를 업데이트할 수 있습니다. 명령줄에서 다음을 실행합니다.

```
$ oc edit configmap -n openshift-config cloud-provider-config
```

3. **clouds.yaml** 파일을 다음 위치 중 하나에 배치합니다.

   a. **OS_CLIENT_CONFIG_FILE** 환경 변수의 값

.

b. 현재 디렉터리

c. Unix 전용 사용자 구성 디렉터리(예: ~/.config/openstack/clouds.yaml)

d. Unix 전용 사이트 구성 디렉터리(예: /etc/openstack/clouds.yaml)

설치 프로그램은 clouds.yaml을 이 순서대로 검색합니다.

15.4.7. 설치 프로그램 받기

OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

프로세스

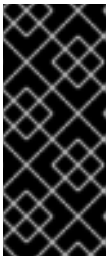
1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.

2. 인프라 공급자를 선택합니다.

3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.

중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.



중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 **OpenShift Container Platform** 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 **Linux** 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 **OpenShift Container Platform** 구성 요소에 대한 컨테이너 이미지를 제공하는 **Quay.io**를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

15.4.8. 설치 구성 파일 만들기

설치하는 **OpenShift Container Platform** 클러스터를 사용자 지정할 수 있습니다.

사전 요구 사항

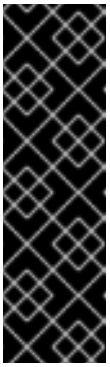
- **OpenShift Container Platform** 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.
- 서브스크립션 수준에서 서비스 주체 권한을 획득합니다.

프로세스

1. **install-config.yaml** 파일을 생성합니다.
   a. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 **<installation_directory>**는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.



중요

비어 있는 디렉터리를 지정합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

b. 화면에 나타나는 지시에 따라 클라우드에 대한 구성 세부 사항을 입력합니다.

i. 선택사항: 클러스터 시스템에 액세스하는 데 사용할 SSH 키를 선택합니다.

참고

설치 디버깅 또는 재해 복구 수행을 원하는 프로덕션 환경 OpenShift Container Platform 클러스터의 경우 ssh-agent 프로세스에 사용할 SSH 키를 지정하십시오.

ii. 클러스터를 설명할 수 있는 이름을 입력합니다.

iii. Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 붙여넣습니다.

2. install-config.yaml 파일을 수정합니다. 사용 가능한 매개변수에 대한 자세한 정보는 “설치 구성 매개변수” 섹션에서 확인할 수 있습니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 install-config.yaml 파일을 백업합니다.

중요

install-config.yaml 파일은 설치 과정에서 사용됩니다. 이 파일을 재사용하려면 지금 백업해야 합니다.

15.4.8.1. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 HTTP 또는 HTTPS 프록시를 사용

할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 **OpenShift Container Platform** 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 **API**에 대한 호출을 포함하여 모든 클러스터 발신(**Egress**) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.

참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), **Google Cloud Platform (GCP)**, **Microsoft Azure** 및 **Red Hat OpenStack Platform (RHOSP)**에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

프로세스

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1

2

클러스터 외부에서 **HTTPS** 연결을 구축하는 데 사용할 프록시 **URL**입니다.

3

대상 도메인 이름, **IP** 주소 또는 프록시에서 제외할 기타 네트워크 **CIDR**로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. ***를 사용하여 모든 대상에 대해 프록시를 바이패스합니다.

4

이 값을 제공하면 설치 프로그램에서 추가 **CA** 인증서를 보유할 **openshift -config** 네임스페이스에 **user-ca- bundle** 이라는 구성 맵을 생성합니다. **additionalTrustBundle** 및 하나 이상의 프록시 설정을 제공하는 경우 프록시 오브젝트는 **trustedCA** 필드의 **user-ca-bundle** 구성 맵을 참조하도록 구성됩니다. 그러면 **Cluster Network Operator**에서 **trustedCA** 매개변수에 대해 지정된 콘텐츠를 **RHCOS** 신뢰 번들과 병합하는 **trusted-ca-bundle** 구성 맵을 생성합니다. 프록시의 **ID** 인증서를 **RHCOS** 트러스트 번들에 있는 기관에서 서명하지 않은 경우 **additionalTrustBundle** 필드가 있어야 합니다.

참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 **OpenShift Container Platform**을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 **nil**이 됩니다.

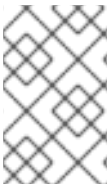
참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

15.4.9. 설치 구성 매개변수

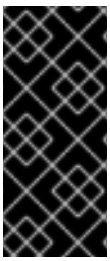
OpenShift Container Platform 클러스터를 배포하기 전에 매개변수 값을 제공하여 클러스터를 호스팅할 클라우드 플랫폼에서 사용자 계정을 설명하고 선택사항으로 클러스터의 플랫폼을 사용자 지정합니다. **install-config.yaml** 설치 구성 파일을 생성할 때 명령줄을 통해 필요한 매개변수 값을 제공합니다. 클

러스터를 사용자 지정하면 install-config.yaml 파일을 수정하여 플랫폼에 대한 세부 정보를 제공할 수 있습니다.



참고

설치한 후에는 install-config.yaml 파일에서 이러한 매개변수를 수정할 수 없습니다.



중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

15.4.9.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 15.14. 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체

매개변수	설명	값
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	소문자, 하이픈(**-**), 마침표(**.**)로 구성되는 문자열(예: **dev**)입니다.
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체
pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	`{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }`

15.4.9.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 15.15. 네트워크 매개변수

매개변수	설명	값

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체 **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.
networking.clusterNetwork	Pod의 IP 주소 블록입니다. 기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된**cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16

매개변수	설명	값
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16** **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

15.4.9.3. 선택적 구성 매개변수

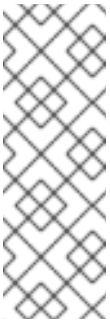
선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 15.16. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열

매개변수	설명	값
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은 **3**입니다.
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열

매개변수	설명	값
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 기본값은 **External**입니다. 이 필드를 **Internal** 로 설정하는 것은 클라우드 이외의 플랫폼에서는 지원되지 않습니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

15.4.9.4. RHOSP 배포의 사용자 지정 서브넷

선택한 **RHOSP(Red Hat OpenStack Platform)** 서브넷에 클러스터를 배포할 수도 있습니다. 서브넷의 **GUID**는 **install-config.yaml** 파일에서 **platform.openstack.machinesSubnet** 값으로 전달됩니다.

이 서브넷은 클러스터의 기본 서브넷으로 사용됩니다. 기본적으로 노드와 포트가 이 서브넷에서 생성됩니다. **platform.openstack.machinesSubnet** 속성의 값을 서브넷의 **UUID**로 설정하여 다른 **RHOSP** 서브넷에서 노드 및 포트를 생성할 수 있습니다.

사용자 지정 서브넷으로 **OpenShift Container Platform** 설치 관리자를 실행하기 전에 구성이 다음 요구 사항을 충족하는지 확인하십시오.

- **platform.openstack.machinesSubnet**에서 사용하는 서브넷에 **DHCP**가 활성화되어 있습니다.
- **platform.openstack.machinesSubnet**의 **CIDR**은 **networking.machineNetwork**의 **CIDR**과 일치합니다.
- 설치 프로그램 사용자에게 고정 **IP** 주소가 있는 포트를 포함하여 이 네트워크에 포트를 생성할 수 있는 권한이 있습니다.

사용자 지정 서브넷을 사용하는 클러스터에는 다음과 같은 제한 사항이 있습니다.

-

유동 IP 주소를 사용하는 클러스터를 설치하려는 경우 platform.openstack.machinesSubnet 서브넷이 externalNetwork 네트워크에 연결된 라우터에 연결되어 있어야 합니다.

- platform.openstack.machinesSubnet 값이 install-config.yaml 파일에 설정된 경우 설치 프로그램에서 RHOSP 머신의 프라이빗 네트워크 또는 서브넷을 생성하지 않습니다.

- 사용자 지정 서브넷과 동시에 platform.openstack.externalDNS 속성을 사용할 수 없습니다. 사용자 지정 서브넷을 사용하는 클러스터에 DNS를 추가하려면 RHOSP 네트워크에서 DNS를 구성합니다.

참고

기본적으로 API VIP는 x.x.x.5를 가져오고 Ingress VIP는 네트워크의 CIDR 블록에서 x.x.x.7을 가져옵니다. 이러한 기본값을 재정의하려면 DHCP 할당 풀 외부에 있는platform.openstack.apiVIP 및 platform.openstack.ingressVIP의 값을 설정합니다.

15.4.9.5. 베어 메탈 머신으로 클러스터 배포

클러스터가 베어 메탈 머신을 사용하도록 하려면 inventory.yaml 파일을 수정합니다. 클러스터는 베어 메탈에서 컨트롤 플레인 및 컴퓨팅 머신 모두를 실행하거나 컴퓨팅 머신만으로 실행할 수 있습니다.

Kuryr를 사용하는 클러스터에서 베어 메탈 컴퓨팅 머신이 지원되지 않습니다.

참고

install-config.yaml 파일에서 베어 메탈 작업자가 유동 IP 주소를 지원하는지 여부를 반영하는지 확인하십시오.

사전 요구 사항

- RHOSP Bare Metal 서비스(Ironic)가 활성화되어 RHOSP Compute API를 통해 액세스할 수 있습니다.

- 베어 메탈은 RHOSP 플레이버로 사용할 수 있습니다.

- RHOSP 네트워크는 VM 및 베어 메탈 서버 연결을 모두 지원합니다.

- 네트워크 구성이 공급자 네트워크를 사용하지 않습니다. 공급자 네트워크는 지원되지 않습니다.

- 기존 네트워크에 머신을 배포하려면 RHOSP 서브넷이 프로비저닝됩니다.

- 설치 관리자 프로비저닝 네트워크에 머신을 배포하려는 경우 RHOSP Bare Metal 서비스(Ironic)가 테넌트 네트워크에서 실행되는 PXE(Preboot eXecution Environment) 부팅 머신을 수신하고 상호 작용할 수 있습니다.

- OpenShift Container Platform 설치 프로세스의 일부로 inventory.yaml 파일을 생성하셨습니다.

프로세스

1. inventory.yaml 파일에서 머신의 플레이버를 편집합니다.

   a. 베어 메탈 컨트롤 플레인 머신을 사용하려면 os_flavor_master 값을 베어 메탈 플레이버로 변경합니다.

   b. os_flavor_worker의 값을 베어 메탈 플레이버로 변경합니다.

   베어 메탈 inventory.yaml 파일 예

```
all:
  hosts:
    localhost:
      ansible_connection: local
      ansible_python_interpreter: "{{ansible_playbook_python}}"

      # User-provided values
      os_subnet_range: '10.0.0.0/16'
      os_flavor_master: 'my-bare-metal-flavor' 1
      os_flavor_worker: 'my-bare-metal-flavor' 2
```

```
    os_image_rhcos: 'rhcos'
    os_external_network: 'external'
...
```

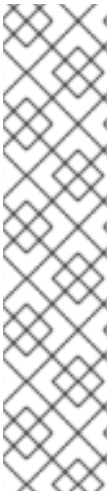
1

베어 메탈 컨트롤 플레인 머신이 필요한 경우 이 값을 베어 메탈 플레이버로 변경합니다.

2

컴퓨팅 머신에 사용할 베어 메탈 플레이버로 이 값을 변경합니다.

업데이트된 **inventory.yaml** 파일을 사용하여 설치 프로세스를 완료합니다. 배포 중에 생성된 머신은 파일에 추가한 플레이버를 사용합니다.



참고

설치 프로그램은 베어 메탈 머신이 부팅될 때까지 대기하는 동안 시간이 초과될 수 있습니다.

설치 프로그램이 시간 초과되면 설치 프로그램의 **wait-for** 명령을 사용하여 배포를 다시 시작한 다음 완료합니다. 예를 들면 다음과 같습니다.

```
./openshift-install wait-for install-complete --log-level debug
```

15.4.9.6. RHOSP용 샘플 사용자 지정 install-config.yaml 파일

이 샘플 **install-config.yaml**은 가능한 모든 **RHOSP(Red Hat OpenStack Platform)** 사용자 지정 옵션을 보여줍니다.

중요

이 샘플 파일은 참조용으로만 제공됩니다. **install-config.yaml** 파일은 설치 프로그램을 사용하여 받아야 합니다.

```
apiVersion: v1
```

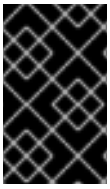
```
baseDomain: example.com
controlPlane:
  name: master
  platform: {}
  replicas: 3
compute:
- name: worker
  platform:
    openstack:
      type: ml.large
  replicas: 3
metadata:
  name: example
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  serviceNetwork:
  - 172.30.0.0/16
  networkType: OpenShiftSDN
platform:
  openstack:
    cloud: mycloud
    externalNetwork: external
    computeFlavor: m1.xlarge
    apiFloatingIP: 128.0.0.1
fips: false
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```

15.4.10. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 core 사용자의 ~/.ssh/authorized_keys 목록에 추가되어 암호 없는 인증을 활성화합니다.

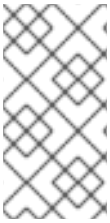
키가 노드에 전달되면 키 쌍을 사용하여 사용자 core로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. ./openshift-install gather 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.



중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.



참고

AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: ~/.ssh/id_ed25519)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 '~/.ssh 디렉터리에 있는지 확인하십시오.

참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 ed25519 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 rsa 또는 ecdsa 알고리즘을 사용하는 키를 생성합니다.

2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 ~/.ssh/id_ed25519.pub 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 ./openshift-install

gather 명령을 사용하려는 경우 필요합니다.

참고

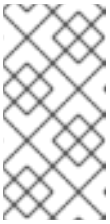
일부 배포에서는 ~/.ssh/id_rsa 및 ~/.ssh/id_dsa와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. ssh-agent 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```



참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. ssh-agent에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: ~/.ssh/id_ed25519).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

15.4.11. 환경에 대한 액세스 활성화

배포 시 모든 OpenShift Container Platform 시스템은 RHOSP(Red Hat OpenStack Platform) 테넌트 네트워크에서 생성됩니다. 따라서 대부분의 RHOSP 배포에서 직접 액세스할 수 없습니다.

설치시 부동 IP 주소 (FIP)를 사용하여 OpenShift Container Platform API 및 애플리케이션의 액세스를 설정할 수 있습니다. FIP를 구성하지 않고 설치를 완료 할 수도 있지만 설치 프로그램은 외부에서 API 또는 애플리케이션에 액세스하는 방법을 설정하지 않습니다.

15.4.11.1. 부동 IP 주소로 액세스 활성화

OpenShift Container Platform API 및 클러스터 애플리케이션에 대한 외부 액세스 용으로 유동 IP (FIP) 주소를 생성합니다.

프로세스

1. RHOSP(Red Hat OpenStack Platform) CLI를 사용하여 API FIP를 생성합니다.

```
$ openstack floating ip create --description "API <cluster_name>.<base_domain>"
<external_network>
```

2. RHOSP(Red Hat OpenStack Platform) CLI를 사용하여 앱 또는 Ingress, FIP를 생성합니다.

```
$ openstack floating ip create --description "Ingress <cluster_name>.<base_domain>"
<external_network>
```

3. API 및 Ingress FIP의 DNS 서버에 이러한 패턴에 따라 레코드를 추가합니다.

```
api.<cluster_name>.<base_domain>.  IN  A  <API_FIP>
*.apps.<cluster_name>.<base_domain>. IN  A <apps_FIP>
```

참고

DNS 서버를 제어하지 않으면 /etc/hosts 파일에 다음과 같은 클러스터 도메인 이름을 추가하여 클러스터에 액세스할 수 있습니다.

- <api_floating_ip> api.<cluster_name>.<base_domain>
- <application_floating_ip> grafana-openshift-monitoring.apps.<cluster_name>.<base_domain>
- <application_floating_ip> prometheus-k8s-openshift-monitoring.apps.<cluster_name>.<base_domain>
- <application_floating_ip> oauth-openshift.apps.<cluster_name>.<base_domain>
- <application_floating_ip> console-openshift-console.apps.<cluster_name>.<base_domain>
- application_floating_ip integrated-oauth-server-openshift-authentication.apps.<cluster_name>.<base_domain>

/etc/hosts 파일의 클러스터 도메인 이름은 웹 콘솔과 클러스터의 모니터링 인터페이스에 로컬로 액세스할 수 있는 권한을 부여합니다. kubectl 또는 oc 에서도 사용할 수 있습니다. <application_floating_ip>를 가리키는 추가 항목을 사용하여 사용자 애플리케이션에 액세스할 수 있습니다. 이 작업을 수행하면 사용자만 API 및 애플리케이션에 액세스할 수 있어 프로덕션 배포에는 적합하지 않지만 개발 및 테스트용 설치는 가능합니다.

4. FIP를 다음 매개 변수의 값으로 install-config.yaml 파일에 추가하십시오.
   - platform.openstack.ingressFloatingIP
   - platform.openstack.apiFloatingIP

이러한 값을 사용하는 경우 install-config.yaml 파일에서 platform.openstack.externalNetwork 매개 변수 값으로 외부 네트워크를 입력해야 합니다.

작은 정보

부동 IP 주소를 할당하고 방화벽 구성을 업데이트하여 클러스터 외부에서 OpenShift Container Platform 리소스를 사용할 수 있습니다.

15.4.11.2. 유동 IP 주소없이 설치 완료

유동 IP 주소를 지정하지 않고도 Red Hat OpenStack Platform (RHOSP)에 OpenShift Container Platform을 설치할 수 있습니다.

파일에서 다음을 정의하지 마십시오.

유동 IP 주소 또는 이름 확인 부족으로 인해 클러스터 API에 연결할 수 없는 시스템에서 설치 프로그램을 실행하면 설치에 실패합니다. 이러한 경우 설치 실패를 방지하기 위해 프록시 네트워크를 사용하거나 시스템과 동일한 네트워크에있는 시스템에서 설치 프로그램을 실행할 수 있습니다.

참고

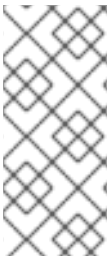
API 및 Ingress 포트의 DNS 레코드를 생성하여 이름 확인을 활성화할 수 있습니다. 예를 들면 다음과 같습니다.

```
api.<cluster_name>.<base_domain>.  IN  A  <api_port_IP>
*.apps.<cluster_name>.<base_domain>. IN  A <ingress_port_IP>
```

DNS 서버를 제어하지 않으면 /etc/hosts 파일에 레코드를 추가할 수 있습니다. 이 작업을 수행하면 사용자만 API에 액세스할 수 있어 프로덕션 배포에는 적합하지 않지만 개발 및 테스트용 설치는 가능합니다.

15.4.12. 컴퓨팅 머신용 SR-IOV 네트워크 생성

RHOSP(Red Hat OpenStack Platform) 배포에서 단일 루트 I/O 가상화(SR-IOV)를 지원하는 경우 컴퓨팅 머신이 실행되는 SR-IOV 네트워크를 프로비저닝할 수 있습니다.

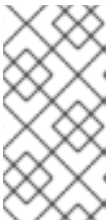


참고

다음 지침은 컴퓨팅 머신에 연결할 수 있는 외부 플랫 네트워크 및 외부 **VLAN** 기반 네트워크 생성에 대해 설명합니다. **RHOSP** 배포에 따라 다른 네트워크 유형이 필요할 수 있습니다.

사전 요구 사항

- 클러스터는 **SR-IOV**를 지원합니다.



참고

클러스터가 지원하는 기능을 잘 모를 경우 **OpenShift Container Platform SR-IOV** 하드웨어 네트워크 설명서를 참조하십시오.

- **RHOSP** 배포의 일부로 **radio** 및 **uplink** 공급자 네트워크를 생성했습니다. **radio** 및 **uplink** 이름은 이러한 네트워크를 표시하기 위해 모든 예제 명령에서 사용됩니다.

절차

1. 명령줄에서 **radio RHOSP** 네트워크를 생성합니다.

```
$ openstack network create radio --provider-physical-network radio --provider-
network-type flat --external
```

2. **uplink RHOSP** 네트워크를 생성합니다.

```
$ openstack network create uplink --provider-physical-network uplink --provider-
network-type vlan --external
```

3. **radio** 네트워크의 서브넷을 생성합니다.

```
$ openstack subnet create --network radio --subnet-range
<radio_network_subnet_range> radio
```

4. **uplink** 네트워크의 서브넷을 생성합니다.

```
$ openstack subnet create --network uplink --subnet-range
<uplink_network_subnet_range> uplink
```

15.4.13. 클러스터 배포

호환되는 클라우드 플랫폼에 OpenShift Container Platform을 설치할 수 있습니다.

중요

최초 설치 과정에서 설치 프로그램의 **create cluster** 명령을 한 번만 실행할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

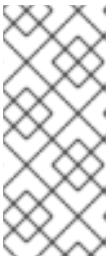
절차

1. 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터 배포를 초기화합니다.

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
  --log-level=info 2
```

1 <installation_directory> 값으로 사용자 지정한 **./install-config.yaml** 파일의 위치를 지정합니다.

2 다른 설치 세부 사항을 보려면 **info** 대신 **warn, debug** 또는 **error**를 지정합니다.



참고

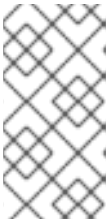
호스트에 구성된 클라우드 공급자 계정에 클러스터를 배포하기에 충분한 권한이 없는 경우, 설치 프로세스가 중단되고 누락된 권한을 알리는 메시지가 표시됩니다.

클러스터 배포가 완료되면 웹 콘솔로 연결되는 링크와 **kubeadmin** 사용자의 인증 정보가 포

함된 클러스터 액세스 지침이 사용자 터미널에 표시됩니다.

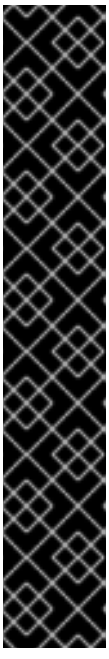
출력 예

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-
ymBZj-Wt5AL"
INFO Time elapsed: 36m22s
```



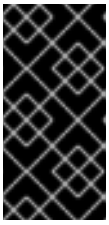
참고

설치에 성공하면 클러스터 액세스 및 인증 정보도 **<installation_directory>/.openshift_install.log**로 출력됩니다.



중요

- 설치 프로그램에서 생성하는 **Ignition** 구성 파일에 **24**시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 **24**시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 **kubelet** 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청**(CSR)**을 수동으로 승인해야 합니다. 자세한 내용은 *만료된 컨트롤 플레인 인증서에서 복구* 문서를 참조하십시오.
- 클러스터를 설치한 후 **24**시간에서 **22**시간까지의 인증서가 교체되기 때문에 생성된 후 **12**시간 이내에 **Ignition** 구성 파일을 사용하는 것이 좋습니다. **12**시간 이내에 **Ignition** 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.



중요

설치 프로그램에서 생성되는 파일이나 설치 프로그램을 삭제해서는 안 됩니다. 클러스터를 삭제하려면 두 가지가 모두 필요합니다.

15.4.14. 클러스터 상태 확인

설치 중 또는 설치 후 OpenShift Container Platform 클러스터의 상태를 확인할 수 있습니다.

프로세스

1. 클러스터 환경에서 관리자의 kubeconfig 파일을 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

kubeconfig 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다.

2. 배포 후 생성된 컨트롤 플레인 및 컴퓨팅 시스템을 확인합니다.

```
$ oc get nodes
```

3. 클러스터 버전을 확인합니다.

```
$ oc get clusterversion
```

4. Operator 상태를 확인합니다.

```
$ oc get clusteroperator
```

5. 클러스터에서 실행 중인 모든 Pod를 확인합니다.

```
$ oc get pods -A
```

15.4.15. CLI를 사용하여 클러스터에 로그인

클러스터 kubeconfig 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다.

kubeconfig 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- oc CLI를 설치했습니다.

프로세스

1. kubeadmin 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 oc 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

클러스터가 작동하고 있습니다. SR-IOV 컴퓨팅 머신을 추가하려면 먼저 추가 작업을 수행해야 합니다.

15.4.16. SR-IOV 용 RHOSP에서 실행되는 클러스터 준비

RHOSP (Red Hat OpenStack Platform)에서 실행되는 클러스터에서 SR-IOV(Single Root I/O Virtualization)를 사용하기 전에 RHOSP 메타데이터 서비스를 드라이브로 마운트하고 가상 함수 I/O (VFIO) 드라이버에서 No-IOMMU Operator를 활성화합니다.

15.4.16.1. RHOSP 메타데이터 서비스를 마운트 가능한 드라이브로 활성화

RHOSP(Red Hat OpenStack Platform) 메타데이터 서비스를 마운트 가능 드라이브로 사용할 수 있도록 시스템 풀에 머신 구성을 적용할 수 있습니다.

다음 머신 구성을 사용하면 SR-IOV Network Operator 내에서 RHOSP 네트워크 UUID를 표시할 수 있습니다. 이 구성으로 SR-IOV 리소스를 클러스터 SR-IOV 리소스에 간단하게 연결할 수 있습니다.

프로세스

1. 다음 템플릿에서 머신 구성 파일을 생성합니다.

마운트 가능한 메타데이터 서비스 머신 구성 파일

```
kind: MachineConfig
apiVersion: machineconfiguration.openshift.io/v1
metadata:
  name: 20-mount-config (1)
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 3.2.0
    systemd:
      units:
        - name: create-mountpoint-var-config.service
          enabled: true
          contents: |
            [Unit]
            Description=Create mountpoint /var/config
            Before=kubelet.service

            [Service]
            ExecStart=/bin/mkdir -p /var/config

            [Install]
            WantedBy=var-config.mount

        - name: var-config.mount
          enabled: true
          contents: |
```

```
[Unit]
Before=local-fs.target
[Mount]
Where=/var/config
What=/dev/disk/by-label/config-2
[Install]
WantedBy=local-fs.target
```

1

선택한 이름을 대체할 수 있습니다.

2. 명령줄에서 머신 구성을 적용합니다.

```
$ oc apply -f <machine_config_file_name>.yaml
```

15.4.16.2. RHOSP VFIO 드라이버의 No-IOMMU 기능 활성화

머신 풀에 머신 구성을 적용하여 RHOSP(Red Hat OpenStack Platform) 가상 기능 I/O(VFIO) 드라이버에 대한 No-IOMMU 기능을 활성화할 수 있습니다. RHOSP vfio-pci 드라이버에는 이 기능이 필요합니다.

프로세스

1. 다음 템플릿에서 머신 구성 파일을 생성합니다.

No-IOMMU VFIO 머신 구성 파일

```
kind: MachineConfig
apiVersion: machineconfiguration.openshift.io/v1
metadata:
  name: 99-vfio-noiommu 1
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
```

```
    - path: /etc/modprobe.d/vfio-noiommu.conf
      mode: 0644
      contents:
        source:
data:;base64,b3B0aW9ucyB2ZmlvIGVuYWJsZV91bnNhZmVfbm9pb21tdV9tb2RlPTEK
```

1

선택한 이름을 대체할 수 있습니다.

2. 명령줄에서 머신 구성을 적용합니다.

```
$ oc apply -f <machine_config_file_name>.yaml
```

클러스터가 SR-IOV 구성을 설치 및 준비합니다. "다음 단계" 섹션에 나열된 설치 후 SR-IOV 작업을 완료합니다.

15.4.17. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

15.4.18. 다음 단계

- 클러스터의 SR-IOV 구성을 완료하려면 다음을 수행합니다.

    - Performance Addon Operator를 설치합니다.
    - 대용량 페이지 지원을 사용하여 Performance Addon Operator를 구성합니다.
    - SR-IOV Operator를 설치합니다.
    - SR-IOV 네트워크 장치를 구성합니다.
    - SR-IOV 컴퓨팅 머신 세트를 추가합니다.

- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.
- 노드 포트에 대한 외부 액세스를 활성화해야 하는 경우 노드 포트를 사용하여 Ingress 클러스터 트래픽을 구성합니다.
- 유동 IP 주소를 통한 애플리케이션 트래픽을 허용하도록 RHOSP를 구성하지 않은 경우 유동 IP 주소로 RHOSP 액세스를 구성합니다.

15.5. 사용자 인프라의 OPENSTACK에 클러스터 설치

OpenShift Container Platform 버전 4.9에서는 사용자 프로비저닝 인프라에서 실행되는 클러스터를 RHOSP(Red Hat OpenStack Platform)에 설치할 수 있습니다.

자체 인프라를 사용하면 클러스터를 기존 인프라 및 수정 사항과 통합할 수 있습니다. 이 프로세스에서는 Nova 서버, Neutron 포트, 보안 그룹과 같은 모든 RHOSP 리소스를 생성해야 하므로 설치 관리자 프로비저닝 설치보다 사용자가 수행해야 하는 작업이 더 많습니다. 그러나 Red Hat은 배포 프로세스에 도움이 되는 Ansible 플레이북을 제공합니다.

15.5.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.
- OpenShift Container Platform 4.9가 OpenShift 클러스터에 지원되는 플랫폼 섹션을 사용하여 현재 RHOSP 버전과 호환되는지 확인했습니다. RHOSP 지원 매트릭스의 OpenShift Container Platform을 확인하여 여러 버전의 플랫폼 지원을 비교할 수도 있습니다.
- OpenShift Container Platform을 설치할 RHOSP 계정을 준비합니다.
- 설치 프로그램을 실행하는 시스템에서 다음을 준비합니다.
  - 설치 프로세스에서 만드는 파일을 저장할 수 있는 단일 디렉터리
  - Python 3

15.5.2. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.

중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

15.5.3. RHOSP에 OpenShift Container Platform을 설치하기 위한 리소스 지침

OpenShift Container Platform 설치를 지원하려면 RHOSP(Red Hat OpenStack Platform) 할당량이 다음 요구사항을 충족해야 합니다.

표 15.17. RHOSP의 기본 OpenShift Container Platform 클러스터에 권장되는 리소스

리소스 이름	값
부동 IP 주소	3
포트	15
라우터	1
서브넷	1
RAM	ECDHEGB
vCPU	22
볼륨 스토리지	275GB
인스턴스	7
보안 그룹	3
보안 그룹 규칙	60

권장 리소스보다 적은 리소스로도 클러스터가 작동할 수 있지만 성능은 보장되지 않습니다.

중요

swiftoperator 역할을 가진 사용자 계정으로 **RHOSP** 개체 스토리지(**Swift**)를 사용하고 운영하는 경우 **OpenShift Container Platform** 이미지 레지스트리의 기본 백엔드로 사용됩니다. 이 경우 볼륨 스토리지 요구사항은 **175GB**입니다. **Swift** 공간 요구사항은 이미지 레지스트리의 크기에 따라 다릅니다.

참고

기본적으로 보안 그룹 및 보안 그룹 규칙 할당량이 적을 수 있습니다. 문제가 발생하면 관리자로 **openstack quota set --secgroups 3 --secgroup-rules 60 <project>**를 실행하여 할당량을 늘립니다.

OpenShift Container Platform 배포는 컨트롤 플레인 시스템, 컴퓨팅 시스템, 부트스트랩 시스템으로 구성됩니다.

15.5.3.1. 컨트롤 플레인 머신

기본적으로 **OpenShift Container Platform** 설치 프로세스는 세 개의 컨트롤 플레인 시스템을 생성합니다.

각 시스템의 요구사항은 다음과 같습니다.

- **RHOSP** 할당량의 인스턴스
- **RHOSP** 할당량의 포트
- 최소 **16GB** 메모리 및 **vCPU 4**개가 있는 플레이버
- **RHOSP** 할당량에서 최소 **100GB** 스토리지 공간

15.5.3.2. 컴퓨팅 머신

기본적으로 **OpenShift Container Platform** 설치 프로세스는 세 개의 컴퓨팅 시스템을 생성합니다.

각 시스템의 요구사항은 다음과 같습니다.

- RHOSP 할당량의 인스턴스
- RHOSP 할당량의 포트
- 8GB 메모리 및 vCPU 2개 이상 있는 플레이버
- RHOSP 할당량에서 최소 100GB 스토리지 공간

작은 정보

컴퓨팅 시스템은 OpenShift Container Platform에서 실행하는 애플리케이션을 호스팅합니다. 최대한 많이 실행하는 것이 좋습니다.

15.5.3.3. 부트스트랩 시스템

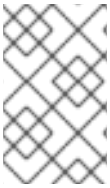
설치하는 동안 컨트롤 플레인을 유지하기 위해 부트스트랩 시스템이 임시로 프로비저닝됩니다. 프로덕션 컨트롤 플레인이 준비되면 부트스트랩 시스템이 프로비저닝 해제됩니다.

부트스트랩 시스템의 요구사항은 다음과 같습니다.

- RHOSP 할당량의 인스턴스
- RHOSP 할당량의 포트
- 최소 16GB 메모리 및 vCPU 4개가 있는 플레이버
- RHOSP 할당량에서 최소 100GB 스토리지 공간

15.5.4. 플레이북 종속 항목 다운로드

사용자 프로비저닝 인프라에서의 설치 프로세스를 간소화하는 **Ansible** 플레이북에는 몇 가지 **Python** 모듈이 필요합니다. 설치 관리자를 실행할 시스템에서 모듈 리포지토리를 추가하고 다운로드합니다.



참고

이 방법은 현재 **Red Hat Enterprise Linux (RHEL) 8**을 사용하는 것으로 가정합니다.

사전 요구 사항

- **Python 3**가 시스템에 설치되어 있습니다.

프로세스

1. 명령줄에서 리포지토리를 추가합니다.
   a. **Red Hat Subscription Manager**에 등록합니다.

```
$ sudo subscription-manager register # If not done already
```

   b. 최신 서브스크립션 데이터를 가져옵니다.

```
$ sudo subscription-manager attach --pool=$YOUR_POOLID # If not done already
```

   c. 현재 리포지토리를 비활성화합니다.

```
$ sudo subscription-manager repos --disable=* # If not done already
```

   d. 필요한 리포지토리를 추가합니다.

```
$ sudo subscription-manager repos \
  --enable=rhel-8-for-x86_64-baseos-rpms \
  --enable=openstack-16-tools-for-rhel-8-x86_64-rpms \
  --enable=ansible-2.9-for-rhel-8-x86_64-rpms \
  --enable=rhel-8-for-x86_64-appstream-rpms
```

2. 모듈을 설치합니다.

```
$ sudo yum install python3-openstackclient ansible python3-openstacksdk python3-
netaddr
```

3. python 명령이 python3를 가리키는지 확인합니다.

```
$ sudo alternatives --set python /usr/bin/python3
```

15.5.5. 설치 플레이북 다운로드

자체 RHOSP(Red Hat OpenStack Platform) 인프라에 OpenShift Container Platform을 설치하는 데 사용할 수 있는 Ansible 플레이북을 다운로드합니다.

사전 요구 사항

- curl 명령줄 툴은 사용자의 머신에서 사용할 수 있습니다.

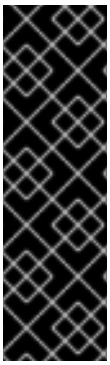
프로세스

- 플레이북을 작업 디렉터리에 다운로드하려면 명령줄에서 다음 스크립트를 실행합니다.

```
$ xargs -n 1 curl -O <<< '
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/bootstrap.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/common.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/compute-nodes.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/control-plane.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/inventory.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/network.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/security-groups.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/down-bootstrap.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/down-compute-nodes.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/down-control-plane.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/down-load-balancers.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/down-network.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
```

```
4.9/upi/openstack/down-security-groups.yaml
      https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/down-containers.yaml'
```

플레이북이 사용자의 머신에 다운로드됩니다.



중요

설치 프로세스 중에 플레이북을 수정하여 배포를 구성할 수 있습니다.

클러스터 수명에 대해 모든 플레이북을 유지합니다. RHOSP에서 OpenShift Container Platform 클러스터를 제거하려면 플레이북이 있어야 합니다.

중요

bootstrap.yaml, compute-nodes.yaml, control-plane.yaml, network.yaml 및 security-groups.yaml 파일에서 만든 모든 편집 사항을 down- 접두어가 있는 해당 플레이북과 일치시켜야 합니다. 예를 들어 bootstrap.yaml 파일에 대한 편집도 down-bootstrap.yaml 파일에 반영해야 합니다. 두 파일을 모두 편집하지 않으면 지원되는 클러스터 제거 프로세스가 실패합니다.

15.5.6. 설치 프로그램 받기

OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

프로세스

1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.

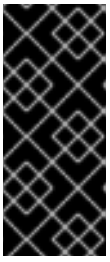
2. 인프라 공급자를 선택합니다.

3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성

파일을 저장할 디렉터리에 파일을 저장합니다.

중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.



중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 OpenShift Container Platform 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

15.5.7. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

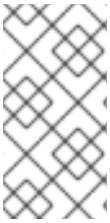
OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 core 사용자의 ~/.ssh/authorized_keys 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 core로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. ./openshift-install gather 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.



참고

AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: ~/.ssh/id_ed25519)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 '~/.ssh 디렉터리에 있는지 확인하십시오.

참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 ed25519 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 rsa 또는 ecdsa 알고리즘을 사용하는 키를 생성합니다.

2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 ~/.ssh/id_ed25519.pub 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 ./openshift-install

gather 명령을 사용하려는 경우 필요합니다.

참고

일부 배포에서는 ~/.ssh/id_rsa 및 ~/.ssh/id_dsa와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. ssh-agent 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. ssh-agent에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: ~/.ssh/id_ed25519).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

15.5.8. RHCOS(Red Hat Enterprise Linux CoreOS) 이미지 생성

OpenShift Container Platform 설치 프로그램을 사용하려면 RHOSP(Red Hat OpenStack Platform) 클러스터에 RHCOS(Red Hat Enterprise Linux CoreOS) 이미지가 있어야 합니다. 최신 RHCOS 이미지를 검색한 다음 RHOSP CLI를 사용하여 업로드하십시오.

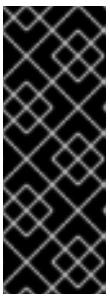
사전 요구 사항

- RHOSP CLI가 설치되어 있습니다.

프로세스

1. Red Hat Customer Portal의 제품 다운로드 페이지에 로그인합니다.

2. Version 에서 Red Hat Enterprise Linux (RHEL) 8용 최신 OpenShift Container Platform 4.9 릴리스를 선택합니다.



중요

RHCOS 이미지는 OpenShift Container Platform 릴리스에 따라 변경되지 않을 수 있습니다. 설치하는 OpenShift Container Platform 버전과 같거나 그 이하의 버전 중 가장 최신 버전의 이미지를 다운로드해야 합니다. 사용 가능한 경우 OpenShift Container Platform 버전과 일치하는 이미지 버전을 사용합니다.

3. *RHCOS(Red Hat Enterprise Linux CoreOS) - OpenStack Image (QCOW)*를 다운로드합니다.

4. 이미지 압축을 풉니다.

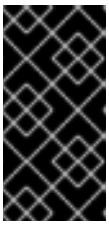
참고

RHOSP 이미지 압축을 풀어야 클러스터가 사용할 수 있습니다. 다운로드한 파일의 이름에 .gz 또는 .tgz와 같은 압축 확장자가 포함되지 않을 수 있습니다. 파일 압축 여부를 확인하려면 명령줄에서 다음을 입력합니다.

```
$ file <name_of_downloaded_file>
```

5. 다운로드한 이미지에서 RHOSP CLI를 사용하여 이름이 rhcos인 이미지를 클러스터에 생성합니다.

```
$ openstack image create --container-format=bare --disk-format=qcow2 --file rhcos-${RHCOS_VERSION}-openstack.qcow2 rhcos
```



중요

RHOSP 환경에 따라 이미지를 .raw 또는 .qcow2 형식으로 업로드할 수 있습니다. Ceph를 사용하는 경우 .raw 형식을 사용해야 합니다.

주의

설치 프로그램이 이름이 같은 여러 이미지를 발견하면 그 중 하나를 임의로 선택합니다. 이 동작을 방지하려면 RHOSP에서 리소스의 고유한 이름을 만듭니다.

이미지를 RHOSP에 업로드한 후 설치 프로세스에서 사용할 수 있습니다.

15.5.9. 외부 네트워크 액세스 확인

OpenShift Container Platform 설치 프로세스에는 외부 네트워크에 액세스해야 합니다. 외부 네트워크값을 제공해야 하며 그렇지 않으면 배포가 실패합니다. 프로세스를 시작하기 전에 외부 라우터 유형의 네트워크가 RHOSP(Red Hat OpenStack Platform)에 있는지 확인합니다.

사전 요구 사항

- DHCP 에이전트가 인스턴스의 DNS 쿼리를 전달하도록 OpenStack의 네트워킹 서비스 구성

프로세스

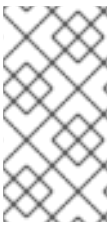
1. RHOSP CLI를 사용하여 '외부' 네트워크의 이름과 ID를 확인합니다.

```
$ openstack network list --long -c ID -c Name -c "Router Type"
```

출력 예

```
+--------------------------------------+----------------+-------------+
| ID                                   | Name           | Router Type |
+--------------------------------------+----------------+-------------+
| 148a8023-62a7-4672-b018-003462f8d7dc | public_network | External    |
+--------------------------------------+----------------+-------------+
```

외부 라우터 유형의 네트워크가 네트워크 목록에 나타납니다. 네트워크가 하나 이상 나타나지 않으면 기본 부동 IP 네트워크 생성 및 기본 공급자 네트워크 생성을 참조하십시오.



참고

Neutron 트렁크 서비스 플러그인이 활성화된 경우 기본적으로 트렁크 포트가 생성됩니다. 자세한 내용은 Neutron 트렁크 포트를 참조하십시오.

15.5.10. 환경에 대한 액세스 활성화

배포 시 모든 OpenShift Container Platform 시스템은 RHOSP(Red Hat OpenStack Platform) 테넌트 네트워크에서 생성됩니다. 따라서 대부분의 RHOSP 배포에서 직접 액세스할 수 없습니다.

설치시 부동 IP 주소 (FIP)를 사용하여 OpenShift Container Platform API 및 애플리케이션의 액세스를 설정할 수 있습니다. FIP를 구성하지 않고 설치를 완료 할 수도 있지만 설치 프로그램은 외부에서 API 또는 애플리케이션에 액세스하는 방법을 설정하지 않습니다.

15.5.10.1. 부동 IP 주소로 액세스 활성화

OpenShift Container Platform API, 클러스터 애플리케이션 및 부트스트랩 프로세스에 대한 외부 액세스 용으로 유동 **IP (FIP)** 주소를 생성합니다.

절차

1. **RHOSP(Red Hat OpenStack Platform) CLI**를 사용하여 **API FIP**를 생성합니다.

```
$ openstack floating ip create --description "API <cluster_name>.<base_domain>"
<external_network>
```

2. **RHOSP(Red Hat OpenStack Platform) CLI**를 사용하여 앱 또는 **Ingress, FIP**를 생성합니다.

```
$ openstack floating ip create --description "Ingress <cluster_name>.<base_domain>"
<external_network>
```

3. **RHOSP(Red Hat OpenStack Platform) CLI**를 사용하여 부트스트랩 **FIP**를 생성합니다.

```
$ openstack floating ip create --description "bootstrap machine" <external_network>
```

4. **API** 및 **Ingress FIP**의 **DNS** 서버에 이러한 패턴에 따라 레코드를 추가합니다.

```
api.<cluster_name>.<base_domain>.  IN  A  <API_FIP>
*.apps.<cluster_name>.<base_domain>. IN  A <apps_FIP>
```

참고

DNS 서버를 제어하지 않으면 /etc/hosts 파일에 다음과 같은 클러스터 도메인 이름을 추가하여 클러스터에 액세스할 수 있습니다.

- <api_floating_ip> api.<cluster_name>.<base_domain>
- <application_floating_ip> grafana-openshift-monitoring.apps.<cluster_name>.<base_domain>
- <application_floating_ip> prometheus-k8s-openshift-monitoring.apps.<cluster_name>.<base_domain>
- <application_floating_ip> oauth-openshift.apps.<cluster_name>.<base_domain>
- <application_floating_ip> console-openshift-console.apps.<cluster_name>.<base_domain>
- application_floating_ip integrated-oauth-server-openshift-authentication.apps.<cluster_name>.<base_domain>

/etc/hosts 파일의 클러스터 도메인 이름은 웹 콘솔과 클러스터의 모니터링 인터페이스에 로컬로 액세스할 수 있는 권한을 부여합니다. kubectl 또는 oc 에서도 사용할 수 있습니다. <application_floating_ip>를 가리키는 추가 항목을 사용하여 사용자 애플리케이션에 액세스할 수 있습니다. 이 작업을 수행하면 사용자만 API 및 애플리케이션에 액세스할 수 있어 프로덕션 배포에는 적합하지 않지만 개발 및 테스트용 설치는 가능합니다.

5. FIP를 다음 변수의 값으로 inventory.yaml 파일에 추가합니다.

- os_api_fip
- os_bootstrap_fip

- os_ingress_fip

이러한 값을 사용하는 경우 **inventory.yaml** 파일의 **os_external_network** 변수 값으로 외부 네트워크를 입력해야 합니다.

작은 정보

부동 **IP** 주소를 할당하고 방화벽 구성을 업데이트하여 클러스터 외부에서 **OpenShift Container Platform** 리소스를 사용할 수 있습니다.

15.5.10.2. 유동 IP 주소없이 설치 완료

유동 **IP** 주소를 지정하지 않고도 **Red Hat OpenStack Platform (RHOSP)**에 **OpenShift Container Platform**을 설치할 수 있습니다.

inventory.yaml 파일에서 다음 변수를 정의하지 마십시오.

- os_api_fip
- os_bootstrap_fip
- os_ingress_fip

외부 네트워크를 제공할 수 없는 경우 **os_external_network**를 비워 둘 수도 있습니다. **os_external_network**의 값을 지정하지 않으면 라우터가 생성되지 않으며 추가 작업없이 설치 프로그램이 **Glance**에서 이미지를 검색하지 못합니다. 나중에 설치 프로세스에서 네트워크 리소스를 만들 때 외부 연결을 직접 구성해야 합니다.

유동 **IP** 주소 또는 이름 확인 부족으로 인해 클러스터 **API**에 연결할 수 없는 시스템에서 **wait-for** 명령으로 설치 프로그램을 실행하면 설치에 실패합니다. 이러한 경우 설치 실패를 방지하기 위해 프록시 네트워크를 사용하거나 시스템과 동일한 네트워크에있는 시스템에서 설치 프로그램을 실행할 수 있습니다.

참고

API 및 Ingress 포트의 DNS 레코드를 생성하여 이름 확인을 활성화할 수 있습니다. 예를 들면 다음과 같습니다.

```
api.<cluster_name>.<base_domain>.  IN  A  <api_port_IP>
*.apps.<cluster_name>.<base_domain>. IN  A <ingress_port_IP>
```

DNS 서버를 제어하지 않으면 /etc/hosts 파일에 레코드를 추가할 수 있습니다. 이 작업을 수행하면 사용자만 API에 액세스할 수 있어 프로덕션 배포에는 적합하지 않지만 개발 및 테스트용 설치는 가능합니다.

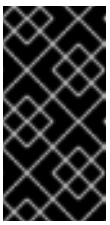
15.5.11. 설치 프로그램에 대한 매개변수 정의

OpenShift Container Platform 설치 프로그램은 clouds.yaml 파일을 사용합니다. 이 파일은 프로젝트 이름, 로그인 정보, 인증 서비스 URL 등 RHOSP(Red Hat OpenStack Platform) 구성 매개변수를 설명합니다.

프로세스

1. clouds.yaml 파일을 만듭니다.

- RHOSP 배포에 Horizon 웹 UI가 포함되어 있으면 그 안에 clouds.yaml 파일을 만듭니다.



중요

auth 필드에 암호를 추가해야 합니다. clouds.yaml과 별도의 파일에 비밀을 저장할 수도 있습니다.

- RHOSP 배포에 Horizon 웹 UI가 포함되어 있지 않거나 Horizon을 사용하지 않으려면 파일을 직접 만듭니다. clouds.yaml에 대한 자세한 내용은 RHOSP 문서의 구성 파일을 참조하십시오.

```
clouds:
  shiftstack:
    auth:
      auth_url: http://10.10.14.42:5000/v3
      project_name: shiftstack
      username: shiftstack_user
```

```
      password: XXX
      user_domain_name: Default
      project_domain_name: Default
    dev-env:
      region_name: RegionOne
      auth:
        username: 'devuser'
        password: XXX
        project_name: 'devonly'
        auth_url: 'https://10.10.14.22:5001/v2.0'
```

2. RHOSP 설치에서 끝점 인증을 위해 자체 서명된 CA(인증 기관) 인증서를 사용하는 경우:

   a. 인증 기관 파일을 시스템에 복사합니다.

   b. cacerts 키를 clouds.yaml 파일에 추가합니다. 값은 CA 인증서에 대한 루트가 아닌 액세스 가능한 절대 경로여야 합니다.

```
clouds:
  shiftstack:
    ...
    cacert: "/etc/pki/ca-trust/source/anchors/ca.crt.pem"
```

작은 정보

사용자 지정 CA 인증서로 설치 관리자를 실행한 후 cloud-provider-config 키맵에서 ca-cert.pem 키의 값을 편집하여 인증서를 업데이트할 수 있습니다. 명령줄에서 다음을 실행합니다.

```
$ oc edit configmap -n openshift-config cloud-provider-config
```

3. clouds.yaml 파일을 다음 위치 중 하나에 배치합니다.

   a. OS_CLIENT_CONFIG_FILE 환경 변수의 값

   b. 현재 디렉터리

   c. Unix 전용 사용자 구성 디렉터리(예: ~/.config/openstack/clouds.yaml)

d.

Unix 전용 사이트 구성 디렉터리(예: /etc/openstack/clouds.yaml)

설치 프로그램은 clouds.yaml을 이 순서대로 검색합니다.

15.5.12. 설치 구성 파일 만들기

RHOSP(Red Hat OpenStack Platform)에 설치하는 OpenShift Container Platform 클러스터를 사용자 지정할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.
- 서브스크립션 수준에서 서비스 주체 권한을 획득합니다.

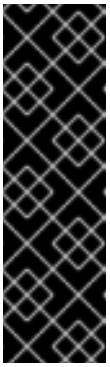
프로세스

1. install-config.yaml 파일을 생성합니다.

a. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 <installation_directory>는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.



중요

비어 있는 디렉터리를 지정합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

b. 화면에 나타나는 지시에 따라 클라우드에 대한 구성 세부 사항을 입력합니다.

i. 선택사항: 클러스터 시스템에 액세스하는 데 사용할 SSH 키를 선택합니다.

참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 ssh-agent 프로세스가 사용하는 SSH 키를 지정합니다.

ii. 대상 플랫폼으로 openstack을 선택합니다.

iii. 클러스터 설치에 사용할 RHOSP(Red Hat OpenStack Platform) 외부 네트워크 이름을 지정합니다.

iv. OpenShift API에 대한 외부 액세스에 사용할 부동 IP 주소를 지정합니다.

v. 컨트롤 플레인 노드에 사용할 최소 16GB의 RAM과 컴퓨팅 노드에 8GB RAM이 있는 RHOSP 플레이버를 지정합니다.

vi. 클러스터를 배포할 기본 도메인을 선택합니다. 모든 DNS 레코드는 이 기본 도메인의 하위 도메인이 되므로 클러스터 이름을 포함합니다.

vii. 클러스터 이름을 입력합니다. 이름은 14자 이하여야 합니다.

viii. Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 붙여넣습니다.

2. install-config.yaml 파일을 수정합니다. 사용 가능한 매개변수에 대한 자세한 정보는 “설치 구성 매개변수” 섹션에서 확인할 수 있습니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 install-config.yaml 파일을 백업합니다.

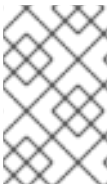
중요

install-config.yaml 파일은 설치 과정에서 사용됩니다. 이 파일을 재사용하려면 지금 백업해야 합니다.

이제 사용자가 지정한 디렉터리에 **install-config.yaml** 파일이 있습니다.

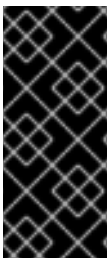
15.5.13. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 매개변수 값을 제공하여 클러스터를 호스팅할 클라우드 플랫폼에서 사용자 계정을 설명하고 선택사항으로 클러스터의 플랫폼을 사용자 지정합니다. **install-config.yaml** 설치 구성 파일을 생성할 때 명령줄을 통해 필요한 매개변수 값을 제공합니다. 클러스터를 사용자 지정하면 **install-config.yaml** 파일을 수정하여 플랫폼에 대한 세부 정보를 제공할 수 있습니다.



참고

설치한 후에는 **install-config.yaml** 파일에서 이러한 매개변수를 수정할 수 없습니다.



중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

15.5.13.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 15.18. 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열

매개변수	설명	값
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	소문자, 하이픈(**-**), 마침표(**.**)로 구성되는 문자열(예: **dev**)입니다. 문자열은 14자 이하여야 합니다.
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

15.5.13.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스

터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 15.19. 네트워크 매개변수

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체  **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.

networking.clusterNetwork	Pod의 IP 주소 블록입니다.
	기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다.
	여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.
		개체의 배열입니다. 예를 들면 다음과 같습니다.
		`networking:`
		`  clusterNetwork:`
		`  - cidr: 10.128.0.0/14`
		`  hostPrefix: 23`

networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된 **cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.

매개변수	설명	값
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16** **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

15.5.13.3. 선택적 구성 매개변수

선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 15.20. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.

매개변수	설명	값
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은**3**입니다.
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열

매개변수	설명	값
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 기본값은 **External**입니다. 이 필드를 **Internal** 로 설정하는 것은 클라우드 이외의 플랫폼에서는 지원되지 않습니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

15.5.13.4. 추가 RHOSP(Red Hat OpenStack Platform) 구성 매개변수

추가 RHOSP 구성 매개변수는 다음 표에 설명되어 있습니다.

표 15.21. 추가 RHOSP 매개변수

매개변수	설명	값
compute.platform.openstack.rootVolume.size	컴퓨팅 시스템의 경우 루트 볼륨의 크기(GB)입니다. 이 값을 설정하지 않으면 시스템은 임시 스토리지를 사용합니다.	정수 (예: **30**)
compute.platform.openstack.rootVolume.type	컴퓨팅 시스템의 경우 루트 볼륨의 유형입니다.	문자열 (예: **performance**)
controlPlane.platform.openstack.rootVolume.size	컨트롤 플레인 시스템의 경우 루트 볼륨의 크기(GB)입니다. 이 값을 설정하지 않으면 시스템은 임시 스토리지를 사용합니다.	정수 (예: **30**)
controlPlane.platform.openstack.rootVolume.type	컨트롤 플레인 시스템의 경우 루트 볼륨의 유형입니다.	문자열 (예: **performance**)
platform.openstack.cloud	**clouds.yaml** 파일의 클라우드 목록에서 사용할 RHOSP 클라우드의 이름입니다.	문자열 (예: **MyCloud**)

매개변수	설명	값
platform.openstack.externalNetwork	설치에 사용할 RHOSP 외부 네트워크 이름입니다.	문자열 (예: **external)**
platform.openstack.computeFlavor	컨트롤 플레인 및 컴퓨팅 시스템에 사용할 RHOSP 버전입니다. 이 속성은 더 이상 사용되지 않습니다. 모든 머신 풀에 플레이버를 기본값으로 사용하려면 **platform.openstack.defaultMachinePlatform** 속성의 **type** 키 값으로 추가합니다. 각 머신 풀의 플레이버 값을 개별적으로 설정할 수도 있습니다.	문자열 (예: **m1.xlarge)**

15.5.13.5. 선택적 RHOSP 구성 매개변수

선택적 RHOSP 구성 매개변수는 다음 표에 설명되어 있습니다.

표 15.22. 선택적 RHOSP 매개변수

매개변수	설명	값
compute.platform.openstack.additionalNetworkIDs	컴퓨팅 시스템과 관련된 추가 네트워크입니다. 추가 네트워크에는 허용되는 주소 쌍이 생성되지 않습니다.	하나 이상의 UUID 목록을 문자열로 나타냅니다. 예: **fa806b2f-ac49-4bce-b9db-124bc64209bf**
compute.platform.openstack.additionalSecurityGroupIDs	컴퓨팅 시스템과 관련된 추가 보안 그룹입니다.	하나 이상의 UUID 목록을 문자열로 나타냅니다. 예: **7ee219f3-d2e9-48a1-96c2-e7429f1b0da7**

매개변수	설명	값
compute.platform.openstack.zones	머신을 설치할 RHOSP Compute (Nova) 가용성 영역 (AZ). 이 매개 변수가 설정되지 않은 경우 설치 프로그램은 RHOSP 관리자가 구성한 Nova의 기본 설정을 사용합니다. Kuryr를 사용하는 클러스터에서 RHOSP Octavia는 가용성 영역을 지원하지 않습니다. 로드 밸런서 및 Amphora 공급자 드라이버를 사용하는 경우 Amphora VM에 의존하는 OpenShift Container Platform 서비스는이 속성의 값에 따라 생성되지 않습니다.	문자열 목록입니다. 예: **["zone-1", "zone-2"]**.
compute.platform.openstack.rootVolume.zones	컴퓨팅 시스템의 경우 루트 볼륨을 설치할 가용성 영역입니다. 이 매개변수의 값을 설정하지 않으면 설치 프로그램에서 기본 가용성 영역을 선택합니다.	문자열 목록(예: **["zone-1", "zone-2"]**).
controlPlane.platform.openstack.additionalNetworkIDs	컨트롤 플레인 시스템과 관련된 추가 네트워크입니다. 추가 네트워크에는 허용되는 주소 쌍이 생성되지 않습니다.	하나 이상의 UUID 목록을 문자열로 나타냅니다. 예: **fa806b2f-ac49-4bce-b9db-124bc64209bf**
controlPlane.platform.openstack.additionalSecurityGroupIDs	컨트롤 플레인 시스템과 관련된 추가 보안 그룹입니다.	하나 이상의 UUID 목록을 문자열로 나타냅니다. 예: **7ee219f3-d2e9-48a1-96c2-e7429f1b0da7**
controlPlane.platform.openstack.zones	머신을 설치할 RHOSP Compute (Nova) 가용성 영역 (AZ). 이 매개 변수가 설정되지 않은 경우 설치 프로그램은 RHOSP 관리자가 구성한 Nova의 기본 설정을 사용합니다. Kuryr를 사용하는 클러스터에서 RHOSP Octavia는 가용성 영역을 지원하지 않습니다. 로드 밸런서 및 Amphora 공급자 드라이버를 사용하는 경우 Amphora VM에 의존하는 OpenShift Container Platform 서비스는이 속성의 값에 따라 생성되지 않습니다.	문자열 목록입니다. 예: **["zone-1", "zone-2"]**.

매개변수	설명	값
controlPlane.platform.openstack.rootVolume.zones	컨트롤 플레인 시스템의 경우 루트 볼륨을 설치할 가용성 영역입니다. 이 값을 설정하지 않으면 설치 프로그램에서 기본 가용성 영역을 선택합니다.	문자열 목록(예: **["zone-1", "zone-2"]**).
platform.openstack.clusterOSImage	설치 프로그램이 RHCOS 이미지를 다운로드하는 위치입니다. 네트워크가 제한된 환경에서 설치를 수행하려면이 매개 변수를 설정해야 합니다.	HTTP 또는 HTTPS URL (선택 옵션으로 SHA-256 형식의 체크섬을 사용) 예: **http://mirror.example.com/images/rhcos-43.81.201912131630.0-openstack.x86_64.qcow2.gz?sha256=ffebbd68e8a1f2a245ca19522c16c86f67f9ac8e4e0c1f0a812b068b16f7265d**. 이 값은 기존의 Glance 이미지의 이름이 될 수도 있습니다 (예: **my-rhcos**).
platform.openstack.clusterOSImageProperties	Glance의 설치 프로그램에서 업로드된 ClusterOSImage에 추가할 속성입니다. **platform.openstack.clusterOSImage**가 기존 Glance 이미지로 설정된 경우 이 속성은 무시됩니다. 이 속성을 사용하여 노드당 26 PV의 RHOSP의 기본 PV(영구 볼륨) 제한을 초과할 수 있습니다. 제한을 초과하려면 **hw_scsi_model** 속성값을 **virtio-scsi**로, **hw_disk_bus** 값을 **scsi**로 설정합니다. 이 속성을 사용하면 **yes** 값이 있는 **hw_qemu_guest_agent** 속성을 포함하여 QEMU 게스트 에이전트를 활성화할 수도 있습니다.	키-값 문자열 쌍 목록입니다. 예를 들면 **["hw_scsi_model": "virtio-scsi", "hw_disk_bus": "scsi"]**가 있습니다.

platform.openstack.defaultMachinePlatform	기본 시스템 풀 플랫폼 구성입니다.	<pre>{
		  "type": "ml.large",
		  "rootVolume": {
		  "size": 30,
		  "type": "performance"
		  }
		}</pre>

매개변수	설명	값
platform.openstack.ingressFloatingIP	Ingress 포트와 연결할 기존 부동 IP 주소입니다. 이 속성을 사용하려면 **platform.openstack.externalNetwork** 속성도 정의해야 합니다.	IP 주소 (예: **128.0.0.1**)
platform.openstack.apiFloatingIP	로드 밸런서 API와 연결할 기존 부동 IP 주소입니다. 이 속성을 사용하려면 **platform.openstack.externalNetwork** 속성도 정의해야 합니다.	IP 주소 (예: **128.0.0.1**)
platform.openstack.externalDNS	클러스터 인스턴스가 DNS 확인에 사용하는 외부 DNS 서버의 IP 주소입니다.	IP 주소 목록을 문자열로 나타냅니다. 예: **["8.8.8.8", "192.168.1.12"]**.
platform.openstack.machinesSubnet	클러스터 노드가 사용하는 RHOSP 서브넷의 UUID입니다. 이 서브넷에 노드와 가상 IP(VIP) 포트가 생성됩니다. **networking.machineNetwork**의 첫 번째 항목은 **machinesSubnet** 값과 일치해야 합니다. 사용자 지정 서브넷에 배포하는 경우 OpenShift Container Platform 설치 관리자에 외부 DNS 서버를 지정할 수 없습니다. 대신 RHOSP의 서브넷에 DNS를 추가합니다.	문자열의 UUID입니다. 예: **fa806b2f-ac49-4bce-b9db-124bc64209bf**

15.5.13.6. RHOSP 배포의 사용자 지정 서브넷

선택한 **RHOSP(Red Hat OpenStack Platform)** 서브넷에 클러스터를 배포할 수도 있습니다. 서브넷의 **GUID**는 **install-config.yaml** 파일에서 **platform.openstack.machinesSubnet** 값으로 전달됩니다.

이 서브넷은 클러스터의 기본 서브넷으로 사용됩니다. 기본적으로 노드와 포트가 이 서브넷에서 생성됩니다. **platform.openstack.machinesSubnet** 속성의 값을 서브넷의 **UUID**로 설정하여 다른 **RHOSP** 서브넷에서 노드 및 포트를 생성할 수 있습니다.

사용자 지정 서브넷으로 **OpenShift Container Platform** 설치 관리자를 실행하기 전에 구성이 다음 요구 사항을 충족하는지 확인하십시오.

- platform.openstack.machinesSubnet에서 사용하는 서브넷에 DHCP가 활성화되어 있습니다.
- platform.openstack.machinesSubnet의 CIDR은 networking.machineNetwork의 CIDR과 일치합니다.
- 설치 프로그램 사용자에게 고정 IP 주소가 있는 포트를 포함하여 이 네트워크에 포트를 생성할 수 있는 권한이 있습니다.

사용자 지정 서브넷을 사용하는 클러스터에는 다음과 같은 제한 사항이 있습니다.

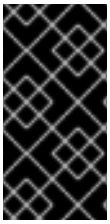
- 유동 IP 주소를 사용하는 클러스터를 설치하려는 경우 platform.openstack.machinesSubnet 서브넷이 externalNetwork 네트워크에 연결된 라우터에 연결되어 있어야 합니다.
- platform.openstack.machinesSubnet 값이 install-config.yaml 파일에 설정된 경우 설치 프로그램에서 RHOSP 머신의 프라이빗 네트워크 또는 서브넷을 생성하지 않습니다.
- 사용자 지정 서브넷과 동시에 platform.openstack.externalDNS 속성을 사용할 수 없습니다. 사용자 지정 서브넷을 사용하는 클러스터에 DNS를 추가하려면 RHOSP 네트워크에서 DNS를 구성합니다.

참고

기본적으로 API VIP는 x.x.x.5를 가져오고 Ingress VIP는 네트워크의 CIDR 블록에서 x.x.x.7을 가져옵니다. 이러한 기본값을 재정의하려면 DHCP 할당 풀 외부에 있는platform.openstack.apiVIP 및 platform.openstack.ingressVIP의 값을 설정합니다.

15.5.13.7. RHOSP용 샘플 사용자 지정 install-config.yaml 파일

이 샘플 install-config.yaml은 가능한 모든 RHOSP(Red Hat OpenStack Platform) 사용자 지정 옵션을 보여줍니다.



중요

이 샘플 파일은 참조용으로만 제공됩니다. **install-config.yaml** 파일은 설치 프로그램을 사용하여 받아야 합니다.

```
apiVersion: v1
baseDomain: example.com
controlPlane:
  name: master
  platform: {}
  replicas: 3
compute:
- name: worker
  platform:
    openstack:
      type: ml.large
  replicas: 3
metadata:
  name: example
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  serviceNetwork:
  - 172.30.0.0/16
  networkType: OpenShiftSDN
platform:
  openstack:
    cloud: mycloud
    externalNetwork: external
    computeFlavor: m1.xlarge
    apiFloatingIP: 128.0.0.1
fips: false
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```

15.5.13.8. 시스템의 사용자 지정 서브넷 설정

설치 프로그램이 기본적으로 사용하는 IP 범위가 OpenShift Container Platform을 설치할 때 생성하는 Neutron 서브넷과 일치하지 않을 수 있습니다. 필요한 경우 설치 구성 파일을 편집하여 새 시스템의 CIDR 값을 업데이트하십시오.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램에서 생성된 **install-config.yaml** 파일이 있습니다.

프로세스

1. 명령줄에서 install-config.yaml이 들어 있는 디렉터리를 찾습니다.

2. 해당 디렉터리에서 스크립트를 실행하여 install-config.yaml 파일을 편집하거나 파일을 수동으로 업데이트합니다.

- 스크립트를 사용하여 값을 설정하려면 다음을 실행합니다.

```
$ python -c '
import yaml;
path = "install-config.yaml";
data = yaml.safe_load(open(path));
data["networking"]["machineNetwork"] = [{"cidr": "192.168.0.0/18"}]; 1
open(path, "w").write(yaml.dump(data, default_flow_style=False))'
```

1 원하는 Neutron 서브넷과 일치하는 값을 삽입합니다(예: 192.0.2.0/24).

- 값을 수동으로 설정하려면 파일을 열고 networking.machineCIDR 값을 원하는 Neutron 서브넷과 일치하는 값으로 설정합니다.

15.5.13.9. 컴퓨팅 시스템 풀 비우기

사용자 인프라를 사용하는 설치를 계속 진행하려면 설치 구성 파일의 컴퓨팅 시스템 수를 0으로 설정합니다. 이러한 시스템은 나중에 수동으로 생성합니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램에서 생성된 install-config.yaml 파일이 있습니다.

프로세스

1. 명령줄에서 install-config.yaml이 들어 있는 디렉터리를 찾습니다.

2. 해당 디렉터리에서 스크립트를 실행하여 install-config.yaml 파일을 편집하거나 파일을 수동으로 업데이트합니다.

- 스크립트를 사용하여 값을 설정하려면 다음을 실행합니다.

```
$ python -c '
import yaml;
path = "install-config.yaml";
data = yaml.safe_load(open(path));
data["compute"][0]["replicas"] = 0;
open(path, "w").write(yaml.dump(data, default_flow_style=False))'
```

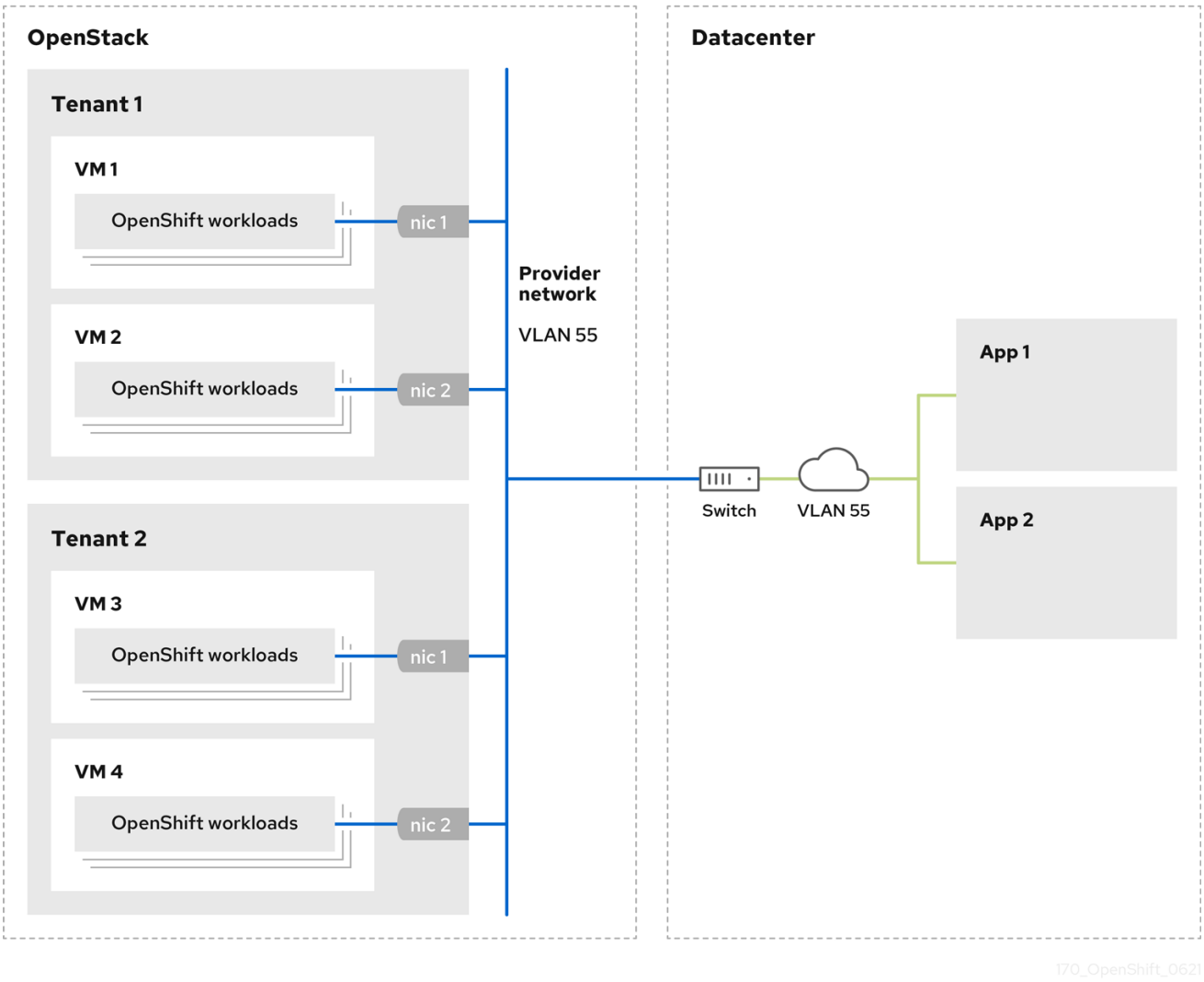
- 값을 수동으로 설정하려면 파일을 열고 **compute.<first entry>.replicas** 값을 **0**으로 설정합니다.

15.5.13.10. RHOSP 공급자 네트워크에서 클러스터 배포

공급자 네트워크의 기본 네트워크 인터페이스를 사용하여 **RHOSP(Red Hat OpenStack Platform)**에 **OpenShift Container Platform** 클러스터를 배포할 수 있습니다. 공급자 네트워크는 일반적으로 프로젝트에 인터넷에 연결하는 데 사용할 수 있는 공용 네트워크에 직접 액세스할 수 있는 권한을 부여하는 데 사용됩니다. 네트워크 생성 프로세스의 일부로 프로젝트 간에 공급자 네트워크를 공유할 수도 있습니다.

RHOSP 공급자 네트워크는 데이터 센터의 기존 실제 네트워크에 직접 매핑됩니다. **RHOSP** 관리자가 이를 생성해야 합니다.

다음 예에서 **OpenShift Container Platform** 워크로드는 공급자 네트워크를 사용하여 데이터 센터에 연결됩니다.



공급자 네트워크에 설치된 **OpenShift Container Platform** 클러스터에는 테넌트 네트워크 또는 유동 **IP** 주소가 필요하지 않습니다. 설치 프로그램에서 설치하는 동안 이러한 리소스를 생성하지 않습니다.

공급자 네트워크 유형에는 **flat**(태그되지 않음) 및 **VLAN**(802.1Q 태그)이 포함됩니다.

참고

클러스터는 네트워크 유형에서 허용하는 만큼의 공급자 네트워크 연결을 지원할 수 있습니다. 예를 들어 **VLAN** 네트워크는 일반적으로 최대 **4096**개의 연결을 지원합니다.

RHOSP 설명서에서 공급자 및 테넌트 네트워크에 대해 자세히 알아볼 수 있습니다.

15.5.13.10.1. 클러스터 설치를 위한 RHOSP 공급자 네트워크 요구 사항

OpenShift Container Platform 클러스터를 설치하기 전에 RHOSP(Red Hat OpenStack Platform) 배포 및 공급자 네트워크가 다음과 같은 여러 조건을 충족해야 합니다.

- RHOSP 네트워킹 서비스(Neutron)가 활성화되어 RHOSP 네트워킹 API를 통해 액세스할 수 있습니다.
- RHOSP 네트워킹 서비스에는 포트 보안 및 허용되는 주소 쌍 확장 기능이 활성화되어 있습니다.
- 공급자 네트워크는 다른 테넌트와 공유할 수 있습니다.

작은 정보

openstack network create 명령을 --share 플래그와 함께 사용하여 공유할 수 있는 네트워크를 생성합니다.

- 클러스터를 설치하는 데 사용하는 RHOSP 프로젝트는 공급자 네트워크와 적절한 서브넷을 소유해야 합니다.

작은 정보

"openshift"라는 프로젝트의 네트워크를 만들려면 다음 명령을 입력합니다.

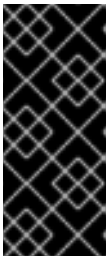
```
$ openstack network create --project openshift
```

"openshift"라는 프로젝트의 서브넷을 만들려면 다음 명령을 입력합니다.

```
$ openstack subnet create --project openshift
```

RHOSP에서 네트워크 생성에 대한 자세한 내용은 공급자 네트워크 설명서를 참조하십시오.

admin 사용자가 클러스터를 소유하는 경우 해당 사용자로 설치 프로그램을 실행하여 네트워크에서 포트를 생성해야 합니다.



중요

공급자 네트워크는 클러스터를 생성하는 데 사용되는 RHOSP 프로젝트에서 소유해야 합니다. RHOSP 컴퓨팅 서비스(Nova)는 해당 네트워크의 포트를 요청할 수 없습니다.

- 공급자 네트워크가 기본적으로 RHOSP 메타데이터 서비스 IP 주소 169.254.169.254에 연결할 수 있는지 확인합니다.

  RHOSP SDN 및 네트워킹 서비스 구성에 따라 서브넷을 생성할 때 경로를 제공해야 할 수 있습니다. 예를 들면 다음과 같습니다.

```
$ openstack subnet create --dhcp --host-route
destination=169.254.169.254/32,gateway=192.0.2.2 ...
```

- 선택 사항: 네트워크를 보호하려면 단일 프로젝트에 대한 네트워크 액세스를 제한하는 역할 기반 액세스 제어(RBAC) 규칙을 생성합니다.

15.5.13.10.2. 공급자 네트워크에 기본 인터페이스가 있는 클러스터 배포

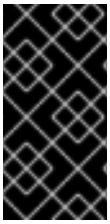
RHOSP(Red Hat OpenStack Platform) 공급자 네트워크에 기본 네트워크 인터페이스가 있는 OpenShift Container Platform 클러스터를 배포할 수 있습니다.

- RHOSP(Red Hat OpenStack Platform) 배포는 "클러스터 설치를 위한 RHOSP 공급자 네트워크 요구 사항"에 설명된 대로 구성됩니다.

프로세스

1. 텍스트 편집기에서 install-config.yaml 파일을 엽니다.
2. platform.openstack.apiVIP 속성의 값을 API VIP의 IP 주소로 설정합니다.
3. platform.openstack.ingressVIP 속성 값을 Ingress VIP의 IP 주소로 설정합니다.
4. platform.openstack.machinesSubnet 속성 값을 공급자 네트워크 서브넷의 UUID로 설정합니다.

5. **networking.machineNetwork.cidr** 속성 값을 공급자 네트워크 서브넷의 **CIDR** 블록으로 설정합니다.



중요

platform.openstack.apiVIP 및 **platform.openstack.ingressVIP** 속성은 모두 **networking.machineNetwork.cidr** 블록에서 할당되지 않은 **IP** 주소여야 합니다.

RHOSP 공급자 네트워크를 사용하는 클러스터의 설치 구성 파일 섹션

```
...
platform:
  openstack:
    apiVIP: 192.0.2.13
    ingressVIP: 192.0.2.23
    machinesSubnet: fa806b2f-ac49-4bce-b9db-124bc64209bf
    # ...
networking:
  machineNetwork:
  - cidr: 192.0.2.0/24
```

주의

기본 네트워크 인터페이스에 공급자 네트워크를 사용하는 동안 **platform.openstack.externalNetwork** 또는 **platform.openstack.externalDNS** 매개변수를 설정할 수 없습니다.

클러스터를 배포할 때 설치 프로그램에서 **install-config.yaml** 파일을 사용하여 공급자 네트워크에 클러스터를 배포합니다.

작은 정보

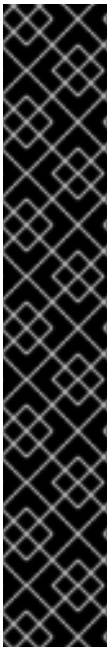
공급자 네트워크를 포함한 다른 네트워크를 **platform.openstack.additionalNetworkIDs** 목록에 추가할 수 있습니다.

클러스터를 배포한 후 **Pod**를 추가 네트워크에 연결할 수 있습니다. 자세한 내용은 여러 네트워크 이해를 참조하십시오.

15.5.14. Kubernetes 매니페스트 및 Ignition 설정 파일 생성

일부 클러스터 정의 파일을 수정하고 클러스터 시스템을 수동으로 시작해야 하므로 클러스터가 시스템을 구성하는 데 필요한 **Kubernetes** 매니페스트 및 **Ignition** 구성 파일을 사용자가 생성해야 합니다.

설치 구성 파일은 **Kubernetes** 매니페스트로 변환됩니다. 매니페스트는 나중에 클러스터 머신을 구성하는 데 사용되는 **Ignition** 구성 파일로 래핑됩니다.



중요

- **OpenShift Container Platform** 설치 프로그램에서 생성하는 **Ignition** 구성 파일에 **24**시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 **24**시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 **kubelet** 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(**CSR**)을 수동으로 승인해야 합니다. 자세한 내용은 *만료된 컨트롤 플레인 인증서에서 복구* 문서를 참조하십시오.
- 클러스터를 설치한 후 **24**시간에서 **22**시간까지의 인증서가 교체되기 때문에 생성된 후 **12**시간 이내에 **Ignition** 구성 파일을 사용하는 것이 좋습니다. **12**시간 이내에 **Ignition** 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

사전 요구 사항

- **OpenShift Container Platform** 설치 프로그램을 가져오셨습니다.
- **install-config.yaml** 설치 구성 파일을 생성하셨습니다.

프로세스

1. OpenShift Container Platform 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터에 대한 Kubernetes 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 <installation_directory>는 사용자가 만든 install-config.yaml 파일이 포함된 설치 디렉터리를 지정합니다.

2. 컨트롤 플레인 시스템 및 컴퓨팅 머신 세트를 정의하는 Kubernetes 매니페스트 파일을 제거합니다.

```
$ rm -f openshift/99_openshift-cluster-api_master-machines-*.yaml
openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

이러한 리소스는 사용자가 직접 생성하고 관리하기 때문에 초기화할 필요가 없습니다.

- 시스템 API로 머신 세트 파일을 보존하여 컴퓨팅 시스템을 생성할 수 있지만 사용자 환경과 일치하도록 해당 참조를 업데이트해야 합니다.

3. <installation_directory>/manifests/cluster-scheduler-02-config.yml Kubernetes 매니페스트 파일의 mastersSchedulable 매개변수가 false로 설정되어 있는지 확인합니다. 이 설정으로 인해 컨트롤 플레인 머신에서 포드가 예약되지 않습니다.

   a. <installation_directory>/manifests/cluster-scheduler-02-config.yml 파일을 엽니다.

   b. mastersSchedulable 매개변수를 찾아서 값을 False로 설정되어 있는지 확인합니다.

   c. 파일을 저장하고 종료합니다.

4. Ignition 구성 파일을 생성하려면 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행합니다.

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1

<installation_directory>는 동일한 설치 디렉터리를 지정합니다.

설치 디렉터리의 부트스트랩, 컨트롤 플레인 및 컴퓨팅 노드에 대한 Ignition 구성 파일이 생성됩니다. kubeadmin-password 및 kubeconfig 파일은 ./<installation_directory>/auth 디렉터리에 생성됩니다.

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

5. 메타데이터 파일의 infraID 키를 환경 변수로 내보냅니다.

```
$ export INFRA_ID=$(jq -r .infraID metadata.json)
```

작은 정보

metadata.json에서 infraID 키를 추출하여 사용자가 생성하는 모든 RHOSP 리소스의 접두사로 사용합니다. 이렇게 하면 동일한 프로젝트에서 여러 배포를 수행할 때 이름 충돌을 방지할 수 있습니다.

15.5.15. 부트스트랩 Ignition 파일 준비

OpenShift Container Platform 설치 프로세스는 부트스트랩 Ignition 구성 파일에서 생성되는 부트스트랩 시스템에 의존합니다.

파일을 편집하고 업로드합니다. 그런 다음 RHOSP(Red Hat OpenStack Platform)에서 기본 파일을 다운로드하는 데 사용하는 보조 부트스트랩 Ignition 구성 파일을 만듭니다.

사전 요구 사항

- 설치 관리자에서 생성되는 부트스트랩 Ignition 파일, bootstrap.ign이 있습니다.

- 설치 관리자 메타데이터 파일의 인프라 ID는 환경 변수($ INFRA_ID)로 설정됩니다.
  - 변수가 설정되지 않은 경우 Kubernetes 매니페스트 및 Ignition 구성 파일 생성을 참조하십시오.
- 부트스트랩 Ignition 파일을 저장할 수 있는 HTTP(S) 액세스 가능 방법이 있습니다.
  - 문서화된 프로시저에는 RHOSP 이미지 서비스(Glance)를 사용하지만 RHOSP 스토리지 서비스(Swift), Amazon S3, 내부 HTTP 서버 또는 애드혹 Nova 서버를 사용할 수도 있습니다.

프로세스

1. 다음 Python 스크립트를 실행합니다. 이 스크립트는 부트스트랩 Ignition 파일을 수정하여 호스트 이름 및 사용 가능한 경우 실행 시 CA 인증서 파일을 설정합니다.

```
import base64
import json
import os

with open('bootstrap.ign', 'r') as f:
    ignition = json.load(f)

files = ignition['storage'].get('files', [])

infra_id = os.environ.get('INFRA_ID', 'openshift').encode()
hostname_b64 = base64.standard_b64encode(infra_id + b'-
bootstrap\n').decode().strip()
files.append(
{
    'path': '/etc/hostname',
    'mode': 420,
    'contents': {
        'source': 'data:text/plain;charset=utf-8;base64,' + hostname_b64
    }
})

ca_cert_path = os.environ.get('OS_CACERT', '')
if ca_cert_path:
    with open(ca_cert_path, 'r') as f:
        ca_cert = f.read().encode()
        ca_cert_b64 = base64.standard_b64encode(ca_cert).decode().strip()

    files.append(
    {
        'path': '/opt/openshift/tls/cloud-ca-cert.pem',
```

```
        'mode': 420,
        'contents': {
            'source': 'data:text/plain;charset=utf-8;base64,' + ca_cert_b64
        }
    })

ignition['storage']['files'] = files;

with open('bootstrap.ign', 'w') as f:
    json.dump(ignition, f)
```

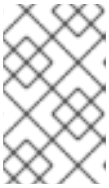
2. RHOSP CLI를 사용하여 부트스트랩 Ignition 파일을 사용하는 이미지를 생성합니다.

```
$ openstack image create --disk-format=raw --container-format=bare --file
bootstrap.ign <image_name>
```

3. 이미지의 세부 사항을 가져옵니다.

```
$ openstack image show <image_name>
```

file 값을 기록해 둡니다. 해당 패턴은 v2/images/<image_ID>/file입니다.



참고

생성된 이미지가 활성화되어 있는지 확인합니다.

4. 이미지 서비스의 공용 주소를 검색합니다.

```
$ openstack catalog show image
```

5. 공용 주소를 이미지 file 값과 결합하고 그 결과를 저장 위치로 저장합니다. 위치 패턴은 <image_service_public_URL>/v2/images/<image_ID>/file입니다.

6. 인증 토큰을 생성하고 토큰 ID를 저장합니다.

```
$ openstack token issue -c id -f value
```

7. $INFRA_ID-bootstrap-ignition.json 파일에 다음 내용을 삽입하고 사용자 값과 일치하도록 자리 표시자를 편집합니다.

```
{
 "ignition": {
  "config": {
   "merge": [{
    "source": "<storage_url>", 1
    "httpHeaders": [{
     "name": "X-Auth-Token", 2
     "value": "<token_ID>" 3
    }]
   }]
  },
  "security": {
   "tls": {
    "certificateAuthorities": [{
     "source": "data:text/plain;charset=utf-8;base64,<base64_encoded_certificate>"
4
    }]
   }
  },
  "version": "3.2.0"
 }
}
```

1 ignition.config.merge.source 값을 부트스트랩 Ignition 파일 스토리지 URL로 바꿉니다.

2 httpHeaders의 name을 "X-Auth-Token"으로 설정합니다.

3 httpHeaders의 value를 토큰의 ID로 설정합니다.

4 부트스트랩 Ignition 파일 서버가 자체 서명 인증서를 사용하는 경우 base64 인코딩 인증서를 포함합니다.

8. 보조 Ignition 구성 파일을 저장합니다.

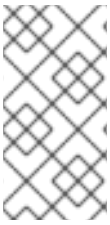
부트스트랩 Ignition 데이터는 설치 중에 RHOSP로 전달됩니다.

주의

부트스트랩 Ignition 파일에는 clouds.yaml 자격 증명과 같은 민감한 정보가 들어 있습니다. 파일을 안전한 곳에 저장하고 설치 프로세스를 완료한 후에는 삭제해야 합니다.

15.5.16. RHOSP에서 컨트롤 플레인 Ignition 구성 파일 생성

자체 인프라의 RHOSP(Red Hat OpenStack Platform)에 OpenShift Container Platform을 설치하려면 컨트롤 플레인 Ignition 구성 파일이 필요합니다. 구성 파일은 여러 개 만들어야 합니다.



참고

부트스트랩 Ignition 구성과 마찬가지로 각 컨트롤 플레인 시스템의 호스트 이름을 명시적으로 정의해야 합니다.

사전 요구 사항

- 설치 프로그램 메타데이터 파일의 인프라 ID는 환경 변수($ INFRA_ID)로 설정됩니다.
  - 변수가 설정되지 않은 경우 “Kubernetes 매니페스트 및 Ignition 구성 파일 생성”을 참조하십시오.

프로세스

- 명령줄에서 다음 Python 스크립트를 실행합니다.

```
$ for index in $(seq 0 2); do
    MASTER_HOSTNAME="$INFRA_ID-master-$index\n"
    python -c "import base64, json, sys;
ignition = json.load(sys.stdin);
storage = ignition.get('storage', {});
files = storage.get('files', []);
files.append({'path': '/etc/hostname', 'mode': 420, 'contents': {'source':
'data:text/plain;charset=utf-8;base64,' +
base64.standard_b64encode(b'$MASTER_HOSTNAME').decode().strip(), 'verification':
{}}, 'filesystem': 'root'});
storage['files'] = files;
```

```
ignition['storage'] = storage
json.dump(ignition, sys.stdout)" <master.ign >"$INFRA_ID-master-$index-
ignition.json"
done
```

이제 <INFRA_ID>-master-0-ignition.json, <INFRA_ID>-master-1-ignition.json, <INFRA_ID>-master-2-ignition.json의 세 가지 컨트롤 플레인 Ignition 파일이 있습니다.

15.5.17. RHOSP에서 네트워크 리소스 생성

자체 인프라의 RHOSP(Red Hat OpenStack Platform)에 OpenShift Container Platform을 설치하는 데 필요한 네트워크 리소스를 생성합니다. 시간을 절약하려면 보안 그룹, 네트워크, 서브넷, 라우터 및 포트를 생성하는 제공된 Ansible 플레이북을 실행합니다.

사전 요구 사항

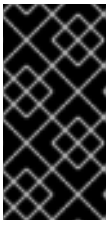
- Python 3가 시스템에 설치되어 있습니다.
- "플레이북 종속 항목 다운로드"에서 모듈을 다운로드했습니다.
- "설치 플레이북 다운로드"에서 플레이북을 다운로드했습니다.

프로세스

1. 선택사항: inventory.yaml 플레이북에 외부 네트워크값을 추가합니다.

inventory.yaml Ansible 플레이북의 외부 네트워크 값 예

```
...
    # The public network providing connectivity to the cluster. If not
    # provided, the cluster external connectivity must be provided in another
    # way.

    # Required for os_api_fip, os_ingress_fip, os_bootstrap_fip.
    os_external_network: 'external'
...
```



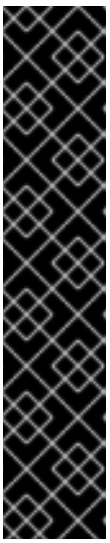
중요

inventory.yaml 파일에 **os_external_network** 값을 지정하지 않은 경우 **VM**이 **Glance** 및 외부 연결에 직접 액세스할 수 있는지 확인해야 합니다.

2. **선택사항: 외부 네트워크 및 부동 IP(FIP) 주소값을 inventory.yaml 플레이북에 추가합니다.**

inventory.yaml Ansible 플레이북의 FIP 값 예

```
...
# OpenShift API floating IP address. If this value is non-empty, the
# corresponding floating IP will be attached to the Control Plane to
# serve the OpenShift API.
os_api_fip: '203.0.113.23'

# OpenShift Ingress floating IP address. If this value is non-empty, the
# corresponding floating IP will be attached to the worker nodes to serve
# the applications.
os_ingress_fip: '203.0.113.19'

# If this value is non-empty, the corresponding floating IP will be
# attached to the bootstrap machine. This is needed for collecting logs
# in case of install failure.
os_bootstrap_fip: '203.0.113.20'
```



중요

os_api_fip 및 **os_ingress_fip**에 대한 값을 정의하지 않으면 설치 후 네트워크 구성을 수행해야 합니다.

os_bootstrap_fip의 값을 정의하지 않으면 설치 프로그램이 실패한 설치에서 디버깅 정보를 다운로드할 수 없습니다.

자세한 정보는 **"환경에 대한 액세스 활성화"**를 참조하십시오.

3. **명령줄에서 security-groups.yaml 플레이북을 실행하여 보안 그룹을 생성합니다.**

```
$ ansible-playbook -i inventory.yaml security-groups.yaml
```

4. 명령줄에서 **network.yaml** 플레이북을 실행하여 네트워크, 서브넷 및 라우터를 생성합니다.

```
$ ansible-playbook -i inventory.yaml network.yaml
```

5. 선택사항: **Nova** 서버가 사용하는 기본 해결 프로그램을 제어하려면 **RHOSP CLI** 명령을 실행합니다.

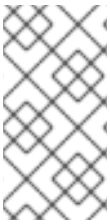
```
$ openstack subnet set --dns-nameserver <server_1> --dns-nameserver <server_2>
"$INFRA_ID-nodes"
```

선택적으로 생성된 **inventory.yaml** 파일을 사용하여 설치를 사용자 지정할 수 있습니다. 예를 들어 베어 메탈 머신을 사용하는 클러스터를 배포할 수 있습니다.

15.5.17.1. 베어 메탈 머신으로 클러스터 배포

클러스터가 베어 메탈 머신을 사용하도록 하려면 **inventory.yaml** 파일을 수정합니다. 클러스터는 베어 메탈에서 컨트롤 플레인 및 컴퓨팅 머신 모두를 실행하거나 컴퓨팅 머신만으로 실행할 수 있습니다.

Kuryr를 사용하는 클러스터에서 베어 메탈 컴퓨팅 머신이 지원되지 않습니다.



참고

install-config.yaml 파일에서 베어 메탈 작업자가 유동 **IP** 주소를 지원하는지 여부를 반영하는지 확인하십시오.

사전 요구 사항

- **RHOSP Bare Metal** 서비스(Ironic)가 활성화되어 **RHOSP Compute API**를 통해 액세스할 수 있습니다.
- 베어 메탈은 **RHOSP** 플레이버로 사용할 수 있습니다.
- **RHOSP** 네트워크는 **VM** 및 베어 메탈 서버 연결을 모두 지원합니다.

- 네트워크 구성이 공급자 네트워크를 사용하지 않습니다. 공급자 네트워크는 지원되지 않습니다.

- 기존 네트워크에 머신을 배포하려면 **RHOSP** 서브넷이 프로비저닝됩니다.

- 설치 관리자 프로비저닝 네트워크에 머신을 배포하려는 경우 **RHOSP Bare Metal** 서비스(Ironic)가 테넌트 네트워크에서 실행되는 **PXE(Preboot eXecution Environment)** 부팅 머신을 수신하고 상호 작용할 수 있습니다.

- **OpenShift Container Platform** 설치 프로세스의 일부로 **inventory.yaml** 파일을 생성하셨습니다.

프로세스

1. **inventory.yaml** 파일에서 머신의 플레이버를 편집합니다.

   a. 베어 메탈 컨트롤 플레인 머신을 사용하려면 **os_flavor_master** 값을 베어 메탈 플레이버로 변경합니다.

   b. **os_flavor_worker**의 값을 베어 메탈 플레이버로 변경합니다.

베어 메탈 **inventory.yaml** 파일 예

```
all:
  hosts:
    localhost:
      ansible_connection: local
      ansible_python_interpreter: "{{ansible_playbook_python}}"

      # User-provided values
      os_subnet_range: '10.0.0.0/16'
      os_flavor_master: 'my-bare-metal-flavor' 1
      os_flavor_worker: 'my-bare-metal-flavor' 2
      os_image_rhcos: 'rhcos'
      os_external_network: 'external'
...
```

1

베어 메탈 컨트롤 플레인 머신이 필요한 경우 이 값을 베어 메탈 플레이버로 변경합니다.

2

컴퓨팅 머신에 사용할 베어 메탈 플레이버로 이 값을 변경합니다.

업데이트된 **inventory.yaml** 파일을 사용하여 설치 프로세스를 완료합니다. 배포 중에 생성된 머신은 파일에 추가한 플레이버를 사용합니다.

참고

설치 프로그램은 베어 메탈 머신이 부팅될 때까지 대기하는 동안 시간이 초과될 수 있습니다.

설치 프로그램이 시간 초과되면 설치 프로그램의 **wait-for** 명령을 사용하여 배포를 다시 시작한 다음 완료합니다. 예를 들면 다음과 같습니다.

```
./openshift-install wait-for install-complete --log-level debug
```

15.5.18. RHOSP에서 부트스트랩 시스템 생성

부트스트랩 시스템을 생성하고 **RHOSP(Red Hat OpenStack Platform)**에서 실행하는 데 필요한 네트워크 액세스 권한을 부여합니다. **Red Hat**은 이 프로세스를 간소화하기 위해 실행하는 **Ansible** 플레이북을 제공합니다.

사전 요구 사항

- "플레이북 종속 항목 다운로드"에서 모듈을 다운로드했습니다.
- "설치 플레이북 다운로드"에서 플레이북을 다운로드했습니다.
- **inventory.yaml, common.yaml** 및 **bootstrap.yaml Ansible** 플레이북이 공통 디렉터리에

있습니다.

- 설치 프로그램에서 생성된 **metadata.json** 파일이 **Ansible** 플레이북과 동일한 디렉터리에 있습니다.

프로세스

1. 명령줄에서 작업 디렉터리를 플레이북의 위치로 변경합니다.
2. 명령줄에서 **bootstrap.yaml** 플레이북을 실행합니다.

```
$ ansible-playbook -i inventory.yaml bootstrap.yaml
```

3. 부트스트랩 서버가 활성화된 후 로그를 보고 **Ignition** 파일이 수신되었는지 확인합니다.

```
$ openstack console log show "$INFRA_ID-bootstrap"
```

15.5.19. RHOSP에 컨트롤 플레인 시스템 생성

사용자가 생성한 **Ignition** 구성 파일을 사용하여 세 개의 컨트롤 플레인 시스템을 생성합니다. **Red Hat**은 이 프로세스를 간소화하기 위해 실행하는 **Ansible** 플레이북을 제공합니다.

사전 요구 사항

- "플레이북 종속 항목 다운로드"에서 모듈을 다운로드했습니다.
- "설치 플레이북 다운로드"에서 플레이북을 다운로드했습니다.
- 설치 프로그램 메타데이터 파일의 인프라 **ID**는 환경 변수(**$ INFRA_ID**)로 설정됩니다.
- **inventory.yaml, common.yaml** 및 **control-plane.yaml Ansible** 플레이북이 공통 디렉터리에 있습니다.
- "컨트롤 플레인 **Ignition** 구성 파일 생성"에서 생성된 세 개의 **Ignition** 파일이 있습니다.

프로세스

1. 명령줄에서 작업 디렉터리를 플레이북의 위치로 변경합니다.

2. 컨트롤 플레인 Ignition 구성 파일이 작업 디렉터리에 없으면 파일을 디렉터리에 복사합니다.

3. 명령줄에서 control-plane.yaml 플레이북을 실행합니다.

```
$ ansible-playbook -i inventory.yaml control-plane.yaml
```

4. 다음 명령을 실행하여 부트스트랩 프로세스를 모니터링합니다.

```
$ openshift-install wait-for bootstrap-complete
```

컨트롤 플레인 시스템이 실행 중이고 클러스터에 연결되었음을 확인하는 메시지가 표시됩니다.

```
INFO API v1.22.1 up
INFO Waiting up to 30m0s for bootstrapping to complete...
...
INFO It is now safe to remove the bootstrap resources
```

15.5.20. CLI를 사용하여 클러스터에 로그인

클러스터 kubeconfig 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. kubeconfig 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- oc CLI를 설치했습니다.

프로세스

1.

1. kubeadmin 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 oc 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

15.5.21. RHOSP에서 부트스트랩 리소스 삭제

더 이상 필요하지 않은 부트스트랩 리소스는 삭제합니다.

사전 요구 사항

- "플레이북 종속 항목 다운로드"에서 모듈을 다운로드했습니다.
- "설치 플레이북 다운로드"에서 플레이북을 다운로드했습니다.
- inventory.yaml, common.yaml 및 down-bootstrap.yaml Ansible 플레이북이 공통 디렉터리에 있습니다.
- 컨트롤 플레인 시스템이 실행 중입니다.
  - 시스템 상태를 모르는 경우 "클러스터 상태 확인"을 참조하십시오.

서드파티 네트워크 또는 정적 플러스의 경우 플러스의 네트워크 를 참조하십시오.

프로세스

1. 명령줄에서 작업 디렉터리를 플레이북의 위치로 변경합니다.

2. 명령줄에서 down-bootstrap.yaml 플레이북을 실행합니다.

```
$ ansible-playbook -i inventory.yaml down-bootstrap.yaml
```

부트스트랩 포트, 서버 및 부동 IP 주소가 삭제됩니다.

주의

이전에 부트스트랩 Ignition 파일 URL을 비활성화하지 않은 경우 지금 비활성화 합니다.

15.5.22. RHOSP에서 컴퓨팅 시스템 생성

컨트롤 플레인을 가동시킨 후 컴퓨팅 시스템을 만듭니다. Red Hat은 이 프로세스를 간소화하기 위해 실행하는 Ansible 플레이북을 제공합니다.

사전 요구 사항

- "플레이북 종속 항목 다운로드"에서 모듈을 다운로드했습니다.
- "설치 플레이북 다운로드"에서 플레이북을 다운로드했습니다.
- inventory.yaml, common.yaml 및 compute-nodes.yaml Ansible 플레이북이 공통 디렉터리에 있습니다.
- 설치 프로그램에서 생성된 metadata.json 파일이 Ansible 플레이북과 동일한 디렉터리에 있습니다.

- 컨트롤 플레인이 활성화되었습니다.

프로세스

1. 명령줄에서 작업 디렉터리를 플레이북의 위치로 변경합니다.
2. 명령줄에서 플레이북을 실행합니다.

```
$ ansible-playbook -i inventory.yaml compute-nodes.yaml
```

다음 단계

- 시스템의 인증서 서명 요청을 승인합니다.

15.5.23. 머신의 인증서 서명 요청 승인

클러스터에 시스템을 추가하면 추가한 시스템별로 보류 중인 인증서 서명 요청(CSR)이 두 개씩 생성됩니다. 이러한 CSR이 승인되었는지 확인해야 하며, 필요한 경우 이를 직접 승인해야 합니다. 클라이언트 요청을 먼저 승인한 다음 서버 요청을 승인해야 합니다.

사전 요구 사항

- 클러스터에 시스템을 추가했습니다.

프로세스

1. 클러스터가 시스템을 인식하는지 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0 Ready    master  63m  v1.22.1
master-1 Ready    master  63m  v1.22.1
```

```
master-2  Ready    master  64m  v1.22.1
```

출력에 생성된 모든 시스템이 나열됩니다.

참고

이전 출력에는 일부 CSR이 승인될 때까지 컴퓨팅 노드(작업자 노드라고도 함)가 포함되지 않을 수 있습니다.

2. 보류 중인 CSR을 검토하고 클러스터에 추가한 각 시스템에 대해 Pending 또는 Approved 상태의 클라이언트 및 서버 요청이 표시되는지 확인합니다.

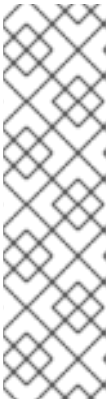
```
$ oc get csr
```

출력 예

```
NAME        AGE    REQUESTOR                                                   CONDITION
csr-8b2br   15m    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

예에서는 두 시스템이 클러스터에 참여하고 있습니다. 목록에는 승인된 CSR이 더 많이 나타날 수도 있습니다.

3. CSR이 승인되지 않은 경우, 추가된 시스템에 대한 모든 보류 중인 CSR이 Pending 상태로 전환된 후 클러스터 시스템의 CSR을 승인합니다.



참고

CSR은 교체 주기가 자동으로 만료되므로 클러스터에 시스템을 추가한 후 1 시간 이내에 CSR을 승인하십시오. 한 시간 내에 승인하지 않으면 인증서가 교체되고 각 노드에 대해 두 개 이상의 인증서가 표시됩니다. 이러한 인증서를 모두 승인해야 합니다. 클라이언트 CSR이 승인되면 Kubelet은 인증서에 대한 보조 CSR을 생성하므로 수동 승인이 필요합니다. 그러면 Kubelet에서 동일한 매개변수를 사용하여 새 인증서를 요청하는 경우 인증서 갱신 요청은 machine-approver에 의해 자동으로 승인됩니다.

참고

베어 메탈 및 기타 사용자 프로비저닝 인프라와 같이 머신 API를 사용하도록 활성화되지 않는 플랫폼에서 실행되는 클러스터의 경우 CSR(Kubelet service Certificate Request)을 자동으로 승인하는 방법을 구현해야 합니다. 요청이 승인되지 않으면 API 서버가 kubelet에 연결될 때 서비스 인증서가 필요하므로 oc exec, oc rsh, oc logs 명령을 성공적으로 수행할 수 없습니다. Kubelet 엔드 포인트에 연결하는 모든 작업을 수행하려면 이 인증서 승인이 필요합니다. 이 방법은 새 CSR을 감시하고 CSR이 system:node 또는 system:admin 그룹의 node-bootstrapper 서비스 계정에 의해 제출되었는지 확인하고 노드의 ID를 확인합니다.

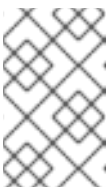
- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 <csr_name>은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```



참고

일부 Operator는 일부 CSR이 승인될 때까지 사용할 수 없습니다.

4. 이제 클라이언트 요청이 승인되었으므로 클러스터에 추가한 각 머신의 서버 요청을 검토해야 합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal        Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal        Pending
...
```

5. 나머지 CSR이 승인되지 않고 Pending 상태인 경우 클러스터 머신의 CSR을 승인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 <csr_name>은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

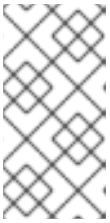
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs oc adm certificate approve
```

6. 모든 클라이언트 및 서버 CSR이 승인된 후 머신은 Ready 상태가 됩니다. 다음 명령을 실행하여 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0  Ready    master  73m  v1.22.1
master-1  Ready    master  73m  v1.22.1
master-2  Ready    master  74m  v1.22.1
worker-0  Ready    worker  11m  v1.22.1
worker-1  Ready    worker  11m  v1.22.1
```



참고

머신이 **Ready** 상태로 전환하는 데 서버 **CSR**의 승인 후 몇 분이 걸릴 수 있습니다.

추가 정보

- **CSR**에 대한 자세한 내용은 인증서 서명 요청을 참조하십시오.

15.5.24. 설치 성공 확인

OpenShift Container Platform 설치가 완료되었는지 확인합니다.

사전 요구 사항

- 설치 프로그램(**openshift-install**)이 있습니다.

프로세스

- 명령줄에 다음을 입력합니다.

```
$ openshift-install --log-level debug wait-for install-complete
```

콘솔 **URL**과 관리자의 로그인 정보가 출력됩니다.

15.5.25. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 **Telemetry** 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결

되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

15.5.26. 다음 단계

- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.
- 노드 포트에 대한 외부 액세스를 활성화해야 하는 경우 노드 포트를 사용하여 Ingress 클러스터 트래픽을 구성합니다.
- 유동 IP 주소를 통한 애플리케이션 트래픽을 허용하도록 RHOSP를 구성하지 않은 경우 유동 IP 주소로 RHOSP 액세스를 구성합니다.

15.6. 사용자 인프라의 KURYR로 OPENSTACK에 클러스터 설치

OpenShift Container Platform 버전 4.9에서는 사용자 프로비저닝 인프라에서 실행되는 클러스터를 RHOSP(Red Hat OpenStack Platform)에 설치할 수 있습니다.

자체 인프라를 사용하면 클러스터를 기존 인프라 및 수정 사항과 통합할 수 있습니다. 이 프로세스에서는 Nova 서버, Neutron 포트, 보안 그룹과 같은 모든 RHOSP 리소스를 생성해야 하므로 설치 관리자 프로비저닝 설치보다 사용자가 수행해야 하는 작업이 더 많습니다. 그러나 Red Hat은 배포 프로세스에 도움이 되는 Ansible 플레이북을 제공합니다.

15.6.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.
- OpenShift Container Platform 4.9가 OpenShift 클러스터에 지원되는 플랫폼 섹션을 사용하여 현재 RHOSP 버전과 호환되는지 확인했습니다. RHOSP 지원 매트릭스의 OpenShift Container Platform을 확인하여 여러 버전의 플랫폼 지원을 비교할 수도 있습니다.
- OpenShift Container Platform을 설치할 RHOSP 계정을 준비합니다.
- 설치 프로그램을 실행하는 시스템에서 다음을 준비합니다.
  - 설치 프로세스에서 만드는 파일을 저장할 수 있는 단일 디렉터리
  - Python 3

15.6.2. Kuryr SDN 정보

Kuryr 는 Neutron 및 Octavia RHOSP(Red Hat OpenStack Platform) 서비스를 사용하여 Pod 및 서비스에 네트워킹을 제공하는 컨테이너 네트워크 인터페이스(CNI) 플러그인 솔루션입니다.

Kuryr와 OpenShift Container Platform 통합은 주로 RHOSP VM에서 실행되는 OpenShift Container Platform 클러스터를 위해 설계되었습니다. Kuryr는 OpenShift Container Platform Pod를 RHOSP SDN에 연결하여 네트워크 성능을 향상시킵니다. 또한 Pod와 RHOSP 가상 인스턴스 간의 상호 연결성을 제공합니다.

Kuryr 구성 요소는 openshift-kuryr 네임스페이스를 사용하여 OpenShift Container Platform에서 Pod로 설치됩니다.

- kuryr-controller- - master 노드에 설치된 단일 서비스 인스턴스로, OpenShift Container Platform에서 Deployment 개체로 모델링됩니다.
- kuryr- cni - 각 OpenShift Container Platform 노드에서 Kuryr를 CNI 드라이버로 설치 및 구성하는 컨테이너로, OpenShift Container Platform에서 DaemonSet 개체로 모델링됩니다.

Kuryr 컨트롤러는 OpenShift Container Platform API 서버에서 Pod, 서비스 및 네임스페이스 생성, 업데이트 및 삭제 이벤트를 감시합니다. OpenShift Container Platform API 호출을 Neutron 및 Octavia의 해당 개체에 매핑합니다. 즉 Neutron 트렁크 포트 기능을 구현하는 모든 네트워크 솔루션을 사용하여 Kuryr를 통해 OpenShift Container Platform을 지원할 수 있습니다. 여기에는 OVS(Open vSwitch), OVN(Open Virtual Network)과 같은 오픈 소스 솔루션과 Neutron 호환 상용 SDN이 포함됩니다.

Kuryr는 캡슐화된 RHOSP 테넌트 네트워크에서의 OpenShift Container Platform 배포에 권장되며 RHOSP 네트워크를 통해 캡슐화된 OpenShift Container Platform SDN을 실행하는 경우와 같은 이중 캡슐화를 피할 수 있습니다.

공급자 네트워크 또는 테넌트 VLAN을 사용하는 경우에는 이중 캡슐화를 피하기 위해 Kuryr를 사용하지 않아도 됩니다. 성능상의 이점은 무시할만한 수준입니다. 그러나 구성에 따라 두 개의 오버레이가 없도록 Kuryr을 사용하는 방법이 계속 유용할 수 있습니다.

다음 기준이 모두 해당되는 배포에는 Kuryr가 권장되지 않습니다.

- RHOSP 버전이 16 미만입니다.
- 배포 시 UDP 서비스 또는 소수의 하이퍼바이저에서 많은 수의 TCP 서비스를 사용합니다.

또는

- ovn-octavia Octavia 드라이버가 비활성화되었습니다.
- 배포 시 소수의 하이퍼바이저에서 많은 수의 TCP 서비스를 사용합니다.

15.6.3. Kuryr를 사용하는 RHOSP에 OpenShift Container Platform을 설치하기 위한 리소스 지침

Kuryr SDN을 사용하는 경우 Pod, 서비스, 네임스페이스, 네트워크 정책은 RHOSP 할당량의 리소스를 사용하며 이로 인해 최소 요구사항이 증가합니다. Kuryr는 또한 기본 설치에 필요한 요구사항 이외에 몇 가지 추가 요구사항을 갖습니다.

기본 클러스터의 최소 요구사항을 충족할 수 있는 할당량은 다음과 같습니다.

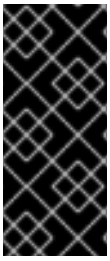
표 15.23. Kuryr를 사용하는 RHOSP의 기본 OpenShift Container Platform 클러스터에 권장되는 리소스

리소스 이름	값
부동 IP 주소	3 - LoadBalancer 유형의 예상 서비스 수 추가
포트	1500 - Pod당 1개 필요
라우터	1
서브넷	250 - 네임스페이스/프로젝트당 1개 필요
네트워크	250 - 네임스페이스/프로젝트당 1개 필요
RAM	112GB
vCPU	28
볼륨 스토리지	275GB
인스턴스	7
보안 그룹	250 - 서비스/NetworkPolicy당 1개 필요
보안 그룹 규칙	1000
로드 밸런서	100 - 서비스당 1개 필요
로드 밸런서 리스너	500 - 서비스 노출 포트당 1개 필요
로드 밸런서 풀	500 - 서비스 노출 포트당 1개 필요

권장 리소스보다 적은 리소스로도 클러스터가 작동할 수 있지만 성능은 보장되지 않습니다.

중요

swiftoperator 역할을 가진 사용자 계정으로 RHOSP 개체 스토리지(Swift)를 사용하고 운영하는 경우 OpenShift Container Platform 이미지 레지스트리의 기본 백엔드로 사용됩니다. 이 경우 볼륨 스토리지 요구사항은 175GB입니다. Swift 공간 요구사항은 이미지 레지스트리의 크기에 따라 다릅니다.



중요

OVN Octavia 드라이버가 아닌 **Amphora** 드라이버와 함께 **RHOSP(Red Hat OpenStack Platform)** 버전 **16**을 사용하는 경우 보안 그룹은 사용자 프로젝트 대신 서비스 계정과 연결됩니다.

리소스를 설정할 때 다음 사항을 고려하십시오.

- 필요한 포트 수가 **Pod** 수보다 많습니다. **Kuryr**는 포트 풀을 사용하여 **Pod**가 미리 생성된 포트를 사용할 수 있도록 준비하여 **Pod**의 부팅 시간을 단축시킵니다.

- 각 **NetworkPolicy**는 **RHOSP** 보안 그룹에 매핑되며 **NetworkPolicy** 사양에 따라 하나 이상의 규칙이 보안 그룹에 추가됩니다.

- 각 서비스는 **RHOSP** 로드 밸런서에 매핑됩니다. 할당량에 필요한 보안 그룹 수를 추정할 때 이 요구사항을 고려하십시오.

  RHOSP 버전 **15** 이하 또는 **ovn-octavia driver**를 사용하는 경우 각 로드 밸런서에 사용자 프로젝트와 보안 그룹이 있습니다.

- 할당량은 로드 밸런서 리소스(예: **VM** 리소스)를 고려하지 않지만 **RHOSP** 배포 크기를 결정할 때 이러한 리소스를 고려해야 합니다. 기본 설치에는 **50**개 이상의 로드 밸런서가 있으며 클러스터가 이 로드 밸런서를 수용할 수 있어야 합니다.

  OVN Octavia 드라이버가 활성화된 상태에서 **RHOSP** 버전 **16**을 사용하는 경우에는 로드 밸런서 **VM**이 하나만 생성됩니다. 서비스는 **OVN** 흐름을 통해 부하를 분산시킵니다.

OpenShift Container Platform 배포는 컨트롤 플레인 시스템, 컴퓨팅 시스템, 부트스트랩 시스템으로 구성됩니다.

Kuryr SDN을 활성화하려면 환경이 다음 요구사항을 충족해야 합니다.

- **RHOSP 13+**를 실행합니다.

- Octavia와 함께 Overcloud가 설치되어 있습니다.
- Neutron 트렁크 포트 확장을 사용합니다.
- ovs-hybrid 대신 ML2/OVS Neutron 드라이버를 사용하는 경우 openvswitch를 사용합니다.

15.6.3.1. 할당량 늘리기

Kuryr SDN을 사용하는 경우 Pod, 서비스, 네임스페이스 및 네트워크 정책에서 사용하는 RHOSP(Red Hat OpenStack Platform) 리소스를 충족하기 위해 할당량을 늘려야 합니다.

프로세스

- 다음 명령을 실행하여 프로젝트 할당량을 늘립니다.

```
$ sudo openstack quota set --secgroups 250 --secgroup-rules 1000 --ports 1500 --
subnets 250 --networks 250 <project>
```

15.6.3.2. Neutron 구성

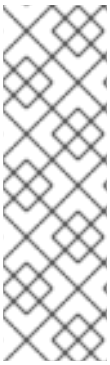
Kuryr CNI는 Neutron 트렁크 확장을 활용하여 컨테이너를 RHOSP(Red Hat OpenStack Platform) SDN에 연결하므로 Kuryr이 제대로 작동하려면 trunks 확장을 사용해야 합니다.

또한 기본 ML2/OVS Neutron 드라이버를 활용하는 경우 보안 그룹이 트렁크 서브포트에서 적용되고 Kuryr가 네트워크 정책을 올바르게 처리할 수 있도록 방화벽이 ovs_hybrid 대신 openvswitch로 설정되어야 합니다.

15.6.3.3. Octavia 구성

Kuryr SDN은 RHOSP(Red Hat OpenStack Platform)의 Octavia LBaaS를 사용하여 OpenShift Container Platform 서비스를 구현합니다. 따라서 Kuryr SDN을 사용하려면 RHOSP에서 Octavia 구성 요소를 설치하고 구성해야 합니다.

Octavia를 활성화하려면 RHOSP Overcloud 설치 중에 Octavia 서비스를 포함하거나 Overcloud가 이미 존재하는 경우 Octavia 서비스를 업그레이드해야 합니다. Octavia를 활성화하는 다음 단계는 Overcloud 새로 설치 또는 Overcloud 업데이트에 모두 적용됩니다.



참고

다음 단계는 Octavia를 처리할 때 RHOSP 배포 과정에서 필요한 주요 부분만을 다룹니다. 레지스트리 방법도 다를 수 있습니다.

이 예에서는 로컬 레지스트리 방법을 사용합니다.

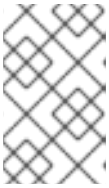
프로세스

1. 로컬 레지스트리를 사용하는 경우 이미지를 레지스트리에 업로드하는 템플릿을 만듭니다. 예를 들면 다음과 같습니다.

```
(undercloud) $ openstack overcloud container image prepare \
-e /usr/share/openstack-tripleo-heat-templates/environments/services-
docker/octavia.yaml \
--namespace=registry.access.redhat.com/rhosp13 \
--push-destination=<local-ip-from-undercloud.conf>:8787 \
--prefix=openstack- \
--tag-from-label {version}-{product-version} \
--output-env-file=/home/stack/templates/overcloud_images.yaml \
--output-images-file /home/stack/local_registry_images.yaml
```

2. local_registry_images.yaml 파일에 Octavia 이미지가 포함되어 있는지 확인합니다. 예를 들면 다음과 같습니다.

```
...
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-api:13.0-43
  push_destination: <local-ip-from-undercloud.conf>:8787
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-health-
manager:13.0-45
  push_destination: <local-ip-from-undercloud.conf>:8787
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-
housekeeping:13.0-45
  push_destination: <local-ip-from-undercloud.conf>:8787
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-worker:13.0-44
  push_destination: <local-ip-from-undercloud.conf>:8787
```



참고

Octavia 컨테이너 버전은 설치된 특정 RHOSP 릴리스에 따라 다릅니다.

3. registry.redhat.io에서 Undercloud 노드로 컨테이너 이미지를 가져옵니다.

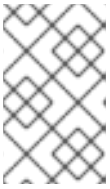
```
(undercloud) $ sudo openstack overcloud container image upload \
  --config-file  /home/stack/local_registry_images.yaml \
  --verbose
```

네트워크 및 Undercloud 디스크의 속도에 따라 다소 시간이 걸릴 수 있습니다.

4.

Octavia 로드 밸런서는 OpenShift Container Platform API에 액세스하는 데 사용되므로 연결에 대한 리스너의 기본 제한 시간을 늘려야 합니다. 기본 제한 시간은 50초입니다. 다음 파일을 Overcloud 배포 명령에 전달하여 제한 시간을 20분으로 늘립니다.

```
(undercloud) $ cat octavia_timeouts.yaml
parameter_defaults:
  OctaviaTimeoutClientData: 1200000
  OctaviaTimeoutMemberData: 1200000
```



참고

RHOSP 13.0.13+에는 필요하지 않습니다.

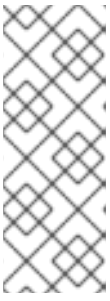
5.

Octavia로 Overcloud 환경을 설치 또는 업데이트합니다.

```
$ openstack overcloud deploy --templates \
  -e /usr/share/openstack-tripleo-heat-templates/environments/services-
docker/octavia.yaml \
  -e octavia_timeouts.yaml
```

참고

이 명령에는 Octavia와 관련된 파일만 포함됩니다. 해당 파일은 RHOSP의 특정 설치에 따라 다릅니다. 자세한 내용은 RHOSP 문서를 참조하십시오. Octavia 설치 사용자 지정에 대한 자세한 내용은 Director를 사용하여 Octavia 설치를 참조하십시오.



참고

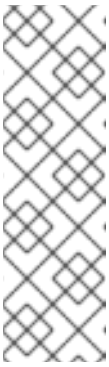
Kuryr SDN을 활용하는 경우 Overcloud 설치에는 Neutron trunk 확장이 필요합니다. 이 확장은 기본적으로 디렉터 배포에서 사용할 수 있습니다. Neutron 백엔드가 ML2/OVS인 경우 기본 ovs-hybrid 대신 openvswitch 방화벽을 사용합니다. 백엔드가 ML2/OVN인 경우에는 수정하지 않아도 됩니다.

6.

RHOSP 13.0.13 이전 버전에서 프로젝트를 생성한 후 프로젝트 ID를 octavia.conf 구성 파일에 추가합니다.

- 트래픽이 Octavia 로드 밸런서를 통과할 때와 같이 서비스 전체에 네트워크 정책을 적용하려면 Octavia가 사용자 프로젝트에서 Amphora VM 보안 그룹을 생성해야 합니다.

  이 변경으로 인해 필요한 로드 밸런서 보안 그룹이 해당 프로젝트에 속하게 되며 서비스 격리를 적용하도록 업데이트될 수 있습니다.



참고

RHOSP 13.0.13 버전 이상에서는 이 작업이 필요하지 않습니다.

Octavia는 로드 밸런서 VIP에 대한 액세스를 제한하는 새로운 ACL API를 구현합니다.

a. 프로젝트 ID 가져오기

```
$ openstack project show <project>
```

출력 예

```
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description |                                  |
| domain_id   | default                          |
| enabled     | True                             |
| id          | PROJECT_ID                       |
| is_domain   | False                            |
| name        | *<project>*                      |
| parent_id   | default                          |
| tags        | []                               |
+-------------+----------------------------------+
```

b. 컨트롤러의 octavia.conf에 프로젝트 ID를 추가합니다.

i. **stackrc** 파일을 소싱합니다.

```
$ source stackrc  # Undercloud credentials
```

ii. **Overcloud** 컨트롤러를 나열합니다.

```
$ openstack server list
```

출력 예

```
+--------------------------------------+-------------+--------+----------------------+-------
---------+------------+
|
| ID                                   | Name        | Status | Networks
| Image         | Flavor     |
|
+--------------------------------------+-------------+--------+----------------------+-------
---------+------------+
|
| 6bef8e73-2ba5-4860-a0b1-3937f8ca7e01 | controller-0 | ACTIVE |
ctlplane=192.168.24.8 | overcloud-full | controller |
|
| dda3173a-ab26-47f8-a2dc-8473b4a67ab9 | compute-0    | ACTIVE |
ctlplane=192.168.24.6 | overcloud-full | compute    |
|
+--------------------------------------+-------------+--------+----------------------+-------
---------+------------+
```

iii. 컨트롤러에 **SSH**를 적용합니다.

```
$ ssh heat-admin@192.168.24.8
```

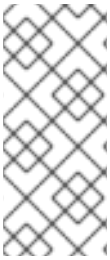
iv. **octavia.conf** 파일을 편집하여 **Amphora** 보안 그룹이 사용자 계정에 있는 프로젝트 목록에 프로젝트를 추가합니다.

```
# List of project IDs that are allowed to have Load balancer security groups
# belonging to them.
amp_secgroup_allowed_projects = PROJECT_ID
```

c.

Octavia 작업자를 다시 시작하여 새 구성을 로드합니다.

```
controller-0$ sudo docker restart octavia_worker
```



참고

RHOSP 환경에 따라 Octavia가 UDP 리스너를 지원하지 않을 수 있습니다. RHOSP 버전 13.0.13 이하에서 Kuryr SDN을 사용하는 경우에는 UDP 서비스가 지원되지 않습니다. RHOSP 버전 16 이상은 UDP를 지원합니다.

15.6.3.3.1. Octavia OVN 드라이버

Octavia는 Octavia API를 통해 여러 공급자 드라이버를 지원합니다.

사용 가능한 모든 Octavia 공급자 드라이버를 보려면 명령줄에서 다음을 입력합니다.

```
$ openstack loadbalancer provider list
```

출력 예

```
+---------+-------------------------------------------------+
| name    | description                                     |
+---------+-------------------------------------------------+
| amphora | The Octavia Amphora driver.                     |
| octavia | Deprecated alias of the Octavia Amphora driver. |
| ovn     | Octavia OVN driver.                             |
+---------+-------------------------------------------------+
```

RHOSP 버전 16부터 Octavia OVN 공급자 드라이버(ovn)는 RHOSP 배포의 OpenShift Container Platform에서 지원됩니다.

ovn은 Octavia와 OVN이 제공하는 로드 밸런싱을 위한 통합 드라이버입니다. 기본 로드 밸런싱 기능을 지원하며 OpenFlow 규칙을 기반으로 합니다. 이 드라이버는 OVN Neutron ML2를 사용하는 배포에서 Director에 의해 Octavia에서 자동으로 활성화됩니다.

기본 드라이버는 Amphora 공급자 드라이버입니다. 그러나 ovn이 활성화되면 Kuryr는 ovn을 사용합니다.

Kuryr가 Amphora 대신 ovn을 사용하는 경우 다음과 같은 이점을 제공합니다.

- 리소스 요구사항 감소. Kuryr는 각 서비스마다 로드 밸런서 VM이 필요하지 않습니다.
- 네트워크 대기 시간 감소.
- 각 서비스마다 VM 대신 OpenFlow 규칙을 사용하므로 서비스 생성 속도가 향상됩니다.
- 로드 밸런싱 작업이 Amphora VM에 집중되지 않고 모든 노드에 분산됩니다.

15.6.3.4. Kuryr를 사용한 설치의 알려진 제한 사항

Kuryr SDN과 함께 OpenShift Container Platform을 사용하는 경우 몇 가지 알려진 제한 사항이 있습니다.

RHOSP 일반 제한 사항

Kuryr SDN을 사용하는 OpenShift Container Platform 사용에는 모든 버전 및 환경에 적용되는 몇 가지 제한 사항이 있습니다.

- NodePort 유형의 Service 개체는 지원되지 않습니다.
- OVN Octavia 공급자 드라이버를 사용하는 클러스터는 끝점 개체의 .spec.selector 속성에 노드 또는 포드의 서브넷이 포함된 경우에만 .subsets.addresses 속성이 지정되지 않은 Service 개체를 지원합니다.
- 시스템이 생성된 서브넷이 라우터에 연결되어 있지 않거나 서브넷이 연결되어 있지만 라우터에 외부 게이트웨이가 설정되지 않은 경우 Kuryr는 LoadBalancer인 Service 개체에 대한 유동 IP를 생성할 수 없습니다.
- Service 오브젝트에 sessionAffinity=ClientIP 속성을 구성해도 효과가 없습니다. Kuryr는 이 설정을 지원하지 않습니다.

RHOSP 버전 제한 사항

Kuryr SDN을 사용하는 OpenShift Container Platform은 RHOSP 버전에 따라 몇 가지 제한 사항이 있습니다.

- 버전 16 미만의 RHOSP는 기본 Octavia로드 밸런서 드라이버(Amphora)를 사용합니다. 이 드라이버를 사용하려면 OpenShift Container Platform 서비스당 하나의 Amphora 로드 밸런서 VM이 배포되어야 합니다. 너무 많은 서비스를 생성하면 리소스가 부족해질 수 있습니다.

  OVN Octavia 드라이버가 비활성화된 상위 버전의 RHOSP 배포에도 Amphora 드라이버를 사용합니다. 하위 버전의 RHOSP와 동일한 리소스 문제가 있습니다.

- 버전 13.0.13 미만의 Octavia RHOSP는 UDP 리스너를 지원하지 않습니다. 따라서 OpenShift Container Platform UDP 서비스는 지원되지 않습니다.

- 버전 13.0.13 미만의 Octavia RHOSP는 동일한 포트에서 여러 프로토콜을 수신할 수 없습니다. TCP, UDP 등 다른 프로토콜에 동일한 포트를 노출하는 서비스는 지원되지 않습니다.

- Kuryr SDN은 서비스에서 자동 유휴 상태 해제를 지원하지 않습니다.

RHOSP 환경 제한 사항

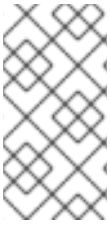
배포 환경에 따라 Kuryr SDN 사용에 제한이 있습니다.

Octavia는 UDP 프로토콜과 다중 리스너를 지원하지 않으므로 RHOSP 버전이 13.0.13 미만이면 Kuryr에 따라 Pod가 DNS 확인을 위해 TCP를 사용해야 합니다.

Go 버전 1.12 이하에서는 CGO 지원이 비활성화된 상태로 컴파일된 애플리케이션이 UDP만 사용합니다. 이 경우 기본 Go 해결 프로그램이 resolv.conf의 use-vc 옵션을 인식하지 못합니다. 이 옵션은 DNS 확인을 위해 TCP가 적용되는지 여부를 제어합니다. 결과적으로 DNS 확인에 계속 UDP를 사용하여 실패하게 됩니다.

TCP 적용을 허용하려면 환경 변수 CGO_ENABLED를 1로 설정하거나(즉 CGO_ENABLED=1) 변수가 없는지 확인하여 애플리케이션을 컴파일합니다.

Go 버전 1.13 이상에서는 UDP를 사용한 DNS 확인이 실패하면 자동으로 TCP를 사용합니다.



참고

Alpine 기반 컨테이너를 포함한 musl 기반 컨테이너는 use-vc 옵션을 지원하지 않습니다.

RHOSP 업그레이드 제한 사항

RHOSP 업그레이드 프로세스에 따라 Octavia API가 변경될 수 있으며 로드 밸런서에 사용되는 Amphora 이미지로 업그레이드해야 할 수 있습니다.

API 변경은 개별적으로 처리할 수 있습니다.

Amphora 이미지가 업그레이드되면 RHOSP Operator가 다음 두 가지 방법으로 기존 로드 밸런서 VM을 처리할 수 있습니다.

- 로드 밸런서 장애 조치를 트리거하여 각 VM을 업그레이드합니다.
- VM 업그레이드를 사용자가 처리하게 합니다.

Operator가 첫 번째 옵션을 선택하는 경우 장애 조치 중에 가동 중지 시간이 짧아질 수 있습니다.

Operator가 두 번째 옵션을 선택하는 경우 기존 로드 밸런서는 UDP 리스너와 같은 업그레이드된 Octavia API 기능을 지원하지 않습니다. 이 경우 이러한 기능을 사용하려면 사용자가 서비스를 다시 만들어야 합니다.

중요

OpenShift Container Platform이 UDP 로드 밸런싱을 지원하는 새로운 Octavia 버전을 감지하면 DNS 서비스를 자동으로 다시 생성합니다. 서비스를 다시 생성함으로써 서비스는 기본적으로 UDP 로드 밸런싱을 지원하게 됩니다.

서비스를 다시 생성하는 경우 DNS 서비스 가동이 약 1분 동안 중지됩니다.

15.6.3.5. 컨트롤 플레인 머신

기본적으로 **OpenShift Container Platform** 설치 프로세스는 세 개의 컨트롤 플레인 시스템을 생성합니다.

각 시스템의 요구사항은 다음과 같습니다.

- **RHOSP** 할당량의 인스턴스
- **RHOSP** 할당량의 포트
- 최소 **16GB** 메모리 및 **vCPU 4**개가 있는 플레이버
- **RHOSP** 할당량에서 최소 **100GB** 스토리지 공간

15.6.3.6. 컴퓨팅 머신

기본적으로 **OpenShift Container Platform** 설치 프로세스는 세 개의 컴퓨팅 시스템을 생성합니다.

각 시스템의 요구사항은 다음과 같습니다.

- **RHOSP** 할당량의 인스턴스
- **RHOSP** 할당량의 포트
- **8GB** 메모리 및 **vCPU 2**개 이상 있는 플레이버
- **RHOSP** 할당량에서 최소 **100GB** 스토리지 공간

작은 정보

컴퓨팅 시스템은 **OpenShift Container Platform**에서 실행하는 애플리케이션을 호스팅합니다. 최대한 많이 실행하는 것이 좋습니다.

15.6.3.7. 부트스트랩 시스템

설치하는 동안 컨트롤 플레인을 유지하기 위해 부트스트랩 시스템이 임시로 프로비저닝됩니다. 프로덕션 컨트롤 플레인이 준비되면 부트스트랩 시스템이 프로비저닝 해제됩니다.

부트스트랩 시스템의 요구사항은 다음과 같습니다.

- RHOSP 할당량의 인스턴스
- RHOSP 할당량의 포트
- 최소 16GB 메모리 및 vCPU 4개가 있는 플레이버
- RHOSP 할당량에서 최소 100GB 스토리지 공간

15.6.4. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.

중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

15.6.5. 플레이북 종속 항목 다운로드

사용자 프로비저닝 인프라에서의 설치 프로세스를 간소화하는 **Ansible** 플레이북에는 몇 가지 **Python** 모듈이 필요합니다. 설치 관리자를 실행할 시스템에서 모듈 리포지토리를 추가하고 다운로드합니다.

참고

이 방법은 현재 **Red Hat Enterprise Linux (RHEL) 8**을 사용하는 것으로 가정합니다.

사전 요구 사항

- **Python 3**가 시스템에 설치되어 있습니다.

프로세스

1. 명령줄에서 리포지토리를 추가합니다.
   a. **Red Hat Subscription Manager**에 등록합니다.

```
$ sudo subscription-manager register # If not done already
```

   b. 최신 서브스크립션 데이터를 가져옵니다.

```
$ sudo subscription-manager attach --pool=$YOUR_POOLID # If not done already
```

   c. 현재 리포지토리를 비활성화합니다.

```
$ sudo subscription-manager repos --disable=* # If not done already
```

.

d.
필요한 리포지토리를 추가합니다.

```
$ sudo subscription-manager repos \
  --enable=rhel-8-for-x86_64-baseos-rpms \
  --enable=openstack-16-tools-for-rhel-8-x86_64-rpms \
  --enable=ansible-2.9-for-rhel-8-x86_64-rpms \
  --enable=rhel-8-for-x86_64-appstream-rpms
```

2.
모듈을 설치합니다.

```
$ sudo yum install python3-openstackclient ansible python3-openstacksdk python3-
netaddr
```

3.
python 명령이 python3를 가리키는지 확인합니다.

```
$ sudo alternatives --set python /usr/bin/python3
```

15.6.6. 설치 플레이북 다운로드

자체 RHOSP(Red Hat OpenStack Platform) 인프라에 OpenShift Container Platform을 설치하는 데 사용할 수 있는 Ansible 플레이북을 다운로드합니다.

사전 요구 사항

- curl 명령줄 툴은 사용자의 머신에서 사용할 수 있습니다.

프로세스

- 플레이북을 작업 디렉터리에 다운로드하려면 명령줄에서 다음 스크립트를 실행합니다.

```
$ xargs -n 1 curl -O <<< '
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/bootstrap.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/common.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/compute-nodes.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/control-plane.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/inventory.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
```

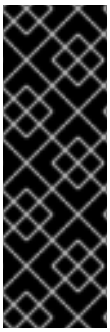
```
4.9/upi/openstack/network.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/security-groups.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/down-bootstrap.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/down-compute-nodes.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/down-control-plane.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/down-load-balancers.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/down-network.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/down-security-groups.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/down-containers.yaml'
```

플레이북이 사용자의 머신에 다운로드됩니다.

중요

설치 프로세스 중에 플레이북을 수정하여 배포를 구성할 수 있습니다.

클러스터 수명에 대해 모든 플레이북을 유지합니다. RHOSP에서 OpenShift Container Platform 클러스터를 제거하려면 플레이북이 있어야 합니다.



중요

bootstrap.yaml, compute-nodes.yaml, control-plane.yaml, network.yaml 및 security-groups.yaml 파일에서 만든 모든 편집 사항을 down- 접두어가 있는 해당 플레이북과 일치시켜야 합니다. 예를 들어 bootstrap.yaml 파일에 대한 편집도 down-bootstrap.yaml 파일에 반영해야 합니다. 두 파일을 모두 편집하지 않으면 지원되는 클러스터 제거 프로세스가 실패합니다.

15.6.7. 설치 프로그램 받기

OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

프로세스

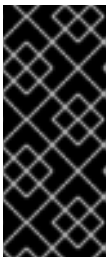
1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.

2. 인프라 공급자를 선택합니다.

3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.

중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.



중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 OpenShift Container Platform 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

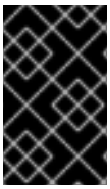
5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

15.6.8. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 core 사용자의 ~/.ssh/authorized_keys 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 core로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. ./openshift-install gather 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.



중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: ~/.ssh/id_ed25519)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 '~/.ssh 디렉터리에 있는지 확인하십시오.

참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 ed25519 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 rsa 또는 ecdsa 알고리즘을 사용하는 키를 생성합니다.

2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 ~/.ssh/id_ed25519.pub 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 ./openshift-install gather 명령을 사용하려는 경우 필요합니다.

참고

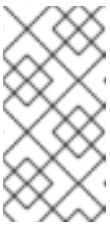
일부 배포에서는 ~/.ssh/id_rsa 및 ~/.ssh/id_dsa와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. ssh-agent 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```



참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. ssh-agent에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: ~/.ssh/id_ed25519).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

15.6.9. RHCOS(Red Hat Enterprise Linux CoreOS) 이미지 생성

OpenShift Container Platform 설치 프로그램을 사용하려면 RHOSP(Red Hat OpenStack Platform) 클러스터에 RHCOS(Red Hat Enterprise Linux CoreOS) 이미지가 있어야 합니다. 최신 RHCOS 이미지를 검색한 다음 RHOSP CLI를 사용하여 업로드하십시오.

사전 요구 사항

- RHOSP CLI가 설치되어 있습니다.

프로세스

1. Red Hat Customer Portal의 제품 다운로드 페이지에 로그인합니다.
2. Version 에서 Red Hat Enterprise Linux (RHEL) 8용 최신 OpenShift Container Platform 4.9 릴리스를 선택합니다.

중요

RHCOS 이미지는 OpenShift Container Platform 릴리스에 따라 변경되지 않을 수 있습니다. 설치하는 OpenShift Container Platform 버전과 같거나 그 이하의 버전 중 가장 최신 버전의 이미지를 다운로드해야 합니다. 사용 가능한 경우 OpenShift Container Platform 버전과 일치하는 이미지 버전을 사용합니다.

3. *RHCOS(Red Hat Enterprise Linux CoreOS) - OpenStack Image (QCOW)*를 다운로드합니다.
4. 이미지 압축을 풉니다.

참고

RHOSP 이미지 압축을 풀어야 클러스터가 사용할 수 있습니다. 다운로드한 파일의 이름에 .gz 또는 .tgz와 같은 압축 확장자가 포함되지 않을 수 있습니다. 파일 압축 여부를 확인하려면 명령줄에서 다음을 입력합니다.

```
$ file <name_of_downloaded_file>
```

5. 다운로드한 이미지에서 RHOSP CLI를 사용하여 이름이 rhcos인 이미지를 클러스터에 생성합니다.

```
$ openstack image create --container-format=bare --disk-format=qcow2 --file rhcos-${RHCOS_VERSION}-openstack.qcow2 rhcos
```

중요

RHOSP 환경에 따라 이미지를 .raw 또는 .qcow2 형식으로 업로드할 수 있습니다. Ceph를 사용하는 경우 .raw 형식을 사용해야 합니다.

주의

설치 프로그램이 이름이 같은 여러 이미지를 발견하면 그 중 하나를 임의로 선택합니다. 이 동작을 방지하려면 RHOSP에서 리소스의 고유한 이름을 만듭니다.

이미지를 RHOSP에 업로드한 후 설치 프로세스에서 사용할 수 있습니다.

15.6.10. 외부 네트워크 액세스 확인

OpenShift Container Platform 설치 프로세스에는 외부 네트워크에 액세스해야 합니다. 외부 네트워크값을 제공해야 하며 그렇지 않으면 배포가 실패합니다. 프로세스를 시작하기 전에 외부 라우터 유형의 네트워크가 RHOSP(Red Hat OpenStack Platform)에 있는지 확인합니다.

사전 요구 사항

- DHCP 에이전트가 인스턴스의 DNS 쿼리를 전달하도록 OpenStack의 네트워킹 서비스 구성

프로세스

1. RHOSP CLI를 사용하여 '외부' 네트워크의 이름과 ID를 확인합니다.

```
$ openstack network list --long -c ID -c Name -c "Router Type"
```

출력 예

```
+--------------------------------------+----------------+-------------+
| ID                                   | Name           | Router Type |
+--------------------------------------+----------------+-------------+
| 148a8023-62a7-4672-b018-003462f8d7dc | public_network | External    |
+--------------------------------------+----------------+-------------+
```

외부 라우터 유형의 네트워크가 네트워크 목록에 나타납니다. 네트워크가 하나 이상 나타나지 않으면 기본 부동 IP 네트워크 생성 및 기본 공급자 네트워크 생성을 참조하십시오.

참고

Neutron 트렁크 서비스 플러그인이 활성화된 경우 기본적으로 트렁크 포트가 생성됩니다. 자세한 내용은 Neutron 트렁크 포트를 참조하십시오.

15.6.11. 환경에 대한 액세스 활성화

배포 시 모든 OpenShift Container Platform 시스템은 RHOSP(Red Hat OpenStack Platform) 테넌트 네트워크에서 생성됩니다. 따라서 대부분의 RHOSP 배포에서 직접 액세스할 수 없습니다.

설치시 부동 IP 주소 (FIP)를 사용하여 OpenShift Container Platform API 및 애플리케이션의 액세스를 설정할 수 있습니다. FIP를 구성하지 않고 설치를 완료 할 수도 있지만 설치 프로그램은 외부에서 API 또는 애플리케이션에 액세스하는 방법을 설정하지 않습니다.

15.6.11.1. 부동 IP 주소로 액세스 활성화

OpenShift Container Platform API, 클러스터 애플리케이션 및 부트스트랩 프로세스에 대한 외부 액세스 용으로 유동 **IP (FIP)** 주소를 생성합니다.

절차

1. **RHOSP(Red Hat OpenStack Platform) CLI**를 사용하여 **API FIP**를 생성합니다.

```
$ openstack floating ip create --description "API <cluster_name>.<base_domain>"
<external_network>
```

2. **RHOSP(Red Hat OpenStack Platform) CLI**를 사용하여 앱 또는 **Ingress, FIP**를 생성합니다.

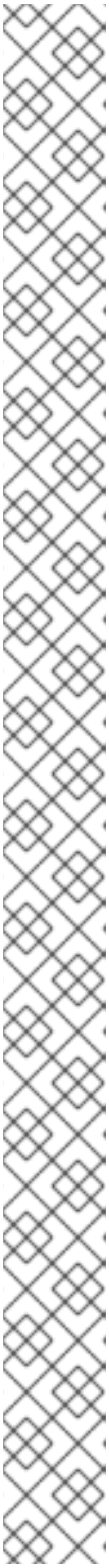
```
$ openstack floating ip create --description "Ingress <cluster_name>.<base_domain>"
<external_network>
```

3. **RHOSP(Red Hat OpenStack Platform) CLI**를 사용하여 부트스트랩 **FIP**를 생성합니다.

```
$ openstack floating ip create --description "bootstrap machine" <external_network>
```

4. **API** 및 **Ingress FIP**의 **DNS** 서버에 이러한 패턴에 따라 레코드를 추가합니다.

```
api.<cluster_name>.<base_domain>.  IN  A  <API_FIP>
*.apps.<cluster_name>.<base_domain>. IN  A <apps_FIP>
```



참고

DNS 서버를 제어하지 않으면 /etc/hosts 파일에 다음과 같은 클러스터 도메인 이름을 추가하여 클러스터에 액세스할 수 있습니다.

- <api_floating_ip> api.<cluster_name>.<base_domain>
- <application_floating_ip> grafana-openshift-monitoring.apps.<cluster_name>.<base_domain>
- <application_floating_ip> prometheus-k8s-openshift-monitoring.apps.<cluster_name>.<base_domain>
- <application_floating_ip> oauth-openshift.apps.<cluster_name>.<base_domain>
- <application_floating_ip> console-openshift-console.apps.<cluster_name>.<base_domain>
- application_floating_ip integrated-oauth-server-openshift-authentication.apps.<cluster_name>.<base_domain>

/etc/hosts 파일의 클러스터 도메인 이름은 웹 콘솔과 클러스터의 모니터링 인터페이스에 로컬로 액세스할 수 있는 권한을 부여합니다. kubectl 또는 oc 에서도 사용할 수 있습니다. <application_floating_ip>를 가리키는 추가 항목을 사용하여 사용자 애플리케이션에 액세스할 수 있습니다. 이 작업을 수행하면 사용자만 API 및 애플리케이션에 액세스할 수 있어 프로덕션 배포에는 적합하지 않지만 개발 및 테스트용 설치는 가능합니다.

5. FIP를 다음 변수의 값으로 inventory.yaml 파일에 추가합니다.

- os_api_fip
- os_bootstrap_fip

- os_ingress_fip

이러한 값을 사용하는 경우 **inventory.yaml** 파일의 **os_external_network** 변수 값으로 외부 네트워크를 입력해야 합니다.

작은 정보

부동 **IP** 주소를 할당하고 방화벽 구성을 업데이트하여 클러스터 외부에서 **OpenShift Container Platform** 리소스를 사용할 수 있습니다.

15.6.11.2. 유동 IP 주소없이 설치 완료

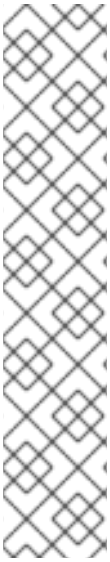
유동 **IP** 주소를 지정하지 않고도 **Red Hat OpenStack Platform (RHOSP)**에 **OpenShift Container Platform**을 설치할 수 있습니다.

inventory.yaml 파일에서 다음 변수를 정의하지 마십시오.

- os_api_fip
- os_bootstrap_fip
- os_ingress_fip

외부 네트워크를 제공할 수 없는 경우 **os_external_network**를 비워 둘 수도 있습니다. **os_external_network**의 값을 지정하지 않으면 라우터가 생성되지 않으며 추가 작업없이 설치 프로그램이 **Glance**에서 이미지를 검색하지 못합니다. 나중에 설치 프로세스에서 네트워크 리소스를 만들 때 외부 연결을 직접 구성해야 합니다.

유동 **IP** 주소 또는 이름 확인 부족으로 인해 클러스터 **API**에 연결할 수 없는 시스템에서 **wait-for** 명령으로 설치 프로그램을 실행하면 설치에 실패합니다. 이러한 경우 설치 실패를 방지하기 위해 프록시 네트워크를 사용하거나 시스템과 동일한 네트워크에있는 시스템에서 설치 프로그램을 실행할 수 있습니다.



참고

API 및 Ingress 포트의 DNS 레코드를 생성하여 이름 확인을 활성화할 수 있습니다. 예를 들면 다음과 같습니다.

```
api.<cluster_name>.<base_domain>.  IN  A  <api_port_IP>
*.apps.<cluster_name>.<base_domain>. IN  A <ingress_port_IP>
```

DNS 서버를 제어하지 않으면 /etc/hosts 파일에 레코드를 추가할 수 있습니다. 이 작업을 수행하면 사용자만 API에 액세스할 수 있어 프로덕션 배포에는 적합하지 않지만 개발 및 테스트용 설치는 가능합니다.

15.6.12. 설치 프로그램에 대한 매개변수 정의

OpenShift Container Platform 설치 프로그램은 clouds.yaml 파일을 사용합니다. 이 파일은 프로젝트 이름, 로그인 정보, 인증 서비스 URL 등 RHOSP(Red Hat OpenStack Platform) 구성 매개변수를 설명합니다.

프로세스

1. clouds.yaml 파일을 만듭니다.

- RHOSP 배포에 Horizon 웹 UI가 포함되어 있으면 그 안에 clouds.yaml 파일을 만듭니다.

중요

auth 필드에 암호를 추가해야 합니다. clouds.yaml과 별도의 파일에 비밀을 저장할 수도 있습니다.

- RHOSP 배포에 Horizon 웹 UI가 포함되어 있지 않거나 Horizon을 사용하지 않으려면 파일을 직접 만듭니다. clouds.yaml에 대한 자세한 내용은 RHOSP 문서의 구성 파일을 참조하십시오.

```
clouds:
  shiftstack:
    auth:
      auth_url: http://10.10.14.42:5000/v3
      project_name: shiftstack
      username: shiftstack_user
```

```
      password: XXX
      user_domain_name: Default
      project_domain_name: Default
    dev-env:
      region_name: RegionOne
      auth:
        username: 'devuser'
        password: XXX
        project_name: 'devonly'
        auth_url: 'https://10.10.14.22:5001/v2.0'
```

2. RHOSP 설치에서 끝점 인증을 위해 자체 서명된 CA(인증 기관) 인증서를 사용하는 경우:

   a. 인증 기관 파일을 시스템에 복사합니다.

   b. cacerts 키를 clouds.yaml 파일에 추가합니다. 값은 CA 인증서에 대한 루트가 아닌 액세스 가능한 절대 경로여야 합니다.

```
clouds:
  shiftstack:
    ...
    cacert: "/etc/pki/ca-trust/source/anchors/ca.crt.pem"
```

작은 정보

사용자 지정 CA 인증서로 설치 관리자를 실행한 후 cloud-provider-config 키맵에서 ca-cert.pem 키의 값을 편집하여 인증서를 업데이트할 수 있습니다. 명령줄에서 다음을 실행합니다.

```
$ oc edit configmap -n openshift-config cloud-provider-config
```

3. clouds.yaml 파일을 다음 위치 중 하나에 배치합니다.

   a. OS_CLIENT_CONFIG_FILE 환경 변수의 값

   b. 현재 디렉터리

   c. Unix 전용 사용자 구성 디렉터리(예: ~/.config/openstack/clouds.yaml)

d.

Unix 전용 사이트 구성 디렉터리(예: /etc/openstack/clouds.yaml)

설치 프로그램은 clouds.yaml을 이 순서대로 검색합니다.

15.6.13. 설치 구성 파일 만들기

RHOSP(Red Hat OpenStack Platform)에 설치하는 OpenShift Container Platform 클러스터를 사용자 지정할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.
- 서브스크립션 수준에서 서비스 주체 권한을 획득합니다.

프로세스

1. install-config.yaml 파일을 생성합니다.

   a. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 <installation_directory>는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.

중요

비어 있는 디렉터리를 지정합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

b. 화면에 나타나는 지시에 따라 클라우드에 대한 구성 세부 사항을 입력합니다.

i. 선택사항: 클러스터 시스템에 액세스하는 데 사용할 SSH 키를 선택합니다.

참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 ssh-agent 프로세스가 사용하는 SSH 키를 지정합니다.

ii. 대상 플랫폼으로 openstack을 선택합니다.

iii. 클러스터 설치에 사용할 RHOSP(Red Hat OpenStack Platform) 외부 네트워크 이름을 지정합니다.

iv. OpenShift API에 대한 외부 액세스에 사용할 부동 IP 주소를 지정합니다.

v. 컨트롤 플레인 노드에 사용할 최소 16GB의 RAM과 컴퓨팅 노드에 8GB RAM이 있는 RHOSP 플레이버를 지정합니다.

vi. 클러스터를 배포할 기본 도메인을 선택합니다. 모든 DNS 레코드는 이 기본 도메인의 하위 도메인이 되므로 클러스터 이름을 포함합니다.

vii. 클러스터 이름을 입력합니다. 이름은 14자 이하여야 합니다.

viii. Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 붙여넣습니다.

2. install-config.yaml 파일을 수정합니다. 사용 가능한 매개변수에 대한 자세한 정보는 “설치 구성 매개변수” 섹션에서 확인할 수 있습니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 install-config.yaml 파일을 백업합니다.

중요

install-config.yaml 파일은 설치 과정에서 사용됩니다. 이 파일을 재사용하려면 지금 백업해야 합니다.

이제 사용자가 지정한 디렉터리에 **install-config.yaml** 파일이 있습니다.

15.6.14. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 매개변수 값을 제공하여 클러스터를 호스팅할 클라우드 플랫폼에서 사용자 계정을 설명하고 선택사항으로 클러스터의 플랫폼을 사용자 지정합니다. **install-config.yaml** 설치 구성 파일을 생성할 때 명령줄을 통해 필요한 매개변수 값을 제공합니다. 클러스터를 사용자 지정하면 **install-config.yaml** 파일을 수정하여 플랫폼에 대한 세부 정보를 제공할 수 있습니다.

참고

설치한 후에는 **install-config.yaml** 파일에서 이러한 매개변수를 수정할 수 없습니다.

중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

15.6.14.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 **15.24.** 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열

매개변수	설명	값
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	소문자, 하이픈(**-**), 마침표(**.**)로 구성되는 문자열(예: **dev**)입니다. 문자열은 14자 이하여야 합니다.
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	{
		"auths":{
		"cloud.openshift.com":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		},
		"quay.io":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		}
		}
		}

15.6.14.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스

터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 15.25. 네트워크 매개변수

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체  **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.

networking.clusterNetwork	Pod의 IP 주소 블록입니다.
	기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다.
	여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.
		개체의 배열입니다. 예를 들면 다음과 같습니다.
		`networking:`
		`  clusterNetwork:`
		`  - cidr: 10.128.0.0/14`
		`  hostPrefix: 23`

networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된 **cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.

매개변수	설명	값
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16** **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

15.6.14.3. 선택적 구성 매개변수

선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

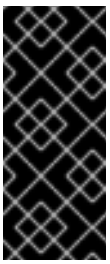
표 15.26. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.

매개변수	설명	값
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은**3**입니다.
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열

매개변수	설명	값
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 기본값은 **External**입니다. 이 필드를 **Internal** 로 설정하는 것은 클라우드 이외의 플랫폼에서는 지원되지 않습니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

15.6.14.4. 추가 RHOSP(Red Hat OpenStack Platform) 구성 매개변수

추가 RHOSP 구성 매개변수는 다음 표에 설명되어 있습니다.

표 15.27. 추가 RHOSP 매개변수

매개변수	설명	값
compute.platform.openstack.rootVolume.size	컴퓨팅 시스템의 경우 루트 볼륨의 크기(GB)입니다. 이 값을 설정하지 않으면 시스템은 임시 스토리지를 사용합니다.	정수 (예: **30**)
compute.platform.openstack.rootVolume.type	컴퓨팅 시스템의 경우 루트 볼륨의 유형입니다.	문자열 (예: **performance**)
controlPlane.platform.openstack.rootVolume.size	컨트롤 플레인 시스템의 경우 루트 볼륨의 크기(GB)입니다. 이 값을 설정하지 않으면 시스템은 임시 스토리지를 사용합니다.	정수 (예: **30**)
controlPlane.platform.openstack.rootVolume.type	컨트롤 플레인 시스템의 경우 루트 볼륨의 유형입니다.	문자열 (예: **performance**)
platform.openstack.cloud	**clouds.yaml** 파일의 클라우드 목록에서 사용할 RHOSP 클라우드의 이름입니다.	문자열 (예: **MyCloud**)

매개변수	설명	값
platform.openstack.externalNetwork	설치에 사용할 RHOSP 외부 네트워크 이름입니다.	문자열 (예: **external)**
platform.openstack.computeFlavor	컨트롤 플레인 및 컴퓨팅 시스템에 사용할 RHOSP 버전입니다. 이 속성은 더 이상 사용되지 않습니다. 모든 머신 풀에 플레이버를 기본값으로 사용하려면 **platform.openstack.defaultMachinePlatform** 속성의 **type** 키 값으로 추가합니다. 각 머신 풀의 플레이버 값을 개별적으로 설정할 수도 있습니다.	문자열 (예: **m1.xlarge)**

15.6.14.5. 선택적 RHOSP 구성 매개변수

선택적 RHOSP 구성 매개변수는 다음 표에 설명되어 있습니다.

표 15.28. 선택적 RHOSP 매개변수

매개변수	설명	값
compute.platform.openstack.additionalNetworkIDs	컴퓨팅 시스템과 관련된 추가 네트워크입니다. 추가 네트워크에는 허용되는 주소 쌍이 생성되지 않습니다.	하나 이상의 UUID 목록을 문자열로 나타냅니다. 예: **fa806b2f-ac49-4bce-b9db-124bc64209bf**
compute.platform.openstack.additionalSecurityGroupIDs	컴퓨팅 시스템과 관련된 추가 보안 그룹입니다.	하나 이상의 UUID 목록을 문자열로 나타냅니다. 예: **7ee219f3-d2e9-48a1-96c2-e7429f1b0da7**

매개변수	설명	값
compute.platform.openstack.zones	머신을 설치할 RHOSP Compute (Nova) 가용성 영역 (AZ). 이 매개 변수가 설정되지 않은 경우 설치 프로그램은 RHOSP 관리자가 구성한 Nova의 기본 설정을 사용합니다. Kuryr를 사용하는 클러스터에서 RHOSP Octavia는 가용성 영역을 지원하지 않습니다. 로드 밸런서 및 Amphora 공급자 드라이버를 사용하는 경우 Amphora VM에 의존하는 OpenShift Container Platform 서비스는이 속성의 값에 따라 생성되지 않습니다.	문자열 목록입니다. 예: **["zone-1", "zone-2"]**.
compute.platform.openstack.rootVolume.zones	컴퓨팅 시스템의 경우 루트 볼륨을 설치할 가용성 영역입니다. 이 매개변수의 값을 설정하지 않으면 설치 프로그램에서 기본 가용성 영역을 선택합니다.	문자열 목록(예: **["zone-1", "zone-2"]**).
controlPlane.platform.openstack.additionalNetworkIDs	컨트롤 플레인 시스템과 관련된 추가 네트워크입니다. 추가 네트워크에는 허용되는 주소 쌍이 생성되지 않습니다.	하나 이상의 UUID 목록을 문자열로 나타냅니다. 예: **fa806b2f-ac49-4bce-b9db-124bc64209bf**
controlPlane.platform.openstack.additionalSecurityGroupIDs	컨트롤 플레인 시스템과 관련된 추가 보안 그룹입니다.	하나 이상의 UUID 목록을 문자열로 나타냅니다. 예: **7ee219f3-d2e9-48a1-96c2-e7429f1b0da7**
controlPlane.platform.openstack.zones	머신을 설치할 RHOSP Compute (Nova) 가용성 영역 (AZ). 이 매개 변수가 설정되지 않은 경우 설치 프로그램은 RHOSP 관리자가 구성한 Nova의 기본 설정을 사용합니다. Kuryr를 사용하는 클러스터에서 RHOSP Octavia는 가용성 영역을 지원하지 않습니다. 로드 밸런서 및 Amphora 공급자 드라이버를 사용하는 경우 Amphora VM에 의존하는 OpenShift Container Platform 서비스는이 속성의 값에 따라 생성되지 않습니다.	문자열 목록입니다. 예: **["zone-1", "zone-2"]**.

매개변수	설명	값
controlPlane.platform.openstack.rootVolume.zones	컨트롤 플레인 시스템의 경우 루트 볼륨을 설치할 가용성 영역입니다. 이 값을 설정하지 않으면 설치 프로그램에서 기본 가용성 영역을 선택합니다.	문자열 목록(예: **["zone-1", "zone-2"]**).
platform.openstack.clusterOSImage	설치 프로그램이 RHCOS 이미지를 다운로드하는 위치입니다. 네트워크가 제한된 환경에서 설치를 수행하려면이 매개 변수를 설정해야 합니다.	HTTP 또는 HTTPS URL (선택 옵션으로 SHA-256 형식의 체크섬을 사용) 예: **http://mirror.example.com/images/rhcos-43.81.201912131630.0-openstack.x86_64.qcow2.gz?sha256=ffebbd68e8a1f2a245ca19522c16c86f67f9ac8e4e0c1f0a812b068b16f7265d**. 이 값은 기존의 Glance 이미지의 이름이 될 수도 있습니다 (예: **my-rhcos**).
platform.openstack.clusterOSImageProperties	Glance의 설치 프로그램에서 업로드된 ClusterOSImage에 추가할 속성입니다. **platform.openstack.clusterOSImage**가 기존 Glance 이미지로 설정된 경우 이 속성은 무시됩니다. 이 속성을 사용하여 노드당 26 PV의 RHOSP의 기본 PV(영구 볼륨) 제한을 초과할 수 있습니다. 제한을 초과하려면 **hw_scsi_model** 속성값을 **virtio-scsi**로, **hw_disk_bus** 값을 **scsi**로 설정합니다. 이 속성을 사용하면 **yes** 값이 있는 **hw_qemu_guest_agent** 속성을 포함하여 QEMU 게스트 에이전트를 활성화할 수도 있습니다.	키-값 문자열 쌍 목록입니다. 예를 들면 **["hw_scsi_model": "virtio-scsi", "hw_disk_bus": "scsi"]**가 있습니다.

platform.openstack.defaultMachinePlatform	기본 시스템 풀 플랫폼 구성입니다.	<pre>{
		  "type": "ml.large",
		  "rootVolume": {
		  "size": 30,
		  "type": "performance"
		  }
		}</pre>

매개변수	설명	값
platform.openstack.ingressFloatingIP	Ingress 포트와 연결할 기존 부동 IP 주소입니다. 이 속성을 사용하려면 **platform.openstack.externalNetwork** 속성도 정의해야 합니다.	IP 주소 (예: **128.0.0.1**)
platform.openstack.apiFloatingIP	로드 밸런서 API와 연결할 기존 부동 IP 주소입니다. 이 속성을 사용하려면 **platform.openstack.externalNetwork** 속성도 정의해야 합니다.	IP 주소 (예: **128.0.0.1**)
platform.openstack.externalDNS	클러스터 인스턴스가 DNS 확인에 사용하는 외부 DNS 서버의 IP 주소입니다.	IP 주소 목록을 문자열로 나타냅니다. 예: **["8.8.8.8", "192.168.1.12"]**.
platform.openstack.machinesSubnet	클러스터 노드가 사용하는 RHOSP 서브넷의 UUID입니다. 이 서브넷에 노드와 가상 IP(VIP) 포트가 생성됩니다. **networking.machineNetwork**의 첫 번째 항목은 **machinesSubnet** 값과 일치해야 합니다. 사용자 지정 서브넷에 배포하는 경우 OpenShift Container Platform 설치 관리자에 외부 DNS 서버를 지정할 수 없습니다. 대신 RHOSP의 서브넷에 DNS를 추가합니다.	문자열의 UUID입니다. 예: **fa806b2f-ac49-4bce-b9db-124bc64209bf**

15.6.14.6. RHOSP 배포의 사용자 지정 서브넷

선택한 **RHOSP(Red Hat OpenStack Platform)** 서브넷에 클러스터를 배포할 수도 있습니다. 서브넷의 **GUID**는 **install-config.yaml** 파일에서 **platform.openstack.machinesSubnet** 값으로 전달됩니다.

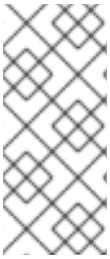
이 서브넷은 클러스터의 기본 서브넷으로 사용됩니다. 기본적으로 노드와 포트가 이 서브넷에서 생성됩니다. **platform.openstack.machinesSubnet** 속성의 값을 서브넷의 **UUID**로 설정하여 다른 **RHOSP** 서브넷에서 노드 및 포트를 생성할 수 있습니다.

사용자 지정 서브넷으로 **OpenShift Container Platform** 설치 관리자를 실행하기 전에 구성이 다음 요구 사항을 충족하는지 확인하십시오.

- platform.openstack.machinesSubnet에서 사용하는 서브넷에 DHCP가 활성화되어 있습니다.
- platform.openstack.machinesSubnet의 CIDR은 networking.machineNetwork의 CIDR과 일치합니다.
- 설치 프로그램 사용자에게 고정 IP 주소가 있는 포트를 포함하여 이 네트워크에 포트를 생성할 수 있는 권한이 있습니다.

사용자 지정 서브넷을 사용하는 클러스터에는 다음과 같은 제한 사항이 있습니다.

- 유동 IP 주소를 사용하는 클러스터를 설치하려는 경우 platform.openstack.machinesSubnet 서브넷이 externalNetwork 네트워크에 연결된 라우터에 연결되어 있어야 합니다.
- platform.openstack.machinesSubnet 값이 install-config.yaml 파일에 설정된 경우 설치 프로그램에서 RHOSP 머신의 프라이빗 네트워크 또는 서브넷을 생성하지 않습니다.
- 사용자 지정 서브넷과 동시에 platform.openstack.externalDNS 속성을 사용할 수 없습니다. 사용자 지정 서브넷을 사용하는 클러스터에 DNS를 추가하려면 RHOSP 네트워크에서 DNS를 구성합니다.



참고

기본적으로 API VIP는 x.x.x.5를 가져오고 Ingress VIP는 네트워크의 CIDR 블록에서 x.x.x.7을 가져옵니다. 이러한 기본값을 재정의하려면 DHCP 할당 풀 외부에 있는platform.openstack.apiVIP 및 platform.openstack.ingressVIP의 값을 설정합니다.

15.6.14.7. Kuryr를 사용한 RHOSP용 샘플 사용자 지정 install-config.yaml 파일

기본 OpenShift SDN 대신 Kuryr SDN으로 배포하려면 Kuryr를 원하는 networking.networkType으로 포함하도록 install-config.yaml 파일을 수정하고 기본 OpenShift Container Platform SDN 설치 단계를 진행해야 합니다. 이 샘플 install-config.yaml은 가능한 모든 RHOSP(Red Hat OpenStack Platform) 사용자 지정 옵션을 보여줍니다.

중요

이 샘플 파일은 참조용으로만 제공됩니다. **install-config.yaml** 파일은 설치 프로그램을 사용하여 받아야 합니다.

```
apiVersion: v1
baseDomain: example.com
controlPlane:
  name: master
  platform: {}
  replicas: 3
compute:
- name: worker
  platform:
    openstack:
      type: ml.large
  replicas: 3
metadata:
  name: example
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  serviceNetwork:
  - 172.30.0.0/16 (1)
  networkType: Kuryr
platform:
  openstack:
    cloud: mycloud
    externalNetwork: external
    computeFlavor: m1.xlarge
    apiFloatingIP: 128.0.0.1
    trunkSupport: true (2)
    octaviaSupport: true (3)
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```

(1)

Amphora Octavia 드라이버는 로드 밸런서당 두 개의 포트를 생성합니다. 결과적으로 설치 프로그램이 생성하는 서비스 서브넷은 **serviceNetwork** 속성 값으로 지정된 CIDR의 두 배 크기입니다. IP 주소 충돌을 방지하려면 더 큰 범위가 필요합니다.

(2) (3)

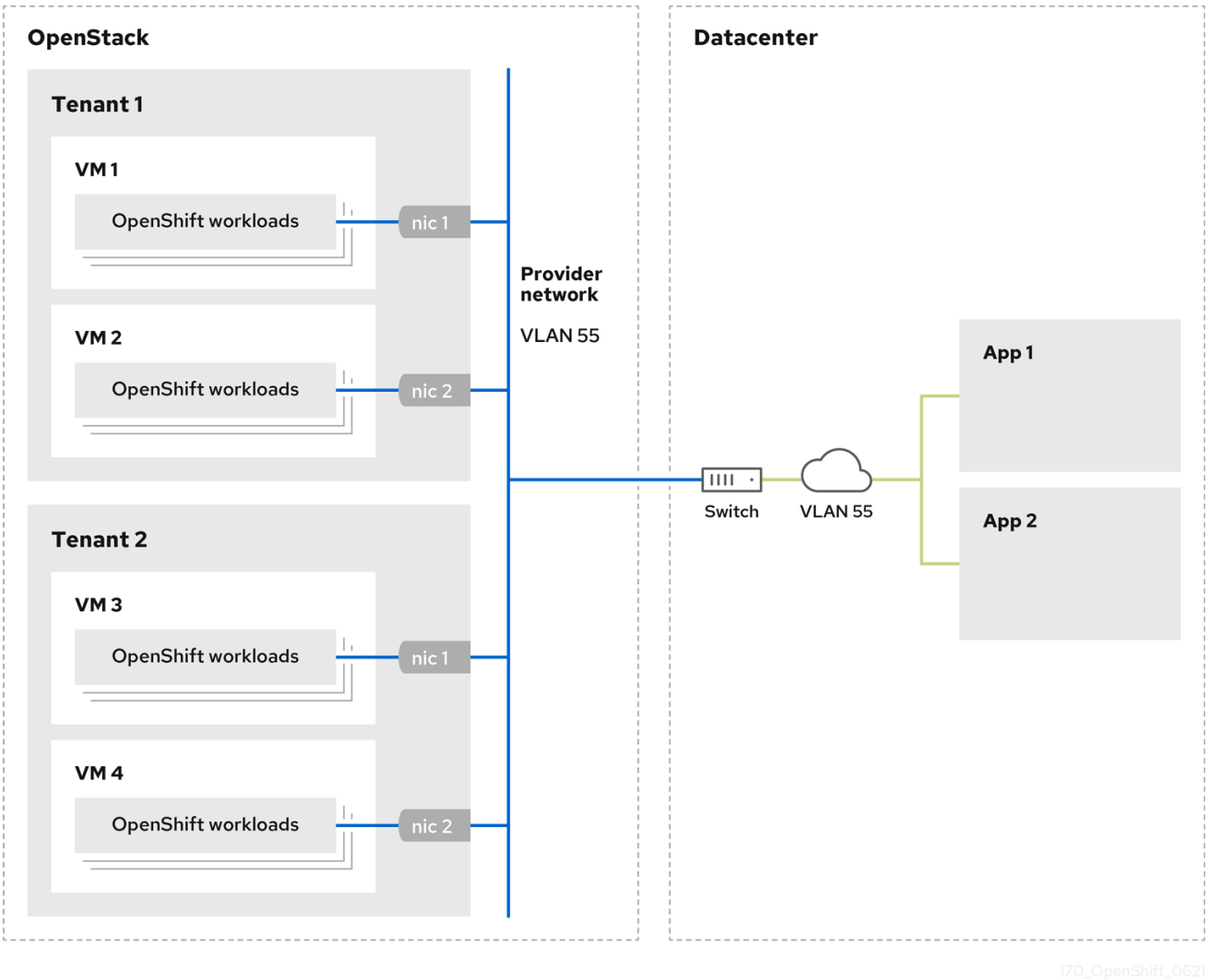
설치 관리자는 **trunkSupport**와 **octaviaSupport**를 자동으로 검색하므로 직접 설정하지 않아도 됩니다. 그러나 사용자 환경이 두 요구사항을 모두 충족하지 않으면 **Kuryr SDN**이 제대로 작동하지 않습니다. **Pod**를 **RHOSP** 네트워크에 연결하려면 트렁크가 필요하며 **OpenShift Container Platform** 서비스를 생성하려면 **Octavia**가 필요합니다.

15.6.14.8. RHOSP 공급자 네트워크에서 클러스터 배포

공급자 네트워크의 기본 네트워크 인터페이스를 사용하여 **RHOSP(Red Hat OpenStack Platform)**에 **OpenShift Container Platform** 클러스터를 배포할 수 있습니다. 공급자 네트워크는 일반적으로 프로젝트에 인터넷에 연결하는 데 사용할 수 있는 공용 네트워크에 직접 액세스할 수 있는 권한을 부여하는 데 사용됩니다. 네트워크 생성 프로세스의 일부로 프로젝트 간에 공급자 네트워크를 공유할 수도 있습니다.

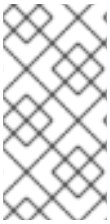
RHOSP 공급자 네트워크는 데이터 센터의 기존 실제 네트워크에 직접 매핑됩니다. **RHOSP** 관리자가 이를 생성해야 합니다.

다음 예에서 **OpenShift Container Platform** 워크로드는 공급자 네트워크를 사용하여 데이터 센터에 연결됩니다.



공급자 네트워크에 설치된 **OpenShift Container Platform** 클러스터에는 테넌트 네트워크 또는 유동 **IP** 주소가 필요하지 않습니다. 설치 프로그램에서 설치하는 동안 이러한 리소스를 생성하지 않습니다.

공급자 네트워크 유형에는 **flat**(태그되지 않음) 및 **VLAN**(802.1Q 태그)이 포함됩니다.



참고

클러스터는 네트워크 유형에서 허용하는 만큼의 공급자 네트워크 연결을 지원할 수 있습니다. 예를 들어 **VLAN** 네트워크는 일반적으로 최대 **4096**개의 연결을 지원합니다.

RHOSP 설명서에서 공급자 및 테넌트 네트워크에 대해 자세히 알아볼 수 있습니다.

15.6.14.8.1. 클러스터 설치를 위한 RHOSP 공급자 네트워크 요구 사항

OpenShift Container Platform 클러스터를 설치하기 전에 **RHOSP**(Red Hat OpenStack Platform) 배포 및 공급자 네트워크가 다음과 같은 여러 조건을 충족해야 합니다.

- RHOSP 네트워킹 서비스(Neutron)가 활성화되어 **RHOSP** 네트워킹 **API**를 통해 액세스할 수 있습니다.
- **RHOSP** 네트워킹 서비스에는 포트 보안 및 허용되는 주소 쌍 확장 기능이 활성화되어 있습니다.
- 공급자 네트워크는 다른 테넌트와 공유할 수 있습니다.

작은 정보

openstack network create 명령을 **--share** 플래그와 함께 사용하여 공유할 수 있는 네트워크를 생성합니다.

- 클러스터를 설치하는 데 사용하는 **RHOSP** 프로젝트는 공급자 네트워크와 적절한 서브넷을 소유해야 합니다.

작은 정보

"openshift"라는 프로젝트의 네트워크를 만들려면 다음 명령을 입력합니다.

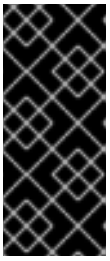
```
$ openstack network create --project openshift
```

"openshift"라는 프로젝트의 서브넷을 만들려면 다음 명령을 입력합니다.

```
$ openstack subnet create --project openshift
```

RHOSP에서 네트워크 생성에 대한 자세한 내용은 공급자 네트워크 설명서를 참조하십시오.

admin 사용자가 클러스터를 소유하는 경우 해당 사용자로 설치 프로그램을 실행하여 네트워크에서 포트를 생성해야 합니다.



중요

공급자 네트워크는 클러스터를 생성하는 데 사용되는 RHOSP 프로젝트에서 소유해야 합니다. RHOSP 컴퓨팅 서비스(Nova)는 해당 네트워크의 포트를 요청할 수 없습니다.

- 공급자 네트워크가 기본적으로 RHOSP 메타데이터 서비스 IP 주소 169.254.169.254에 연결할 수 있는지 확인합니다.

  RHOSP SDN 및 네트워킹 서비스 구성에 따라 서브넷을 생성할 때 경로를 제공해야 할 수 있습니다. 예를 들면 다음과 같습니다.

  ```
  $ openstack subnet create --dhcp --host-route
  destination=169.254.169.254/32,gateway=192.0.2.2 ...
  ```

- 선택 사항: 네트워크를 보호하려면 단일 프로젝트에 대한 네트워크 액세스를 제한하는 역할 기반 액세스 제어(RBAC) 규칙을 생성합니다.

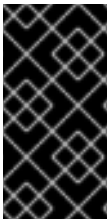
15.6.14.8.2. 공급자 네트워크에 기본 인터페이스가 있는 클러스터 배포

RHOSP(Red Hat OpenStack Platform) 공급자 네트워크에 기본 네트워크 인터페이스가 있는 OpenShift Container Platform 클러스터를 배포할 수 있습니다.

- RHOSP(Red Hat OpenStack Platform) 배포는 "클러스터 설치를 위한 RHOSP 공급자 네트워크 요구 사항"에 설명된 대로 구성됩니다.

프로세스

1. 텍스트 편집기에서 install-config.yaml 파일을 엽니다.
2. platform.openstack.apiVIP 속성의 값을 API VIP의 IP 주소로 설정합니다.
3. platform.openstack.ingressVIP 속성 값을 Ingress VIP의 IP 주소로 설정합니다.
4. platform.openstack.machinesSubnet 속성 값을 공급자 네트워크 서브넷의 UUID로 설정합니다.
5. networking.machineNetwork.cidr 속성 값을 공급자 네트워크 서브넷의 CIDR 블록으로 설정합니다.



중요

platform.openstack.apiVIP 및 platform.openstack.ingressVIP 속성은 모두 networking.machineNetwork.cidr 블록에서 할당되지 않은 IP 주소여야 합니다.

RHOSP 공급자 네트워크를 사용하는 클러스터의 설치 구성 파일 섹션

```
...
platform:
  openstack:
    apiVIP: 192.0.2.13
    ingressVIP: 192.0.2.23
    machinesSubnet: fa806b2f-ac49-4bce-b9db-124bc64209bf
    # ...
networking:
  machineNetwork:
  - cidr: 192.0.2.0/24
```

주의

기본 네트워크 인터페이스에 공급자 네트워크를 사용하는 동안 **platform.openstack.externalNetwork** 또는 **platform.openstack.externalDNS** 매개변수를 설정할 수 없습니다.

클러스터를 배포할 때 설치 프로그램에서 **install-config.yaml** 파일을 사용하여 공급자 네트워크에 클러스터를 배포합니다.

작은 정보

공급자 네트워크를 포함한 다른 네트워크를 **platform.openstack.additionalNetworkIDs** 목록에 추가할 수 있습니다.

클러스터를 배포한 후 **Pod**를 추가 네트워크에 연결할 수 있습니다. 자세한 내용은 여러 네트워크 이해를 참조하십시오.

15.6.14.9. Kuryr 포트 풀

Kuryr 포트 풀은 **Pod** 생성을 위해 대기 중인 다수의 포트를 유지 관리합니다.

포트를 대기 상태로 유지하면 **Pod** 생성 시간이 최소화됩니다. 포트 풀이 없으면 **Kuryr**는 **Pod**를 생성하거나 삭제할 때마다 포트 생성 또는 삭제를 명시적으로 요청해야 합니다.

Kuryr가 사용하는 **Neutron** 포트는 네임스페이스에 연결된 서브넷에 생성됩니다. 이러한 **Pod** 포트도 **OpenShift Container Platform** 클러스터 노드의 기본 포트에 하위 포트로 추가됩니다.

Kuryr는 각 네임스페이스를 별도의 서브넷에 유지하므로 각 네임스페이스-작업자 쌍에 대해 별도의 포트 풀이 유지됩니다.

클러스터를 설치하기 전에 **cluster-network-03-config.yml** 매니페스트 파일에서 다음 매개변수를 설정하여 포트 풀 동작을 구성할 수 있습니다.

- enablePortPoolsPrepopulation 매개변수는 풀 사전 채우기를 제어하므로 새 호스트가 추가되거나 새 네임스페이스가 생성되는 경우와 같이 풀 생성 시 Kuryr가 풀에 포트를 추가하도록 합니다. 기본값은 false입니다.

- poolMinPorts 매개변수는 풀에 보관되는 사용 가능한 최소 포트 수입니다. 기본값은 1 입니다.

- poolMaxPorts 매개변수는 풀에 보관되는 사용 가능한 최대 포트 수입니다. 값이 0이면 해당 상한이 비활성화됩니다. 이 설정은 기본 설정입니다.

  OpenStack 포트 할당량이 낮거나 pod 네트워크에 IP 주소가 제한된 경우 이 옵션을 설정하여 불필요한 포트가 삭제되었는지 확인합니다.

- poolBatchPorts 매개 변수는 한 번에 생성할 수 있는 최대 Neutron 포트 수를 정의합니다. 기본값은 3입니다.

15.6.14.10. 설치 중에 Kuryr 포트 풀 조정

설치하는 동안 Kuryr가 RHOSP(Red Hat OpenStack Platform) Neutron 포트를 관리하여 Pod 생성 속도와 효율성을 제어하는 방법을 구성할 수 있습니다.

사전 요구 사항

- install-config.yaml 파일을 생성하고 수정합니다.

프로세스

1. 명령줄에서 매니페스트 파일을 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1. <installation_directory>는 클러스터의 install-config.yaml 파일이 포함된 디렉터리의 이름을 지정합니다.

2. <installation_directory>/ manifests/ 디렉토리에 cluster-network-03-config.yml이라는

이름으로 파일을 만듭니다.

```
$ touch <installation_directory>/manifests/cluster-network-03-config.yml 1
```

1

<installation_directory>는 클러스터의 manifests / 디렉터리가 포함된 디렉터리 이름을 지정합니다.

파일이 생성되면 다음과 같이 여러 네트워크 구성 파일이 manifests/ 디렉토리에 나타납니다.

```
$ ls <installation_directory>/manifests/cluster-network-*
```

출력 예

```
cluster-network-01-crd.yml
cluster-network-02-config.yml
cluster-network-03-config.yml
```

3. 편집기에서 cluster-network-03-config.yml 파일을 열고 원하는 Cluster Network Operator 구성을 설명하는 CR(사용자 정의 리소스)을 입력합니다.

```
$ oc edit networks.operator.openshift.io cluster
```

4. 요구 사항에 맞게 설정을 편집합니다. 다음 파일은 예제로 제공됩니다.

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  serviceNetwork:
  - 172.30.0.0/16
  defaultNetwork:
    type: Kuryr
```

```
kuryrConfig:
  enablePortPoolsPrepopulation: false 1
  poolMinPorts: 1 2
  poolBatchPorts: 3 3
  poolMaxPorts: 5 4
  openstackServiceNetwork: 172.30.0.0/15 5
```

1

Kuryr가 네임스페이스를 생성하거나 클러스터에 새 노드를 추가한 후 Kuryr가 새 Neutron 포트를 생성하도록 하려면 enablePortPoolsPrepopulation 값을 true로 설정합니다. 이 설정은 Neutron 포트 할당량을 높이지만 pod를 생성하는 데 필요한 시간을 줄일 수 있습니다. 기본값은 false입니다.

2

Kuryr는 풀의 사용 가능한 포트 수가 poolMinPorts 값보다 낮은 경우 풀에 대한 새 포트를 만듭니다. 기본값은 1 입니다.

3

poolBatchPorts는 사용 가능한 포트 수가 poolMinPorts 값보다 낮은 경우 생성되는 새 포트 수를 제어합니다. 기본값은 3입니다.

4

풀에서 사용 가능한 포트 수가 poolMaxPorts 값보다 크면 Kuryr는 숫자가 해당 값과 일치할 때까지 해당 포트를 삭제합니다. 이 값을 0으로 설정하면 이 상한이 비활성화되므로 풀이 축소되지 않습니다. 기본값은 0입니다.

5

openStackServiceNetwork 매개변수는 IP 주소가 RHOSP Octavia의 LoadBalancers에 할당되는 네트워크의 CIDR 범위를 정의합니다.

이 매개변수가 Amphora 드라이버와 함께 사용되는 경우 Octavia는 각 로드 밸런서에 대해 이 네트워크에서 두 개의 IP 주소를 사용합니다. 하나는 OpenShift 및 VRRP 연결에 사용됩니다. 이러한 IP 주소는 각각 OpenShift Container Platform 및 Neutron에서 관리하므로 서로 다른 풀에서 가져와야 합니다. 따라서 openStackServiceNetwork의 값은 serviceNetwork 값의 2배 이상이어야 하며 serviceNetwork의 값은 openStackServiceNetwork에서 정의된 범위와 완전히 겹쳐야 합니다.

CNO는 이 매개변수에서 정의한 범위에서 가져온 VRRP IP 주소가 serviceNetwork 매개변수에 정의된 범위와 겹치지 않음을 확인합니다.

이 매개변수가 설정되지 않은 경우 CNO는 접두사 크기를 1로 줄임으로써 결정되는 serviceNetwork의 확장된 값을 사용합니다.

5. cluster-network-03-config.yml 파일을 저장하고 텍스트 편집기를 종료합니다.

6. 선택사항: manifests/cluster-network-03-config.yml 파일을 백업합니다. 설치 프로그램은 클러스터를 생성하는 동안 manifests/ 디렉터리를 삭제합니다.

15.6.14.11. 시스템의 사용자 지정 서브넷 설정

설치 프로그램이 기본적으로 사용하는 IP 범위가 OpenShift Container Platform을 설치할 때 생성하는 Neutron 서브넷과 일치하지 않을 수 있습니다. 필요한 경우 설치 구성 파일을 편집하여 새 시스템의 CIDR 값을 업데이트하십시오.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램에서 생성된 install-config.yaml 파일이 있습니다.

프로세스

1. 명령줄에서 install-config.yaml이 들어 있는 디렉터리를 찾습니다.

2. 해당 디렉터리에서 스크립트를 실행하여 install-config.yaml 파일을 편집하거나 파일을 수동으로 업데이트합니다.

   - 스크립트를 사용하여 값을 설정하려면 다음을 실행합니다.

```
$ python -c '
import yaml;
path = "install-config.yaml";
data = yaml.safe_load(open(path));
data["networking"]["machineNetwork"] = [{"cidr": "192.168.0.0/18"}]; 1
open(path, "w").write(yaml.dump(data, default_flow_style=False))'
```

1 원하는 Neutron 서브넷과 일치하는 값을 삽입합니다(예: 192.0.2.0/24).

- 값을 수동으로 설정하려면 파일을 열고 **networking.machineCIDR** 값을 원하는 **Neutron** 서브넷과 일치하는 값으로 설정합니다.

15.6.14.12. 컴퓨팅 시스템 풀 비우기

사용자 인프라를 사용하는 설치를 계속 진행하려면 설치 구성 파일의 컴퓨팅 시스템 수를 **0**으로 설정합니다. 이러한 시스템은 나중에 수동으로 생성합니다.

사전 요구 사항

- **OpenShift Container Platform** 설치 프로그램에서 생성된 **install-config.yaml** 파일이 있습니다.

프로세스

1. 명령줄에서 **install-config.yaml**이 들어 있는 디렉터리를 찾습니다.

2. 해당 디렉터리에서 스크립트를 실행하여 **install-config.yaml** 파일을 편집하거나 파일을 수동으로 업데이트합니다.

   - 스크립트를 사용하여 값을 설정하려면 다음을 실행합니다.

     ```
     $ python -c '
     import yaml;
     path = "install-config.yaml";
     data = yaml.safe_load(open(path));
     data["compute"][0]["replicas"] = 0;
     open(path, "w").write(yaml.dump(data, default_flow_style=False))'
     ```

   - 값을 수동으로 설정하려면 파일을 열고 **compute.<first entry>.replicas** 값을 **0**으로 설정합니다.

15.6.14.13. 네트워크 유형 수정

기본적으로 설치 프로그램은 **OpenShiftSDN** 네트워크 유형을 선택합니다. 대신 **Kuryr**를 사용하려면 프로그램에서 생성된 설치 구성 파일의 값을 변경합니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램에서 생성된 install-config.yaml 파일이 있습니다.

프로세스

1. 명령 프롬프트에서 install-config.yaml이 들어 있는 디렉터리로 이동합니다.

2. 해당 디렉터리에서 스크립트를 실행하여 install-config.yaml 파일을 편집하거나 파일을 수동으로 업데이트합니다.

   - 스크립트를 사용하여 값을 설정하려면 다음을 실행합니다.

```
$ python -c '
import yaml;
path = "install-config.yaml";
data = yaml.safe_load(open(path));
data["networking"]["networkType"] = "Kuryr";
open(path, "w").write(yaml.dump(data, default_flow_style=False))'
```

   - 값을 수동으로 설정하려면 파일을 열고 networking.networkType을 "Kuryr"로 설정합니다.

15.6.15. Kubernetes 매니페스트 및 Ignition 설정 파일 생성

일부 클러스터 정의 파일을 수정하고 클러스터 시스템을 수동으로 시작해야 하므로 클러스터가 시스템을 구성하는 데 필요한 Kubernetes 매니페스트 및 Ignition 구성 파일을 사용자가 생성해야 합니다.

설치 구성 파일은 Kubernetes 매니페스트로 변환됩니다. 매니페스트는 나중에 클러스터 머신을 구성하는 데 사용되는 Ignition 구성 파일로 래핑됩니다.

중요

- **OpenShift Container Platform** 설치 프로그램에서 생성하는 **Ignition** 구성 파일에 **24**시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 **24**시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 **kubelet** 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청**(CSR)**을 수동으로 승인해야 합니다. 자세한 내용은 *만료된 컨트롤 플레인 인증서에서 복구* 문서를 참조하십시오.

- 클러스터를 설치한 후 **24**시간에서 **22**시간까지의 인증서가 교체되기 때문에 생성된 후 **12**시간 이내에 **Ignition** 구성 파일을 사용하는 것이 좋습니다. **12**시간 이내에 **Ignition** 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

사전 요구 사항

- **OpenShift Container Platform** 설치 프로그램을 가져오셨습니다.

- **install-config.yaml** 설치 구성 파일을 생성하셨습니다.

절차

1. **OpenShift Container Platform** 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터에 대한 **Kubernetes** 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>**는 사용자가 만든 **install-config.yaml** 파일이 포함된 설치 디렉터리를 지정합니다.

2. 컨트롤 플레인 시스템 및 컴퓨팅 머신 세트를 정의하는 **Kubernetes** 매니페스트 파일을 제거합니다.

```
$ rm -f openshift/99_openshift-cluster-api_master-machines-*.yaml
openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

이러한 리소스는 사용자가 직접 생성하고 관리하기 때문에 초기화할 필요가 없습니다.

- 시스템 API로 머신 세트 파일을 보존하여 컴퓨팅 시스템을 생성할 수 있지만 사용자 환경과 일치하도록 해당 참조를 업데이트해야 합니다.

3. <installation_directory>/manifests/cluster-scheduler-02-config.yml Kubernetes 매니페스트 파일의 mastersSchedulable 매개변수가 false로 설정되어 있는지 확인합니다. 이 설정으로 인해 컨트롤 플레인 머신에서 포드가 예약되지 않습니다.

   a. <installation_directory>/manifests/cluster-scheduler-02-config.yml 파일을 엽니다.

   b. mastersSchedulable 매개변수를 찾아서 값을 False로 설정되어 있는지 확인합니다.

   c. 파일을 저장하고 종료합니다.

4. Ignition 구성 파일을 생성하려면 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행합니다.

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 <installation_directory>는 동일한 설치 디렉터리를 지정합니다.

설치 디렉터리의 부트스트랩, 컨트롤 플레인 및 컴퓨팅 노드에 대한 Ignition 구성 파일이 생성됩니다. kubeadmin-password 및 kubeconfig 파일은 ./<installation_directory>/auth 디렉터리에 생성됩니다.

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

5. 메타데이터 파일의 **infraID** 키를 환경 변수로 내보냅니다.

```
$ export INFRA_ID=$(jq -r .infraID metadata.json)
```

작은 정보

metadata.json에서 **infraID** 키를 추출하여 사용자가 생성하는 모든 **RHOSP** 리소스의 접두사로 사용합니다. 이렇게 하면 동일한 프로젝트에서 여러 배포를 수행할 때 이름 충돌을 방지할 수 있습니다.

15.6.16. 부트스트랩 Ignition 파일 준비

OpenShift Container Platform 설치 프로세스는 부트스트랩 **Ignition** 구성 파일에서 생성되는 부트스트랩 시스템에 의존합니다.

파일을 편집하고 업로드합니다. 그런 다음 **RHOSP(Red Hat OpenStack Platform)**에서 기본 파일을 다운로드하는 데 사용하는 보조 부트스트랩 **Ignition** 구성 파일을 만듭니다.

사전 요구 사항

- 설치 관리자에서 생성되는 부트스트랩 **Ignition** 파일, **bootstrap.ign**이 있습니다.
- 설치 관리자 메타데이터 파일의 인프라 **ID**는 환경 변수(**$ INFRA_ID**)로 설정됩니다.
  - 변수가 설정되지 않은 경우 **Kubernetes** 매니페스트 및 **Ignition** 구성 파일 생성을 참조하십시오.
- 부트스트랩 **Ignition** 파일을 저장할 수 있는 **HTTP(S)** 액세스 가능 방법이 있습니다.
  - 문서화된 프로시저에는 **RHOSP** 이미지 서비스(**Glance**)를 사용하지만 **RHOSP** 스토리지 서비스(**Swift**), **Amazon S3**, 내부 **HTTP** 서버 또는 애드혹 **Nova** 서버를 사용할 수도 있습니다.

프로세스

1. 다음 **Python** 스크립트를 실행합니다. 이 스크립트는 부트스트랩 **Ignition** 파일을 수정하여 호스트 이름 및 사용 가능한 경우 실행 시 **CA** 인증서 파일을 설정합니다.

```
import base64
import json
import os

with open('bootstrap.ign', 'r') as f:
    ignition = json.load(f)

files = ignition['storage'].get('files', [])

infra_id = os.environ.get('INFRA_ID', 'openshift').encode()
hostname_b64 = base64.standard_b64encode(infra_id + b'-
bootstrap\n').decode().strip()
files.append(
{
    'path': '/etc/hostname',
    'mode': 420,
    'contents': {
        'source': 'data:text/plain;charset=utf-8;base64,' + hostname_b64
    }
})

ca_cert_path = os.environ.get('OS_CACERT', '')
if ca_cert_path:
    with open(ca_cert_path, 'r') as f:
        ca_cert = f.read().encode()
        ca_cert_b64 = base64.standard_b64encode(ca_cert).decode().strip()

    files.append(
    {
        'path': '/opt/openshift/tls/cloud-ca-cert.pem',
        'mode': 420,
        'contents': {
            'source': 'data:text/plain;charset=utf-8;base64,' + ca_cert_b64
        }
    })

ignition['storage']['files'] = files;

with open('bootstrap.ign', 'w') as f:
    json.dump(ignition, f)
```

2. RHOSP CLI를 사용하여 부트스트랩 Ignition 파일을 사용하는 이미지를 생성합니다.

```
$ openstack image create --disk-format=raw --container-format=bare --file
bootstrap.ign <image_name>
```

3. 이미지의 세부 사항을 가져옵니다.

```
$ openstack image show <image_name>
```

file 값을 기록해 둡니다. 해당 패턴은 **v2/images/<image_ID>/file**입니다.

참고

생성된 이미지가 활성화되어 있는지 확인합니다.

4. 이미지 서비스의 공용 주소를 검색합니다.

```
$ openstack catalog show image
```

5. 공용 주소를 이미지 **file** 값과 결합하고 그 결과를 저장 위치로 저장합니다. 위치 패턴은 **<image_service_public_URL>/v2/images/<image_ID>/file**입니다.

6. 인증 토큰을 생성하고 토큰 **ID**를 저장합니다.

```
$ openstack token issue -c id -f value
```

7. **$INFRA_ID-bootstrap-ignition.json** 파일에 다음 내용을 삽입하고 사용자 값과 일치하도록 자리 표시자를 편집합니다.

```
{
  "ignition": {
    "config": {
      "merge": [{
        "source": "<storage_url>", 1
        "httpHeaders": [{
          "name": "X-Auth-Token", 2
          "value": "<token_ID>" 3
        }]
      }]
    },
    "security": {
      "tls": {
        "certificateAuthorities": [{
          "source": "data:text/plain;charset=utf-8;base64,<base64_encoded_certificate>"
4
        }]
      }
    },
    "version": "3.2.0"
  }
}
```

-

1

ignition.config.merge.source 값을 부트스트랩 **Ignition** 파일 스토리지 **URL**로 바꿉니다.

2

httpHeaders의 **name**을 **"X-Auth-Token"**으로 설정합니다.

3

httpHeaders의 **value**를 토큰의 **ID**로 설정합니다.

4

부트스트랩 **Ignition** 파일 서버가 자체 서명 인증서를 사용하는 경우 **base64** 인코딩 인증서를 포함합니다.

8. 보조 **Ignition** 구성 파일을 저장합니다.

부트스트랩 **Ignition** 데이터는 설치 중에 **RHOSP**로 전달됩니다.

주의

부트스트랩 **Ignition** 파일에는 **clouds.yaml** 자격 증명과 같은 민감한 정보가 들어 있습니다. 파일을 안전한 곳에 저장하고 설치 프로세스를 완료한 후에는 삭제해야 합니다.

15.6.17. RHOSP에서 컨트롤 플레인 Ignition 구성 파일 생성

자체 인프라의 **RHOSP(Red Hat OpenStack Platform)**에 **OpenShift Container Platform**을 설치하려면 컨트롤 플레인 **Ignition** 구성 파일이 필요합니다. 구성 파일은 여러 개 만들어야 합니다.

참고

부트스트랩 Ignition 구성과 마찬가지로 각 컨트롤 플레인 시스템의 호스트 이름을 명시적으로 정의해야 합니다.

사전 요구 사항

- 설치 프로그램 메타데이터 파일의 인프라 ID는 환경 변수($ INFRA_ID)로 설정됩니다.
  - 변수가 설정되지 않은 경우 “Kubernetes 매니페스트 및 Ignition 구성 파일 생성”을 참조하십시오.

프로세스

- 명령줄에서 다음 Python 스크립트를 실행합니다.

```
$ for index in $(seq 0 2); do
    MASTER_HOSTNAME="$INFRA_ID-master-$index\n"
    python -c "import base64, json, sys;
ignition = json.load(sys.stdin);
storage = ignition.get('storage', {});
files = storage.get('files', []);
files.append({'path': '/etc/hostname', 'mode': 420, 'contents': {'source':
'data:text/plain;charset=utf-8;base64,' +
base64.standard_b64encode(b'$MASTER_HOSTNAME').decode().strip(), 'verification':
{}}, 'filesystem': 'root'});
storage['files'] = files;
ignition['storage'] = storage
json.dump(ignition, sys.stdout)" <master.ign >"$INFRA_ID-master-$index-
ignition.json"
done
```

이제 <INFRA_ID>-master-0-ignition.json, <INFRA_ID>-master-1-ignition.json, <INFRA_ID>-master-2-ignition.json의 세 가지 컨트롤 플레인 Ignition 파일이 있습니다.

15.6.18. RHOSP에서 네트워크 리소스 생성

자체 인프라의 RHOSP(Red Hat OpenStack Platform)에 OpenShift Container Platform을 설치하는 데 필요한 네트워크 리소스를 생성합니다. 시간을 절약하려면 보안 그룹, 네트워크, 서브넷, 라우터 및 포트를 생성하는 제공된 Ansible 플레이북을 실행합니다.

사전 요구 사항

- Python 3가 시스템에 설치되어 있습니다.
- "플레이북 종속 항목 다운로드"에서 모듈을 다운로드했습니다.
- "설치 플레이북 다운로드"에서 플레이북을 다운로드했습니다.

프로세스

1. 선택사항: inventory.yaml 플레이북에 외부 네트워크값을 추가합니다.

inventory.yaml Ansible 플레이북의 외부 네트워크 값 예

```
...
    # The public network providing connectivity to the cluster. If not
    # provided, the cluster external connectivity must be provided in another
    # way.

    # Required for os_api_fip, os_ingress_fip, os_bootstrap_fip.
    os_external_network: 'external'
...
```

중요

inventory.yaml 파일에 os_external_network 값을 지정하지 않은 경우 VM이 Glance 및 외부 연결에 직접 액세스할 수 있는지 확인해야 합니다.

2. 선택사항: 외부 네트워크 및 부동 IP(FIP) 주소값을 inventory.yaml 플레이북에 추가합니다.

inventory.yaml Ansible 플레이북의 FIP 값 예

```
...
    # OpenShift API floating IP address. If this value is non-empty, the
    # corresponding floating IP will be attached to the Control Plane to
```

```
# serve the OpenShift API.
os_api_fip: '203.0.113.23'

# OpenShift Ingress floating IP address. If this value is non-empty, the
# corresponding floating IP will be attached to the worker nodes to serve
# the applications.
os_ingress_fip: '203.0.113.19'

# If this value is non-empty, the corresponding floating IP will be
# attached to the bootstrap machine. This is needed for collecting logs
# in case of install failure.
os_bootstrap_fip: '203.0.113.20'
```

중요

os_api_fip 및 **os_ingress_fip**에 대한 값을 정의하지 않으면 설치 후 네트워크 구성을 수행해야 합니다.

os_bootstrap_fip의 값을 정의하지 않으면 설치 프로그램이 실패한 설치에서 디버깅 정보를 다운로드할 수 없습니다.

자세한 정보는 "환경에 대한 액세스 활성화"를 참조하십시오.

3. 명령줄에서 **security-groups.yaml** 플레이북을 실행하여 보안 그룹을 생성합니다.

```
$ ansible-playbook -i inventory.yaml security-groups.yaml
```

4. 명령줄에서 **network.yaml** 플레이북을 실행하여 네트워크, 서브넷 및 라우터를 생성합니다.

```
$ ansible-playbook -i inventory.yaml network.yaml
```

5. 선택사항: **Nova** 서버가 사용하는 기본 해결 프로그램을 제어하려면 **RHOSP CLI** 명령을 실행합니다.

```
$ openstack subnet set --dns-nameserver <server_1> --dns-nameserver <server_2>
"$INFRA_ID-nodes"
```

15.6.19. RHOSP에서 부트스트랩 시스템 생성

부트스트랩 시스템을 생성하고 RHOSP(Red Hat OpenStack Platform)에서 실행하는 데 필요한 네트워크 액세스 권한을 부여합니다. Red Hat은 이 프로세스를 간소화하기 위해 실행하는 Ansible 플레이북을 제공합니다.

사전 요구 사항

- "플레이북 종속 항목 다운로드"에서 모듈을 다운로드했습니다.
- "설치 플레이북 다운로드"에서 플레이북을 다운로드했습니다.
- inventory.yaml, common.yaml 및 bootstrap.yaml Ansible 플레이북이 공통 디렉터리에 있습니다.
- 설치 프로그램에서 생성된 metadata.json 파일이 Ansible 플레이북과 동일한 디렉터리에 있습니다.

프로세스

1. 명령줄에서 작업 디렉터리를 플레이북의 위치로 변경합니다.
2. 명령줄에서 bootstrap.yaml 플레이북을 실행합니다.

```
$ ansible-playbook -i inventory.yaml bootstrap.yaml
```

3. 부트스트랩 서버가 활성화된 후 로그를 보고 Ignition 파일이 수신되었는지 확인합니다.

```
$ openstack console log show "$INFRA_ID-bootstrap"
```

15.6.20. RHOSP에 컨트롤 플레인 시스템 생성

사용자가 생성한 Ignition 구성 파일을 사용하여 세 개의 컨트롤 플레인 시스템을 생성합니다. Red Hat은 이 프로세스를 간소화하기 위해 실행하는 Ansible 플레이북을 제공합니다.

사전 요구 사항

- "플레이북 종속 항목 다운로드"에서 모듈을 다운로드했습니다.
- "설치 플레이북 다운로드"에서 플레이북을 다운로드했습니다.
- 설치 프로그램 메타데이터 파일의 인프라 ID는 환경 변수($ INFRA_ID)로 설정됩니다.
- inventory.yaml, common.yaml 및 control-plane.yaml Ansible 플레이북이 공통 디렉터리에 있습니다.
- "컨트롤 플레인 Ignition 구성 파일 생성"에서 생성된 세 개의 Ignition 파일이 있습니다.

프로세스

1. 명령줄에서 작업 디렉터리를 플레이북의 위치로 변경합니다.
2. 컨트롤 플레인 Ignition 구성 파일이 작업 디렉터리에 없으면 파일을 디렉터리에 복사합니다.
3. 명령줄에서 control-plane.yaml 플레이북을 실행합니다.

```
$ ansible-playbook -i inventory.yaml control-plane.yaml
```

4. 다음 명령을 실행하여 부트스트랩 프로세스를 모니터링합니다.

```
$ openshift-install wait-for bootstrap-complete
```

컨트롤 플레인 시스템이 실행 중이고 클러스터에 연결되었음을 확인하는 메시지가 표시됩니다.

```
INFO API v1.22.1 up
INFO Waiting up to 30m0s for bootstrapping to complete...
...
INFO It is now safe to remove the bootstrap resources
```

15.6.21. CLI를 사용하여 클러스터에 로그인

클러스터 kubeconfig 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. kubeconfig 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- oc CLI를 설치했습니다.

프로세스

1. kubeadmin 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 oc 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

15.6.22. RHOSP에서 부트스트랩 리소스 삭제

더 이상 필요하지 않은 부트스트랩 리소스는 삭제합니다.

사전 요구 사항

사전 요구 사항

- "플레이북 종속 항목 다운로드"에서 모듈을 다운로드했습니다.
- "설치 플레이북 다운로드"에서 플레이북을 다운로드했습니다.
- inventory.yaml, common.yaml 및 down-bootstrap.yaml Ansible 플레이북이 공통 디렉터리에 있습니다.
- 컨트롤 플레인 시스템이 실행 중입니다.
  - 시스템 상태를 모르는 경우 “클러스터 상태 확인”을 참조하십시오.

프로세스

1. 명령줄에서 작업 디렉터리를 플레이북의 위치로 변경합니다.
2. 명령줄에서 down-bootstrap.yaml 플레이북을 실행합니다.

```
$ ansible-playbook -i inventory.yaml down-bootstrap.yaml
```

부트스트랩 포트, 서버 및 부동 IP 주소가 삭제됩니다.

주의

이전에 부트스트랩 Ignition 파일 URL을 비활성화하지 않은 경우 지금 비활성화합니다.

15.6.23. RHOSP에서 컴퓨팅 시스템 생성

컨트롤 플레인을 가동시킨 후 컴퓨팅 시스템을 만듭니다. Red Hat은 이 프로세스를 간소화하기 위해 실행하는 Ansible 플레이북을 제공합니다.

사전 요구 사항

- "플레이북 종속 항목 다운로드"에서 모듈을 다운로드했습니다.
- "설치 플레이북 다운로드"에서 플레이북을 다운로드했습니다.
- inventory.yaml, common.yaml 및 compute-nodes.yaml Ansible 플레이북이 공통 디렉터리에 있습니다.
- 설치 프로그램에서 생성된 metadata.json 파일이 Ansible 플레이북과 동일한 디렉터리에 있습니다.
- 컨트롤 플레인이 활성화되었습니다.

프로세스

1. 명령줄에서 작업 디렉터리를 플레이북의 위치로 변경합니다.
2. 명령줄에서 플레이북을 실행합니다.

```
$ ansible-playbook -i inventory.yaml compute-nodes.yaml
```

다음 단계

- 시스템의 인증서 서명 요청을 승인합니다.

15.6.24. 머신의 인증서 서명 요청 승인

클러스터에 시스템을 추가하면 추가한 시스템별로 보류 중인 인증서 서명 요청(CSR)이 두 개씩 생성됩니다. 이러한 CSR이 승인되었는지 확인해야 하며, 필요한 경우 이를 직접 승인해야 합니다. 클라이언트 요청을 먼저 승인한 다음 서버 요청을 승인해야 합니다.

사전 요구 사항

- 클러스터에 시스템을 추가했습니다.

프로세스

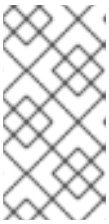
1. 클러스터가 시스템을 인식하는지 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0  Ready    master  63m  v1.22.1
master-1  Ready    master  63m  v1.22.1
master-2  Ready    master  64m  v1.22.1
```

출력에 생성된 모든 시스템이 나열됩니다.



참고

이전 출력에는 일부 CSR이 승인될 때까지 컴퓨팅 노드(작업자 노드라고도 함)가 포함되지 않을 수 있습니다.

2. 보류 중인 CSR을 검토하고 클러스터에 추가한 각 시스템에 대해 Pending 또는 Approved 상태의 클라이언트 및 서버 요청이 표시되는지 확인합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE    REQUESTOR                                                   CONDITION
csr-8b2br   15m    system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
```

```
csr-8vnps   15m    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

예에서는 두 시스템이 클러스터에 참여하고 있습니다. 목록에는 승인된 CSR이 더 많이 나타날 수도 있습니다.

3.

CSR이 승인되지 않은 경우, 추가된 시스템에 대한 모든 보류 중인 CSR이 Pending 상태로 전환된 후 클러스터 시스템의 CSR을 승인합니다.

참고

CSR은 교체 주기가 자동으로 만료되므로 클러스터에 시스템을 추가한 후 1 시간 이내에 CSR을 승인하십시오. 한 시간 내에 승인하지 않으면 인증서가 교체되고 각 노드에 대해 두 개 이상의 인증서가 표시됩니다. 이러한 인증서를 모두 승인해야 합니다. 클라이언트 CSR이 승인되면 Kubelet은 인증서에 대한 보조 CSR을 생성하므로 수동 승인이 필요합니다. 그러면 Kubelet에서 동일한 매개변수를 사용하여 새 인증서를 요청하는 경우 인증서 갱신 요청은 machine-approver에 의해 자동으로 승인됩니다.

참고

베어 메탈 및 기타 사용자 프로비저닝 인프라와 같이 머신 API를 사용하도록 활성화되지 않는 플랫폼에서 실행되는 클러스터의 경우 CSR(Kubelet service Certificate Request)을 자동으로 승인하는 방법을 구현해야 합니다. 요청이 승인되지 않으면 API 서버가 kubelet에 연결될 때 서비스 인증서가 필요하므로 oc exec, oc rsh, oc logs 명령을 성공적으로 수행할 수 없습니다. Kubelet 엔드 포인트에 연결하는 모든 작업을 수행하려면 이 인증서 승인이 필요합니다. 이 방법은 새 CSR을 감시하고 CSR이 system:node 또는 system:admin 그룹의 node-bootstrapper 서비스 계정에 의해 제출되었는지 확인하고 노드의 ID를 확인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

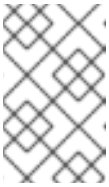
```
$ oc adm certificate approve <csr_name> 1
```

1

<csr_name>은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```



참고

일부 Operator는 일부 CSR이 승인될 때까지 사용할 수 없습니다.

4. 이제 클라이언트 요청이 승인되었으므로 클러스터에 추가한 각 머신의 서버 요청을 검토해야 합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 나머지 CSR이 승인되지 않고 Pending 상태인 경우 클러스터 머신의 CSR을 승인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1

<csr_name>은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs oc adm certificate approve
```

6. 모든 클라이언트 및 서버 CSR이 승인된 후 머신은 Ready 상태가 됩니다. 다음 명령을 실행하여 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0  Ready    master  73m  v1.22.1
master-1  Ready    master  73m  v1.22.1
master-2  Ready    master  74m  v1.22.1
worker-0  Ready    worker  11m  v1.22.1
worker-1  Ready    worker  11m  v1.22.1
```

참고

머신이 Ready 상태로 전환하는 데 서버 CSR의 승인 후 몇 분이 걸릴 수 있습니다.

추가 정보

- CSR에 대한 자세한 내용은 인증서 서명 요청을 참조하십시오.

15.6.25. 설치 성공 확인

OpenShift Container Platform 설치가 완료되었는지 확인합니다.

사전 요구 사항

- 설치 프로그램(openshift-install)이 있습니다.

프로세스

- 명령줄에 다음을 입력합니다.

```
$ openshift-install --log-level debug wait-for install-complete
```

콘솔 URL과 관리자의 로그인 정보가 출력됩니다.

15.6.26. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

15.6.27. 다음 단계

- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.
- 노드 포트에 대한 외부 액세스를 활성화해야 하는 경우 노드 포트를 사용하여 Ingress 클러스터 트래픽을 구성합니다.
- 유동 IP 주소를 통한 애플리케이션 트래픽을 허용하도록 RHOSP를 구성하지 않은 경우 유동 IP 주소로 RHOSP 액세스를 구성합니다.

15.7. 자체 SR-IOV 인프라에 OPENSTACK에 클러스터를 설치

OpenShift Container Platform 4.9에서는 사용자 프로비저닝 인프라에서 실행되고 SR-IOV(Single-root input/output virtualization) 네트워크를 사용하여 컴퓨팅 머신을 실행하는 RHOSP(Red Hat OpenStack Platform)에 클러스터를 설치할 수 있습니다.

자체 인프라를 사용하면 클러스터를 기존 인프라 및 수정 사항과 통합할 수 있습니다. 이 프로세스에서는 Nova 서버, Neutron 포트, 보안 그룹과 같은 모든 RHOSP 리소스를 생성해야 하므로 설치 관리자 프로비저닝 설치보다 사용자가 수행해야 하는 작업이 더 많습니다. 그러나 Red Hat은 배포 프로세스에 도움이 되는 Ansible 플레이북을 제공합니다.

15.7.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.
- OpenShift Container Platform 4.9가 OpenShift 클러스터에 지원되는 플랫폼 섹션을 사용하여 현재 RHOSP 버전과 호환되는지 확인했습니다. RHOSP 지원 매트릭스의 OpenShift Container Platform을 확인하여 여러 버전의 플랫폼 지원을 비교할 수도 있습니다.
- 네트워크 구성이 공급자 네트워크를 사용하지 않습니다. 공급자 네트워크는 지원되지 않습니다.
- OpenShift Container Platform을 설치할 RHOSP 계정을 준비합니다.
- 설치 프로그램을 실행하는 시스템에서 다음을 준비합니다.
  - 설치 프로세스에서 만드는 파일을 저장할 수 있는 단일 디렉터리
  - Python 3

15.7.2. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.

중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

15.7.3. RHOSP에 OpenShift Container Platform을 설치하기 위한 리소스 지침

OpenShift Container Platform 설치를 지원하려면 RHOSP(Red Hat OpenStack Platform) 할당량이 다음 요구사항을 충족해야 합니다.

표 15.29. RHOSP의 기본 OpenShift Container Platform 클러스터에 권장되는 리소스

리소스 이름	값
부동 IP 주소	3
포트	15
라우터	1
서브넷	1
RAM	ECDHEGB

리소스 이름	값
vCPU	22
볼륨 스토리지	275GB
인스턴스	7
보안 그룹	3
보안 그룹 규칙	60

권장 리소스보다 적은 리소스로도 클러스터가 작동할 수 있지만 성능은 보장되지 않습니다.

중요

swiftoperator 역할을 가진 사용자 계정으로 **RHOSP** 개체 스토리지(**Swift**)를 사용하고 운영하는 경우 **OpenShift Container Platform** 이미지 레지스트리의 기본 백엔드로 사용됩니다. 이 경우 볼륨 스토리지 요구사항은 **175GB**입니다. **Swift** 공간 요구사항은 이미지 레지스트리의 크기에 따라 다릅니다.

참고

기본적으로 보안 그룹 및 보안 그룹 규칙 할당량이 적을 수 있습니다. 문제가 발생하면 관리자로 **openstack quota set --secgroups 3 --secgroup-rules 60 <project>**를 실행하여 할당량을 늘립니다.

OpenShift Container Platform 배포는 컨트롤 플레인 시스템, 컴퓨팅 시스템, 부트스트랩 시스템으로 구성됩니다.

15.7.3.1. 컨트롤 플레인 머신

기본적으로 **OpenShift Container Platform** 설치 프로세스는 세 개의 컨트롤 플레인 시스템을 생성합니다.

각 시스템의 요구사항은 다음과 같습니다.

- RHOSP 할당량의 인스턴스
- RHOSP 할당량의 포트
- 최소 16GB 메모리 및 vCPU 4개가 있는 플레이버
- RHOSP 할당량에서 최소 100GB 스토리지 공간

15.7.3.2. 컴퓨팅 머신

기본적으로 OpenShift Container Platform 설치 프로세스는 세 개의 컴퓨팅 시스템을 생성합니다.

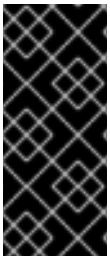
각 시스템의 요구사항은 다음과 같습니다.

- RHOSP 할당량의 인스턴스
- RHOSP 할당량의 포트
- 8GB 메모리 및 vCPU 2개 이상 있는 플레이버
- RHOSP 할당량에서 최소 100GB 스토리지 공간

작은 정보

컴퓨팅 시스템은 OpenShift Container Platform에서 실행하는 애플리케이션을 호스팅합니다. 최대한 많이 실행하는 것이 좋습니다.

또한 SR-IOV(Single-root input/output virtualization)를 사용하는 클러스터의 경우 RHOSP 컴퓨팅 노드에 대규모 페이지를 지원하는 플레이버가 필요합니다.



중요

SR-IOV 배포는 종종 전용 또는 분리된 CPU와 같은 성능 최적화를 사용합니다. 성능을 극대화하려면 이러한 최적화를 사용하도록 기본 RHOSP 배포를 구성한 다음 최적화된 인프라에서 OpenShift Container Platform 컴퓨팅 머신을 실행합니다.

추가 리소스

- 성능 강화를 위한 RHOSP 컴퓨팅 노드 구성에 대한 자세한 내용은 성능 개선을 위한 컴퓨팅 노드 구성을 참조하십시오.

15.7.3.3. 부트스트랩 시스템

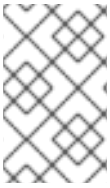
설치하는 동안 컨트롤 플레인을 유지하기 위해 부트스트랩 시스템이 임시로 프로비저닝됩니다. 프로덕션 컨트롤 플레인이 준비되면 부트스트랩 시스템이 프로비저닝 해제됩니다.

부트스트랩 시스템의 요구사항은 다음과 같습니다.

- RHOSP 할당량의 인스턴스
- RHOSP 할당량의 포트
- 최소 16GB 메모리 및 vCPU 4개가 있는 플레이버
- RHOSP 할당량에서 최소 100GB 스토리지 공간

15.7.4. 플레이북 종속 항목 다운로드

사용자 프로비저닝 인프라에서의 설치 프로세스를 간소화하는 Ansible 플레이북에는 몇 가지 Python 모듈이 필요합니다. 설치 관리자를 실행할 시스템에서 모듈 리포지토리를 추가하고 다운로드합니다.



참고

이 방법은 현재 Red Hat Enterprise Linux (RHEL) 8을 사용하는 것으로 가정합니다.

사전 요구 사항

- Python 3가 시스템에 설치되어 있습니다.

프로세스

1. 명령줄에서 리포지토리를 추가합니다.

   a. Red Hat Subscription Manager에 등록합니다.

   ```
   $ sudo subscription-manager register # If not done already
   ```

   b. 최신 서브스크립션 데이터를 가져옵니다.

   ```
   $ sudo subscription-manager attach --pool=$YOUR_POOLID # If not done already
   ```

   c. 현재 리포지토리를 비활성화합니다.

   ```
   $ sudo subscription-manager repos --disable=* # If not done already
   ```

   d. 필요한 리포지토리를 추가합니다.

   ```
   $ sudo subscription-manager repos \
     --enable=rhel-8-for-x86_64-baseos-rpms \
     --enable=openstack-16-tools-for-rhel-8-x86_64-rpms \
     --enable=ansible-2.9-for-rhel-8-x86_64-rpms \
     --enable=rhel-8-for-x86_64-appstream-rpms
   ```

2. 모듈을 설치합니다.

   ```
   $ sudo yum install python3-openstackclient ansible python3-openstacksdk python3-
   netaddr
   ```

3. python 명령이 python3를 가리키는지 확인합니다.

   ```
   $ sudo alternatives --set python /usr/bin/python3
   ```

15.7.5. 설치 플레이북 다운로드

자체 **RHOSP(Red Hat OpenStack Platform)** 인프라에 **OpenShift Container Platform**을 설치하는 데 사용할 수 있는 **Ansible** 플레이북을 다운로드합니다.

사전 요구 사항

- **curl** 명령줄 툴은 사용자의 머신에서 사용할 수 있습니다.

프로세스

- 플레이북을 작업 디렉터리에 다운로드하려면 명령줄에서 다음 스크립트를 실행합니다.

```
$ xargs -n 1 curl -O <<< '
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/bootstrap.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/common.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/compute-nodes.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/control-plane.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/inventory.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/network.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/security-groups.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/down-bootstrap.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/down-compute-nodes.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/down-control-plane.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/down-load-balancers.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/down-network.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/down-security-groups.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
4.9/upi/openstack/down-containers.yaml'
```

플레이북이 사용자의 머신에 다운로드됩니다.

중요

설치 프로세스 중에 플레이북을 수정하여 배포를 구성할 수 있습니다.

클러스터 수명에 대해 모든 플레이북을 유지합니다. RHOSP에서 OpenShift Container Platform 클러스터를 제거하려면 플레이북이 있어야 합니다.

중요

bootstrap.yaml, compute-nodes.yaml, control-plane.yaml, network.yaml 및 security-groups.yaml 파일에서 만든 모든 편집 사항을 down- 접두어가 있는 해당 플레이북과 일치시켜야 합니다. 예를 들어 bootstrap.yaml 파일에 대한 편집도 down-bootstrap.yaml 파일에 반영해야 합니다. 두 파일을 모두 편집하지 않으면 지원되는 클러스터 제거 프로세스가 실패합니다.

15.7.6. 설치 프로그램 받기

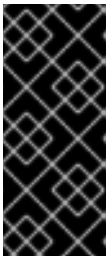
OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

프로세스

1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.
2. 인프라 공급자를 선택합니다.
3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.



중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.

중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 **OpenShift Container Platform** 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 **Linux** 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 **OpenShift Container Platform** 구성 요소에 대한 컨테이너 이미지를 제공하는 **Quay.io**를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

15.7.7. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

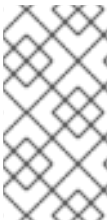
OpenShift Container Platform을 설치하는 동안 **SSH** 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 **Ignition** 구성 파일을 통해 **RHCOS(Red Hat Enterprise Linux CoreOS)** 노드에 전달되며 노드에 대한 **SSH** 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 **core** 사용자의 **~/.ssh/authorized_keys** 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 **core**로 **RHCOS** 노드에 **SSH**로 **SSH** 연결을 수행할 수 있습니다 . **SSH**를 통해 노드에 액세스하려면 로컬 사용자의 **SSH**에서 개인 키 **ID**를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 **SSH**를 실행하려면 설치 프로세스 중에 **SSH** 공용 키를 지정해야 합니다. **./openshift-install gather** 명령에도 **SSH** 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.



참고

AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: ~/.ssh/id_ed25519)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 '~/.ssh 디렉터리에 있는지 확인하십시오.

참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 ed25519 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 rsa 또는 ecdsa 알고리즘을 사용하는 키를 생성합니다.

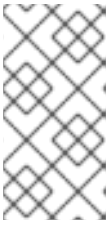
2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 ~/.ssh/id_ed25519.pub 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 ./openshift-install gather 명령을 사용하려는 경우 필요합니다.



참고

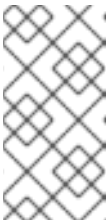
일부 배포에서는 ~/.ssh/id_rsa 및 ~/.ssh/id_dsa와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. ssh-agent 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```



참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. ssh-agent에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1

SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: ~/.ssh/id_ed25519).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

15.7.8. RHCOS(Red Hat Enterprise Linux CoreOS) 이미지 생성

OpenShift Container Platform 설치 프로그램을 사용하려면 RHOSP(Red Hat OpenStack Platform) 클러스터에 RHCOS(Red Hat Enterprise Linux CoreOS) 이미지가 있어야 합니다. 최신 RHCOS 이미지를 검색한 다음 RHOSP CLI를 사용하여 업로드하십시오.

사전 요구 사항

- RHOSP CLI가 설치되어 있습니다.

프로세스

1. Red Hat Customer Portal의 제품 다운로드 페이지에 로그인합니다.

2. Version 에서 Red Hat Enterprise Linux (RHEL) 8용 최신 OpenShift Container Platform 4.9 릴리스를 선택합니다.

중요

RHCOS 이미지는 OpenShift Container Platform 릴리스에 따라 변경되지 않을 수 있습니다. 설치하는 OpenShift Container Platform 버전과 같거나 그 이하의 버전 중 가장 최신 버전의 이미지를 다운로드해야 합니다. 사용 가능한 경우 OpenShift Container Platform 버전과 일치하는 이미지 버전을 사용합니다.

3. *RHCOS(Red Hat Enterprise Linux CoreOS) - OpenStack Image (QCOW)*를 다운로드합니다.

4. 이미지 압축을 풉니다.

참고

RHOSP 이미지 압축을 풀어야 클러스터가 사용할 수 있습니다. 다운로드한 파일의 이름에 .gz 또는 .tgz와 같은 압축 확장자가 포함되지 않을 수 있습니다. 파일 압축 여부를 확인하려면 명령줄에서 다음을 입력합니다.

```
$ file <name_of_downloaded_file>
```

5. 다운로드한 이미지에서 RHOSP CLI를 사용하여 이름이 rhcos인 이미지를 클러스터에 생성합니다.

```
$ openstack image create --container-format=bare --disk-format=qcow2 --file rhcos-${RHCOS_VERSION}-openstack.qcow2 rhcos
```

중요

RHOSP 환경에 따라 이미지를 .raw 또는 .qcow2 형식으로 업로드할 수 있습니다. Ceph를 사용하는 경우 .raw 형식을 사용해야 합니다.

주의

설치 프로그램이 이름이 같은 여러 이미지를 발견하면 그 중 하나를 임의로 선택합니다. 이 동작을 방지하려면 RHOSP에서 리소스의 고유한 이름을 만듭니다.

이미지를 RHOSP에 업로드한 후 설치 프로세스에서 사용할 수 있습니다.

15.7.9. 외부 네트워크 액세스 확인

OpenShift Container Platform 설치 프로세스에는 외부 네트워크에 액세스해야 합니다. 외부 네트워크값을 제공해야 하며 그렇지 않으면 배포가 실패합니다. 프로세스를 시작하기 전에 외부 라우터 유형의 네트워크가 RHOSP(Red Hat OpenStack Platform)에 있는지 확인합니다.

사전 요구 사항

- DHCP 에이전트가 인스턴스의 DNS 쿼리를 전달하도록 OpenStack의 네트워킹 서비스 구성

프로세스

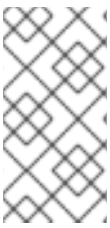
1. RHOSP CLI를 사용하여 '외부' 네트워크의 이름과 ID를 확인합니다.

```
$ openstack network list --long -c ID -c Name -c "Router Type"
```

출력 예

```
+--------------------------------------+--------------+-------------+
| ID                                   | Name         | Router Type |
+--------------------------------------+--------------+-------------+
| 148a8023-62a7-4672-b018-003462f8d7dc | public_network | External    |
+--------------------------------------+--------------+-------------+
```

외부 라우터 유형의 네트워크가 네트워크 목록에 나타납니다. 네트워크가 하나 이상 나타나지 않으면 기본 부동 IP 네트워크 생성 및 기본 공급자 네트워크 생성을 참조하십시오.



참고

Neutron 트렁크 서비스 플러그인이 활성화된 경우 기본적으로 트렁크 포트가 생성됩니다. 자세한 내용은 Neutron 트렁크 포트를 참조하십시오.

15.7.10. 환경에 대한 액세스 활성화

배포 시 모든 OpenShift Container Platform 시스템은 RHOSP(Red Hat OpenStack Platform) 테넌트 네트워크에서 생성됩니다. 따라서 대부분의 RHOSP 배포에서 직접 액세스할 수 없습니다.

설치시 부동 IP 주소 (FIP)를 사용하여 OpenShift Container Platform API 및 애플리케이션의 액세스를 설정할 수 있습니다. FIP를 구성하지 않고 설치를 완료 할 수도 있지만 설치 프로그램은 외부에서 API 또는 애플리케이션에 액세스하는 방법을 설정하지 않습니다.

15.7.10.1. 부동 IP 주소로 액세스 활성화

OpenShift Container Platform API, 클러스터 애플리케이션 및 부트스트랩 프로세스에 대한 외부 액세스 용으로 유동 **IP (FIP)** 주소를 생성합니다.

절차

1. **RHOSP(Red Hat OpenStack Platform) CLI**를 사용하여 **API FIP**를 생성합니다.

```
$ openstack floating ip create --description "API <cluster_name>.<base_domain>"
<external_network>
```

2. **RHOSP(Red Hat OpenStack Platform) CLI**를 사용하여 앱 또는 **Ingress, FIP**를 생성합니다.

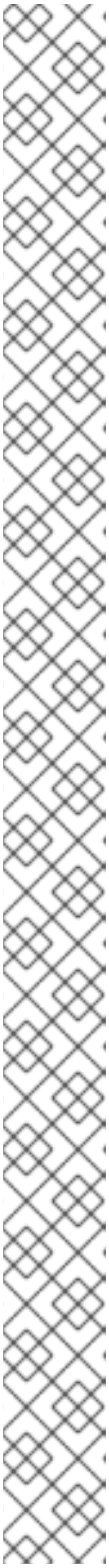
```
$ openstack floating ip create --description "Ingress <cluster_name>.<base_domain>"
<external_network>
```

3. **RHOSP(Red Hat OpenStack Platform) CLI**를 사용하여 부트스트랩 **FIP**를 생성합니다.

```
$ openstack floating ip create --description "bootstrap machine" <external_network>
```

4. **API** 및 **Ingress FIP**의 **DNS** 서버에 이러한 패턴에 따라 레코드를 추가합니다.

```
api.<cluster_name>.<base_domain>.  IN  A  <API_FIP>
*.apps.<cluster_name>.<base_domain>. IN  A <apps_FIP>
```



참고

DNS 서버를 제어하지 않으면 /etc/hosts 파일에 다음과 같은 클러스터 도메인 이름을 추가하여 클러스터에 액세스할 수 있습니다.

- <api_floating_ip> api.<cluster_name>.<base_domain>
- <application_floating_ip> grafana-openshift-monitoring.apps.<cluster_name>.<base_domain>
- <application_floating_ip> prometheus-k8s-openshift-monitoring.apps.<cluster_name>.<base_domain>
- <application_floating_ip> oauth-openshift.apps.<cluster_name>.<base_domain>
- <application_floating_ip> console-openshift-console.apps.<cluster_name>.<base_domain>
- application_floating_ip integrated-oauth-server-openshift-authentication.apps.<cluster_name>.<base_domain>

/etc/hosts 파일의 클러스터 도메인 이름은 웹 콘솔과 클러스터의 모니터링 인터페이스에 로컬로 액세스할 수 있는 권한을 부여합니다. kubectl 또는 oc 에서도 사용할 수 있습니다. <application_floating_ip>를 가리키는 추가 항목을 사용하여 사용자 애플리케이션에 액세스할 수 있습니다. 이 작업을 수행하면 사용자만 API 및 애플리케이션에 액세스할 수 있어 프로덕션 배포에는 적합하지 않지만 개발 및 테스트용 설치는 가능합니다.

5. FIP를 다음 변수의 값으로 inventory.yaml 파일에 추가합니다.

- os_api_fip
- os_bootstrap_fip

- os_ingress_fip

이러한 값을 사용하는 경우 **inventory.yaml** 파일의 **os_external_network** 변수 값으로 외부 네트워크를 입력해야 합니다.

작은 정보

부동 **IP** 주소를 할당하고 방화벽 구성을 업데이트하여 클러스터 외부에서 **OpenShift Container Platform** 리소스를 사용할 수 있습니다.

15.7.10.2. 유동 IP 주소없이 설치 완료

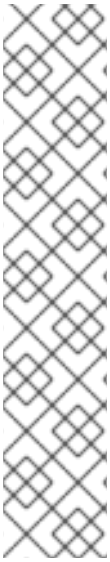
유동 **IP** 주소를 지정하지 않고도 **Red Hat OpenStack Platform (RHOSP)**에 **OpenShift Container Platform**을 설치할 수 있습니다.

inventory.yaml 파일에서 다음 변수를 정의하지 마십시오.

- os_api_fip
- os_bootstrap_fip
- os_ingress_fip

외부 네트워크를 제공할 수 없는 경우 **os_external_network**를 비워 둘 수도 있습니다. **os_external_network**의 값을 지정하지 않으면 라우터가 생성되지 않으며 추가 작업없이 설치 프로그램이 **Glance**에서 이미지를 검색하지 못합니다. 나중에 설치 프로세스에서 네트워크 리소스를 만들 때 외부 연결을 직접 구성해야 합니다.

유동 **IP** 주소 또는 이름 확인 부족으로 인해 클러스터 **API**에 연결할 수 없는 시스템에서 **wait-for** 명령으로 설치 프로그램을 실행하면 설치에 실패합니다. 이러한 경우 설치 실패를 방지하기 위해 프록시 네트워크를 사용하거나 시스템과 동일한 네트워크에있는 시스템에서 설치 프로그램을 실행할 수 있습니다.



참고

API 및 Ingress 포트의 DNS 레코드를 생성하여 이름 확인을 활성화할 수 있습니다. 예를 들면 다음과 같습니다.

```
api.<cluster_name>.<base_domain>.  IN  A  <api_port_IP>
*.apps.<cluster_name>.<base_domain>. IN  A <ingress_port_IP>
```

DNS 서버를 제어하지 않으면 /etc/hosts 파일에 레코드를 추가할 수 있습니다. 이 작업을 수행하면 사용자만 API에 액세스할 수 있어 프로덕션 배포에는 적합하지 않지만 개발 및 테스트용 설치는 가능합니다.

15.7.11. 설치 프로그램에 대한 매개변수 정의

OpenShift Container Platform 설치 프로그램은 clouds.yaml 파일을 사용합니다. 이 파일은 프로젝트 이름, 로그인 정보, 인증 서비스 URL 등 RHOSP(Red Hat OpenStack Platform) 구성 매개변수를 설명합니다.

프로세스

1. clouds.yaml 파일을 만듭니다.

- RHOSP 배포에 Horizon 웹 UI가 포함되어 있으면 그 안에 clouds.yaml 파일을 만듭니다.

중요

auth 필드에 암호를 추가해야 합니다. clouds.yaml과 별도의 파일에 비밀을 저장할 수도 있습니다.

- RHOSP 배포에 Horizon 웹 UI가 포함되어 있지 않거나 Horizon을 사용하지 않으려면 파일을 직접 만듭니다. clouds.yaml에 대한 자세한 내용은 RHOSP 문서의 구성 파일을 참조하십시오.

```
clouds:
  shiftstack:
    auth:
      auth_url: http://10.10.14.42:5000/v3
      project_name: shiftstack
      username: shiftstack_user
```

```
      password: XXX
      user_domain_name: Default
      project_domain_name: Default
    dev-env:
      region_name: RegionOne
      auth:
        username: 'devuser'
        password: XXX
        project_name: 'devonly'
        auth_url: 'https://10.10.14.22:5001/v2.0'
```

2. RHOSP 설치에서 끝점 인증을 위해 자체 서명된 CA(인증 기관) 인증서를 사용하는 경우:

   a. 인증 기관 파일을 시스템에 복사합니다.

   b. cacerts 키를 clouds.yaml 파일에 추가합니다. 값은 CA 인증서에 대한 루트가 아닌 액세스 가능한 절대 경로여야 합니다.

```
clouds:
  shiftstack:
    ...
    cacert: "/etc/pki/ca-trust/source/anchors/ca.crt.pem"
```

작은 정보

사용자 지정 CA 인증서로 설치 관리자를 실행한 후 cloud-provider-config 키맵에서 ca-cert.pem 키의 값을 편집하여 인증서를 업데이트할 수 있습니다. 명령줄에서 다음을 실행합니다.

```
$ oc edit configmap -n openshift-config cloud-provider-config
```

3. clouds.yaml 파일을 다음 위치 중 하나에 배치합니다.

   a. OS_CLIENT_CONFIG_FILE 환경 변수의 값

   b. 현재 디렉터리

   c. Unix 전용 사용자 구성 디렉터리(예: ~/.config/openstack/clouds.yaml)

d.

Unix 전용 사이트 구성 디렉터리(예: /etc/openstack/clouds.yaml)

설치 프로그램은 clouds.yaml을 이 순서대로 검색합니다.

15.7.12. 설치 구성 파일 만들기

RHOSP(Red Hat OpenStack Platform)에 설치하는 OpenShift Container Platform 클러스터를 사용자 지정할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.
- 서브스크립션 수준에서 서비스 주체 권한을 획득합니다.

프로세스

1. install-config.yaml 파일을 생성합니다.

a. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 <installation_directory>는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.

중요

비어 있는 디렉터리를 지정합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

b. 화면에 나타나는 지시에 따라 클라우드에 대한 구성 세부 사항을 입력합니다.

i. 선택사항: 클러스터 시스템에 액세스하는 데 사용할 SSH 키를 선택합니다.

참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 ssh-agent 프로세스가 사용하는 SSH 키를 지정합니다.

ii. 대상 플랫폼으로 openstack을 선택합니다.

iii. 클러스터 설치에 사용할 RHOSP(Red Hat OpenStack Platform) 외부 네트워크 이름을 지정합니다.

iv. OpenShift API에 대한 외부 액세스에 사용할 부동 IP 주소를 지정합니다.

v. 컨트롤 플레인 노드에 사용할 최소 16GB의 RAM과 컴퓨팅 노드에 8GB RAM이 있는 RHOSP 플레이버를 지정합니다.

vi. 클러스터를 배포할 기본 도메인을 선택합니다. 모든 DNS 레코드는 이 기본 도메인의 하위 도메인이 되므로 클러스터 이름을 포함합니다.

vii. 클러스터 이름을 입력합니다. 이름은 14자 이하여야 합니다.

viii. Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 붙여넣습니다.

2. install-config.yaml 파일을 수정합니다. 사용 가능한 매개변수에 대한 자세한 정보는 “설치 구성 매개변수” 섹션에서 확인할 수 있습니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 install-config.yaml 파일을 백업합니다.

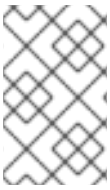
중요

install-config.yaml 파일은 설치 과정에서 사용됩니다. 이 파일을 재사용하려면 지금 백업해야 합니다.

이제 사용자가 지정한 디렉터리에 **install-config.yaml** 파일이 있습니다.

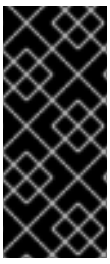
15.7.13. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 매개변수 값을 제공하여 클러스터를 호스팅할 클라우드 플랫폼에서 사용자 계정을 설명하고 선택사항으로 클러스터의 플랫폼을 사용자 지정합니다. **install-config.yaml** 설치 구성 파일을 생성할 때 명령줄을 통해 필요한 매개변수 값을 제공합니다. 클러스터를 사용자 지정하면 **install-config.yaml** 파일을 수정하여 플랫폼에 대한 세부 정보를 제공할 수 있습니다.



참고

설치한 후에는 **install-config.yaml** 파일에서 이러한 매개변수를 수정할 수 없습니다.



중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

15.7.13.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 15.30. 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열

매개변수	설명	값
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	소문자, 하이픈(**-**), 마침표(**.**)로 구성되는 문자열(예: **dev**)입니다. 문자열은 14자 이하여야 합니다.
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

15.7.13.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스

터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 15.31. 네트워크 매개변수

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체  **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.

networking.clusterNetwork	Pod의 IP 주소 블록입니다.
	기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다.
	여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.
		개체의 배열입니다. 예를 들면 다음과 같습니다.
		`networking:`
		`  clusterNetwork:`
		`  - cidr: 10.128.0.0/14`
		`  hostPrefix: 23`

networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된**cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.

매개변수	설명	값
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16** **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

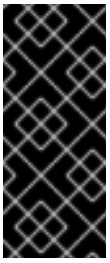
15.7.13.3. 선택적 구성 매개변수

선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 15.32. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.

매개변수	설명	값
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은**3**입니다.
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열

매개변수	설명	값
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 기본값은 **External**입니다. 이 필드를 **Internal** 로 설정하는 것은 클라우드 이외의 플랫폼에서는 지원되지 않습니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

15.7.13.4. 추가 RHOSP(Red Hat OpenStack Platform) 구성 매개변수

추가 RHOSP 구성 매개변수는 다음 표에 설명되어 있습니다.

표 15.33. 추가 RHOSP 매개변수

매개변수	설명	값
compute.platform.openstack.rootVolume.size	컴퓨팅 시스템의 경우 루트 볼륨의 크기(GB)입니다. 이 값을 설정하지 않으면 시스템은 임시 스토리지를 사용합니다.	정수 (예: **30**)
compute.platform.openstack.rootVolume.type	컴퓨팅 시스템의 경우 루트 볼륨의 유형입니다.	문자열 (예: **performance**)
controlPlane.platform.openstack.rootVolume.size	컨트롤 플레인 시스템의 경우 루트 볼륨의 크기(GB)입니다. 이 값을 설정하지 않으면 시스템은 임시 스토리지를 사용합니다.	정수 (예: **30**)
controlPlane.platform.openstack.rootVolume.type	컨트롤 플레인 시스템의 경우 루트 볼륨의 유형입니다.	문자열 (예: **performance**)
platform.openstack.cloud	**clouds.yaml** 파일의 클라우드 목록에서 사용할 RHOSP 클라우드의 이름입니다.	문자열 (예: **MyCloud**)

매개변수	설명	값
platform.openstack.externalNetwork	설치에 사용할 RHOSP 외부 네트워크 이름입니다.	문자열 (예: **external)**
platform.openstack.computeFlavor	컨트롤 플레인 및 컴퓨팅 시스템에 사용할 RHOSP 버전입니다. 이 속성은 더 이상 사용되지 않습니다. 모든 머신 풀에 플레이버를 기본값으로 사용하려면 **platform.openstack.defaultMachinePlatform** 속성의 **type** 키 값으로 추가합니다. 각 머신 풀의 플레이버 값을 개별적으로 설정할 수도 있습니다.	문자열 (예: **m1.xlarge)**

15.7.13.5. 선택적 RHOSP 구성 매개변수

선택적 RHOSP 구성 매개변수는 다음 표에 설명되어 있습니다.

표 15.34. 선택적 RHOSP 매개변수

매개변수	설명	값
compute.platform.openstack.additionalNetworkIDs	컴퓨팅 시스템과 관련된 추가 네트워크입니다. 추가 네트워크에는 허용되는 주소 쌍이 생성되지 않습니다.	하나 이상의 UUID 목록을 문자열로 나타냅니다. 예: **fa806b2f-ac49-4bce-b9db-124bc64209bf**
compute.platform.openstack.additionalSecurityGroupIDs	컴퓨팅 시스템과 관련된 추가 보안 그룹입니다.	하나 이상의 UUID 목록을 문자열로 나타냅니다. 예: **7ee219f3-d2e9-48a1-96c2-e7429f1b0da7**

매개변수	설명	값
compute.platform.openstack.zones	머신을 설치할 RHOSP Compute (Nova) 가용성 영역 (AZ). 이 매개 변수가 설정되지 않은 경우 설치 프로그램은 RHOSP 관리자가 구성한 Nova의 기본 설정을 사용합니다. Kuryr를 사용하는 클러스터에서 RHOSP Octavia는 가용성 영역을 지원하지 않습니다. 로드 밸런서 및 Amphora 공급자 드라이버를 사용하는 경우 Amphora VM에 의존하는 OpenShift Container Platform 서비스는이 속성의 값에 따라 생성되지 않습니다.	문자열 목록입니다. 예: **["zone-1", "zone-2"]**.
compute.platform.openstack.rootVolume.zones	컴퓨팅 시스템의 경우 루트 볼륨을 설치할 가용성 영역입니다. 이 매개변수의 값을 설정하지 않으면 설치 프로그램에서 기본 가용성 영역을 선택합니다.	문자열 목록(예: **["zone-1", "zone-2"]**).
controlPlane.platform.openstack.additionalNetworkIDs	컨트롤 플레인 시스템과 관련된 추가 네트워크입니다. 추가 네트워크에는 허용되는 주소 쌍이 생성되지 않습니다.	하나 이상의 UUID 목록을 문자열로 나타냅니다. 예: **fa806b2f-ac49-4bce-b9db-124bc64209bf**
controlPlane.platform.openstack.additionalSecurityGroupIDs	컨트롤 플레인 시스템과 관련된 추가 보안 그룹입니다.	하나 이상의 UUID 목록을 문자열로 나타냅니다. 예: **7ee219f3-d2e9-48a1-96c2-e7429f1b0da7**
controlPlane.platform.openstack.zones	머신을 설치할 RHOSP Compute (Nova) 가용성 영역 (AZ). 이 매개 변수가 설정되지 않은 경우 설치 프로그램은 RHOSP 관리자가 구성한 Nova의 기본 설정을 사용합니다. Kuryr를 사용하는 클러스터에서 RHOSP Octavia는 가용성 영역을 지원하지 않습니다. 로드 밸런서 및 Amphora 공급자 드라이버를 사용하는 경우 Amphora VM에 의존하는 OpenShift Container Platform 서비스는이 속성의 값에 따라 생성되지 않습니다.	문자열 목록입니다. 예: **["zone-1", "zone-2"]**.

매개변수	설명	값
controlPlane.platform.openstack.rootVolume.zones	컨트롤 플레인 시스템의 경우 루트 볼륨을 설치할 가용성 영역입니다. 이 값을 설정하지 않으면 설치 프로그램에서 기본 가용성 영역을 선택합니다.	문자열 목록(예: **["zone-1", "zone-2"]**).
platform.openstack.clusterOSImage	설치 프로그램이 RHCOS 이미지를 다운로드하는 위치입니다. 네트워크가 제한된 환경에서 설치를 수행하려면이 매개 변수를 설정해야 합니다.	HTTP 또는 HTTPS URL (선택 옵션으로 SHA-256 형식의 체크섬을 사용) 예: **http://mirror.example.com/images/rhcos-43.81.201912131630.0-openstack.x86_64.qcow2.gz?sha256=ffebbd68e8a1f2a245ca19522c16c86f67f9ac8e4e0c1f0a812b068b16f7265d**. 이 값은 기존의 Glance 이미지의 이름이 될 수도 있습니다 (예: **my-rhcos**).
platform.openstack.clusterOSImageProperties	Glance의 설치 프로그램에서 업로드된 ClusterOSImage에 추가할 속성입니다. **platform.openstack.clusterOSImage**가 기존 Glance 이미지로 설정된 경우 이 속성은 무시됩니다. 이 속성을 사용하여 노드당 26 PV의 RHOSP의 기본 PV(영구 볼륨) 제한을 초과할 수 있습니다. 제한을 초과하려면 **hw_scsi_model** 속성값을 **virtio-scsi**로, **hw_disk_bus** 값을 **scsi**로 설정합니다. 이 속성을 사용하면 **yes** 값이 있는 **hw_qemu_guest_agent** 속성을 포함하여 QEMU 게스트 에이전트를 활성화할 수도 있습니다.	키-값 문자열 쌍 목록입니다. 예를 들면 **["hw_scsi_model": "virtio-scsi", "hw_disk_bus": "scsi"]**가 있습니다.

platform.openstack.defaultMachinePlatform	기본 시스템 풀 플랫폼 구성입니다.	<pre>{
		  "type": "ml.large",
		  "rootVolume": {
		  "size": 30,
		  "type": "performance"
		  }
		}</pre>

매개변수	설명	값
platform.openstack.ingressFloatingIP	Ingress 포트와 연결할 기존 부동 IP 주소입니다. 이 속성을 사용하려면 **platform.openstack.externalNetwork** 속성도 정의해야 합니다.	IP 주소 (예: **128.0.0.1**)
platform.openstack.apiFloatingIP	로드 밸런서 API와 연결할 기존 부동 IP 주소입니다. 이 속성을 사용하려면 **platform.openstack.externalNetwork** 속성도 정의해야 합니다.	IP 주소 (예: **128.0.0.1**)
platform.openstack.externalDNS	클러스터 인스턴스가 DNS 확인에 사용하는 외부 DNS 서버의 IP 주소입니다.	IP 주소 목록을 문자열로 나타냅니다. 예: **["8.8.8.8", "192.168.1.12"]**.
platform.openstack.machinesSubnet	클러스터 노드가 사용하는 RHOSP 서브넷의 UUID입니다. 이 서브넷에 노드와 가상 IP(VIP) 포트가 생성됩니다. **networking.machineNetwork**의 첫 번째 항목은 **machinesSubnet** 값과 일치해야 합니다. 사용자 지정 서브넷에 배포하는 경우 OpenShift Container Platform 설치 관리자에 외부 DNS 서버를 지정할 수 없습니다. 대신 RHOSP의 서브넷에 DNS를 추가합니다.	문자열의 UUID입니다. 예: **fa806b2f-ac49-4bce-b9db-124bc64209bf**

15.7.13.6. RHOSP용 샘플 사용자 지정 install-config.yaml 파일

이 샘플 install-config.yaml은 가능한 모든 RHOSP(Red Hat OpenStack Platform) 사용자 지정 옵션을 보여줍니다.

중요

이 샘플 파일은 참조용으로만 제공됩니다. install-config.yaml 파일은 설치 프로그램을 사용하여 받아야 합니다.

```
apiVersion: v1
```

```
baseDomain: example.com
controlPlane:
  name: master
  platform: {}
  replicas: 3
compute:
- name: worker
  platform:
    openstack:
      type: ml.large
  replicas: 3
metadata:
  name: example
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  serviceNetwork:
  - 172.30.0.0/16
  networkType: OpenShiftSDN
platform:
  openstack:
    cloud: mycloud
    externalNetwork: external
    computeFlavor: m1.xlarge
    apiFloatingIP: 128.0.0.1
fips: false
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```

15.7.13.7. RHOSP 배포의 사용자 지정 서브넷

선택한 **RHOSP(Red Hat OpenStack Platform)** 서브넷에 클러스터를 배포할 수도 있습니다. 서브넷의 **GUID**는 **install-config.yaml** 파일에서 **platform.openstack.machinesSubnet** 값으로 전달됩니다.

이 서브넷은 클러스터의 기본 서브넷으로 사용됩니다. 기본적으로 노드와 포트가 이 서브넷에서 생성됩니다. **platform.openstack.machinesSubnet** 속성의 값을 서브넷의 **UUID**로 설정하여 다른 **RHOSP** 서브넷에서 노드 및 포트를 생성할 수 있습니다.

사용자 지정 서브넷으로 **OpenShift Container Platform** 설치 관리자를 실행하기 전에 구성이 다음 요구 사항을 충족하는지 확인하십시오.

- **platform.openstack.machinesSubnet**에서 사용하는 서브넷에 **DHCP**가 활성화되어 있습니다.
-

platform.openstack.machinesSubnet의 CIDR은 networking.machineNetwork의 CIDR과 일치합니다.

- 설치 프로그램 사용자에게 고정 IP 주소가 있는 포트를 포함하여 이 네트워크에 포트를 생성할 수 있는 권한이 있습니다.

사용자 지정 서브넷을 사용하는 클러스터에는 다음과 같은 제한 사항이 있습니다.

- 유동 IP 주소를 사용하는 클러스터를 설치하려는 경우 platform.openstack.machinesSubnet 서브넷이 externalNetwork 네트워크에 연결된 라우터에 연결되어 있어야 합니다.

- platform.openstack.machinesSubnet 값이 install-config.yaml 파일에 설정된 경우 설치 프로그램에서 RHOSP 머신의 프라이빗 네트워크 또는 서브넷을 생성하지 않습니다.

- 사용자 지정 서브넷과 동시에 platform.openstack.externalDNS 속성을 사용할 수 없습니다. 사용자 지정 서브넷을 사용하는 클러스터에 DNS를 추가하려면 RHOSP 네트워크에서 DNS를 구성합니다.

참고

기본적으로 API VIP는 x.x.x.5를 가져오고 Ingress VIP는 네트워크의 CIDR 블록에서 x.x.x.7을 가져옵니다. 이러한 기본값을 재정의하려면 DHCP 할당 풀 외부에 있는platform.openstack.apiVIP 및 platform.openstack.ingressVIP의 값을 설정합니다.

15.7.13.8. 시스템의 사용자 지정 서브넷 설정

설치 프로그램이 기본적으로 사용하는 IP 범위가 OpenShift Container Platform을 설치할 때 생성하는 Neutron 서브넷과 일치하지 않을 수 있습니다. 필요한 경우 설치 구성 파일을 편집하여 새 시스템의 CIDR 값을 업데이트하십시오.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램에서 생성된 install-config.yaml 파일이 있습니다.

프로세스

1. 명령줄에서 **install-config.yaml**이 들어 있는 디렉터리를 찾습니다.

2. 해당 디렉터리에서 스크립트를 실행하여 **install-config.yaml** 파일을 편집하거나 파일을 수동으로 업데이트합니다.

   - 스크립트를 사용하여 값을 설정하려면 다음을 실행합니다.

```
$ python -c '
import yaml;
path = "install-config.yaml";
data = yaml.safe_load(open(path));
data["networking"]["machineNetwork"] = [{"cidr": "192.168.0.0/18"}]; 1
open(path, "w").write(yaml.dump(data, default_flow_style=False))'
```

   1 원하는 Neutron 서브넷과 일치하는 값을 삽입합니다(예: 192.0.2.0/24).

   - 값을 수동으로 설정하려면 파일을 열고 **networking.machineCIDR** 값을 원하는 Neutron 서브넷과 일치하는 값으로 설정합니다.

15.7.13.9. 컴퓨팅 시스템 풀 비우기

사용자 인프라를 사용하는 설치를 계속 진행하려면 설치 구성 파일의 컴퓨팅 시스템 수를 0으로 설정합니다. 이러한 시스템은 나중에 수동으로 생성합니다.

사전 요구 사항

- **OpenShift Container Platform** 설치 프로그램에서 생성된 **install-config.yaml** 파일이 있습니다.

프로세스

1. 명령줄에서 **install-config.yaml**이 들어 있는 디렉터리를 찾습니다.

2. 해당 디렉터리에서 스크립트를 실행하여 **install-config.yaml** 파일을 편집하거나 파일을 수동으로 업데이트합니다.

- 스크립트를 사용하여 값을 설정하려면 다음을 실행합니다.

```
$ python -c '
import yaml;
path = "install-config.yaml";
data = yaml.safe_load(open(path));
data["compute"][0]["replicas"] = 0;
open(path, "w").write(yaml.dump(data, default_flow_style=False))'
```

- 값을 수동으로 설정하려면 파일을 열고 **compute.<first entry>.replicas** 값을 **0**으로 설정합니다.

15.7.14. Kubernetes 매니페스트 및 Ignition 설정 파일 생성

일부 클러스터 정의 파일을 수정하고 클러스터 시스템을 수동으로 시작해야 하므로 클러스터가 시스템을 구성하는 데 필요한 **Kubernetes** 매니페스트 및 **Ignition** 구성 파일을 사용자가 생성해야 합니다.

설치 구성 파일은 **Kubernetes** 매니페스트로 변환됩니다. 매니페스트는 나중에 클러스터 머신을 구성하는 데 사용되는 **Ignition** 구성 파일로 래핑됩니다.

중요

- **OpenShift Container Platform** 설치 프로그램에서 생성하는 **Ignition** 구성 파일에 **24**시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 **24**시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 **kubelet** 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청**(CSR)**을 수동으로 승인해야 합니다. 자세한 내용은 ***Recovering from expired control plane certificates*** 문서를 참조하십시오.

- 클러스터를 설치한 후 **24**시간에서 **22**시간까지의 인증서가 교체되기 때문에 생성된 후 **12**시간 이내에 **Ignition** 구성 파일을 사용하는 것이 좋습니다. **12**시간 이내에 **Ignition** 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

사전 요구 사항

- **OpenShift Container Platform** 설치 프로그램을 가져오셨습니다.

- install-config.yaml 설치 구성 파일을 생성하셨습니다.

프로세스

1. OpenShift Container Platform 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터에 대한 Kubernetes 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 <installation_directory>는 사용자가 만든 install-config.yaml 파일이 포함된 설치 디렉터리를 지정합니다.

2. 컨트롤 플레인 시스템 및 컴퓨팅 머신 세트를 정의하는 Kubernetes 매니페스트 파일을 제거합니다.

```
$ rm -f openshift/99_openshift-cluster-api_master-machines-*.yaml
openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

이러한 리소스는 사용자가 직접 생성하고 관리하기 때문에 초기화할 필요가 없습니다.

- 시스템 API로 머신 세트 파일을 보존하여 컴퓨팅 시스템을 생성할 수 있지만 사용자 환경과 일치하도록 해당 참조를 업데이트해야 합니다.

3. <installation_directory>/manifests/cluster-scheduler-02-config.yml Kubernetes 매니페스트 파일의 mastersSchedulable 매개변수가 false로 설정되어 있는지 확인합니다. 이 설정으로 인해 컨트롤 플레인 머신에서 포드가 예약되지 않습니다.

   a. <installation_directory>/manifests/cluster-scheduler-02-config.yml 파일을 엽니다.

   b. mastersSchedulable 매개변수를 찾아서 값을 False로 설정되어 있는지 확인합니다.

   c. 파일을 저장하고 종료합니다.

4. Ignition 구성 파일을 생성하려면 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행합니다.

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 <installation_directory>는 동일한 설치 디렉터리를 지정합니다.

설치 디렉터리의 부트스트랩, 컨트롤 플레인 및 컴퓨팅 노드에 대한 Ignition 구성 파일이 생성됩니다. kubeadmin-password 및 kubeconfig 파일은 ./<installation_directory>/auth 디렉터리에 생성됩니다.

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

5. 메타데이터 파일의 infraID 키를 환경 변수로 내보냅니다.

```
$ export INFRA_ID=$(jq -r .infraID metadata.json)
```

작은 정보

metadata.json에서 infraID 키를 추출하여 사용자가 생성하는 모든 RHOSP 리소스의 접두사로 사용합니다. 이렇게 하면 동일한 프로젝트에서 여러 배포를 수행할 때 이름 충돌을 방지할 수 있습니다.

15.7.15. 부트스트랩 Ignition 파일 준비

OpenShift Container Platform 설치 프로세스는 부트스트랩 Ignition 구성 파일에서 생성되는 부트스트랩 시스템에 의존합니다.

파일을 편집하고 업로드합니다. 그런 다음 RHOSP(Red Hat OpenStack Platform)에서 기본 파일을 다운로드하는 데 사용하는 보조 부트스트랩 Ignition 구성 파일을 만듭니다.

사전 요구 사항

- 설치 관리자에서 생성되는 부트스트랩 Ignition 파일, bootstrap.ign이 있습니다.
- 설치 관리자 메타데이터 파일의 인프라 ID는 환경 변수($ INFRA_ID)로 설정됩니다.
  - 변수가 설정되지 않은 경우 Kubernetes 매니페스트 및 Ignition 구성 파일 생성을 참조하십시오.
- 부트스트랩 Ignition 파일을 저장할 수 있는 HTTP(S) 액세스 가능 방법이 있습니다.
  - 문서화된 프로시저에는 RHOSP 이미지 서비스(Glance)를 사용하지만 RHOSP 스토리지 서비스(Swift), Amazon S3, 내부 HTTP 서버 또는 애드혹 Nova 서버를 사용할 수도 있습니다.

프로세스

1. 다음 Python 스크립트를 실행합니다. 이 스크립트는 부트스트랩 Ignition 파일을 수정하여 호스트 이름 및 사용 가능한 경우 실행 시 CA 인증서 파일을 설정합니다.

```
import base64
import json
import os

with open('bootstrap.ign', 'r') as f:
    ignition = json.load(f)

files = ignition['storage'].get('files', [])

infra_id = os.environ.get('INFRA_ID', 'openshift').encode()
hostname_b64 = base64.standard_b64encode(infra_id + b'-
bootstrap\n').decode().strip()
files.append(
{
    'path': '/etc/hostname',
    'mode': 420,
    'contents': {
        'source': 'data:text/plain;charset=utf-8;base64,' + hostname_b64
    }
})

ca_cert_path = os.environ.get('OS_CACERT', '')
if ca_cert_path:
    with open(ca_cert_path, 'r') as f:
        ca_cert = f.read().encode()
        ca_cert_b64 = base64.standard_b64encode(ca_cert).decode().strip()
```

```
    files.append(
    {
        'path': '/opt/openshift/tls/cloud-ca-cert.pem',
        'mode': 420,
        'contents': {
            'source': 'data:text/plain;charset=utf-8;base64,' + ca_cert_b64
        }
    })

ignition['storage']['files'] = files;

with open('bootstrap.ign', 'w') as f:
    json.dump(ignition, f)
```

2. RHOSP CLI를 사용하여 부트스트랩 Ignition 파일을 사용하는 이미지를 생성합니다.

```
$ openstack image create --disk-format=raw --container-format=bare --file
bootstrap.ign <image_name>
```

3. 이미지의 세부 사항을 가져옵니다.

```
$ openstack image show <image_name>
```

file 값을 기록해 둡니다. 해당 패턴은 v2/images/<image_ID>/file입니다.

참고

생성된 이미지가 활성화되어 있는지 확인합니다.

4. 이미지 서비스의 공용 주소를 검색합니다.

```
$ openstack catalog show image
```

5. 공용 주소를 이미지 file 값과 결합하고 그 결과를 저장 위치로 저장합니다. 위치 패턴은 <image_service_public_URL>/v2/images/<image_ID>/file입니다.

6. 인증 토큰을 생성하고 토큰 ID를 저장합니다.

```
$ openstack token issue -c id -f value
```

7. $INFRA_ID-bootstrap-ignition.json 파일에 다음 내용을 삽입하고 사용자 값과 일치하도록 자리 표시자를 편집합니다.

```
{
 "ignition": {
  "config": {
   "merge": [{
    "source": "<storage_url>", 1
    "httpHeaders": [{
     "name": "X-Auth-Token", 2
     "value": "<token_ID>" 3
    }]
   }]
  },
  "security": {
   "tls": {
    "certificateAuthorities": [{
     "source": "data:text/plain;charset=utf-8;base64,<base64_encoded_certificate>"
4
    }]
   }
  },
  "version": "3.2.0"
 }
}
```

1 ignition.config.merge.source 값을 부트스트랩 Ignition 파일 스토리지 URL로 바꿉니다.

2 httpHeaders의 name을 "X-Auth-Token"으로 설정합니다.

3 httpHeaders의 value를 토큰의 ID로 설정합니다.

4 부트스트랩 Ignition 파일 서버가 자체 서명 인증서를 사용하는 경우 base64 인코딩 인증서를 포함합니다.

8. 보조 Ignition 구성 파일을 저장합니다.

부트스트랩 Ignition 데이터는 설치 중에 RHOSP로 전달됩니다.

주의

부트스트랩 Ignition 파일에는 clouds.yaml 자격 증명과 같은 민감한 정보가 들어 있습니다. 파일을 안전한 곳에 저장하고 설치 프로세스를 완료한 후에는 삭제해야 합니다.

15.7.16. RHOSP에서 컨트롤 플레인 Ignition 구성 파일 생성

자체 인프라의 RHOSP(Red Hat OpenStack Platform)에 OpenShift Container Platform을 설치하려면 컨트롤 플레인 Ignition 구성 파일이 필요합니다. 구성 파일은 여러 개 만들어야 합니다.

참고

부트스트랩 Ignition 구성과 마찬가지로 각 컨트롤 플레인 시스템의 호스트 이름을 명시적으로 정의해야 합니다.

사전 요구 사항

- 설치 프로그램 메타데이터 파일의 인프라 ID는 환경 변수($ INFRA_ID)로 설정됩니다.
  - 변수가 설정되지 않은 경우 “Kubernetes 매니페스트 및 Ignition 구성 파일 생성”을 참조하십시오.

프로세스

- 명령줄에서 다음 Python 스크립트를 실행합니다.

```
$ for index in $(seq 0 2); do
    MASTER_HOSTNAME="$INFRA_ID-master-$index\n"
    python -c "import base64, json, sys;
ignition = json.load(sys.stdin);
storage = ignition.get('storage', {});
files = storage.get('files', []);
files.append({'path': '/etc/hostname', 'mode': 420, 'contents': {'source':
```

```
'data:text/plain;charset=utf-8;base64,' +
base64.standard_b64encode(b'$MASTER_HOSTNAME').decode().strip(), 'verification':
{}}, 'filesystem': 'root'});
storage['files'] = files;
ignition['storage'] = storage
json.dump(ignition, sys.stdout)" <master.ign >"$INFRA_ID-master-$index-
ignition.json"
done
```

이제 <INFRA_ID>-master-0-ignition.json, <INFRA_ID>-master-1-ignition.json, <INFRA_ID>-master-2-ignition.json의 세 가지 컨트롤 플레인 Ignition 파일이 있습니다.

15.7.17. RHOSP에서 네트워크 리소스 생성

자체 인프라의 RHOSP(Red Hat OpenStack Platform)에 OpenShift Container Platform을 설치하는 데 필요한 네트워크 리소스를 생성합니다. 시간을 절약하려면 보안 그룹, 네트워크, 서브넷, 라우터 및 포트를 생성하는 제공된 Ansible 플레이북을 실행합니다.

사전 요구 사항

- Python 3가 시스템에 설치되어 있습니다.
- "플레이북 종속 항목 다운로드"에서 모듈을 다운로드했습니다.
- "설치 플레이북 다운로드"에서 플레이북을 다운로드했습니다.

프로세스

1. 선택사항: inventory.yaml 플레이북에 외부 네트워크값을 추가합니다.

inventory.yaml Ansible 플레이북의 외부 네트워크 값 예

```
...
    # The public network providing connectivity to the cluster. If not
    # provided, the cluster external connectivity must be provided in another
    # way.

    # Required for os_api_fip, os_ingress_fip, os_bootstrap_fip.
    os_external_network: 'external'
...
```

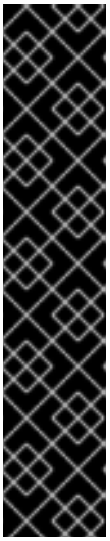
중요

inventory.yaml 파일에 **os_external_network** 값을 지정하지 않은 경우 **VM**이 **Glance** 및 외부 연결에 직접 액세스할 수 있는지 확인해야 합니다.

2. 선택사항: 외부 네트워크 및 부동 **IP(FIP)** 주소값을 **inventory.yaml** 플레이북에 추가합니다.

inventory.yaml Ansible 플레이북의 **FIP** 값 예

```
...
    # OpenShift API floating IP address. If this value is non-empty, the
    # corresponding floating IP will be attached to the Control Plane to
    # serve the OpenShift API.
    os_api_fip: '203.0.113.23'

    # OpenShift Ingress floating IP address. If this value is non-empty, the
    # corresponding floating IP will be attached to the worker nodes to serve
    # the applications.
    os_ingress_fip: '203.0.113.19'

    # If this value is non-empty, the corresponding floating IP will be
    # attached to the bootstrap machine. This is needed for collecting logs
    # in case of install failure.
    os_bootstrap_fip: '203.0.113.20'
```



중요

os_api_fip 및 **os_ingress_fip**에 대한 값을 정의하지 않으면 설치 후 네트워크 구성을 수행해야 합니다.

os_bootstrap_fip의 값을 정의하지 않으면 설치 프로그램이 실패한 설치에서 디버깅 정보를 다운로드할 수 없습니다.

자세한 정보는 "환경에 대한 액세스 활성화"를 참조하십시오.

3. 명령줄에서 **security-groups.yaml** 플레이북을 실행하여 보안 그룹을 생성합니다.

```
$ ansible-playbook -i inventory.yaml security-groups.yaml
```

4. 명령줄에서 **network.yaml** 플레이북을 실행하여 네트워크, 서브넷 및 라우터를 생성합니다.

```
$ ansible-playbook -i inventory.yaml network.yaml
```

5. 선택사항: Nova 서버가 사용하는 기본 해결 프로그램을 제어하려면 RHOSP CLI 명령을 실행합니다.

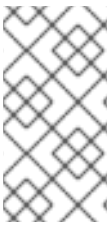
```
$ openstack subnet set --dns-nameserver <server_1> --dns-nameserver <server_2>
"$INFRA_ID-nodes"
```

선택적으로 생성된 **inventory.yaml** 파일을 사용하여 설치를 사용자 지정할 수 있습니다. 예를 들어 베어 메탈 머신을 사용하는 클러스터를 배포할 수 있습니다.

15.7.17.1. 베어 메탈 머신으로 클러스터 배포

클러스터가 베어 메탈 머신을 사용하도록 하려면 **inventory.yaml** 파일을 수정합니다. 클러스터는 베어 메탈에서 컨트롤 플레인 및 컴퓨팅 머신 모두를 실행하거나 컴퓨팅 머신만으로 실행할 수 있습니다.

Kuryr를 사용하는 클러스터에서 베어 메탈 컴퓨팅 머신이 지원되지 않습니다.



참고

install-config.yaml 파일에서 베어 메탈 작업자가 유동 IP 주소를 지원하는지 여부를 반영하는지 확인하십시오.

사전 요구 사항

- RHOSP Bare Metal 서비스(Ironic)가 활성화되어 RHOSP Compute API를 통해 액세스할 수 있습니다.
- 베어 메탈은 RHOSP 플레이버로 사용할 수 있습니다.

- RHOSP 네트워크는 VM 및 베어 메탈 서버 연결을 모두 지원합니다.
- 네트워크 구성이 공급자 네트워크를 사용하지 않습니다. 공급자 네트워크는 지원되지 않습니다.
- 기존 네트워크에 머신을 배포하려면 RHOSP 서브넷이 프로비저닝됩니다.
- 설치 관리자 프로비저닝 네트워크에 머신을 배포하려는 경우 RHOSP Bare Metal 서비스(Ironic)가 테넌트 네트워크에서 실행되는 PXE(Preboot eXecution Environment) 부팅 머신을 수신하고 상호 작용할 수 있습니다.
- OpenShift Container Platform 설치 프로세스의 일부로 inventory.yaml 파일을 생성하셨습니다.

프로세스

1. inventory.yaml 파일에서 머신의 플레이버를 편집합니다.
   a. 베어 메탈 컨트롤 플레인 머신을 사용하려면 os_flavor_master 값을 베어 메탈 플레이버로 변경합니다.
   b. os_flavor_worker의 값을 베어 메탈 플레이버로 변경합니다.

베어 메탈 inventory.yaml 파일 예

```
all:
  hosts:
    localhost:
      ansible_connection: local
      ansible_python_interpreter: "{{ansible_playbook_python}}"

      # User-provided values
      os_subnet_range: '10.0.0.0/16'
      os_flavor_master: 'my-bare-metal-flavor' 1
      os_flavor_worker: 'my-bare-metal-flavor' 2
```

```
    os_image_rhcos: 'rhcos'
    os_external_network: 'external'
...
```

1

베어 메탈 컨트롤 플레인 머신이 필요한 경우 이 값을 베어 메탈 플레이버로 변경합니다.

2

컴퓨팅 머신에 사용할 베어 메탈 플레이버로 이 값을 변경합니다.

업데이트된 **inventory.yaml** 파일을 사용하여 설치 프로세스를 완료합니다. 배포 중에 생성된 머신은 파일에 추가한 플레이버를 사용합니다.

참고

설치 프로그램은 베어 메탈 머신이 부팅될 때까지 대기하는 동안 시간이 초과될 수 있습니다.

설치 프로그램이 시간 초과되면 설치 프로그램의 **wait-for** 명령을 사용하여 배포를 다시 시작한 다음 완료합니다. 예를 들면 다음과 같습니다.

```
./openshift-install wait-for install-complete --log-level debug
```

15.7.18. RHOSP에서 부트스트랩 시스템 생성

부트스트랩 시스템을 생성하고 **RHOSP(Red Hat OpenStack Platform)**에서 실행하는 데 필요한 네트워크 액세스 권한을 부여합니다. **Red Hat**은 이 프로세스를 간소화하기 위해 실행하는 **Ansible** 플레이북을 제공합니다.

사전 요구 사항

- "플레이북 종속 항목 다운로드"에서 모듈을 다운로드했습니다.

- "설치 플레이북 다운로드"에서 플레이북을 다운로드했습니다.
- inventory.yaml, common.yaml 및 bootstrap.yaml Ansible 플레이북이 공통 디렉터리에 있습니다.
- 설치 프로그램에서 생성된 metadata.json 파일이 Ansible 플레이북과 동일한 디렉터리에 있습니다.

프로세스

1. 명령줄에서 작업 디렉터리를 플레이북의 위치로 변경합니다.
2. 명령줄에서 bootstrap.yaml 플레이북을 실행합니다.

```
$ ansible-playbook -i inventory.yaml bootstrap.yaml
```

3. 부트스트랩 서버가 활성화된 후 로그를 보고 Ignition 파일이 수신되었는지 확인합니다.

```
$ openstack console log show "$INFRA_ID-bootstrap"
```

15.7.19. RHOSP에 컨트롤 플레인 시스템 생성

사용자가 생성한 Ignition 구성 파일을 사용하여 세 개의 컨트롤 플레인 시스템을 생성합니다. Red Hat은 이 프로세스를 간소화하기 위해 실행하는 Ansible 플레이북을 제공합니다.

사전 요구 사항

- "플레이북 종속 항목 다운로드"에서 모듈을 다운로드했습니다.
- "설치 플레이북 다운로드"에서 플레이북을 다운로드했습니다.
- 설치 프로그램 메타데이터 파일의 인프라 ID는 환경 변수($ INFRA_ID)로 설정됩니다.
- inventory.yaml, common.yaml 및 control-plane.yaml Ansible 플레이북이 공통 디렉터리

에 있습니다.

- “컨트롤 플레인 Ignition 구성 파일 생성”에서 생성된 세 개의 Ignition 파일이 있습니다.

프로세스

1. 명령줄에서 작업 디렉터리를 플레이북의 위치로 변경합니다.
2. 컨트롤 플레인 Ignition 구성 파일이 작업 디렉터리에 없으면 파일을 디렉터리에 복사합니다.
3. 명령줄에서 control-plane.yaml 플레이북을 실행합니다.

```
$ ansible-playbook -i inventory.yaml control-plane.yaml
```

4. 다음 명령을 실행하여 부트스트랩 프로세스를 모니터링합니다.

```
$ openshift-install wait-for bootstrap-complete
```

컨트롤 플레인 시스템이 실행 중이고 클러스터에 연결되었음을 확인하는 메시지가 표시됩니다.

```
INFO API v1.22.1 up
INFO Waiting up to 30m0s for bootstrapping to complete...
...
INFO It is now safe to remove the bootstrap resources
```

15.7.20. CLI를 사용하여 클러스터에 로그인

클러스터 kubeconfig 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. kubeconfig 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.

- oc CLI를 설치했습니다.

프로세스

1. kubeadmin 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 oc 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

15.7.21. RHOSP에서 부트스트랩 리소스 삭제

더 이상 필요하지 않은 부트스트랩 리소스는 삭제합니다.

사전 요구 사항

- "플레이북 종속 항목 다운로드"에서 모듈을 다운로드했습니다.
- "설치 플레이북 다운로드"에서 플레이북을 다운로드했습니다.
- inventory.yaml, common.yaml 및 down-bootstrap.yaml Ansible 플레이북이 공통 디렉터리에 있습니다.

- 컨트롤 플레인 시스템이 실행 중입니다.
  - 시스템 상태를 모르는 경우 “클러스터 상태 확인”을 참조하십시오.

프로세스

1. 명령줄에서 작업 디렉터리를 플레이북의 위치로 변경합니다.
2. 명령줄에서 **down-bootstrap.yaml** 플레이북을 실행합니다.

```
$ ansible-playbook -i inventory.yaml down-bootstrap.yaml
```

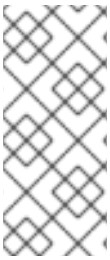
부트스트랩 포트, 서버 및 부동 **IP** 주소가 삭제됩니다.

주의

이전에 부트스트랩 **Ignition** 파일 **URL**을 비활성화하지 않은 경우 지금 비활성화합니다.

15.7.22. 컴퓨팅 머신용 SR-IOV 네트워크 생성

RHOSP(Red Hat OpenStack Platform) 배포에서 단일 루트 **I/O** 가상화(**SR-IOV**)를 지원하는 경우 컴퓨팅 머신이 실행되는 **SR-IOV** 네트워크를 프로비저닝할 수 있습니다.



참고

다음 지침은 컴퓨팅 머신에 연결할 수 있는 외부 플랫 네트워크 및 외부 **VLAN** 기반 네트워크 생성에 대해 설명합니다. **RHOSP** 배포에 따라 다른 네트워크 유형이 필요할 수 있습니다.

사전 요구 사항

- 클러스터는 **SR-IOV**를 지원합니다.

참고

클러스터가 지원하는 기능을 잘 모를 경우 **OpenShift Container Platform SR-IOV** 하드웨어 네트워크 설명서를 참조하십시오.

- **RHOSP** 배포의 일부로 **radio** 및 **uplink** 공급자 네트워크를 생성했습니다. **radio** 및 **uplink** 이름은 이러한 네트워크를 표시하기 위해 모든 예제 명령에서 사용됩니다.

프로세스

1. 명령줄에서 **radio RHOSP** 네트워크를 생성합니다.

```
$ openstack network create radio --provider-physical-network radio --provider-
network-type flat --external
```

2. **uplink RHOSP** 네트워크를 생성합니다.

```
$ openstack network create uplink --provider-physical-network uplink --provider-
network-type vlan --external
```

3. **radio** 네트워크의 서브넷을 생성합니다.

```
$ openstack subnet create --network radio --subnet-range
<radio_network_subnet_range> radio
```

4. **uplink** 네트워크의 서브넷을 생성합니다.

```
$ openstack subnet create --network uplink --subnet-range
<uplink_network_subnet_range> uplink
```

15.7.23. SR-IOV 네트워크에서 실행되는 컴퓨팅 시스템 생성

컨트롤 플레인을 가동시킨 후 " 컴퓨팅 머신의 **SR-IOV** 네트워크 생성"에서 만든 **SR-IOV** 네트워크에서 실행되는 컴퓨팅 머신을 생성합니다.

사전 요구 사항

- "플레이북 종속 항목 다운로드"에서 모듈을 다운로드했습니다.
- "설치 플레이북 다운로드"에서 플레이북을 다운로드했습니다.
- 설치 프로그램에서 생성된 **metadata.yaml** 파일이 **Ansible** 플레이북과 동일한 디렉토리에 있습니다.
- 컨트롤 플레인이 활성화되었습니다.
- "컴퓨팅 머신의 SR-IOV 네트워크 생성"에 설명된 대로 **radio** 및 **uplink SR-IOV** 네트워크를 생성했습니다.

프로세스

1. 명령줄에서 작업 디렉토리를 **inventory.yaml** 및 **common.yaml** 파일의 위치로 변경합니다.
2. **additionalNetworks** 매개변수를 사용하여 **inventory.yaml** 파일의 끝에 **radio** 및 **uplink** 네트워크를 추가합니다.

```
....
# If this value is non-empty, the corresponding floating IP will be
# attached to the bootstrap machine. This is needed for collecting logs
# in case of install failure.
  os_bootstrap_fip: '203.0.113.20'

  additionalNetworks:
  - id: radio
    count: 4 (1)
    type: direct
    port_security_enabled: no
  - id: uplink
    count: 4 (2)
    type: direct
    port_security_enabled: no
```

(1) (2) **count** 매개변수는 각 작업자 노드에 연결할 **SR-IOV** 가상 함수(VF)의 수를 정의합니다. 이 경우 각 네트워크에는 **4** 개의 **VF**가 있습니다.

3. compute-nodes.yaml 파일의 내용을 다음 텍스트로 바꿉니다.

예 15.1. compute-nodes.yaml

```
- import_playbook: common.yaml

- hosts: all
  gather_facts: no

  vars:
    worker_list: []
    port_name_list: []
    nic_list: []

  tasks:
  # Create the SDN/primary port for each worker node
  - name: 'Create the Compute ports'
    os_port:
      name: "{{ item.1 }}-{{ item.0 }}"
      network: "{{ os_network }}"
      security_groups:
      - "{{ os_sg_worker }}"
      allowed_address_pairs:
      - ip_address: "{{ os_ingressVIP }}"
    with_indexed_items: "{{ [os_port_worker] * os_compute_nodes_number }}"
    register: ports

  # Tag each SDN/primary port with cluster name
  - name: 'Set Compute ports tag'
    command:
      cmd: "openstack port set --tag {{ cluster_id_tag }} {{ item.1 }}-{{ item.0 }}"
    with_indexed_items: "{{ [os_port_worker] * os_compute_nodes_number }}"

  - name: 'List the Compute Trunks'
    command:
      cmd: "openstack network trunk list"
    when: os_networking_type == "Kuryr"
    register: compute_trunks

  - name: 'Create the Compute trunks'
    command:
      cmd: "openstack network trunk create --parent-port {{ item.1.id }} {{
os_compute_trunk_name }}-{{ item.0 }}"
    with_indexed_items: "{{ ports.results }}"
    when:
    - os_networking_type == "Kuryr"
    - "os_compute_trunk_name|string not in compute_trunks.stdout"

  - name: 'Call additional-port processing'
    include_tasks: additional-ports.yaml

  # Create additional ports in OpenStack
  - name: 'Create additionalNetworks ports'
    os_port:
      name:  "{{ item.0 }}-{{ item.1.name }}"
```

```
    vnic_type: "{{ item.1.type }}"
    network: "{{ item.1.uuid }}"
    port_security_enabled: "{{ item.1.port_security_enabled|default(omit) }}"
    no_security_groups: "{{ 'true' if item.1.security_groups is not defined else omit
}}"
    security_groups: "{{ item.1.security_groups | default(omit) }}"
  with_nested:
    - "{{ worker_list }}"
    - "{{ port_name_list }}"

# Tag the ports with the cluster info
- name: 'Set additionalNetworks ports tag'
  command:
    cmd: "openstack port set --tag {{ cluster_id_tag }} {{ item.0 }}-{{ item.1.name }}"
  with_nested:
    - "{{ worker_list }}"
    - "{{ port_name_list }}"

# Build the nic list to use for server create
- name: Build nic list
  set_fact:
    nic_list: "{{ nic_list | default([]) + [ item.name ] }}"
  with_items: "{{ port_name_list }}"

# Create the servers
- name: 'Create the Compute servers'
  vars:
    worker_nics: "{{ [ item.1 ] | product(nic_list) | map('join','-') |
map('regex_replace', '(.*)', 'port-name=\\1') | list }}"
  os_server:
    name: "{{ item.1 }}"
    image: "{{ os_image_rhcos }}"
    flavor: "{{ os_flavor_worker }}"
    auto_ip: no
    userdata: "{{ lookup('file', 'worker.ign') | string }}"
    security_groups: []
    nics:  "{{ [ 'port-name=' + os_port_worker + '-' + item.0|string ] + worker_nics }}"
    config_drive: yes
  with_indexed_items: "{{ worker_list }}"
```

4. 로컬 파일 **additional-ports.yaml**에 다음 내용을 삽입합니다.

예 15.2. additional-ports.yaml

```
# Build a list of worker nodes with indexes
- name: 'Build worker list'
  set_fact:
    worker_list: "{{ worker_list | default([]) + [ item.1 + '-' + item.0 | string ] }}"
  with_indexed_items: "{{ [ os_compute_server_name ] *
os_compute_nodes_number }}"

# Ensure that each network specified in additionalNetworks exists
- name: 'Verify additionalNetworks'
```

```
  os_networks_info:
    name: "{{ item.id }}"
  with_items: "{{ additionalNetworks }}"
  register: network_info

# Expand additionalNetworks by the count parameter in each network definition
- name: 'Build port and port index list for additionalNetworks'
  set_fact:
    port_list: "{{ port_list | default([]) + [ {
                  'net_name' : item.1.id,
                  'uuid' : network_info.results[item.0].openstack_networks[0].id,
                  'type' : item.1.type|default('normal'),
                  'security_groups' : item.1.security_groups|default(omit),
                  'port_security_enabled' : item.1.port_security_enabled|default(omit)
                  } ] * item.1.count|default(1) }}"
    index_list: "{{ index_list | default([]) + range(item.1.count|default(1)) | list }}"
  with_indexed_items: "{{ additionalNetworks }}"

# Calculate and save the name of the port
# The format of the name is cluster_name-worker-workerID-networkUUID(partial)-
count
# i.e. fdp-nz995-worker-1-99bcd111-1
- name: 'Calculate port name'
  set_fact:
    port_name_list: "{{ port_name_list | default([]) + [ item.1 | combine( {'name' :
item.1.uuid | regex_search('([^-]+)') + '-' + index_list[item.0]|string } ) ] }}"
  with_indexed_items: "{{ port_list }}"
  when: port_list is defined
```

5. 명령줄에서 **compute-nodes.yaml** 플레이북을 실행합니다.

```
$ ansible-playbook -i inventory.yaml compute-nodes.yaml
```

15.7.24. 머신의 인증서 서명 요청 승인

클러스터에 시스템을 추가하면 추가한 시스템별로 보류 중인 인증서 서명 요청(CSR)이 두 개씩 생성됩니다. 이러한 CSR이 승인되었는지 확인해야 하며, 필요한 경우 이를 직접 승인해야 합니다. 클라이언트 요청을 먼저 승인한 다음 서버 요청을 승인해야 합니다.

사전 요구 사항

- 클러스터에 시스템을 추가했습니다.

프로세스

1. 클러스터가 시스템을 인식하는지 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0  Ready    master  63m  v1.22.1
master-1  Ready    master  63m  v1.22.1
master-2  Ready    master  64m  v1.22.1
```

출력에 생성된 모든 시스템이 나열됩니다.

참고

이전 출력에는 일부 **CSR**이 승인될 때까지 컴퓨팅 노드(작업자 노드라고도 함)가 포함되지 않을 수 있습니다.

2. 보류 중인 **CSR**을 검토하고 클러스터에 추가한 각 시스템에 대해 **Pending** 또는 **Approved** 상태의 클라이언트 및 서버 요청이 표시되는지 확인합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE    REQUESTOR                                                   CONDITION
csr-8b2br   15m    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

예에서는 두 시스템이 클러스터에 참여하고 있습니다. 목록에는 승인된 CSR이 더 많이 나타날 수도 있습니다.

3. CSR이 승인되지 않은 경우, 추가된 시스템에 대한 모든 보류 중인 CSR이 Pending 상태로 전환된 후 클러스터 시스템의 CSR을 승인합니다.

참고

CSR은 교체 주기가 자동으로 만료되므로 클러스터에 시스템을 추가한 후 1시간 이내에 CSR을 승인하십시오. 한 시간 내에 승인하지 않으면 인증서가 교체되고 각 노드에 대해 두 개 이상의 인증서가 표시됩니다. 이러한 인증서를 모두 승인해야 합니다. 클라이언트 CSR이 승인되면 Kubelet은 인증서에 대한 보조 CSR을 생성하므로 수동 승인이 필요합니다. 그러면 Kubelet에서 동일한 매개변수를 사용하여 새 인증서를 요청하는 경우 인증서 갱신 요청은 machine-approver에 의해 자동으로 승인됩니다.

참고

베어 메탈 및 기타 사용자 프로비저닝 인프라와 같이 머신 API를 사용하도록 활성화되지 않는 플랫폼에서 실행되는 클러스터의 경우 CSR(Kubelet service Certificate Request)을 자동으로 승인하는 방법을 구현해야 합니다. 요청이 승인되지 않으면 API 서버가 kubelet에 연결될 때 서비스 인증서가 필요하므로 oc exec, oc rsh, oc logs 명령을 성공적으로 수행할 수 없습니다. Kubelet 엔드 포인트에 연결하는 모든 작업을 수행하려면 이 인증서 승인이 필요합니다. 이 방법은 새 CSR을 감시하고 CSR이 system:node 또는 system:admin 그룹의 node-bootstrapper 서비스 계정에 의해 제출되었는지 확인하고 노드의 ID를 확인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 <csr_name>은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

참고

일부 Operator는 일부 CSR이 승인될 때까지 사용할 수 없습니다.

4. 이제 클라이언트 요청이 승인되었으므로 클러스터에 추가한 각 머신의 서버 요청을 검토해야 합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal        Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal        Pending
...
```

5. 나머지 CSR이 승인되지 않고 Pending 상태인 경우 클러스터 머신의 CSR을 승인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 <csr_name>은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs oc adm certificate approve
```

6. 모든 클라이언트 및 서버 CSR이 승인된 후 머신은 Ready 상태가 됩니다. 다음 명령을 실행하여 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0  Ready    master  73m  v1.22.1
master-1  Ready    master  73m  v1.22.1
master-2  Ready    master  74m  v1.22.1
worker-0  Ready    worker  11m  v1.22.1
worker-1  Ready    worker  11m  v1.22.1
```

참고

머신이 **Ready** 상태로 전환하는 데 서버 **CSR**의 승인 후 몇 분이 걸릴 수 있습니다.

추가 정보

- **CSR**에 대한 자세한 내용은 인증서 서명 요청을 참조하십시오.

15.7.25. 설치 성공 확인

OpenShift Container Platform 설치가 완료되었는지 확인합니다.

사전 요구 사항

- 설치 프로그램(**openshift-install**)이 있습니다.

프로세스

- 명령줄에 다음을 입력합니다.

```
$ openshift-install --log-level debug wait-for install-complete
```

콘솔 URL과 관리자의 로그인 정보가 출력됩니다.

클러스터가 작동하고 있습니다. SR-IOV 네트워크에 맞게 구성하려면 먼저 추가 작업을 수행해야 합니다.

15.7.26. SR-IOV 용 RHOSP에서 실행되는 클러스터 준비

RHOSP (Red Hat OpenStack Platform)에서 실행되는 클러스터에서 SR-IOV(Single Root I/O Virtualization)를 사용하기 전에 RHOSP 메타데이터 서비스를 드라이브로 마운트하고 가상 함수 I/O (VFIO) 드라이버에서 No-IOMMU Operator를 활성화합니다.

15.7.26.1. RHOSP 메타데이터 서비스를 마운트 가능한 드라이브로 활성화

RHOSP(Red Hat OpenStack Platform) 메타데이터 서비스를 마운트 가능 드라이브로 사용할 수 있도록 시스템 풀에 머신 구성을 적용할 수 있습니다.

다음 머신 구성을 사용하면 SR-IOV Network Operator 내에서 RHOSP 네트워크 UUID를 표시할 수 있습니다. 이 구성으로 SR-IOV 리소스를 클러스터 SR-IOV 리소스에 간단하게 연결할 수 있습니다.

프로세스

1. 다음 템플릿에서 머신 구성 파일을 생성합니다.

마운트 가능한 메타데이터 서비스 머신 구성 파일

```
kind: MachineConfig
apiVersion: machineconfiguration.openshift.io/v1
metadata:
  name: 20-mount-config (1)
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 3.2.0
    systemd:
      units:
        - name: create-mountpoint-var-config.service
          enabled: true
          contents: |
```

```
      [Unit]
      Description=Create mountpoint /var/config
      Before=kubelet.service

      [Service]
      ExecStart=/bin/mkdir -p /var/config

      [Install]
      WantedBy=var-config.mount

    - name: var-config.mount
      enabled: true
      contents: |
      [Unit]
      Before=local-fs.target
      [Mount]
      Where=/var/config
      What=/dev/disk/by-label/config-2
      [Install]
      WantedBy=local-fs.target
```

1

선택한 이름을 대체할 수 있습니다.

2. 명령줄에서 머신 구성을 적용합니다.

```
$ oc apply -f <machine_config_file_name>.yaml
```

15.7.26.2. RHOSP VFIO 드라이버의 No-IOMMU 기능 활성화

머신 풀에 머신 구성을 적용하여 **RHOSP(Red Hat OpenStack Platform)** 가상 기능 **I/O(VFIO)** 드라이버에 대한 **No-IOMMU** 기능을 활성화할 수 있습니다. **RHOSP vfio-pci** 드라이버에는 이 기능이 필요합니다.

프로세스

1. 다음 템플릿에서 머신 구성 파일을 생성합니다.

No-IOMMU VFIO 머신 구성 파일

-

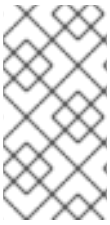
```
kind: MachineConfig
apiVersion: machineconfiguration.openshift.io/v1
metadata:
  name: 99-vfio-noiommu 1
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
      - path: /etc/modprobe.d/vfio-noiommu.conf
        mode: 0644
        contents:
          source:
data:;base64,b3B0aW9ucyB2ZmlvIGVuYWJsZV91bnNhZmVfbm9pb21tdV9tb2RlPTEK
```

1

선택한 이름을 대체할 수 있습니다.

2. 명령줄에서 머신 구성을 적용합니다.

```
$ oc apply -f <machine_config_file_name>.yaml
```



참고

머신 풀에 머신 구성을 적용한 후 머신 구성 풀 상태를 확인하여 머신을 사용할 수 있는지 확인할 수 있습니다.

클러스터가 SR-IOV 구성을 설치 및 준비합니다. 이제 "다음 단계"에서 SR-IOV 구성 작업을 수행해야 합니다.

15.7.27. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

15.7.28. 추가 리소스

- 실시간 실행 및 짧은 대기 시간을 위해 배포를 구성하는 방법에 대한 정보는 대기 시간이 짧은 노드의 Performance Addon Operator에서 참조하십시오.

15.7.29. 다음 단계

- 클러스터의 SR-IOV 구성을 완료하려면 다음을 수행합니다.
  - Performance Addon Operator를 설치합니다.
  - 대용량 페이지 지원을 사용하여 Performance Addon Operator를 구성합니다.
  - SR-IOV Operator를 설치합니다.
  - SR-IOV 네트워크 장치를 구성합니다.
- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.
- 노드 포트에 대한 외부 액세스를 활성화해야 하는 경우 노드 포트를 사용하여 Ingress 클러스터 트래픽을 구성합니다.
-

유동 IP 주소를 통한 애플리케이션 트래픽을 허용하도록 RHOSP를 구성하지 않은 경우 유동 IP 주소로 RHOSP 액세스를 구성 합니다.

15.8. 네트워크가 제한된 환경에서 OPENSTACK에 클러스터 설치

OpenShift Container Platform 4.9에서는 설치 릴리스 콘텐츠의 내부 미러를 생성하여 제한된 네트워크의 RHOSP(Red Hat OpenStack Platform)에 클러스터를 설치할 수 있습니다.

15.8.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.
- OpenShift Container Platform 4.9가 OpenShift 클러스터에 지원되는 플랫폼 섹션을 사용하여 현재 RHOSP 버전과 호환되는지 확인했습니다. RHOSP 지원 매트릭스의 OpenShift Container Platform을 확인하여 여러 버전의 플랫폼 지원을 비교할 수도 있습니다.
- 미러 호스트에 레지스트리를 생성하고 사용 중인 OpenShift Container Platform 버전의 imageContentSources 데이터를 가져옵니다.

중요

미러 호스트에 설치 미디어가 있으므로 해당 컴퓨터를 사용하여 모든 설치 단계를 완료하십시오.

- RHOSP에서 메타데이터 서비스 활성화

15.8.2. 네트워크가 제한된 환경에서의 설치 정보

OpenShift Container Platform 4.9에서는 소프트웨어 구성 요소를 받기 위한 인터넷 접속이 필요하지 않은 설치를 수행할 수 있습니다. 제한된 네트워크 설치는 클러스터를 설치하는 클라우드 플랫폼에 따라 설치 관리자 프로비저닝 인프라 또는 사용자 프로비저닝 인프라를 사용하여 완료할 수 있습니다.

클라우드 플랫폼에 제한된 네트워크 설치를 수행하는 방법을 선택해도 클라우드 API에 액세스는 가능해야 합니다. Amazon Web Service의 Route 53 DNS 및 IAM 서비스와 같은 일부 클라우드 기능에는 인

터넷 액세스가 필요합니다. 사용 중인 네트워크에 따라 베어메탈 하드웨어 또는 VMware vSphere에 설치하기 위해 필요한 인터넷 액세스가 줄어들 수 있습니다.

제한된 네트워크 설치를 완료하려면 OpenShift Container Platform 레지스트리의 내용을 미러링하고 설치 미디어를 포함할 레지스트리를 생성해야 합니다. 인터넷과 폐쇄 네트워크에 모두 액세스하거나 제한 사항을 따르는 다른 방법을 통해 미러 호스트에 레지스트리를 생성할 수 있습니다.

15.8.2.1. 추가 제한

제한된 네트워크의 클러스터에는 다음과 같은 추가 제한이 있습니다.

- ClusterVersion 상태에 사용 가능한 업데이트를 검색할 수 없음 오류가 포함되어 있습니다.
- 기본적으로 필요한 이미지 스트림 태그에 액세스할 수 없기 때문에 개발자 카탈로그의 내용을 사용할 수 없습니다.

15.8.3. RHOSP에 OpenShift Container Platform을 설치하기 위한 리소스 지침

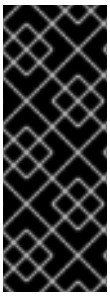
OpenShift Container Platform 설치를 지원하려면 RHOSP(Red Hat OpenStack Platform) 할당량이 다음 요구사항을 충족해야 합니다.

표 15.35. RHOSP의 기본 OpenShift Container Platform 클러스터에 권장되는 리소스

리소스 이름	값
부동 IP 주소	3
포트	15
라우터	1
서브넷	1
RAM	ECDHEGB
vCPU	22
볼륨 스토리지	275GB
인스턴스	7

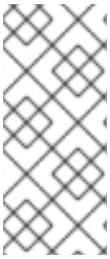
리소스 이름	값
보안 그룹	3
보안 그룹 규칙	60

권장 리소스보다 적은 리소스로도 클러스터가 작동할 수 있지만 성능은 보장되지 않습니다.



중요

swiftoperator 역할을 가진 사용자 계정으로 **RHOSP** 개체 스토리지(**Swift**)를 사용하고 운영하는 경우 **OpenShift Container Platform** 이미지 레지스트리의 기본 백엔드로 사용됩니다. 이 경우 볼륨 스토리지 요구사항은 **175GB**입니다. **Swift** 공간 요구사항은 이미지 레지스트리의 크기에 따라 다릅니다.



참고

기본적으로 보안 그룹 및 보안 그룹 규칙 할당량이 적을 수 있습니다. 문제가 발생하면 관리자로 **openstack quota set --secgroups 3 --secgroup-rules 60 <project>**를 실행하여 할당량을 늘립니다.

OpenShift Container Platform 배포는 컨트롤 플레인 시스템, 컴퓨팅 시스템, 부트스트랩 시스템으로 구성됩니다.

15.8.3.1. 컨트롤 플레인 머신

기본적으로 **OpenShift Container Platform** 설치 프로세스는 세 개의 컨트롤 플레인 시스템을 생성합니다.

각 시스템의 요구사항은 다음과 같습니다.

- **RHOSP** 할당량의 인스턴스
- **RHOSP** 할당량의 포트

- 최소 16GB 메모리 및 vCPU 4개가 있는 플레이버
- RHOSP 할당량에서 최소 100GB 스토리지 공간

15.8.3.2. 컴퓨팅 머신

기본적으로 OpenShift Container Platform 설치 프로세스는 세 개의 컴퓨팅 시스템을 생성합니다.

각 시스템의 요구사항은 다음과 같습니다.

- RHOSP 할당량의 인스턴스
- RHOSP 할당량의 포트
- 8GB 메모리 및 vCPU 2개 이상 있는 플레이버
- RHOSP 할당량에서 최소 100GB 스토리지 공간

작은 정보

컴퓨팅 시스템은 OpenShift Container Platform에서 실행하는 애플리케이션을 호스팅합니다. 최대한 많이 실행하는 것이 좋습니다.

15.8.3.3. 부트스트랩 시스템

설치하는 동안 컨트롤 플레인을 유지하기 위해 부트스트랩 시스템이 임시로 프로비저닝됩니다. 프로덕션 컨트롤 플레인이 준비되면 부트스트랩 시스템이 프로비저닝 해제됩니다.

부트스트랩 시스템의 요구사항은 다음과 같습니다.

- RHOSP 할당량의 인스턴스

- RHOSP 할당량의 포트
- 최소 16GB 메모리 및 vCPU 4개가 있는 플레이버
- RHOSP 할당량에서 최소 100GB 스토리지 공간

15.8.4. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하기 위해 필요한 이미지를 받으려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

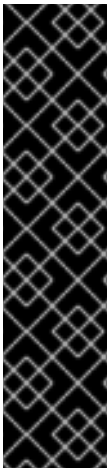
- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.

중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

15.8.5. RHOSP에서 Swift 활성화

Swift는 swiftoperator 역할을 가진 사용자 계정으로 운영됩니다. 설치 프로그램을 실행하려면 먼저 계정에 이 역할을 추가합니다.



중요

RHOSP(Red Hat OpenStack Platform) 개체 스토리지 서비스(일반적으로 Swift로 알려짐)를 사용할 수 있는 경우 OpenShift Container Platform이 이미지 레지스트리 스토리지로 사용합니다. 이 서비스를 사용할 수 없는 경우에는 설치 프로그램이 RHOSP 블록 스토리지 서비스(일반적으로 Cinder로 알려짐)를 사용합니다.

Swift가 있고 Swift를 사용하려면 액세스를 활성화해야 합니다. 존재하지 않거나 사용하지 않으려면 이 섹션을 건너 뛰십시오.

사전 요구 사항

- 대상 환경에 RHOSP 관리자 계정이 있습니다.
- Swift 서비스가 설치되어 있습니다.
- Ceph RGW에서 url의 계정 옵션이 활성화되어 있습니다.

프로세스

RHOSP에서 Swift를 활성화하려면:

1. RHOSP CLI의 관리자로서 Swift에 액세스할 계정에 swiftoperator 역할을 추가하십시오.

```
$ openstack role add --user <user> --project <project> swiftoperator
```

이제 RHOSP 배포에서 이미지 레지스트리에 Swift를 사용할 수 있습니다.

15.8.6. 설치 프로그램에 대한 매개변수 정의

OpenShift Container Platform 설치 프로그램은 clouds.yaml 파일을 사용합니다. 이 파일은 프로젝트 이름, 로그인 정보, 인증 서비스 URL 등 RHOSP(Red Hat OpenStack Platform) 구성 매개변수를 설명합니다.

프로세스

1. clouds.yaml 파일을 만듭니다.

   - RHOSP 배포에 Horizon 웹 UI가 포함되어 있으면 그 안에 clouds.yaml 파일을 만듭니다.

     중요

     auth 필드에 암호를 추가해야 합니다. clouds.yaml과 별도의 파일에 비밀을 저장할 수도 있습니다.

   - RHOSP 배포에 Horizon 웹 UI가 포함되어 있지 않거나 Horizon을 사용하지 않으려면 파일을 직접 만듭니다. clouds.yaml에 대한 자세한 내용은 RHOSP 문서의 구성 파일을 참조하십시오.

     ```
     clouds:
       shiftstack:
         auth:
           auth_url: http://10.10.14.42:5000/v3
           project_name: shiftstack
           username: shiftstack_user
           password: XXX
           user_domain_name: Default
           project_domain_name: Default
       dev-env:
         region_name: RegionOne
         auth:
           username: 'devuser'
           password: XXX
           project_name: 'devonly'
           auth_url: 'https://10.10.14.22:5001/v2.0'
     ```

2. RHOSP 설치에서 끝점 인증을 위해 자체 서명된 CA(인증 기관) 인증서를 사용하는 경우:

   a. 인증 기관 파일을 시스템에 복사합니다.

   b. cacerts 키를 clouds.yaml 파일에 추가합니다. 값은 CA 인증서에 대한 루트가 아닌 액세스 가능한 절대 경로여야 합니다.

      ```
      clouds:
        shiftstack:
          ...
          cacert: "/etc/pki/ca-trust/source/anchors/ca.crt.pem"
      ```

작은 정보

사용자 지정 CA 인증서로 설치 관리자를 실행한 후 cloud-provider-config 키맵에서 ca-cert.pem 키의 값을 편집하여 인증서를 업데이트할 수 있습니다. 명령줄에서 다음을 실행합니다.

```
$ oc edit configmap -n openshift-config cloud-provider-config
```

3. clouds.yaml 파일을 다음 위치 중 하나에 배치합니다.

a. OS_CLIENT_CONFIG_FILE 환경 변수의 값

b. 현재 디렉터리

c. Unix 전용 사용자 구성 디렉터리(예: ~/.config/openstack/clouds.yaml)

d. Unix 전용 사이트 구성 디렉터리(예: /etc/openstack/clouds.yaml)

설치 프로그램은 clouds.yaml을 이 순서대로 검색합니다.

15.8.7. 클라우드 공급자 옵션 설정

선택적으로 클러스터의 클라우드 공급자 구성을 편집할 수 있습니다. 클라우드 공급자 구성은 OpenShift Container Platform이 RHOSP(Red Hat OpenStack Platform)와 상호 작용하는 방법을 제어합니다.

클라우드 공급자 구성 매개변수의 전체 목록은 "OpenStack 클라우드 구성 참조 가이드" 페이지의 "OpenStack 클라우드 구성 참조 가이드" 페이지를 참조하십시오.

프로세스

1. 클러스터에 대해 이미 생성된 매니페스트 파일이 없는 경우 다음 명령을 실행하여 생성합니다.

```
$ openshift-install --dir <destination_directory> create manifests
```

2. 텍스트 편집기에서 **cloud-provider** 구성 매니페스트 파일을 엽니다. 예를 들면 다음과 같습니다.

```
$ vi openshift/manifests/cloud-provider-config.yaml
```

3. 클라우드 구성 사양에 따라 옵션을 수정합니다.

로드 밸런싱을 위해 **Octavia**를 구성하는 것은 **Kuryr**를 사용하지 않는 클러스터의 일반적인 사례입니다. 예를 들면 다음과 같습니다.

```
#...
[LoadBalancer]
use-octavia=true 1
lb-provider = "amphora" 2
floating-network-id="d3deb660-4190-40a3-91f1-37326fe6ec4a"3
#...
```

1 이 속성은 **Octavia** 통합을 활성화합니다.

2 이 속성은 로드 밸런서에서 사용하는 **Octavia** 공급자를 설정합니다. **"ovn"** 또는 **"amphora"** 를 값으로 허용합니다. **OVN**을 사용하도록 선택하는 경우 **lb-method** 도 **SOURCE_IP_PORT** 로 설정해야 합니다.

3 이 속성은 클러스터에서 여러 외부 네트워크를 사용하려는 경우 필요합니다. 클라우드 프로바이더는 여기에 지정된 네트워크에 유동 **IP** 주소를 생성합니다.

중요

변경 사항을 저장하기 전에 파일이 올바르게 구성되었는지 확인합니다. 속성을 적절한 섹션에 배치하지 않으면 클러스터가 실패할 수 있습니다.

중요

Kuryr를 사용하는 설치의 경우 **Kuryr**는 관련 서비스를 처리합니다. 클라우드 공급자에서 **Octavia** 로드 밸런싱을 구성할 필요가 없습니다.

4. 변경 사항을 파일에 저장하고 설치를 진행합니다.

작은 정보

설치 프로그램을 실행한 후 클라우드 공급자 구성을 업데이트할 수 있습니다. 명령줄에서 다음을 실행합니다.

```
$ oc edit configmap -n openshift-config cloud-provider-config
```

변경 사항을 저장한 후 클러스터를 재구성하는 데 다소 시간이 걸립니다. 노드가 **SchedulingDisabled** 상태가 없는 경우 프로세스가 완료됩니다.

추가 리소스

- 클라우드 공급자 구성에 대한 자세한 내용은 **OpenStack** 클라우드 공급자 옵션을 참조하십시오.

15.8.8. 제한된 네트워크 설치를 위한 RHCOS 이미지 생성

Red Hat Enterprise Linux CoreOS(RHCOS) 이미지를 다운로드하여 제한된 네트워크 **RHOSP(Red Hat OpenStack Platform)** 환경에 **OpenShift Container Platform**을 설치하십시오.

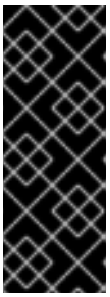
사전 요구 사항

- **OpenShift Container Platform** 설치 프로그램을 가져옵니다. 제한된 네트워크 설치의 경우 프로그램은 미러 레지스트리 호스트에 있습니다.

프로세스

1. **Red Hat Customer Portal**의 제품 다운로드 페이지에 로그인합니다.

2. **Version**에서 **RHEL 8**용 최신 **OpenShift Container Platform 4.9** 릴리스를 선택합니다.



중요

RHCOS 이미지는 **OpenShift Container Platform** 릴리스에 따라 변경되지 않을 수 있습니다. 설치하는 **OpenShift Container Platform** 버전과 같거나 그 이하의 버전 중 가장 최신 버전의 이미지를 다운로드해야 합니다. 사용 가능한 경우 **OpenShift Container Platform** 버전과 일치하는 이미지 버전을 사용합니다.

3. **RHCOS(Red Hat Enterprise Linux CoreOS) - OpenStack Image (QCOW)** 이미지를 다운로드합니다.

4. 이미지 압축을 풉니다.

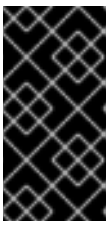
참고

이미지 압축을 풀어야 클러스터에서 이미지를 사용할 수 있습니다. 다운로드한 파일의 이름에 **.gz** 또는 **.tgz**와 같은 압축 확장자가 포함되지 않을 수 있습니다. 파일 압축 여부를 확인하려면 명령줄에서 다음을 입력합니다.

```
$ file <name_of_downloaded_file>
```

5. **Glance**와 같이 **bastion** 서버에서 액세스할 수 있는 위치에 압축을 푼 이미지를 업로드합니다. 예를 들면 다음과 같습니다.

```
$ openstack image create --file rhcos-44.81.202003110027-0-openstack.x86_64.qcow2 --
disk-format qcow2 rhcos-${RHCOS_VERSION}
```



중요

RHOSP 환경에 따라 이미지를 **.raw** 또는 **.qcow2** 형식으로 업로드할 수 있습니다. **Ceph**를 사용하는 경우 **.raw** 형식을 사용해야 합니다.

주의

설치 프로그램이 이름이 같은 여러 이미지를 발견하면 그 중 하나를 임의로 선택합니다. 이 동작을 방지하려면 RHOSP에서 리소스의 고유한 이름을 만듭니다.

이제 이미지를 제한된 설치에 사용할 수 있습니다. OpenShift Container Platform 배포에 사용할 이미지 이름 또는 위치를 기록해 둡니다.

15.8.9. 설치 구성 파일 만들기

RHOSP(Red Hat OpenStack Platform)에 설치하는 OpenShift Container Platform 클러스터를 사용자 지정할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다. 제한된 네트워크 설치의 경우, 해당 파일은 미러 호스트에 있습니다.
- 미러 레지스트리 작성 중에 생성된 imageContentSources 값이 있어야 합니다.
- 미러 레지스트리에 대한 인증서의 내용을 가져옵니다.
- RHCOS (Red Hat Enterprise Linux CoreOS) 이미지를 검색하여 액세스 가능한 위치에 업로드합니다.
- 서브스크립션 수준에서 서비스 주체 권한을 획득합니다.

절차

1. install-config.yaml 파일을 생성합니다.

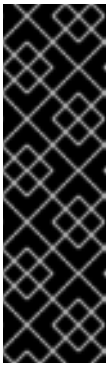
   a.

설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1

<installation_directory>는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.

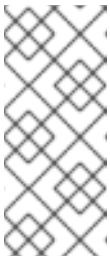


중요

비어 있는 디렉터리를 지정합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

b. 화면에 나타나는 지시에 따라 클라우드에 대한 구성 세부 사항을 입력합니다.

i. 선택사항: 클러스터 시스템에 액세스하는 데 사용할 SSH 키를 선택합니다.



참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 ssh-agent 프로세스가 사용하는 SSH 키를 지정합니다.

ii. 대상 플랫폼으로 openstack을 선택합니다.

iii. 클러스터 설치에 사용할 RHOSP(Red Hat OpenStack Platform) 외부 네트워크 이름을 지정합니다.

iv. OpenShift API에 대한 외부 액세스에 사용할 부동 IP 주소를 지정합니다.

v. 컨트롤 플레인 노드에 사용할 최소 16GB의 RAM과 컴퓨팅 노드에 8GB RAM이 있는 RHOSP 플레이버를 지정합니다.

vi. 클러스터를 배포할 기본 도메인을 선택합니다. 모든 **DNS** 레코드는 이 기본 도메인의 하위 도메인이 되므로 클러스터 이름을 포함합니다.

vii. 클러스터 이름을 입력합니다. 이름은 **14**자 이하여야 합니다.

viii. Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 붙여넣습니다.

2. **install-config.yaml** 파일에서 **platform.openstack.clusterOSImage** 값을 이미지 위치 또는 이름으로 설정합니다. 예를 들면 다음과 같습니다.

```
platform:
 openstack:
   clusterOSImage: http://mirror.example.com/images/rhcos-43.81.201912131630.0-
openstack.x86_64.qcow2.gz?
sha256=ffebbd68e8a1f2a245ca19522c16c86f67f9ac8e4e0c1f0a812b068b16f7265d
```

3. 제한된 네트워크에서의 설치에 필요한 추가 정보를 제공하려면 **install-config.yaml** 파일을 편집합니다.

a. 레지스트리의 인증 정보를 포함하도록 **pullSecret** 값을 업데이트합니다.

```
pullSecret: '{"auths":{"<mirror_host_name>:5000": {"auth": "
<credentials>","email": "you@example.com"}}}'
```

<mirror_host_name>의 경우 미러 레지스트리의 인증서에 지정한 레지스트리 도메인 이름을 지정하고 **<credentials>**의 경우 미러 레지스트리에 **base64**로 인코딩된 사용자 이름 및 암호를 지정합니다.

b. **additionalTrustBundle** 매개변수와 값을 추가합니다.

```
additionalTrustBundle: |
 -----BEGIN CERTIFICATE-----

ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
 -----END CERTIFICATE-----
```

값은 미러 레지스트리에 사용한 인증서 파일의 내용이어야 하며, 신뢰할 수 있는 기존

인증 기관 또는 미러 레지스트리에 대해 생성한 자체 서명 인증서일 수 있습니다.

c. 다음과 같은 이미지 콘텐츠 리소스를 추가합니다.

```
imageContentSources:
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: registry.redhat.io/ocp/release
```

이러한 값을 완료하려면 미러 레지스트리 생성 중에 기록한 **imageContentSources**를 사용하십시오.

4. 필요한 **install-config.yaml** 파일을 수정합니다. 사용 가능한 매개변수에 대한 자세한 정보는 **Installation configuration parameters** 섹션에서 확인할 수 있습니다.

5. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.

중요

install-config.yaml 파일은 설치 과정에서 사용됩니다. 이 파일을 재사용하려면 지금 백업해야 합니다.

15.8.9.1. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 **HTTP** 또는 **HTTPS** 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 **OpenShift Container Platform** 클러스터를 구성할 수 있습니다.

참고

Kuryr 설치의 기본값은 **HTTP** 프록시입니다.

사전 요구 사항

- 프록시 오브젝트를 사용하는 제한된 네트워크에 **Kuryr**를 설치하려면 프록시가 클러스터가 사용하는 라우터에 응답할 수 있어야 합니다. 프록시 구성에 대한 정적 경로를 추가하려면 **root**

사용자로 명령줄에서 다음을 입력합니다.

```
$ ip route add <cluster_network_cidr> via <installer_subnet_gateway>
```

- 제한된 서브넷에는 정의된 게이트웨이가 있고 **Kuryr**에서 생성하는 **Router** 리소스에 연결할 수 있는 게이트웨이가 있어야 합니다.

- 기존 **install-config.yaml** 파일이 있습니다.

- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 **API**에 대한 호출을 포함하여 모든 클러스터 발신(**Egress**) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.

참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), **Google Cloud Platform (GCP)**, **Microsoft Azure** 및 **Red Hat OpenStack Platform (RHOSP)**에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1

클러스터 외부에서 **HTTP** 연결을 구축하는 데 사용할 프록시 **URL**입니다. **URL** 스키마는 **http**여야 합니다.

2

클러스터 외부에서 **HTTPS** 연결을 구축하는 데 사용할 프록시 **URL**입니다.

3

대상 도메인 이름, **IP** 주소 또는 프록시에서 제외할 기타 네트워크 **CIDR**로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. ***를 사용하여 모든 대상에 대해 프록시를 바이패스합니다.

4

이 값을 제공하면 설치 프로그램에서 추가 **CA** 인증서를 보유할 **openshift -config** 네임스페이스에 **user-ca- bundle** 이라는 구성 맵을 생성합니다. **additionalTrustBundle** 및 하나 이상의 프록시 설정을 제공하는 경우 프록시 오브젝트는 **trustedCA** 필드의 **user-ca-bundle** 구성 맵을 참조하도록 구성됩니다. 그러면 **Cluster Network Operator**에서 **trustedCA** 매개변수에 대해 지정된 콘텐츠를 **RHCOS** 신뢰 번들과 병합하는 **trusted-ca-bundle** 구성 맵을 생성합니다. 프록시의 **ID** 인증서를 **RHCOS** 트러스트 번들에 있는 기관에서 서명하지 않은 경우 **additionalTrustBundle** 필드가 있어야 합니다.

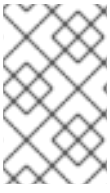
참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2.

파일을 저장해 놓고 **OpenShift Container Platform**을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 **nil**이 됩니다.

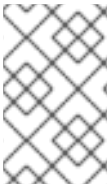


참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

15.8.9.2. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 매개변수 값을 제공하여 클러스터를 호스팅할 클라우드 플랫폼에서 사용자 계정을 설명하고 선택사항으로 클러스터의 플랫폼을 사용자 지정합니다. install-config.yaml 설치 구성 파일을 생성할 때 명령줄을 통해 필요한 매개변수 값을 제공합니다. 클러스터를 사용자 지정하면 install-config.yaml 파일을 수정하여 플랫폼에 대한 세부 정보를 제공할 수 있습니다.



참고

설치한 후에는 install-config.yaml 파일에서 이러한 매개변수를 수정할 수 없습니다.

중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

15.8.9.2.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 15.36. 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체

매개변수	설명	값
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	소문자, 하이픈(**-**), 마침표(**.**)로 구성되는 문자열(예: **dev**)입니다. 문자열은 14자 이하여야 합니다.
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	{
		"auths":{
		"cloud.openshift.com":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		},
		"quay.io":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		}
		}
		}

15.8.9.2.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 15.37. 네트워크 매개변수

매개변수	설명	값

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체 **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.
networking.clusterNetwork	Pod의 IP 주소 블록입니다. 기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된 **cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.

매개변수	설명	값
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16** **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

15.8.9.2.3. 선택적 구성 매개변수

선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 15.38. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.

매개변수	설명	값
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은 **3**입니다.
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열

매개변수	설명	값
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 기본값은 **External**입니다. 이 필드를 **Internal** 로 설정하는 것은 클라우드 이외의 플랫폼에서는 지원되지 않습니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

15.8.9.2.4. 추가 RHOSP(Red Hat OpenStack Platform) 구성 매개변수

추가 RHOSP 구성 매개변수는 다음 표에 설명되어 있습니다.

표 15.39. 추가 RHOSP 매개변수

매개변수	설명	값
compute.platform.openstack.rootVolume.size	컴퓨팅 시스템의 경우 루트 볼륨의 크기(GB)입니다. 이 값을 설정하지 않으면 시스템은 임시 스토리지를 사용합니다.	정수 (예: **30**)
compute.platform.openstack.rootVolume.type	컴퓨팅 시스템의 경우 루트 볼륨의 유형입니다.	문자열 (예: **performance**)
controlPlane.platform.openstack.rootVolume.size	컨트롤 플레인 시스템의 경우 루트 볼륨의 크기(GB)입니다. 이 값을 설정하지 않으면 시스템은 임시 스토리지를 사용합니다.	정수 (예: **30**)
controlPlane.platform.openstack.rootVolume.type	컨트롤 플레인 시스템의 경우 루트 볼륨의 유형입니다.	문자열 (예: **performance**)
platform.openstack.cloud	**clouds.yaml** 파일의 클라우드 목록에서 사용할 RHOSP 클라우드의 이름입니다.	문자열 (예: **MyCloud**)

매개변수	설명	값
platform.openstack.externalNetwork	설치에 사용할 RHOSP 외부 네트워크 이름입니다.	문자열 (예: **external)**
platform.openstack.computeFlavor	컨트롤 플레인 및 컴퓨팅 시스템에 사용할 RHOSP 버전입니다. 이 속성은 더 이상 사용되지 않습니다. 모든 머신 풀에 플레이버를 기본값으로 사용하려면 **platform.openstack.defaultMachinePlatform** 속성의 **type** 키 값으로 추가합니다. 각 머신 풀의 플레이버 값을 개별적으로 설정할 수도 있습니다.	문자열 (예: **m1.xlarge)**

15.8.9.2.5. 선택적 RHOSP 구성 매개변수

선택적 RHOSP 구성 매개변수는 다음 표에 설명되어 있습니다.

표 15.40. 선택적 RHOSP 매개변수

매개변수	설명	값
compute.platform.openstack.additionalNetworkIDs	컴퓨팅 시스템과 관련된 추가 네트워크입니다. 추가 네트워크에는 허용되는 주소 쌍이 생성되지 않습니다.	하나 이상의 UUID 목록을 문자열로 나타냅니다. 예: **fa806b2f-ac49-4bce-b9db-124bc64209bf**
compute.platform.openstack.additionalSecurityGroupIDs	컴퓨팅 시스템과 관련된 추가 보안 그룹입니다.	하나 이상의 UUID 목록을 문자열로 나타냅니다. 예: **7ee219f3-d2e9-48a1-96c2-e7429f1b0da7**

매개변수	설명	값
compute.platform.openstack.zones	머신을 설치할 RHOSP Compute (Nova) 가용성 영역 (AZ). 이 매개 변수가 설정되지 않은 경우 설치 프로그램은 RHOSP 관리자가 구성한 Nova의 기본 설정을 사용합니다. Kuryr를 사용하는 클러스터에서 RHOSP Octavia는 가용성 영역을 지원하지 않습니다. 로드 밸런서 및 Amphora 공급자 드라이버를 사용하는 경우 Amphora VM에 의존하는 OpenShift Container Platform 서비스는이 속성의 값에 따라 생성되지 않습니다.	문자열 목록입니다. 예: **["zone-1", "zone-2"]**.
compute.platform.openstack.rootVolume.zones	컴퓨팅 시스템의 경우 루트 볼륨을 설치할 가용성 영역입니다. 이 매개변수의 값을 설정하지 않으면 설치 프로그램에서 기본 가용성 영역을 선택합니다.	문자열 목록(예: **["zone-1", "zone-2"]**).
controlPlane.platform.openstack.additionalNetworkIDs	컨트롤 플레인 시스템과 관련된 추가 네트워크입니다. 추가 네트워크에는 허용되는 주소 쌍이 생성되지 않습니다.	하나 이상의 UUID 목록을 문자열로 나타냅니다. 예: **fa806b2f-ac49-4bce-b9db-124bc64209bf**
controlPlane.platform.openstack.additionalSecurityGroupIDs	컨트롤 플레인 시스템과 관련된 추가 보안 그룹입니다.	하나 이상의 UUID 목록을 문자열로 나타냅니다. 예: **7ee219f3-d2e9-48a1-96c2-e7429f1b0da7**
controlPlane.platform.openstack.zones	머신을 설치할 RHOSP Compute (Nova) 가용성 영역 (AZ). 이 매개 변수가 설정되지 않은 경우 설치 프로그램은 RHOSP 관리자가 구성한 Nova의 기본 설정을 사용합니다. Kuryr를 사용하는 클러스터에서 RHOSP Octavia는 가용성 영역을 지원하지 않습니다. 로드 밸런서 및 Amphora 공급자 드라이버를 사용하는 경우 Amphora VM에 의존하는 OpenShift Container Platform 서비스는이 속성의 값에 따라 생성되지 않습니다.	문자열 목록입니다. 예: **["zone-1", "zone-2"]**.

매개변수	설명	값
controlPlane.platform.openstack.rootVolume.zones	컨트롤 플레인 시스템의 경우 루트 볼륨을 설치할 가용성 영역입니다. 이 값을 설정하지 않으면 설치 프로그램에서 기본 가용성 영역을 선택합니다.	문자열 목록(예: **["zone-1", "zone-2"]**).
platform.openstack.clusterOSImage	설치 프로그램이 RHCOS 이미지를 다운로드하는 위치입니다. 네트워크가 제한된 환경에서 설치를 수행하려면이 매개 변수를 설정해야 합니다.	HTTP 또는 HTTPS URL (선택 옵션으로 SHA-256 형식의 체크섬을 사용) 예: **http://mirror.example.com/images/rhcos-43.81.201912131630.0-openstack.x86_64.qcow2.gz?sha256=ffebbd68e8a1f2a245ca19522c16c86f67f9ac8e4e0c1f0a812b068b16f7265d**. 이 값은 기존의 Glance 이미지의 이름이 될 수도 있습니다 (예: **my-rhcos**).
platform.openstack.clusterOSImageProperties	Glance의 설치 프로그램에서 업로드된 ClusterOSImage에 추가할 속성입니다. **platform.openstack.clusterOSImage**가 기존 Glance 이미지로 설정된 경우 이 속성은 무시됩니다. 이 속성을 사용하여 노드당 26 PV의 RHOSP의 기본 PV(영구 볼륨) 제한을 초과할 수 있습니다. 제한을 초과하려면 **hw_scsi_model** 속성값을 **virtio-scsi**로, **hw_disk_bus** 값을 **scsi**로 설정합니다. 이 속성을 사용하면 **yes** 값이 있는 **hw_qemu_guest_agent** 속성을 포함하여 QEMU 게스트 에이전트를 활성화할 수도 있습니다.	키-값 문자열 쌍 목록입니다. 예를 들면 **["hw_scsi_model": "virtio-scsi", "hw_disk_bus": "scsi"]**가 있습니다.

platform.openstack.defaultMachinePlatform	기본 시스템 풀 플랫폼 구성입니다.	<pre>{
		  "type": "ml.large",
		  "rootVolume": {
		  "size": 30,
		  "type": "performance"
		  }
		}</pre>

매개변수	설명	값
platform.openstack.ingressFloatingIP	Ingress 포트와 연결할 기존 부동 IP 주소입니다. 이 속성을 사용하려면 **platform.openstack.externalNetwork** 속성도 정의해야 합니다.	IP 주소 (예: **128.0.0.1**)
platform.openstack.apiFloatingIP	로드 밸런서 API와 연결할 기존 부동 IP 주소입니다. 이 속성을 사용하려면 **platform.openstack.externalNetwork** 속성도 정의해야 합니다.	IP 주소 (예: **128.0.0.1**)
platform.openstack.externalDNS	클러스터 인스턴스가 DNS 확인에 사용하는 외부 DNS 서버의 IP 주소입니다.	IP 주소 목록을 문자열로 나타냅니다. 예: **["8.8.8.8", "192.168.1.12"]**.
platform.openstack.machinesSubnet	클러스터 노드가 사용하는 RHOSP 서브넷의 UUID입니다. 이 서브넷에 노드와 가상 IP(VIP) 포트가 생성됩니다. **networking.machineNetwork**의 첫 번째 항목은 **machinesSubnet** 값과 일치해야 합니다. 사용자 지정 서브넷에 배포하는 경우 OpenShift Container Platform 설치 관리자에 외부 DNS 서버를 지정할 수 없습니다. 대신 RHOSP의 서브넷에 DNS를 추가합니다.	문자열의 UUID입니다. 예: **fa806b2f-ac49-4bce-b9db-124bc64209bf**

15.8.9.3. 제한된 OpenStack 설치를 위한 사용자 정의 install-config.yaml 파일 샘플

이 샘플 install-config.yaml은 가능한 모든 RHOSP(Red Hat OpenStack Platform) 사용자 지정 옵션을 보여줍니다.

중요

이 샘플 파일은 참조용으로만 제공됩니다. install-config.yaml 파일은 설치 프로그램을 사용하여 받아야 합니다.

```
apiVersion: v1
```

```
baseDomain: example.com
controlPlane:
  name: master
  platform: {}
  replicas: 3
compute:
- name: worker
  platform:
    openstack:
      type: ml.large
  replicas: 3
metadata:
  name: example
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  serviceNetwork:
  - 172.30.0.0/16
  networkType: OpenShiftSDN
platform:
  openstack:
    region: region1
    cloud: mycloud
    externalNetwork: external
    computeFlavor: m1.xlarge
    apiFloatingIP: 128.0.0.1
fips: false
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
additionalTrustBundle: |

  -----BEGIN CERTIFICATE-----

  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ

  -----END CERTIFICATE-----

imageContentSources:
- mirrors:
  - <mirror_registry>/<repo_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <mirror_registry>/<repo_name>/release
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

15.8.10. 컴퓨팅 머신 유사성 설정

선택적으로 설치하는 동안 컴퓨팅 시스템의 선호도 정책을 설정할 수 있습니다. 설치 프로그램은 기본적으로 컴퓨팅 시스템의 선호도 정책을 선택하지 않습니다.

설치 후 특정 RHOSP 서버 그룹을 사용하는 머신 세트를 생성할 수도 있습니다.

참고

컨트롤 플레인 시스템은 **soft-anti-affinity** 정책을 사용하여 생성됩니다.

작은 정보

RHOSP 인스턴스 스케줄링 및 배치에 대한 자세한 내용은 RHOSP 설명서에서 확인할 수 있습니다.

사전 요구 사항

- **install-config.yaml** 파일을 생성하고 수정합니다.

절차

1. RHOSP 명령줄 인터페이스를 사용하여 컴퓨팅 시스템의 서버 그룹을 생성합니다. 예를 들면 다음과 같습니다.

```
$ openstack \
  --os-compute-api-version=2.15 \
  server group create \
  --policy anti-affinity \
  my-openshift-worker-group
```

자세한 내용은 서버 그룹 생성 명령 설명서를 참조하십시오.

2. 설치 프로그램이 포함된 디렉터리로 변경하고 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory>
```

다음과 같습니다.

installation_directory

클러스터의 **install-config.yaml** 파일이 포함된 디렉토리의 이름을 지정합니다.

3. **MachineSet 정의 파일인 manifests/99_openshift-cluster-api_worker-machineset-0.yaml 을 엽니다.**

4. **spec.template.spec.providerSpec.value 속성 아래에 있는 정의에 속성 serverGroupID 를 추가합니다. 예를 들면 다음과 같습니다.**

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_ID>
    machine.openshift.io/cluster-api-machine-role: <node_role>
    machine.openshift.io/cluster-api-machine-type: <node_role>
  name: <infrastructure_ID>-<node_role>
  namespace: openshift-machine-api
spec:
  replicas: <number_of_replicas>
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_ID>
      machine.openshift.io/cluster-api-machineset: <infrastructure_ID>-<node_role>
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_ID>
        machine.openshift.io/cluster-api-machine-role: <node_role>
        machine.openshift.io/cluster-api-machine-type: <node_role>
        machine.openshift.io/cluster-api-machineset: <infrastructure_ID>-<node_role>
    spec:
      providerSpec:
        value:
          apiVersion: openstackproviderconfig.openshift.io/v1alpha1
          cloudName: openstack
          cloudsSecret:
            name: openstack-cloud-credentials
            namespace: openshift-machine-api
          flavor: <nova_flavor>
          image: <glance_image_name_or_location>
          serverGroupID: aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee (1)
          kind: OpenstackProviderSpec
          networks:
          - filter: {}
            subnets:
            - filter:
                name: <subnet_name>
                tags: openshiftClusterID=<infrastructure_ID>
          securityGroups:
          - filter: {}
            name: <infrastructure_ID>-<node_role>
          serverMetadata:
            Name: <infrastructure_ID>-<node_role>
            openshiftClusterID: <infrastructure_ID>
```

```
  tags:
  - openshiftClusterID=<infrastructure_ID>
  trunk: true
  userDataSecret:
    name: <node_role>-user-data
  availabilityZone: <optional_openstack_availability_zone>
```

1

서버 그룹의 UUID를 여기에 추가합니다.

5. 선택 사항: manifests/99_openshift-cluster-api_worker-machineset-0.yaml 파일을 백업합니다. 설치 프로그램은 클러스터를 생성할 때 manifests/ 디렉터리를 삭제합니다.

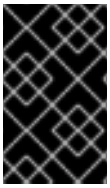
클러스터를 설치할 때 설치 프로그램은 수정한 MachineSet 정의를 사용하여 RHOSP 서버 그룹 내에서 컴퓨팅 머신을 생성합니다.

15.8.11. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 core 사용자의 ~/.ssh/authorized_keys 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 core로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. ./openshift-install gather 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.



중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

참고

AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

절차

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: ~/.ssh/id_ed25519)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 '~/.ssh 디렉터리에 있는지 확인하십시오.

참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 ed25519 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 rsa 또는 ecdsa 알고리즘을 사용하는 키를 생성합니다.

2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 ~/.ssh/id_ed25519.pub 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 ./openshift-install gather 명령을 사용하려는 경우 필요합니다.

참고

일부 배포에서는 ~/.ssh/id_rsa 및 ~/.ssh/id_dsa와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. ssh-agent 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. ssh-agent에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: ~/.ssh/id_ed25519).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

15.8.12. 환경에 대한 액세스 활성화

배포 시 모든 OpenShift Container Platform 시스템은 RHOSP(Red Hat OpenStack Platform) 테넌트 네트워크에서 생성됩니다. 따라서 대부분의 RHOSP 배포에서 직접 액세스할 수 없습니다.

설치시 부동 IP 주소 (FIP)를 사용하여 OpenShift Container Platform API 및 애플리케이션의 액세스를 설정할 수 있습니다. FIP를 구성하지 않고 설치를 완료 할 수도 있지만 설치 프로그램은 외부에서 API 또는 애플리케이션에 액세스하는 방법을 설정하지 않습니다.

15.8.12.1. 부동 IP 주소로 액세스 활성화

OpenShift Container Platform API 및 클러스터 애플리케이션에 대한 외부 액세스 용으로 유동 IP (FIP) 주소를 생성합니다.

절차

1. RHOSP(Red Hat OpenStack Platform) CLI를 사용하여 API FIP를 생성합니다.

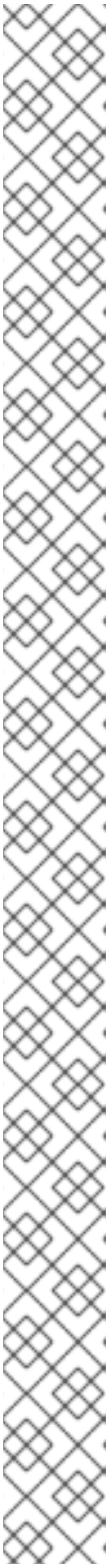
```
$ openstack floating ip create --description "API <cluster_name>.<base_domain>"
<external_network>
```

2. RHOSP(Red Hat OpenStack Platform) CLI를 사용하여 앱 또는 Ingress, FIP를 생성합니다.

```
$ openstack floating ip create --description "Ingress <cluster_name>.<base_domain>"
<external_network>
```

3. API 및 Ingress FIP의 DNS 서버에 이러한 패턴에 따라 레코드를 추가합니다.

```
api.<cluster_name>.<base_domain>.  IN  A  <API_FIP>
*.apps.<cluster_name>.<base_domain>. IN  A <apps_FIP>
```



참고

DNS 서버를 제어하지 않으면 /etc/hosts 파일에 다음과 같은 클러스터 도메인 이름을 추가하여 클러스터에 액세스할 수 있습니다.

- <api_floating_ip> api.<cluster_name>.<base_domain>
- <application_floating_ip> grafana-openshift-monitoring.apps.<cluster_name>.<base_domain>
- <application_floating_ip> prometheus-k8s-openshift-monitoring.apps.<cluster_name>.<base_domain>
- <application_floating_ip> oauth-openshift.apps.<cluster_name>.<base_domain>
- <application_floating_ip> console-openshift-console.apps.<cluster_name>.<base_domain>
- application_floating_ip integrated-oauth-server-openshift-authentication.apps.<cluster_name>.<base_domain>

/etc/hosts 파일의 클러스터 도메인 이름은 웹 콘솔과 클러스터의 모니터링 인터페이스에 로컬로 액세스할 수 있는 권한을 부여합니다. kubectl 또는 oc 에서도 사용할 수 있습니다. <application_floating_ip>를 가리키는 추가 항목을 사용하여 사용자 애플리케이션에 액세스할 수 있습니다. 이 작업을 수행하면 사용자만 API 및 애플리케이션에 액세스할 수 있어 프로덕션 배포에는 적합하지 않지만 개발 및 테스트용 설치는 가능합니다.

4. FIP를 다음 매개 변수의 값으로 install-config.yaml 파일에 추가하십시오.
   - platform.openstack.ingressFloatingIP
   - platform.openstack.apiFloatingIP

이러한 값을 사용하는 경우 **install-config.yaml** 파일에서 **platform.openstack.externalNetwork** 매개 변수 값으로 외부 네트워크를 입력해야 합니다.

작은 정보

부동 **IP** 주소를 할당하고 방화벽 구성을 업데이트하여 클러스터 외부에서 **OpenShift Container Platform** 리소스를 사용할 수 있습니다.

15.8.12.2. 유동 IP 주소없이 설치 완료

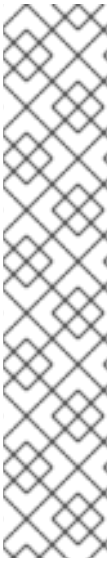
유동 **IP** 주소를 지정하지 않고도 **Red Hat OpenStack Platform (RHOSP)**에 **OpenShift Container Platform**을 설치할 수 있습니다.

install-config.yaml 파일에서 다음 매개 변수를 정의하지 마십시오.

- **platform.openstack.ingressFloatingIP**
- **platform.openstack.apiFloatingIP**

외부 네트워크를 제공 할 수 없는 경우 **platform.openstack.externalNetwork**를 비워 둘 수도 있습니다. **platform.openstack.externalNetwork** 값을 지정하지 않으면 라우터가 생성되지 않으며 추가 작업없이 설치 프로그램이 **Glance**에서 이미지를 검색하지 못합니다. 외부 연결을 직접 구성해야 합니다.

유동 **IP** 주소 또는 이름 확인 부족으로 인해 클러스터 **API**에 연결할 수 없는 시스템에서 설치 프로그램을 실행하면 설치에 실패합니다. 이러한 경우 설치 실패를 방지하기 위해 프록시 네트워크를 사용하거나 시스템과 동일한 네트워크에있는 시스템에서 설치 프로그램을 실행할 수 있습니다.



참고

API 및 Ingress 포트의 DNS 레코드를 생성하여 이름 확인을 활성화할 수 있습니다. 예를 들면 다음과 같습니다.

```
api.<cluster_name>.<base_domain>.  IN  A  <api_port_IP>
*.apps.<cluster_name>.<base_domain>. IN  A <ingress_port_IP>
```

DNS 서버를 제어하지 않으면 /etc/hosts 파일에 레코드를 추가할 수 있습니다. 이 작업을 수행하면 사용자만 API에 액세스할 수 있어 프로덕션 배포에는 적합하지 않지만 개발 및 테스트용 설치는 가능합니다.

15.8.13. 클러스터 배포

호환되는 클라우드 플랫폼에 OpenShift Container Platform을 설치할 수 있습니다.

중요

최초 설치 과정에서 설치 프로그램의 create cluster 명령을 한 번만 실행할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

절차

1. 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터 배포를 초기화합니다.

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
  --log-level=info 2
```

1 <installation_directory> 값으로 사용자 지정한 ./install-config.yaml 파일의 위치를 지정합니다.

2 다른 설치 세부 사항을 보려면 info 대신 warn, debug 또는 error를 지정합니다.

참고

호스트에 구성된 클라우드 공급자 계정에 클러스터를 배포하기에 충분한 권한이 없는 경우, 설치 프로세스가 중단되고 누락된 권한을 알리는 메시지가 표시됩니다.

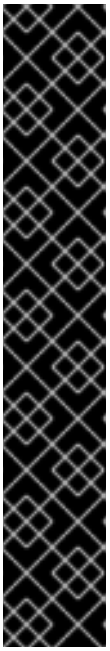
클러스터 배포가 완료되면 웹 콘솔로 연결되는 링크와 **kubeadmin** 사용자의 인증 정보가 포함된 클러스터 액세스 지침이 사용자 터미널에 표시됩니다.

출력 예

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-
ymBZj-Wt5AL"
INFO Time elapsed: 36m22s
```

참고

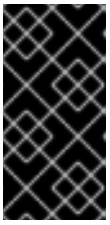
설치에 성공하면 클러스터 액세스 및 인증 정보도 **<installation_directory>/.openshift_install.log**로 출력됩니다.



중요

- 설치 프로그램에서 생성하는 **Ignition** 구성 파일에 **24**시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 **24**시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 **kubelet** 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(**CSR**)을 수동으로 승인해야 합니다. 자세한 내용은 ***Recovering from expired control plane certificates*** 문서를 참조하십시오.

- 클러스터를 설치한 후 **24**시간에서 **22**시간까지의 인증서가 교체되기 때문에 생성된 후 **12**시간 이내에 **Ignition** 구성 파일을 사용하는 것이 좋습니다. **12**시간 이내에 **Ignition** 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.



중요

설치 프로그램에서 생성되는 파일이나 설치 프로그램을 삭제해서는 안 됩니다. 클러스터를 삭제하려면 두 가지가 모두 필요합니다.

15.8.14. 클러스터 상태 확인

설치 중 또는 설치 후 **OpenShift Container Platform** 클러스터의 상태를 확인할 수 있습니다.

절차

1. 클러스터 환경에서 관리자의 **kubeconfig** 파일을 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

kubeconfig 파일에는 **CLI**에서 올바른 클러스터 및 **API** 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다.

2. 배포 후 생성된 컨트롤 플레인 및 컴퓨팅 시스템을 확인합니다.

```
$ oc get nodes
```

3. 클러스터 버전을 확인합니다.

```
$ oc get clusterversion
```

4. Operator 상태를 확인합니다.

```
$ oc get clusteroperator
```

5. 클러스터에서 실행 중인 모든 Pod를 확인합니다.

```
$ oc get pods -A
```

15.8.15. CLI를 사용하여 클러스터에 로그인

클러스터 kubeconfig 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. kubeconfig 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- oc CLI를 설치했습니다.

절차

1. kubeadmin 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1. <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 **oc** 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

추가 리소스

- **OpenShift Container Platform** 웹 콘솔 액세스 및 이해에 대한 자세한 내용은 웹 콘솔에 액세스를 참조하십시오.

15.8.16. 기본 OperatorHub 소스 비활성화

Red Hat 및 커뮤니티 프로젝트에서 제공하는 콘텐츠를 소싱하는 **Operator** 카탈로그는 **OpenShift Container Platform**을 설치하는 동안 기본적으로 **OperatorHub**용으로 구성됩니다. 제한된 네트워크 환경에서는 클러스터 관리자로서 기본 카탈로그를 비활성화해야 합니다.

절차

- **OperatorHub** 오브젝트에 **disableAllDefaultSources: true**를 추가하여 기본 카탈로그의 소스를 비활성화합니다.

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

작은 정보

또는 웹 콘솔을 사용하여 카탈로그 소스를 관리할 수 있습니다. 관리 → 클러스터 설정 → 구성 → **OperatorHub** 페이지에서 개별 소스를 생성, 삭제, 비활성화 및 활성화할 수 있는 소스 탭을 클릭합니다.

15.8.17. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기

위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

15.8.18. 다음 단계

- 클러스터를 사용자 지정합니다.
- 클러스터를 설치하는 데 사용한 미러 레지스트리에 신뢰할 수 있는 CA가 있는 경우 추가 신뢰 저장소를 구성하여 클러스터에 추가합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.
- Cluster Samples Operator 및 must-gather 툴의 이미지 스트림을 구성합니다.
- 제한된 네트워크에서 Operator Lifecycle Manager (OLM) 사용 방법에 대해 살펴봅니다.
- 유동 IP 주소를 통한 애플리케이션 트래픽을 허용하도록 RHOSP를 구성하지 않은 경우 유동 IP 주소로 RHOSP 액세스를 구성합니다.

15.9. OPENSTACK 클라우드 구성 참조 가이드

클라우드 공급자 구성은 OpenShift Container Platform이 RHOSP(Red Hat OpenStack Platform)와 상호 작용하는 방법을 제어합니다. cloud-provider 구성 매니페스트 파일에서 다음 매개변수를 사용하여 클러스터를 구성합니다.

15.9.1. OpenStack 클라우드 공급자 옵션

일반적으로 **cloud.conf** 라는 파일로 저장된 클라우드 공급자 구성은 **OpenShift Container Platform** 이 **RHOSP(Red Hat OpenStack Platform)**와 상호 작용하는 방법을 제어합니다.

다음 옵션을 지정하여 유효한 **cloud.conf** 파일을 생성할 수 있습니다.

15.9.1.1. 글로벌 옵션

다음 옵션은 **Keystone**이라고도 하는 **RHOSP ID** 서비스를 사용하는 **RHOSP CCM** 인증에 사용됩니다. **openstack CLI**를 사용하여 설정할 수 있는 글로벌 옵션과 관련이 있습니다.

옵션	설명
auth-url	RHOSP ID 서비스 URL입니다. 예: **http://128.110.154.166/identity**.
ca-file	선택 사항: RHOSP ID 서비스와의 통신을 위한 CA 인증서 번들 파일입니다. ID 서비스 URL과 함께 HTTPS 프로토콜을 사용하는 경우 이 옵션이 필요합니다.
domain-id	ID 서비스 사용자 도메인 ID입니다. ID 서비스 애플리케이션 자격 증명을 사용하는 경우 이 옵션을 설정되지 않은 상태로 두십시오.
domain-name	ID 서비스 사용자 도메인 이름입니다. **domain-id** 를 설정하는 경우 이 옵션이 필요하지 않습니다.
tenant-id	ID 서비스 프로젝트 ID입니다. ID 서비스 애플리케이션 자격 증명을 사용하는 경우 이 옵션을 설정되지 않은 상태로 두십시오. ID API의 버전 3에서 식별자 **테넌트** 를 **프로젝트로** 변경하면 **tenant-id** 값이 API의 프로젝트 구성에 자동으로 매핑됩니다.
tenant-name	ID 서비스 프로젝트 이름입니다.
사용자 이름	ID 서비스 사용자 이름입니다. ID 서비스 애플리케이션 자격 증명을 사용하는 경우 이 옵션을 설정되지 않은 상태로 두십시오.

옵션	설명
암호	ID 서비스 사용자 암호입니다. ID 서비스 애플리케이션 자격 증명을 사용하는 경우 이 옵션을 설정되지 않은 상태로 두십시오.
region	ID 서비스 지역 이름입니다.
trust-id	ID 서비스 신뢰 ID입니다. 신뢰는 다른 사용자에게 역할을 위임하거나 신뢰할 수 있는 사용자 또는 신뢰자의 권한을 나타냅니다. 필요한 경우 신뢰는 신뢰자가 신뢰자에게 가장할 수 있도록 권한을 부여합니다. ID 서비스 API의 **/v3/OS-TRUST/trusts** 끝점을 쿼리하여 사용 가능한 신뢰를 찾을 수 있습니다.

15.9.1.2. 로드 밸런서 옵션

클라우드 공급자는 Octavia를 사용하는 배포에 대한 여러 로드 밸런서 옵션을 지원합니다.

옵션	설명
use-octavia	Neutron-LECDHE 대신 **LoadBalancer** 유형의 서비스 구현에 Octavia를 사용할지 여부입니다. 기본값은 **true**입니다.
floating-network-id	선택 사항: 로드 밸런서 가상 IP 주소(VIP)에 대한 유동 IP 주소를 생성하는 데 사용되는 외부 네트워크입니다. 클라우드에 외부 네트워크가 여러 개 있는 경우 이 옵션을 설정하거나 사용자가 서비스 주석에 **loadbalancer.openstack.org/floating-network-id** 를 지정해야 합니다.
lb-method	로드 밸런서 풀을 생성하는 데 사용되는 로드 밸런싱 알고리즘입니다. Amphora 공급자의 경우 값은 **ROUND_ROBIN**,**LEAST_CONNECTIONS**, 또는 **SOURCE_IP** 일 수 있습니다. 기본값은 **ROUND_ROBIN** 입니다. OVN 공급자의 경우 **SOURCE_IP_PORT** 알고리즘만 지원됩니다. Amphora 공급자의 경우 **LEAST_CONNECTIONS** 또는 **SOURCE_IP** 방법을 사용하는 경우, **openshift-config** 네임스페이스의 **cloud-provider-config** 구성 맵에서 **create-monitor** 옵션을 **true** 로 구성하고 로드 밸런서 유형 서비스에서 **ETP:Local** 을 사용하여 클라이언트에서 서비스 엔드포인트 연결에 알고리즘 적용을 허용하도록 합니다.

옵션	설명
lb-provider	선택 사항: 로드 밸런서의 공급자를 지정하는 데 사용됩니다(예: **amphora** 또는 **octavia**). Amphora 및 Octavia 공급자만 지원됩니다.
lb-version	선택 사항: 로드 밸런서 API 버전입니다. **"v2"** 만 지원됩니다.
subnet-id	로드 밸런서 VIP가 생성되는 네트워킹 서비스 서브넷의 ID입니다.
create-monitor	서비스 로드 밸런서에 대한 상태 모니터를 생성할지 여부입니다. **externalTrafficPolicy: Local** 을 선언하는 서비스에는 상태 모니터가 필요합니다. 기본값은 **false**입니다. **ovn** 공급자와 함께 버전 17 이전의 RHOSP를 사용하는 경우 이 옵션은 지원되지 않습니다.
monitor-delay	프로브가 로드 밸런서의 멤버로 전송되는 시간(초)입니다. 기본값은 **5** 입니다.
monitor-max-retries	로드 밸런서 멤버의 작동 상태를 **ONLINE** 으로 변경하는 데 필요한 성공적인 점검 수입니다. 유효한 범위는 **1~10** 이며 기본값은 **1** 입니다.
monitor-timeout	모니터가 시간이 초과되기 전에 모니터가 백엔드에 연결하기 위해 대기하는 시간(초)입니다. 기본값은 **3**입니다.

15.9.1.3. 메타데이터 옵션

옵션	설명

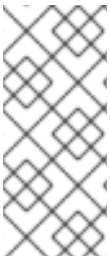
옵션	설명
search-order	이 구성 키는 공급자가 실행되는 인스턴스와 관련된 메타데이터를 검색하는 방식에 영향을 미칩니다. **configDrive,metadataService** 의 기본값은 공급자에서 사용 가능한 경우 먼저 구성 드라이브에서 인스턴스 메타데이터를 검색한 다음 메타데이터 서비스를 검색합니다. 대체 값은 다음과 같습니다. ● **configDrive**: 구성 드라이브에서 인스턴스 메타데이터만 검색합니다. ● **metadataService**: 메타데이터 서비스에서 인스턴스 메타데이터만 검색합니다. ● **metadataService,configDrive**: 메타데이터 서비스에서 먼저 메타데이터 메타데이터를 검색한 다음 구성 드라이브에서 인스턴스 메타데이터를 검색합니다.

15.10. OPENSTACK의 클러스터 설치 제거

RHOSP(Red Hat OpenStack Platform)에 배포한 클러스터를 제거할 수 있습니다.

15.10.1. 설치 관리자가 프로비저닝한 인프라를 사용하는 클러스터 제거

클라우드에서 설치 관리자 프로비저닝 인프라를 사용하는 클러스터를 제거할 수 있습니다.



참고

설치 제거 후 특히 **UPI(User Provisioned Infrastructure)** 클러스터에서 제거되지 않은 리소스에 대해 클라우드 공급자를 확인합니다. 설치 관리자가 생성하지 않았거나 설치 프로그램이 액세스할 수 없는 리소스가 있을 수 있습니다.

사전 요구 사항

- 클러스터를 배포하는 데 사용한 설치 프로그램의 사본을 준비합니다.
- 클러스터를 만들 때 설치 프로그램에 의해 생성된 파일을 준비합니다.

절차

1.

클러스터를 설치하는 데 사용한 컴퓨터에 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행합니다.

```
$ ./openshift-install destroy cluster \
--dir <installation_directory> --log-level info 1 2
```

1

<installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2

다른 세부 사항을 보려면 info 대신 warn, debug 또는 error를 지정합니다.

참고

클러스터의 클러스터 정의 파일이 포함되어 있는 디렉터리를 지정해야 합니다. 설치 프로그램이 클러스터를 삭제하려면 이 디렉터리에 있는 metadata.json 파일이 필요합니다.

2. 선택사항: <installation_directory> 디렉터리와 OpenShift Container Platform 설치 프로그램을 삭제합니다.

15.11. 사용자 인프라의 RHOSP에서 클러스터 설치 제거

사용자 프로비저닝 인프라에서 RHOSP(Red Hat OpenStack Platform)에 배포한 클러스터를 제거할 수 있습니다.

15.11.1. 플레이북 종속 항목 다운로드

사용자 프로비저닝 인프라에서의 제거 프로세스를 간소화하는 Ansible 플레이북에는 몇 가지 Python 모듈이 필요합니다. 프로세스를 실행할 시스템에서 모듈 리포지토리를 추가하고 다운로드합니다.

참고

이 방법은 현재 Red Hat Enterprise Linux (RHEL) 8을 사용하는 것으로 가정합니다.

사전 요구 사항

- Python 3가 시스템에 설치되어 있습니다.

절차

1. 명령줄에서 리포지토리를 추가합니다.
   a. Red Hat Subscription Manager에 등록합니다.

```
$ sudo subscription-manager register # If not done already
```

   b. 최신 서브스크립션 데이터를 가져옵니다.

```
$ sudo subscription-manager attach --pool=$YOUR_POOLID # If not done already
```

   c. 현재 리포지토리를 비활성화합니다.

```
$ sudo subscription-manager repos --disable=* # If not done already
```

   d. 필요한 리포지토리를 추가합니다.

```
$ sudo subscription-manager repos \
  --enable=rhel-8-for-x86_64-baseos-rpms \
  --enable=openstack-16-tools-for-rhel-8-x86_64-rpms \
  --enable=ansible-2.9-for-rhel-8-x86_64-rpms \
  --enable=rhel-8-for-x86_64-appstream-rpms
```

2. 모듈을 설치합니다.

```
$ sudo yum install python3-openstackclient ansible python3-openstacksdk
```

3. python 명령이 python3를 가리키는지 확인합니다.

```
$ sudo alternatives --set python /usr/bin/python3
```

15.11.2. 사용자 인프라를 사용하는 RHOSP에서 클러스터 제거

사용자 인프라를 사용하는 RHOSP(Red Hat OpenStack Platform)에서 OpenShift Container Platform 클러스터를 제거할 수 있습니다. 제거 프로세스를 빠르게 완료하려면 Ansible 플레이북을 여러 개 실행합니다.

사전 요구 사항

- Python 3가 시스템에 설치되어 있습니다.
- "플레이북 종속 항목 다운로드"에서 모듈을 다운로드했습니다.
- 클러스터를 설치하는 데 사용한 플레이북이 있습니다.
- 해당 설치 플레이북에 수행한 변경 사항을 반영하도록 down- 접두사가 있는 플레이북을 수정하셨습니다. 예를 들어 bootstrap.yaml 파일의 변경 사항은 down-bootstrap.yaml 파일에 반영됩니다.
- 모든 플레이북은 공통 디렉터리에 있습니다.

절차

1. 명령줄에서 사용자가 다운로드한 플레이북을 실행합니다.

```
$ ansible-playbook -i inventory.yaml  \
 down-bootstrap.yaml      \
 down-control-plane.yaml  \
 down-compute-nodes.yaml  \
 down-load-balancers.yaml \
 down-network.yaml        \
 down-security-groups.yaml
```

2. OpenShift Container Platform 설치에 대한 DNS 레코드 변경 내용을 제거합니다.

OpenShift Container Platform이 인프라에서 제거되었습니다.

16장. RHV에 설치

16.1. RHV(RED HAT VIRTUALIZATION)에 설치 준비

16.1.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- RHV(Red Hat Virtualization)의 OpenShift Container Platform 지원 매트릭스에서 지원되는 버전 조합이 있습니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.

16.1.2. RHV에 OpenShift Container Platform을 설치할 방법 선택

설치 관리자 프로비저닝 또는 사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치할 수 있습니다. 기본 설치 유형은 설치 프로그램이 클러스터의 기본 인프라를 프로비저닝하는 설치 관리자 프로비저닝 인프라를 사용합니다. 제공하는 인프라에 OpenShift Container Platform도 설치할 수 있습니다. 설치 프로그램에서 프로비저닝한 인프라를 사용하지 않는 경우 클러스터 리소스를 직접 관리하고 유지보수해야 합니다.

설치 관리자 프로비저닝 및 사용자 프로비저닝 설치 프로세스에 대한 자세한 내용은 설치 프로세스를 참조하십시오.

16.1.2.1. 설치 관리자 프로비저닝 인프라를 사용하여 클러스터 설치

다음 방법 중 하나를 사용하여 OpenShift Container Platform 설치 프로그램에서 프로비저닝하는 RHOSP(Red Hat OpenStack Platform) 가상 머신에 클러스터를 설치할 수 있습니다.

- RHV에 클러스터 빠른 설치: OpenShift Container Platform 설치 프로그램에서 프로비저닝한 RHV 가상 머신에 OpenShift Container Platform의 빠른 설치를 수행할 수 있습니다.
- 사용자 지정으로 RHV에 클러스터 설치: 사용자 지정 OpenShift Container Platform 클러스터를 RHV의 설치 관리자 프로비저닝 게스트에 설치할 수 있습니다. 설치 프로그램을 통해 설치 단계에서 일부 사용자 지정을 적용할 수 있습니다. 다른 많은 사용자 정의 옵션은 설치 후사용할 수 있습니다.

16.1.2.2. 사용자 프로비저닝 인프라를 사용하여 클러스터 설치

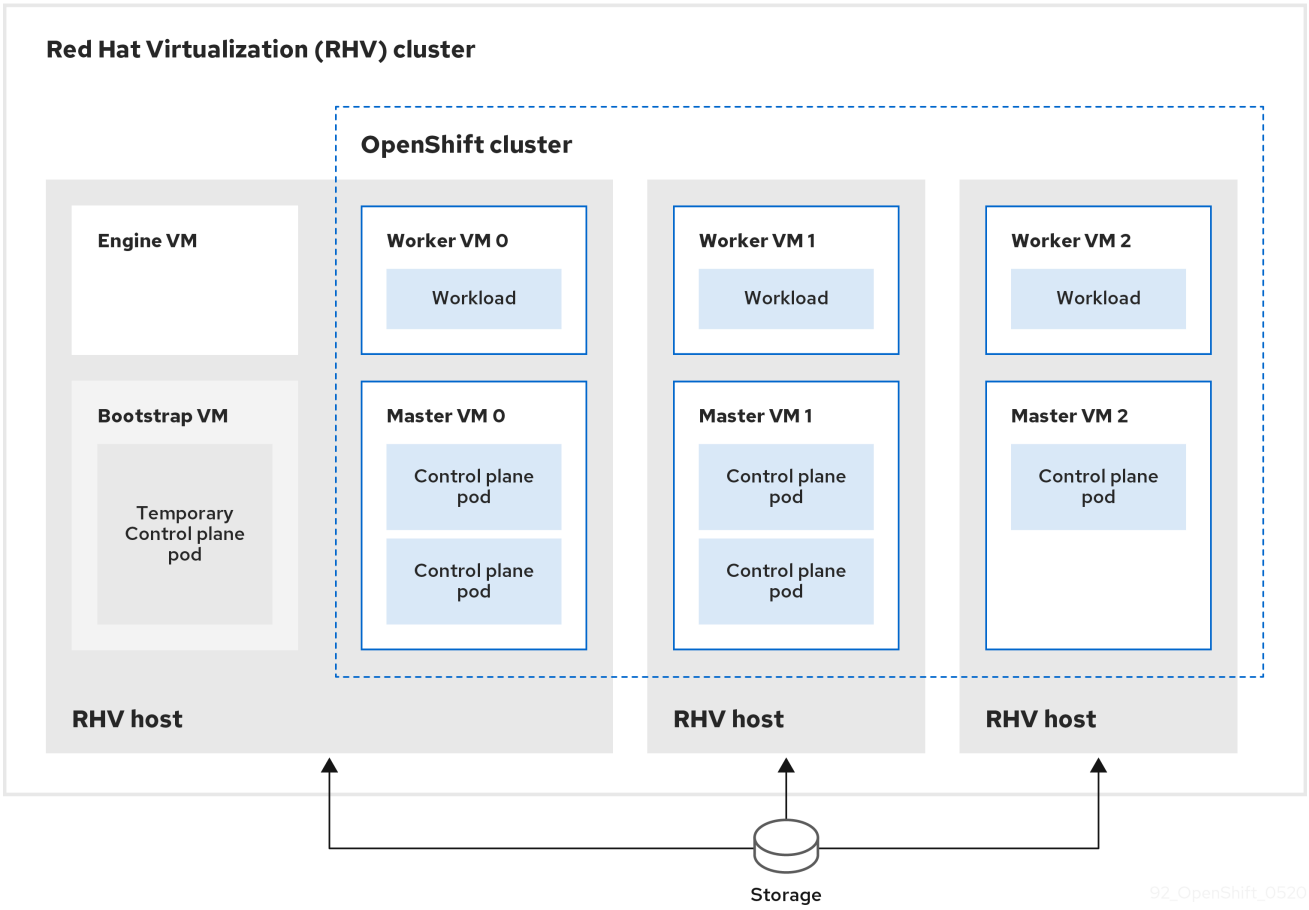
다음 방법 중 하나를 사용하여 프로비저닝하는 RHV 가상 머신에 클러스터를 설치할 수 있습니다.

- 사용자 프로비저닝 인프라를 사용하여 RHV에 클러스터 설치: 프로비저닝하는 RHV 가상 시스템에 OpenShift Container Platform을 설치할 수 있습니다. 제공된 Ansible 플레이북을 사용하여 설치를 지원할 수 있습니다.

- 제한된 네트워크의 RHV에 클러스터 설치: 설치 릴리스 컨텐츠의 내부 미러를 만들어 RHV에 OpenShift Container Platform을 제한되거나 연결이 끊긴 네트워크에 설치할 수 있습니다. 이 방법을 사용하여 소프트웨어 구성 요소를 받기 위해 활성 인터넷 연결이 필요하지 않은 사용자 프로비저닝 클러스터를 설치할 수 있습니다. 이 설치 방법을 사용하여 클러스터가 외부 콘텐츠에 대해 조직의 제어 조건을 충족하는 컨테이너 이미지만 사용하도록 할 수 있습니다.

16.2. RHV에 빠르게 클러스터 설치

다음 다이어그램에 표시된 것과 유사한 RHV(Red Hat Virtualization) 클러스터에 사용자 지정되지 않은 기본 OpenShift Container Platform 클러스터를 빠르게 설치할 수 있습니다.



설치 프로그램은 설치 관리자가 프로비저닝한 인프라를 사용하여 클러스터 생성 및 배포를 자동화합니다.

기본 클러스터를 설치하려면 환경을 준비하고 설치 프로그램을 실행한 후 프롬프트 내용을 따릅니다. 그러면 설치 프로그램이 **OpenShift Container Platform** 클러스터를 만듭니다.

기본 클러스터 설치 대안은 사용자 지정으로 클러스터 설치를 참조하십시오.

참고

이 설치 프로그램은 **Linux** 및 **macOS**에서만 사용할 수 있습니다.

16.2.1. 사전 요구 사항

- **OpenShift Container Platform** 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- **RHV(Red Hat Virtualization)**의 **OpenShift Container Platform** 지원 매트릭스에서 지원되는 버전 조합이 있습니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.
- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성해야 합니다.

16.2.2. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- **OpenShift Cluster Manager** 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 **Telemetry** 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.

- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.

중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

16.2.3. RHV 환경 요구사항

OpenShift Container Platform 버전 **4.9** 클러스터를 설치하고 실행하려면 **RHV** 환경이 다음 요구 사항을 충족해야 합니다.

이러한 요구 사항을 충족하지 않으면 설치 또는 프로세스가 실패할 수 있습니다. 또한 이러한 요구 사항을 충족하지 않으면 **OpenShift Container Platform** 클러스터가 설치 후 며칠 또는 몇 주에 실패할 수 있습니다.

CPU, 메모리, 스토리지에 대한 다음 요구사항은 설치 프로그램이 생성하는 기본 가상 시스템 수를 곱한 기본 값을 기반으로 합니다. 이러한 리소스는 **RHV** 환경에서 비 **OpenShift Container Platform** 작업에 사용하는 리소스와함께 사용할 수 있어야 합니다.

설치 프로그램은 기본적으로 설치 프로세스 중에 일곱 개의 가상 시스템을 생성합니다. 먼저 나머지 **OpenShift Container Platform** 클러스터를 생성하는 동안 임시 서비스와 컨트롤 플레인 영역을 제공하는 부트스트랩 가상 머신을 생성합니다. 설치 프로그램이 클러스터 생성을 완료하면 부트스트랩 시스템을 삭제하여 리소스를 확보합니다.

RHV 환경에서 가상 머신 수를 늘리면 그에 따라 리소스를 늘려야합니다.

요구사항

- **RHV** 버전은 **4.4**입니다.

- RHV 환경에는 상태가 Up인 데이터 센터가 하나 있습니다.
- RHV 데이터 센터에는 RHV 클러스터가 포함되어 있습니다.
- RHV 클러스터에는 다음과 같은 OpenShift Container Platform 클러스터 전용 리소스가 있습니다.
  - 최소 28개의 vCPU(설치 중 생성된 일곱 개의 가상 시스템마다 각각 4개)
  - 다음을 포함한 112GiB RAM 이상
    - 임시 컨트롤 플레인을 제공하는 부트스트랩 시스템의 경우 16GiB 이상
    - 컨트롤 플레인을 제공하는 컨트롤 플레인 시스템 세 개 각각에 대해 16GiB 이상
    - 애플리케이션 워크로드를 실행하는 컴퓨팅 시스템 세 개 각각에 대해 16GiB 이상
- RHV 스토리지 도메인은 이러한 etcd 백엔드 성능 요구사항을 충족해야 합니다.
- 선호도 그룹 지원의 경우: RHV 클러스터에서 호스트를 3개 이상 지원합니다. 필요한 경우 선호도 그룹을 비활성화할 수 있습니다. 자세한 내용은 *사용자 지정을 사용하여 RHV에 클러스터 설치*에서 *프로덕션 환경 외 랩 설정에 대한 모든 선호도 그룹 제거*를 참조하십시오.
- 프로덕션 환경에서 각 가상 머신은 120GiB 이상이어야 합니다. 따라서 스토리지 도메인은 기본 OpenShift Container Platform 클러스터에 대해 840GiB 이상을 제공해야 합니다. 리소스가 제한적인 환경이나 프로덕션 이외의 환경에서는 각 가상 시스템에 32GiB 이상이 있어야 하므로 스토리지 도메인에는 기본 OpenShift Container Platform 클러스터에 필요한 230GiB 이상이 있어야 합니다.
- 설치 및 업데이트 중에 Red Hat Ecosystem Catalog에서 이미지를 다운로드하려면 RHV 클러스터가 인터넷 연결에 액세스할 수 있어야 합니다. Telemetry 서비스에는 서브스크립션 및 권한 부여 프로세스를 단순화하기 위해 인터넷 연결이 필요합니다.
-

RHV 클러스터에는 RHV Manager의 REST API에 액세스할 수 있는 가상 네트워크가 있어야 합니다. 설치 관리자가 생성한 VM에서 DHCP를 사용하여 IP 주소를 얻을 수 있으므로 이 네트워크에서 DHCP가 활성화되어 있는지 확인합니다.

- 대상 RHV 클러스터에서 OpenShift Container Platform 클러스터를 설치 및 관리하기 위한 다음과 같은 최소 권한이있는 사용자 계정 및 그룹:
  - DiskOperator
  - DiskCreator
  - UserTemplateBasedVm
  - TemplateOwner
  - TemplateCreator
  - 대상 클러스터의 ClusterAdmin

주의

최소 권한 원칙을 적용합니다. 설치 과정에서 RHV에 대한 SuperUser 권한이 있는 관리자 계정을 사용하지 않도록 합니다. 설치 프로그램은 사용자가 제공한 인증 정보를 손상된 임시 ovirt-config.yaml 파일에 저장합니다.

추가 리소스

- 예: 비 프로덕션 랩 설정에 대한 모든 선호도 그룹 제거.

16.2.4. RHV 환경에 대한 요구사항 확인

RHV 환경이 OpenShift Container Platform 클러스터 설치 및 실행 요구사항을 충족하는지 확인합니다. 이러한 요구사항을 충족하지 않으면 실패할 수 있습니다.

중요

이러한 요구사항은 설치 프로그램이 컨트롤 플레인과 컴퓨팅 시스템을 생성하는 데 사용하는 기본 리소스를 기반으로 합니다. 이러한 리소스에는 vCPU, 메모리 및 스토리지가 포함됩니다. 이러한 리소스를 변경하거나 OpenShift Container Platform 시스템 수를 늘리는 경우에는 그에 따라 이 요구사항을 조정합니다.

프로세스

1. RHV 버전이 OpenShift Container Platform 버전 4.9의 설치를 지원하는지 확인합니다.

   a. RHV 관리 포털에서 오른쪽 상단에 있는 ? 도움말 아이콘을 클릭하고 정보를 선택합니다.

   b. 창이 열리면 RHV 소프트웨어 버전을 기록합니다.

   c. RHV 버전이 4.4인지 확인합니다. 지원되는 버전 조합에 대한 자세한 내용은 RHV의 OpenShift Container Platform에 대한 지원 매트릭스를 참조하십시오.

2. 데이터 센터, 클러스터 및 스토리지를 검사합니다.

   a. RHV 관리 포털에서 Compute → Data Centers를 클릭합니다.

   b. OpenShift Container Platform을 설치하려는 데이터 센터에 액세스할 수 있는지 확인합니다.

   c. 해당 데이터 센터의 이름을 클릭합니다.

   d. 데이터 센터 세부 사항의 스토리지 탭에서 OpenShift Container Platform을 설치하려는 스토리지 도메인이 활성인지 확인합니다.

e. 나중에 사용할 수 있도록 도메인 이름을 기록합니다.

f. 여유 공간이 **230GiB** 이상인지 확인합니다.

g. 스토리지 도메인이 **fio** 성능 벤치마킹 툴을 사용하여 측정할 수 있는 이러한 **etcd** 백엔드 성능 요구사항을 충족하는지 확인합니다.

h. 데이터 센터 세부 사항에서 클러스터 탭을 클릭합니다.

i. **OpenShift Container Platform**을 설치할 **RHV** 클러스터를 찾습니다. 나중에 사용할 수 있도록 클러스터 이름을 기록합니다.

3. **RHV** 호스트 리소스를 검사합니다.

a. **RHV** 관리 포털에서 컴퓨팅 > 클러스터를 클릭합니다.

b. **OpenShift Container Platform**을 설치할 클러스터를 클릭합니다.

c. 클러스터 세부 사항에서 호스트 탭을 클릭합니다.

d. 호스트를 검사하고 **OpenShift Container Platform** 클러스터 *전용*으로 사용할 수 있는 총 **28**개 이상의 논리 **CPU** 코어가 있는지 확인합니다.

e. 나중에 사용할 수 있도록 사용 가능한 논리 **CPU** 코어 수를 기록합니다.

f. 설치 중에 생성된 **7**개의 가상 시스템 각각에 **4**개의 코어가 있을 수 있도록 이러한 **CPU** 코어가 분산되어 있는지 확인합니다.

g. 다음 **OpenShift Container Platform** 시스템 각각에 대한 요구사항을 충족하기 위해 배포된 새 가상 시스템 예약에 필요한 **112GiB**의 최대 여유 메모리가 모두 호스트에 있는지 확인합니다.

- 부트스트랩 시스템에 필요한 16GiB
- 세 개의 컨트롤 플레인 시스템 각각에 필요한 16GiB
- 세 개의 컴퓨팅 시스템 각각에 대해 16GiB

h. 나중에 사용할 수 있도록 새 가상 시스템 예약에 필요한 최대 여유 메모리의 양을 기록합니다.

4. OpenShift Container Platform을 설치할 가상 네트워크가 RHV Manager의 REST API에 액세스할 수 있는지 확인합니다. 이 네트워크의 가상 시스템에서 RHV 관리자의 REST API에 도달하기 위해 curl을 사용합니다.

```
$ curl -k -u <username>@<profile>:<password> \ 1
https://<engine-fqdn>/ovirt-engine/api 2
```

1 <username> 의 경우 RHV에서 OpenShift Container Platform 클러스터를 만들고 관리할 수 있는 권한이 있는 RHV 계정의 사용자 이름을 지정합니다. <profile>은 로그인 프로파일을 지정합니다(RHV 관리 포털 로그인 페이지로 이동하여 프로파일 드롭다운 목록 검토). <password>의 경우 해당 사용자 이름에 대한 암호를 지정합니다.

2 <engine-fqdn>은 RHV 환경의 정규화된 도메인 이름을 지정합니다.

예를 들면 다음과 같습니다.

```
$ curl -k -u ocpadmin@internal:pw123 \
https://rhv-env.virtlab.example.com/ovirt-engine/api
```

16.2.5. RHV에서 네트워크 환경 준비

OpenShift Container Platform 클러스터의 고정 IP 주소 두 개를 구성하고 이 주소를 사용하여 DNS 항목을 만듭니다.

절차

1. 고정 IP 주소 두 개 예약

   a. OpenShift Container Platform을 설치하려는 네트워크에서 DHCP 임대 풀 외부에 있는 두 개의 고정 IP 주소를 식별합니다.

   b. 이 네트워크의 호스트에 연결하고 각 IP 주소를 사용하지 않는지 확인합니다. 예를 들어 Address Resolution Protocol (ARP)를 사용하여 IP 주소에 항목이 없는지 확인합니다.

   ```
   $ arp 10.35.1.19
   ```

   출력 예

   ```
   10.35.1.19 (10.35.1.19) -- no entry
   ```

   c. 네트워크 환경에 대한 표준 방식에 따라 두 개의 고정 IP 주소를 예약합니다.

   d. 나중에 참조할 수 있도록 이 IP 주소를 기록합니다.

2. 이 형식을 사용하여 OpenShift Container Platform REST API 및 앱 도메인 이름에 대한 DNS 항목을 만듭니다.

   ```
   api.<cluster-name>.<base-domain>   <ip-address> 1
   *.apps.<cluster-name>.<base-domain>   <ip-address> 2
   ```

   1 <cluster-name>, <base-domain>, <ip-address>는 각각 OpenShift Container Platform API의 클러스터 이름, 기본 도메인, 고정 IP 주소를 지정합니다.

   2 인그레스 및 로드 밸런서 용 OpenShift Container Platform 앱의 클러스터 이름, 기본 도메인, 고정 IP 주소를 지정합니다.

예를 들면 다음과 같습니다.

```
api.my-cluster.virtlab.example.com 10.35.1.19
*.apps.my-cluster.virtlab.example.com 10.35.1.20
```

16.2.6. 비보안 모드에서 RHV에 OpenShift Container Platform 설치

기본적으로 설치 관리자는 CA 인증서를 생성하고 확인을 요청한 후 설치 중에 사용할 인증서를 저장합니다. 수동으로 생성하거나 설치할 필요가 없습니다.

권장되지는 않지만 비보안 모드로 OpenShift Container Platform을 설치하여 인증서를 확인하지 않고도 이 기능을 재정의하고 OpenShift Container Platform을 설치할 수 있습니다.

주의

잠재적인 공격자가 중Man-in-the-Middle 공격을 수행할 수 있고 네트워크에서 중요한 인증 정보를 캡처할 수 있으므로 비보안 모드에 설치하는 것은 권장되지 않습니다.

절차

1. ~/.ovirt/ovirt-config.yaml이라는 파일을 생성합니다.

2. ovirt-config.yaml에 다음 내용을 추가합니다.

```
ovirt_url: https://ovirt.example.com/ovirt-engine/api (1)
ovirt_fqdn: ovirt.example.com (2)
ovirt_pem_url: ""
ovirt_username: ocpadmin@internal
ovirt_password: super-secret-password (3)
ovirt_insecure: true
```

(1) oVirt 엔진의 호스트 이름 또는 주소를 지정합니다.

2

oVirt 엔진의 정규화된 도메인 이름을 지정합니다.

3

oVirt 엔진의 admin 암호를 지정합니다.

3. 설치 프로그램을 실행합니다.

16.2.7. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 core 사용자의 ~/.ssh/authorized_keys 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 core로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. ./openshift-install gather 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

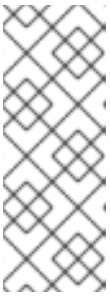
절차

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1

새 SSH 키의 경로 및 파일 이름(예: ~/.ssh/id_ed25519)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 '~/.ssh 디렉터리에 있는지 확인하십시오.



참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 ed25519 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 rsa 또는 ecdsa 알고리즘을 사용하는 키를 생성합니다.

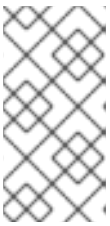
2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 ~/.ssh/id_ed25519.pub 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 ./openshift-install gather 명령을 사용하려는 경우 필요합니다.



참고

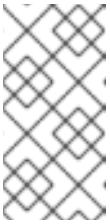
일부 배포에서는 ~/.ssh/id_rsa 및 ~/.ssh/id_dsa와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. ssh-agent 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```



참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. ssh-agent에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: ~/.ssh/id_ed25519).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

16.2.8. 설치 프로그램 받기

OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

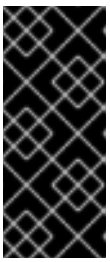
- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

절차

1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.

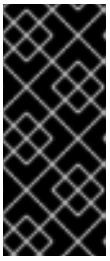
2. 인프라 공급자를 선택합니다.

3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.



중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.



중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 **OpenShift Container Platform** 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 **Linux** 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 **OpenShift Container Platform** 구성 요소에 대한 컨테이너 이미지를 제공하는 **Quay.io**를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

16.2.9. 클러스터 배포

호환되는 클라우드 플랫폼에 **OpenShift Container Platform**을 설치할 수 있습니다.

중요

최초 설치 과정에서 설치 프로그램의 **create cluster** 명령을 한 번만 실행할 수 있습니다.

사전 요구 사항

- 설치 프로그램을 실행하는 시스템에서 **Manager**에 대한 **ovirt-imageio** 포트를 엽니다. 기본적으로 포트는 **54322**입니다.

- **OpenShift Container Platform** 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

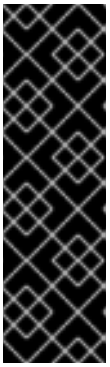
절차

1. 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터 배포를 초기화합니다.

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
  --log-level=info 2
```

1 **<installation_directory>**는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.

2 다른 설치 세부 사항을 보려면 **info** 대신 **warn, debug** 또는 **error**를 지정합니다.



중요

비어 있는 디렉터리를 지정합니다. 부트스트랩 **X.509** 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 **OpenShift Container Platform** 버전에서 설치 파일을 복사할 때는 주의하십시오.

설치 프로그램 메시지를 따릅니다.

a. 선택사항: **SSH Public Key**는 암호가 없는 공개 키(예: ~ / **.ssh** / **id_rsa.pub**)를 선택합니다. 이 키는 새로운 **OpenShift Container Platform** 클러스터와의 연결을 인증합니다.

참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 **OpenShift Container Platform** 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 **SSH** 키를 선택합니다.

b. **Platform**에 대해 **ovirt**를 선택합니다.

c. **Engine FQDN[:PORT]**은 **RHV** 환경의 정규화된 도메인 이름**(FQDN)**을 지정합니다.

예를 들면 다음과 같습니다.

```
rhv-env.virtlab.example.com:443
```

d. 설치 프로그램이 **CA** 인증서를 자동으로 생성합니다. **Would you like to use the above certificate to connect to the Manager?**에 대해서는 **y** 또는 **N**으로 응답합니다. **N**으로 응답하는 경우 비보안 모드로 **OpenShift Container Platform**을 설치해야 합니다.

e. **Engine username**은 다음 형식을 사용하여 **RHV** 관리자의 사용자 이름과 프로파일을 입력합니다.

```
<username>@<profile> 1
```

1

<username>은 **RHV** 관리자의 사용자 이름을 지정합니다. **<profile>**은 로그인 프로파일을 지정합니다**(RHV** 관리 포털 로그인 페이지로 이동하여 프로파일 드롭다운 목록 검토**). 예: admin@internal**

f. **Engine password**는 **RHV** 관리자 암호를 입력합니다.

g. **Cluster**는 **OpenShift Container Platform**을 설치할 **RHV** 클러스터를 선택합니다.

h. **Storage domain**은 **OpenShift Container Platform**을 설치할 스토리지 도메인을 선택합니다.

i. **Network**는 **RHV Manager REST API**에 액세스할 수있는 가상 네트워크를 선택합니다.

j. **Internal API Virtual IP**는 클러스터의 **REST API**에 대해 별도로 설정한 고정 **IP** 주소를 입력합니다.

k. **Ingress virtual IP**는 와일드카드 앱 도메인용으로 예약한 고정 **IP** 주소를 입력합니다.

l. **Base Domain**은 **OpenShift Container Platform** 클러스터의 기본 도메인을 입력합니다. 이 클러스터가 외부에 노출된 경우 **DNS** 인프라에서 인식할 수 있는 유효한 도메인이어야 합니다. 예를 들어 **virtlab.example.com**을 입력합니다.

m. **Cluster Name**은 클러스터 이름을 입력합니다. 예: **myca.crt OpenShift Container Platform REST API** 및 앱 도메인 이름에 대해 생성한 외부 등록/확인 가능 **DNS** 항목에서 클러스터 이름을 사용합니다. 설치 프로그램은 **RHV** 환경의 클러스터에도 이 이름을 지정합니다.

n. **Pull Secret**은 이전에 다운로드한 **pull-secret.txt** 파일에서 풀 시크릿을 복사하여 여기에 붙여 넣습니다. **Red Hat OpenShift Cluster Manager**에서 동일한 풀 시크릿 의 사본을 가져올 수도 있습니다.

참고

호스트에 구성된 클라우드 공급자 계정에 클러스터를 배포하기에 충분한 권한이 없는 경우, 설치 프로세스가 중단되고 누락된 권한을 알리는 메시지가 표시됩니다.

클러스터 배포가 완료되면 웹 콘솔로 연결되는 링크와 **kubeadmin** 사용자의 인증 정보가 포함된 클러스터 액세스 지침이 사용자 터미널에 표시됩니다.

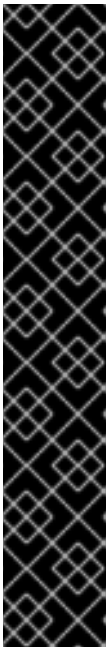
출력 예

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
```

```
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-Wt5AL"
INFO Time elapsed: 36m22s
```

참고

설치에 성공하면 클러스터 액세스 및 인증 정보도 **<installation_directory>/.openshift_install.log**로 출력됩니다.



중요

- 설치 프로그램에서 생성하는 **Ignition** 구성 파일에 **24**시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 **24**시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 **kubelet** 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(**CSR**)을 수동으로 승인해야 합니다. 자세한 내용은 ***Recovering from expired control plane certificates*** 문서를 참조하십시오.
- 클러스터를 설치한 후 **24**시간에서 **22**시간까지의 인증서가 교체되기 때문에 생성된 후 **12**시간 이내에 **Ignition** 구성 파일을 사용하는 것이 좋습니다. **12**시간 이내에 **Ignition** 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

중요

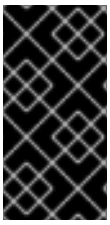
설치 프로그램에서 생성되는 파일이나 설치 프로그램을 삭제해서는 안 됩니다. 클러스터를 삭제하려면 두 가지가 모두 필요합니다.

중요

클러스터를 설치하는 데 필요한 단계를 완료했습니다. 나머지 단계는 클러스터를 확인하고 설치 문제를 해결하는 방법을 보여줍니다.

16.2.10. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 **OpenShift Container Platform**과 상호 작용하기 위해 **OpenShift CLI(oc)**를 설치할 수 있습니다. **Linux, Windows** 또는 **macOS**에 **oc**를 설치할 수 있습니다.



중요

이전 버전의 oc를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 oc를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 Linux에서 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.
2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. OpenShift v4.9 Linux Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. oc 바이너리를 PATH에 있는 디렉터리에 배치합니다.

   PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치

다음 절차에 따라 Windows에 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.
2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. OpenShift v4.9 Windows Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.
4. ZIP 프로그램으로 아카이브의 압축을 풉니다.
5. oc 바이너리를 PATH에 있는 디렉터리로 이동합니다.

   PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치

다음 절차에 따라 macOS에서 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.
2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. OpenShift v4.9 MacOSX Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 해제하고 압축을 풉니다.

5. oc 바이너리 PATH의 디렉터리로 이동합니다.

   PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
$ oc <command>
```

자세한 내용은 OpenShift CLI 시작하기를 참조하십시오.

16.2.11. CLI를 사용하여 클러스터에 로그인

클러스터 kubeconfig 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. kubeconfig 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- oc CLI를 설치했습니다.

프로세스

1. kubeadmin 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1. <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 **oc** 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

추가 리소스

- **OpenShift Container Platform** 웹 콘솔 액세스 및 이해에 대한 자세한 내용은 웹 콘솔에 액세스를 참조하십시오.

16.2.12. 클러스터 상태 확인

설치 중 또는 설치 후 **OpenShift Container Platform** 클러스터의 상태를 확인할 수 있습니다.

프로세스

1. 클러스터 환경에서 관리자의 **kubeconfig** 파일을 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

kubeconfig 파일에는 **CLI**에서 올바른 클러스터 및 **API** 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다.

2. 배포 후 생성된 컨트롤 플레인 및 컴퓨팅 시스템을 확인합니다.

```
$ oc get nodes
```

3. 클러스터 버전을 확인합니다.

```
$ oc get clusterversion
```

4. Operator 상태를 확인합니다.

```
$ oc get clusteroperator
```

5. 클러스터에서 실행 중인 모든 Pod를 확인합니다.

```
$ oc get pods -A
```

문제 해결

설치에 실패하면 설치 프로그램이 시간 초과되고 오류 메시지가 표시됩니다. 자세한 내용은 설치 문제 해결을 참조하십시오.

16.2.13. RHV의 OpenShift Container Platform 웹 콘솔 액세스

OpenShift Container Platform 클러스터가 초기화된 후 OpenShift Container Platform 웹 콘솔에 로그인할 수 있습니다.

프로세스

1. 선택사항: RHV(Red Hat Virtualization) 관리 포털에서 컴퓨팅 → 클러스터를 엽니다.

2. 설치 프로그램이 가상 시스템을 생성하는지 확인합니다.

3. 설치 프로그램이 실행 중인 명령줄로 돌아갑니다. 설치 프로그램이 완료되면 OpenShift Container Platform 웹 콘솔에 로그인하기 위한 사용자 이름과 임시 암호가 표시됩니다.

4. 브라우저에서 OpenShift Container Platform 웹 콘솔의 URL을 입력합니다. URL 형식은 다음과 같습니다.

```
console-openshift-console.apps.<clustername>.<basedomain> 1
```

1 **<clustername>. <basedomain>**은 클러스터 이름과 기본 도메인을 지정합니다.

예를 들면 다음과 같습니다.

```
console-openshift-console.apps.my-cluster.virtlab.example.com
```

16.2.14. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 **Telemetry** 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 **Telemetry**가 자동으로 실행되고 **OpenShift Cluster Manager** 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 **OpenShift Cluster Manager**를 사용하여 자동으로 또는 **OpenShift Cluster Manager**를 사용하여 수동으로 유지 관리되는지 확인한 후 **subscription watch**를 사용하여 계정 또는 다중 클러스터 수준에서 **OpenShift Container Platform** 서브스크립션을 추적합니다.

추가 리소스

- **Telemetry** 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

16.2.15. RHV(Red Hat Virtualization)에 설치와 관련된 일반적인 문제 해결

다음은 발생할 수 있는 일반적인 문제와 가능한 원인 및 해결 방법입니다.

16.2.15.1. CPU 로드가 증가하고 노드가 Not Ready 상태가 됨

- 증상: **CPU** 로드가 크게 증가하고 노드가 **Not Ready** 상태가 되기 시작합니다.
- 원인: 특히 컨트롤 플레인 노드의 경우 스토리지 도메인 대기 시간이 너무 길 수 있습니다.
- 해결책:

kubelet 서비스를 다시 시작하여 노드를 다시 준비합니다.

```
$ systemctl restart kubelet
```

OpenShift Container Platform 메트릭 서비스를 검사하면 etcd 디스크 동기화 기간과 같은 중요한 데이터를 자동으로 수집하고 보고합니다. 클러스터가 작동 중인 경우 이 데이터를 사용하여 스토리지 대기 시간 또는 처리량이 근본적인 문제인지 여부를 판별합니다. 근본적인 문제가 맞다면 지연 시간이 짧고 처리량이 많은 스토리지 리소스를 사용합니다.

원시 메트릭을 가져오려면 kubeadmin 또는 cluster-admin 권한이 있는 사용자로 다음 명령을 입력합니다.

```
$ oc get --insecure-skip-tls-verify --server=https://localhost:<port> --raw=/metrics
```

자세한 내용은 Exploring Application Endpoints for the purposes of Debugging with OpenShift 4.x를 참조하십시오.

16.2.15.2. OpenShift Container Platform 클러스터 API 연결 문제

- 증상: 설치 프로그램이 완료되었지만 OpenShift Container Platform 클러스터 API를 사용할 수 없습니다. 부트스트랩 가상 시스템은 부트스트랩 프로세스가 완료된 후에도 유지됩니다. 다음 명령을 입력하면 응답 시간이 초과됩니다.

```
$ oc login -u kubeadmin -p *** <apiurl>
```

- 원인: 부트스트랩 VM이 설치 프로그램으로 삭제되지 않았으며 클러스터의 API IP 주소를 해제하지 않았습니다.

- 해결책: 부트스트랩 프로세스가 완료되면 wait-for 하위 명령을 사용합니다.

```
$ ./openshift-install wait-for bootstrap-complete
```

부트스트랩 프로세스가 완료되면 부트스트랩 가상 시스템을 삭제합니다.

```
$ ./openshift-install destroy bootstrap
```

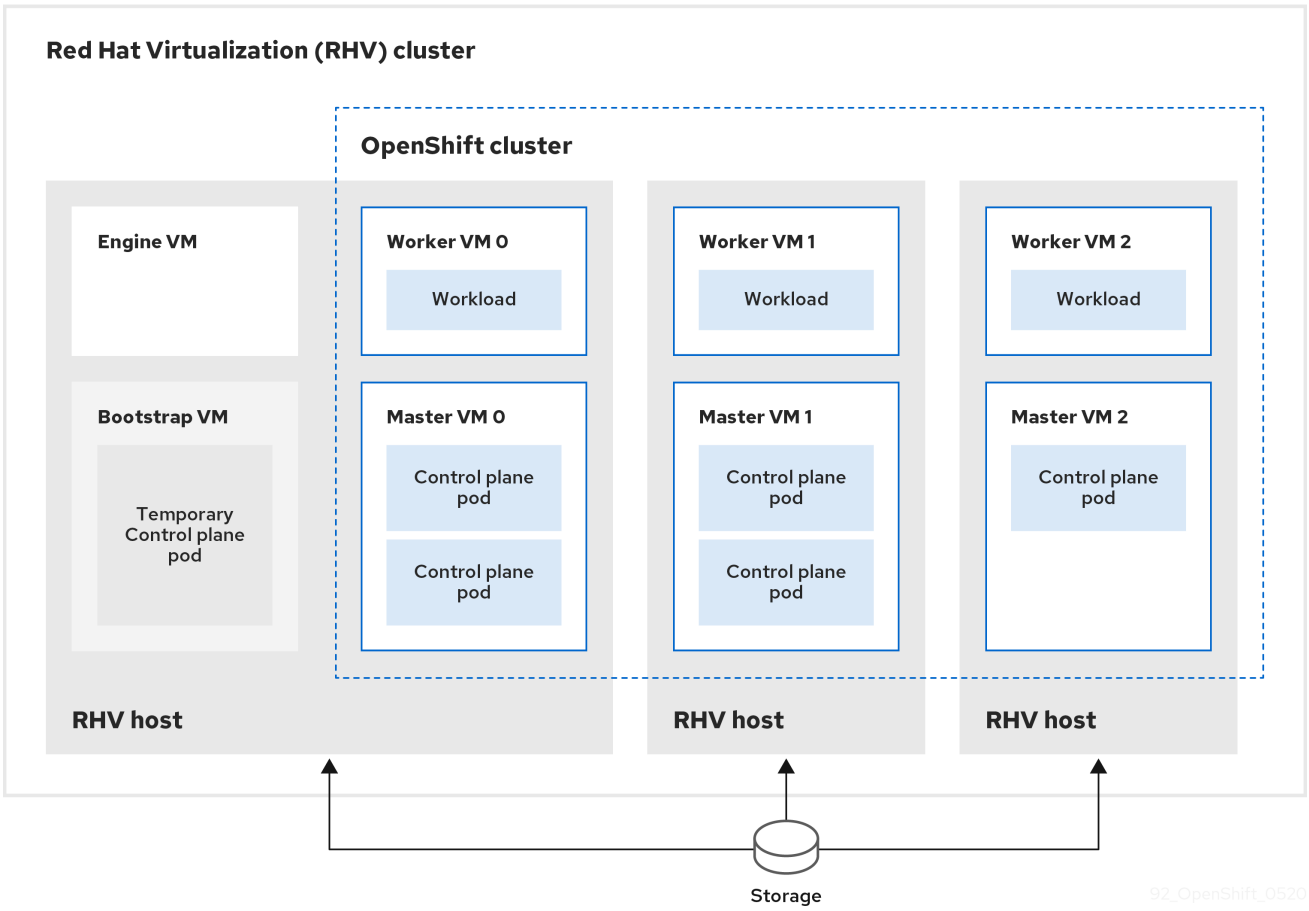
16.2.16. 설치 후 작업

OpenShift Container Platform 클러스터가 초기화된 후 다음 작업을 수행할 수 있습니다.

- 선택사항: 배포 후 OpenShift Container Platform의 MCO(Machine Config Operator)를 사용하여 SSH 키를 추가하거나 교체합니다.
- 선택사항: kubeadmin 사용자를 제거합니다. 대신 인증 공급자를 사용하여 클러스터-관리자 권한이 있는 사용자를 만듭니다.

16.3. 사용자 지정으로 RHV에 클러스터 설치

다음 다이어그램과 같이 RHV(Red Hat Virtualization)에서 OpenShift Container Platform 클러스터를 사용자 지정하고 설치할 수 있습니다.



설치 프로그램은 설치 관리자가 프로비저닝한 인프라를 사용하여 클러스터 생성 및 배포를 자동화합니다.

사용자 지정된 클러스터를 설치하려면 환경을 준비하고 다음 단계를 수행합니다.

1. 설치 프로그램을 실행하고 메시지에 따라 설치 구성 파일인 **install-config.yaml** 파일을 만듭니다.
2. **install-config.yaml** 파일에서 매개변수를 검사하고 수정합니다.
3. **install-config.yaml** 파일의 작업 사본을 만듭니다.
4. **install-config.yaml** 파일의 사본으로 설치 프로그램을 실행합니다.

그러면 설치 프로그램이 **OpenShift Container Platform** 클러스터를 만듭니다.

사용자 지정 클러스터 설치 대안은 기본 클러스터 설치를 참조하십시오.

참고

이 설치 프로그램은 **Linux** 및 **macOS**에서만 사용할 수 있습니다.

16.3.1. 사전 요구 사항

- **OpenShift Container Platform** 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- **RHV(Red Hat Virtualization)**의 **OpenShift Container Platform** 지원 매트릭스에서 지원되는 버전 조합이 있습니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.
- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성해야 합니다.

16.3.2. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.

중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

16.3.3. RHV 환경 요구사항

OpenShift Container Platform 버전 4.9 클러스터를 설치하고 실행하려면 RHV 환경이 다음 요구 사항을 충족해야 합니다.

이러한 요구 사항을 충족하지 않으면 설치 또는 프로세스가 실패할 수 있습니다. 또한 이러한 요구 사항을 충족하지 않으면 OpenShift Container Platform 클러스터가 설치 후 며칠 또는 몇 주에 실패할 수 있습니다.

CPU, 메모리, 스토리지에 대한 다음 요구사항은 설치 프로그램이 생성하는 기본 가상 시스템 수를 곱한 기본 값을 기반으로 합니다. 이러한 리소스는 RHV 환경에서 비 OpenShift Container Platform 작업에 사용하는 리소스와함께 사용할 수 있어야 합니다.

설치 프로그램은 기본적으로 설치 프로세스 중에 일곱 개의 가상 시스템을 생성합니다. 먼저 나머지 OpenShift Container Platform 클러스터를 생성하는 동안 임시 서비스와 컨트롤 플레인 영역을 제공하

는 부트스트랩 가상 머신을 생성합니다. 설치 프로그램이 클러스터 생성을 완료하면 부트스트랩 시스템을 삭제하여 리소스를 확보합니다.

RHV 환경에서 가상 머신 수를 늘리면 그에 따라 리소스를 늘려야합니다.

요구사항

- RHV 버전은 4.4입니다.
- RHV 환경에는 상태가 Up인 데이터 센터가 하나 있습니다.
- RHV 데이터 센터에는 RHV 클러스터가 포함되어 있습니다.
- RHV 클러스터에는 다음과 같은 OpenShift Container Platform 클러스터 전용 리소스가 있습니다.
  - 최소 28개의 vCPU(설치 중 생성된 일곱 개의 가상 시스템마다 각각 4개)
  - 다음을 포함한 112GiB RAM 이상
    - 임시 컨트롤 플레인을 제공하는 부트스트랩 시스템의 경우 16GiB 이상
    - 컨트롤 플레인을 제공하는 컨트롤 플레인 시스템 세 개 각각에 대해 16GiB 이상
    - 애플리케이션 워크로드를 실행하는 컴퓨팅 시스템 세 개 각각에 대해 16GiB 이상
- RHV 스토리지 도메인은 이러한 etcd 백엔드 성능 요구사항을 충족해야 합니다.
- 선호도 그룹 지원의 경우 다음을 수행합니다.

  작업자 또는 컨트롤 플레인당 하나의 물리적 시스템입니다. 작업자와 컨트롤 플레인은 동일한 실제 시스템에 있을 수 있습니다. 예를 들어 작업자 3개와 컨트롤 플레인 3개가 있는 경우 세

개의 실제 시스템이 필요합니다. 4개의 작업자와 컨트롤 플레인 3개가 있는 경우 4개의 물리적 시스템이 필요합니다.

- 하드 선호도 방지(기본값)의 경우 다음을 수행합니다. 최소 3대의 실제 시스템입니다. 3개 이상의 작업자 노드의 경우 작업자 또는 컨트롤 플레인당 하나의 물리적 시스템이 사용됩니다. 작업자와 컨트롤 플레인은 동일한 실제 시스템에 있을 수 있습니다.
- 사용자 지정 선호도 그룹의 경우 다음을 수행합니다. 리소스가 정의한 선호도 그룹 규칙에 적합한지 확인합니다.

- 프로덕션 환경에서 각 가상 머신은 120GiB 이상이어야 합니다. 따라서 스토리지 도메인은 기본 OpenShift Container Platform 클러스터에 대해 840GiB 이상을 제공해야 합니다. 리소스가 제한적인 환경이나 프로덕션 이외의 환경에서는 각 가상 시스템에 32GiB 이상이 있어야 하므로 스토리지 도메인에는 기본 OpenShift Container Platform 클러스터에 필요한 230GiB 이상이 있어야 합니다.

- 설치 및 업데이트 중에 Red Hat Ecosystem Catalog에서 이미지를 다운로드하려면 RHV 클러스터가 인터넷 연결에 액세스할 수 있어야 합니다. Telemetry 서비스에는 서브스크립션 및 권한 부여 프로세스를 단순화하기 위해 인터넷 연결이 필요합니다.

- RHV 클러스터에는 RHV Manager의 REST API에 액세스할 수 있는 가상 네트워크가 있어야 합니다. 설치 관리자가 생성한 VM에서 DHCP를 사용하여 IP 주소를 얻을 수 있으므로 이 네트워크에서 DHCP가 활성화되어 있는지 확인합니다.

- 대상 RHV 클러스터에서 OpenShift Container Platform 클러스터를 설치 및 관리하기 위한 다음과 같은 최소 권한이있는 사용자 계정 및 그룹:
  - DiskOperator
  - DiskCreator
  - UserTemplateBasedVm
  - TemplateOwner

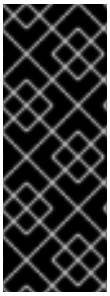
- TemplateCreator
- 대상 클러스터의 ClusterAdmin

주의

최소 권한 원칙을 적용합니다. 설치 과정에서 RHV에 대한 SuperUser 권한이 있는 관리자 계정을 사용하지 않도록 합니다. 설치 프로그램은 사용자가 제공한 인증 정보를 손상된 임시 ovirt-config.yaml 파일에 저장합니다.

16.3.4. RHV 환경에 대한 요구사항 확인

RHV 환경이 OpenShift Container Platform 클러스터 설치 및 실행 요구사항을 충족하는지 확인합니다. 이러한 요구사항을 충족하지 않으면 실패할 수 있습니다.



중요

이러한 요구사항은 설치 프로그램이 컨트롤 플레인과 컴퓨팅 시스템을 생성하는 데 사용하는 기본 리소스를 기반으로 합니다. 이러한 리소스에는 vCPU, 메모리 및 스토리지가 포함됩니다. 이러한 리소스를 변경하거나 OpenShift Container Platform 시스템 수를 늘리는 경우에는 그에 따라 이 요구사항을 조정합니다.

절차

1. RHV 버전이 OpenShift Container Platform 버전 4.9의 설치를 지원하는지 확인합니다.
   a. RHV 관리 포털에서 오른쪽 상단에 있는 ? 도움말 아이콘을 클릭하고 정보를 선택합니다.
   b. 창이 열리면 RHV 소프트웨어 버전을 기록합니다.
   c. RHV 버전이 4.4인지 확인합니다. 지원되는 버전 조합에 대한 자세한 내용은 RHV의 OpenShift Container Platform에 대한 지원 매트릭스를 참조하십시오.

2. 데이터 센터, 클러스터 및 스토리지를 검사합니다.

   a. RHV 관리 포털에서 **Compute** → **Data Centers**를 클릭합니다.

   b. **OpenShift Container Platform**을 설치하려는 데이터 센터에 액세스할 수 있는지 확인합니다.

   c. 해당 데이터 센터의 이름을 클릭합니다.

   d. 데이터 센터 세부 사항의 스토리지 탭에서 **OpenShift Container Platform**을 설치하려는 스토리지 도메인이 활성인지 확인합니다.

   e. 나중에 사용할 수 있도록 도메인 이름을 기록합니다.

   f. 여유 공간이 **230GiB** 이상인지 확인합니다.

   g. 스토리지 도메인이 **fio** 성능 벤치마킹 툴을 사용하여 측정할 수 있는 이러한 **etcd** 백엔드 성능 요구사항을 충족하는지 확인합니다.

   h. 데이터 센터 세부 사항에서 클러스터 탭을 클릭합니다.

   i. **OpenShift Container Platform**을 설치할 **RHV** 클러스터를 찾습니다. 나중에 사용할 수 있도록 클러스터 이름을 기록합니다.

3. **RHV** 호스트 리소스를 검사합니다.

   a. **RHV** 관리 포털에서 컴퓨팅 **>** 클러스터를 클릭합니다.

   b. **OpenShift Container Platform**을 설치할 클러스터를 클릭합니다.

c. 클러스터 세부 사항에서 호스트 탭을 클릭합니다.

d. 호스트를 검사하고 **OpenShift Container Platform** 클러스터 *전용*으로 사용할 수 있는 총 **28**개 이상의 논리 **CPU** 코어가 있는지 확인합니다.

e. 나중에 사용할 수 있도록 사용 가능한 논리 **CPU** 코어 수를 기록합니다.

f. 설치 중에 생성된 **7**개의 가상 시스템 각각에 **4**개의 코어가 있을 수 있도록 이러한 **CPU** 코어가 분산되어 있는지 확인합니다.

g. 다음 **OpenShift Container Platform** 시스템 각각에 대한 요구사항을 충족하기 위해 배포된 새 가상 시스템 예약에 필요한 **112GiB**의 최대 여유 메모리가 모두 호스트에 있는지 확인합니다.

- 부트스트랩 시스템에 필요한 **16GiB**
- 세 개의 컨트롤 플레인 시스템 각각에 필요한 **16GiB**
- 세 개의 컴퓨팅 시스템 각각에 대해 **16GiB**

h. 나중에 사용할 수 있도록 새 가상 시스템 예약에 필요한 최대 여유 메모리의 양을 기록합니다.

4. **OpenShift Container Platform**을 설치할 가상 네트워크가 **RHV Manager**의 **REST API**에 액세스할 수 있는지 확인합니다. 이 네트워크의 가상 시스템에서 **RHV** 관리자의 **REST API**에 도달하기 위해 **curl**을 사용합니다.

```
$ curl -k -u <username>@<profile>:<password> \ 1
https://<engine-fqdn>/ovirt-engine/api 2
```

1 **<username>** 의 경우 **RHV**에서 **OpenShift Container Platform** 클러스터를 만들고 관리할 수 있는 권한이 있는 **RHV** 계정의 사용자 이름을 지정합니다. **<profile>**은 로그인 프로파일을 지정합니다(**RHV** 관리 포털 로그인 페이지로 이동하여 프로파일 드롭다운 목록 검토). **<password>**의 경우 해당 사용자 이름에 대한 암호를 지정합니다.

2

<engine-fqdn>은 RHV 환경의 정규화된 도메인 이름을 지정합니다.

예를 들면 다음과 같습니다.

```
$ curl -k -u ocpadmin@internal:pw123 \
https://rhv-env.virtlab.example.com/ovirt-engine/api
```

16.3.5. RHV에서 네트워크 환경 준비

OpenShift Container Platform 클러스터의 고정 IP 주소 두 개를 구성하고 이 주소를 사용하여 DNS 항목을 만듭니다.

절차

1. 고정 IP 주소 두 개 예약

   a. OpenShift Container Platform을 설치하려는 네트워크에서 DHCP 임대 풀 외부에 있는 두 개의 고정 IP 주소를 식별합니다.

   b. 이 네트워크의 호스트에 연결하고 각 IP 주소를 사용하지 않는지 확인합니다. 예를 들어 Address Resolution Protocol (ARP)를 사용하여 IP 주소에 항목이 없는지 확인합니다.

   ```
   $ arp 10.35.1.19
   ```

   출력 예

   ```
   10.35.1.19 (10.35.1.19) -- no entry
   ```

   c. 네트워크 환경에 대한 표준 방식에 따라 두 개의 고정 IP 주소를 예약합니다.

d.

나중에 참조할 수 있도록 이 IP 주소를 기록합니다.

2.

이 형식을 사용하여 OpenShift Container Platform REST API 및 앱 도메인 이름에 대한 DNS 항목을 만듭니다.

```
api.<cluster-name>.<base-domain>   <ip-address> 1
*.apps.<cluster-name>.<base-domain>   <ip-address> 2
```

1

<cluster-name>, <base-domain>, <ip-address>는 각각 OpenShift Container Platform API의 클러스터 이름, 기본 도메인, 고정 IP 주소를 지정합니다.

2

인그레스 및 로드 밸런서 용 OpenShift Container Platform 앱의 클러스터 이름, 기본 도메인, 고정 IP 주소를 지정합니다.

예를 들면 다음과 같습니다.

```
api.my-cluster.virtlab.example.com 10.35.1.19
*.apps.my-cluster.virtlab.example.com 10.35.1.20
```

16.3.6. 비보안 모드에서 RHV에 OpenShift Container Platform 설치

기본적으로 설치 관리자는 CA 인증서를 생성하고 확인을 요청한 후 설치 중에 사용할 인증서를 저장합니다. 수동으로 생성하거나 설치할 필요가 없습니다.

권장되지는 않지만 비보안 모드로 OpenShift Container Platform을 설치하여 인증서를 확인하지 않고도 이 기능을 재정의하고 OpenShift Container Platform을 설치할 수 있습니다.

주의

잠재적인 공격자가 중Man-in-the-Middle 공격을 수행할 수 있고 네트워크에서 중요한 인증 정보를 캡처할 수 있으므로 비보안 모드에 설치하는 것은 권장되지 않습니다.

절차

1. ~/.ovirt/ovirt-config.yaml이라는 파일을 생성합니다.

2. ovirt-config.yaml에 다음 내용을 추가합니다.

```
ovirt_url: https://ovirt.example.com/ovirt-engine/api (1)
ovirt_fqdn: ovirt.example.com (2)
ovirt_pem_url: ""
ovirt_username: ocpadmin@internal
ovirt_password: super-secret-password (3)
ovirt_insecure: true
```

(1) oVirt 엔진의 호스트 이름 또는 주소를 지정합니다.

(2) oVirt 엔진의 정규화된 도메인 이름을 지정합니다.

(3) oVirt 엔진의 admin 암호를 지정합니다.

3. 설치 프로그램을 실행합니다.

16.3.7. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드

에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 core 사용자의 ~/.ssh/authorized_keys 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 core로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. ./openshift-install gather 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

절차

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: ~/.ssh/id_ed25519)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 '~/.ssh 디렉터리에 있는지 확인하십시오.

참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 ed25519 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 rsa 또는 ecdsa 알고리즘을 사용하는 키를 생성합니다.

2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 ~/.ssh/id_ed25519.pub 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 ./openshift-install gather 명령을 사용하려는 경우 필요합니다.

참고

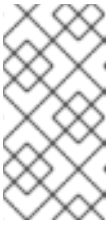
일부 배포에서는 ~/.ssh/id_rsa 및 ~/.ssh/id_dsa와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. ssh-agent 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```



참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. ssh-agent에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1

SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: ~/.ssh/id_ed25519).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

16.3.8. 설치 프로그램 받기

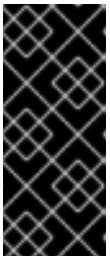
OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

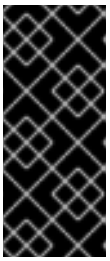
절차

1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.
2. 인프라 공급자를 선택합니다.
3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.



중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.



중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 **OpenShift Container Platform** 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 **Linux** 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 **OpenShift Container Platform** 구성 요소에 대한 컨테이너 이미지를 제공하는 **Quay.io**를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

16.3.9. 설치 구성 파일 만들기

RHV(Red Hat Virtualization)에 설치하는 **OpenShift Container Platform** 클러스터를 사용자 지정할 수 있습니다.

사전 요구 사항

- **OpenShift Container Platform** 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.
- 서브스크립션 수준에서 서비스 주체 권한을 획득합니다.

절차

1. **install-config.yaml** 파일을 생성합니다.

a.

설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1

<installation_directory>는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.

중요

비어 있는 디렉터리를 지정합니다. 부트스트랩 **X.509** 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 **OpenShift Container Platform** 버전에서 설치 파일을 복사할 때는 주의하십시오.

b.

설치 프로그램 메시지를 따릅니다.

i.

SSH Public Key는 암호가 없는 공개 키(예: ~ / **.ssh** / **id_rsa.pub**)를 선택합니다. 이 키는 새로운 **OpenShift Container Platform** 클러스터와의 연결을 인증합니다.

참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 **OpenShift Container Platform** 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 **SSH** 키를 선택합니다.

ii.

Platform에 대해 **ovirt**를 선택합니다.

iii.

Enter oVirt's API endpoint URL은 다음 형식을 사용하여 **RHV API**의 **URL**을 입력합니다.

```
https://<engine-fqdn>/ovirt-engine/api 1
```

1

예를 들면 다음과 같습니다.

```
$ curl -k -u ocpadmin@internal:pw123 \
https://rhv-env.virtlab.example.com/ovirt-engine/api
```

iv. **Is the oVirt CA trusted locally?**는 이미 CA 인증서를 설정했으므로 **Yes**를 입력합니다. 그렇지 않으면 **No**를 입력합니다.

v. **oVirt's CA bundle**의 경우 이전 질문에 **Yes**를 입력했으면 **/etc/pki/ca-trust/source/anchors/ca.pem**에서 인증서 내용을 복사하여 여기에 붙여 넣습니다. 그런 다음 **Enter**를 두 번 누릅니다. 이전 질문에 **No**를 입력했으면 이 질문이 나타나지 않습니다.

vi. **oVirt engine username**은 다음 형식을 사용하여 RHV 관리자의 사용자 이름과 프로파일을 입력합니다.

```
<username>@<profile> 1
```

1 **<username>**은 RHV 관리자의 사용자 이름을 지정합니다. **<profile>**은 로그인 프로파일을 지정합니다(RHV 관리 포털 로그인 페이지로 이동하여 프로파일 드롭다운 목록 검토). 사용자 이름과 프로파일의 올바른 예는 다음과 같습니다.

```
ocpadmin@internal
```

vii. **oVirt engine password**는 RHV 관리자 암호를 입력합니다.

viii. **oVirt cluster**는 **OpenShift Container Platform**을 설치할 클러스터를 선택합니다.

ix. **oVirt storage domain**은 **OpenShift Container Platform**을 설치할 스토리지 도메인을 선택합니다.

x. **oVirt network**는 RHV Manager REST API에 액세스할 수있는 가상 네트워크를

선택합니다.

xi. **Internal API Virtual IP**는 클러스터의 **REST API**에 대해 별도로 설정한 고정 **IP** 주소를 입력합니다.

xii. **Ingress virtual IP**는 와일드카드 앱 도메인용으로 예약한 고정 **IP** 주소를 입력합니다.

xiii. **Base Domain**은 **OpenShift Container Platform** 클러스터의 기본 도메인을 입력합니다. 이 클러스터가 외부에 노출된 경우 **DNS** 인프라에서 인식할 수 있는 유효한 도메인이어야 합니다. 예를 들어 **virtlab.example.com**을 입력합니다.

xiv. **Cluster Name**은 클러스터 이름을 입력합니다. 예: **myca.crt OpenShift Container Platform REST API** 및 앱 도메인 이름에 대해 생성한 외부 등록/확인 가능 **DNS** 항목에서 클러스터 이름을 사용합니다. 설치 프로그램은 **RHV** 환경의 클러스터에도 이 이름을 지정합니다.

xv. **Pull Secret**은 이전에 다운로드한 **pull-secret.txt** 파일에서 풀 시크릿을 복사하여 여기에 붙여 넣습니다. **Red Hat OpenShift Cluster Manager**에서 동일한 풀 시크릿 의 사본을 가져올 수도 있습니다.

2. **install-config.yaml** 파일을 수정합니다. 사용 가능한 매개변수에 대한 자세한 정보는 “설치 구성 매개변수” 섹션에서 확인할 수 있습니다.

참고

Manager에 중간 CA 인증서가 있는 경우 인증서가 ovirt-config.yaml 파일과 install-config.yaml 파일에 표시되는지 확인합니다. 표시되지 않는 경우 다음과 같이 추가합니다.

1. ~/.ovirt/ovirt-config.yaml 파일에서 다음을 수행합니다.

```
[ovirt_ca_bundle]: |
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA>
    -----END CERTIFICATE-----
    -----BEGIN CERTIFICATE-----
    <INTERMEDIATE_CA>
    -----END CERTIFICATE-----
```

2. install-config.yaml 파일에서 다음을 수행합니다.

```
[additionalTrustBundle]: |
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA>
    -----END CERTIFICATE-----
    -----BEGIN CERTIFICATE-----
    <INTERMEDIATE_CA>
    -----END CERTIFICATE-----
```

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 install-config.yaml 파일을 백업합니다.

중요

install-config.yaml 파일은 설치 과정에서 사용됩니다. 이 파일을 재사용하려면 지금 백업해야 합니다.

16.3.9.1. RHV(Red Hat Virtualization)용 install-config.yaml 파일 예

install-config.yaml 파일에서 매개변수와 매개변수 값을 변경하여 설치 프로그램이 생성하는 OpenShift Container Platform 클러스터를 사용자 지정할 수 있습니다.

다음은 RHV에 OpenShift Container Platform을 설치하는 예입니다.

install-config.yaml은 다음 명령을 실행할 때 지정한 <installation_directory>에 있습니다.

```
$ ./openshift-install create install-config --dir <installation_directory>
```

참고

- 이 예제 파일은 참조용으로만 제공됩니다. install-config.yaml 파일은 설치 프로그램을 사용하여 받아야 합니다.
- install-config.yaml 파일을 변경하면 클러스터에 필요한 리소스가 늘어날 수 있습니다. RHV 환경에 이러한 추가 리소스가 있는지 확인합니다. 추가 리소스가 없으면 설치 또는 클러스터가 실패합니다.

예: 기본 install-config.yaml 파일

```
apiVersion: v1
baseDomain: example.com
compute:
- architecture: amd64
  hyperthreading: Enabled
  name: worker
  platform: {}
  replicas: 3
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  platform: {}
  replicas: 3
metadata:
  creationTimestamp: null
  name: my-cluster
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  ovirt:
    api_vip: 10.46.8.230
    ingress_vip: 192.168.1.5
    ovirt_cluster_id: 68833f9f-e89c-4891-b768-e2ba0815b76b
    ovirt_storage_domain_id: ed7b0f4e-0e96-492a-8fff-279213ee1468
    ovirt_network_name: ovirtmgmt
    vnicProfileID: 3fa86930-0be5-4052-b667-b79f0a729692
```

```
publish: External
pullSecret: '{"auths": ...}'
sshKey: ssh-ed12345 AAAA...
```

예: 최소 install-config.yaml 파일

```
apiVersion: v1
baseDomain: example.com
metadata:
  name: test-cluster
platform:
  ovirt:
    api_vip: 10.46.8.230
    ingress_vip: 10.46.8.232
    ovirt_cluster_id: 68833f9f-e89c-4891-b768-e2ba0815b76b
    ovirt_storage_domain_id: ed7b0f4e-0e96-492a-8fff-279213ee1468
    ovirt_network_name: ovirtmgmt
    vnicProfileID: 3fa86930-0be5-4052-b667-b79f0a729692
pullSecret: '{"auths": ...}'
sshKey: ssh-ed12345 AAAA...
```

예: install-config.yaml 파일의 사용자 지정 시스템 풀

```
apiVersion: v1
baseDomain: example.com
controlPlane:
  name: master
  platform:
    ovirt:
      cpu:
        cores: 4
        sockets: 2
      memoryMB: 65536
      osDisk:
        sizeGB: 100
      vmType: server
  replicas: 3
compute:
- name: worker
  platform:
    ovirt:
      cpu:
        cores: 4
        sockets: 4
      memoryMB: 65536
      osDisk:
        sizeGB: 200
      vmType: server
  replicas: 5
metadata:
  name: test-cluster
platform:
  ovirt:
    api_vip: 10.46.8.230
    ingress_vip: 10.46.8.232
    ovirt_cluster_id: 68833f9f-e89c-4891-b768-e2ba0815b76b
```

```
    ovirt_storage_domain_id: ed7b0f4e-0e96-492a-8fff-279213ee1468
    ovirt_network_name: ovirtmgmt
    vnicProfileID: 3fa86930-0be5-4052-b667-b79f0a729692
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```

강제 적용되지 않는 선호도 그룹의 예

가능한 경우 컨트롤 플레인과 작업자를 배포할 수 있도록 강제 적용되지 않는 선호도 그룹을 추가하여 최대한 많은 클러스터를 사용하는 것이 좋습니다.

```
platform:
  ovirt:
    affinityGroups:
    - description: AffinityGroup to place each compute machine on a separate host
      enforcing: true
      name: compute
      priority: 3
    - description: AffinityGroup to place each control plane machine on a separate host
      enforcing: true
      name: controlplane
      priority: 5
    - description: AffinityGroup to place worker nodes and control plane nodes on separate
hosts
      enforcing: false
      name: openshift
      priority: 5
compute:
- architecture: amd64
  hyperthreading: Enabled
  name: worker
  platform:
    ovirt:
      affinityGroupsNames:
      - compute
      - openshift
  replicas: 3
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  platform:
    ovirt:
      affinityGroupsNames:
      - controlplane
      - openshift
  replicas: 3
```

프로덕션 환경 외 랩 설정에 대한 모든 선호도 그룹 제거 예

프로덕션 환경 이외의 랩 설정의 경우 모든 선호도 그룹을 제거하여 **OpenShift Container Platform** 클러스터를 보유한 몇 개의 호스트에 집중시켜야 합니다.

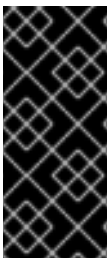
```
platform:
  ovirt:
    affinityGroups: []
compute:
- architecture: amd64
  hyperthreading: Enabled
  name: worker
  platform:
    ovirt:
      affinityGroupsNames: []
  replicas: 3
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  platform:
    ovirt:
      affinityGroupsNames: []
  replicas: 3
```

16.3.9.2. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 매개변수 값을 제공하여 클러스터를 호스팅할 클라우드 플랫폼에서 사용자 계정을 설명하고 선택사항으로 클러스터의 플랫폼을 사용자 지정합니다. **install-config.yaml** 설치 구성 파일을 생성할 때 명령줄을 통해 필요한 매개변수 값을 제공합니다. 클러스터를 사용자 지정하면 **install-config.yaml** 파일을 수정하여 플랫폼에 대한 세부 정보를 제공할 수 있습니다.

참고

설치한 후에는 **install-config.yaml** 파일에서 이러한 매개변수를 수정할 수 없습니다.



중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

16.3.9.2.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 16.1. 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	소문자, 하이픈(**-**), 마침표(**.**)로 구성되는 문자열(예: **dev**)입니다.
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

매개변수	설명	값

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

16.3.9.2.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 16.2. 네트워크 매개변수

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체 **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.

매개변수	설명	값
networking.clusterNetwork	Pod의 IP 주소 블록입니다. 기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된 **cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16

매개변수	설명	값
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16**  **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

16.3.9.2.3. 선택적 구성 매개변수

선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 16.3. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다. 자세한 내용은 "시스템 풀에 대한 추가 RHV 매개변수" 표를 참조하십시오.
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열

매개변수	설명	값
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은 **3**입니다.
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다. 자세한 내용은 "시스템 풀에 대한 추가 RHV 매개변수" 표를 참조하십시오.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열

매개변수	설명	값
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 기본값은 **External**입니다. 이 필드를 **Internal** 로 설정하는 것은 클라우드 이외의 플랫폼에서는 지원되지 않습니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

16.3.9.2.4. 추가 RHV(Red Hat Virtualization) 구성 매개 변수

추가 RHV 구성 매개변수는 다음 표에 설명되어 있습니다.

표 16.4. 클러스터의 추가 RHV(Red Hat Virtualization) 매개변수

매개변수	설명	값
platform.ovirt.ovirt_cluster_id	필수 항목입니다. VM이 생성될 클러스터입니다.	문자열. 예: **68833f9f-e89c-4891-b768-e2ba0815b76b**
platform.ovirt.ovirt_storage_domain_id	필수 항목입니다. VM 디스크가 생성될 스토리지 도메인 ID입니다.	문자열. 예: **ed7b0f4e-0e96-492a-8fff-279213ee1468**
platform.ovirt.ovirt_network_name	필수 항목입니다. VM NIC가 생성될 네트워크 이름입니다.	문자열. 예: **ocpcluster**
platform.ovirt.vnicProfileID	필수 항목입니다. VM 네트워크 인터페이스의 vNIC 프로파일 ID입니다. 클러스터 네트워크의 프로파일이 하나인 경우 유추할 수 있습니다.	문자열. 예: **3fa86930-0be5-4052-b667-b79f0a729692**
platform.ovirt.api_vip	필수 항목입니다. API 가상 IP(VIP)에 할당될 시스템 네트워크의 IP 주소입니다. 이 끝점에서 OpenShift API에 액세스할 수 있습니다.	문자열. 예: **10.46.8.230**
platform.ovirt.ingress_vip	필수 항목입니다. Ingress 가상 IP(VIP)에 할당될 머신 네트워크의 IP 주소입니다.	문자열. 예: **10.46.8.232**
platform.ovirt.affinityGroups	선택 사항: 설치 프로세스 중에 생성할 선호도 그룹 목록입니다.	개체 목록.

매개변수	설명	값
platform.ovirt.affinityGroups.description	**platform.ovirt.affinityGroups**를 포함하는 경우 필수 항목입니다. 선호도 그룹에 대한 설명입니다.	문자열. 예: **AffinityGroup for spreading each compute machine to a different host**
platform.ovirt.affinityGroups.enforcing	**platform.ovirt.affinityGroups**를 포함하는 경우 필수 항목입니다. **true**로 설정하면 사용 가능한 하드웨어 노드가 충분하지 않은 경우 RHV에서 시스템을 프로비저닝하지 않습니다. **false**로 설정하면 하드웨어 노드가 충분하지 않은 경우에도 RHV에서 시스템을 프로비저닝하므로 동일한 실제 시스템에서 여러 가상 시스템이 호스팅됩니다.	문자열. 예: **true**
platform.ovirt.affinityGroups.name	**platform.ovirt.affinityGroups**를 포함하는 경우 필수 항목입니다. 선호도 그룹의 이름입니다.	문자열. 예: **compute**
platform.ovirt.affinityGroups.priority	**platform.ovirt.affinityGroups**를 포함하는 경우 필수 항목입니다. **platform.ovirt.affinityGroups.enforcing = false**인 경우 선호도 그룹에 제공되는 우선 순위입니다. RHV는 우선 순위에 따라 선호도 그룹을 적용합니다. 여기서 더 큰 수가 더 적은 수보다 우선합니다. 여러 선호도 그룹에 동일한 우선 순위가 있는 경우 적용되는 순서가 보장되지 않습니다.	정수. 예: **3**

16.3.9.2.5. 시스템 풀의 추가 RHV 매개변수

시스템 풀의 추가 RHV 구성 매개변수는 다음 표에 설명되어 있습니다.

표 16.5. 시스템 풀의 추가 RHV 매개변수

매개변수	설명	값
<machine-pool>.platform.ovirt.cpu	선택 사항: VM의 CPU를 정의합니다.	개체
<machine-pool>.platform.ovirt.cpu.cores	**<machine-pool>.platform.ovirt.cpu**를 사용하는 경우 필수 항목입니다. 코어 수입니다. 총 가상 CPU(vCPU) 수는 코어 수와 소켓 수를 곱한 값입니다.	정수

매개변수	설명	값
<machine-pool>.platform.ovirt.cpu.sockets	**<machine-pool>.platform.ovirt.cpu**를 사용하는 경우 필수 항목입니다. 코어당 소켓 수입니다. 총 가상 CPU(vCPU) 수는 코어 수와 소켓 수를 곱한 값입니다.	정수
<machine-pool>.platform.ovirt.memoryMB	선택 사항: MiB의 VM 메모리입니다.	정수
<machine-pool>.platform.ovirt.instanceTypeID	선택 사항: **https://<engine-fqdn>/ovirt-engine/api/instancetypes** 끝점에서 가져올 수 있는 인스턴스 유형 UUID(예: **00000009-0009-0009-0009-0000000000f1**)입니다. 주의 **instance_type_id** 필드는 더 이상 사용되지 않으며 향후 릴리스에서 제거됩니다.	UUID 문자열
<machine-pool>.platform.ovirt.osDisk	선택 사항: VM의 첫 번째 및 부팅 가능한 디스크를 정의합니다.	문자열
<machine-pool>.platform.ovirt.osDisk.sizeGB	**<machine-pool>.platform.ovirt.osDisk**를 사용하는 경우 필수 항목입니다. 디스크 크기(GiB)입니다.	숫자

매개변수	설명	값
<machine-pool>.platform.ovirt.vmType	선택 사항: **high-performance**, **server** 또는 **desktop**과 같은 VM 워크로드 유형입니다. 기본적으로 컨트롤 플레인 노드는 **고성능** 을 사용하고 작업자 노드는 **서버를** 사용합니다. 자세한 내용은 *가상 머신 관리 가이드*에서 새 가상 머신의 설정 설명 및 가상 머신 편집 및 고성능 가상 머신, 템플릿 및 풀 구성을 참조하십시오.  **참고** **high_performance** 는 VM의 성능을 개선하지만 제한 사항이 있습니다. 예를 들어 그래픽 콘솔을 사용하여 VM에 액세스할 수 없습니다. 자세한 내용은 *가상 머신 관리 가이드*에서 고성능 가상 머신, 템플릿 및 풀 구성을 참조하십시오.	문자열

매개변수	설명	값

<machine-pool>.platform.ovirt.affinityGroupsNames	선택 사항: 가상 머신에 적용해야 하는 선호도 그룹 이름 목록입니다. 선호도 그룹은 RHV에 있거나 이 항목의 *클러스터에 대한 추가RHV 매개변수*에 설명된 대로 설치 중에 생성해야 합니다. 이 항목은 비워 둘 수 있습니다.
	
	두 개의 선호도 그룹 예
	
	이 예제에서는 **compute** 및 **clusterWideNonEnforcing**이라는 두 개의 선호도 그룹을 정의합니다.
	
	`<machine-pool>:`
	`  platform:`
	`  ovirt:`
	`  affinityGroupNames:`
	`  - compute`
	`  - clusterWideNonEnforcing`
	
	이 예제에서는 유사성 그룹을 정의하지 않습니다.
	
	`<machine-pool>:`
	`  platform:`
	`  ovirt:`
	`  affinityGroupNames: []`
		문자열

<machine-pool>.platform.ovirt.AutoPinningPolicy	선택 사항: AutoPinningPolicy는 인스턴스의 호스트에 고정을 포함하여 CPU 및 NUMA 설정을 자동으로 설정하는 정책을 정의합니다. 필드를 생략하면 기본값은 **none**입니다. 지원되는 값: **none**,**resize_and_pin**. 자세한 내용은 *가상 머신 관리 가이드*에서 NUMA 노드 설정을 참조하십시오.	문자열
<machine-pool>.platform.ovirt.hugepages	선택 사항: Hugepages는 VM에서 hugepages를 정의하는 KiB 크기입니다. 지원되는 값: **2048** 또는 **1048576**. 자세한 내용은 *가상 머신 관리 가이드*에서 Huge Pages 구성을 참조하십시오.	정수

참고

<machine-pool>은 controlPlane 또는 compute로 바꿀 수 있습니다.

16.3.10. 클러스터 배포

호환되는 클라우드 플랫폼에 **OpenShift Container Platform**을 설치할 수 있습니다.

중요

최초 설치 과정에서 설치 프로그램의 **create cluster** 명령을 한 번만 실행할 수 있습니다.

사전 요구 사항

- 설치 프로그램을 실행하는 시스템에서 **Manager**에 대한 **ovirt-imageio** 포트를 엽니다. 기본적으로 포트는 **54322**입니다.
- **OpenShift Container Platform** 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

프로세스

1. 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터 배포를 초기화합니다.

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
  --log-level=info 2
```

1 **<installation_directory>** 값으로 사용자 지정한 **./install-config.yaml** 파일의 위치를 지정합니다.

2 다른 설치 세부 사항을 보려면 **info** 대신 **warn, debug** 또는 **error**를 지정합니다.

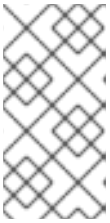
참고

호스트에 구성된 클라우드 공급자 계정에 클러스터를 배포하기에 충분한 권한이 없는 경우, 설치 프로세스가 중단되고 누락된 권한을 알리는 메시지가 표시됩니다.

클러스터 배포가 완료되면 웹 콘솔로 연결되는 링크와 **kubeadmin** 사용자의 인증 정보가 포함된 클러스터 액세스 지침이 사용자 터미널에 표시됩니다.

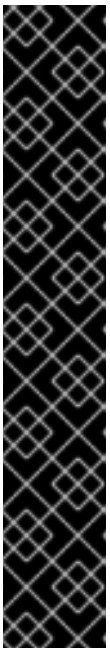
출력 예

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-
ymBZj-Wt5AL"
INFO Time elapsed: 36m22s
```



참고

설치에 성공하면 클러스터 액세스 및 인증 정보도 **<installation_directory>/.openshift_install.log**로 출력됩니다.



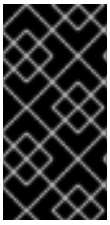
중요

- 설치 프로그램에서 생성하는 **Ignition** 구성 파일에 **24**시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 **24**시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 **kubelet** 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청**(CSR)**을 수동으로 승인해야 합니다. 자세한 내용은 ***Recovering from expired control plane certificates*** 문서를 참조하십시오.

- 클러스터를 설치한 후 **24**시간에서 **22**시간까지의 인증서가 교체되기 때문에 생성된 후 **12**시간 이내에 **Ignition** 구성 파일을 사용하는 것이 좋습니다. **12**시간 이내에 **Ignition** 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

중요

설치 프로그램에서 생성되는 파일이나 설치 프로그램을 삭제해서는 안 됩니다. 클러스터를 삭제하려면 두 가지가 모두 필요합니다.



중요

클러스터를 설치하는 데 필요한 단계를 완료했습니다. 나머지 단계는 클러스터를 확인하고 설치 문제를 해결하는 방법을 보여줍니다.

16.3.11. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift CLI(oc)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 oc를 설치할 수 있습니다.

중요

이전 버전의 oc를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 oc를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 Linux에서 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.
2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. OpenShift v4.9 Linux Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. oc 바이너리를 PATH에 있는 디렉터리에 배치합니다.

PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치

다음 절차에 따라 Windows에 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.
2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. OpenShift v4.9 Windows Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.
4. ZIP 프로그램으로 아카이브의 압축을 풉니다.
5. oc 바이너리를 PATH에 있는 디렉터리로 이동합니다.

   PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

   ```
   C:\> path
   ```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치

다음 절차에 따라 macOS에서 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.

2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.

3. OpenShift v4.9 MacOSX Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.

4. 아카이브의 압축을 해제하고 압축을 풉니다.

5. oc 바이너리 PATH의 디렉터리로 이동합니다.

   PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
$ oc <command>
```

16.3.12. CLI를 사용하여 클러스터에 로그인

클러스터 kubeconfig 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. kubeconfig 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- oc CLI를 설치했습니다.

프로세스

1.

1. kubeadmin 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 oc 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

자세한 내용은 OpenShift CLI 시작하기를 참조하십시오.

16.3.13. 클러스터 상태 확인

설치 중 또는 설치 후 OpenShift Container Platform 클러스터의 상태를 확인할 수 있습니다.

프로세스

1. 클러스터 환경에서 관리자의 kubeconfig 파일을 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

kubeconfig 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사

용하는 클러스터에 대한 정보가 포함되어 있습니다.

2. 배포 후 생성된 컨트롤 플레인 및 컴퓨팅 시스템을 확인합니다.

```
$ oc get nodes
```

3. 클러스터 버전을 확인합니다.

```
$ oc get clusterversion
```

4. Operator 상태를 확인합니다.

```
$ oc get clusteroperator
```

5. 클러스터에서 실행 중인 모든 Pod를 확인합니다.

```
$ oc get pods -A
```

문제 해결

설치에 실패하면 설치 프로그램이 시간 초과되고 오류 메시지가 표시됩니다. 자세한 내용은 설치 문제 해결을 참조하십시오.

16.3.14. RHV의 OpenShift Container Platform 웹 콘솔 액세스

OpenShift Container Platform 클러스터가 초기화된 후 OpenShift Container Platform 웹 콘솔에 로그인할 수 있습니다.

프로세스

1. 선택사항: RHV(Red Hat Virtualization) 관리 포털에서 컴퓨팅 → 클러스터를 엽니다.

2. 설치 프로그램이 가상 시스템을 생성하는지 확인합니다.

3. 설치 프로그램이 실행 중인 명령줄로 돌아갑니다. 설치 프로그램이 완료되면 OpenShift Container Platform 웹 콘솔에 로그인하기 위한 사용자 이름과 임시 암호가 표시됩니다.

4. 브라우저에서 **OpenShift Container Platform** 웹 콘솔의 **URL**을 입력합니다. **URL** 형식은 다음과 같습니다.

```
console-openshift-console.apps.<clustername>.<basedomain> 1
```

1 **<clustername>. <basedomain>**은 클러스터 이름과 기본 도메인을 지정합니다.

예를 들면 다음과 같습니다.

```
console-openshift-console.apps.my-cluster.virtlab.example.com
```

16.3.15. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 **Telemetry** 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 **Telemetry**가 자동으로 실행되고 **OpenShift Cluster Manager** 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 **OpenShift Cluster Manager**를 사용하여 자동으로 또는 **OpenShift Cluster Manager**를 사용하여 수동으로 유지 관리되는지 확인한 후 **subscription watch**를 사용하여 계정 또는 다중 클러스터 수준에서 **OpenShift Container Platform** 서브스크립션을 추적합니다.

추가 리소스

- **Telemetry** 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

16.3.16. RHV(Red Hat Virtualization)에 설치와 관련된 일반적인 문제 해결

다음은 발생할 수 있는 일반적인 문제와 가능한 원인 및 해결 방법입니다.

16.3.16.1. CPU 로드가 증가하고 노드가 Not Ready 상태가 됨

- 증상: **CPU** 로드가 크게 증가하고 노드가 **Not Ready** 상태가 되기 시작합니다.

- 원인: 특히 컨트롤 플레인 노드의 경우 스토리지 도메인 대기 시간이 너무 길 수 있습니다.

- 해결책:

  kubelet 서비스를 다시 시작하여 노드를 다시 준비합니다.

```
$ systemctl restart kubelet
```

  OpenShift Container Platform 메트릭 서비스를 검사하면 etcd 디스크 동기화 기간과 같은 중요한 데이터를 자동으로 수집하고 보고합니다. 클러스터가 작동 중인 경우 이 데이터를 사용하여 스토리지 대기 시간 또는 처리량이 근본적인 문제인지 여부를 판별합니다. 근본적인 문제가 맞다면 지연 시간이 짧고 처리량이 많은 스토리지 리소스를 사용합니다.

  원시 메트릭을 가져오려면 kubeadmin 또는 cluster-admin 권한이 있는 사용자로 다음 명령을 입력합니다.

```
$ oc get --insecure-skip-tls-verify --server=https://localhost:<port> --raw=/metrics
```

  자세한 내용은 Exploring Application Endpoints for the purposes of Debugging with OpenShift 4.x를 참조하십시오.

16.3.16.2. OpenShift Container Platform 클러스터 API 연결 문제

- 증상: 설치 프로그램이 완료되었지만 OpenShift Container Platform 클러스터 API를 사용할 수 없습니다. 부트스트랩 가상 시스템은 부트스트랩 프로세스가 완료된 후에도 유지됩니다. 다음 명령을 입력하면 응답 시간이 초과됩니다.

```
$ oc login -u kubeadmin -p *** <apiurl>
```

- 원인: 부트스트랩 VM이 설치 프로그램으로 삭제되지 않았으며 클러스터의 API IP 주소를 해제하지 않았습니다.

- 해결책: 부트스트랩 프로세스가 완료되면 wait-for 하위 명령을 사용합니다.

```
$ ./openshift-install wait-for bootstrap-complete
```

부트스트랩 프로세스가 완료되면 부트스트랩 가상 시스템을 삭제합니다.

```
$ ./openshift-install destroy bootstrap
```

16.3.17. 설치 후 작업

OpenShift Container Platform 클러스터가 초기화된 후 다음 작업을 수행할 수 있습니다.

- 선택사항: 배포 후 **OpenShift Container Platform**의 **MCO(Machine Config Operator)**를 사용하여 **SSH** 키를 추가하거나 교체합니다.
- 선택사항: **kubeadmin** 사용자를 제거합니다. 대신 인증 공급자를 사용하여 클러스터-관리자 권한이 있는 사용자를 만듭니다.

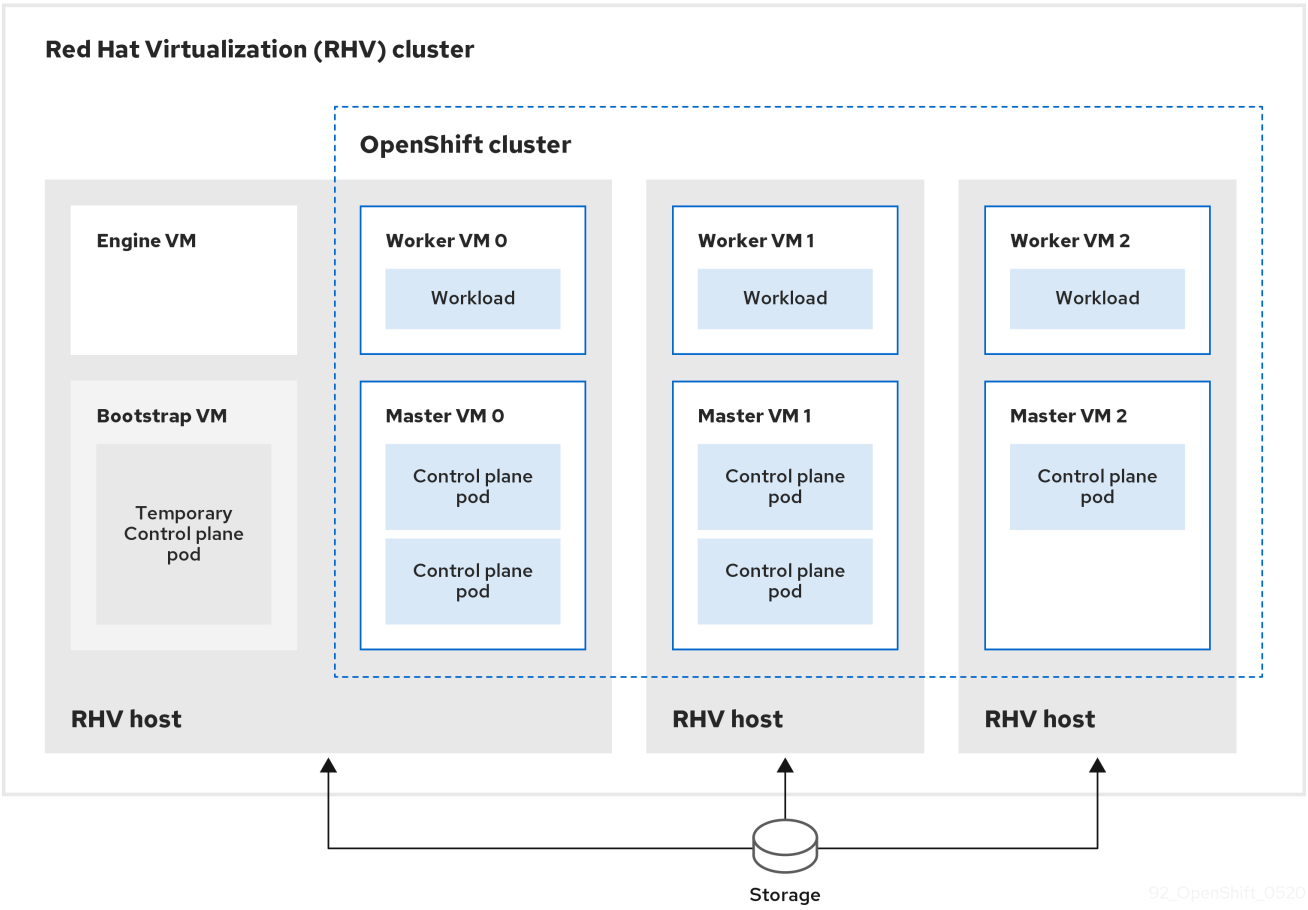
16.3.18. 다음 단계

- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.

16.4. 사용자 프로비저닝 인프라를 사용하여 RHV에 클러스터 설치

OpenShift Container Platform 버전 **4.9**에서는 **RHV(Red Hat Virtualization)** 및 사용자가 제공하는 기타 인프라에 사용자 지정 **OpenShift Container Platform** 클러스터를 설치할 수 있습니다. **OpenShift Container Platform** 문서에서는 *사용자 프로비저닝 인프라*라는 용어를 사용하여 이러한 인프라 유형을 나타냅니다.

다음 다이어그램은 **RHV** 클러스터에서 실행되는 잠재적인 **OpenShift Container Platform** 클러스터의 예를 보여줍니다.



RHV 호스트는 컨트롤 플레인과 컴퓨팅 pod를 모두 포함하는 가상 머신을 실행합니다. 또한 호스트 중 하나는 관리자 가상 머신과 임시 컨트롤 플레인 pod가 포함된 부트스트랩 가상 머신을 실행합니다.]

16.4.1. 사전 요구 사항

RHV 환경에 OpenShift Container Platform 클러스터를 설치하려면 다음 요구사항을 충족해야 합니다.

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.
- RHV(Red Hat Virtualization)의 OpenShift Container Platform 지원 매트릭스에서 지원되는 버전 조합이 있습니다.

16.4.2. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.

중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

16.4.3. RHV 환경 요구사항

OpenShift Container Platform 버전 4.9 클러스터를 설치하고 실행하려면 RHV 환경이 다음 요구 사항을 충족해야 합니다.

이러한 요구 사항을 충족하지 않으면 설치 또는 프로세스가 실패할 수 있습니다. 또한 이러한 요구 사항을 충족하지 않으면 OpenShift Container Platform 클러스터가 설치 후 며칠 또는 몇 주에 실패할 수 있습니다.

CPU, 메모리, 스토리지에 대한 다음 요구사항은 설치 프로그램이 생성하는 기본 가상 시스템 수를 곱한 기본 값을 기반으로 합니다. 이러한 리소스는 RHV 환경에서 비 OpenShift Container Platform 작업에 사용하는 리소스와함께 사용할 수 있어야 합니다.

설치 프로그램은 기본적으로 설치 프로세스 중에 일곱 개의 가상 시스템을 생성합니다. 먼저 나머지 OpenShift Container Platform 클러스터를 생성하는 동안 임시 서비스와 컨트롤 플레인 영역을 제공하

는 부트스트랩 가상 머신을 생성합니다. 설치 프로그램이 클러스터 생성을 완료하면 부트스트랩 시스템을 삭제하여 리소스를 확보합니다.

RHV 환경에서 가상 머신 수를 늘리면 그에 따라 리소스를 늘려야합니다.

요구사항

- RHV 버전은 4.4입니다.
- RHV 환경에는 상태가 Up인 데이터 센터가 하나 있습니다.
- RHV 데이터 센터에는 RHV 클러스터가 포함되어 있습니다.
- RHV 클러스터에는 다음과 같은 OpenShift Container Platform 클러스터 전용 리소스가 있습니다.
  - 최소 28개의 vCPU(설치 중 생성된 일곱 개의 가상 시스템마다 각각 4개)
  - 다음을 포함한 112GiB RAM 이상
    - 임시 컨트롤 플레인을 제공하는 부트스트랩 시스템의 경우 16GiB 이상
    - 컨트롤 플레인을 제공하는 컨트롤 플레인 시스템 세 개 각각에 대해 16GiB 이상
    - 애플리케이션 워크로드를 실행하는 컴퓨팅 시스템 세 개 각각에 대해 16GiB 이상
- RHV 스토리지 도메인은 이러한 etcd 백엔드 성능 요구사항을 충족해야 합니다.
- 프로덕션 환경에서 각 가상 머신은 120GiB 이상이어야 합니다. 따라서 스토리지 도메인은 기본 OpenShift Container Platform 클러스터에 대해 840GiB 이상을 제공해야 합니다. 리소스가 제한적인 환경이나 프로덕션 이외의 환경에서는 각 가상 시스템에 32GiB 이상이 있어야 하므로

스토리지 도메인에는 기본 **OpenShift Container Platform** 클러스터에 필요한 **230GiB** 이상이 있어야 합니다.

- 설치 및 업데이트 중에 **Red Hat Ecosystem Catalog**에서 이미지를 다운로드하려면 **RHV** 클러스터가 인터넷 연결에 액세스할 수 있어야 합니다. **Telemetry** 서비스에는 서브스크립션 및 권한 부여 프로세스를 단순화하기 위해 인터넷 연결이 필요합니다.

- **RHV** 클러스터에는 **RHV Manager**의 **REST API**에 액세스할 수 있는 가상 네트워크가 있어야 합니다. 설치 관리자가 생성한 **VM**에서 **DHCP**를 사용하여 **IP** 주소를 얻을 수 있으므로 이 네트워크에서 **DHCP**가 활성화되어 있는지 확인합니다.

- 대상 **RHV** 클러스터에서 **OpenShift Container Platform** 클러스터를 설치 및 관리하기 위한 다음과 같은 최소 권한이있는 사용자 계정 및 그룹:

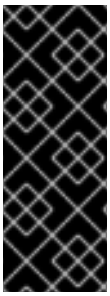
  - **DiskOperator**
  - **DiskCreator**
  - **UserTemplateBasedVm**
  - **TemplateOwner**
  - **TemplateCreator**
  - 대상 클러스터의 **ClusterAdmin**

주의

최소 권한 원칙을 적용합니다. 설치 과정에서 **RHV**에 대한 **SuperUser** 권한이 있는 관리자 계정을 사용하지 않도록 합니다. 설치 프로그램은 사용자가 제공한 인증 정보를 손상된 임시 **ovirt-config.yaml** 파일에 저장합니다.

16.4.4. RHV 환경에 대한 요구사항 확인

RHV 환경이 OpenShift Container Platform 클러스터 설치 및 실행 요구사항을 충족하는지 확인합니다. 이러한 요구사항을 충족하지 않으면 실패할 수 있습니다.



중요

이러한 요구사항은 설치 프로그램이 컨트롤 플레인과 컴퓨팅 시스템을 생성하는 데 사용하는 기본 리소스를 기반으로 합니다. 이러한 리소스에는 vCPU, 메모리 및 스토리지가 포함됩니다. 이러한 리소스를 변경하거나 OpenShift Container Platform 시스템 수를 늘리는 경우에는 그에 따라 이 요구사항을 조정합니다.

절차

1. RHV 버전이 OpenShift Container Platform 버전 4.9의 설치를 지원하는지 확인합니다.
   a. RHV 관리 포털에서 오른쪽 상단에 있는 ? 도움말 아이콘을 클릭하고 정보를 선택합니다.
   b. 창이 열리면 RHV 소프트웨어 버전을 기록합니다.
   c. RHV 버전이 4.4인지 확인합니다. 지원되는 버전 조합에 대한 자세한 내용은 RHV의 OpenShift Container Platform에 대한 지원 매트릭스를 참조하십시오.
2. 데이터 센터, 클러스터 및 스토리지를 검사합니다.
   a. RHV 관리 포털에서 Compute → Data Centers를 클릭합니다.
   b. OpenShift Container Platform을 설치하려는 데이터 센터에 액세스할 수 있는지 확인합니다.
   c. 해당 데이터 센터의 이름을 클릭합니다.
   d. 데이터 센터 세부 사항의 스토리지 탭에서 OpenShift Container Platform을 설치하려는 스토리지 도메인이 활성인지 확인합니다.

e. 나중에 사용할 수 있도록 도메인 이름을 기록합니다.

f. 여유 공간이 **230GiB** 이상인지 확인합니다.

g. 스토리지 도메인이 **fio** 성능 벤치마킹 툴을 사용하여 측정할 수 있는 이러한 **etcd** 백엔드 성능 요구사항을 충족하는지 확인합니다.

h. 데이터 센터 세부 사항에서 클러스터 탭을 클릭합니다.

i. **OpenShift Container Platform**을 설치할 **RHV** 클러스터를 찾습니다. 나중에 사용할 수 있도록 클러스터 이름을 기록합니다.

3. **RHV** 호스트 리소스를 검사합니다.

a. **RHV** 관리 포털에서 컴퓨팅 > 클러스터를 클릭합니다.

b. **OpenShift Container Platform**을 설치할 클러스터를 클릭합니다.

c. 클러스터 세부 사항에서 호스트 탭을 클릭합니다.

d. 호스트를 검사하고 **OpenShift Container Platform** 클러스터 *전용*으로 사용할 수 있는 총 **28**개 이상의 논리 **CPU** 코어가 있는지 확인합니다.

e. 나중에 사용할 수 있도록 사용 가능한 논리 **CPU** 코어 수를 기록합니다.

f. 설치 중에 생성된 **7**개의 가상 시스템 각각에 **4**개의 코어가 있을 수 있도록 이러한 **CPU** 코어가 분산되어 있는지 확인합니다.

g. 다음 **OpenShift Container Platform** 시스템 각각에 대한 요구사항을 충족하기 위해 배포된 새 가상 시스템 예약에 필요한 **112GiB**의 최대 여유 메모리가 모두 호스트에 있는지 확

인합니다.

- 부트스트랩 시스템에 필요한 16GiB
- 세 개의 컨트롤 플레인 시스템 각각에 필요한 16GiB
- 세 개의 컴퓨팅 시스템 각각에 대해 16GiB

h. 나중에 사용할 수 있도록 새 가상 시스템 예약에 필요한 최대 여유 메모리의 양을 기록합니다.

4. OpenShift Container Platform을 설치할 가상 네트워크가 RHV Manager의 REST API에 액세스할 수 있는지 확인합니다. 이 네트워크의 가상 시스템에서 RHV 관리자의 REST API에 도달하기 위해 curl을 사용합니다.

```
$ curl -k -u <username>@<profile>:<password> \ 1
https://<engine-fqdn>/ovirt-engine/api 2
```

1 <username> 의 경우 RHV에서 OpenShift Container Platform 클러스터를 만들고 관리할 수 있는 권한이 있는 RHV 계정의 사용자 이름을 지정합니다. <profile>은 로그인 프로파일을 지정합니다(RHV 관리 포털 로그인 페이지로 이동하여 프로파일 드롭다운 목록 검토). <password>의 경우 해당 사용자 이름에 대한 암호를 지정합니다.

2 <engine-fqdn>은 RHV 환경의 정규화된 도메인 이름을 지정합니다.

예를 들면 다음과 같습니다.

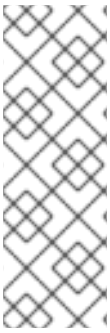
```
$ curl -k -u ocpadmin@internal:pw123 \
https://rhv-env.virtlab.example.com/ovirt-engine/api
```

16.4.5. 사용자 프로비저닝 인프라에 대한 네트워킹 요구사항

모든 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템이 부팅 중에 Ignition 구성 파일을 가져오려면 initramfs에 네트워킹을 구성해야 합니다.

초기 부팅 과정에서 시스템에 필요한 부팅 옵션을 제공하여 **DHCP** 서버를 통해 설정하거나 정적으로 설정하는 **IP** 주소 구성이 필요합니다. 네트워크 연결이 설정된 후 시스템은 **HTTP** 또는 **HTTPS** 서버에서 **Ignition** 구성 파일을 다운로드합니다. 그런 다음 **Ignition** 구성 파일을 사용하여 각 머신의 정확한 상태를 설정합니다. **Machine Config Operator**는 설치 후 새 인증서 또는 키 적용과 같은 머신에 대한 추가 변경을 완료합니다.

클러스터 시스템의 장기적인 관리를 위해 **DHCP** 서버를 사용하는 것이 좋습니다. **DHCP** 서버가 클러스터 시스템에 영구 **IP** 주소, **DNS** 서버 정보 및 호스트 이름을 제공하도록 구성되었는지 확인합니다.



참고

사용자 프로비저닝 인프라에 **DHCP** 서비스를 사용할 수 없는 경우 **RHCOS** 설치 시 노드에 **IP** 네트워킹 구성과 **DNS** 서버의 주소를 대신 제공할 수 있습니다. **ISO** 이미지에서 설치하는 경우 부팅 인수로 전달할 수 있습니다. 고정 **IP** 프로비저닝 *및 고급 네트워킹 옵션에 대한 자세한 내용은 **RHCOS** 설치 및 **OpenShift Container Platform** 부트스트랩 프로세스 시작* 섹션을 참조하십시오.

Kubernetes API 서버가 클러스터 시스템의 노드 이름을 확인할 수 있어야 합니다. **API** 서버와 작업자 노드가 서로 다른 영역에 있는 경우, **API** 서버가 노드 이름을 확인할 수 있도록 기본 **DNS** 검색 영역을 설정할 수 있습니다. 노드 개체와 모든 **DNS** 요청에서 항상 정규화된 도메인 이름으로 호스트를 가리키는 것도 지원되는 방법입니다

방화벽

클러스터가 필수 사이트에 액세스할 수 있도록 방화벽을 구성하십시오.

또한 다음을 참조하십시오.

- Red Hat Virtualization Manager 방화벽 요구 사항
- 호스트 방화벽 요구 사항

로드 밸런서

하나 또는 두 개의 계층-4 로드 밸런서를 구성합니다.

- 컨트롤 플레인 및 부트스트랩 머신에서 포트 **6443** 및 **22623**에 대한 로드 밸런싱을 제공합니다. 포트 **6443** 은 **Kubernetes API** 서버에 대한 액세스를 제공하며 내부 및 외부 모두에 연결할 수 있어야합니다. 포트 **22623**은 클러스터 내부 노드에서 액세스할 수 있어야 합니다.

- 일반적으로 기본 구성의 계산 노드인 **Ingress** 라우터를 실행하는 시스템에 포트 **443** 및 **80**의 로드 밸런싱을 제공합니다. 두 포트 모두 클러스터 내부와 외부에서 액세스할 수 있어야합니다.

DNS

기본 구성 요소 및 서비스를 올바르게 확인할 수 있도록 인프라 제공 **DNS**를 구성합니다. 로드 밸런서를 하나만 사용하는 경우 이러한 **DNS** 레코드는 동일한 **IP** 주소를 가리킬 수 있습니다.

- 컨트롤 플레인 시스템의 로드 밸런서를 가리키는 **api.<cluster_name>.<base_domain> (internal and external resolution)** 및 **api-int.<cluster_name>.<base_domain> (internal resolution)**에 대한 **DNS** 레코드를 생성합니다.

- **Ingress** 라우터의 로드 밸런서를 가리키는 ***.apps.<cluster_name>.<base_domain>**의 **DNS** 레코드를 생성합니다. 예를 들어 컴퓨팅 머신의 포트 **443** 및 **80**입니다.

16.4.5.1. DHCP를 통해 클러스터 노드의 호스트 이름 설정

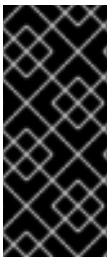
RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 호스트 이름은 **NetworkManager**를 통해 설정됩니다. 기본적으로 시스템은 **DHCP**를 통해 호스트 이름을 가져옵니다. **DHCP**에서 호스트 이름을 제공하지 않거나, 커널 인수를 통해 정적으로 설정하거나 다른 방법을 통해 설정하면 역방향 **DNS** 조회를 통해 얻을 수 있습니다. 역방향 **DNS** 조회는 노드에서 네트워크를 초기화한 후 수행되며 확인하는 데 시간이 걸릴 수 있습니다. 다른 시스템 서비스는 이 보다 먼저 시작하여 호스트 이름을 **localhost** 등으로 감지할 수 있습니다. **DHCP**를 사용하여 각 클러스터 노드의 호스트 이름을 제공하여 이 문제를 방지할 수 있습니다.

또한 **DHCP**를 통해 호스트 이름을 설정하면 **DNS** 분할 수평 구현 환경에서 수동으로 **DNS** 레코드 이름 구성 오류를 무시할 수 있습니다.

16.4.5.2. 네트워크 연결 요구사항

OpenShift Container Platform 클러스터 구성 요소가 통신할 수 있도록 시스템 간 네트워크 연결을 구성해야 합니다. 각 시스템에서 클러스터에 있는 다른 모든 시스템의 호스트 이름을 확인할 수 있어야 합니다.

이 섹션에서는 필요한 포트에 대해 자세히 설명합니다.



중요

연결된 **OpenShift Container Platform** 환경에서 모든 노드는 플랫폼 컨테이너의 이미지를 가져오고 **Red Hat**에 원격 측정 데이터를 제공하기 위해 인터넷에 액세스할 수 있어야 합니다.

표 16.6. 모든 시스템 간 통신에 사용되는 포트

프로토콜	포트	설명
ICMP	해당 없음	네트워크 연결성 테스트
TCP	**1936**	메트릭
	9000-**9999**	**9100**-**9101** 포트의 노드 내보내기 및 **9099** 포트의 Cluster Version Operator를 포함한 호스트 수준 서비스.
	10250-**10259**	Kubernetes에서 예약하는 기본 포트
	10256	openshift-sdn
UDP	**4789**	VXLAN 및 Geneve
	6081	VXLAN 및 Geneve
	9000-**9999**	**9100**-**9101** 포트의 노드 내보내기를 포함한 호스트 수준 서비스.
	500	IPsec IKE 패킷
	4500	IPsec NAT-T 패킷
TCP/UDP	**30000**-**32767**	Kubernetes 노드 포트
ESP	해당 없음	IPsec Encapsulating Security Payload (ESP)

표 16.7. 모든 시스템과 컨트롤 플레인 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**6443**	Kubernetes API

표 16.8. 컨트롤 플레인 머신 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**2379-2380**	etcd 서버 및 피어 포트

사용자 프로비저닝 인프라에 대한 NTP 구성

OpenShift Container Platform 클러스터는 기본적으로 공용 **NTP(Network Time Protocol)** 서버를 사용하도록 구성되어 있습니다. 로컬 엔터프라이즈 **NTP** 서버를 사용하거나 클러스터가 연결이 끊긴 네트워크에 배포되는 경우 특정 시간 서버를 사용하도록 클러스터를 구성할 수 있습니다. 자세한 내용은 ***chrony** 타임 서비스 설정* 문서를 참조하십시오.

DHCP 서버가 **NTP** 서버 정보를 제공하는 경우 **RHCOS(Red Hat Enterprise Linux CoreOS)** 시스템의 **chrony** 타임 서비스에서 정보를 읽고 **NTP** 서버와 클럭을 동기화할 수 있습니다.

16.4.6. 설치 시스템 설정

바이너리 **openshift-install** 설치 프로그램 및 **Ansible** 스크립트를 실행하려면 **RHV Manager** 또는 **RHV** 환경에 대한 네트워크 액세스 권한이 있는 **RHEL(Red Hat Enterprise Linux)** 컴퓨터와 **Manager**에서 **REST API**를 설정합니다.

절차

1. **Python3** 및 **Ansible**을 업데이트하거나 설치합니다. 예를 들면 다음과 같습니다.

```
# dnf update python3 ansible
```

2. python3-ovirt-engine-sdk4 패키지를 설치하여 **Python** 소프트웨어 개발 키트를 가져옵니다.

3. **ovirt.image-template Ansible** 역할을 설치합니다. **RHV Manager** 및 기타 **RHEL(Red Hat Enterprise Linux)** 시스템에서 이 역할은 **ovirt-ansible-image-template** 패키지로 배포됩니다. 예를 들면 다음과 같습니다.

```
# dnf install ovirt-ansible-image-template
```

4. **ovirt.vm-infra Ansible** 역할을 설치합니다. **RHV Manager** 및 기타 **RHEL** 시스템에서 이 역할은 **ovirt-ansible-vm-infra** 패키지로 배포됩니다.

```
# dnf install ovirt-ansible-vm-infra
```

5. 환경 변수를 만들고 절대 또는 상대 경로를 할당합니다. 예를 들면 다음과 같습니다.

```
$ export ASSETS_DIR=./wrk
```

참고

설치 프로그램은 이 변수를 사용하여 중요한 설치 관련 파일을 저장하는 디렉터리를 만듭니다. 나중에 설치 프로세스는 이 변수를 사용하여 해당 자산 파일을 찾습니다. 이 **assets** 디렉터리를 삭제하지 마십시오. 이는 클러스터를 제거하는 데 필요합니다.

16.4.7. 비보안 모드에서 RHV에 OpenShift Container Platform 설치

기본적으로 설치 관리자는 **CA** 인증서를 생성하고 확인을 요청한 후 설치 중에 사용할 인증서를 저장합니다. 수동으로 생성하거나 설치할 필요가 없습니다.

권장되지는 않지만 비보안 모드로 **OpenShift Container Platform**을 설치하여 인증서를 확인하지 않고도 이 기능을 재정의하고 **OpenShift Container Platform**을 설치할 수 있습니다.

주의

잠재적인 공격자가 중**Man-in-the-Middle** 공격을 수행할 수 있고 네트워크에서 중요한 인증 정보를 캡처할 수 있으므로 비보안 모드에 설치하는 것은 권장되지 않습니다.

절차

1. **~/.ovirt/ovirt-config.yaml**이라는 파일을 생성합니다.

2. **ovirt-config.yaml**에 다음 내용을 추가합니다.

```
ovirt_url: https://ovirt.example.com/ovirt-engine/api (1)
ovirt_fqdn: ovirt.example.com (2)
ovirt_pem_url: ""
```

```
ovirt_username: ocpadmin@internal
ovirt_password: super-secret-password 3
ovirt_insecure: true
```

1 oVirt 엔진의 호스트 이름 또는 주소를 지정합니다.

2 oVirt 엔진의 정규화된 도메인 이름을 지정합니다.

3 oVirt 엔진의 admin 암호를 지정합니다.

3. 설치 프로그램을 실행합니다.

16.4.8. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

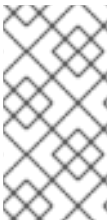
OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 core 사용자의 ~/.ssh/authorized_keys 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 core로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. ./openshift-install gather 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.



참고

AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

절차

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: ~/.ssh/id_ed25519)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 '~/.ssh 디렉터리에 있는지 확인하십시오.

참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 ed25519 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 rsa 또는 ecdsa 알고리즘을 사용하는 키를 생성합니다.

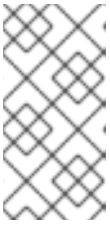
2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 ~/.ssh/id_ed25519.pub 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 ./openshift-install gather 명령을 사용하려는 경우 필요합니다.



참고

일부 배포에서는 ~/.ssh/id_rsa 및 ~/.ssh/id_dsa와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a.

ssh-agent 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4.

ssh-agent에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1

SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: ~/.ssh/id_ed25519).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

16.4.9. 설치 프로그램 받기

OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

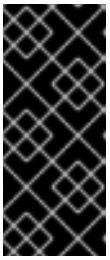
- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

절차

1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.

2. 인프라 공급자를 선택합니다.

3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.



중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.

중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 OpenShift Container Platform 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

16.4.10. Ansible Playbook 다운로드

RHV에 OpenShift Container Platform 4.9 버전을 설치하기 위한 Ansible 플레이북을 다운로드합니다.

절차

- 설치 시스템에서 다음 명령을 실행하십시오.

```
$ mkdir playbooks
```

```
$ cd playbooks
```

```
$ curl -s -L -X GET https://api.github.com/repos/openshift/installer/contents/upi/ovirt?
ref=release-4.9 |
grep 'download_url.*\.yml' |
awk '{ print $2 }' | sed -r 's/("|",)//g' |
xargs -n 1 curl -O
```

다음 단계

- 이러한 Ansible Playbook을 다운로드한 후 설치 프로그램을 실행하여 설치 구성 파일을 생성하기 전에 assets 디렉터리에 대한 환경 변수를 생성하고 inventory.yml 파일을 사용자 정의해야 합니다.

16.4.11. inventory.yml 파일

inventory.yml 파일을 사용하여 설치하려는 OpenShift Container Platform 클러스터의 요소를 정의하고 생성합니다. 여기에는 RHCOS (Red Hat Enterprise Linux CoreOS) 이미지, 가상 머신 템플릿, 부트스트랩 머신, 컨트롤 플레인 노드 및 작업자 노드와 같은 요소가 포함됩니다. 또한 inventory.yml을 사용하여 클러스터를 제거합니다.

다음 inventory.yml에에서는 매개 변수와 매개 변수 기본값을 보여줍니다. 이러한 기본값의 수량과 숫자는 RHV 환경에서 프로덕션 OpenShift Container Platform 클러스터를 실행하기 위한 요구 사항을 충족합니다.

inventory.yml 파일 예

```
---
all:
  vars:

    ovirt_cluster: "Default"
    ocp:
      assets_dir: "{{ lookup('env', 'ASSETS_DIR') }}"
      ovirt_config_path: "{{ lookup('env', 'HOME') }}/.ovirt/ovirt-config.yaml"

    # ---
    # {op-system} section
    # ---
    rhcos:
      image_url: "https://mirror.openshift.com/pub/openshift-
v4/dependencies/rhcos/4.9/latest/rhcos-openstack.x86_64.qcow2.gz"
      local_cmp_image_path: "/tmp/rhcos.qcow2.gz"
      local_image_path: "/tmp/rhcos.qcow2"

    # ---
    # Profiles section
    # ---
    control_plane:
      cluster: "{{ ovirt_cluster }}"
      memory: 16GiB
      sockets: 4
      cores: 1
      template: rhcos_tpl
      operating_system: "rhcos_x64"
      type: high_performance
      graphical_console:
        headless_mode: false
        protocol:
        - spice
        - vnc
      disks:
      - size: 120GiB
        name: os
        interface: virtio_scsi
        storage_domain: depot_nvme
      nics:
      - name: nic1
        network: lab
        profile: lab

    compute:
      cluster: "{{ ovirt_cluster }}"
      memory: 16GiB
      sockets: 4
      cores: 1
      template: worker_rhcos_tpl
      operating_system: "rhcos_x64"
```

```
  type: high_performance
  graphical_console:
    headless_mode: false
    protocol:
    - spice
    - vnc
  disks:
  - size: 120GiB
    name: os
    interface: virtio_scsi
    storage_domain: depot_nvme
  nics:
  - name: nic1
    network: lab
    profile: lab

# ---
# Virtual machines section
# ---
vms:
- name: "{{ metadata.infraID }}-bootstrap"
  ocp_type: bootstrap
  profile: "{{ control_plane }}"
  type: server
- name: "{{ metadata.infraID }}-master0"
  ocp_type: master
  profile: "{{ control_plane }}"
- name: "{{ metadata.infraID }}-master1"
  ocp_type: master
  profile: "{{ control_plane }}"
- name: "{{ metadata.infraID }}-master2"
  ocp_type: master
  profile: "{{ control_plane }}"
- name: "{{ metadata.infraID }}-worker0"
  ocp_type: worker
  profile: "{{ compute }}"
- name: "{{ metadata.infraID }}-worker1"
  ocp_type: worker
  profile: "{{ compute }}"
- name: "{{ metadata.infraID }}-worker2"
  ocp_type: worker
  profile: "{{ compute }}"
```

중요

매개 변수 설명이 **"Enter"**로 시작하는 매개 변수의 값을 입력합니다. 그렇지 않으면 기본값을 사용하거나 새 값으로 변경할 수 있습니다.

일반 섹션

- **ovirt_cluster: OpenShift Container Platform** 클러스터를 설치할 기존 **RHV** 클러스터의 이름을 입력합니다.

- **ocp.assets_dir: openshift-install** 설치 프로그램이 생성하는 파일을 저장하기 위해 생성하는 디렉터리의 경로입니다.

- **ocp.ovirt_config_path:** 설치 프로그램이 생성하는 **ovirt-config.yaml** 파일의 경로 **(예: ./wrk/install-config.yaml)**입니다. 이 파일에는 **Manager**의 **REST API**와 상호 작용하는 데 필요한 인증 정보가 포함되어 있습니다.

RHCOS(Red Hat Enterprise Linux CoreOS) 섹션

- **image_url:** 다운로드를 위해 지정한 **RHCOS** 이미지의 **URL**을 입력합니다.

- **local_cmp_image_path:** 압축된 **RHCOS** 이미지에 대한 로컬 다운로드 디렉터리의 경로입니다.

- **local_image_path:** 추출된 **RHCOS** 이미지에 대한 로컬 디렉터리의 경로입니다.

프로필 섹션

이 섹션은 두 가지 프로필로 구성됩니다.

- **control_plane:** 부트스트랩 및 컨트롤 플레인 노드의 프로필입니다.

- **compute:** 컴퓨팅 플레인에 있는 작업자 노드의 프로필입니다.

이러한 프로필에는 다음과 같은 매개 변수가 있습니다. 매개 변수의 기본값은 프로덕션 클러스터 실행을 위한 최소 요구 사항을 충족합니다. 이러한 값을 증가시키거나 사용자 지정하여 워크로드 요구 사항을 충족할 수 있습니다.

- **cluster:** 이 값은 일반 섹션의 **ovirt_cluster**에서 클러스터 이름을 가져옵니다.

- **memory:** 가상 시스템의 메모리 양(GB)입니다.

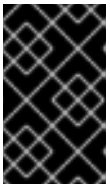
- **sockets:** 가상 시스템의 소켓 수입니다.

- **cores:** 가상 시스템의 코어 수입니다.

- **template:** 가상 시스템 템플릿의 이름입니다. 여러 클러스터를 설치할 계획이고 이러한 클러스터가 서로 다른 사양을 포함하는 템플릿을 사용하는 경우 템플릿 이름 앞에 클러스터 **ID**를 추가합니다.

- **operating_system:** 가상 시스템의 게스트 운영 체제 유형입니다. **oVirt/RHV** 버전 **4.4**에서 이 값은 **rhcos_x64**이어야 하므로 **Ignition script**의 값이 가상 시스템에 전달될 수 있습니다.

- **type:** 가상 머신의 유형으로 **server**를 입력합니다.



중요

type 매개 변수의 값을 **high_performance**에서 **server**로 변경해야 합니다.

- **disks:** 디스크 사양. **control_plane** 및 **compute** 노드는 서로 다른 스토리지 도메인을 가질 수 있습니다.

- **size:** 최소 디스크 크기.

- **name: RHV**에서 대상 클러스터에 연결된 디스크의 이름을 입력합니다.

- **interface:** 지정한 디스크의 인터페이스 유형을 입력합니다.

- **storage_domain:** 지정한 디스크의 스토리지 도메인을 입력합니다.

- **nics:** 가상 머신이 사용하는 **name** 및 **network**를 입력합니다. 가상 네트워크 인터페이스 프

로필을 지정할 수도 있습니다. 기본적으로 NIC는 oVirt/RHV MAC 풀에서 MAC 주소를 가져옵니다.

가상 머신 섹션

이 마지막 섹션인 vms는 클러스터에서 생성하고 배포할 가상 시스템을 정의합니다. 기본적으로 프로덕션 환경에 대해 최소한의 컨트롤 플레인 및 작업자 노드를 제공합니다.

vms에는 세 가지 필수 요소가 있습니다.

- name: 가상 머신의 이름입니다. 이 경우 metadata.infraID는 가상 머신 이름 앞에 metadata.yml 파일의 인프라 ID를 추가합니다.
- ocp_type: OpenShift Container Platform 클러스터에서 가상 머신의 역할입니다. 가능한 값은 bootstrap, master, worker입니다.
- profile: 각 가상 머신이 사양을 상속하는 프로필의 이름입니다. 이 예제에서 가능한 값은 control_plane 또는 compute입니다.

  가상 머신이 해당 프로필에서 상속하는 값을 재정의할 수 있습니다. 이렇게 하려면 inventory.yml의 가상 머신에 프로필 속성의 이름을 추가하고 재정의 값을 할당합니다. 이 예제를 보려면 이전 inventory.yml 예제에서 name: "{{ metadata.infraID }}-bootstrap" 가상 머신을 검사합니다. 여기에는 server 값이 있는 type 속성이 이 가상 머신이 control_plane 프로필에서 상속하는 type 속성 값을 재정의합니다.

메타 데이터 변수

가상 머신의 경우 metadata.infraID는 Ignition 파일을 빌드할 때 생성한 metadata.json 파일의 인프라 ID를 가상 머신의 이름 앞에 추가합니다.

Playbook은 ocp.assets_dir에 있는 특정 파일에서 infraID 읽고 다음 코드를 사용합니다.

```
---
- name: include metadata.json vars
  include_vars:
    file: "{{ ocp.assets_dir }}/metadata.json"
    name: metadata

  ...
```

16.4.12. RHCOS 이미지 설정 지정

inventory.yml 파일의 RHCOS (Red Hat Enterprise Linux CoreOS) 이미지 설정을 업데이트합니다. 나중에 Playbook 중 파일 하나를 실행하면 image_url URL에서 local_cmp_image_path 디렉터리로 압축된 Red Hat Enterprise 리눅스 CoreOS(RHCOS) 이미지가 다운로드됩니다. 그런 다음 Playbook은 이미지를 local_image_path 디렉터리에 압축 해제하고 이를 사용하여 oVirt/RHV 템플릿을 만듭니다.

절차

1. 설치 중인 OpenShift Container Platform 버전에 대한 RHCOS 이미지 다운로드 페이지(예: Index of /pub/openshift-v4/dependencies/rhcos/latest/latest)를 찾습니다.

2. 해당 다운로드 페이지에서 https://mirror.openshift.com/pub/openshift-v4/dependencies/rhcos/4.9/latest/rhcos-openstack.x86_64.qcow2.gz와 같은 OpenStack qcow2 이미지의 URL을 복사합니다.

3. 이전에 다운로드 한 inventory.yml Playbook을 편집합니다. 여기에 URL을 image_url 값으로 붙여 넣습니다. 예를 들면 다음과 같습니다.

```
rhcos:
  "https://mirror.openshift.com/pub/openshift-v4/dependencies/rhcos/4.9/latest/rhcos-openstack.x86_64.qcow2.gz"
```

16.4.13. 설치 구성 파일 만들기

설치 프로그램 openshift-install을 실행하고 이전에 지정했거나 수집한 정보로 해당 프롬프트에 응답하여 설치 구성 파일을 만듭니다.

프롬프트에 대한 응답을 마치면 설치 프로그램이 이전에 지정한 assets 디렉터리 (예 : ./wrk/install-config.yaml)에 install-config.yaml 파일의 초기 버전을 생성합니다.

설치 프로그램은 $HOME/.ovirt/ovirt-config.yaml 파일을 생성합니다. 이 파일에는 Manager에 연결하여 REST API를 사용하는 데 필요한 모든 연결 매개 변수가 포함되어 있습니다.

참고: 설치 프로세스는 Internal API virtual IP 및 Ingress virtual IP와 같은 일부 매개 변수에 제공하는 값을 사용하지 않습니다. 이미 인프라 DNS에서 구성했기 때문입니다.

또한 oVirt cluster, oVirt storage 및 oVirt network에 대한 값과 같이 inventory.yml의 매개 변수에

제공하는 값을 사용합니다. 그리고 스크립트를 사용하여 **install-config.yaml**에서 이러한 동일한 값을 이전에 언급한 **virtual IP**로 제거하거나 변경합니다.

절차

1. 설치 프로그램을 실행합니다.

```
$ openshift-install create install-config --dir $ASSETS_DIR
```

2. 시스템에 대한 정보를 사용하여 설치 프로그램의 프롬프트 메시지에 응답합니다.

출력 예

```
? SSH Public Key /home/user/.ssh/id_dsa.pub
? Platform <ovirt>
? Engine FQDN[:PORT] [? for help] <engine.fqdn>
? Enter ovirt-engine username <ocpadmin@internal>
? Enter password <******>
? oVirt cluster <cluster>
? oVirt storage <storage>
? oVirt network <net>
? Internal API virtual IP <172.16.0.252>
? Ingress virtual IP <172.16.0.251>
? Base Domain <example.org>
? Cluster Name <ocp4>
? Pull Secret [? for help] <********>
```

```
? SSH Public Key /home/user/.ssh/id_dsa.pub
? Platform <ovirt>
? Engine FQDN[:PORT] [? for help] <engine.fqdn>
? Enter ovirt-engine username <ocpadmin@internal>
? Enter password <******>
? oVirt cluster <cluster>
? oVirt storage <storage>
? oVirt network <net>
? Internal API virtual IP <172.16.0.252>
? Ingress virtual IP <172.16.0.251>
? Base Domain <example.org>
? Cluster Name <ocp4>
? Pull Secret [? for help] <********>
```

Internal API virtual IP 및 **Ingress virtual IP**의 경우 **DNS** 서비스를 구성할 때 지정한 **IP** 주소를 제공

합니다.

oVirt cluster 및 **Base Domain** 프롬프트에서 입력한 값은 **REST API** 및 생성하는 모든 애플리케이션의 **URL**의 일부를 구성합니다. **(예: https://api.ocp4.example.org:6443/** 및 **https://console-openshift-console.apps.ocp4.example.org)**

Red Hat OpenShift Cluster Manager에서 풀 시크릿을 가져올 수 있습니다.

16.4.14. install-config.yaml 사용자 정의

여기에서는 **3** 개의 **Python** 스크립트를 사용하여 설치 프로그램의 일부 기본 동작을 재정의합니다.

- 기본적으로 설치 프로그램은 머신 **API**를 사용하여 노드를 만듭니다. 이 기본 동작을 재정의하려면 컴퓨팅 노드 수를 복제본 **0**으로 설정합니다. 나중에 **Anable Playbook**을 사용하여 컴퓨팅 노드를 생성합니다.
- 기본적으로 설치 프로그램은 노드의 시스템 네트워크의 **IP** 범위를 설정합니다. 이 기본 동작을 재정의하려면 인프라와 일치하도록 **IP** 범위를 설정합니다.
- 기본적으로 설치 프로그램은 플랫폼을 **ovirt**로 설정합니다. 그러나 사용자 프로비저닝 인프라에 클러스터를 설치하는 것은 베어 메탈에 클러스터를 설치하는 것과 비슷합니다. 따라서 **install-config.yaml**에서 **ovirt** 플랫폼 섹션을 삭제하고 플랫폼을 **none**으로 변경합니다. 대신 **inventory.yml**을 사용하여 모든 필수 설정을 지정합니다.

참고

이 스니펫은 **Python 3** 및 **Python 2**에서 작동합니다.

절차

1. 컴퓨팅 노드 수를 복제본 **0**으로 설정합니다.

```
$ python3 -c 'import os, yaml
path = "%s/install-config.yaml" % os.environ["ASSETS_DIR"]
conf = yaml.safe_load(open(path))
conf["compute"][0]["replicas"] = 0
open(path, "w").write(yaml.dump(conf, default_flow_style=False))'
```

2. 머신 네트워크의 IP 범위를 설정합니다. 예를 들어 범위를 172.16.0.0/16으로 설정하려면 다음을 입력합니다.

```
$ python3 -c 'import os, yaml
path = "%s/install-config.yaml" % os.environ["ASSETS_DIR"]
conf = yaml.safe_load(open(path))
conf["networking"]["machineNetwork"][0]["cidr"] = "172.16.0.0/16"
open(path, "w").write(yaml.dump(conf, default_flow_style=False))'
```

3. ovirt 섹션을 제거하고 플랫폼을 none으로 변경합니다.

```
$ python3 -c 'import os, yaml
path = "%s/install-config.yaml" % os.environ["ASSETS_DIR"]
conf = yaml.safe_load(open(path))
platform = conf["platform"]
del platform["ovirt"]
platform["none"] = {}
open(path, "w").write(yaml.dump(conf, default_flow_style=False))'
```

주의

Red Hat Virtualization은 현재 oVirt 플랫폼에서 사용자 프로비저닝 인프라를 사용하여 설치를 지원하지 않습니다. 따라서 플랫폼을 none 으로 설정해야 OpenShift Container Platform에서 각 노드를 베어 메탈 노드로 식별하고 클러스터를 베어 메탈 클러스터로 식별할 수 있습니다. 이는 모든 플랫폼에 클러스터를 설치하는 것과 동일하며 다음과 같은 제한 사항이 있습니다.

1. 클러스터 공급자가 없으므로 각 머신을 수동으로 추가해야 하며 노드 확장 기능이 없습니다.

2. oVirt CSI 드라이버가 설치되지 않으며 CSI 기능이 없습니다.

16.4.15. 매니페스트 파일 생성

설치 프로그램을 사용하여 assets 디렉터리에 매니페스트 파일 세트를 생성하십시오.

매니페스트 파일을 생성하는 명령은 **install-config.yaml** 파일을 사용하기 전에 경고 메시지를 표시합니다.

install-config.yaml 파일을 재사용하려는 경우 매니페스트 파일을 생성하기 전에 해당 파일의 백업 사본을 생성합니다.

프로세스

1. 선택 사항: **install-config.yaml** 파일의 작업 사본을 만듭니다.

```
$ cp install-config.yaml install-config.yaml.backup
```

2. **assets** 디렉터리에 매니페스트 세트를 생성합니다.

```
$ openshift-install create manifests --dir $ASSETS_DIR
```

이 명령은 다음 메시지를 표시합니다.

출력 예

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true
for Scheduler cluster settings
```

이 명령은 다음 매니페스트 파일을 생성합니다.

출력 예

```
$ tree
.
└── wrk
    ├── manifests
    │   ├── 04-openshift-machine-config-operator.yaml
    │   ├── cluster-config.yaml
```

```
│   ├── cluster-dns-02-config.yml
│   ├── cluster-infrastructure-02-config.yml
│   ├── cluster-ingress-02-config.yml
│   ├── cluster-network-01-crd.yml
│   ├── cluster-network-02-config.yml
│   ├── cluster-proxy-01-config.yaml
│   ├── cluster-scheduler-02-config.yml
│   ├── cvo-overrides.yaml
│   ├── etcd-ca-bundle-configmap.yaml
│   ├── etcd-client-secret.yaml
│   ├── etcd-host-service-endpoints.yaml
│   ├── etcd-host-service.yaml
│   ├── etcd-metric-client-secret.yaml
│   ├── etcd-metric-serving-ca-configmap.yaml
│   ├── etcd-metric-signer-secret.yaml
│   ├── etcd-namespace.yaml
│   ├── etcd-service.yaml
│   ├── etcd-serving-ca-configmap.yaml
│   ├── etcd-signer-secret.yaml
│   ├── kube-cloud-config.yaml
│   ├── kube-system-configmap-root-ca.yaml
│   ├── machine-config-server-tls-secret.yaml
│   └── openshift-config-secret-pull-secret.yaml
└── openshift
    ├── 99_kubeadmin-password-secret.yaml
    ├── 99_openshift-cluster-api_master-user-data-secret.yaml
    ├── 99_openshift-cluster-api_worker-user-data-secret.yaml
    ├── 99_openshift-machineconfig_99-master-ssh.yaml
    ├── 99_openshift-machineconfig_99-worker-ssh.yaml
    └── openshift-install-manifests.yaml
```

다음 단계

- 컨트롤 플레인 노드를 예약 불가능하게 설정합니다.

16.4.16. 컨트롤 플레인 노드를 예약 불가능하게 설정하기

컨트롤 플레인 시스템을 수동으로 생성 및 배포하고 있으므로 컨트롤 플레인 노드를 예약할 수 없도록 매니페스트 파일을 구성해야 합니다.

프로세스

1. 컨트롤 플레인 노드를 스케줄 불가능하게하려면 다음을 입력합니다.

```
$ python3 -c 'import os, yaml
```

```
path = "%s/manifests/cluster-scheduler-02-config.yml" % os.environ["ASSETS_DIR"]
data = yaml.safe_load(open(path))
data["spec"]["mastersSchedulable"] = False
open(path, "w").write(yaml.dump(data, default_flow_style=False))'
```

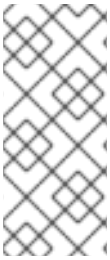
16.4.17. Ignition 파일 빌드하기

방금 생성 및 수정한 매니페스트 파일에서 **Ignition** 파일을 빌드하려면 설치 프로그램을 실행합니다. 이 작업은 **RHCOS (Red Hat Enterprise Linux CoreOS)** 시스템 **initramfs** 를 생성하여 **Ignition** 파일을 가져오고 노드를 만드는 데 필요한 구성을 수행합니다.

설치 프로그램은 **Ignition** 파일 외에도 다음을 생성합니다.

- **oc** 및 **kubectl** 유틸리티를 사용하여 클러스터에 연결하기위한 관리자 인증 정보가 포함된 **auth** 디렉터리입니다.
- 현재 설치에 대한 **OpenShift Container Platform** 클러스터 이름, 클러스터 **ID** 및 인프라 **ID** 와 같은 정보가 포함된 **metadata.json** 파일입니다.

설치 프로세스를 위한 **Ansible Playbook**은 생성한 가상 머신의 접두사로 **infraID** 값을 사용합니다. 이는 동일한 **oVirt/RHV** 클러스터에 여러 설치가 있을 때 이름 지정 충돌을 방지합니다.



참고

Ignition 구성 파일의 인증서는 **24** 시간 후에 만료됩니다. 첫 번째 인증서 교체가 완료될 수 있도록 클러스터 설치를 완료하고 성능이 저하되지 않은 상태에서 **24**시간 동안 클러스터를 계속 실행해야 합니다.

프로세스

1. **Ignition** 파일을 빌드하려면 다음을 입력합니다.

```
$ openshift-install create ignition-configs --dir $ASSETS_DIR
```

출력 예

```
$ tree
.
```

```
└── wrk
    ├── auth
    │   ├── kubeadmin-password
    │   └── kubeconfig
    ├── bootstrap.ign
    ├── master.ign
    ├── metadata.json
    └── worker.ign
```

16.4.18. 템플릿 및 가상 머신 생성

inventory.yml에서 변수를 확인한 후 첫 번째 **Ansible** 프로비저닝 **Playbook**인 **create-templates-and-vms.yml**을 실행합니다.

이 **Playbook**은 **$HOME/.ovirt/ovirt-config.yaml**의 **RHV Manager**에 대한 연결 매개 변수를 사용하고 **assets** 디렉터리에서 **metadata.json**을 읽습니다.

로컬 **Red Hat Enterprise Linux CoreOS (RHCOS)** 이미지가 존재하지 않는 경우 **Playbook**은 **inventory.yml**의 **image_url**에서 지정한 **URL**에서 해당 이미지를 다운로드합니다. 이미지를 추출하여 **RHV**에 업로드하여 템플릿을 생성합니다.

Playbook은 **inventory.yml** 파일의 **control_plane** 및 **compute** 프로필을 기반으로 템플릿을 생성합니다. 이러한 프로필의 이름이 다른 경우 두 개의 템플릿을 만듭니다.

Playbook이 완료되면 생성된 가상 머신이 중지됩니다. 다른 인프라 요소를 구성하는 데 도움이되는 정보를 얻을 수 있습니다. 예를 들어 가상 머신의 **MAC** 주소를 가져 와서 가상 머신에 영구 **IP** 주소를 할당하도록 **DHCP**를 구성할 수 있습니다.

프로세스

1. **inventory.yml** 의 **control_plane** 및 **compute** 변수에서 **type: high_performance**의 두 인스턴스를 **type: server**로 변경합니다.

2. 선택 사항: 동일한 클러스터에 여러번 설치를 수행하려는 경우 각 **OpenShift Container Platform** 설치에 대해 서로 다른 템플릿을 생성합니다. **inventory.yml** 파일에서 **template** 값 앞에 **infraID**를 추가합니다. 예를 들면 다음과 같습니다.

```
control_plane:
```

```
    cluster: "{{ ovirt_cluster }}"
    memory: 16GiB
    sockets: 4
    cores: 1
    template: "{{ metadata.infraID }}-rhcos_tpl"
    operating_system: "rhcos_x64"
    ...
```

3. 템플릿 및 가상 머신을 만듭니다.

```
$ ansible-playbook -i inventory.yml create-templates-and-vms.yml
```

16.4.19. 부트스트랩 시스템 생성

bootstrap.yml Playbook을 실행하여 부트스트랩 머신을 생성합니다. 이 Playbook은 부트스트랩 가상 머신을 시작하고 assets 디렉터리에서 bootstrap.ign Ignition 파일을 전달합니다. 부트스트랩 노드는 컨트롤 플레인 노드에 Ignition 파일을 제공할 수 있도록 자체적으로 구성됩니다.

부트스트랩 프로세스를 모니터링하려면 RHV 관리 포털의 콘솔을 사용하거나 SSH를 사용하여 가상 머신에 연결합니다.

프로세스

1. 부트스트랩 시스템을 생성합니다.

```
$ ansible-playbook -i inventory.yml bootstrap.yml
```

2. 관리 포털 또는 SSH에서 콘솔을 사용하여 부트스트랩 머신에 연결합니다. <bootstrap_ip>를 부트스트랩 노드 IP 주소로 바꿉니다. SSH를 사용하려면 다음을 입력합니다.

```
$ ssh core@<boostrap.ip>
```

3. 부트스트랩 노드에서 릴리스 이미지 서비스에 대한bootkube.service journald 장치 로그를 수집합니다.

```
[core@ocp4-lk6b4-bootstrap ~]$ journalctl -b -f -u release-image.service -u
bootkube.service
```

참고

부트스트랩 노드의 **bootkube.service** 로그는 **etcd connection rejectd** 오류를 출력하고 부트스트랩 서버가 컨트롤 플레인 노드의 **etcd**에 연결할 수 없음을 나타냅니다. 각 컨트롤 플레인 노드에서 **etcd**를 시작하고 노드가 클러스터에 가입되면 오류가 중지됩니다.

16.4.20. 컨트롤 플레인 노드 생성

masters.yml Playbook을 실행하여 컨트롤 플레인 노드를 생성합니다. 이 **Playbook**은 각 가상 머신에 **master.ign Ignition** 파일을 전달합니다. **Ignition** 파일에는 **https://api-int.ocp4.example.org:22623/config/master**와 같은 **URL**에서 **Ignition**을 가져 오는 컨트롤 플레인 노드에 대한 지시문이 포함되어 있습니다. 이 **URL**의 포트 번호는 로드 밸런서에 의해 관리되며 클러스터 내에서만 액세스할 수 있습니다.

프로세스

1. 컨트롤 플레인 노드를 생성합니다.

```
$ ansible-playbook -i inventory.yml masters.yml
```

2. **Playbook**이 컨트롤 플레인을 생성하는 동안 부트스트랩 프로세스를 모니터링합니다.

```
$ openshift-install wait-for bootstrap-complete --dir $ASSETS_DIR
```

출력 예

```
INFO API v1.22.1 up
INFO Waiting up to 40m0s for bootstrapping to complete...
```

3. 컨트롤 플레인 노드 및 **etcd**의 모든 **pod**가 실행 중이면 설치 프로그램에 다음과 같은 출력이 표시됩니다.

출력 예

```
INFO It is now safe to remove the bootstrap resources
```

16.4.21. 클러스터 상태 확인

설치 중 또는 설치 후 OpenShift Container Platform 클러스터의 상태를 확인할 수 있습니다.

프로세스

1. 클러스터 환경에서 관리자의 kubeconfig 파일을 내보냅니다.

```
$ export KUBECONFIG=$ASSETS_DIR/auth/kubeconfig
```

kubeconfig 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다.

2. 배포 후 생성된 컨트롤 플레인 및 컴퓨팅 시스템을 확인합니다.

```
$ oc get nodes
```

3. 클러스터 버전을 확인합니다.

```
$ oc get clusterversion
```

4. Operator 상태를 확인합니다.

```
$ oc get clusteroperator
```

5. 클러스터에서 실행 중인 모든 Pod를 확인합니다.

```
$ oc get pods -A
```

16.4.22. 부트스트랩 시스템 제거

wait-for 명령이 부트스트랩 프로세스가 완료되었음을 표시한 후 부트스트랩 가상 머신을 제거하여 컴퓨팅, 메모리 및 스토리지 리소스를 확보해야 합니다. 또한 로드 밸런서 지시문에서 부트스트랩 시스템 설정을 제거해야 합니다.

프로세스

1. 클러스터에서 부트스트랩 시스템을 제거하려면 다음을 입력하십시오.

```
$ ansible-playbook -i inventory.yml retire-bootstrap.yml
```

2. 로드 밸런서 지시문에서 부트스트랩 머신의 설정을 제거합니다.

16.4.23. 작업자 노드 작성 및 설치 완료

작업자 노드 생성은 컨트롤 플레인 노드를 생성하는 것과 유사합니다. 그러나 작업자 노드는 클러스터에 자동으로 참여하지 않습니다. 클러스터에 추가하려면 작업자의 보류중인 CSR (인증서 서명 요청)을 검토하고 승인합니다.

첫 번째 요청을 승인한 후 모든 작업자 노드가 승인될 때까지 CSR을 계속 승인합니다. 이 프로세스를 완료하면 작업자 노드가 Ready 상태가 되고 해당 노드에서 실행되도록 Pod를 예약할 수 있습니다.

마지막으로 명령줄을 모니터링하여 설치 프로세스가 완료되는지 확인합니다.

프로세스

1. 작업자 노드를 생성합니다.

```
$ ansible-playbook -i inventory.yml workers.yml
```

2. 모든 CSR을 나열하려면 다음을 입력하십시오.

```
$ oc get csr -A
```

결국이 명령은 노드 당 하나의 CSR을 표시합니다. 예를 들면 다음과 같습니다.

출력 예

```
NAME        AGE    SIGNERNAME                                    REQUESTOR
CONDITION
csr-2lnxd   63m    kubernetes.io/kubelet-serving                 system:node:ocp4-lk6b4-
master0.ocp4.example.org                 Approved,Issued
csr-hff4q   64m    kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper
Approved,Issued
csr-hsn96   60m    kubernetes.io/kubelet-serving                 system:node:ocp4-lk6b4-
master2.ocp4.example.org                 Approved,Issued
csr-m724n   6m2s   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper
Pending
csr-p4dz2   60m    kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper
Approved,Issued
csr-t9vfj   60m    kubernetes.io/kubelet-serving                 system:node:ocp4-lk6b4-
master1.ocp4.example.org                 Approved,Issued
csr-tggtr   61m    kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper
Approved,Issued
csr-wcbrf   7m6s   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper
Pending
```

3. 목록을 필터링하고 보류중인 **CSR** 만 보려면 다음을 입력하십시오.

```
$ watch "oc get csr -A | grep pending -i"
```

이 명령은 **2** 초마다 출력을 새로 고침하고 보류중인 **CSR** 만 표시합니다. 예를 들면 다음과 같습니다.

출력 예

```
Every 2.0s: oc get csr -A | grep pending -i

csr-m724n   10m   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper
Pending
csr-wcbrf   11m   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper
Pending
```

4. 보류 중인 각 요청을 검사합니다. 예를 들면 다음과 같습니다.

출력 예

```
$ oc describe csr csr-m724n
```

출력 예

```
Name:               csr-m724n
Labels:             <none>
Annotations:        <none>
CreationTimestamp:  Sun, 19 Jul 2020 15:59:37 +0200
Requesting User:    system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper
Signer:             kubernetes.io/kube-apiserver-client-kubelet
Status:             Pending
Subject:
        Common Name:    system:node:ocp4-lk6b4-worker1.ocp4.example.org
        Serial Number:
        Organization:   system:nodes
Events:  <none>
```

5. CSR 정보가 정확하면 다음 요청을 승인합니다.

```
$ oc adm certificate approve csr-m724n
```

6. 설치 프로세스가 완료될 때까지 기다립니다.

```
$ openshift-install wait-for install-complete --dir $ASSETS_DIR --log-level debug
```

설치가 완료되면 명령행에 OpenShift Container Platform 웹 콘솔의 URL과 관리자 이름 및 암호가 표시됩니다.

16.4.24. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

16.5. 제한된 네트워크에서 RHV에 클러스터 설치

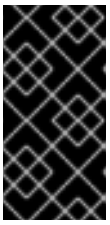
OpenShift Container Platform 버전 4.9에서는 설치 릴리스 콘텐츠의 내부 미러를 생성하여 제한된 네트워크의 RHV(Red Hat Virtualization)에 사용자 지정 OpenShift Container Platform 클러스터를 설치할 수 있습니다.

16.5.1. 사전 요구 사항

RHV 환경에 OpenShift Container Platform 클러스터를 설치하려면 다음 요구사항을 충족해야 합니다.

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.
- RHV의 OpenShift Container Platform에 대한 지원 매트릭스에서 지원되는 버전 조합이 있습니다.
-

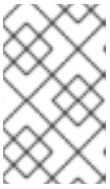
미러 호스트에 레지스트리를 생성하고 사용 중인 **OpenShift Container Platform** 버전의 **imageContentSources** 데이터를 가져옵니다.



중요

미러 호스트에 설치 미디어가 있으므로 해당 컴퓨터를 사용하여 모든 설치 단계를 완료하십시오.

- 클러스터용 영구 스토리지를 프로비저닝합니다. 프라이빗 이미지 레지스트리를 배포하려면 스토리지에서 **ReadWriteMany** 액세스 모드를 제공해야 합니다.
- 방화벽을 사용하며 **Telemetry** 서비스를 사용할 예정인 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성해야 합니다.



참고

프록시를 구성하는 경우에도 해당 사이트 목록을 검토하십시오.

16.5.2. 네트워크가 제한된 환경에서의 설치 정보

OpenShift Container Platform 4.9에서는 소프트웨어 구성 요소를 받기 위한 인터넷 접속이 필요하지 않은 설치를 수행할 수 있습니다. 제한된 네트워크 설치는 클러스터를 설치하는 클라우드 플랫폼에 따라 설치 관리자 프로비저닝 인프라 또는 사용자 프로비저닝 인프라를 사용하여 완료할 수 있습니다.

클라우드 플랫폼에 제한된 네트워크 설치를 수행하는 방법을 선택해도 클라우드 **API**에 액세스는 가능해야 합니다. **Amazon Web Service**의 **Route 53 DNS** 및 **IAM** 서비스와 같은 일부 클라우드 기능에는 인터넷 액세스가 필요합니다. 사용 중인 네트워크에 따라 베어메탈 하드웨어 또는 **VMware vSphere**에 설치하기 위해 필요한 인터넷 액세스가 줄어들 수 있습니다.

제한된 네트워크 설치를 완료하려면 **OpenShift Container Platform** 레지스트리의 내용을 미러링하고 설치 미디어를 포함할 레지스트리를 생성해야 합니다. 인터넷과 폐쇄 네트워크에 모두 액세스하거나 제한 사항을 따르는 다른 방법을 통해 미러 호스트에 레지스트리를 생성할 수 있습니다.

16.5.2.1. 추가 제한

제한된 네트워크의 클러스터에는 다음과 같은 추가 제한이 있습니다.

- ClusterVersion 상태에 사용 가능한 업데이트를 검색할 수 없음 오류가 포함되어 있습니다.

- 기본적으로 필요한 이미지 스트림 태그에 액세스할 수 없기 때문에 개발자 카탈로그의 내용을 사용할 수 없습니다.

16.5.3. OpenShift Container Platform 용 인터넷 액세스

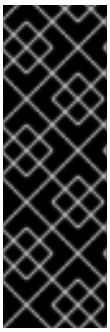
OpenShift Container Platform 4.9에서 클러스터를 설치하기 위해 필요한 이미지를 받으려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.

- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.

- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.



중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

16.5.4. RHV 환경 요구사항

OpenShift Container Platform 버전 4.9 클러스터를 설치하고 실행하려면 RHV 환경이 다음 요구 사항을 충족해야 합니다.

이러한 요구 사항을 충족하지 않으면 설치 또는 프로세스가 실패할 수 있습니다. 또한 이러한 요구 사항을 충족하지 않으면 OpenShift Container Platform 클러스터가 설치 후 며칠 또는 몇 주에 실패할 수

있습니다.

CPU, 메모리, 스토리지에 대한 다음 요구사항은 설치 프로그램이 생성하는 기본 가상 시스템 수를 곱한 기본 값을 기반으로 합니다. 이러한 리소스는 RHV 환경에서 비 OpenShift Container Platform 작업에 사용하는 리소스와함께 사용할 수 있어야 합니다.

설치 프로그램은 기본적으로 설치 프로세스 중에 일곱 개의 가상 시스템을 생성합니다. 먼저 나머지 OpenShift Container Platform 클러스터를 생성하는 동안 임시 서비스와 컨트롤 플레인 영역을 제공하는 부트스트랩 가상 머신을 생성합니다. 설치 프로그램이 클러스터 생성을 완료하면 부트스트랩 시스템을 삭제하여 리소스를 확보합니다.

RHV 환경에서 가상 머신 수를 늘리면 그에 따라 리소스를 늘려야합니다.

요구사항

- RHV 버전은 4.4입니다.
- RHV 환경에는 상태가 Up인 데이터 센터가 하나 있습니다.
- RHV 데이터 센터에는 RHV 클러스터가 포함되어 있습니다.
- RHV 클러스터에는 다음과 같은 OpenShift Container Platform 클러스터 전용 리소스가 있습니다.
  - 최소 28개의 vCPU(설치 중 생성된 일곱 개의 가상 시스템마다 각각 4개)
  - 다음을 포함한 112GiB RAM 이상
    - 임시 컨트롤 플레인을 제공하는 부트스트랩 시스템의 경우 16GiB 이상
    - 컨트롤 플레인을 제공하는 컨트롤 플레인 시스템 세 개 각각에 대해 16GiB 이상

- 애플리케이션 워크로드를 실행하는 컴퓨팅 시스템 세 개 각각에 대해 16GiB 이상

- RHV 스토리지 도메인은 이러한 etcd 백엔드 성능 요구사항을 충족해야 합니다.

- 프로덕션 환경에서 각 가상 머신은 120GiB 이상이어야 합니다. 따라서 스토리지 도메인은 기본 OpenShift Container Platform 클러스터에 대해 840GiB 이상을 제공해야 합니다. 리소스가 제한적인 환경이나 프로덕션 이외의 환경에서는 각 가상 시스템에 32GiB 이상이 있어야 하므로 스토리지 도메인에는 기본 OpenShift Container Platform 클러스터에 필요한 230GiB 이상이 있어야 합니다.

- 설치 및 업데이트 중에 Red Hat Ecosystem Catalog에서 이미지를 다운로드하려면 RHV 클러스터가 인터넷 연결에 액세스할 수 있어야 합니다. Telemetry 서비스에는 서브스크립션 및 권한 부여 프로세스를 단순화하기 위해 인터넷 연결이 필요합니다.

- RHV 클러스터에는 RHV Manager의 REST API에 액세스할 수 있는 가상 네트워크가 있어야 합니다. 설치 관리자가 생성한 VM에서 DHCP를 사용하여 IP 주소를 얻을 수 있으므로 이 네트워크에서 DHCP가 활성화되어 있는지 확인합니다.

- 대상 RHV 클러스터에서 OpenShift Container Platform 클러스터를 설치 및 관리하기 위한 다음과 같은 최소 권한이있는 사용자 계정 및 그룹:

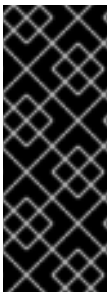
  - DiskOperator
  - DiskCreator
  - UserTemplateBasedVm
  - TemplateOwner
  - TemplateCreator
  - 대상 클러스터의 ClusterAdmin

주의

최소 권한 원칙을 적용합니다. 설치 과정에서 **RHV**에 대한 **SuperUser** 권한이 있는 관리자 계정을 사용하지 않도록 합니다. 설치 프로그램은 사용자가 제공한 인증 정보를 손상된 임시 **ovirt-config.yaml** 파일에 저장합니다.

16.5.5. RHV 환경에 대한 요구사항 확인

RHV 환경이 **OpenShift Container Platform** 클러스터 설치 및 실행 요구사항을 충족하는지 확인합니다. 이러한 요구사항을 충족하지 않으면 실패할 수 있습니다.



중요

이러한 요구사항은 설치 프로그램이 컨트롤 플레인과 컴퓨팅 시스템을 생성하는 데 사용하는 기본 리소스를 기반으로 합니다. 이러한 리소스에는 **vCPU**, 메모리 및 스토리지가 포함됩니다. 이러한 리소스를 변경하거나 **OpenShift Container Platform** 시스템 수를 늘리는 경우에는 그에 따라 이 요구사항을 조정합니다.

프로세스

1. **RHV** 버전이 **OpenShift Container Platform** 버전 **4.9**의 설치를 지원하는지 확인합니다.
   a. **RHV** 관리 포털에서 오른쪽 상단에 있는 **?** 도움말 아이콘을 클릭하고 정보를 선택합니다.
   b. 창이 열리면 **RHV** 소프트웨어 버전을 기록합니다.
   c. **RHV** 버전이 **4.4**인지 확인합니다. 지원되는 버전 조합에 대한 자세한 내용은 **RHV**의 **OpenShift Container Platform**에 대한 지원 매트릭스를 참조하십시오.
2. 데이터 센터, 클러스터 및 스토리지를 검사합니다.
   a. **RHV** 관리 포털에서 **Compute → Data Centers**를 클릭합니다.

b. **OpenShift Container Platform**을 설치하려는 데이터 센터에 액세스할 수 있는지 확인합니다.

c. 해당 데이터 센터의 이름을 클릭합니다.

d. 데이터 센터 세부 사항의 스토리지 탭에서 **OpenShift Container Platform**을 설치하려는 스토리지 도메인이 활성인지 확인합니다.

e. 나중에 사용할 수 있도록 도메인 이름을 기록합니다.

f. 여유 공간이 **230GiB** 이상인지 확인합니다.

g. 스토리지 도메인이 **fio** 성능 벤치마킹 툴을 사용하여 측정할 수 있는 이러한 **etcd** 백엔드 성능 요구사항을 충족하는지 확인합니다.

h. 데이터 센터 세부 사항에서 클러스터 탭을 클릭합니다.

i. **OpenShift Container Platform**을 설치할 **RHV** 클러스터를 찾습니다. 나중에 사용할 수 있도록 클러스터 이름을 기록합니다.

3. **RHV** 호스트 리소스를 검사합니다.

a. **RHV** 관리 포털에서 컴퓨팅 > 클러스터를 클릭합니다.

b. **OpenShift Container Platform**을 설치할 클러스터를 클릭합니다.

c. 클러스터 세부 사항에서 호스트 탭을 클릭합니다.

d. 호스트를 검사하고 **OpenShift Container Platform** 클러스터 *전용*으로 사용할 수 있는 총 **28**개 이상의 논리 **CPU** 코어가 있는지 확인합니다.

e. 나중에 사용할 수 있도록 사용 가능한 논리 **CPU** 코어 수를 기록합니다.

f. 설치 중에 생성된 **7**개의 가상 시스템 각각에 **4**개의 코어가 있을 수 있도록 이러한 **CPU** 코어가 분산되어 있는지 확인합니다.

g. 다음 **OpenShift Container Platform** 시스템 각각에 대한 요구사항을 충족하기 위해 배포된 새 가상 시스템 예약에 필요한 **112GiB**의 최대 여유 메모리가 모두 호스트에 있는지 확인합니다.

- 부트스트랩 시스템에 필요한 **16GiB**
- 세 개의 컨트롤 플레인 시스템 각각에 필요한 **16GiB**
- 세 개의 컴퓨팅 시스템 각각에 대해 **16GiB**

h. 나중에 사용할 수 있도록 새 가상 시스템 예약에 필요한 최대 여유 메모리의 양을 기록합니다.

4. **OpenShift Container Platform**을 설치할 가상 네트워크가 **RHV Manager**의 **REST API**에 액세스할 수 있는지 확인합니다. 이 네트워크의 가상 시스템에서 **RHV** 관리자의 **REST API**에 도달하기 위해 **curl**을 사용합니다.

```
$ curl -k -u <username>@<profile>:<password> \ 1
https://<engine-fqdn>/ovirt-engine/api 2
```

1 **<username>** 의 경우 **RHV**에서 **OpenShift Container Platform** 클러스터를 만들고 관리할 수 있는 권한이 있는 **RHV** 계정의 사용자 이름을 지정합니다. **<profile>**은 로그인 프로파일을 지정합니다(**RHV** 관리 포털 로그인 페이지로 이동하여 프로파일 드롭다운 목록 검토). **<password>**의 경우 해당 사용자 이름에 대한 암호를 지정합니다.

2 **<engine-fqdn>**은 **RHV** 환경의 정규화된 도메인 이름을 지정합니다.

예를 들면 다음과 같습니다.

```
$ curl -k -u ocpadmin@internal:pw123 \
https://rhv-env.virtlab.example.com/ovirt-engine/api
```

16.5.6. 사용자 프로비저닝 인프라에 대한 네트워킹 요구사항

모든 **RHCOS(Red Hat Enterprise Linux CoreOS)** 시스템이 부팅 중에 **Ignition** 구성 파일을 가져오려면 **initramfs**에 네트워킹을 구성해야 합니다.

초기 부팅 과정에서 시스템에 필요한 부팅 옵션을 제공하여 **DHCP** 서버를 통해 설정하거나 정적으로 설정하는 **IP** 주소 구성이 필요합니다. 네트워크 연결이 설정된 후 시스템은 **HTTP** 또는 **HTTPS** 서버에서 **Ignition** 구성 파일을 다운로드합니다. 그런 다음 **Ignition** 구성 파일을 사용하여 각 머신의 정확한 상태를 설정합니다. **Machine Config Operator**는 설치 후 새 인증서 또는 키 적용과 같은 머신에 대한 추가 변경을 완료합니다.

클러스터 시스템의 장기적인 관리를 위해 **DHCP** 서버를 사용하는 것이 좋습니다. **DHCP** 서버가 클러스터 시스템에 영구 **IP** 주소, **DNS** 서버 정보 및 호스트 이름을 제공하도록 구성되었는지 확인합니다.

참고

사용자 프로비저닝 인프라에 **DHCP** 서비스를 사용할 수 없는 경우 **RHCOS** 설치 시 노드에 **IP** 네트워킹 구성과 **DNS** 서버의 주소를 대신 제공할 수 있습니다. **ISO** 이미지에서 설치하는 경우 부팅 인수로 전달할 수 있습니다. 고정 **IP** 프로비저닝 *및 고급 네트워킹 옵션에 대한 자세한 내용은 **RHCOS** 설치 및 **OpenShift Container Platform** 부트스트랩 프로세스 시작* 섹션을 참조하십시오.

Kubernetes API 서버가 클러스터 시스템의 노드 이름을 확인할 수 있어야 합니다. **API** 서버와 작업자 노드가 서로 다른 영역에 있는 경우, **API** 서버가 노드 이름을 확인할 수 있도록 기본 **DNS** 검색 영역을 설정할 수 있습니다. 노드 개체와 모든 **DNS** 요청에서 항상 정규화된 도메인 이름으로 호스트를 가리키는 것도 지원되는 방법입니다

방화벽

클러스터가 필수 사이트에 액세스할 수 있도록 방화벽을 구성하십시오.

또한 다음을 참조하십시오.

- Red Hat Virtualization Manager 방화벽 요구 사항
- 호스트 방화벽 요구 사항

DNS

기본 구성 요소 및 서비스를 올바르게 확인할 수 있도록 인프라 제공 **DNS**를 구성합니다. 로드 밸런서를 하나만 사용하는 경우 이러한 **DNS** 레코드는 동일한 **IP** 주소를 가리킬 수 있습니다.

- 컨트롤 플레인 시스템의 로드 밸런서를 가리키는 **api.<cluster_name>.<base_domain> (internal and external resolution)** 및 **api-int.<cluster_name>.<base_domain> (internal resolution)**에 대한 **DNS** 레코드를 생성합니다.
- **Ingress** 라우터의 로드 밸런서를 가리키는 ***.apps.<cluster_name>.<base_domain>**의 **DNS** 레코드를 생성합니다. 예를 들어 컴퓨팅 머신의 포트 **443** 및 **80**입니다.

16.5.6.1. DHCP를 통해 클러스터 노드의 호스트 이름 설정

RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 호스트 이름은 **NetworkManager**를 통해 설정됩니다. 기본적으로 시스템은 **DHCP**를 통해 호스트 이름을 가져옵니다. **DHCP**에서 호스트 이름을 제공하지 않거나, 커널 인수를 통해 정적으로 설정하거나 다른 방법을 통해 설정하면 역방향 **DNS** 조회를 통해 얻을 수 있습니다. 역방향 **DNS** 조회는 노드에서 네트워크를 초기화한 후 수행되며 확인하는 데 시간이 걸릴 수 있습니다. 다른 시스템 서비스는 이 보다 먼저 시작하여 호스트 이름을 **localhost** 등으로 감지할 수 있습니다. **DHCP**를 사용하여 각 클러스터 노드의 호스트 이름을 제공하여 이 문제를 방지할 수 있습니다.

또한 **DHCP**를 통해 호스트 이름을 설정하면 **DNS** 분할 수평 구현 환경에서 수동으로 **DNS** 레코드 이름 구성 오류를 무시할 수 있습니다.

16.5.6.2. 네트워크 연결 요구사항

OpenShift Container Platform 클러스터 구성 요소가 통신할 수 있도록 시스템 간 네트워크 연결을 구성해야 합니다. 각 시스템에서 클러스터에 있는 다른 모든 시스템의 호스트 이름을 확인할 수 있어야 합니다.

이 섹션에서는 필요한 포트에 대해 자세히 설명합니다.

중요

연결된 OpenShift Container Platform 환경에서 모든 노드는 플랫폼 컨테이너의 이미지를 가져오고 Red Hat에 원격 측정 데이터를 제공하기 위해 인터넷에 액세스할 수 있어야 합니다.

표 16.9. 모든 시스템 간 통신에 사용되는 포트

프로토콜	포트	설명
ICMP	해당 없음	네트워크 연결성 테스트
TCP	**1936**	메트릭
	9000-9999	**9100-9101** 포트의 노드 내보내기 및 **9099** 포트의 Cluster Version Operator를 포함한 호스트 수준 서비스.
	10250-10259	Kubernetes에서 예약하는 기본 포트
	10256	openshift-sdn
UDP	**4789**	VXLAN 및 Geneve
	6081	VXLAN 및 Geneve
	9000-9999	**9100-9101** 포트의 노드 내보내기를 포함한 호스트 수준 서비스.
	500	IPsec IKE 패킷
	4500	IPsec NAT-T 패킷
TCP/UDP	**30000-32767**	Kubernetes 노드 포트
ESP	해당 없음	IPsec Encapsulating Security Payload (ESP)

표 16.10. 모든 시스템과 컨트롤 플레인 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**6443**	Kubernetes API

표 16.11. 컨트롤 플레인 머신 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**2379-2380**	etcd 서버 및 피어 포트

사용자 프로비저닝 인프라에 대한 NTP 구성

OpenShift Container Platform 클러스터는 기본적으로 공용 **NTP(Network Time Protocol)** 서버를 사용하도록 구성되어 있습니다. 로컬 엔터프라이즈 **NTP** 서버를 사용하거나 클러스터가 연결이 끊긴 네트워크에 배포되는 경우 특정 시간 서버를 사용하도록 클러스터를 구성할 수 있습니다. 자세한 내용은 ***chrony** 타임 서비스 설정* 문서를 참조하십시오.

DHCP 서버가 **NTP** 서버 정보를 제공하는 경우 **RHCOS(Red Hat Enterprise Linux CoreOS)** 시스템의 **chrony** 타임 서비스에서 정보를 읽고 **NTP** 서버와 클럭을 동기화할 수 있습니다.

16.5.7. 사용자 프로비저닝 DNS 요구사항

OpenShift Container Platform 배포의 경우 다음 구성 요소에 **DNS** 이름을 확인해야 합니다.

- **Kubernetes API**
- **OpenShift Container Platform** 애플리케이션 와일드카드
- 부트스트랩, 컨트롤 플레인 및 컴퓨팅 시스템

Kubernetes API, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템에 대한 역방향 **DNS** 확인이 필요합니다.

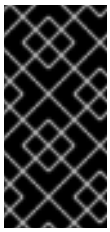
DNS A/AAAA 또는 **CNAME** 레코드는 이름 확인에 사용되며 **PTR** 레코드는 역방향 이름 확인에 사용됩니다. **RHCOS (Red Hat Enterprise Linux CoreOS)**는 **DHCP**에서 호스트 이름을 제공하지 않는 한 모든 노드의 호스트 이름을 설정할 때 역방향 레코드를 사용하기 때문에 역방향 레코드가 중요합니다. 또한 역방향 레코드는 **OpenShift Container Platform**이 작동하는 데 필요한 인증서 서명 요청 **(CSR)**을 생성하는 데 사용됩니다.

참고

DHCP 서버를 사용하여 각 클러스터 노드에 호스트 이름을 제공하는 것이 좋습니다. 자세한 내용은 *사용자 프로비저닝 인프라 섹션에 대한 DHCP 권장 사항* 섹션을 참조하십시오.

사용자가 프로비저닝한 OpenShift Container Platform 클러스터에 대해 다음 DNS 레코드가 필요하며 설치 전에 있어야 합니다. 각 레코드에서 <cluster_name>은 클러스터 이름이고 <base_domain>은 install-config.yaml 파일에서 지정하는 기반 도메인입니다. 전체 DNS 레코드는 <component>.<cluster_name>.<base_domain> 형식입니다.

표 16.12. 필수 DNS 레코드

구성 요소	레코드	설명
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API 로드 밸런서를 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.
	api-int.<cluster_name>.<base_domain>.	내부적으로 API 로드 밸런서를 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.  **중요** API 서버는 Kubernetes에 기록된 호스트 이름으로 작업자 노드를 확인할 수 있어야 합니다. API 서버가 노드 이름을 확인할 수 없는 경우 프록시된 API 호출이 실패할 수 있으며 pod에서 로그를 검색할 수 없습니다.
라우트	***.apps.<cluster_name>.<base_domain>.**	애플리케이션 인그레스 로드 밸런서를 참조하는 와일드카드 DNS A/AAA 또는 CNAME 레코드입니다. 애플리케이션 인그레스 로드 밸런서는 Ingress 컨트롤러 Pod를 실행하는 머신을 대상으로 합니다. Ingress 컨트롤러 Pod는 기본적으로 컴퓨팅 머신에서 실행됩니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다. 예를 들어 **console-openshift-console.apps.<cluster_name>.<base_domain>**은 OpenShift Container Platform 콘솔의 와일드카드 경로로 사용됩니다.
부트스트랩 시스템	**bootstrap.<cluster_name>.<base_domain>.**	부트스트랩 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.

구성 요소	레코드	설명
컨트롤 플레인 머신	**<master><n>. <cluster_name>. <base_domain>.**	컨트롤 플레인 노드의 각 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.
컴퓨팅 머신	**<worker><n>. <cluster_name>. <base_domain>.**	작업자 노드의 각 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.

참고

OpenShift Container Platform 4.4 이상에서는 DNS 구성에서 etcd 호스트 및 SRV 레코드를 지정할 필요가 없습니다.

작은 정보

dig 명령을 사용하여 이름과 역방향 이름을 확인할 수 있습니다. 자세한 검증 단계는 *사용자 프로비저닝 인프라의 DNS 확인* 섹션을 참조하십시오.

16.5.7.1. 사용자 프로비저닝 클러스터의 DNS 구성 예

이 섹션에서는 사용자 프로비저닝 인프라에 OpenShift Container Platform을 배포하기 위한 DNS 요구 사항을 충족하는 A 및 PTR 레코드 구성 샘플을 제공합니다. 샘플은 하나의 DNS 솔루션을 선택하기 위한 조언을 제공하기 위한 것이 아닙니다.

이 예제에서 클러스터 이름은 ocp4이고 기본 도메인은 example.com입니다.

사용자 프로비저닝 클러스터의 DNS A 레코드 구성 예

다음 BIND 영역 파일의 예제에서는 사용자가 프로비저닝한 클러스터의 이름 확인을 위한 샘플 A 레코드를 보여줍니다.

예 16.1. 샘플 DNS 영역 데이터베이스

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
```

```
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
master0.ocp4.example.com. IN A 192.168.1.97 5
master1.ocp4.example.com. IN A 192.168.1.98 6
master2.ocp4.example.com. IN A 192.168.1.99 7
;
worker0.ocp4.example.com. IN A 192.168.1.11 8
worker1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1

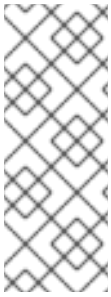
Kubernetes API의 이름 확인을 제공합니다. 레코드는 **API** 로드 밸런서의 **IP** 주소를 나타냅니다.

2

Kubernetes API의 이름 확인을 제공합니다. 레코드는 **API** 로드 밸런서의 **IP** 주소를 참조하며 내부 클러스터 통신에 사용됩니다.

3

와일드카드 경로의 이름 확인을 제공합니다. 레코드는 애플리케이션 인그레스 로드 밸런서의 **IP** 주소를 나타냅니다. 애플리케이션 인그레스 로드 밸런서는 **Ingress** 컨트롤러 **Pod**를 실행하는 머신을 대상으로 합니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.



참고

이 예제에서는 **Kubernetes API** 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 **API** 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

4

부트스트랩 시스템의 이름 확인을 제공합니다.

5 6 7

컨트롤 플레인 시스템의 이름 확인을 제공합니다.

8 9

컴퓨팅 시스템의 이름 확인을 제공합니다.

사용자 프로비저닝 클러스터의 **DNS PTR** 레코드 구성 예

다음 예제 **BIND** 영역 파일은 사용자 프로비저닝 클러스터의 역방향 이름 확인을 위한 샘플 **PTR** 레코드를 보여줍니다.

예 16.2. 역방향 레코드의 샘플 DNS 영역 데이터베이스

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR master0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR master1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR master2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 7
```

```
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com. 8
;
;EOF
```

1

Kubernetes API의 역방향 **DNS** 확인을 제공합니다. **PTR** 레코드는 **API** 로드 밸런서의 레코드 이름을 참조합니다.

2

Kubernetes API의 역방향 **DNS** 확인을 제공합니다. **PTR** 레코드는 **API** 로드 밸런서의 레코드 이름을 참조하며 내부 클러스터 통신에 사용됩니다.

3

부트스트랩 시스템의 역방향 **DNS** 확인을 제공합니다.

4 5 6

컨트롤 플레인 시스템의 역방향 **DNS** 확인을 제공합니다.

7 8

컴퓨팅 시스템의 역방향 **DNS** 확인을 제공합니다.

참고

OpenShift Container Platform 애플리케이션 와일드카드에는 **PTR** 레코드가 필요하지 않습니다.

16.5.7.2. 사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구사항

OpenShift Container Platform을 설치하기 전에 **API** 및 애플리케이션 인그레스 로드 밸런싱 인프라를 프로비저닝해야 합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 **API** 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

참고

RHEL(Red Hat Enterprise Linux) 인스턴스를 사용하여 **API** 및 애플리케이션 인그레스 로드 밸런서를 배포하려면 **RHEL** 서브스크립션을 별도로 구매해야 합니다.

로드 밸런서 인프라는 다음 요구 사항을 충족해야 합니다.

1. **API 로드 밸런서:** 플랫폼과 상호 작용하고 플랫폼을 구성할 수 있도록 사용자(인간과 시스템 모두)에게 공통 끝점을 제공합니다. 다음 조건을 설정합니다.

   - **Layer 4** 로드 밸런싱 전용입니다. 이를 **Raw TCP, SSL Passthrough** 또는 **SSL Bridge** 모드라고 합니다. **SSL Bridge** 모드를 사용하는 경우, **API** 경로에 대해 **SNI(Server Name Indication,** 서버 이름 표시)를 활성화해야 합니다.

   - 스테이트리스 로드 밸런싱 알고리즘입니다. 옵션은 로드 밸런서 구현에 따라 달라집니다.

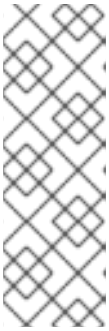
   참고

   API 로드 밸런서가 제대로 작동하기 위해 세션 지속성이 필요하지 않습니다.

   로드 밸런서의 전면과 후면 모두에서 다음 포트를 구성하십시오.

표 16.13. API 로드 밸런서

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
6443	부트스트랩 및 컨트롤 플레인. 부트스트랩 시스템이 클러스터 컨트롤 플레인을 초기화한 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다. API 서버 상태 검사 프로브에 대한 **/readyz** 끝점을 구성해야 합니다.	X	X	Kubernetes API 서버
22623	부트스트랩 및 컨트롤 플레인. 부트스트랩 시스템이 클러스터 컨트롤 플레인을 초기화한 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다.	X		시스템 구성 서버



참고

API 서버가 /readyz 엔드포인트를 해제하는 시점부터 풀에서 API 서버 인스턴스가 제거되는 시점까지 시간이 30초를 넘지 않도록 로드 밸런서를 구성해야 합니다. /readyz가 오류를 반환하거나 정상 상태가 된 후 정해진 시간 안에 끝점이 제거 또는 추가되어야 합니다. 5초 또는 10초의 프로빙 주기(두 번의 성공적인 요청은 정상 상태, 세 번의 요청은 비정상 상태)는 충분한 테스트를 거친 값입니다.

2. 애플리케이션 인그레스 로드 밸런서: 클러스터 외부에서 유입되는 애플리케이션 트래픽에 대한 인그래스 포인트를 제공합니다. 다음 조건을 설정합니다.

- Layer 4 로드 밸런싱 전용입니다. 이를 Raw TCP, SSL Passthrough 또는 SSL Bridge 모드라고 합니다. SSL Bridge 모드를 사용하는 경우 인그레스 경로에 대해 SNI(Server Name Indication, 서버 이름 표시)를 활성화해야 합니다.

- 사용 가능한 옵션과 플랫폼에서 호스팅되는 애플리케이션 유형에 따라 연결 기반 또는 세션 기반 지속성이 권장됩니다.

작은 정보

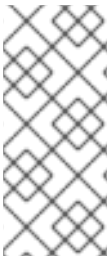
애플리케이션 인그레스 로드 밸런서에서 클라이언트의 실제 IP 주소를 확인할 수 있는 경우 소스 IP 기반 세션 지속성을 활성화하면 엔드 투 엔드 TLS 암호화를 사용하는 애플리케이션의 성능을 향상시킬 수 있습니다.

로드 밸런서의 전면과 후면 모두에서 다음 포트를 구성하십시오.

표 16.14. 애플리케이션 인그레스 로드 밸런서

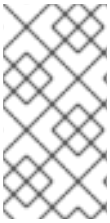
포트	백엔드 시스템(풀 멤버)	내부	외부	설명
443	기본적으로 인그레스 컨트롤러 pod, 컴퓨팅 또는 작업자를 실행하는 시스템입니다.	X	X	HTTPS 트래픽
80	기본적으로 인그레스 컨트롤러 pod, 컴퓨팅 또는 작업자를 실행하는 시스템입니다.	X	X	HTTP 트래픽

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
1936	기본적으로 Ingress 컨트롤러 Pod를 실행하는 작업자 노드입니다. 수신 상태 점검 프로브에 대해 **/healthz/ready** 끝점을 구성해야 합니다.	X	X	HTTP 트래픽



참고

컴퓨팅 노드가 0인 3-노드 클러스터를 배포하는 경우 Ingress 컨트롤러 Pod는 컨트롤 플레인 노드에서 실행됩니다. 3-노드 클러스터 배포에서 HTTP 및 HTTPS 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다.

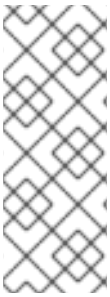


참고

인그레스 라우터에 대한 작업 구성이 OpenShift Container Platform 클러스터에 필요합니다. 컨트롤 플레인 초기화 후 인그레스 라우터를 설정해야 합니다.

16.5.7.2.1. 사용자 프로비저닝 클러스터의 로드 밸런서 구성 예

이 섹션에서는 사용자 프로비저닝 클러스터의 로드 밸런싱 요구 사항을 충족하는 API 및 애플리케이션 수신 로드 밸런서 구성 예를 제공합니다. 샘플은 HAProxy 로드 밸런서에 대한 /etc/haproxy/haproxy.cfg 구성입니다. 이 예제에서는 하나의 로드 밸런싱 솔루션을 선택하기 위한 조언을 제공하는 것을 목적으로 하지 않습니다.



참고

이 예제에서는 Kubernetes API 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 API 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

예 16.3. API 및 애플리케이션 인그레스 로드 밸런서 구성 샘플

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
```

```
    option                  redispatch
    retries                 3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s
    maxconn                 3000
  frontend stats
    bind *:1936
    mode            http
    log             global
    maxconn 10
    stats enable
    stats hide-version
    stats refresh 30s
    stats show-node
    stats show-desc Stats for ocp4 cluster (1)
    stats auth admin:ocp4
    stats uri /stats
  listen api-server-6443 (2)
    bind *:6443
    mode tcp
    server bootstrap bootstrap.ocp4.example.com:6443 check inter 1s backup (3)
    server master0 master0.ocp4.example.com:6443 check inter 1s
    server master1 master1.ocp4.example.com:6443 check inter 1s
    server master2 master2.ocp4.example.com:6443 check inter 1s
  listen machine-config-server-22623 (4)
    bind *:22623
    mode tcp
    server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup (5)
    server master0 master0.ocp4.example.com:22623 check inter 1s
    server master1 master1.ocp4.example.com:22623 check inter 1s
    server master2 master2.ocp4.example.com:22623 check inter 1s
  listen ingress-router-443 (6)
    bind *:443
    mode tcp
    balance source
    server worker0 worker0.ocp4.example.com:443 check inter 1s
    server worker1 worker1.ocp4.example.com:443 check inter 1s
  listen ingress-router-80 (7)
    bind *:80
    mode tcp
    balance source
    server worker0 worker0.ocp4.example.com:80 check inter 1s
    server worker1 worker1.ocp4.example.com:80 check inter 1s
```

(1)

이 예에서 클러스터 이름은 **ocp4** 입니다.

(2)

3 5

부트스트랩 항목은 **OpenShift Container Platform** 클러스터 설치 전에 있어야 하며 부트스트랩 프로세스가 완료된 후 제거해야 합니다.

4

포트 **22623**은 머신 구성 서버 트래픽을 처리하고 컨트롤 플레인 시스템을 가리킵니다.

6

포트 **443**은 **HTTPS** 트래픽을 처리하고 **Ingress** 컨트롤러 **Pod**를 실행하는 시스템을 가리킵니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

7

포트 **80**은 **HTTP** 트래픽을 처리하고 **Ingress** 컨트롤러 **Pod**를 실행하는 머신을 가리킵니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

참고

컴퓨팅 노드가 **0**인 **3-**노드 클러스터를 배포하는 경우 **Ingress** 컨트롤러 **Pod**는 컨트롤 플레인 노드에서 실행됩니다. **3-**노드 클러스터 배포에서 **HTTP** 및 **HTTPS** 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다.

작은 정보

HAProxy를 로드 밸런서로 사용하는 경우 **HAProxy** 노드에서 **netstat -nltupe**를 실행하여 **haproxy** 프로세스가 포트 **6443, 22623, 443** 및 **80**에서 수신 대기 중인지 확인할 수 있습니다.

참고

HAProxy를 로드 밸런서로 사용하고 **SELinux**가 **enforcing**으로 설정된 경우 **HAProxy** 서비스가 **setsebool -P haproxy_connect_any=1**을 실행하여 구성된 **TCP** 포트에 바인딩할 수 있는지 확인해야 합니다.

16.5.8. 설치 시스템 설정

바이너리 openshift-install 설치 프로그램 및 Ansible 스크립트를 실행하려면 RHV Manager 또는 RHV 환경에 대한 네트워크 액세스 권한이 있는 RHEL(Red Hat Enterprise Linux) 컴퓨터와 Manager에서 REST API를 설정합니다.

절차

1. Python3 및 Ansible을 업데이트하거나 설치합니다. 예를 들면 다음과 같습니다.

```
# dnf update python3 ansible
```

2. python3-ovirt-engine-sdk4 패키지를 설치하여 Python 소프트웨어 개발 키트를 가져옵니다.

3. ovirt.image-template Ansible 역할을 설치합니다. RHV Manager 및 기타 RHEL(Red Hat Enterprise Linux) 시스템에서 이 역할은 ovirt-ansible-image-template 패키지로 배포됩니다. 예를 들면 다음과 같습니다.

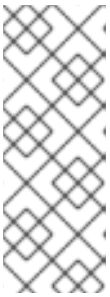
```
# dnf install ovirt-ansible-image-template
```

4. ovirt.vm-infra Ansible 역할을 설치합니다. RHV Manager 및 기타 RHEL 시스템에서 이 역할은 ovirt-ansible-vm-infra 패키지로 배포됩니다.

```
# dnf install ovirt-ansible-vm-infra
```

5. 환경 변수를 만들고 절대 또는 상대 경로를 할당합니다. 예를 들면 다음과 같습니다.

```
$ export ASSETS_DIR=./wrk
```



참고

설치 프로그램은 이 변수를 사용하여 중요한 설치 관련 파일을 저장하는 디렉터리를 만듭니다. 나중에 설치 프로세스는 이 변수를 사용하여 해당 자산 파일을 찾습니다. 이 assets 디렉터리를 삭제하지 마십시오. 이는 클러스터를 제거하는 데 필요합니다.

16.5.9. RHV의 CA 인증서 설정

RHV(Red Hat Virtualization) Manager에서 CA 인증서를 다운로드하여 설치 시스템에서 설정합니다.

RHV Manager의 웹 페이지에서 또는 curl 명령을 사용하여 인증서를 다운로드할 수 있습니다.

나중에 설치 프로그램에 이 인증서를 제공합니다.

절차

1. 다음 두 가지 방법 중 하나를 사용하여 CA 인증서를 다운로드합니다.

   - Manager 웹 페이지(https://<engine-fqdn>/ovirt-engine/)로 이동합니다. 그런 다음 다운로드 아래에서 CA 인증서 링크를 클릭합니다.

   - 다음 명령을 실행합니다.

```
$ curl -k 'https://<engine-fqdn>/ovirt-engine/services/pki-resource?resource=ca-certificate&format=X509-PEM-CA' -o /tmp/ca.pem 1
```

   1 <engine-fqdn>은 rhv-env.virtlab.example.com과 같이 RHV Manager의 정규화된 도메인 이름을 지정합니다.

2. Manager에 대한 루트리스(rootless) 사용자 액세스 권한을 부여하도록 CA 파일을 구성합니다. 8진수 값 0644(기호 값: -rw-r— r--)를 갖도록 CA 파일 권한을 설정합니다.

```
$ sudo chmod 0644 /tmp/ca.pem
```

3. Linux의 경우 서버 인증서를 CA 인증서 디렉터리에 복사합니다. 권한을 유지하려면 -p를 사용합니다.

```
$ sudo cp -p /tmp/ca.pem /etc/pki/ca-trust/source/anchors/ca.pem
```

4. 운영 체제의 인증서 관리자에 인증서를 추가합니다.

   - macOS의 경우 인증서 파일을 두 번 클릭하고 키체인 액세스 유틸리티를 사용하여 파일을 시스템 키 체인에 추가합니다.

- Linux의 경우 다음과 같이 CA 트러스트를 업데이트합니다.

```
$ sudo update-ca-trust
```

참고

자체 인증 기관을 사용하는 경우 시스템이 해당 인증 기관을 신뢰하는지 확인합니다.

추가 리소스

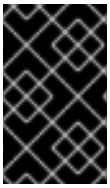
- 자세한 내용은 RHV 설명서의 인증 및 보안을 참조하십시오.

16.5.10. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 core 사용자의 ~/.ssh/authorized_keys 목록에 추가되어 암호 없는 인증을 활성화합니다.

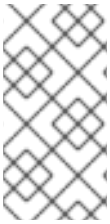
키가 노드에 전달되면 키 쌍을 사용하여 사용자 core로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. ./openshift-install gather 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.



중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.



참고

AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

절차

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: ~/.ssh/id_ed25519)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 '~/.ssh 디렉터리에 있는지 확인하십시오.

참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 ed25519 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 rsa 또는 ecdsa 알고리즘을 사용하는 키를 생성합니다.

2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 ~/.ssh/id_ed25519.pub 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 ./openshift-install gather 명령을 사용하려는 경우 필요합니다.

참고

일부 배포에서는 ~/.ssh/id_rsa 및 ~/.ssh/id_dsa와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. ssh-agent 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. ssh-agent에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: ~/.ssh/id_ed25519).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

16.5.11. Ansible Playbook 다운로드

RHV에 OpenShift Container Platform 4.9 버전을 설치하기 위한 Ansible 플레이북을 다운로드합니

다.

절차

- 설치 시스템에서 다음 명령을 실행하십시오.

```
$ mkdir playbooks
```

```
$ cd playbooks
```

```
$ curl -s -L -X GET https://api.github.com/repos/openshift/installer/contents/upi/ovirt?
ref=release-4.9 |
grep 'download_url.*\.yml' |
awk '{ print $2 }' | sed -r 's/("|",)//g' |
xargs -n 1 curl -O
```

다음 단계

- 이러한 **Ansible Playbook**을 다운로드한 후 설치 프로그램을 실행하여 설치 구성 파일을 생성하기 전에 **assets** 디렉터리에 대한 환경 변수를 생성하고 **inventory.yml** 파일을 사용자 정의해야 합니다.

16.5.12. inventory.yml 파일

inventory.yml 파일을 사용하여 설치하려는 **OpenShift Container Platform** 클러스터의 요소를 정의하고 생성합니다. 여기에는 **RHCOS (Red Hat Enterprise Linux CoreOS)** 이미지, 가상 머신 템플릿, 부트스트랩 머신, 컨트롤 플레인 노드 및 작업자 노드와 같은 요소가 포함됩니다. 또한 **inventory.yml**을 사용하여 클러스터를 제거합니다.

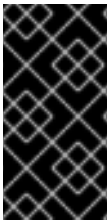
다음 **inventory.yml**에에서는 매개 변수와 매개 변수 기본값을 보여줍니다. 이러한 기본값의 수량과 숫자는 **RHV** 환경에서 프로덕션 **OpenShift Container Platform** 클러스터를 실행하기 위한 요구 사항을 충족합니다.

inventory.yml 파일 예

```
---
all:
  vars:

    ovirt_cluster: "Default"
    ocp:
      assets_dir: "{{ lookup('env', 'ASSETS_DIR') }}"
```

```
    ovirt_config_path: "{{ lookup('env', 'HOME') }}/.ovirt/ovirt-config.yaml"

  # ---
  # {op-system} section
  # ---
  rhcos:
    image_url: "https://mirror.openshift.com/pub/openshift-
v4/dependencies/rhcos/4.9/latest/rhcos-openstack.x86_64.qcow2.gz"
    local_cmp_image_path: "/tmp/rhcos.qcow2.gz"
    local_image_path: "/tmp/rhcos.qcow2"

  # ---
  # Profiles section
  # ---
  control_plane:
    cluster: "{{ ovirt_cluster }}"
    memory: 16GiB
    sockets: 4
    cores: 1
    template: rhcos_tpl
    operating_system: "rhcos_x64"
    type: high_performance
    graphical_console:
      headless_mode: false
      protocol:
      - spice
      - vnc
    disks:
    - size: 120GiB
      name: os
      interface: virtio_scsi
      storage_domain: depot_nvme
    nics:
    - name: nic1
      network: lab
      profile: lab

  compute:
    cluster: "{{ ovirt_cluster }}"
    memory: 16GiB
    sockets: 4
    cores: 1
    template: worker_rhcos_tpl
    operating_system: "rhcos_x64"
    type: high_performance
    graphical_console:
      headless_mode: false
      protocol:
      - spice
      - vnc
    disks:
    - size: 120GiB
      name: os
      interface: virtio_scsi
      storage_domain: depot_nvme
    nics:
```

```
  - name: nic1
    network: lab
    profile: lab

# ---
# Virtual machines section
# ---
vms:
- name: "{{ metadata.infraID }}-bootstrap"
  ocp_type: bootstrap
  profile: "{{ control_plane }}"
  type: server
- name: "{{ metadata.infraID }}-master0"
  ocp_type: master
  profile: "{{ control_plane }}"
- name: "{{ metadata.infraID }}-master1"
  ocp_type: master
  profile: "{{ control_plane }}"
- name: "{{ metadata.infraID }}-master2"
  ocp_type: master
  profile: "{{ control_plane }}"
- name: "{{ metadata.infraID }}-worker0"
  ocp_type: worker
  profile: "{{ compute }}"
- name: "{{ metadata.infraID }}-worker1"
  ocp_type: worker
  profile: "{{ compute }}"
- name: "{{ metadata.infraID }}-worker2"
  ocp_type: worker
  profile: "{{ compute }}"
```



중요

매개 변수 설명이 **"Enter"**로 시작하는 매개 변수의 값을 입력합니다. 그렇지 않으면 기본값을 사용하거나 새 값으로 변경할 수 있습니다.

일반 섹션

- **ovirt_cluster: OpenShift Container Platform** 클러스터를 설치할 기존 **RHV** 클러스터의 이름을 입력합니다.
- **ocp.assets_dir: openshift-install** 설치 프로그램이 생성하는 파일을 저장하기 위해 생성하는 디렉터리의 경로입니다.
- **ocp.ovirt_config_path:** 설치 프로그램이 생성하는 **ovirt-config.yaml** 파일의 경로 (예:

./wrk/install-config.yaml)입니다. 이 파일에는 Manager의 REST API와 상호 작용하는 데 필요한 인증 정보가 포함되어 있습니다.

RHCOS(Red Hat Enterprise Linux CoreOS) 섹션

- image_url: 다운로드를 위해 지정한 RHCOS 이미지의 URL을 입력합니다.
- local_cmp_image_path: 압축된 RHCOS 이미지에 대한 로컬 다운로드 디렉터리의 경로입니다.
- local_image_path: 추출된 RHCOS 이미지에 대한 로컬 디렉터리의 경로입니다.

프로필 섹션

이 섹션은 두 가지 프로필로 구성됩니다.

- control_plane: 부트스트랩 및 컨트롤 플레인 노드의 프로필입니다.
- compute: 컴퓨팅 플레인에 있는 작업자 노드의 프로필입니다.

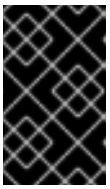
이러한 프로필에는 다음과 같은 매개 변수가 있습니다. 매개 변수의 기본값은 프로덕션 클러스터 실행을 위한 최소 요구 사항을 충족합니다. 이러한 값을 증가시키거나 사용자 지정하여 워크로드 요구 사항을 충족할 수 있습니다.

- cluster: 이 값은 일반 섹션의 ovirt_cluster에서 클러스터 이름을 가져옵니다.
- memory: 가상 시스템의 메모리 양(GB)입니다.
- sockets: 가상 시스템의 소켓 수입니다.
- cores: 가상 시스템의 코어 수입니다.
- template: 가상 시스템 템플릿의 이름입니다. 여러 클러스터를 설치할 계획이고 이러한 클러

스터가 서로 다른 사양을 포함하는 템플릿을 사용하는 경우 템플릿 이름 앞에 클러스터 ID를 추가합니다.

- operating_system: 가상 시스템의 게스트 운영 체제 유형입니다. oVirt/RHV 버전 4.4에서 이 값은 rhcos_x64이어야 하므로 Ignition script의 값이 가상 시스템에 전달될 수 있습니다.

- type: 가상 머신의 유형으로 server를 입력합니다.



중요

type 매개 변수의 값을 high_performance에서 server로 변경해야 합니다.

- disks: 디스크 사양. control_plane 및 compute 노드는 서로 다른 스토리지 도메인을 가질 수 있습니다.

- size: 최소 디스크 크기.

- name: RHV에서 대상 클러스터에 연결된 디스크의 이름을 입력합니다.

- interface: 지정한 디스크의 인터페이스 유형을 입력합니다.

- storage_domain: 지정한 디스크의 스토리지 도메인을 입력합니다.

- nics: 가상 머신이 사용하는 name 및 network를 입력합니다. 가상 네트워크 인터페이스 프로필을 지정할 수도 있습니다. 기본적으로 NIC는 oVirt/RHV MAC 풀에서 MAC 주소를 가져옵니다.

가상 머신 섹션

이 마지막 섹션인 vms는 클러스터에서 생성하고 배포할 가상 시스템을 정의합니다. 기본적으로 프로덕션 환경에 대해 최소한의 컨트롤 플레인 및 작업자 노드를 제공합니다.

vms에는 세 가지 필수 요소가 있습니다.

- **name:** 가상 머신의 이름입니다. 이 경우 **metadata.infraID**는 가상 머신 이름 앞에 **metadata.yml** 파일의 인프라 **ID**를 추가합니다.

- **ocp_type: OpenShift Container Platform** 클러스터에서 가상 머신의 역할입니다. 가능한 값은 **bootstrap, master, worker**입니다.

- **profile:** 각 가상 머신이 사양을 상속하는 프로필의 이름입니다. 이 예제에서 가능한 값은 **control_plane** 또는 **compute**입니다.

가상 머신이 해당 프로필에서 상속하는 값을 재정의할 수 있습니다. 이렇게 하려면 **inventory.yml**의 가상 머신에 프로필 속성의 이름을 추가하고 재정의 값을 할당합니다. 이 예제를 보려면 이전 **inventory.yml** 예제에서 **name: "{{ metadata.infraID }}-bootstrap"** 가상 머신을 검사합니다. 여기에는 **server** 값이 있는 **type** 속성이 이 가상 머신이 **control_plane** 프로필에서 상속하는 **type** 속성 값을 재정의합니다.

메타 데이터 변수

가상 머신의 경우 **metadata.infraID**는 **Ignition** 파일을 빌드할 때 생성한 **metadata.json** 파일의 인프라 **ID**를 가상 머신의 이름 앞에 추가합니다.

Playbook은 **ocp.assets_dir**에 있는 특정 파일에서 **infraID** 읽고 다음 코드를 사용합니다.

```
---
- name: include metadata.json vars
  include_vars:
    file: "{{ ocp.assets_dir }}/metadata.json"
    name: metadata

  ...
```

16.5.13. RHCOS 이미지 설정 지정

inventory.yml 파일의 **RHCOS (Red Hat Enterprise Linux CoreOS)** 이미지 설정을 업데이트합니다. 나중에 **Playbook** 중 파일 하나를 실행하면 **image_url URL**에서 **local_cmp_image_path** 디렉터리로 압축된 **Red Hat Enterprise** 리눅스 **CoreOS(RHCOS)** 이미지가 다운로드됩니다. 그런 다음 **Playbook**은 이미지를 **local_image_path** 디렉터리에 압축 해제하고 이를 사용하여 **oVirt/RHV** 템플릿을 만듭니다.

절차

1. 설치 중인 **OpenShift Container Platform** 버전에 대한 **RHCOS** 이미지 다운로드 페이지(예: **Index of /pub/openshift-v4/dependencies/rhcos/latest/latest**)를 찾습니다.

2. 해당 다운로드 페이지에서 **https://mirror.openshift.com/pub/openshift-v4/dependencies/rhcos/4.9/latest/rhcos-openstack.x86_64.qcow2.gz**와 같은 **OpenStack qcow2** 이미지의 **URL**을 복사합니다.

3. 이전에 다운로드 한 **inventory.yml Playbook**을 편집합니다. 여기에 **URL**을 **image_url** 값으로 붙여 넣습니다. 예를 들면 다음과 같습니다.

```
rhcos:
  "https://mirror.openshift.com/pub/openshift-v4/dependencies/rhcos/4.9/latest/rhcos-openstack.x86_64.qcow2.gz"
```

16.5.14. 설치 구성 파일 만들기

설치 프로그램 **openshift-install**을 실행하고 이전에 지정했거나 수집한 정보로 해당 프롬프트에 응답하여 설치 구성 파일을 만듭니다.

프롬프트에 대한 응답을 마치면 설치 프로그램이 이전에 지정한 **assets** 디렉터리 (예 : **./wrk/install-config.yaml**)에 **install-config.yaml** 파일의 초기 버전을 생성합니다.

설치 프로그램은 **$HOME/.ovirt/ovirt-config.yaml** 파일을 생성합니다. 이 파일에는 **Manager**에 연결하여 **REST API**를 사용하는 데 필요한 모든 연결 매개 변수가 포함되어 있습니다.

참고: 설치 프로세스는 **Internal API virtual IP** 및 **Ingress virtual IP**와 같은 일부 매개 변수에 제공하는 값을 사용하지 않습니다. 이미 인프라 **DNS**에서 구성했기 때문입니다.

또한 **oVirt cluster, oVirt storage** 및 **oVirt network**에 대한 값과 같이 **inventory.yml**의 매개 변수에 제공하는 값을 사용합니다. 그리고 스크립트를 사용하여 **install-config.yaml**에서 이러한 동일한 값을 이전에 언급한 **virtual IP**로 제거하거나 변경합니다.

절차

1. 설치 프로그램을 실행합니다.

```
$ openshift-install create install-config --dir $ASSETS_DIR
```

2. 시스템에 대한 정보를 사용하여 설치 프로그램의 프롬프트 메시지에 응답합니다.

출력 예

```
? SSH Public Key /home/user/.ssh/id_dsa.pub
? Platform <ovirt>
? Engine FQDN[:PORT] [? for help] <engine.fqdn>
? Enter ovirt-engine username <ocpadmin@internal>
? Enter password <******>
? oVirt cluster <cluster>
? oVirt storage <storage>
? oVirt network <net>
? Internal API virtual IP <172.16.0.252>
? Ingress virtual IP <172.16.0.251>
? Base Domain <example.org>
? Cluster Name <ocp4>
? Pull Secret [? for help] <********>
```

```
? SSH Public Key /home/user/.ssh/id_dsa.pub
? Platform <ovirt>
? Engine FQDN[:PORT] [? for help] <engine.fqdn>
? Enter ovirt-engine username <ocpadmin@internal>
? Enter password <******>
? oVirt cluster <cluster>
? oVirt storage <storage>
? oVirt network <net>
? Internal API virtual IP <172.16.0.252>
? Ingress virtual IP <172.16.0.251>
? Base Domain <example.org>
? Cluster Name <ocp4>
? Pull Secret [? for help] <********>
```

Internal API virtual IP 및 **Ingress virtual IP**의 경우 **DNS** 서비스를 구성할 때 지정한 **IP** 주소를 제공합니다.

oVirt cluster 및 **Base Domain** 프롬프트에서 입력한 값은 **REST API** 및 생성하는 모든 애플리케이션의 **URL**의 일부를 구성합니다. **(예: https://api.ocp4.example.org:6443/** 및 **https://console-openshift-console.apps.ocp4.example.org)**

Red Hat OpenShift Cluster Manager에서 풀 시크릿을 가져올 수 있습니다.

16.5.15. RHV의 샘플 install-config.yaml 파일

install-config.yaml 파일을 사용자 지정하여 **OpenShift Container Platform** 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.

```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 0 (4)
controlPlane: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3 (7)
metadata:
  name: test (8)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 (9)
    hostPrefix: 23 (10)
  networkType: OpenShiftSDN
  serviceNetwork: (11)
  - 172.30.0.0/16
platform:
  none: {} (12)
fips: false (13)
pullSecret: '{"auths": ...}' (14)
sshKey: 'ssh-ed25519 AAAA...' (15)
```

(1)

클러스터의 기본 도메인입니다. 모든 **DNS** 레코드는 이 기본 도메인의 하위 도메인이어야 하며 클러스터 이름을 포함해야 합니다.

(2) (5)

controlPlane 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

(3) (6)

동시 멀티스레딩(**SMT**) 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 **SMT**가 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. **SMT**를 비활성화하는 경우 모든 클러스터 머신에서 이를 비활성화해야 합니다. 여기에는 컨트롤 플레인과 컴퓨팅 머신이 모두 포함됩니다.

참고

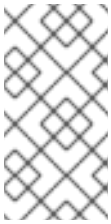
SMT(동시 멀티 스레딩)는 기본적으로 활성화되어 있습니다. BIOS 설정에서 SMT를 활성화하지 않으면 hyperthreading 매개변수가 적용되지 않습니다.

중요

BIOS에서든 install-config.yaml 파일에서든 hyperthreading을 비활성화한 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.

4

사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치할 때 이 값을 0으로 설정해야 합니다. 설치 프로그램에서 제공하는 설치에서 매개 변수는 클러스터가 생성 및 관리하는 컴퓨팅 머신 수를 제어합니다. 사용자 프로비저닝 설치에서는 클러스터 설치를 완료하기 전에 컴퓨팅 시스템을 수동으로 배포해야 합니다.



참고

3-노드 클러스터를 설치하는 경우 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템을 설치할 때 컴퓨팅 머신을 배포하지 마십시오.

7

클러스터에 추가하는 컨트롤 플레인 시스템의 수입니다. 클러스터에서 이 값을 클러스터의 etcd 끝점 수로 사용하므로 이 값은 배포하는 컨트롤 플레인 시스템의 수와 일치해야 합니다.

8

DNS 레코드에 지정한 클러스터 이름입니다.

9

Pod IP 주소가 할당되는 IP 주소 블록입니다. 이 블록은 기존 물리적 네트워크와 중복되지 않아야합니다. 이러한 IP 주소는 Pod 네트워크에 사용됩니다. 외부 네트워크에서 Pod에 액세스해야 하는 경우, 트래픽을 관리하도록 로드 밸런서와 라우터를 설정해야 합니다.

참고

클래스 E CIDR 범위는 향후 사용을 위해 예약되어 있습니다. 클래스 E CIDR 범위를 사용하려면 네트워킹 환경에서 클래스 E CIDR 범위 내에서 IP 주소를 수락하는지 확인해야 합니다.

10

개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하면 지정된 **cidr** 이외 **/23** 서브넷이 각 노드에 할당되어 **510(2^(32 - 23) - 2) Pod IP** 주소가 허용됩니다. 외부 네트워크에서 노드에 액세스해야 하는 경우 트래픽을 관리하도록 로드 밸런서와 라우터를 구성합니다.

11

서비스 **IP** 주소에 사용할 **IP** 주소 풀입니다. **IP** 주소 풀은 하나만 입력할 수 있습니다. 이 블록은 기존 물리적 네트워크와 중복되지 않아야합니다. 외부 네트워크에서 서비스에 액세스해야 하는 경우, 트래픽을 관리하도록 로드 밸런서와 라우터를 구성합니다.

12

플랫폼을 **none**으로 설정해야 합니다. **RHV** 인프라에 대한 추가 플랫폼 구성 변수는 지정할 수 없습니다.

중요

플랫폼 유형으로 설치된 클러스터는 **Machine API**로 컴퓨팅 머신 관리와 같은 일부 기능을 사용할 수 없습니다. 이 제한은 클러스터에 연결된 컴퓨팅 시스템이 일반적으로 기능을 지원하는 플랫폼에 설치된 경우에도 적용됩니다. 이 매개변수는 설치 후 변경할 수 없습니다.

13

FIPS 모드 활성화 또는 비활성화 여부입니다. 기본적으로 **FIPS** 모드는 비활성화됩니다. **FIPS** 모드가 활성화되면 **OpenShift Container Platform**이 실행되는 **RHCOS(Red Hat Enterprise Linux CoreOS)** 시스템에서 기본 **Kubernetes** 암호화 제품군은 우회하고 **RHCOS**와 함께 제공되는 암호화 모듈을 대신 사용합니다.

중요

FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 **OpenShift Container Platform** 배포에서만 지원됩니다.

14

Red Hat OpenShift Cluster Manager의 풀 시크릿. 이 풀 시크릿을 사용하면 **OpenShift Container Platform** 구성 요소에 대한 컨테이너 이미지를 제공하는 **Quay.io**를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

15

참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 **OpenShift Container Platform** 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 **SSH** 키를 지정합니다.

16.5.15.1. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 **HTTP** 또는 **HTTPS** 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 **OpenShift Container Platform** 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 **API**에 대한 호출을 포함하여 모든 클러스터 발신(**Egress**) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.

참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), **Google Cloud Platform (GCP)**, **Microsoft Azure** 및 **Red Hat OpenStack Platform (RHOSP)**에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

프로세스

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
```

```
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1

클러스터 외부에서 **HTTP** 연결을 구축하는 데 사용할 프록시 **URL**입니다. **URL** 스키마는 **http**여야 합니다.

2

클러스터 외부에서 **HTTPS** 연결을 구축하는 데 사용할 프록시 **URL**입니다.

3

대상 도메인 이름, **IP** 주소 또는 프록시에서 제외할 기타 네트워크 **CIDR**로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. ***를 사용하여 모든 대상에 대해 프록시를 바이패스합니다.

4

이 값을 제공하면 설치 프로그램에서 추가 **CA** 인증서를 보유할 **openshift -config** 네임스페이스에 **user-ca- bundle** 이라는 구성 맵을 생성합니다. **additionalTrustBundle** 및 하나 이상의 프록시 설정을 제공하는 경우 프록시 오브젝트는 **trustedCA** 필드의 **user-ca-bundle** 구성 맵을 참조하도록 구성됩니다. 그러면 **Cluster Network Operator**에서 **trustedCA** 매개변수에 대해 지정된 콘텐츠를 **RHCOS** 신뢰 번들과 병합하는 **trusted-ca-bundle** 구성 맵을 생성합니다. 프록시의 **ID** 인증서를 **RHCOS** 트러스트 번들에 있는 기관에서 서명하지 않은 경우 **additionalTrustBundle** 필드가 있어야 합니다.

참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 **OpenShift Container Platform**을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계

속 생성되지만 **spec**은 **nil**이 됩니다.

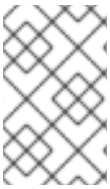
참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

16.5.16. install-config.yaml 사용자 정의

여기에서는 **3** 개의 **Python** 스크립트를 사용하여 설치 프로그램의 일부 기본 동작을 재정의합니다.

- 기본적으로 설치 프로그램은 머신 **API**를 사용하여 노드를 만듭니다. 이 기본 동작을 재정의하려면 컴퓨팅 노드 수를 복제본 **0**으로 설정합니다. 나중에 **Anable Playbook**을 사용하여 컴퓨팅 노드를 생성합니다.
- 기본적으로 설치 프로그램은 노드의 시스템 네트워크의 **IP** 범위를 설정합니다. 이 기본 동작을 재정의하려면 인프라와 일치하도록 **IP** 범위를 설정합니다.
- 기본적으로 설치 프로그램은 플랫폼을 **ovirt**로 설정합니다. 그러나 사용자 프로비저닝 인프라에 클러스터를 설치하는 것은 베어 메탈에 클러스터를 설치하는 것과 비슷합니다. 따라서 **install-config.yaml**에서 **ovirt** 플랫폼 섹션을 삭제하고 플랫폼을 **none**으로 변경합니다. 대신 **inventory.yml**을 사용하여 모든 필수 설정을 지정합니다.



참고

이 스니펫은 **Python 3** 및 **Python 2**에서 작동합니다.

프로세스

1. 컴퓨팅 노드 수를 복제본 **0**으로 설정합니다.

```
$ python3 -c 'import os, yaml
path = "%s/install-config.yaml" % os.environ["ASSETS_DIR"]
conf = yaml.safe_load(open(path))
conf["compute"][0]["replicas"] = 0
open(path, "w").write(yaml.dump(conf, default_flow_style=False))'
```

2. 머신 네트워크의 **IP** 범위를 설정합니다. 예를 들어 범위를 **172.16.0.0/16**으로 설정하려면 다음을 입력합니다.

```
$ python3 -c 'import os, yaml
path = "%s/install-config.yaml" % os.environ["ASSETS_DIR"]
conf = yaml.safe_load(open(path))
conf["networking"]["machineNetwork"][0]["cidr"] = "172.16.0.0/16"
open(path, "w").write(yaml.dump(conf, default_flow_style=False))'
```

3. ovirt 섹션을 제거하고 플랫폼을 none으로 변경합니다.

```
$ python3 -c 'import os, yaml
path = "%s/install-config.yaml" % os.environ["ASSETS_DIR"]
conf = yaml.safe_load(open(path))
platform = conf["platform"]
del platform["ovirt"]
platform["none"] = {}
open(path, "w").write(yaml.dump(conf, default_flow_style=False))'
```

주의

Red Hat Virtualization은 현재 oVirt 플랫폼에서 사용자 프로비저닝 인프라를 사용하여 설치를 지원하지 않습니다. 따라서 플랫폼을 none 으로 설정해야 OpenShift Container Platform에서 각 노드를 베어 메탈 노드로 식별하고 클러스터를 베어 메탈 클러스터로 식별할 수 있습니다. 이는 모든 플랫폼에 클러스터를 설치하는 것과 동일하며 다음과 같은 제한 사항이 있습니다.

1. 클러스터 공급자가 없으므로 각 머신을 수동으로 추가해야 하며 노드 확장 기능이 없습니다.

2. oVirt CSI 드라이버가 설치되지 않으며 CSI 기능이 없습니다.

16.5.17. 매니페스트 파일 생성

설치 프로그램을 사용하여 assets 디렉터리에 매니페스트 파일 세트를 생성하십시오.

매니페스트 파일을 생성하는 명령은 install-config.yaml 파일을 사용하기 전에 경고 메시지를 표시합니다.

install-config.yaml 파일을 재사용하려는 경우 매니페스트 파일을 생성하기 전에 해당 파일의 백업 사본을 생성합니다.

절차

1. 선택 사항: **install-config.yaml** 파일의 작업 사본을 만듭니다.

```
$ cp install-config.yaml install-config.yaml.backup
```

2. **assets** 디렉터리에 매니페스트 세트를 생성합니다.

```
$ openshift-install create manifests --dir $ASSETS_DIR
```

이 명령은 다음 메시지를 표시합니다.

출력 예

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true
for Scheduler cluster settings
```

이 명령은 다음 매니페스트 파일을 생성합니다.

출력 예

```
$ tree
.
└── wrk
    ├── manifests
    │   ├── 04-openshift-machine-config-operator.yaml
    │   ├── cluster-config.yaml
    │   ├── cluster-dns-02-config.yml
    │   ├── cluster-infrastructure-02-config.yml
    │   ├── cluster-ingress-02-config.yml
    │   ├── cluster-network-01-crd.yml
```

```
│   ├── cluster-network-02-config.yml
│   ├── cluster-proxy-01-config.yaml
│   ├── cluster-scheduler-02-config.yml
│   ├── cvo-overrides.yaml
│   ├── etcd-ca-bundle-configmap.yaml
│   ├── etcd-client-secret.yaml
│   ├── etcd-host-service-endpoints.yaml
│   ├── etcd-host-service.yaml
│   ├── etcd-metric-client-secret.yaml
│   ├── etcd-metric-serving-ca-configmap.yaml
│   ├── etcd-metric-signer-secret.yaml
│   ├── etcd-namespace.yaml
│   ├── etcd-service.yaml
│   ├── etcd-serving-ca-configmap.yaml
│   ├── etcd-signer-secret.yaml
│   ├── kube-cloud-config.yaml
│   ├── kube-system-configmap-root-ca.yaml
│   ├── machine-config-server-tls-secret.yaml
│   └── openshift-config-secret-pull-secret.yaml
└── openshift
    ├── 99_kubeadmin-password-secret.yaml
    ├── 99_openshift-cluster-api_master-user-data-secret.yaml
    ├── 99_openshift-cluster-api_worker-user-data-secret.yaml
    ├── 99_openshift-machineconfig_99-master-ssh.yaml
    ├── 99_openshift-machineconfig_99-worker-ssh.yaml
    └── openshift-install-manifests.yaml
```

다음 단계

- 컨트롤 플레인 노드를 예약 불가능하게 설정합니다.

16.5.18. 컨트롤 플레인 노드를 예약 불가능하게 설정하기

컨트롤 플레인 시스템을 수동으로 생성 및 배포하고 있으므로 컨트롤 플레인 노드를 예약할 수 없도록 매니페스트 파일을 구성해야 합니다.

절차

1. 컨트롤 플레인 노드를 스케줄 불가능하게하려면 다음을 입력합니다.

```
$ python3 -c 'import os, yaml
path = "%s/manifests/cluster-scheduler-02-config.yml" % os.environ["ASSETS_DIR"]
data = yaml.safe_load(open(path))
data["spec"]["mastersSchedulable"] = False
open(path, "w").write(yaml.dump(data, default_flow_style=False))'
```

16.5.19. Ignition 파일 빌드하기

방금 생성 및 수정한 매니페스트 파일에서 **Ignition** 파일을 빌드하려면 설치 프로그램을 실행합니다. 이 작업은 **RHCOS (Red Hat Enterprise Linux CoreOS)** 시스템 **initramfs** 를 생성하여 **Ignition** 파일을 가져오고 노드를 만드는 데 필요한 구성을 수행합니다.

설치 프로그램은 **Ignition** 파일 외에도 다음을 생성합니다.

- **oc** 및 **kubectl** 유틸리티를 사용하여 클러스터에 연결하기위한 관리자 인증 정보가 포함된 **auth** 디렉터리입니다.
- 현재 설치에 대한 **OpenShift Container Platform** 클러스터 이름, 클러스터 **ID** 및 인프라 **ID** 와 같은 정보가 포함된 **metadata.json** 파일입니다.

설치 프로세스를 위한 **Ansible Playbook**은 생성한 가상 머신의 접두사로 **infraID** 값을 사용합니다. 이는 동일한 **oVirt/RHV** 클러스터에 여러 설치가 있을 때 이름 지정 충돌을 방지합니다.

참고

Ignition 구성 파일의 인증서는 **24** 시간 후에 만료됩니다. 첫 번째 인증서 교체가 완료될 수 있도록 클러스터 설치를 완료하고 성능이 저하되지 않은 상태에서 **24**시간 동안 클러스터를 계속 실행해야 합니다.

절차

1. **Ignition** 파일을 빌드하려면 다음을 입력합니다.

```
$ openshift-install create ignition-configs --dir $ASSETS_DIR
```

출력 예

```
$ tree
.
└── wrk
    ├── auth
    │   ├── kubeadmin-password
    │   └── kubeconfig
    ├── bootstrap.ign
```

```
├── master.ign
├── metadata.json
└── worker.ign
```

16.5.20. 템플릿 및 가상 머신 생성

inventory.yml에서 변수를 확인한 후 첫 번째 Ansible 프로비저닝 Playbook인 create-templates-and-vms.yml을 실행합니다.

이 Playbook은 $HOME/.ovirt/ovirt-config.yaml의 RHV Manager에 대한 연결 매개 변수를 사용하고 assets 디렉터리에서 metadata.json을 읽습니다.

로컬 Red Hat Enterprise Linux CoreOS (RHCOS) 이미지가 존재하지 않는 경우 Playbook은 inventory.yml의 image_url에서 지정한 URL에서 해당 이미지를 다운로드합니다. 이미지를 추출하여 RHV에 업로드하여 템플릿을 생성합니다.

Playbook은 inventory.yml 파일의 control_plane 및 compute 프로필을 기반으로 템플릿을 생성합니다. 이러한 프로필의 이름이 다른 경우 두 개의 템플릿을 만듭니다.

Playbook이 완료되면 생성된 가상 머신이 중지됩니다. 다른 인프라 요소를 구성하는 데 도움이되는 정보를 얻을 수 있습니다. 예를 들어 가상 머신의 MAC 주소를 가져 와서 가상 머신에 영구 IP 주소를 할당하도록 DHCP를 구성할 수 있습니다.

절차

1. inventory.yml 의 control_plane 및 compute 변수에서 type: high_performance의 두 인스턴스를 type: server로 변경합니다.

2. 선택 사항: 동일한 클러스터에 여러번 설치를 수행하려는 경우 각 OpenShift Container Platform 설치에 대해 서로 다른 템플릿을 생성합니다. inventory.yml 파일에서 template 값 앞에 infraID를 추가합니다. 예를 들면 다음과 같습니다.

```
control_plane:
  cluster: "{{ ovirt_cluster }}"
  memory: 16GiB
  sockets: 4
  cores: 1
```

```
    template: "{{ metadata.infraID }}-rhcos_tpl"
    operating_system: "rhcos_x64"
    ...
```

3. 템플릿 및 가상 머신을 만듭니다.

```
$ ansible-playbook -i inventory.yml create-templates-and-vms.yml
```

16.5.21. 부트스트랩 시스템 생성

bootstrap.yml Playbook을 실행하여 부트스트랩 머신을 생성합니다. 이 Playbook은 부트스트랩 가상 머신을 시작하고 assets 디렉터리에서 bootstrap.ign Ignition 파일을 전달합니다. 부트스트랩 노드는 컨트롤 플레인 노드에 Ignition 파일을 제공할 수 있도록 자체적으로 구성됩니다.

부트스트랩 프로세스를 모니터링하려면 RHV 관리 포털의 콘솔을 사용하거나 SSH를 사용하여 가상 머신에 연결합니다.

프로세스

1. 부트스트랩 시스템을 생성합니다.

```
$ ansible-playbook -i inventory.yml bootstrap.yml
```

2. 관리 포털 또는 SSH에서 콘솔을 사용하여 부트스트랩 머신에 연결합니다. <bootstrap_ip>를 부트스트랩 노드 IP 주소로 바꿉니다. SSH를 사용하려면 다음을 입력합니다.

```
$ ssh core@<boostrap.ip>
```

3. 부트스트랩 노드에서 릴리스 이미지 서비스에 대한bootkube.service journald 장치 로그를 수집합니다.

```
[core@ocp4-lk6b4-bootstrap ~]$ journalctl -b -f -u release-image.service -u
bootkube.service
```

참고

부트스트랩 노드의 **bootkube.service** 로그는 **etcd connection rejectd** 오류를 출력하고 부트스트랩 서버가 컨트롤 플레인 노드의 **etcd**에 연결할 수 없음을 나타냅니다. 각 컨트롤 플레인 노드에서 **etcd**를 시작하고 노드가 클러스터에 가입되면 오류가 중지됩니다.

16.5.22. 컨트롤 플레인 노드 생성

masters.yml Playbook을 실행하여 컨트롤 플레인 노드를 생성합니다. 이 **Playbook**은 각 가상 머신에 **master.ign Ignition** 파일을 전달합니다. **Ignition** 파일에는 **https://api-int.ocp4.example.org:22623/config/master**와 같은 **URL**에서 **Ignition**을 가져 오는 컨트롤 플레인 노드에 대한 지시문이 포함되어 있습니다. 이 **URL**의 포트 번호는 로드 밸런서에 의해 관리되며 클러스터 내에서만 액세스할 수 있습니다.

프로세스

1. 컨트롤 플레인 노드를 생성합니다.

```
$ ansible-playbook -i inventory.yml masters.yml
```

2. **Playbook**이 컨트롤 플레인을 생성하는 동안 부트스트랩 프로세스를 모니터링합니다.

```
$ openshift-install wait-for bootstrap-complete --dir $ASSETS_DIR
```

출력 예

```
INFO API v1.22.1 up
INFO Waiting up to 40m0s for bootstrapping to complete...
```

3. 컨트롤 플레인 노드 및 **etcd**의 모든 **pod**가 실행 중이면 설치 프로그램에 다음과 같은 출력이 표시됩니다.

출력 예

```
INFO It is now safe to remove the bootstrap resources
```

16.5.23. 클러스터 상태 확인

설치 중 또는 설치 후 OpenShift Container Platform 클러스터의 상태를 확인할 수 있습니다.

프로세스

1. 클러스터 환경에서 관리자의 kubeconfig 파일을 내보냅니다.

```
$ export KUBECONFIG=$ASSETS_DIR/auth/kubeconfig
```

kubeconfig 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다.

2. 배포 후 생성된 컨트롤 플레인 및 컴퓨팅 시스템을 확인합니다.

```
$ oc get nodes
```

3. 클러스터 버전을 확인합니다.

```
$ oc get clusterversion
```

4. Operator 상태를 확인합니다.

```
$ oc get clusteroperator
```

5. 클러스터에서 실행 중인 모든 Pod를 확인합니다.

```
$ oc get pods -A
```

16.5.24. 부트스트랩 시스템 제거

wait-for 명령이 부트스트랩 프로세스가 완료되었음을 표시한 후 부트스트랩 가상 머신을 제거하여 컴퓨팅, 메모리 및 스토리지 리소스를 확보해야 합니다. 또한 로드 밸런서 지시문에서 부트스트랩 시스템 설정을 제거해야 합니다.

프로세스

1. 클러스터에서 부트스트랩 시스템을 제거하려면 다음을 입력하십시오.

```
$ ansible-playbook -i inventory.yml retire-bootstrap.yml
```

2. 로드 밸런서 지시문에서 부트스트랩 머신의 설정을 제거합니다.

16.5.25. 작업자 노드 작성 및 설치 완료

작업자 노드 생성은 컨트롤 플레인 노드를 생성하는 것과 유사합니다. 그러나 작업자 노드는 클러스터에 자동으로 참여하지 않습니다. 클러스터에 추가하려면 작업자의 보류중인 CSR (인증서 서명 요청)을 검토하고 승인합니다.

첫 번째 요청을 승인한 후 모든 작업자 노드가 승인될 때까지 CSR을 계속 승인합니다. 이 프로세스를 완료하면 작업자 노드가 Ready 상태가 되고 해당 노드에서 실행되도록 Pod를 예약할 수 있습니다.

마지막으로 명령줄을 모니터링하여 설치 프로세스가 완료되는지 확인합니다.

프로세스

1. 작업자 노드를 생성합니다.

```
$ ansible-playbook -i inventory.yml workers.yml
```

2. 모든 CSR을 나열하려면 다음을 입력하십시오.

```
$ oc get csr -A
```

결국이 명령은 노드 당 하나의 CSR을 표시합니다. 예를 들면 다음과 같습니다.

출력 예

```
NAME        AGE    SIGNERNAME                                    REQUESTOR
CONDITION
csr-2lnxd   63m    kubernetes.io/kubelet-serving                 system:node:ocp4-lk6b4-
master0.ocp4.example.org                 Approved,Issued
csr-hff4q   64m    kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper
Approved,Issued
csr-hsn96   60m    kubernetes.io/kubelet-serving                 system:node:ocp4-lk6b4-
master2.ocp4.example.org                 Approved,Issued
csr-m724n   6m2s   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper
Pending
csr-p4dz2   60m    kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper
Approved,Issued
csr-t9vfj   60m    kubernetes.io/kubelet-serving                 system:node:ocp4-lk6b4-
master1.ocp4.example.org                 Approved,Issued
csr-tggtr   61m    kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper
Approved,Issued
csr-wcbrf   7m6s   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper
Pending
```

3. 목록을 필터링하고 보류중인 **CSR** 만 보려면 다음을 입력하십시오.

```
$ watch "oc get csr -A | grep pending -i"
```

이 명령은 **2** 초마다 출력을 새로 고침하고 보류중인 **CSR** 만 표시합니다. 예를 들면 다음과 같습니다.

출력 예

```
Every 2.0s: oc get csr -A | grep pending -i

csr-m724n   10m   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper
Pending
csr-wcbrf   11m   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper
Pending
```

4. 보류 중인 각 요청을 검사합니다. 예를 들면 다음과 같습니다.

출력 예

```
$ oc describe csr csr-m724n
```

출력 예

```
Name:               csr-m724n
Labels:             <none>
Annotations:        <none>
CreationTimestamp:  Sun, 19 Jul 2020 15:59:37 +0200
Requesting User:    system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper
Signer:             kubernetes.io/kube-apiserver-client-kubelet
Status:             Pending
Subject:
        Common Name:    system:node:ocp4-lk6b4-worker1.ocp4.example.org
        Serial Number:
        Organization:   system:nodes
Events:  <none>
```

5. CSR 정보가 정확하면 다음 요청을 승인합니다.

```
$ oc adm certificate approve csr-m724n
```

6. 설치 프로세스가 완료될 때까지 기다립니다.

```
$ openshift-install wait-for install-complete --dir $ASSETS_DIR --log-level debug
```

설치가 완료되면 명령행에 OpenShift Container Platform 웹 콘솔의 URL과 관리자 이름 및 암호가 표시됩니다.

16.5.26. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

16.5.27. 기본 OperatorHub 소스 비활성화

Red Hat 및 커뮤니티 프로젝트에서 제공하는 콘텐츠를 소싱하는 Operator 카탈로그는 OpenShift Container Platform을 설치하는 동안 기본적으로 OperatorHub용으로 구성됩니다. 제한된 네트워크 환경에서는 클러스터 관리자로서 기본 카탈로그를 비활성화해야 합니다.

프로세스

- OperatorHub 오브젝트에 disableAllDefaultSources: true를 추가하여 기본 카탈로그의 소스를 비활성화합니다.

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

작은 정보

또는 웹 콘솔을 사용하여 카탈로그 소스를 관리할 수 있습니다. 관리 → 클러스터 설정 → 구성 → OperatorHub 페이지에서 개별 소스를 생성, 삭제, 비활성화 및 활성화할 수 있는 소스 탭을 클릭합니다.

16.6. RHV의 클러스터 설치 제거

RHV(Red Hat Virtualization)에서 OpenShift Container Platform 클러스터를 제거할 수 있습니다.

16.6.1. 설치 관리자가 프로비저닝한 인프라를 사용하는 클러스터 제거

클라우드에서 설치 관리자 프로비저닝 인프라를 사용하는 클러스터를 제거할 수 있습니다.

참고

설치 제거 후 특히 UPI(User Provisioned Infrastructure) 클러스터에서 제거되지 않은 리소스에 대해 클라우드 공급자를 확인합니다. 설치 관리자가 생성하지 않았거나 설치 프로그램이 액세스할 수 없는 리소스가 있을 수 있습니다.

사전 요구 사항

- 클러스터를 배포하는 데 사용한 설치 프로그램의 사본을 준비합니다.
- 클러스터를 만들 때 설치 프로그램에 의해 생성된 파일을 준비합니다.

프로세스

1. 클러스터를 설치하는 데 사용한 컴퓨터에 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행합니다.

```
$ ./openshift-install destroy cluster \
--dir <installation_directory> --log-level info 1 2
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2 다른 세부 사항을 보려면 info 대신 warn, debug 또는 error를 지정합니다.

참고

클러스터의 클러스터 정의 파일이 포함되어 있는 디렉터리를 지정해야 합니다. 설치 프로그램이 클러스터를 삭제하려면 이 디렉터리에 있는 **metadata.json** 파일이 필요합니다.

2. 선택사항: **<installation_directory>** 디렉터리와 **OpenShift Container Platform** 설치 프로그램을 삭제합니다.

16.6.2. 사용자 프로비저닝 인프라를 사용하는 클러스터 제거

클러스터 사용을 완료하면 클라우드에서 설치 관리자 프로비저닝 인프라를 사용하는 클러스터를 제거할 수 있습니다.

사전 요구 사항

- 클러스터를 설치하는 데 사용한 원본 플레이 북 파일, **assets** 디렉터리 및 파일, **$ ASSETS_DIR** 환경 변수가 있어야 합니다. 일반적으로 클러스터를 설치할 때 사용한 것과 동일한 컴퓨터를 사용하여 이를 수행 할 수 있습니다.

프로세스

1. 클러스터를 제거하려면 다음을 입력하십시오.

```
$ ansible-playbook -i inventory.yml \
  retire-bootstrap.yml \
  retire-masters.yml   \
  retire-workers.yml
```

2. **DNS,** 로드 밸런서 및 클러스터의 기타 인프라에 추가한 모든 구성을 제거합니다.

17장. VSPHERE에 설치

17.1. VSPHERE에 설치 준비

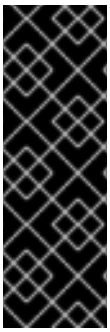
17.1.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.
- 방화벽을 사용하며 Telemetry를 사용할 예정이면 클러스터에 필요한 사이트를 허용하도록 방화벽을 구성해야 합니다.
- VMware 플랫폼 라이센스를 검토하셨습니다. Red Hat은 VMware 라이센스에 제한이 없지만 일부 VMware 인프라 구성 요소에는 라이센스가 필요합니다.

17.1.2. vSphere에 OpenShift Container Platform을 설치할 방법 선택

설치 관리자 프로비저닝 또는 사용자 프로비저닝 인프라를 사용하여 vSphere에 OpenShift Container Platform을 설치할 수 있습니다. 기본 설치 유형은 설치 프로그램이 클러스터의 기본 인프라를 프로비저닝하는 설치 관리자 프로비저닝 인프라를 사용합니다. 제공하는 인프라에 OpenShift Container Platform도 설치할 수 있습니다. 설치 프로그램에서 프로비저닝한 인프라를 사용하지 않는 경우 클러스터 리소스를 직접 관리하고 유지보수해야 합니다.

설치 관리자 프로비저닝 및 사용자 프로비저닝 설치 프로세스에 대한 자세한 내용은 설치 프로세스를 참조하십시오.



중요

사용자가 프로비저닝한 인프라 설치를 수행하는 단계는 예시용으로만 제공됩니다. 사용자가 제공하는 인프라를 사용하여 클러스터를 설치하려면 vSphere 플랫폼 및 OpenShift Container Platform 설치 프로세스에 대한 정보가 필요합니다. 사용자 프로비저닝 인프라 설치 지침을 가이드로 사용하십시오. 다른 방법을 통해 필요한 리소스를 자유롭게 생성할 수 있습니다.

17.1.2.1. 설치 관리자가 프로비저닝한 vSphere에 OpenShift Container Platform을 설치

설치 프로그램에서 프로비저닝한 인프라를 사용하면 설치 프로그램에서 OpenShift Container

Platform에 필요한 리소스의 프로비저닝을 사전 구성하고 자동화할 수 있습니다.

- vSphere에 클러스터 설치: 사용자 지정 없이 설치 관리자 프로비저닝 인프라 설치를 사용하여 vSphere에 OpenShift Container Platform을 설치할 수 있습니다.
- 사용자 지정으로 vSphere에 클러스터 설치: 기본 사용자 지정 옵션으로 설치 관리자 프로비저닝 인프라 설치를 사용하여 vSphere에 OpenShift Container Platform을 설치할 수 있습니다.
- 네트워크 사용자 지정으로 vSphere에 클러스터 설치: 설치 관리자 프로비저닝 vSphere 인프라에 네트워크 사용자 지정을 사용하여 OpenShift Container Platform을 설치할 수 있습니다. 설치 중에 OpenShift Container Platform 네트워크 구성을 사용자 지정할 수 있으므로 클러스터가 기존 IP 주소 할당과 공존하고 네트워크 요구 사항을 준수할 수 있습니다.
- 제한된 네트워크의 vSphere에 클러스터 설치: 설치 릴리스 컨텐츠의 내부 미러를 생성하여 제한된 네트워크의 VMware vSphere 인프라에 클러스터를 설치할 수 있습니다. 이 방법을 사용하여 인터넷에 표시되지 않는 내부 네트워크에 OpenShift Container Platform을 배포할 수 있습니다.

17.1.2.2. vSphere에서 OpenShift Container Platform의 사용자 프로비저닝 인프라 설치

사용자 프로비저닝 인프라를 사용하려면 OpenShift Container Platform에 필요한 모든 리소스를 프로비저닝해야 합니다.

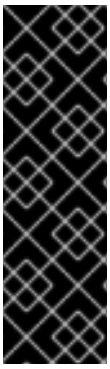
- 사용자 프로비저닝 인프라를 사용하여 vSphere에 클러스터 설치: 사용자가 제공하는 VMware vSphere 인프라에 OpenShift Container Platform을 설치할 수 있습니다.
- 사용자 프로비저닝 인프라가 있는 네트워크 사용자 지정 기능이 있는 vSphere에 클러스터 설치: 사용자 지정 네트워크 구성 옵션으로 프로비저닝하는 VMware vSphere 인프라에 OpenShift Container Platform을 설치할 수 있습니다.
- 사용자 프로비저닝 인프라가 있는 제한된 네트워크의 vSphere에 클러스터 설치: 제한된 네트워크에서 프로비저닝하는 VMware vSphere 인프라에 OpenShift Container Platform을 설치할 수 있습니다.

17.1.3. VMware vSphere 인프라 요구사항

사용하는 구성 요소의 요구사항을 충족하는 VMware vSphere 버전 6 또는 7 인스턴스에 OpenShift Container Platform 클러스터를 설치해야 합니다.

표 17.1. vSphere 가상 환경의 버전 요구 사항

가상 환경 제품	필요한 버전
VM 하드웨어 버전	13 이상
vSphere ESXi hosts	6.5 이상
vCenter 호스트	6.5 이상



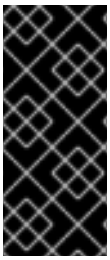
중요

이제 VMware vSphere 버전 6.7U2 또는 이전 버전에 클러스터를 설치하는 것이 더 이상 사용되지 않으며 가상 하드웨어 버전 13은 더 이상 사용되지 않습니다. 이러한 버전은 여전히 완전히 지원되지만 향후 OpenShift Container Platform 버전에서는 지원이 제거됩니다. 이제 OpenShift Container Platform의 vSphere 가상 머신의 하드웨어 버전 15가 기본값이 되었습니다. vSphere 노드의 하드웨어 버전을 업데이트하려면 " vSphere에서 실행되는 노드에서 하드웨어 업데이트" 문서를 참조하십시오.

표 17.2. VMware 구성 요소에 지원되는 최소 vSphere 버전

구성 요소	지원되는 최소 버전	설명
하이퍼바이저	HW 버전 13이 포함된 vSphere 6.5 이상	이 버전은 RHCOS(Red Hat Enterprise Linux CoreOS)가 지원하는 최소 버전입니다. Red Hat Enterprise Linux 8 지원 하이퍼바이저 목록을 참조하십시오.
In-tree 드라이버가 있는 스토리지	vSphere 6.5 이상	이 플러그인은 OpenShift Container Platform에 포함된 vSphere용 인트리 스토리지 드라이버를 사용하여 vSphere 스토리지를 생성합니다.
선택사항: 네트워킹(NSX-T)	vSphere 6.5U3 또는 vSphere 6.7U2 이상	OpenShift Container Platform에는 vSphere 6.5U3 또는 vSphere 6.7U2+가 필요합니다. NSX 및 OpenShift Container Platform의 호환성에 대한 자세한 내용은 VMware의 NSX 컨테이너 플러그인 설명서의 릴리스 노트 섹션을 참조하십시오.

vSphere 버전 6.5 인스턴스를 사용하는 경우 OpenShift Container Platform을 설치하기 전에 6.7U3 또는 7.0로 업그레이드하는 것이 좋습니다.



중요

OpenShift Container Platform을 설치하기 전에 ESXi 호스트의 시간이 동기화되었는지 확인해야 합니다. VMware 문서에서 Edit Time Configuration for a Host를 참조하십시오.

17.1.4. vSphere에서 OpenShift Container Platform의 설치 관리자 프로비저닝된 인프라 설치를 제거

- vSphere에서 설치 관리자 프로비저닝 인프라를 사용하는 클러스터 설치: 설치 관리자 프로비저닝 인프라를 사용하는 VMware vSphere 인프라에 배포한 클러스터를 제거할 수 있습니다.

17.2. VSPHERE에 클러스터 설치

OpenShift Container Platform 4.9 버전에서는 설치 관리자 프로비저닝 인프라를 사용하여 VMware vSphere 인스턴스에 클러스터를 설치할 수 있습니다.

참고

OpenShift Container Platform은 단일 VMware vCenter에만 클러스터 배포를 지원합니다. 여러 vCenter에 머신/머신 세트가 있는 클러스터 배포는 지원되지 않습니다.

17.2.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.
- 클러스터용 영구 스토리지를 프로비저닝합니다. 프라이빗 이미지 레지스트리를 배포하려면 스토리지에서 ReadWriteMany 액세스 모드를 제공해야 합니다.
- OpenShift Container Platform 설치 프로그램은 vCenter 및 ESXi 호스트의 포트 443에 액세스해야 합니다. 포트 443에 액세스할 수 있는지 확인했습니다.
- 방화벽을 사용하는 경우 관리자가 포트 443에 액세스할 수 있음을 확인합니다. 설치에 성공하려면 포트 443에서 컨트롤 플레인 노드가 vCenter 및 ESXi 호스트에 연결할 수 있어야 합니다.

- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성해야 합니다.

참고

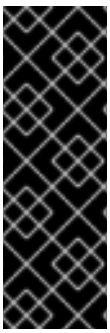
프록시를 구성하는 경우에도 해당 사이트 목록을 검토하십시오.

17.2.2. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.



중요

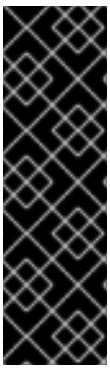
클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

17.2.3. VMware vSphere 인프라 요구사항

사용하는 구성 요소의 요구사항을 충족하는 VMware vSphere 버전 6 또는 7 인스턴스에 OpenShift Container Platform 클러스터를 설치해야 합니다.

표 17.3. vSphere 가상 환경의 버전 요구 사항

가상 환경 제품	필요한 버전
VM 하드웨어 버전	13 이상
vSphere ESXi hosts	6.5 이상
vCenter 호스트	6.5 이상



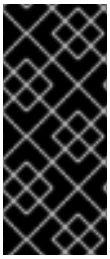
중요

이제 VMware vSphere 버전 6.7U2 또는 이전 버전에 클러스터를 설치하는 것이 더 이상 사용되지 않으며 가상 하드웨어 버전 13은 더 이상 사용되지 않습니다. 이러한 버전은 여전히 완전히 지원되지만 향후 OpenShift Container Platform 버전에서는 지원이 제거됩니다. 이제 OpenShift Container Platform의 vSphere 가상 머신의 하드웨어 버전 15가 기본값이 되었습니다. vSphere 노드의 하드웨어 버전을 업데이트하려면 " vSphere에서 실행되는 노드에서 하드웨어 업데이트" 문서를 참조하십시오.

표 17.4. VMware 구성 요소에 지원되는 최소 vSphere 버전

구성 요소	지원되는 최소 버전	설명
하이퍼바이저	HW 버전 13이 포함된 vSphere 6.5 이상	이 버전은 RHCOS(Red Hat Enterprise Linux CoreOS)가 지원하는 최소 버전입니다. Red Hat Enterprise Linux 8 지원 하이퍼바이저 목록을 참조하십시오.
In-tree 드라이버가 있는 스토리지	vSphere 6.5 이상	이 플러그인은 OpenShift Container Platform에 포함된 vSphere용 인트리 스토리지 드라이버를 사용하여 vSphere 스토리지를 생성합니다.
선택사항: 네트워킹(NSX-T)	vSphere 6.5U3 또는 vSphere 6.7U2 이상	OpenShift Container Platform에는 vSphere 6.5U3 또는 vSphere 6.7U2+가 필요합니다. NSX 및 OpenShift Container Platform의 호환성에 대한 자세한 내용은 VMware의 NSX 컨테이너 플러그인 설명서의 릴리스 노트 섹션을 참조하십시오.

vSphere 버전 6.5 인스턴스를 사용하는 경우 OpenShift Container Platform을 설치하기 전에 6.7U3 또는 7.0로 업그레이드하는 것이 좋습니다.



중요

OpenShift Container Platform을 설치하기 전에 **ESXi** 호스트의 시간이 동기화되었는지 확인해야 합니다. **VMware** 문서에서 **Edit Time Configuration for a Host**를 참조하십시오.

17.2.4. 네트워크 연결 요구사항

OpenShift Container Platform 클러스터 구성 요소가 통신할 수 있도록 시스템 간 네트워크 연결을 구성해야 합니다.

필요한 네트워크 포트에 대한 다음 세부 정보를 검토합니다.

표 17.5. 모든 시스템 간 통신에 사용되는 포트

프로토콜	포트	설명
ICMP	해당 없음	네트워크 연결성 테스트
TCP	**1936**	메트릭
	9000-9999	**9100-9101** 포트의 노드 내보내기 및 **9099** 포트의 Cluster Version Operator를 포함한 호스트 수준 서비스.
	10250-10259	Kubernetes에서 예약하는 기본 포트
	10256	openshift-sdn
UDP	**4789**	VXLAN(Virtual extensible LAN)
	6081	Geneve
	9000-9999	**9100-9101** 포트의 노드 내보내기를 포함한 호스트 수준 서비스.
	500	IPsec IKE 패킷
	4500	IPsec NAT-T 패킷
TCP/UDP	**30000-32767**	Kubernetes 노드 포트
ESP	해당 없음	IPsec Encapsulating Security Payload (ESP)

표 17.6. 모든 시스템과 컨트롤 플레인 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**6443**	Kubernetes API

표 17.7. 컨트롤 플레인 머신 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**2379-2380**	etcd 서버 및 피어 포트

추가 리소스

- **vSphere** 노드의 하드웨어 버전을 업데이트하려면 **vSphere**에서 실행 중인 노드에서 하드웨어 업데이트를 참조하십시오.

17.2.5. vCenter 요구사항

설치 관리자가 프로비저닝하는 인프라를 사용하는 **vCenter**에 **OpenShift Container Platform** 클러스터를 설치하기 전에 환경을 준비해야 합니다.

필요한 **vCenter** 계정 권한

vCenter에 **OpenShift Container Platform** 클러스터를 설치하려면 설치 프로그램에서 필요한 리소스를 읽고 생성할 수 있는 권한이 있는 계정에 액세스해야 합니다. 필요한 모든 권한에 액세스할 수 있는 가장 간단한 방법은 글로벌 관리 권한이 있는 계정을 사용하는 것입니다.

글로벌 관리 권한이 있는 계정을 사용할 수 없는 경우 **OpenShift Container Platform** 클러스터 설치에 필요한 권한을 부여하려면 역할을 생성해야 합니다. 대부분의 권한이 항상 필요하지만 기본 동작인 **vCenter** 인스턴스에 **OpenShift Container Platform** 클러스터를 포함하는 폴더를 프로비저닝할 설치 프로그램인 경우에만 필요합니다. 필요한 권한을 부여하려면 지정된 오브젝트에 대해 **vSphere** 역할을 생성하거나 수정해야 합니다.

설치 프로그램이 **vSphere** 가상 머신 폴더를 생성하는 경우 추가 역할이 필요합니다.

예 17.1. vSphere API에 설치에 필요한 역할 및 권한

역할의 vSphere 개체	필요한 경우	vSphere API에서 필요한 권한

역할의 vSphere 개체	필요한 경우	vSphere API에서 필요한 권한
vSphere vCenter	Always	CNS .**Searchable** **InventoryService.Tagging.AttachTag** **InventoryService.Tagging.CreateCategory** **InventoryService.Tagging.CreateTag** **InventoryService.Tagging.Delete**Tag **InventoryService.tagged.DeleteTag** **InventoryService.Tagging.EditCategory** **InventoryService.Tagging.EditTag**ECDHE**Session** **ValidateSession**Forwarded**StorageProfile.Update**Forwarded**StorageProfile.View**
vSphere vCenter Cluster	Always	**Host.Config.Storage** **Resource.AssignVMToPool** **VApp.AssignResourcePool** **VApp.Import** **VirtualMachine.Config.AddNewDisk**
vSphere Datastore	Always	**Datastore.AllocateSpace** **Datastore.LoadBalancer** **Datastore.FileManagement** **InventoryService.Tagging.ObjectAttachable**
vSphere Port Group	Always	**Network.Assign**

역할의 vSphere 개체	필요한 경우	vSphere API에서 필요한 권한
가상 머신 폴더	Always	**InventoryService.ume.ObjectAttachable Resource.AssignVMToPool VApp.Import VirtualMachine.Config.Add ExistingDisk VirtualMachine.Config.Add NewDisk VirtualMachine.Config.Add RemoveDevice VirtualMachine.Config.Advanced Config VirtualMachine.Config.Annotation ECDHE VirtualMachine.Config.DiskExtend** Forwarded VirtualMachine**.Config.** DiskExtend Forwarded **VirtualMachine.Config.DiskLease VirtualMachine.Config.ReCreate** VolumeClaim.Config**.MemoryDriver.**

역할의 vSphere 개체	필요한 경우	vSphere API에서 필요한 권한
vSphere vCenter Datacenter	설치 프로그램이 가상 머신 폴더를 생성하는 경우	**InventoryService.Tagging.ObjectAttachable Resource.AssignVMToPool VApp.Import VirtualMachine.Config.Add ExistingDisk VirtualMachine.Config.Add NewDisk VirtualMachine.Config.Add RemoveDevice VirtualMachine.Config.Adva ncedConfig VirtualMachine.Config.Anno tationECDHEVirtualMachine .Config.DiskExtend**Forwarde d VirtualMachine**.**Config**.DiskExt end ForwardedVirtualMachine.C onfig.DiskLeaseVirtualMach ine.Config.ReCreate VolumeClaim.Config.Reset**d efined**.**

예 17.2. vCenter 그래픽 사용자 인터페이스(GUI)에 설치하는 데 필요한 역할 및 권한

역할의 vSphere 개체	필요한 경우	vCenter GUI에서 필요한 권한
vSphere vCenter	Always	CNS **.Searchable "vSphere agged""Assign or unassign vSphere Tag" "vSphere tagging" "vSphere" "vSphere" "vSphere"CreatevSphere Tag" vSphere 태그""vSphere" 삭제 태그 지정" "vSphere Assignment""Edit vSphere Tag" "vSphere Tagging" "vSphere 태그" 세션" "Profile 기반 스토리지""Profile 기반 스토리지 업데이트""Profile 기반 스토리지"Profile 중심 스토리지 업데이트"Profile 기반 스토리지 업데이트"Profile 기반 스토리지 업데이트 스토리지 보기"**
vSphere vCenter Cluster	클러스터 루트에서 VM이 생성되는 경우	**Host.Configuration"Storage partition configuration" 리소스" 리소스 풀에 가상 머신을 서명" VApp"Assign resource pool" VApp.Import"Virtual machine""Change Configuration"."새 디스크 추가"**
vSphere vCenter 리소스 풀	기존 리소스 풀이 제공되는 경우	**Host.Configuration"Storage partition configuration" 리소스" 리소스 풀에 가상 머신을 서명" VApp"Assign resource pool" VApp.Import"Virtual machine""Change Configuration"."새 디스크 추가"**

역할의 vSphere 개체	필요한 경우	vCenter GUI에서 필요한 권한
vSphere Datastore	Always	**Datastore"Allocate space"** 데이터**저장소"** **데이터 저장소"** **Datastore"Low level file operations""vSphere Assignment"Assign or Unassign vSphere Tag on Object"**
vSphere Port Group	Always	**네트워크 "Assign network"**
가상 머신 폴더	Always	**"vSphere Assignment""Assign or unassign vSphere Tag on Object"** **Resource"Assign virtual machine to resource pool"**: **VApp.Import** **"Virtual machine""Change Configuration","Add existing disk""**Add existing disk**""Add new disk""Add new disk" "Virtual""** **머신 "시스템 변경", "가상 머신""** **"가상 머신""시스템 변경"** **"가상 머신""변경 구성",** **"가상 머신", 변경, CPU 수 변경"** **"가상 머신", 가상 디스크 변경", 가상 디스크 변경"** **"가상머신". 머신 변경 "구성 변경"** **"가상 머신""마이크 구성 변경"** **"가상 머신""마이크 구성 변경" "구성 변경",** **"가상 머신", 변경" 디스크 삭제"** **"가상 머신".변경 구성".Rename** **"가상 머신 변경" VIX API**로가상 머신 변경**"Gregradevirtual machine"Guestoperating system management by VIX API**

역할의 vSphere 개체	필요한 경우	vCenter GUI에서 필요한 권한
		"Virtual machine". Interconnection"Power on" "Virtual machine". interactionion.Reset "Virtual machine"."Edit Inventory"."Create new" "Virtual machine"."Create existing" "Virtual machine"Edit Inventory""Create from existing" "Virtual machine"Edit Inventory""Create from existing" virtual machine".Edit Inventory""Create from existing" virtual machine".Edit Inventory""Create from existing""Virtual machine" machine".Provisioning."Clone 가상 머신" "가상 머신".Provisioning." "가상 머신".Provisioning."Deploy 템플릿"
vSphere vCenter Datacenter	설치 프로그램이 가상 머신 폴더를 생성하는 경우	**"vSphere Assignment""Assign or unassign vSphere Tag on Object" Resource"Assign virtual machine to resource pool"**: **VApp.Import "Virtual machine""Change Configuration","Add existing disk""**Add existing disk**""Add new disk""Add new disk" "Virtual""** **머신 "시스템 변경", "가상 머신"" "가상 머신""시스템 변경" "가상 머신""변경 구성", "가상 머신", 변경, CPU 수 변경" "가상 머신", 가상 디스크 변경", 가상 디스크 변경" "가상머신". 머신 변경 "구성 변경" "가상 머신""마이크 구성 변경" "가상 머신""마이크 구성 변경" "구성 변경",**

역할의 vSphere 개체	필요한 경우	"가상 머신", 변경" 디스크 삭 vCenter GUI에서 필요한 권한
		"가상 머신".변경 구 성".Rename "가상 머신 변경" VIX API로가 상 머신 변 경"Gregradevirtual machine"Guestoperating system management by VIX API machine". interaction" "Virtual machine". Interconnection"Power on" "Virtual machine". interactionion.Reset "Virtual machine"."Edit Inventory"."Create new" "Virtual machine"."Create existing" "Virtual machine"Edit Inventory""Create from existing" "Virtual machine"Edit Inventory""Create from existing" virtual machine".Edit Inventory""Create from existing" virtual machine".Edit Inventory""Create from existing""Virtual machine" machine".Provisioning."Clo ne 가상 머신" "가상 머 신".Provisioning."Deploy 템 플릿" "가상 머신".Provisioning" 폴더 만들기" 폴더 생성"폴더 삭제"

또한 사용자에게 일부 **ReadOnly** 권한이 필요하며 일부 역할에는 하위 오브젝트에 권한을 부여하는 사용 권한이 필요합니다. 이러한 설정은 클러스터를 기존 폴더에 설치할지 여부에 따라 달라집니다.

예 17.3. 필수 권한 및 권한 부여 설정

vSphere 오브젝트	폴더 유형	하위 항목으로 권한 부여	권한 필요
vSphere vCenter	Always	False	나열된 필수 권한
vSphere vCenter Datacenter	기존 폴더	False	**ReadOnly** 권한
	설치 프로그램은 폴더를 생성	True	나열된 필수 권한
vSphere vCenter Cluster	Always	True	나열된 필수 권한
vSphere vCenter Datastore	Always	False	나열된 필수 권한
vSphere Switch	Always	False	**ReadOnly** 권한
vSphere Port Group	Always	False	나열된 필수 권한
vSphere vCenter Virtual Machine Folder	기존 폴더	True	나열된 필수 권한

필요한 권한만으로 계정을 생성하는 방법에 대한 자세한 내용은 **vSphere** 문서에서 **vSphere 권한 및 사용자 관리 작업**을 참조하십시오.

vMotion으로 OpenShift Container Platform 사용

vSphere 환경에서 **vMotion**을 사용하려는 경우 **OpenShift Container Platform** 클러스터를 설치하기 전에 다음을 고려하십시오.

- **OpenShift Container Platform**은 일반적으로 **compute-only vMotion** 을 지원합니다. 스토리지 **vMotion**을 사용하면 문제가 발생할 수 있으며 지원되지 않습니다.

  컴퓨팅 및 컨트롤 플레인 노드의 가동 시간을 보장하기 위해 **vMotion**에 대한 **VMware** 모범 사례를 따르는 것이 좋습니다. 또한 **VMware** 유사성 방지 규칙을 사용하여 유지 관리 또는 하드웨어 문제 중에 **OpenShift Container Platform**의 가용성을 개선하는 것이 좋습니다.

  vMotion 및 유사성 방지 규칙에 대한 자세한 내용은 **vMotion 네트워킹 요구 사항** 및 **VM 유사성 방지 규칙에** 대한 **VMware vSphere** 설명서를 참조하십시오.

- **Pod**에서 **vSphere** 볼륨을 사용하는 경우 데이터 저장소를 통해 **VM**을 마이그레이션하면 수

동으로 또는 스토리지 **vMotion**을 통해 **VM**을 마이그레이션하면 **OpenShift Container Platform PV(**영구 볼륨**)** 오브젝트 내에서 잘못된 참조가 표시됩니다**.** 이러한 참조는 영향을 받는 포드가 시작되지 않게 하고 데이터 손실을 초래할 수 있습니다**.**

- 마찬가지로 **OpenShift Container Platform**은 데이터 저장소에서 **VMDK**의 선택적 마이그레이션을 지원하지 않으며 **VM** 프로비저닝 또는 **PV**의 동적 또는 정적 프로비저닝을 위해 데이터 저장소 클러스터의 일부 또는 **PV**의 동적 또는 정적 프로비저닝을 위해 데이터 저장소 클러스터의 일부인 데이터 저장소를 사용하는 경우도 있습니다**.**

클러스터 리소스

설치 관리자 프로비저닝 인프라를 사용하는 **OpenShift Container Platform** 클러스터를 배포할 때는 설치 프로그램이 **vCenter** 인스턴스에서 여러 리소스를 생성할 수 있어야 합니다**.**

표준 **OpenShift Container Platform**을 설치하면 다음 **vCenter** 리소스가 생성됩니다**.**

- 폴더 한 개
- 태그 카테고리 한 개
- 태그 한 개
- 가상 머신**:**
  - 템플릿 한 개
  - 임시 부트스트랩 노드 한 개
  - 컨트롤 플레인 노드 세 개
  - 컴퓨팅 시스템 세 개

이러한 리소스는 **856GB**의 스토리지를 사용하지만 부트스트랩 노드는 클러스터 설치 프로세스 중에 제거됩니다**.** 표준 클러스터를 사용하려면 최소 **800GB**의 스토리지가 필요합니다**.**

컴퓨팅 시스템을 더 많이 배포할수록 **OpenShift Container Platform** 클러스터는 더 많은 스토리지를 사용합니다.

클러스터 제한

사용 가능한 리소스는 클러스터마다 다릅니다. **vCenter** 내에서 가능한 클러스터 수는 주로 사용 가능한 스토리지 공간과 필요한 리소스 수에 대한 제한으로 제한됩니다. 클러스터가 생성하는 **vCenter** 리소스와 **IP** 주소 및 네트워크와 같이 클러스터를 배포하는 데 필요한 리소스에 대한 제한 사항을 모두 고려해야 합니다.

네트워킹 요구사항

네트워크에 **DHCP**를 사용하고 **DHCP** 서버가 클러스터 시스템에 영구 **IP** 주소를 제공하도록 구성되어 있는지 확인합니다. **DHCP** 서버를 사용하도록 기본 게이트웨이를 구성해야 합니다. 모든 노드는 동일한 **VLAN**에 있어야 합니다. 두 번째 **VLAN**을 **Day 2** 작업으로 사용하여 클러스터를 확장할 수 없습니다. 또한 **OpenShift Container Platform** 클러스터를 설치하기 전에 다음 네트워킹 리소스를 생성해야 합니다.

참고

클러스터의 각 **OpenShift Container Platform** 노드는 **DHCP**를 통해 검색할 수 있는 **NTP(Network Time Protocol)** 서버에 액세스할 수 있어야 합니다. **NTP** 서버 없이도 설치할 수 있습니다. 그러나 비동기 서버 클록으로 인해 **NTP** 서버가 차단하는 오류가 발생합니다.

필요한 IP 주소

설치 관리자 프로비저닝 **vSphere** 설치에는 다음 두 개의 고정 **IP** 주소가 필요합니다.

- **API** 주소는 클러스터 **API**에 액세스하는 데 사용됩니다.
- **Ingress** 주소는 클러스터 인그레스 트래픽에 사용됩니다.

OpenShift Container Platform 클러스터를 설치할 때 설치 프로그램에 이 **IP** 주소를 제공해야 합니다.

DNS 레코드

OpenShift Container Platform 클러스터를 호스팅하는 **vCenter** 인스턴스에 적절한 **DNS** 서버에서 고정 **IP** 주소 두 개의 **DNS** 레코드를 생성해야 합니다. 각 레코드에서 **<cluster_name>**은 클러스터 이름

이고 <base_domain>은 클러스터를 설치할 때 지정하는 클러스터 기본 도메인입니다. 전체 DNS 레코드는 <component>.<cluster_name>.<base_domain> 형식입니다.

표 17.8. 필수 DNS 레코드

구성 요소	레코드	설명
API VIP	api.<cluster_name>.<base_domain>.	이 DNS A/AAAA 또는 CNAME 레코드는 컨트롤 플레인 시스템의 로드 밸런서를 가리켜야 합니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.
Ingress VIP	*.apps.<cluster_name>.<base_domain>.	기본적으로 작업자 노드인 인그레스 라우터 Pod를 실행하는 시스템을 대상으로 하는 로드 밸런서를 가리키는 와일드카드 DNS A/AAAA 또는 CNAME 레코드입니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.

17.2.6. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 core 사용자의 ~/.ssh/authorized_keys 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 core로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. ./openshift-install gather 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

참고

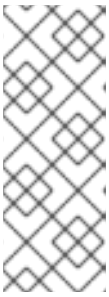
AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

절차

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: ~/.ssh/id_ed25519)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 '~/.ssh 디렉터리에 있는지 확인하십시오.



참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 ed25519 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 rsa 또는 ecdsa 알고리즘을 사용하는 키를 생성합니다.

2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 ~/.ssh/id_ed25519.pub 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 ./openshift-install gather 명령을 사용하려는 경우 필요합니다.

참고

일부 배포에서는 ~/.ssh/id_rsa 및 ~/.ssh/id_dsa와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. ssh-agent 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. ssh-agent에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: ~/.ssh/id_ed25519).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

17.2.7. 설치 프로그램 받기

OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

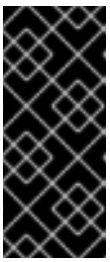
- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

절차

1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.

2. 인프라 공급자를 선택합니다.

3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.



중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.

중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 OpenShift Container Platform 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

17.2.8. 시스템 신뢰에 vCenter 루트 CA 인증서 추가

설치 프로그램이 vCenter API에 액세스해야 하므로 OpenShift Container Platform 클러스터를 설치하기 전에 vCenter의 신뢰할 수 있는 루트 CA 인증서를 시스템 신뢰에 추가해야 합니다.

절차

1. vCenter 홈 페이지에서 vCenter의 루트 CA 인증서를 다운로드합니다. vSphere Web Services SDK 섹션에서 Download trusted root CA certificates를 클릭합니다. <vCenter>/certs/download.zip 파일이 다운로드됩니다.

2. vCenter 루트 CA 인증서가 포함된 압축 파일의 압축을 풉니다. 압축 파일의 내용은 다음 파일 구조와 유사합니다.

```
certs
├── lin
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
├── mac
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
└── win
    ├── 108f4d17.0.crt
    ├── 108f4d17.r1.crl
    ├── 7e757f6a.0.crt
    ├── 8e4f8471.0.crt
    └── 8e4f8471.r0.crl

3 directories, 15 files
```

3. 운영 체제 파일을 시스템 신뢰에 추가합니다. 예를 들어 Fedora 운영 체제에서는 다음 명령을 실행합니다.

```
# cp certs/lin/* /etc/pki/ca-trust/source/anchors
```

4. 시스템 신뢰를 업데이트합니다. 예를 들어 **Fedora** 운영 체제에서는 다음 명령을 실행합니다.

```
# update-ca-trust extract
```

17.2.9. 클러스터 배포

호환되는 클라우드 플랫폼에 **OpenShift Container Platform**을 설치할 수 있습니다.

중요

최초 설치 과정에서 설치 프로그램의 **create cluster** 명령을 한 번만 실행할 수 있습니다.

사전 요구 사항

- **OpenShift Container Platform** 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

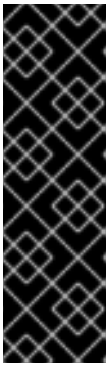
절차

1. 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터 배포를 초기화합니다.

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
  --log-level=info 2
```

1 **<installation_directory>**는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.

2 다른 설치 세부 사항을 보려면 **info** 대신 **warn, debug** 또는 **error**를 지정합니다.

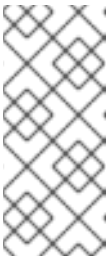


중요

비어 있는 디렉터리를 지정합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

화면에 나타나는 지시에 따라 필요한 값을 입력합니다.

a. 선택사항: 클러스터 시스템에 액세스하는 데 사용할 SSH 키를 선택합니다.



참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 ssh-agent 프로세스가 사용하는 SSH 키를 지정합니다.

b. 대상 플랫폼으로 vsphere를 선택합니다.

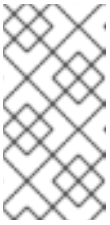
c. vCenter 인스턴스의 이름을 지정합니다.

d. 클러스터를 생성하는 데 필요한 권한이 있는 vCenter 계정의 사용자 이름과 암호를 지정합니다.

설치 프로그램이 vCenter 인스턴스에 연결됩니다.

e. 연결할 vCenter 인스턴스에서 데이터 센터를 선택합니다.

f. 사용할 기본 vCenter 데이터 저장소를 선택합니다.



참고

Datastore 및 클러스터 이름은 **60**자를 초과할 수 없으므로 결합된 문자열 길이가 **60**자 제한을 초과하지 않도록 합니다.

g. **OpenShift Container Platform** 클러스터를 설치할 **vCenter** 클러스터를 선택합니다. 설치 프로그램은 **vSphere** 클러스터의 루트 리소스 풀을 기본 리소스 풀로 사용합니다.

h. **vCenter** 인스턴스에서 구성한 가상 **IP** 주소 및 **DNS** 레코드가 포함된 네트워크를 선택합니다.

i. 컨트롤 플레인 **API** 액세스를 위해 구성한 가상 **IP** 주소를 입력합니다.

j. 클러스터 인그레스용으로 구성한 가상 **IP** 주소를 입력합니다.

k. 기본 도메인을 입력합니다. 이 기본 도메인은 구성한 **DNS** 레코드에서 사용한 것과 동일해야 합니다.

l. 클러스터를 설명할 수 있는 이름을 입력합니다. 클러스터 이름은 구성한 **DNS** 레코드에서 사용한 것과 동일해야 합니다.

참고

Datastore 및 클러스터 이름은 **60**자를 초과할 수 없으므로 결합된 문자열 길이가 **60**자 제한을 초과하지 않도록 합니다.

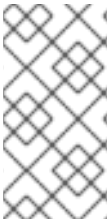
m. Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 붙여넣습니다.

+ 클러스터 배포가 완료되면 웹 콘솔에 대한 링크 및 **kubeadmin** 사용자의 인증 정보가 포함된 클러스터 액세스 지침이 터미널에 표시됩니다.

+ .출력 예

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-Wt5AL"
INFO Time elapsed: 36m22s
```

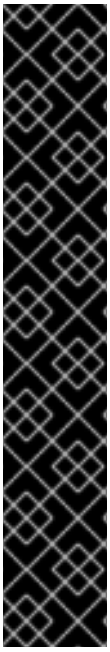
+



참고

설치에 성공하면 클러스터 액세스 및 인증 정보도 **<installation_directory>/.openshift_install.log**로 출력됩니다.

+



중요

- 설치 프로그램에서 생성하는 **Ignition** 구성 파일에 **24**시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 **24**시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 **kubelet** 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청**(CSR)**을 수동으로 승인해야 합니다. 자세한 내용은 ***Recovering from expired control plane certificates*** 문서를 참조하십시오.

- 클러스터를 설치한 후 **24**시간에서 **22**시간까지의 인증서가 교체되기 때문에 생성된 후 **12**시간 이내에 **Ignition** 구성 파일을 사용하는 것이 좋습니다. **12**시간 이내에 **Ignition** 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

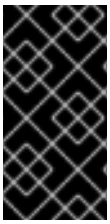
+

중요

설치 프로그램에서 생성되는 파일이나 설치 프로그램을 삭제해서는 안 됩니다. 클러스터를 삭제하려면 두 가지가 모두 필요합니다.

17.2.10. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift CLI(oc)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 oc를 설치할 수 있습니다.



중요

이전 버전의 oc를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 oc를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 Linux에서 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

절차

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.
2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. OpenShift v4.9 Linux Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. oc 바이너리를 PATH에 있는 디렉터리에 배치합니다.

   PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치

다음 절차에 따라 Windows에 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

절차

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.
2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. OpenShift v4.9 Windows Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.
4. ZIP 프로그램으로 아카이브의 압축을 풉니다.
5. oc 바이너리를 PATH에 있는 디렉터리로 이동합니다.

   PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치

다음 절차에 따라 macOS에서 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

절차

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.
2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3.

OpenShift v4.9 MacOSX Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.

4. 아카이브의 압축을 해제하고 압축을 풉니다.

5. **oc** 바이너리 **PATH**의 디렉터리로 이동합니다.

   PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

17.2.11. CLI를 사용하여 클러스터에 로그인

클러스터 **kubeconfig** 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. **kubeconfig** 파일에는 **CLI**에서 올바른 클러스터 및 **API** 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 **OpenShift Container Platform** 설치 과정에서 생성됩니다.

사전 요구 사항

- **OpenShift Container Platform** 클러스터를 배포했습니다.
- **oc CLI**를 설치했습니다.

프로세스

1. **kubeadmin** 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1. **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 **oc** 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

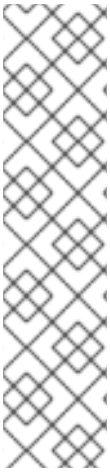
17.2.12. 레지스트리 스토리지 생성

클러스터를 설치한 후 레지스트리 **Operator**를 위한 스토리지를 생성해야 합니다.

17.2.12.1. 설치 중 제거된 이미지 레지스트리

공유 가능한 개체 스토리지를 제공하지 않는 플랫폼에서 **OpenShift Image Registry Operator**는 자체적으로 **Removed**로 부트스트랩합니다. 이를 통해 **openshift-installer**가 이러한 플랫폼 유형에서 설치를 완료할 수 있습니다.

설치 후 **managementState**를 **Removed**에서 **Managed**로 전환하도록 **Image Registry Operator** 구성을 편집해야 합니다.



참고

Prometheus 콘솔은 **ImageRegistryRemoved** 경고를 제공합니다. 다음은 예시 경고입니다.

"이미지 레지스트리가 제거되었습니다. **ImageStreamTags**를 참조하는 **ImageStreamTags, BuildConfigs** 및 **DeploymentConfigs**가 예상대로 작동하지 않을 수 있습니다. 스토리지를 구성하고 **configs.imageregistry.operator.openshift.io**를 편집하여 구성을 **Managed** 상태로 업데이트하십시오."

17.2.12.2. 이미지 레지스트리 스토리지 구성

기본 스토리지를 제공하지 않는 플랫폼에서는 처음에 **Image Registry Operator**를 사용할 수 없습니

다. 설치한 후에 스토리지를 사용하도록 레지스트리를 구성하여 **Registry Operator**를 사용 가능하도록 만들어야 합니다.

프로덕션 클러스터에 필요한 영구 볼륨을 구성하는 과정의 지침이 표시됩니다. 해당하는 경우, 프로덕션 환경 외 클러스터에서만 사용할 수 있는 저장 위치로서 빈 디렉터리를 구성하는 과정의 지침이 표시됩니다.

업그레이드 중에 **Recreate** 롤아웃 전략을 사용하여 이미지 레지스트리의 블록 스토리지 유형 사용을 허용하기 위한 추가 지침이 제공됩니다.

17.2.12.2.1. VMware vSphere용 레지스트리 스토리지 구성

클러스터 관리자는 설치한 후 스토리지를 사용하도록 레지스트리를 구성해야 합니다.

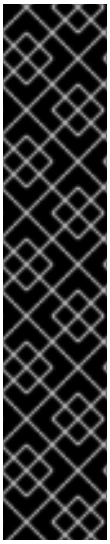
사전 요구 사항

- 클러스터 관리자 권한이 있어야 합니다.
- **VMware vSphere**에 클러스터가 있어야 합니다.
- **Red Hat OpenShift Container Storage**와 같이 클러스터용 영구 스토리지 프로비저닝.

중요

OpenShift Container Platform은 복제본이 하나만 있는 경우 이미지 레지스트리 스토리지에 대한 **ReadWriteOnce** 액세스를 지원합니다. **ReadWriteOnce** 액세스에서는 레지스트리가 **Recreate** 롤아웃 전략을 사용해야 합니다. 두 개 이상의 복제본으로 고 가용성을 지원하는 이미지 레지스트리를 배포하려면 **ReadWriteMany** 액세스가 필요합니다.

- **"100Gi"** 용량이 필요합니다.



중요

테스트 결과, RHEL의 NFS 서버를 핵심 서비스용 스토리지 백엔드로 사용하는 데 문제가 있는 것으로 나타납니다. 여기에는 OpenShift Container Registry and Quay, 스토리지 모니터링을 위한 Prometheus, 로깅 스토리지를 위한 Elasticsearch가 포함됩니다. 따라서 RHEL NFS를 사용하여 핵심 서비스에서 사용하는 PV를 백업하는 것은 권장되지 않습니다.

마켓플레이스의 다른 NFS 구현에는 이러한 문제가 나타나지 않을 수 있습니다. 이러한 OpenShift Container Platform 핵심 구성 요소에 대해 완료된 테스트에 대한 자세한 내용은 개별 NFS 구현 공급업체에 문의하십시오.

프로세스

1. 스토리지를 사용하도록 레지스트리를 구성하기 위해 configs.imageregistry/cluster 리소스에서 spec.storage.pvc를 변경합니다.

참고

공유 스토리지를 사용할 때 보안 설정을 확인하여 외부에서의 액세스를 방지합니다.

2. 레지스트리 pod가 없는지 확인합니다.

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

출력 예

```
No resourses found in openshift-image-registry namespace
```

참고

출력에 레지스트리 Pod가 있는 경우 이 절차를 계속할 필요가 없습니다.

3. 레지스트리 구성을 확인합니다.

```
$ oc edit configs.imageregistry.operator.openshift.io
```

출력 예

```
storage:
  pvc:
    claim: 1
```

1 image-registry-storage 영구 볼륨 클레임 (PVC)을 자동으로 생성할 수 있도록 claim 필드를 비워 둡니다. PVC는 기본 스토리지 클래스를 기반으로 생성됩니다. 그러나 기본 스토리지 클래스에서 RADOS 블록 장치(RBD)와 같은 ReadWriteOnce(ReadWriteOnce) 볼륨을 제공할 수 있으므로 복제본이 두 개 이상될 때 문제가 발생할 수 있습니다.

4. clusteroperator 상태를 확인합니다.

```
$ oc get clusteroperator image-registry
```

출력 예

```
NAME             VERSION                             AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                                 True        False         False      6h50m
```

17.2.12.2.2. VMware vSphere용 블록 레지스트리 스토리지 구성

클러스터 관리자로서 업그레이드 중에 이미지 레지스트리가 vSphere VMDK(Virtual Machine Disk)와 같은 블록 스토리지 유형을 사용할 수 있도록 허용하기 위해 Recreate 롤아웃 전략을 사용할 수 있습니다.

중요

블록 스토리지 볼륨이 지원되지만 프로덕션 클러스터에서 이미지 레지스트리와 함께 사용하는 것은 권장되지 않습니다. 레지스트리가 블록 스토리지에 구성된 설치는 레지스트리가 둘 이상의 복제본을 가질 수 없기 때문에 가용성이 높지 않습니다.

프로세스

1. 이미지 레지스트리 스토리지를 블록 스토리지 유형으로 설정하려면 **Recreate** 롤아웃 전략을 사용하고 복제본 **1**개 만으로 실행되도록 레지스트리를 패치합니다.

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. 블록 스토리지 장치에 **PV**를 프로비저닝하고 해당 볼륨의 **PVC**를 생성합니다. 요청된 블록 볼륨은 **RWO(ReadWriteOnce)** 액세스 모드를 사용합니다.

   a. **VMware vSphere PersistentVolumeClaim** 개체를 정의하려면 다음 내용이 포함된 **pvc.yaml** 파일을 생성합니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage 1
  namespace: openshift-image-registry 2
spec:
  accessModes:
  - ReadWriteOnce 3
  resources:
    requests:
      storage: 100Gi 4
```

1 **PersistentVolumeClaim** 개체를 표시하는 고유한 이름입니다.

2 **PersistentVolumeClaim** 오브젝트의 네임스페이스로 **openshift-image-registry**입니다.

3 영구 볼륨 클레임의 액세스 모드입니다. **ReadWriteOnce**를 사용하면 단일 노드에서 읽기 및 쓰기 권한으로 볼륨을 마운트할 수 있습니다.

4

영구 볼륨 클레임의 크기입니다.

b. 파일에서 **PersistentVolumeClaim** 오브젝트를 만듭니다.

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 올바른 **PVC**를 참조하도록 레지스트리 설정을 편집합니다.

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

출력 예

```
storage:
  pvc:
    claim: 1
```

1

사용자 지정 **PVC**를 만들면 **image-registry-storage PVC**의 기본 자동 생성을 위해 **claim** 필드를 비워둘 수 있습니다.

올바른 **PVC**를 참조하도록 레지스트리 스토리지를 구성하는 방법은 **vSphere** 용 레지스트리 구성을 참조하십시오.

17.2.13. VMware vSphere 볼륨 백업

OpenShift Container Platform은 새 볼륨을 독립 영구 디스크로 프로비저닝하여 클러스터의 모든 노드에서 볼륨을 자유롭게 연결 및 분리합니다. 결과적으로 스냅샷을 사용하는 볼륨을 백업하거나 스냅샷에서 볼륨을 복원할 수 없습니다. 자세한 내용은 스냅 샷 제한 을 참조하십시오.

프로세스

영구 볼륨의 백업을 생성하려면 다음을 수행합니다.

1. 영구 볼륨을 사용중인 애플리케이션을 중지합니다.
2. 영구 볼륨을 복제합니다.
3. 애플리케이션을 다시 시작합니다.
4. 복제된 볼륨의 백업을 만듭니다.
5. 복제된 볼륨을 삭제합니다.

17.2.14. 스틸 클럭 계정

기본적으로 설치 프로그램은 스틸 클럭 회계 매개 변수 (**stealclock.enabled**)를 활성화하지 않고 클러스터의 가상 머신을 프로비저닝합니다. 스틸링 클럭 계정을 사용하면 클러스터 문제 해결에 도움이 될 수 있습니다. 클러스터가 배포된 후 **vSphere Client**를 사용하여 각 가상 머신에서 이 매개변수를 활성화합니다.

자세한 내용은 이 **Red Hat** 기술 자료 문서 를 참조하십시오.

17.2.15. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 **Telemetry** 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 **Telemetry**가 자동으로 실행되고 **OpenShift Cluster Manager** 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 **OpenShift Cluster Manager**를 사용하여 자동으로 또는 **OpenShift Cluster Manager**를 사용하여 수동으로 유지 관리되는지 확인한 후 **subscription watch**를 사용하여 계정 또는 다중 클러스터 수준에서 **OpenShift Container Platform** 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

17.2.16. 다음 단계

- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.
- 레지스트리를 설정하고 레지스트리 스토리지를 구성합니다.
- 선택 사항: vSphere Problem Detector Operator에서 이벤트를 보고 클러스터에 권한 또는 스토리지 구성 문제가 있는지 확인합니다.

17.3. 사용자 지정으로 VSPHERE에 클러스터 설치

OpenShift Container Platform 4.9 버전에서는 설치 관리자 프로비저닝 인프라를 사용하여 VMware vSphere 인스턴스에 클러스터를 설치할 수 있습니다. 설치를 사용자 지정하려면 클러스터를 설치하기 전에 install-config.yaml 파일에서 매개변수를 수정합니다.

참고

OpenShift Container Platform은 단일 VMware vCenter에만 클러스터 배포를 지원합니다. 여러 vCenter에 머신/머신 세트가 있는 클러스터 배포는 지원되지 않습니다.

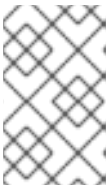
17.3.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.
- 클러스터용 영구 스토리지를 프로비저닝합니다. 프라이빗 이미지 레지스트리를 배포하려면 스토리지에서 ReadWriteMany 액세스 모드를 제공해야 합니다.
- OpenShift Container Platform 설치 프로그램은 vCenter 및 ESXi 호스트의 포트 443에 액

세스해야 합니다. 포트 443에 액세스할 수 있는지 확인했습니다.

- 방화벽을 사용하는 경우 관리자가 포트 443에 액세스할 수 있음을 확인합니다. 설치에 성공하려면 포트 443에서 컨트롤 플레인 노드가 vCenter 및 ESXi 호스트에 연결할 수 있어야 합니다.

- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성해야 합니다.



참고

프록시를 구성하는 경우에도 해당 사이트 목록을 검토하십시오.

17.3.2. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.

- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.

- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.

중요

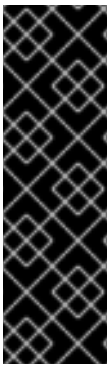
클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

17.3.3. VMware vSphere 인프라 요구사항

사용하는 구성 요소의 요구사항을 충족하는 VMware vSphere 버전 6 또는 7 인스턴스에 OpenShift Container Platform 클러스터를 설치해야 합니다.

표 17.9. vSphere 가상 환경의 버전 요구 사항

가상 환경 제품	필요한 버전
VM 하드웨어 버전	13 이상
vSphere ESXi hosts	6.5 이상
vCenter 호스트	6.5 이상



중요

이제 VMware vSphere 버전 6.7U2 또는 이전 버전에 클러스터를 설치하는 것이 더 이상 사용되지 않으며 가상 하드웨어 버전 13은 더 이상 사용되지 않습니다. 이러한 버전은 여전히 완전히 지원되지만 향후 OpenShift Container Platform 버전에서는 지원이 제거됩니다. 이제 OpenShift Container Platform의 vSphere 가상 머신의 하드웨어 버전 15가 기본값이 되었습니다. vSphere 노드의 하드웨어 버전을 업데이트하려면 " vSphere에서 실행되는 노드에서 하드웨어 업데이트" 문서를 참조하십시오.

표 17.10. VMware 구성 요소에 지원되는 최소 vSphere 버전

구성 요소	지원되는 최소 버전	설명
하이퍼바이저	HW 버전 13이 포함된 vSphere 6.5 이상	이 버전은 RHCOS(Red Hat Enterprise Linux CoreOS)가 지원하는 최소 버전입니다. Red Hat Enterprise Linux 8 지원 하이퍼바이저 목록을 참조하십시오.
In-tree 드라이버가 있는 스토리지	vSphere 6.5 이상	이 플러그인은 OpenShift Container Platform에 포함된 vSphere용 인트리 스토리지 드라이버를 사용하여 vSphere 스토리지를 생성합니다.
선택사항: 네트워킹(NSX-T)	vSphere 6.5U3 또는 vSphere 6.7U2 이상	OpenShift Container Platform에는 vSphere 6.5U3 또는 vSphere 6.7U2+가 필요합니다. NSX 및 OpenShift Container Platform의 호환성에 대한 자세한 내용은 VMware의 NSX 컨테이너 플러그인 설명서의 릴리스 노트 섹션을 참조하십시오.

vSphere 버전 6.5 인스턴스를 사용하는 경우 OpenShift Container Platform을 설치하기 전에 6.7U3 또는 7.0로 업그레이드하는 것이 좋습니다.

중요

OpenShift Container Platform을 설치하기 전에 ESXi 호스트의 시간이 동기화되었는지 확인해야 합니다. VMware 문서에서 Edit Time Configuration for a Host를 참조하십시오.

17.3.4. 네트워크 연결 요구사항

OpenShift Container Platform 클러스터 구성 요소가 통신할 수 있도록 시스템 간 네트워크 연결을 구성해야 합니다.

필요한 네트워크 포트에 대한 다음 세부 정보를 검토합니다.

표 17.11. 모든 시스템 간 통신에 사용되는 포트

프로토콜	포트	설명
ICMP	해당 없음	네트워크 연결성 테스트
TCP	**1936**	메트릭
	9000-9999	**9100-9101** 포트의 노드 내보내기 및 **9099** 포트의 Cluster Version Operator를 포함한 호스트 수준 서비스.
	10250-10259	Kubernetes에서 예약하는 기본 포트
	10256	openshift-sdn
UDP	**4789**	VXLAN(Virtual extensible LAN)
	6081	Geneve
	9000-9999	**9100-9101** 포트의 노드 내보내기를 포함한 호스트 수준 서비스.
	500	IPsec IKE 패킷
	4500	IPsec NAT-T 패킷
TCP/UDP	**30000-32767**	Kubernetes 노드 포트

프로토콜	포트	설명
ESP	해당 없음	IPsec Encapsulating Security Payload (ESP)

표 17.12. 모든 시스템과 컨트롤 플레인 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**6443**	Kubernetes API

표 17.13. 컨트롤 플레인 머신 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**2379-2380**	etcd 서버 및 피어 포트

추가 리소스

- **vSphere** 노드의 하드웨어 버전을 업데이트하려면 **vSphere**에서 실행 중인 노드에서 하드웨어 업데이트를 참조하십시오.

17.3.5. vCenter 요구사항

설치 관리자가 프로비저닝하는 인프라를 사용하는 **vCenter**에 **OpenShift Container Platform** 클러스터를 설치하기 전에 환경을 준비해야 합니다.

필요한 **vCenter** 계정 권한

vCenter에 **OpenShift Container Platform** 클러스터를 설치하려면 설치 프로그램에서 필요한 리소스를 읽고 생성할 수 있는 권한이 있는 계정에 액세스해야 합니다. 필요한 모든 권한에 액세스할 수 있는 가장 간단한 방법은 글로벌 관리 권한이 있는 계정을 사용하는 것입니다.

글로벌 관리 권한이 있는 계정을 사용할 수 없는 경우 **OpenShift Container Platform** 클러스터 설치에 필요한 권한을 부여하려면 역할을 생성해야 합니다. 대부분의 권한이 항상 필요하지만 기본 동작인 **vCenter** 인스턴스에 **OpenShift Container Platform** 클러스터를 포함하는 폴더를 프로비저닝할 설치 프로그램인 경우에만 필요합니다. 필요한 권한을 부여하려면 지정된 오브젝트에 대해 **vSphere** 역할을 생성하거나 수정해야 합니다.

설치 프로그램이 **vSphere** 가상 머신 폴더를 생성하는 경우 추가 역할이 필요합니다.

> **예 17.4. vSphere API에 설치에 필요한 역할 및 권한**

역할의 vSphere 개체	필요한 경우	vSphere API에서 필요한 권한
vSphere vCenter	Always	CNS .**Searchable** **InventoryService.Tagging.AttachTag** **InventoryService.Tagging.CreateCategory** **InventoryService.Tagging.CreateTag** **InventoryService.Tagging.Delete**Tag **InventoryService.tagged.DeleteTag** **InventoryService.Tagging.EditCategory** **InventoryService.Tagging.EditTag**ECDHE**Session** **ValidateSession**Forwarded**StorageProfile.Update**Forwarded**StorageProfile.View**
vSphere vCenter Cluster	Always	**Host.Config.Storage** **Resource.AssignVMToPool** **VApp.AssignResourcePool** **VApp.Import** **VirtualMachine.Config.AddNewDisk**
vSphere Datastore	Always	**Datastore.AllocateSpace** **Datastore.LoadBalancer** **Datastore.FileManagement** **InventoryService.Tagging.ObjectAttachable**
vSphere Port Group	Always	**Network.Assign**

역할의 vSphere 개체	필요한 경우	vSphere API에서 필요한 권한
가상 머신 폴더	Always	**InventoryService.ume.ObjectAttachable Resource.AssignVMToPool VApp.Import VirtualMachine.Config.Add ExistingDisk VirtualMachine.Config.Add NewDisk VirtualMachine.Config.Add RemoveDevice VirtualMachine.Config.Advanced Config VirtualMachine.Config.Anno tationECDHEVirtualMachine .Config.DiskExtend**Forwarde d VirtualMachine**.Config.**DiskExte nd Forwarded**VirtualMachine.Co nfig.DiskLeaseVirtualMachi ne.Config.ReCreate**VolumeCl aim.Config**.MemoryDriver.**

역할의 vSphere 개체	필요한 경우	vSphere API에서 필요한 권한
vSphere vCenter Datacenter	설치 프로그램이 가상 머신 폴더를 생성하는 경우	**InventoryService.Tagging.ObjectAttachable Resource.AssignVMToPool VApp.Import VirtualMachine.Config.Add ExistingDisk VirtualMachine.Config.Add NewDisk VirtualMachine.Config.Add RemoveDevice VirtualMachine.Config.Adva ncedConfig VirtualMachine.Config.Anno tationECDHEVirtualMachine .Config.DiskExtend**Forwarde d VirtualMachine**.**Config**.DiskExt end ForwardedVirtualMachine.C onfig.DiskLeaseVirtualMach ine.Config.ReCreate VolumeClaim.Config.Reset**defined**.**

예 17.5. vCenter 그래픽 사용자 인터페이스(GUI)에 설치하는 데 필요한 역할 및 권한

역할의 vSphere 개체	필요한 경우	vCenter GUI에서 필요한 권한
vSphere vCenter	Always	CNS **.Searchable "vSphere agged""Assign or unassign vSphere Tag" "vSphere tagging" "vSphere" "vSphere" "vSphere"CreatevSphere Tag" vSphere 태그""vSphere" 삭제 태그 지정" "vSphere Assignment""Edit vSphere Tag" "vSphere Tagging" "vSphere 태그" 세션" "Profile 기반 스토리지""Profile 기반 스토리지 업데이트""Profile 기반 스토리지"Profile 중심 스토리지 업데이트"Profile 기반 스토리지 업데이트"Profile 기반 스토리지 업데이트 스토리지 보기"**
vSphere vCenter Cluster	클러스터 루트에서 VM이 생성되는 경우	**Host.Configuration"Storage partition configuration" 리소스" 리소스 풀에 가상 머신을 서명" VApp"Assign resource pool" VApp.Import"Virtual machine""Change Configuration"."새 디스크 추가"**
vSphere vCenter 리소스 풀	기존 리소스 풀이 제공되는 경우	**Host.Configuration"Storage partition configuration" 리소스" 리소스 풀에 가상 머신을 서명" VApp"Assign resource pool" VApp.Import"Virtual machine""Change Configuration"."새 디스크 추가"**

역할의 vSphere 개체	필요한 경우	vCenter GUI에서 필요한 권한
vSphere Datastore	Always	**Datastore"Allocate space"** 데이터**저장소"** **데이터 저장소"** **Datastore"Low level file operations""vSphere Assignment"Assign or Unassign vSphere Tag on Object"**
vSphere Port Group	Always	**네트워크 "Assign network"**
가상 머신 폴더	Always	**"vSphere Assignment""Assign or unassign vSphere Tag on Object"** **Resource"Assign virtual machine to resource pool"**: **VApp.Import** **"Virtual machine""Change Configuration","Add existing disk""**Add existing disk**""Add new disk""Add new disk" "Virtual""** **머신 "시스템 변경", "가상 머신""** **"가상 머신""시스템 변경"** **"가상 머신""변경 구성",** **"가상 머신", 변경, CPU 수 변경"** **"가상 머신", 가상 디스크 변경", 가상 디스크 변경"** **"가상머신". 머신 변경 "구성 변경"** **"가상 머신""마이크 구성 변경"** **"가상 머신""마이크 구성 변경" "구성 변경",** **"가상 머신", 변경" 디스크 삭제"** **"가상 머신".변경 구성".Rename** **"가상 머신 변경" VIX API**로가상 머신 변경**"Gregradevirtual machine"Guestoperating system management by VIX API**

역할의 vSphere 개체	필요한 경우	vCenter GUI에서 필요한 권한
		machine". interaction" "Virtual machine". Interconnection"Power on" "Virtual machine". interactionion.Reset "Virtual machine"."Edit Inventory"."Create new" "Virtual machine"."Create existing" "Virtual machine"Edit Inventory""Create from existing" "Virtual machine"Edit Inventory""Create from existing" virtual machine".Edit Inventory""Create from existing" virtual machine".Edit Inventory""Create from existing""Virtual machine" machine".Provisioning."Clone 가상 머신" "가상 머신".Provisioning." "가상 머신".Provisioning."Deploy 템플릿"
vSphere vCenter Datacenter	설치 프로그램이 가상 머신 폴더를 생성하는 경우	"vSphere Assignment""Assign or unassign vSphere Tag on Object" Resource"Assign virtual machine to resource pool": VApp.Import "Virtual machine""Change Configuration","Add existing disk""Add existing disk""Add new disk""Add new disk" "Virtual"" 머신 "시스템 변경", "가상 머신"" "가상 머신""시스템 변경" "가상 머신""변경 구성", "가상 머신", 변경, CPU 수 변경" "가상 머신", 가상 디스크 변경", 가상 디스크 변경" "가상머신". 머신 변경 "구성 변경" "가상 머신""마이크 구성 변경" "가상 머신""마이크 구성 변경" "구성 변경", "가상 머신", 변경" 디스크 삭

역할의 vSphere 개체	필요한 경우	제" vCenter GUI에서 필요한 권한
		성".Rename "가상 머신 변경" VIX API로가상 머신 변 경"Gregradevirtual machine"Guestoperating system management by VIX API machine". interaction" "Virtual machine". Interconnection"Power on" "Virtual machine". interactionion.Reset "Virtual machine"."Edit Inventory"."Create new" "Virtual machine"."Create existing" "Virtual machine"Edit Inventory""Create from existing" "Virtual machine"Edit Inventory""Create from existing" virtual machine".Edit Inventory""Create from existing" virtual machine".Edit Inventory""Create from existing""Virtual machine" machine".Provisioning."Clone 가상 머신" "가상 머 신".Provisioning."Deploy 템플릿" "가상 머신".Provisioning" 폴더 만들기" 폴더 생성"폴더 삭제"

또한 사용자에게 일부 **ReadOnly** 권한이 필요하며 일부 역할에는 하위 오브젝트에 권한을 부여하는 사용 권한이 필요합니다. 이러한 설정은 클러스터를 기존 폴더에 설치할지 여부에 따라 달라집니다.

예 17.6. 필수 권한 및 권한 부여 설정

vSphere 오브젝트	폴더 유형	하위 항목으로 권한 부여	권한 필요
vSphere vCenter	Always	False	나열된 필수 권한
vSphere vCenter Datacenter	기존 폴더	False	**ReadOnly** 권한
	설치 프로그램은 폴더를 생성	True	나열된 필수 권한
vSphere vCenter Cluster	Always	True	나열된 필수 권한
vSphere vCenter Datastore	Always	False	나열된 필수 권한
vSphere Switch	Always	False	**ReadOnly** 권한
vSphere Port Group	Always	False	나열된 필수 권한
vSphere vCenter Virtual Machine Folder	기존 폴더	True	나열된 필수 권한

필요한 권한만으로 계정을 생성하는 방법에 대한 자세한 내용은 **vSphere** 문서에서 **vSphere 권한 및 사용자 관리 작업**을 참조하십시오.

vMotion으로 **OpenShift Container Platform** 사용

vSphere 환경에서 **vMotion**을 사용하려는 경우 **OpenShift Container Platform** 클러스터를 설치하기 전에 다음을 고려하십시오.

- **OpenShift Container Platform**은 일반적으로 **compute-only vMotion** 을 지원합니다. 스토리지 **vMotion**을 사용하면 문제가 발생할 수 있으며 지원되지 않습니다.

  컴퓨팅 및 컨트롤 플레인 노드의 가동 시간을 보장하기 위해 **vMotion**에 대한 **VMware** 모범 사례를 따르는 것이 좋습니다. 또한 **VMware** 유사성 방지 규칙을 사용하여 유지 관리 또는 하드웨어 문제 중에 **OpenShift Container Platform**의 가용성을 개선하는 것이 좋습니다.

  vMotion 및 유사성 방지 규칙에 대한 자세한 내용은 **vMotion 네트워킹 요구 사항** 및 **VM 유사성 방지 규칙에** 대한 **VMware vSphere** 설명서를 참조하십시오.

- **Pod**에서 **vSphere** 볼륨을 사용하는 경우 데이터 저장소를 통해 **VM**을 마이그레이션하면 수동으로 또는 스토리지 **vMotion**을 통해 **VM**을 마이그레이션하면 **OpenShift Container Platform**

PV(영구 볼륨) 오브젝트 내에서 잘못된 참조가 표시됩니다. 이러한 참조는 영향을 받는 포드가 시작되지 않게 하고 데이터 손실을 초래할 수 있습니다.

- 마찬가지로 OpenShift Container Platform은 데이터 저장소에서 VMDK의 선택적 마이그레이션을 지원하지 않으며 VM 프로비저닝 또는 PV의 동적 또는 정적 프로비저닝을 위해 데이터 저장소 클러스터의 일부 또는 PV의 동적 또는 정적 프로비저닝을 위해 데이터 저장소 클러스터의 일부인 데이터 저장소를 사용하는 경우도 있습니다.

클러스터 리소스

설치 관리자 프로비저닝 인프라를 사용하는 OpenShift Container Platform 클러스터를 배포할 때는 설치 프로그램이 vCenter 인스턴스에서 여러 리소스를 생성할 수 있어야 합니다.

표준 OpenShift Container Platform을 설치하면 다음 vCenter 리소스가 생성됩니다.

- 폴더 한 개
- 태그 카테고리 한 개
- 태그 한 개
- 가상 머신:
  - 템플릿 한 개
  - 임시 부트스트랩 노드 한 개
  - 컨트롤 플레인 노드 세 개
  - 컴퓨팅 시스템 세 개

이러한 리소스는 856GB의 스토리지를 사용하지만 부트스트랩 노드는 클러스터 설치 프로세스 중에 제거됩니다. 표준 클러스터를 사용하려면 최소 800GB의 스토리지가 필요합니다.

컴퓨팅 시스템을 더 많이 배포할수록 **OpenShift Container Platform** 클러스터는 더 많은 스토리지를 사용합니다.

클러스터 제한

사용 가능한 리소스는 클러스터마다 다릅니다. **vCenter** 내에서 가능한 클러스터 수는 주로 사용 가능한 스토리지 공간과 필요한 리소스 수에 대한 제한으로 제한됩니다. 클러스터가 생성하는 **vCenter** 리소스와 **IP** 주소 및 네트워크와 같이 클러스터를 배포하는 데 필요한 리소스에 대한 제한 사항을 모두 고려해야 합니다.

네트워킹 요구사항

네트워크에 **DHCP**를 사용하고 **DHCP** 서버가 클러스터 시스템에 영구 **IP** 주소를 제공하도록 구성되어 있는지 확인합니다. **DHCP** 서버를 사용하도록 기본 게이트웨이를 구성해야 합니다. 모든 노드는 동일한 **VLAN**에 있어야 합니다. 두 번째 **VLAN**을 **Day 2** 작업으로 사용하여 클러스터를 확장할 수 없습니다. 또한 **OpenShift Container Platform** 클러스터를 설치하기 전에 다음 네트워킹 리소스를 생성해야 합니다.

참고

클러스터의 각 **OpenShift Container Platform** 노드는 **DHCP**를 통해 검색할 수 있는 **NTP(Network Time Protocol)** 서버에 액세스할 수 있어야 합니다. **NTP** 서버 없이도 설치할 수 있습니다. 그러나 비동기 서버 클록으로 인해 **NTP** 서버가 차단하는 오류가 발생합니다.

필요한 IP 주소

설치 관리자 프로비저닝 **vSphere** 설치에는 다음 두 개의 고정 **IP** 주소가 필요합니다.

- **API** 주소는 클러스터 **API**에 액세스하는 데 사용됩니다.
- **Ingress** 주소는 클러스터 인그레스 트래픽에 사용됩니다.

OpenShift Container Platform 클러스터를 설치할 때 설치 프로그램에 이 **IP** 주소를 제공해야 합니다.

DNS 레코드

OpenShift Container Platform 클러스터를 호스팅하는 **vCenter** 인스턴스에 적절한 **DNS** 서버에서 고정 **IP** 주소 두 개의 **DNS** 레코드를 생성해야 합니다. 각 레코드에서 **<cluster_name>**은 클러스터 이름이고 **<base_domain>**은 클러스터를 설치할 때 지정하는 클러스터 기본 도메인입니다. 전체 **DNS** 레코드는 **<component>.<cluster_name>.<base_domain>** 형식입니다.

표 17.14. 필수 DNS 레코드

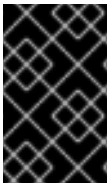
구성 요소	레코드	설명
API VIP	**api.<cluster_name>.<base_domain>.**	이 DNS A/AAAA 또는 CNAME 레코드는 컨트롤 플레인 시스템의 로드 밸런서를 가리켜야 합니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.
Ingress VIP	***.apps.<cluster_name>.<base_domain>.**	기본적으로 작업자 노드인 인그레스 라우터 Pod를 실행하는 시스템을 대상으로 하는 로드 밸런서를 가리키는 와일드카드 DNS A/AAAA 또는 CNAME 레코드입니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.

17.3.6. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 **SSH** 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 **Ignition** 구성 파일을 통해 **RHCOS(Red Hat Enterprise Linux CoreOS)** 노드에 전달되며 노드에 대한 **SSH** 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 **core** 사용자의 **~/.ssh/authorized_keys** 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 **core**로 **RHCOS** 노드에 **SSH**로 **SSH** 연결을 수행할 수 있습니다 . **SSH**를 통해 노드에 액세스하려면 로컬 사용자의 **SSH**에서 개인 키 **ID**를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 **SSH**를 실행하려면 설치 프로세스 중에 **SSH** 공용 키를 지정해야 합니다. **./openshift-install gather** 명령에도 **SSH** 공개 키가 클러스터 노드에 있어야 합니다.



중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

참고

AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

절차

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: ~/.ssh/id_ed25519)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 '~/.ssh 디렉터리에 있는지 확인하십시오.

참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 ed25519 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 rsa 또는 ecdsa 알고리즘을 사용하는 키를 생성합니다.

2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 ~/.ssh/id_ed25519.pub 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 ./openshift-install gather 명령을 사용하려는 경우 필요합니다.

참고

일부 배포에서는 ~/.ssh/id_rsa 및 ~/.ssh/id_dsa와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. ssh-agent 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. ssh-agent에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: ~/.ssh/id_ed25519).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

17.3.7. 설치 프로그램 받기

OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

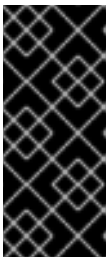
- **500MB**의 로컬 디스크 공간이 있는 **Linux** 또는 **macOS**를 실행하는 컴퓨터가 있습니다.

절차

1. **OpenShift Cluster Manager** 사이트의 인프라 공급자 페이지에 액세스합니다. **Red Hat** 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.

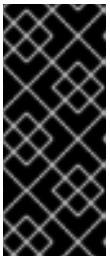
2. 인프라 공급자를 선택합니다.

3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.



중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.



중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 **OpenShift Container Platform** 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 **Linux** 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 **OpenShift Container Platform** 구성 요소에 대한 컨테이너 이미지를 제공하는 **Quay.io**를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

17.3.8. 시스템 신뢰에 vCenter 루트 CA 인증서 추가

설치 프로그램이 vCenter API에 액세스해야 하므로 OpenShift Container Platform 클러스터를 설치하기 전에 vCenter의 신뢰할 수 있는 루트 CA 인증서를 시스템 신뢰에 추가해야 합니다.

절차

1. vCenter 홈 페이지에서 vCenter의 루트 CA 인증서를 다운로드합니다. vSphere Web Services SDK 섹션에서 Download trusted root CA certificates를 클릭합니다. <vCenter>/certs/download.zip 파일이 다운로드됩니다.

2. vCenter 루트 CA 인증서가 포함된 압축 파일의 압축을 풉니다. 압축 파일의 내용은 다음 파일 구조와 유사합니다.

```
certs
├── lin
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
├── mac
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
└── win
    ├── 108f4d17.0.crt
    ├── 108f4d17.r1.crl
    ├── 7e757f6a.0.crt
    ├── 8e4f8471.0.crt
    └── 8e4f8471.r0.crl

3 directories, 15 files
```

3. 운영 체제 파일을 시스템 신뢰에 추가합니다. 예를 들어 Fedora 운영 체제에서는 다음 명령을 실행합니다.

```
# cp certs/lin/* /etc/pki/ca-trust/source/anchors
```

4. 시스템 신뢰를 업데이트합니다. 예를 들어 Fedora 운영 체제에서는 다음 명령을 실행합니다.

```
# update-ca-trust extract
```

17.3.9. 설치 구성 파일 만들기

VMware vSphere에 설치하는 OpenShift Container Platform 클러스터를 사용자 지정할 수 있습니다.

사전 요구 사항

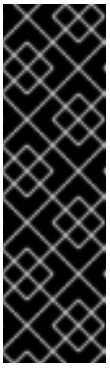
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.
- 서브스크립션 수준에서 서비스 주체 권한을 획득합니다.

절차

1. install-config.yaml 파일을 생성합니다.
   a. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

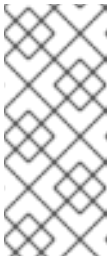
1 <installation_directory>는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.



중요

비어 있는 디렉터리를 지정합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

   b. 화면에 나타나는 지시에 따라 클라우드에 대한 구성 세부 사항을 입력합니다.
      i. 선택사항: 클러스터 시스템에 액세스하는 데 사용할 SSH 키를 선택합니다.



참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 **OpenShift Container Platform** 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 **SSH** 키를 지정합니다.

ii. 대상 플랫폼으로 **vsphere**를 선택합니다.

iii. **vCenter** 인스턴스의 이름을 지정합니다.

iv. 클러스터를 생성하는 데 필요한 권한이 있는 **vCenter** 계정의 사용자 이름과 암호를 지정합니다.

설치 프로그램이 **vCenter** 인스턴스에 연결됩니다.

v. 연결할 **vCenter** 인스턴스에서 데이터 센터를 선택합니다.

vi. 사용할 기본 **vCenter** 데이터 저장소를 선택합니다.

vii. **OpenShift Container Platform** 클러스터를 설치할 **vCenter** 클러스터를 선택합니다. 설치 프로그램은 **vSphere** 클러스터의 루트 리소스 풀을 기본 리소스 풀로 사용합니다.

viii. **vCenter** 인스턴스에서 구성한 가상 **IP** 주소 및 **DNS** 레코드가 포함된 네트워크를 선택합니다.

ix. 컨트롤 플레인 **API** 액세스를 위해 구성한 가상 **IP** 주소를 입력합니다.

x. 클러스터 인그레스용으로 구성한 가상 **IP** 주소를 입력합니다.

xi. 기본 도메인을 입력합니다. 이 기본 도메인은 구성한 **DNS** 레코드에서 사용한 것과 동일해야 합니다.

xii. 클러스터를 설명할 수 있는 이름을 입력합니다. 클러스터 이름은 구성한 **DNS** 레코드에서 사용한 것과 동일해야 합니다.

xiii. Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 붙여넣습니다.

2. **install-config.yaml** 파일을 수정합니다. 사용 가능한 매개변수에 대한 자세한 정보는 “설치 구성 매개변수” 섹션에서 확인할 수 있습니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.

중요

install-config.yaml 파일은 설치 과정에서 사용됩니다. 이 파일을 재사용하려면 지금 백업해야 합니다.

17.3.9.1. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 매개변수 값을 제공하여 클러스터를 호스팅할 클라우드 플랫폼에서 사용자 계정을 설명하고 선택사항으로 클러스터의 플랫폼을 사용자 지정합니다. **install-config.yaml** 설치 구성 파일을 생성할 때 명령줄을 통해 필요한 매개변수 값을 제공합니다. 클러스터를 사용자 지정하면 **install-config.yaml** 파일을 수정하여 플랫폼에 대한 세부 정보를 제공할 수 있습니다.

참고

설치한 후에는 **install-config.yaml** 파일에서 이러한 매개변수를 수정할 수 없습니다.

중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

17.3.9.1.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 17.15. 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	**dev** 와 같은 소문자 및 하이픈(**-**)의 문자열입니다.
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

매개변수	설명	값

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

17.3.9.1.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 17.16. 네트워크 매개변수

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체 **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.

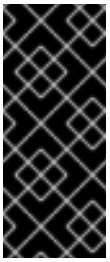
매개변수	설명	값
networking.clusterNetwork	Pod의 IP 주소 블록입니다. 기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된 **cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16** **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

17.3.9.1.3. 선택적 구성 매개변수

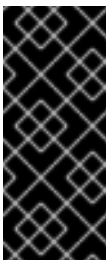
선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 17.17. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은**3**입니다.

매개변수	설명	값
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 기본값은 **External**입니다. 이 필드를 **Internal** 로 설정하는 것은 클라우드 이외의 플랫폼에서는 지원되지 않습니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

17.3.9.1.4. 추가 VMware vSphere 구성 매개변수

추가 VMware vSphere 구성 매개변수는 다음 표에 설명되어 있습니다.

표 17.18. 추가 VMware vSphere 클러스터 매개변수

매개변수	설명	값
platform.vsphere.vCenter	vCenter 서버의 정규화된 호스트 이름 또는 IP 주소입니다.	문자열
platform.vsphere.username	vCenter 인스턴스에 연결하는 데 사용할 사용자 이름입니다. 이 사용자에게는 최소한 vSphere에서 정적 또는 동적 영구 볼륨 프로비저닝 에 필요한 역할과 권한이 있어야 합니다.	문자열
platform.vsphere.password	vCenter 사용자 이름의 암호입니다.	문자열
platform.vsphere.datacenter	vCenter 인스턴스에서 사용할 데이터 센터의 이름입니다.	문자열
platform.vsphere.defaultDatastore	프로비저닝 볼륨에 사용할 기본 데이터 저장소의 이름입니다.	문자열
platform.vsphere.folder	*선택사항*입니다. 설치 프로그램이 가상 머신을 생성하는 기존 폴더의 절대 경로입니다. 이 값을 제공하지 않으면 설치 프로그램이 데이터 센터 가상 머신 폴더에 인프라 ID로 이름이 지정된 폴더를 생성합니다.	문자열(예: **/<datacenter_name>/vm/<folder_name>/<subfolder_name>**).

매개변수	설명	값
platform.vsphere.network	vCenter 인스턴스에서 구성한 가상 IP 주소 및 DNS 레코드가 포함된 네트워크입니다.	문자열
platform.vsphere.cluster	OpenShift Container Platform 클러스터를 설치할 vCenter 클러스터입니다.	문자열
platform.vsphere.apiVIP	컨트롤 플레인 API 액세스를 위해 구성한 가상 IP(VIP) 주소입니다.	IP 주소 (예: **128.0.0.1**)
platform.vsphere.ingressVIP	클러스터 인그레스용으로 구성한 가상 IP(VIP) 주소입니다.	IP 주소 (예: **128.0.0.1**)

17.3.9.1.5. 선택적 VMware vSphere 시스템 풀 구성 매개변수

다음 표에는 선택적 VMware vSphere 시스템 풀 구성 매개 변수가 설명되어 있습니다.

표 17.19. 선택적 VMware vSphere 시스템 풀 매개변수

매개변수	설명	값
platform.vsphere.clusterOSImage	설치 프로그램이 RHCOS 이미지를 다운로드하는 위치입니다. 네트워크가 제한된 환경에서 설치를 수행하려면이 매개변수를 설정해야 합니다.	HTTP 또는 HTTPS URL (선택 옵션으로 SHA-256 형식의 체크섬을 사용) 예: **https://mirror.openshift.com/images/rhcos-<version>-vmware.<architecture>.ova**.
platform.vsphere.osDisk.diskSizeGB	디스크 크기(GB)입니다.	정수
platform.vsphere.cpus	가상 머신을 할당할 총 가상 프로세서 코어 수입니다. **platform.vsphere.cpus** 의 값은 **platform.vsphere.coresPerSocket** 값의 여러 개여야 합니다.	정수
platform.vsphere.coresPerSocket	가상 머신의 소켓당 코어 수입니다. 가상 머신의 가상 소켓 수는 **platform.vsphere.cpus/platform.vsphere.coresPerSocket** 입니다. 컨트롤 플레인 노드 및 작업자 노드의 기본값은 각각 **4** 및 **2** 입니다.	정수
platform.vsphere.memoryMB	가상 머신의 메모리 크기(MB)입니다.	정수

17.3.9.2. 설치 관리자 프로비저닝 VMware vSphere 클러스터의 샘플 install-config.yaml 파일

install-config.yaml 파일을 사용자 지정하여 OpenShift Container Platform 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.

```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 3
  platform:
    vsphere: (4)
      cpus: 2
      coresPerSocket: 2
      memoryMB: 8192
      osDisk:
        diskSizeGB: 120
controlPlane: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3
  platform:
    vsphere: (7)
      cpus: 4
      coresPerSocket: 2
      memoryMB: 16384
      osDisk:
        diskSizeGB: 120
metadata:
  name: cluster (8)
platform:
  vsphere:
    vcenter: your.vcenter.server
    username: username
    password: password
    datacenter: datacenter
    defaultDatastore: datastore
    folder: folder
    network: VM_Network
    cluster: vsphere_cluster_name (9)
    apiVIP: api_vip
    ingressVIP: ingress_vip
fips: false
pullSecret: '{"auths": ...}'
sshKey: 'ssh-ed25519 AAAA...'
```

(1) 클러스터의 기본 도메인입니다. 모든 DNS 레코드는 이 기본 도메인의 하위 도메인이어야 하며 클러스터 이름을 포함해야 합니다.

2 5

controlPlane 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

3 6

동시 멀티스레딩 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. 일부 클러스터 시스템에서 동시 멀티스레딩을 비활성화할 경우에는 해당 멀티스레딩을 모든 클러스터 시스템에서 비활성화해야 합니다.

중요

동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다. 동시 멀티스레딩을 비활성화하는 경우 시스템은 8개 이상의 **CPU**와 **32GB**의 **RAM**을 사용해야 합니다.

4 7

선택사항: 컴퓨팅 및 컨트롤 플레인 시스템의 시스템 풀 매개변수에 대한 추가 구성을 제공하십시오.

8

DNS 레코드에 지정한 클러스터 이름입니다.

9

OpenShift Container Platform 클러스터를 설치할 **vSphere** 클러스터입니다. 설치 프로그램은 **vSphere** 클러스터의 루트 리소스 풀을 기본 리소스 풀로 사용합니다.

17.3.9.3. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 **HTTP** 또는 **HTTPS** 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 **OpenShift Container Platform** 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.

- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 API에 대한 호출을 포함하여 모든 클러스터 발신(Egress) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 Proxy 오브젝트의 spec.noProxy 필드에 사이트를 추가했습니다.

참고

Proxy 오브젝트의 status.noProxy 필드는 설치 구성에 있는 networking.machineNetwork[].cidr, networking.clusterNetwork[].cidr, networking.serviceNetwork[] 필드의 값으로 채워집니다.

Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 Red Hat OpenStack Platform (RHOSP)에 설치하는 경우 Proxy 오브젝트 status.noProxy 필드도 인스턴스 메타데이터 끝점(169.254.169.254)로 채워집니다.

절차

1. install-config.yaml 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

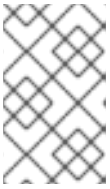
1. 클러스터 외부에서 HTTP 연결을 구축하는 데 사용할 프록시 URL입니다. URL 스키마는 http여야 합니다.

2. 클러스터 외부에서 HTTPS 연결을 구축하는 데 사용할 프록시 URL입니다.

3.

4

이 값을 제공하면 설치 프로그램에서 추가 CA 인증서를 보유할 openshift -config 네임스페이스에 user-ca- bundle 이라는 구성 맵을 생성합니다. additionalTrustBundle 및 하나 이상의 프록시 설정을 제공하는 경우 프록시 오브젝트는 trustedCA 필드의 user-ca-bundle 구성 맵을 참조하도록 구성됩니다. 그러면 Cluster Network Operator에서 trustedCA 매개변수에 대해 지정된 콘텐츠를 RHCOS 신뢰 번들과 병합하는 trusted-ca-bundle 구성 맵을 생성합니다. 프록시의 ID 인증서를 RHCOS 트러스트 번들에 있는 기관에서 서명하지 않은 경우 additionalTrustBundle 필드가 있어야 합니다.



참고

설치 프로그램에서 프록시 adinessEndpoints 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 OpenShift Container Platform을 설치할 때 참조하십시오.

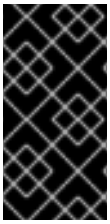
제공되는 install-config.yaml 파일의 프록시 설정을 사용하는 cluster라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 cluster Proxy 오브젝트는 계속 생성되지만 spec은 nil이 됩니다.

참고

cluster라는 Proxy 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

17.3.10. 클러스터 배포

호환되는 클라우드 플랫폼에 OpenShift Container Platform을 설치할 수 있습니다.



중요

최초 설치 과정에서 설치 프로그램의 create cluster 명령을 한 번만 실행할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

절차

1. 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터 배포를 초기화합니다.

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
  --log-level=info 2
```

1 <installation_directory> 값으로 사용자 지정한 ./install-config.yaml 파일의 위치를 지정합니다.

2 다른 설치 세부 사항을 보려면 info 대신 warn, debug 또는 error를 지정합니다.

클러스터 배포가 완료되면 웹 콘솔로 연결되는 링크와 kubeadmin 사용자의 인증 정보가 포함된 클러스터 액세스 지침이 사용자 터미널에 표시됩니다.

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-Wt5AL"
INFO Time elapsed: 36m22s
```

+

참고

설치에 성공하면 클러스터 액세스 및 인증 정보도 <installation_directory>/.openshift_install.log로 출력됩니다.

+

중요

- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 node-bootstrapper 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.

- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

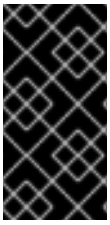
+

중요

설치 프로그램에서 생성되는 파일이나 설치 프로그램을 삭제해서는 안 됩니다. 클러스터를 삭제하려면 두 가지가 모두 필요합니다.

17.3.11. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift CLI(oc)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 oc를 설치할 수 있습니다.



중요

이전 버전의 oc를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 oc를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 Linux에서 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

절차

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.

2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.

3. **OpenShift v4.9 Linux Client** 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.

4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. **oc** 바이너리를 **PATH**에 있는 디렉터리에 배치합니다.

   PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 **OpenSfhit CLI** 설치

다음 절차에 따라 **Windows**에 **OpenShift CLI(oc)** 바이너리를 설치할 수 있습니다.

절차

1. **Red Hat** 고객 포털에서 **OpenShift Container Platform** 다운로드 페이지로 이동합니다.

2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.

3. **OpenShift v4.9 Windows Client** 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.

4. **ZIP** 프로그램으로 아카이브의 압축을 풉니다.

5. oc 바이너리를 PATH에 있는 디렉터리로 이동합니다.

   PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치

다음 절차에 따라 macOS에서 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

절차

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.
2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. OpenShift v4.9 MacOSX Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 해제하고 압축을 풉니다.
5. oc 바이너리 PATH의 디렉터리로 이동합니다.

   PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
$ oc <command>
```

17.3.12. CLI를 사용하여 클러스터에 로그인

클러스터 kubeconfig 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. kubeconfig 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- oc CLI를 설치했습니다.

절차

1. kubeadmin 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 oc 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

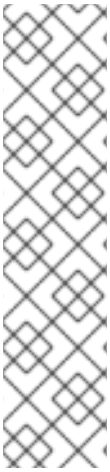
17.3.13. 레지스트리 스토리지 생성

클러스터를 설치한 후 레지스트리 Operator를 위한 스토리지를 생성해야 합니다.

17.3.13.1. 설치 중 제거된 이미지 레지스트리

공유 가능한 개체 스토리지를 제공하지 않는 플랫폼에서 OpenShift Image Registry Operator는 자체적으로 Removed로 부트스트랩합니다. 이를 통해 openshift-installer가 이러한 플랫폼 유형에서 설치를 완료할 수 있습니다.

설치 후 managementState를 Removed에서 Managed로 전환하도록 Image Registry Operator 구성을 편집해야 합니다.



참고

Prometheus 콘솔은 ImageRegistryRemoved 경고를 제공합니다. 다음은 예시 경고입니다.

"이미지 레지스트리가 제거되었습니다. ImageStreamTags를 참조하는 ImageStreamTags, BuildConfigs 및 DeploymentConfigs가 예상대로 작동하지 않을 수 있습니다. 스토리지를 구성하고 configs.imageregistry.operator.openshift.io를 편집하여 구성을 Managed 상태로 업데이트하십시오."

17.3.13.2. 이미지 레지스트리 스토리지 구성

기본 스토리지를 제공하지 않는 플랫폼에서는 처음에 Image Registry Operator를 사용할 수 없습니다. 설치한 후에 스토리지를 사용하도록 레지스트리를 구성하여 Registry Operator를 사용 가능하도록 만들어야 합니다.

프로덕션 클러스터에 필요한 영구 볼륨을 구성하는 과정의 지침이 표시됩니다. 해당하는 경우, 프로덕션 환경 외 클러스터에서만 사용할 수 있는 저장 위치로서 빈 디렉터리를 구성하는 과정의 지침이 표시됩니다.

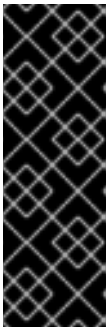
업그레이드 중에 Recreate 롤아웃 전략을 사용하여 이미지 레지스트리의 블록 스토리지 유형 사용을 허용하기 위한 추가 지침이 제공됩니다.

17.3.13.2.1. VMware vSphere용 레지스트리 스토리지 구성

클러스터 관리자는 설치한 후 스토리지를 사용하도록 레지스트리를 구성해야 합니다.

사전 요구 사항

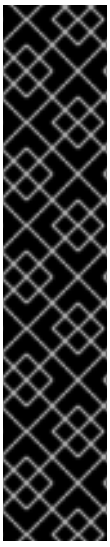
- 클러스터 관리자 권한이 있어야 합니다.
- VMware vSphere에 클러스터가 있어야 합니다.
- Red Hat OpenShift Container Storage와 같이 클러스터용 영구 스토리지 프로비저닝.



중요

OpenShift Container Platform은 복제본이 하나만 있는 경우 이미지 레지스트리 스토리지에 대한 ReadWriteOnce 액세스를 지원합니다. ReadWriteOnce 액세스에서는 레지스트리가 Recreate 롤아웃 전략을 사용해야 합니다. 두 개 이상의 복제본으로 고 가용성을 지원하는 이미지 레지스트리를 배포하려면 ReadWriteMany 액세스가 필요합니다.

- "100Gi" 용량이 필요합니다.



중요

테스트 결과, RHEL의 NFS 서버를 핵심 서비스용 스토리지 백엔드로 사용하는 데 문제가 있는 것으로 나타납니다. 여기에는 OpenShift Container Registry and Quay, 스토리지 모니터링을 위한 Prometheus, 로깅 스토리지를 위한 Elasticsearch가 포함됩니다. 따라서 RHEL NFS를 사용하여 핵심 서비스에서 사용하는 PV를 백업하는 것은 권장되지 않습니다.

마켓플레이스의 다른 NFS 구현에는 이러한 문제가 나타나지 않을 수 있습니다. 이러한 OpenShift Container Platform 핵심 구성 요소에 대해 완료된 테스트에 대한 자세한 내용은 개별 NFS 구현 공급업체에 문의하십시오.

프로세스

1. 스토리지를 사용하도록 레지스트리를 구성하기 위해 configs.imageregistry/cluster 리소스에서 spec.storage.pvc를 변경합니다.

참고

공유 스토리지를 사용할 때 보안 설정을 확인하여 외부에서의 액세스를 방지합니다.

2. 레지스트리 pod가 없는지 확인합니다.

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

출력 예

```
No resourses found in openshift-image-registry namespace
```

참고

출력에 레지스트리 Pod가 있는 경우 이 절차를 계속할 필요가 없습니다.

3. 레지스트리 구성을 확인합니다.

```
$ oc edit configs.imageregistry.operator.openshift.io
```

출력 예

```
storage:
  pvc:
    claim: 1
```

1

image-registry-storage 영구 볼륨 클레임 (PVC)을 자동으로 생성할 수 있도록 claim 필드를 비워 둡니다. PVC는 기본 스토리지 클래스를 기반으로 생성됩니다. 그러나 기본 스토리지 클래스에서 RADOS 블록 장치(RBD)와 같은

ReadWriteOnce(ReadWriteOnce) 볼륨을 제공할 수 있으므로 복제본이 두 개 이상될 때 문제가 발생할 수 있습니다.

4. **clusteroperator** 상태를 확인합니다.

```
$ oc get clusteroperator image-registry
```

출력 예

```
NAME              VERSION                             AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                                 True        False         False     6h50m
```

17.3.13.2.2. VMware vSphere용 블록 레지스트리 스토리지 구성

클러스터 관리자로서 업그레이드 중에 이미지 레지스트리가 **vSphere VMDK(Virtual Machine Disk)**와 같은 블록 스토리지 유형을 사용할 수 있도록 허용하기 위해 **Recreate** 롤아웃 전략을 사용할 수 있습니다.

중요

블록 스토리지 볼륨이 지원되지만 프로덕션 클러스터에서 이미지 레지스트리와 함께 사용하는 것은 권장되지 않습니다. 레지스트리가 블록 스토리지에 구성된 설치는 레지스트리가 둘 이상의 복제본을 가질 수 없기 때문에 가용성이 높지 않습니다.

프로세스

1. 이미지 레지스트리 스토리지를 블록 스토리지 유형으로 설정하려면 **Recreate** 롤아웃 전략을 사용하고 복제본 **1**개 만으로 실행되도록 레지스트리를 패치합니다.

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. 블록 스토리지 장치에 **PV**를 프로비저닝하고 해당 볼륨의 **PVC**를 생성합니다. 요청된 블록 볼륨은 **RWO(ReadWriteOnce)** 액세스 모드를 사용합니다.

a. VMware vSphere PersistentVolumeClaim 개체를 정의하려면 다음 내용이 포함된 pvc.yaml 파일을 생성합니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage 1
  namespace: openshift-image-registry 2
spec:
  accessModes:
  - ReadWriteOnce 3
  resources:
    requests:
      storage: 100Gi 4
```

1 PersistentVolumeClaim 개체를 표시하는 고유한 이름입니다.

2 PersistentVolumeClaim 오브젝트의 네임스페이스로 openshift-image-registry입니다.

3 영구 볼륨 클레임의 액세스 모드입니다. ReadWriteOnce를 사용하면 단일 노드에서 읽기 및 쓰기 권한으로 볼륨을 마운트할 수 있습니다.

4 영구 볼륨 클레임의 크기입니다.

b. 파일에서 PersistentVolumeClaim 오브젝트를 만듭니다.

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 올바른 PVC를 참조하도록 레지스트리 설정을 편집합니다.

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

출력 예

```
storage:
  pvc:
    claim: 1
```

1

사용자 지정 PVC를 만들면 image-registry-storage PVC의 기본 자동 생성을 위해 claim 필드를 비워둘 수 있습니다.

올바른 PVC를 참조하도록 레지스트리 스토리지를 구성하는 방법은 vSphere 용 레지스트리 구성을 참조하십시오.

17.3.14. VMware vSphere 볼륨 백업

OpenShift Container Platform은 새 볼륨을 독립 영구 디스크로 프로비저닝하여 클러스터의 모든 노드에서 볼륨을 자유롭게 연결 및 분리합니다. 결과적으로 스냅샷을 사용하는 볼륨을 백업하거나 스냅샷에서 볼륨을 복원할 수 없습니다. 자세한 내용은 스냅 샷 제한 을 참조하십시오.

프로세스

영구 볼륨의 백업을 생성하려면 다음을 수행합니다.

1. 영구 볼륨을 사용중인 애플리케이션을 중지합니다.
2. 영구 볼륨을 복제합니다.
3. 애플리케이션을 다시 시작합니다.
4. 복제된 볼륨의 백업을 만듭니다.
5. 복제된 볼륨을 삭제합니다.

17.3.15. 스틸 클럭 계정

기본적으로 설치 프로그램은 스틸 클럭 회계 매개 변수 (**stealclock.enabled**)를 활성화하지 않고 클러스터의 가상 머신을 프로비저닝합니다. 스틸링 클럭 계정을 사용하면 클러스터 문제 해결에 도움이 될 수 있습니다. 클러스터가 배포된 후 **vSphere Client**를 사용하여 각 가상 머신에서 이 매개변수를 활성화합니다.

자세한 내용은 이 **Red Hat** 기술 자료 문서 를 참조하십시오.

17.3.16. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 **Telemetry** 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 **Telemetry**가 자동으로 실행되고 **OpenShift Cluster Manager** 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 **OpenShift Cluster Manager**를 사용하여 자동으로 또는 **OpenShift Cluster Manager**를 사용하여 수동으로 유지 관리되는지 확인한 후 **subscription watch**를 사용하여 계정 또는 다중 클러스터 수준에서 **OpenShift Container Platform** 서브스크립션을 추적합니다.

추가 리소스

- **Telemetry** 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

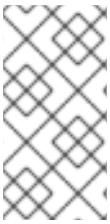
17.3.17. 다음 단계

- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.
- 레지스트리를 설정하고 레지스트리 스토리지를 구성합니다.
- 선택 사항: **vSphere Problem Detector Operator**에서 이벤트를 보고 클러스터에 권한 또는 스토리지 구성 문제가 있는지 확인합니다.

17.4. 네트워크 사용자 지정으로 VSPHERE에 클러스터 설치

OpenShift Container Platform 4.9 버전에서는 사용자 지정 네트워크 구성 옵션이 있는 설치 관리자 프로비저닝 인프라를 사용하여 VMware vSphere 인스턴스에 클러스터를 설치할 수 있습니다. 네트워크 구성을 사용자 지정할 경우, 클러스터가 사용자 환경의 기존 IP 주소 할당과 공존하고 기존 MTU 및 VXLAN 구성과 통합될 수 있습니다. 설치를 사용자 지정하려면 클러스터를 설치하기 전에 install-config.yaml 파일에서 매개변수를 수정합니다.

설치 과정에서 네트워크 구성 매개변수를 대부분 설정해야 하며, 실행 중인 클러스터에서는 kubeProxy 구성 매개변수만 수정할 수 있습니다.

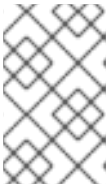


참고

OpenShift Container Platform은 단일 VMware vCenter에만 클러스터 배포를 지원합니다. 여러 vCenter에 머신/머신 세트가 있는 클러스터 배포는 지원되지 않습니다.

17.4.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.
- 클러스터용 영구 스토리지를 프로비저닝합니다. 프라이빗 이미지 레지스트리를 배포하려면 스토리지에서 ReadWriteMany 액세스 모드를 제공해야 합니다.
- OpenShift Container Platform 설치 프로그램은 vCenter 및 ESXi 호스트의 포트 443에 액세스해야 합니다. 포트 443에 액세스할 수 있는지 확인했습니다.
- 방화벽을 사용하는 경우 관리자가 포트 443에 액세스할 수 있는지 확인합니다. 설치에 성공하려면 포트 443에서 컨트롤 플레인 노드가 vCenter 및 ESXi 호스트에 연결할 수 있어야 합니다.
- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성해야 합니다.



참고

프록시를 구성하는 경우에도 해당 사이트 목록을 검토하십시오.

17.4.2. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.

중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

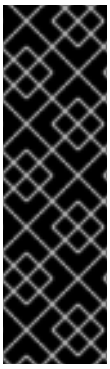
17.4.3. VMware vSphere 인프라 요구사항

사용하는 구성 요소의 요구사항을 충족하는 VMware vSphere 버전 6 또는 7 인스턴스에 OpenShift Container Platform 클러스터를 설치해야 합니다.

표 17.20. vSphere 가상 환경의 버전 요구 사항

가상 환경 제품	필요한 버전
VM 하드웨어 버전	13 이상
vSphere ESXi hosts	6.5 이상

가상 환경 제품	필요한 버전
vCenter 호스트	6.5 이상



중요

이제 VMware vSphere 버전 6.7U2 또는 이전 버전에 클러스터를 설치하는 것이 더 이상 사용되지 않으며 가상 하드웨어 버전 13은 더 이상 사용되지 않습니다. 이러한 버전은 여전히 완전히 지원되지만 향후 OpenShift Container Platform 버전에서는 지원이 제거됩니다. 이제 OpenShift Container Platform의 vSphere 가상 머신의 하드웨어 버전 15가 기본값이 되었습니다. vSphere 노드의 하드웨어 버전을 업데이트하려면 " vSphere에서 실행되는 노드에서 하드웨어 업데이트" 문서를 참조하십시오.

표 17.21. VMware 구성 요소에 지원되는 최소 vSphere 버전

구성 요소	지원되는 최소 버전	설명
하이퍼바이저	HW 버전 13이 포함된 vSphere 6.5 이상	이 버전은 RHCOS(Red Hat Enterprise Linux CoreOS)가 지원하는 최소 버전입니다. Red Hat Enterprise Linux 8 지원 하이퍼바이저 목록을 참조하십시오.
In-tree 드라이버가 있는 스토리지	vSphere 6.5 이상	이 플러그인은 OpenShift Container Platform에 포함된 vSphere용 인트리 스토리지 드라이버를 사용하여 vSphere 스토리지를 생성합니다.
선택사항: 네트워킹(NSX-T)	vSphere 6.5U3 또는 vSphere 6.7U2 이상	OpenShift Container Platform에는 vSphere 6.5U3 또는 vSphere 6.7U2+가 필요합니다. NSX 및 OpenShift Container Platform의 호환성에 대한 자세한 내용은 VMware의 NSX 컨테이너 플러그인 설명서의 릴리스 노트 섹션을 참조하십시오.

vSphere 버전 6.5 인스턴스를 사용하는 경우 OpenShift Container Platform을 설치하기 전에 6.7U3 또는 7.0로 업그레이드하는 것이 좋습니다.

중요

OpenShift Container Platform을 설치하기 전에 **ESXi** 호스트의 시간이 동기화되었는지 확인해야 합니다. **VMware** 문서에서 **Edit Time Configuration for a Host**를 참조하십시오.

17.4.4. 네트워크 연결 요구사항

OpenShift Container Platform 클러스터 구성 요소가 통신할 수 있도록 시스템 간 네트워크 연결을 구성해야 합니다.

필요한 네트워크 포트에 대한 다음 세부 정보를 검토합니다.

표 17.22. 모든 시스템 간 통신에 사용되는 포트

프로토콜	포트	설명
ICMP	해당 없음	네트워크 연결성 테스트
TCP	**1936**	메트릭
	9000-9999	**9100-9101** 포트의 노드 내보내기 및 **9099** 포트의 Cluster Version Operator를 포함한 호스트 수준 서비스.
	10250-10259	Kubernetes에서 예약하는 기본 포트
	10256	openshift-sdn
UDP	**4789**	VXLAN(Virtual extensible LAN)
	6081	Geneve
	9000-9999	**9100-9101** 포트의 노드 내보내기를 포함한 호스트 수준 서비스.
	500	IPsec IKE 패킷
	4500	IPsec NAT-T 패킷
TCP/UDP	**30000-32767**	Kubernetes 노드 포트
ESP	해당 없음	IPsec Encapsulating Security Payload (ESP)

표 17.23. 모든 시스템과 컨트롤 플레인 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**6443**	Kubernetes API

표 17.24. 컨트롤 플레인 머신 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**2379-2380**	etcd 서버 및 피어 포트

추가 리소스

- **vSphere** 노드의 하드웨어 버전을 업데이트하려면 **vSphere**에서 실행 중인 노드에서 하드웨어 업데이트를 참조하십시오.

17.4.5. vCenter 요구사항

설치 관리자가 프로비저닝하는 인프라를 사용하는 **vCenter**에 **OpenShift Container Platform** 클러스터를 설치하기 전에 환경을 준비해야 합니다.

필요한 **vCenter** 계정 권한

vCenter에 **OpenShift Container Platform** 클러스터를 설치하려면 설치 프로그램에서 필요한 리소스를 읽고 생성할 수 있는 권한이 있는 계정에 액세스해야 합니다. 필요한 모든 권한에 액세스할 수 있는 가장 간단한 방법은 글로벌 관리 권한이 있는 계정을 사용하는 것입니다.

글로벌 관리 권한이 있는 계정을 사용할 수 없는 경우 **OpenShift Container Platform** 클러스터 설치에 필요한 권한을 부여하려면 역할을 생성해야 합니다. 대부분의 권한이 항상 필요하지만 기본 동작인 **vCenter** 인스턴스에 **OpenShift Container Platform** 클러스터를 포함하는 폴더를 프로비저닝할 설치 프로그램인 경우에만 필요합니다. 필요한 권한을 부여하려면 지정된 오브젝트에 대해 **vSphere** 역할을 생성하거나 수정해야 합니다.

설치 프로그램이 **vSphere** 가상 머신 폴더를 생성하는 경우 추가 역할이 필요합니다.

예 17.7. vSphere API에 설치에 필요한 역할 및 권한

역할의 vSphere 개체	필요한 경우	vSphere API에서 필요한 권한

역할의 vSphere 개체	필요한 경우	vSphere API에서 필요한 권한
vSphere vCenter	Always	CNS .**Searchable** **InventoryService.Tagging.AttachTag** **InventoryService.Tagging.CreateCategory** **InventoryService.Tagging.CreateTag** **InventoryService.Tagging.Delete**Tag **InventoryService.tagged.DeleteTag** **InventoryService.Tagging.EditCategory** **InventoryService.Tagging.EditTag**ECDHE**Session** **ValidateSession**Forwarded**StorageProfile.Update**Forwarded**StorageProfile.View**
vSphere vCenter Cluster	Always	**Host.Config.Storage** **Resource.AssignVMToPool** **VApp.AssignResourcePool** **VApp.Import** **VirtualMachine.Config.AddNewDisk**
vSphere Datastore	Always	**Datastore.AllocateSpace** **Datastore.LoadBalancer** **Datastore.FileManagement** **InventoryService.Tagging.ObjectAttachable**
vSphere Port Group	Always	**Network.Assign**

역할의 vSphere 개체	필요한 경우	vSphere API에서 필요한 권한
가상 머신 폴더	Always	**InventoryService.ume.ObjectAttachable Resource.AssignVMToPool VApp.Import VirtualMachine.Config.Add ExistingDisk VirtualMachine.Config.Add NewDisk VirtualMachine.Config.Add RemoveDevice VirtualMachine.Config.Advanced Config VirtualMachine.Config.Annotation ECDHEVirtualMachine .Config.DiskExtend** Forwarded VirtualMachine**.Config.** DiskExtend Forwarded**VirtualMachine.Config.DiskLease VirtualMachine.Config.ReCreate** VolumeClaim.Config**.MemoryDriver.**

역할의 vSphere 개체	필요한 경우	vSphere API에서 필요한 권한
vSphere vCenter Datacenter	설치 프로그램이 가상 머신 폴더를 생성하는 경우	**InventoryService.Tagging.ObjectAttachable Resource.AssignVMToPool VApp.Import VirtualMachine.Config.Add ExistingDisk VirtualMachine.Config.Add NewDisk VirtualMachine.Config.Add RemoveDevice VirtualMachine.Config.Adva ncedConfig VirtualMachine.Config.Anno tationECDHEVirtualMachine .Config.DiskExtend**Forwarde d VirtualMachine**.**Config**.DiskExt end ForwardedVirtualMachine.C onfig.DiskLeaseVirtualMach ine.Config.ReCreate VolumeClaim.Config.Reset**defined**.**

예 17.8. vCenter 그래픽 사용자 인터페이스(GUI)에 설치하는 데 필요한 역할 및 권한

역할의 vSphere 개체	필요한 경우	vCenter GUI에서 필요한 권한
vSphere vCenter	Always	CNS **.Searchable "vSphere agged""Assign or unassign vSphere Tag" "vSphere tagging" "vSphere" "vSphere" "vSphere"CreatevSphere Tag" vSphere 태그""vSphere" 삭제 태그 지정" "vSphere Assignment""Edit vSphere Tag" "vSphere Tagging" "vSphere 태그" 세션" "Profile 기반 스토리지""Profile 기반 스토리지 업데이트""Profile 기반 스토리지"Profile 중심 스토리지 업데이트"Profile 기반 스토리지 업데이트"Profile 기반 스토리지 업데이트 스토리지 보기"**
vSphere vCenter Cluster	클러스터 루트에서 VM이 생성되는 경우	**Host.Configuration"Storage partition configuration" 리소스" 리소스 풀에 가상 머신을 서명" VApp"Assign resource pool" VApp.Import"Virtual machine""Change Configuration"."새 디스크 추가"**
vSphere vCenter 리소스 풀	기존 리소스 풀이 제공되는 경우	**Host.Configuration"Storage partition configuration" 리소스" 리소스 풀에 가상 머신을 서명" VApp"Assign resource pool" VApp.Import"Virtual machine""Change Configuration"."새 디스크 추가"**

역할의 vSphere 개체	필요한 경우	vCenter GUI에서 필요한 권한
vSphere Datastore	Always	**Datastore"Allocate space"** 데이터**저장소"** **데이터 저장소"** **Datastore"Low level file operations""vSphere Assignment"Assign or Unassign vSphere Tag on Object"**
vSphere Port Group	Always	**네트워크 "Assign network"**
가상 머신 폴더	Always	**"vSphere Assignment""Assign or unassign vSphere Tag on Object"** **Resource"Assign virtual machine to resource pool"**: **VApp.Import** **"Virtual machine""Change Configuration","Add existing disk""**Add existing disk**""Add new disk""Add new disk" "Virtual""** **머신 "시스템 변경", "가상 머신""** **"가상 머신""시스템 변경"** **"가상 머신""변경 구성",** **"가상 머신", 변경, CPU 수 변경"** **"가상 머신", 가상 디스크 변경", 가상 디스크 변경"** **"가상머신". 머신 변경 "구성 변경"** **"가상 머신""마이크 구성 변경"** **"가상 머신""마이크 구성 변경" "구성 변경",** **"가상 머신", 변경" 디스크 삭제"** **"가상 머신".변경 구성".Rename** **"가상 머신 변경" VIX API**로가상 머신 변경**"Gregradevirtual machine"Guestoperating system management by VIX API**

역할의 vSphere 개체	필요한 경우	vCenter GUI에서 필요한 권한
		machine". interaction" "Virtual machine". Interconnection"Power on" "Virtual machine". interactionion.Reset "Virtual machine"."Edit Inventory"."Create new" "Virtual machine"."Create existing" "Virtual machine"Edit Inventory""Create from existing" "Virtual machine"Edit Inventory""Create from existing" virtual machine".Edit Inventory""Create from existing" virtual machine".Edit Inventory""Create from existing""Virtual machine" machine".Provisioning."Clone 가상 머신" "가상 머신".Provisioning." "가상 머신".Provisioning."Deploy 템플릿"
vSphere vCenter Datacenter	설치 프로그램이 가상 머신 폴더를 생성하는 경우	"vSphere Assignment""Assign or unassign vSphere Tag on Object" Resource"Assign virtual machine to resource pool": VApp.Import "Virtual machine""Change Configuration","Add existing disk""Add existing disk""Add new disk""Add new disk" "Virtual"" 머신 "시스템 변경", "가상 머신"" "가상 머신""시스템 변경" "가상 머신""변경 구성", "가상 머신", 변경, CPU 수 변경" "가상 머신", 가상 디스크 변경", 가상 디스크 변경" "가상머신". 머신 변경 "구성 변경" "가상 머신""마이크 구성 변경" "가상 머신""마이크 구성 변경" "구성 변경",

역할의 vSphere 개체	필요한 경우	"가상 머신", 변경" 디스크 삭 vCenter GUI에서 필요한 권한
		"가상 머신".변경 구 성".Rename "가상 머신 변경" VIX API로가 상 머신 변 경"Gregradevirtual machine"Guestoperating system management by VIX API machine". interaction" "Virtual machine". Interconnection"Power on" "Virtual machine". interactionion.Reset "Virtual machine"."Edit Inventory"."Create new" "Virtual machine"."Create existing" "Virtual machine"Edit Inventory""Create from existing" "Virtual machine"Edit Inventory""Create from existing" virtual machine".Edit Inventory""Create from existing" virtual machine".Edit Inventory""Create from existing""Virtual machine" machine".Provisioning."Clo ne 가상 머신" "가상 머 신".Provisioning."Deploy 템 플릿" "가상 머신".Provisioning" 폴더 만들기" 폴더 생성"폴더 삭제"

또한 사용자에게 일부 **ReadOnly** 권한이 필요하며 일부 역할에는 하위 오브젝트에 권한을 부여하는 사용 권한이 필요합니다. 이러한 설정은 클러스터를 기존 폴더에 설치할지 여부에 따라 달라집니다.

예 17.9. 필수 권한 및 권한 부여 설정

vSphere 오브젝트	폴더 유형	하위 항목으로 권한 부여	권한 필요
vSphere vCenter	Always	False	나열된 필수 권한
vSphere vCenter Datacenter	기존 폴더	False	**ReadOnly** 권한
	설치 프로그램은 폴더를 생성	True	나열된 필수 권한
vSphere vCenter Cluster	Always	True	나열된 필수 권한
vSphere vCenter Datastore	Always	False	나열된 필수 권한
vSphere Switch	Always	False	**ReadOnly** 권한
vSphere Port Group	Always	False	나열된 필수 권한
vSphere vCenter Virtual Machine Folder	기존 폴더	True	나열된 필수 권한

필요한 권한만으로 계정을 생성하는 방법에 대한 자세한 내용은 **vSphere** 문서에서 **vSphere 권한 및 사용자 관리 작업**을 참조하십시오.

vMotion으로 OpenShift Container Platform 사용

vSphere 환경에서 **vMotion**을 사용하려는 경우 **OpenShift Container Platform** 클러스터를 설치하기 전에 다음을 고려하십시오.

- **OpenShift Container Platform**은 일반적으로 **compute-only vMotion** 을 지원합니다. 스토리지 **vMotion**을 사용하면 문제가 발생할 수 있으며 지원되지 않습니다.

  컴퓨팅 및 컨트롤 플레인 노드의 가동 시간을 보장하기 위해 **vMotion**에 대한 **VMware** 모범 사례를 따르는 것이 좋습니다. 또한 **VMware** 유사성 방지 규칙을 사용하여 유지 관리 또는 하드웨어 문제 중에 **OpenShift Container Platform**의 가용성을 개선하는 것이 좋습니다.

  vMotion 및 유사성 방지 규칙에 대한 자세한 내용은 **vMotion 네트워킹 요구 사항** 및 **VM 유사성 방지 규칙에** 대한 **VMware vSphere** 설명서를 참조하십시오.

- **Pod**에서 **vSphere** 볼륨을 사용하는 경우 데이터 저장소를 통해 **VM**을 마이그레이션하면 수

동으로 또는 스토리지 **vMotion**을 통해 **VM**을 마이그레이션하면 **OpenShift Container Platform PV(**영구 볼륨**)** 오브젝트 내에서 잘못된 참조가 표시됩니다**.** 이러한 참조는 영향을 받는 포드가 시작되지 않게 하고 데이터 손실을 초래할 수 있습니다**.**

- 마찬가지로 **OpenShift Container Platform**은 데이터 저장소에서 **VMDK**의 선택적 마이그레이션을 지원하지 않으며 **VM** 프로비저닝 또는 **PV**의 동적 또는 정적 프로비저닝을 위해 데이터 저장소 클러스터의 일부 또는 **PV**의 동적 또는 정적 프로비저닝을 위해 데이터 저장소 클러스터의 일부인 데이터 저장소를 사용하는 경우도 있습니다**.**

클러스터 리소스

설치 관리자 프로비저닝 인프라를 사용하는 **OpenShift Container Platform** 클러스터를 배포할 때는 설치 프로그램이 **vCenter** 인스턴스에서 여러 리소스를 생성할 수 있어야 합니다**.**

표준 **OpenShift Container Platform**을 설치하면 다음 **vCenter** 리소스가 생성됩니다**.**

- 폴더 한 개
- 태그 카테고리 한 개
- 태그 한 개
- 가상 머신**:**
  - 템플릿 한 개
  - 임시 부트스트랩 노드 한 개
  - 컨트롤 플레인 노드 세 개
  - 컴퓨팅 시스템 세 개

이러한 리소스는 **856GB**의 스토리지를 사용하지만 부트스트랩 노드는 클러스터 설치 프로세스 중에 제거됩니다**.** 표준 클러스터를 사용하려면 최소 **800GB**의 스토리지가 필요합니다**.**

컴퓨팅 시스템을 더 많이 배포할수록 **OpenShift Container Platform** 클러스터는 더 많은 스토리지를 사용합니다.

클러스터 제한

사용 가능한 리소스는 클러스터마다 다릅니다. **vCenter** 내에서 가능한 클러스터 수는 주로 사용 가능한 스토리지 공간과 필요한 리소스 수에 대한 제한으로 제한됩니다. 클러스터가 생성하는 **vCenter** 리소스와 **IP** 주소 및 네트워크와 같이 클러스터를 배포하는 데 필요한 리소스에 대한 제한 사항을 모두 고려해야 합니다.

네트워킹 요구사항

네트워크에 **DHCP**를 사용하고 **DHCP** 서버가 클러스터 시스템에 영구 **IP** 주소를 제공하도록 구성되어 있는지 확인합니다. **DHCP** 서버를 사용하도록 기본 게이트웨이를 구성해야 합니다. 모든 노드는 동일한 **VLAN**에 있어야 합니다. 두 번째 **VLAN**을 **Day 2** 작업으로 사용하여 클러스터를 확장할 수 없습니다. 또한 **OpenShift Container Platform** 클러스터를 설치하기 전에 다음 네트워킹 리소스를 생성해야 합니다.

참고

클러스터의 각 **OpenShift Container Platform** 노드는 **DHCP**를 통해 검색할 수 있는 **NTP(Network Time Protocol)** 서버에 액세스할 수 있어야 합니다. **NTP** 서버 없이도 설치할 수 있습니다. 그러나 비동기 서버 클록으로 인해 **NTP** 서버가 차단하는 오류가 발생합니다.

필요한 IP 주소

설치 관리자 프로비저닝 **vSphere** 설치에는 다음 두 개의 고정 **IP** 주소가 필요합니다.

- **API** 주소는 클러스터 **API**에 액세스하는 데 사용됩니다.
- **Ingress** 주소는 클러스터 인그레스 트래픽에 사용됩니다.

OpenShift Container Platform 클러스터를 설치할 때 설치 프로그램에 이 **IP** 주소를 제공해야 합니다.

DNS 레코드

OpenShift Container Platform 클러스터를 호스팅하는 **vCenter** 인스턴스에 적절한 **DNS** 서버에서 고정 **IP** 주소 두 개의 **DNS** 레코드를 생성해야 합니다. 각 레코드에서 **<cluster_name>**은 클러스터 이름

이고 <base_domain>은 클러스터를 설치할 때 지정하는 클러스터 기본 도메인입니다. 전체 DNS 레코드는 <component>.<cluster_name>.<base_domain> 형식입니다.

표 17.25. 필수 DNS 레코드

구성 요소	레코드	설명
API VIP	api.<cluster_name>.<base_domain>.	이 DNS A/AAAA 또는 CNAME 레코드는 컨트롤 플레인 시스템의 로드 밸런서를 가리켜야 합니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.
Ingress VIP	*.apps.<cluster_name>.<base_domain>.	기본적으로 작업자 노드인 인그레스 라우터 Pod를 실행하는 시스템을 대상으로 하는 로드 밸런서를 가리키는 와일드카드 DNS A/AAAA 또는 CNAME 레코드입니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.

17.4.6. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

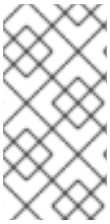
OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 core 사용자의 ~/.ssh/authorized_keys 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 core로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. ./openshift-install gather 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.



참고

AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

절차

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: ~/.ssh/id_ed25519)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 '~/.ssh 디렉터리에 있는지 확인하십시오.

참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 ed25519 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 rsa 또는 ecdsa 알고리즘을 사용하는 키를 생성합니다.

2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 ~/.ssh/id_ed25519.pub 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 ./openshift-install gather 명령을 사용하려는 경우 필요합니다.

참고

일부 배포에서는 ~/.ssh/id_rsa 및 ~/.ssh/id_dsa와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. ssh-agent 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. ssh-agent에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1

SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: ~/.ssh/id_ed25519).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

17.4.7. 설치 프로그램 받기

OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

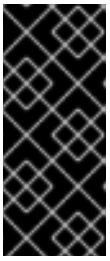
- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

절차

1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.

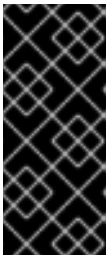
2. 인프라 공급자를 선택합니다.

3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.



중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.



중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 OpenShift Container Platform 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

17.4.8. 시스템 신뢰에 vCenter 루트 CA 인증서 추가

설치 프로그램이 vCenter API에 액세스해야 하므로 OpenShift Container Platform 클러스터를 설치하기 전에 vCenter의 신뢰할 수 있는 루트 CA 인증서를 시스템 신뢰에 추가해야 합니다.

절차

1. vCenter 홈 페이지에서 vCenter의 루트 CA 인증서를 다운로드합니다. vSphere Web Services SDK 섹션에서 Download trusted root CA certificates를 클릭합니다. <vCenter>/certs/download.zip 파일이 다운로드됩니다.

2. vCenter 루트 CA 인증서가 포함된 압축 파일의 압축을 풉니다. 압축 파일의 내용은 다음 파일 구조와 유사합니다.

```
certs
├── lin
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
├── mac
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
└── win
    ├── 108f4d17.0.crt
    ├── 108f4d17.r1.crl
    ├── 7e757f6a.0.crt
    ├── 8e4f8471.0.crt
    └── 8e4f8471.r0.crl

3 directories, 15 files
```

3. 운영 체제 파일을 시스템 신뢰에 추가합니다. 예를 들어 Fedora 운영 체제에서는 다음 명령을 실행합니다.

-

```
# cp certs/lin/* /etc/pki/ca-trust/source/anchors
```

4. 시스템 신뢰를 업데이트합니다. 예를 들어 **Fedora** 운영 체제에서는 다음 명령을 실행합니다.

```
# update-ca-trust extract
```

17.4.9. 설치 구성 파일 만들기

VMware vSphere에 설치하는 **OpenShift Container Platform** 클러스터를 사용자 지정할 수 있습니다.

사전 요구 사항

- **OpenShift Container Platform** 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.
- 서브스크립션 수준에서 서비스 주체 권한을 획득합니다.

절차

1. **install-config.yaml** 파일을 생성합니다.
   a. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 **<installation_directory>**는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.

중요

비어 있는 디렉터리를 지정합니다. 부트스트랩 **X.509** 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 **OpenShift Container Platform** 버전에서 설치 파일을 복사할 때는 주의하십시오.

b.

화면에 나타나는 지시에 따라 클라우드에 대한 구성 세부 사항을 입력합니다.

i.

선택사항: 클러스터 시스템에 액세스하는 데 사용할 SSH 키를 선택합니다.

참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 ssh-agent 프로세스가 사용하는 SSH 키를 지정합니다.

ii.

대상 플랫폼으로 vsphere를 선택합니다.

iii.

vCenter 인스턴스의 이름을 지정합니다.

iv.

클러스터를 생성하는 데 필요한 권한이 있는 vCenter 계정의 사용자 이름과 암호를 지정합니다.

설치 프로그램이 vCenter 인스턴스에 연결됩니다.

v.

연결할 vCenter 인스턴스에서 데이터 센터를 선택합니다.

vi.

사용할 기본 vCenter 데이터 저장소를 선택합니다.

vii.

OpenShift Container Platform 클러스터를 설치할 vCenter 클러스터를 선택합니다. 설치 프로그램은 vSphere 클러스터의 루트 리소스 풀을 기본 리소스 풀로 사용합니다.

viii.

vCenter 인스턴스에서 구성한 가상 IP 주소 및 DNS 레코드가 포함된 네트워크를 선택합니다.

ix.

컨트롤 플레인 API 액세스를 위해 구성한 가상 IP 주소를 입력합니다.

v

x.
클러스터 인그레스용으로 구성한 가상 **IP** 주소를 입력합니다.

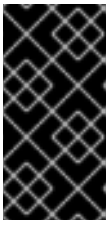
xi.
기본 도메인을 입력합니다. 이 기본 도메인은 구성한 **DNS** 레코드에서 사용한 것과 동일해야 합니다.

xii.
클러스터를 설명할 수 있는 이름을 입력합니다. 클러스터 이름은 구성한 **DNS** 레코드에서 사용한 것과 동일해야 합니다.

xiii.
Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 붙여넣습니다.

2.
install-config.yaml 파일을 수정합니다. 사용 가능한 매개변수에 대한 자세한 정보는 “설치 구성 매개변수” 섹션에서 확인할 수 있습니다.

3.
여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.



중요

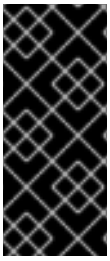
install-config.yaml 파일은 설치 과정에서 사용됩니다. 이 파일을 재사용하려면 지금 백업해야 합니다.

17.4.9.1. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 매개변수 값을 제공하여 클러스터를 호스팅할 클라우드 플랫폼에서 사용자 계정을 설명하고 선택사항으로 클러스터의 플랫폼을 사용자 지정합니다. **install-config.yaml** 설치 구성 파일을 생성할 때 명령줄을 통해 필요한 매개변수 값을 제공합니다. 클러스터를 사용자 지정하면 **install-config.yaml** 파일을 수정하여 플랫폼에 대한 세부 정보를 제공할 수 있습니다.

참고

설치한 후에는 **install-config.yaml** 파일에서 이러한 매개변수를 수정할 수 없습니다.



중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

17.4.9.1.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 17.26. 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	**dev** 와 같은 소문자 및 하이픈(**-**)의 문자열입니다.

매개변수	설명	값
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

17.4.9.1.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 17.27. 네트워크 매개변수

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체 **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.

매개변수	설명	값
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.
networking.clusterNetwork	Pod의 IP 주소 블록입니다. 기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된**cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16

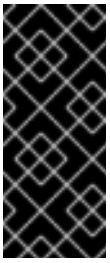
매개변수	설명	값
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16**  **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

17.4.9.1.3. 선택적 구성 매개변수

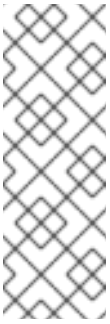
선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 17.28. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열

매개변수	설명	값
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은 **3**입니다.
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열

매개변수	설명	값
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 기본값은 **External**입니다. 이 필드를 **Internal** 로 설정하는 것은 클라우드 이외의 플랫폼에서는 지원되지 않습니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

17.4.9.1.4. 추가 VMware vSphere 구성 매개변수

추가 VMware vSphere 구성 매개변수는 다음 표에 설명되어 있습니다.

표 17.29. 추가 VMware vSphere 클러스터 매개변수

매개변수	설명	값
platform.vsphere.vCenter	vCenter 서버의 정규화된 호스트 이름 또는 IP 주소입니다.	문자열
platform.vsphere.username	vCenter 인스턴스에 연결하는 데 사용할 사용자 이름입니다. 이 사용자에게는 최소한 vSphere에서 정적 또는 동적 영구 볼륨 프로비저닝 에 필요한 역할과 권한이 있어야 합니다.	문자열
platform.vsphere.password	vCenter 사용자 이름의 암호입니다.	문자열
platform.vsphere.datacenter	vCenter 인스턴스에서 사용할 데이터 센터의 이름입니다.	문자열
platform.vsphere.defaultDatastore	프로비저닝 볼륨에 사용할 기본 데이터 저장소의 이름입니다.	문자열
platform.vsphere.folder	*선택사항*입니다. 설치 프로그램이 가상 머신을 생성하는 기존 폴더의 절대 경로입니다. 이 값을 제공하지 않으면 설치 프로그램이 데이터 센터 가상 머신 폴더에 인프라 ID로 이름이 지정된 폴더를 생성합니다.	문자열(예: **/<datacenter_name>/vm/<folder_name>/<subfolder_name>**).

매개변수	설명	값
platform.vsphere.network	vCenter 인스턴스에서 구성한 가상 IP 주소 및 DNS 레코드가 포함된 네트워크입니다.	문자열
platform.vsphere.cluster	OpenShift Container Platform 클러스터를 설치할 vCenter 클러스터입니다.	문자열
platform.vsphere.apiVIP	컨트롤 플레인 API 액세스를 위해 구성한 가상 IP(VIP) 주소입니다.	IP 주소 (예: **128.0.0.1**)
platform.vsphere.ingressVIP	클러스터 인그레스용으로 구성한 가상 IP(VIP) 주소입니다.	IP 주소 (예: **128.0.0.1**)

17.4.9.1.5. 선택적 VMware vSphere 시스템 풀 구성 매개변수

다음 표에는 선택적 VMware vSphere 시스템 풀 구성 매개 변수가 설명되어 있습니다.

표 17.30. 선택적 VMware vSphere 시스템 풀 매개변수

매개변수	설명	값
platform.vsphere.clusterOSImage	설치 프로그램이 RHCOS 이미지를 다운로드하는 위치입니다. 네트워크가 제한된 환경에서 설치를 수행하려면이 매개변수를 설정해야 합니다.	HTTP 또는 HTTPS URL (선택 옵션으로 SHA-256 형식의 체크섬을 사용) 예: **https://mirror.openshift.com/images/rhcos-<version>-vmware.<architecture>.ova**.
platform.vsphere.osDisk.diskSizeGB	디스크 크기(GB)입니다.	정수
platform.vsphere.cpus	가상 머신을 할당할 총 가상 프로세서 코어 수입니다. **platform.vsphere.cpus** 의 값은 **platform.vsphere.coresPerSocket** 값의 여러 개여야 합니다.	정수
platform.vsphere.coresPerSocket	가상 머신의 소켓당 코어 수입니다. 가상 머신의 가상 소켓 수는 **platform.vsphere.cpus/platform.vsphere.coresPerSocket** 입니다. 컨트롤 플레인 노드 및 작업자 노드의 기본값은 각각 **4** 및 **2** 입니다.	정수
platform.vsphere.memoryMB	가상 머신의 메모리 크기(MB)입니다.	정수

17.4.9.2. 설치 관리자 프로비저닝 VMware vSphere 클러스터의 샘플 install-config.yaml 파일

install-config.yaml 파일을 사용자 지정하여 OpenShift Container Platform 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.

```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 3
  platform:
    vsphere: (4)
      cpus: 2
      coresPerSocket: 2
      memoryMB: 8192
      osDisk:
        diskSizeGB: 120
controlPlane: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3
  platform:
    vsphere: (7)
      cpus: 4
      coresPerSocket: 2
      memoryMB: 16384
      osDisk:
        diskSizeGB: 120
metadata:
  name: cluster (8)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  vsphere:
    vcenter: your.vcenter.server
    username: username
    password: password
    datacenter: datacenter
    defaultDatastore: datastore
    folder: folder
    network: VM_Network
    cluster: vsphere_cluster_name (9)
    apiVIP: api_vip
    ingressVIP: ingress_vip
```

```
fips: false
pullSecret: '{"auths": ...}'
sshKey: 'ssh-ed25519 AAAA...'
```

1

클러스터의 기본 도메인입니다. 모든 **DNS** 레코드는 이 기본 도메인의 하위 도메인이어야 하며 클러스터 이름을 포함해야 합니다.

2 5

controlPlane 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

3 6

동시 멀티스레딩 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. 일부 클러스터 시스템에서 동시 멀티스레딩을 비활성화할 경우에는 해당 멀티스레딩을 모든 클러스터 시스템에서 비활성화해야 합니다.

중요

동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다. 동시 멀티스레딩을 비활성화하는 경우 시스템은 **8**개 이상의 **CPU**와 **32GB**의 **RAM**을 사용해야 합니다.

4 7

선택사항: 컴퓨팅 및 컨트롤 플레인 시스템의 시스템 풀 매개변수에 대한 추가 구성을 제공하십시오.

8

DNS 레코드에 지정한 클러스터 이름입니다.

9

OpenShift Container Platform 클러스터를 설치할 **vSphere** 클러스터입니다. 설치 프로그램은 **vSphere** 클러스터의 루트 리소스 풀을 기본 리소스 풀로 사용합니다.

17.4.9.3. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 **HTTP** 또는 **HTTPS** 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 **OpenShift Container Platform** 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 **API**에 대한 호출을 포함하여 모든 클러스터 발신 **(Egress)** 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.

참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr, networking.clusterNetwork[].cidr, networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 **Red Hat OpenStack Platform (RHOSP)**에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점**(169.254.169.254)**로 채워집니다.

절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1

2

클러스터 외부에서 **HTTPS** 연결을 구축하는 데 사용할 프록시 **URL**입니다.

3

대상 도메인 이름, **IP** 주소 또는 프록시에서 제외할 기타 네트워크 **CIDR**로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. ***를 사용하여 모든 대상에 대해 프록시를 바이패스합니다. **vCenter**의 **IP** 주소와 해당 시스템에 사용하는 **IP** 범위를 포함해야 합니다.

4

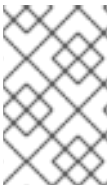
이 값을 제공하면 설치 프로그램에서 추가 **CA** 인증서를 보유할 **openshift -config** 네임스페이스에 **user-ca- bundle** 이라는 구성 맵을 생성합니다. **additionalTrustBundle** 및 하나 이상의 프록시 설정을 제공하는 경우 프록시 오브젝트는 **trustedCA** 필드의 **user-ca-bundle** 구성 맵을 참조하도록 구성됩니다. 그러면 **Cluster Network Operator**에서 **trustedCA** 매개변수에 대해 지정된 콘텐츠를 **RHCOS** 신뢰 번들과 병합하는 **trusted-ca-bundle** 구성 맵을 생성합니다. 프록시의 **ID** 인증서를 **RHCOS** 트러스트 번들에 있는 기관에서 서명하지 않은 경우 **additionalTrustBundle** 필드가 있어야 합니다.

참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 **OpenShift Container Platform**을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 **nil**이 됩니다.



참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

17.4.10. 네트워크 구성 단계

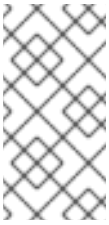
OpenShift Container Platform을 설치하기 전에 네트워크 구성을 사용자 지정할 수 있습니다.

1 단계

매니페스트 파일을 생성하기 전에 **install-config.yaml** 파일에서 다음 네트워크 관련 필드를 사용자 지정할 수 있습니다.

- **networking.networkType**
- **networking.clusterNetwork**
- **networking.serviceNetwork**
- **networking.machineNetwork**

이러한 필드에 대한 자세한 내용은 *설치 구성 매개 변수*에서 참조하십시오.



참고

기본 **NIC**가 상주하는 **CIDR**과 일치하도록 **networking.machineNetwork**를 설정합니다.

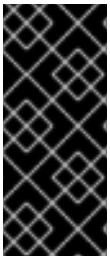
2 단계

openshift-install create manifests 를 실행하여 매니페스트 파일을 생성한 후 수정할 필드로 사용자 지정된 **Cluster Network Operator** 매니페스트를 정의할 수 있습니다. 매니페스트를 사용하여 고급 네트워크 구성을 지정할 수 있습니다.

install-config.yaml 파일에서 **1** 단계에 지정된 값을 **2** 단계에서 재정의할 수 없습니다. 그러나 **2** 단계에서 클러스터 네트워크 공급자를 추가로 사용자 지정할 수 있습니다.

17.4.11. 고급 네트워크 구성 지정

클러스터 네트워크 제공자의 고급 네트워크 구성을 사용하여 클러스터를 기존 네트워크 환경에 통합할 수 있습니다. 클러스터를 설치하기 전에만 고급 네트워크 구성을 지정할 수 있습니다.



중요

설치 프로그램에서 생성한 **OpenShift Container Platform** 매니페스트 파일을 수정하여 네트워크 구성을 사용자 정의하는 것은 지원되지 않습니다. 다음 절차에서와 같이 생성한 매니페스트 파일을 적용할 수 있습니다.

사전 요구 사항

- **install-config.yaml** 파일을 생성하고 수정 작업을 완료했습니다.

절차

1. 설치 프로그램이 포함된 디렉터리로 변경하고 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>**는 클러스터의 **install-config.yaml** 파일이 포함된 디렉터리의 이름을 지정합니다.

2. **<installation_directory>/ manifests/** 디렉토리에 **cluster-network-03-config.yml**이라는 **stub** 매니페스트 파일을 만듭니다.

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
```

3. 다음과 같이 **cluster-network-03-config.yml** 파일에서 클러스터의 고급 네트워크 구성을 지정합니다.

OpenShift SDN 네트워크 공급자에 대한 다른 **VXLAN** 포트 지정

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
```

```
  defaultNetwork:
    openshiftSDNConfig:
      vxlanPort: 4800
```

OVN-Kubernetes 네트워크 공급자의 IPsec 활성화

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    ovnKubernetesConfig:
      ipsecConfig: {}
```

4. 선택사항: manifests/cluster-network-03-config.yml 파일을 백업합니다. 설치 프로그램은 Ignition 구성 파일을 생성할 때 manifests/ 디렉터리를 사용합니다.

17.4.12. CNO(Cluster Network Operator) 구성

클러스터 네트워크의 구성은 CNO(Cluster Network Operator) 구성의 일부로 지정되며 cluster라는 이름의 CR(사용자 정의 리소스) 오브젝트에 저장됩니다. CR은 operator.openshift.io API 그룹에서 Network API의 필드를 지정합니다.

CNO 구성은 Network.config.openshift.io API 그룹의 Network API에서 클러스터 설치 중에 다음 필드를 상속하며 이러한 필드는 변경할 수 없습니다.

clusterNetwork

Pod IP 주소가 할당되는 IP 주소 풀입니다.

serviceNetwork

서비스를 위한 IP 주소 풀입니다.

defaultNetwork.type

OpenShift SDN 또는 OVN-Kubernetes와 같은 클러스터 네트워크 공급자입니다.

cluster라는 CNO 오브젝트에서 defaultNetwork 오브젝트의 필드를 설정하여 클러스터의 클러스터 네트워크 공급자 구성을 지정할 수 있습니다.

17.4.12.1. CNO(Cluster Network Operator) 구성 오브젝트

CNO(Cluster Network Operator)의 필드는 다음 표에 설명되어 있습니다.

표 17.31. CNO(Cluster Network Operator) 구성 오브젝트

필드	유형	설명
metadata.name	**string**	CNO 개체 이름입니다. 이 이름은 항상 **cluster**입니다.
spec.clusterNetwork	**array**	Pod IP 주소가 할당되는 IP 주소 블록과 클러스터의 각 개별 노드에 할당된 서브넷 접두사 길이를 지정하는 목록입니다. 예를 들면 다음과 같습니다. spec: clusterNetwork: - cidr: 10.128.0.0/19 hostPrefix: 23 - cidr: 10.128.32.0/19 hostPrefix: 23 매니페스트를 생성하기 전에 **install-config.yaml** 파일에서만 이 필드를 사용자 지정할 수 있습니다. 값은 매니페스트 파일에서 읽기 전용입니다.
spec.serviceNetwork	**array**	서비스의 IP 주소 블록입니다. OpenShift SDN 및 OVN-Kubernetes CNI(Container Network Interface) 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다. 예를 들면 다음과 같습니다. spec: serviceNetwork: - 172.30.0.0/14 매니페스트를 생성하기 전에 **install-config.yaml** 파일에서만 이 필드를 사용자 지정할 수 있습니다. 값은 매니페스트 파일에서 읽기 전용입니다.
spec.defaultNetwork	**object**	클러스터 네트워크의 CNI(Container Network Interface) 클러스터 네트워크 공급자를 구성합니다.

필드	유형	설명
spec.kubeProxyConfig	**object**	이 개체의 필드는 kube-proxy 구성을 지정합니다. OVN-Kubernetes 클러스터 네트워크 공급자를 사용하는 경우 kube-proxy 구성이 적용되지 않습니다.

defaultNetwork 오브젝트 구성

defaultNetwork 오브젝트의 값은 다음 표에 정의되어 있습니다.

표 17.32. defaultNetwork 오브젝트

필드	유형	설명
type	**string**	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, 클러스터 네트워크 공급자가 설치 중에 선택됩니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다. **참고** OpenShift Container Platform은 기본적으로 OpenShift SDN CNI(Container Network Interface) 클러스터 네트워크 공급자를 사용합니다.
openshiftSDNConfig	**object**	이 오브젝트는 OpenShift SDN 클러스터 네트워크 공급자에만 유효합니다.
ovnKubernetesConfig	**object**	이 오브젝트는 OVN-Kubernetes 클러스터 네트워크 공급자에만 유효합니다.

OpenShift SDN CNI 네트워크 공급자에 대한 구성

다음 표에서는 OpenShift SDN Container Network Interface (CNI) 클러스터 네트워크 공급자의 구성 필드를 설명합니다.

표 17.33. openshiftSDNConfig 오브젝트

필드	유형	설명

필드	유형	설명
mode	**string**	OpenShift SDN의 네트워크 격리 모드를 구성합니다. 기본값은 **NetworkPolicy**입니다. **Multitenant** 및 **Subnet** 값은 OpenShift Container Platform 3.x 버전과의 역호환할 수 있지만 권장되지는 않습니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
mtu	**integer**	VXLAN 오버레이 네트워크의 최대 전송 단위(MTU)입니다. 이는 기본 네트워크 인터페이스의 MTU를 기준으로 자동 탐지됩니다. 일반적으로 감지된 MTU를 재정의할 필요는 없습니다. 자동 감지 값이 예상 밖인 경우 노드의 기본 네트워크 인터페이스의 MTU가 올바른지 확인합니다. 이 옵션을 사용하여 노드의 기본 네트워크 인터페이스의 MTU 값을 변경할 수 없습니다. 클러스터에 다른 노드에 대한 다른 MTU 값이 필요한 경우 이 값을 클러스터의 가장 낮은 MTU 값보다 **50** 미만으로 설정해야 합니다. 예를 들어 클러스터의 일부 노드에는 **9001**의 MTU가 있고 일부에는 **1500**의 MTU가 있는 경우 이 값을 **1450**으로 설정해야 합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
vxlanPort	**integer**	모든 VXLAN 패킷에 사용할 포트입니다. 기본값은 **4789**입니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다. 기존 노드가 다른 VXLAN 네트워크에 속하는 가상 환경에서 실행 중인 경우에는 기본값을 변경해야 할 수도 있습니다. 예를 들어 VMware NSX-T 위에서 OpenShift SDN 오버레이를 실행할 때 두 SDN이 동일한 기본 VXLAN 포트 번호를 사용하므로 VXLAN의 대체 포트를 선택해야 합니다. AWS(Amazon Web Services)에서는 포트 **9000**과 포트 **9999** 사이에서 VXLAN의 대체 포트를 선택할 수 있습니다.

OpenShift SDN 구성 예

```
defaultNetwork:
  type: OpenShiftSDN
  openshiftSDNConfig:
    mode: NetworkPolicy
    mtu: 1450
    vxlanPort: 4789
```

OVN-Kubernetes CNI 클러스터 네트워크 공급자에 대한 구성

다음 표에서는 OVN-Kubernetes CNI 클러스터 네트워크 공급자의 구성 필드를 설명합니다.

표 17.34. ovnKubernetesConfig object

필드	유형	설명
mtu	**integer**	Geneve(Generic Network Virtualization Encapsulation) 오버레이 네트워크의 MTU(최대 전송 단위)입니다. 이는 기본 네트워크 인터페이스의 MTU를 기준으로 자동 탐지됩니다. 일반적으로 감지된 MTU를 재정의할 필요는 없습니다. 자동 감지 값이 예상 밖인 경우 노드의 기본 네트워크 인터페이스의 MTU가 올바른지 확인합니다. 이 옵션을 사용하여 노드의 기본 네트워크 인터페이스의 MTU 값을 변경할 수 없습니다. 클러스터에 다른 노드에 대한 다른 MTU 값이 필요한 경우, 이 값을 클러스터의 가장 낮은 MTU 값보다 **100** 미만으로 설정해야 합니다. 예를 들어, 클러스터의 일부 노드에 **9001**의 MTU가 있고 일부에는 **1500**의 MTU가 있는 경우 이 값을 **1400**으로 설정해야 합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
genevePort	**integer**	모든 Geneve 패킷에 사용할 포트입니다. 기본값은 **6081**입니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
ipsecConfig	**object**	IPsec 암호화를 활성화하려면 빈 오브젝트를 지정합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
policyAuditConfig	**object**	네트워크 정책 감사 로깅을 사용자 정의할 구성 오브젝트를 지정합니다. 설정되지 않으면 기본값 감사 로그 설정이 사용됩니다.

표 17.35. policyAuditConfig object

필드	유형	설명
rateLimit	integer	노드당 1초마다 생성할 최대 메시지 수입니다. 기본값은 초당 **20** 개의 메시지입니다.
maxFileSize	integer	감사 로그의 최대 크기(바이트)입니다. 기본값은 **50000000** 또는 **50** MB입니다.

필드	유형	설명
대상	string	다음 추가 감사 로그 대상 중 하나입니다. **libc** 호스트에서 journald 프로세스의 libc **syslog()** 함수입니다. **udp:<host>:<port>** syslog 서버입니다. **<host>:<port>**를 syslog 서버의 호스트 및 포트로 바꿉니다. **unix:<file>** **<file>**로 지정된 Unix Domain Socket 파일입니다. **null** 감사 로그를 추가 대상으로 보내지 마십시오.
syslogFacility	string	RFC5424에 정의된 **kern**과 같은 syslog 기능입니다. 기본값은 **local0**입니다.

OVN-Kubernetes 구성 예

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
    ipsecConfig: {}
```

kubeProxyConfig 오브젝트 구성

kubeProxyConfig 오브젝트의 값은 다음 표에 정의되어 있습니다.

표 17.36. kubeProxyConfig object

필드	유형	설명

필드	유형	설명
iptablesSyncPeriod	**string**	**iptables** 규칙의 새로 고침 간격입니다. 기본값은 **30s**입니다. 유효 접미사로 **s**, **m**, **h**가 있으며, 자세한 설명은 Go **time** 패키지 문서를 참조하십시오. **참고** OpenShift Container Platform 4.3 이상에서는 성능이 개선되어 더 이상 **iptablesSyncPeriod** 매개변수를 조정할 필요가 없습니다.

proxyArguments.iptables-min-sync-period	**array**	**iptables** 규칙을 새로 고치기 전 최소 기간입니다. 이 필드를 통해 새로 고침 간격이 너무 짧지 않도록 조정할 수 있습니다. 유효 접미사로 **s**, **m**, **h**가 있으며, 자세한 설명은 Go **time** 패키지를 참조하십시오. 기본값은 다음과 같습니다.
		
		`kubeProxyConfig:`
		`  proxyArguments:`
		`  iptables-min-sync-period:`
		`  - 0s`

17.4.13. 클러스터 배포

호환되는 클라우드 플랫폼에 OpenShift Container Platform을 설치할 수 있습니다.

중요

최초 설치 과정에서 설치 프로그램의 create cluster 명령을 한 번만 실행할 수 있습니다.

사전 요구 사항

- **OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.**

프로세스

1. 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터 배포를 초기화합니다.

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
  --log-level=info 2
```

1 **<installation_directory>** 값으로 사용자 지정한 **./install-config.yaml** 파일의 위치를 지정합니다.

2 다른 설치 세부 사항을 보려면 **info** 대신 **warn, debug** 또는 **error**를 지정합니다.

클러스터 배포가 완료되면 웹 콘솔로 연결되는 링크와 **kubeadmin** 사용자의 인증 정보가 포함된 클러스터 액세스 지침이 사용자 터미널에 표시됩니다.

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-Wt5AL"
INFO Time elapsed: 36m22s
```

+

참고

설치에 성공하면 클러스터 액세스 및 인증 정보도 **<installation_directory>/.openshift_install.log**로 출력됩니다.

+

중요

- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 node-bootstrapper 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.

- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

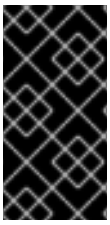
+

중요

설치 프로그램에서 생성되는 파일이나 설치 프로그램을 삭제해서는 안 됩니다. 클러스터를 삭제하려면 두 가지가 모두 필요합니다.

17.4.14. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift CLI(oc)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 oc를 설치할 수 있습니다.



중요

이전 버전의 oc를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 oc를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 Linux에서 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.

2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.

3. **OpenShift v4.9 Linux Client** 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.

4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. **oc** 바이너리를 **PATH**에 있는 디렉터리에 배치합니다.

   PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 **OpenSfhit CLI** 설치

다음 절차에 따라 **Windows**에 **OpenShift CLI(oc)** 바이너리를 설치할 수 있습니다.

프로세스

1. **Red Hat** 고객 포털에서 **OpenShift Container Platform** 다운로드 페이지로 이동합니다.

2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.

3. **OpenShift v4.9 Windows Client** 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.

4. **ZIP** 프로그램으로 아카이브의 압축을 풉니다.

5. oc 바이너리를 PATH에 있는 디렉터리로 이동합니다.

   PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치

다음 절차에 따라 macOS에서 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.
2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. OpenShift v4.9 MacOSX Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 해제하고 압축을 풉니다.
5. oc 바이너리 PATH의 디렉터리로 이동합니다.

   PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
$ oc <command>
```

17.4.15. CLI를 사용하여 클러스터에 로그인

클러스터 kubeconfig 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. kubeconfig 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- oc CLI를 설치했습니다.

프로세스

1. kubeadmin 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 oc 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

17.4.16. 레지스트리 스토리지 생성

클러스터를 설치한 후 레지스트리 Operator를 위한 스토리지를 생성해야 합니다.

17.4.16.1. 설치 중 제거된 이미지 레지스트리

공유 가능한 개체 스토리지를 제공하지 않는 플랫폼에서 OpenShift Image Registry Operator는 자체적으로 Removed로 부트스트랩합니다. 이를 통해 openshift-installer가 이러한 플랫폼 유형에서 설치를 완료할 수 있습니다.

설치 후 managementState를 Removed에서 Managed로 전환하도록 Image Registry Operator 구성을 편집해야 합니다.

참고

Prometheus 콘솔은 ImageRegistryRemoved 경고를 제공합니다. 다음은 예시 경고입니다.

"이미지 레지스트리가 제거되었습니다. ImageStreamTags를 참조하는 ImageStreamTags, BuildConfigs 및 DeploymentConfigs가 예상대로 작동하지 않을 수 있습니다. 스토리지를 구성하고 configs.imageregistry.operator.openshift.io를 편집하여 구성을 Managed 상태로 업데이트하십시오."

17.4.16.2. 이미지 레지스트리 스토리지 구성

기본 스토리지를 제공하지 않는 플랫폼에서는 처음에 Image Registry Operator를 사용할 수 없습니다. 설치한 후에 스토리지를 사용하도록 레지스트리를 구성하여 Registry Operator를 사용 가능하도록 만들어야 합니다.

프로덕션 클러스터에 필요한 영구 볼륨을 구성하는 과정의 지침이 표시됩니다. 해당하는 경우, 프로덕션 환경 외 클러스터에서만 사용할 수 있는 저장 위치로서 빈 디렉터리를 구성하는 과정의 지침이 표시됩니다.

업그레이드 중에 Recreate 롤아웃 전략을 사용하여 이미지 레지스트리의 블록 스토리지 유형 사용을 허용하기 위한 추가 지침이 제공됩니다.

17.4.16.2.1. VMware vSphere용 레지스트리 스토리지 구성

클러스터 관리자는 설치한 후 스토리지를 사용하도록 레지스트리를 구성해야 합니다.

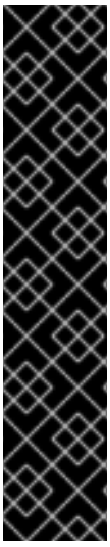
사전 요구 사항

- 클러스터 관리자 권한이 있어야 합니다.
- VMware vSphere에 클러스터가 있어야 합니다.
- Red Hat OpenShift Container Storage와 같이 클러스터용 영구 스토리지 프로비저닝.

중요

OpenShift Container Platform은 복제본이 하나만 있는 경우 이미지 레지스트리 스토리지에 대한 ReadWriteOnce 액세스를 지원합니다. ReadWriteOnce 액세스에서는 레지스트리가 Recreate 롤아웃 전략을 사용해야 합니다. 두 개 이상의 복제본으로 고 가용성을 지원하는 이미지 레지스트리를 배포하려면 ReadWriteMany 액세스가 필요합니다.

- "100Gi" 용량이 필요합니다.



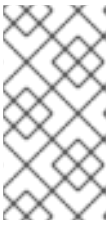
중요

테스트 결과, RHEL의 NFS 서버를 핵심 서비스용 스토리지 백엔드로 사용하는 데 문제가 있는 것으로 나타납니다. 여기에는 OpenShift Container Registry and Quay, 스토리지 모니터링을 위한 Prometheus, 로깅 스토리지를 위한 Elasticsearch가 포함됩니다. 따라서 RHEL NFS를 사용하여 핵심 서비스에서 사용하는 PV를 백업하는 것은 권장되지 않습니다.

마켓플레이스의 다른 NFS 구현에는 이러한 문제가 나타나지 않을 수 있습니다. 이러한 OpenShift Container Platform 핵심 구성 요소에 대해 완료된 테스트에 대한 자세한 내용은 개별 NFS 구현 공급업체에 문의하십시오.

프로세스

1. 스토리지를 사용하도록 레지스트리를 구성하기 위해 configs.imageregistry/cluster 리소스에서 spec.storage.pvc를 변경합니다.



참고

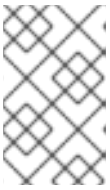
공유 스토리지를 사용할 때 보안 설정을 확인하여 외부에서의 액세스를 방지합니다.

2. 레지스트리 pod가 없는지 확인합니다.

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

출력 예

```
No resourses found in openshift-image-registry namespace
```



참고

출력에 레지스트리 Pod가 있는 경우 이 절차를 계속할 필요가 없습니다.

3. 레지스트리 구성을 확인합니다.

```
$ oc edit configs.imageregistry.operator.openshift.io
```

출력 예

```
storage:
  pvc:
    claim: 1
```

1

image-registry-storage 영구 볼륨 클레임 (PVC)을 자동으로 생성할 수 있도록 claim 필드를 비워 둡니다. PVC는 기본 스토리지 클래스를 기반으로 생성됩니다. 그러나 기본 스토리지 클래스에서 RADOS 블록 장치(RBD)와 같은

ReadWriteOnce(ReadWriteOnce) 볼륨을 제공할 수 있으므로 복제본이 두 개 이상될 때 문제가 발생할 수 있습니다.

4. **clusteroperator** 상태를 확인합니다.

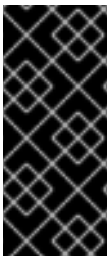
```
$ oc get clusteroperator image-registry
```

출력 예

```
NAME              VERSION                             AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                                 True        False         False      6h50m
```

17.4.16.2.2. VMware vSphere용 블록 레지스트리 스토리지 구성

클러스터 관리자로서 업그레이드 중에 이미지 레지스트리가 **vSphere VMDK(Virtual Machine Disk)**와 같은 블록 스토리지 유형을 사용할 수 있도록 허용하기 위해 **Recreate** 롤아웃 전략을 사용할 수 있습니다.



중요

블록 스토리지 볼륨이 지원되지만 프로덕션 클러스터에서 이미지 레지스트리와 함께 사용하는 것은 권장되지 않습니다. 레지스트리가 블록 스토리지에 구성된 설치는 레지스트리가 둘 이상의 복제본을 가질 수 없기 때문에 가용성이 높지 않습니다.

프로세스

1. 이미지 레지스트리 스토리지를 블록 스토리지 유형으로 설정하려면 **Recreate** 롤아웃 전략을 사용하고 복제본 **1**개 만으로 실행되도록 레지스트리를 패치합니다.

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. 블록 스토리지 장치에 **PV**를 프로비저닝하고 해당 볼륨의 **PVC**를 생성합니다. 요청된 블록 볼륨은 **RWO(ReadWriteOnce)** 액세스 모드를 사용합니다.

a. VMware vSphere PersistentVolumeClaim 개체를 정의하려면 다음 내용이 포함된 pvc.yaml 파일을 생성합니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage 1
  namespace: openshift-image-registry 2
spec:
  accessModes:
  - ReadWriteOnce 3
  resources:
    requests:
      storage: 100Gi 4
```

1 PersistentVolumeClaim 개체를 표시하는 고유한 이름입니다.

2 PersistentVolumeClaim 오브젝트의 네임스페이스로 openshift-image-registry입니다.

3 영구 볼륨 클레임의 액세스 모드입니다. ReadWriteOnce를 사용하면 단일 노드에서 읽기 및 쓰기 권한으로 볼륨을 마운트할 수 있습니다.

4 영구 볼륨 클레임의 크기입니다.

b. 파일에서 PersistentVolumeClaim 오브젝트를 만듭니다.

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 올바른 PVC를 참조하도록 레지스트리 설정을 편집합니다.

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

출력 예

```
storage:
  pvc:
    claim: 1
```

1

사용자 지정 **PVC**를 만들면 **image-registry-storage PVC**의 기본 자동 생성을 위해 **claim** 필드를 비워둘 수 있습니다.

올바른 **PVC**를 참조하도록 레지스트리 스토리지를 구성하는 방법은 **vSphere** 용 레지스트리 구성을 참조하십시오.

17.4.17. VMware vSphere 볼륨 백업

OpenShift Container Platform은 새 볼륨을 독립 영구 디스크로 프로비저닝하여 클러스터의 모든 노드에서 볼륨을 자유롭게 연결 및 분리합니다. 결과적으로 스냅샷을 사용하는 볼륨을 백업하거나 스냅샷에서 볼륨을 복원할 수 없습니다. 자세한 내용은 스냅 샷 제한 을 참조하십시오.

프로세스

영구 볼륨의 백업을 생성하려면 다음을 수행합니다.

1. 영구 볼륨을 사용중인 애플리케이션을 중지합니다.
2. 영구 볼륨을 복제합니다.
3. 애플리케이션을 다시 시작합니다.
4. 복제된 볼륨의 백업을 만듭니다.
5. 복제된 볼륨을 삭제합니다.

17.4.18. 스틸 클럭 계정

기본적으로 설치 프로그램은 스틸 클럭 회계 매개 변수 (**stealclock.enabled**)를 활성화하지 않고 클러스터의 가상 머신을 프로비저닝합니다. 스틸링 클럭 계정을 사용하면 클러스터 문제 해결에 도움이 될 수 있습니다. 클러스터가 배포된 후 **vSphere Client**를 사용하여 각 가상 머신에서 이 매개변수를 활성화합니다.

자세한 내용은 이 **Red Hat** 기술 자료 문서 를 참조하십시오.

17.4.19. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 **Telemetry** 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 **Telemetry**가 자동으로 실행되고 **OpenShift Cluster Manager** 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 **OpenShift Cluster Manager**를 사용하여 자동으로 또는 **OpenShift Cluster Manager**를 사용하여 수동으로 유지 관리되는지 확인한 후 **subscription watch**를 사용하여 계정 또는 다중 클러스터 수준에서 **OpenShift Container Platform** 서브스크립션을 추적합니다.

추가 리소스

- **Telemetry** 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

17.4.20. 다음 단계

- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.
- 레지스트리를 설정하고 레지스트리 스토리지를 구성합니다.
- 선택 사항: **vSphere Problem Detector Operator**에서 이벤트를 보고 클러스터에 권한 또는 스토리지 구성 문제가 있는지 확인합니다.

17.5. 사용자 프로비저닝 인프라를 사용하여 VSPHERE에 클러스터 설치

OpenShift Container Platform 4.9 버전에서는 사용자가 프로비저닝하는 **VMware vSphere** 인프라에 클러스터를 설치할 수 있습니다.

참고

OpenShift Container Platform은 단일 **VMware vCenter**에만 클러스터 배포를 지원합니다. 여러 **vCenter**에 머신/머신 세트가 있는 클러스터 배포는 지원되지 않습니다.

중요

사용자가 프로비저닝한 인프라 설치를 수행하는 단계는 예시용으로만 제공됩니다. 사용자가 제공하는 인프라를 사용하여 클러스터를 설치하려면 **vSphere** 플랫폼 및 **OpenShift Container Platform** 설치 프로세스에 대한 정보가 필요합니다. 사용자 프로비저닝 인프라 설치 지침을 가이드로 사용하십시오. 다른 방법을 통해 필요한 리소스를 자유롭게 생성할 수 있습니다.

17.5.1. 사전 요구 사항

- **OpenShift Container Platform** 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.
- 클러스터용 영구 스토리지를 프로비저닝합니다. 프라이빗 이미지 레지스트리를 배포하려면 스토리지에서 **ReadWriteMany** 액세스 모드를 제공해야 합니다.
- 설치를 완료하려면 **vSphere** 호스트에 **RHCOS(Red Hat Enterprise Linux CoreOS) OVA**를 업로드해야 합니다. 이 프로세스를 완료하는 시스템에는 **vCenter** 및 **ESXi** 호스트의 포트 **443**에 액세스해야 합니다. 포트 **443**에 액세스할 수 있는지 확인했습니다.
- 방화벽을 사용하는 경우 관리자가 포트 **443**에 액세스할 수 있음을 확인합니다. 설치에 성공하려면 포트 **443**에서 컨트롤 플레인 노드가 **vCenter** 및 **ESXi** 호스트에 연결할 수 있어야 합니다.
- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성해야 합니다.

참고

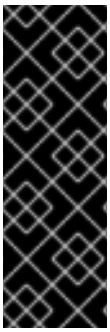
프록시를 구성하는 경우에도 해당 사이트 목록을 검토하십시오.

17.5.2. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.



중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

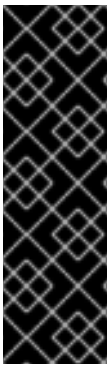
17.5.3. VMware vSphere 인프라 요구사항

사용하는 구성 요소의 요구사항을 충족하는 VMware vSphere 버전 6 또는 7 인스턴스에 OpenShift Container Platform 클러스터를 설치해야 합니다.

표 17.37. vSphere 가상 환경의 버전 요구 사항

가상 환경 제품	필요한 버전
VM 하드웨어 버전	13 이상
vSphere ESXi hosts	6.5 이상

가상 환경 제품	필요한 버전
vCenter 호스트	6.5 이상



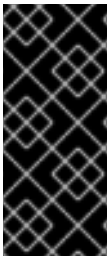
중요

이제 VMware vSphere 버전 6.7U2 또는 이전 버전에 클러스터를 설치하는 것이 더 이상 사용되지 않으며 가상 하드웨어 버전 13은 더 이상 사용되지 않습니다. 이러한 버전은 여전히 완전히 지원되지만 향후 OpenShift Container Platform 버전에서는 지원이 제거됩니다. 이제 OpenShift Container Platform의 vSphere 가상 머신의 하드웨어 버전 15가 기본값이 되었습니다. vSphere 노드의 하드웨어 버전을 업데이트하려면 " vSphere에서 실행되는 노드에서 하드웨어 업데이트" 문서를 참조하십시오.

표 17.38. VMware 구성 요소에 지원되는 최소 vSphere 버전

구성 요소	지원되는 최소 버전	설명
하이퍼바이저	HW 버전 13이 포함된 vSphere 6.5 이상	이 버전은 RHCOS(Red Hat Enterprise Linux CoreOS)가 지원하는 최소 버전입니다. Red Hat Enterprise Linux 8 지원 하이퍼바이저 목록을 참조하십시오.
In-tree 드라이버가 있는 스토리지	vSphere 6.5 이상	이 플러그인은 OpenShift Container Platform에 포함된 vSphere용 인트리 스토리지 드라이버를 사용하여 vSphere 스토리지를 생성합니다.
선택사항: 네트워킹(NSX-T)	vSphere 6.5U3 또는 vSphere 6.7U2 이상	OpenShift Container Platform에는 vSphere 6.5U3 또는 vSphere 6.7U2+가 필요합니다. NSX 및 OpenShift Container Platform의 호환성에 대한 자세한 내용은 VMware의 NSX 컨테이너 플러그인 설명서의 릴리스 노트 섹션을 참조하십시오.

vSphere 버전 6.5 인스턴스를 사용하는 경우 OpenShift Container Platform을 설치하기 전에 6.7U3 또는 7.0로 업그레이드하는 것이 좋습니다.



중요

OpenShift Container Platform을 설치하기 전에 ESXi 호스트의 시간이 동기화되었는지 확인해야 합니다. VMware 문서에서 Edit Time Configuration for a Host를 참조하십시오.

추가 리소스

- vSphere 노드의 하드웨어 버전을 업데이트하려면 vSphere에서 실행 중인 노드에서 하드웨어 업데이트를 참조하십시오.

17.5.4. 사용자 프로비저닝 인프라를 포함한 클러스터의 시스템 요구사항

사용자 프로비저닝 인프라가 포함된 클러스터의 경우, 필요한 모든 시스템을 배포해야 합니다.

이 섹션에서는 사용자 프로비저닝 인프라에 OpenShift Container Platform을 배포해야 하는 요구 사항에 대해 설명합니다.

17.5.4.1. 클러스터 설치에 필요한 시스템

최소 OpenShift Container Platform 클러스터에 다음과 같은 호스트가 필요합니다.

표 17.39. 최소 필수 호스트

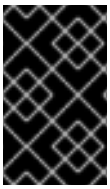
호스트	설명
임시 부트스트랩 시스템 한 개	컨트롤 플레인 시스템 세 개에 OpenShift Container Platform 클러스터를 배포하기 위한 부트스트랩 시스템이 클러스터에 필요합니다. 클러스터를 설치한 후 부트스트랩 시스템을 제거할 수 있습니다.
컨트롤 플레인 시스템 세 개	컨트롤 플레인 시스템은 컨트롤 플레인을 구성하는 Kubernetes 및 OpenShift Container Platform 서비스를 실행합니다.
두 개 이상의 컴퓨팅 시스템(작업자 시스템이라고도 함).	OpenShift Container Platform 사용자가 요청한 워크로드는 컴퓨팅 머신에서 실행됩니다.

중요

클러스터의 고가용성을 유지하려면 이러한 클러스터 시스템에 대해 별도의 물리적 호스트를 사용하십시오.

부트스트랩, 컨트롤 플레인 시스템은 운영 체제로 **RHCOS (Red Hat Enterprise Linux CoreOS)**를 사용해야 합니다. 그러나 컴퓨팅 머신은 **RHCOS(Red Hat Enterprise Linux CoreOS), RHEL (Red Hat Enterprise Linux) 7.9** 또는 **RHEL 8.4** 중에서 선택할 수 있습니다.

RHCOS는 **RHEL 8(Red Hat Enterprise Linux)**을 기반으로하며 모든 하드웨어 인증 및 요구사항을 모두 이어받습니다. Red Hat Enterprise Linux 기술 기능 및 제한을 참조하십시오.



중요

모든 가상 머신은 설치 프로그램과 동일한 데이터 저장소 및 폴더에 있어야 합니다.

17.5.4.2. 클러스터 설치를 위한 최소 리소스 요구 사항

각 클러스터 시스템이 다음과 같은 최소 요구사항을 충족해야 합니다.

표 17.40. 최소 리소스 요구사항

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
부트스트랩	RHCOS	4	16GB	100GB	300
컨트롤 플레인	RHCOS	4	16GB	100GB	300
컴퓨팅	RHCOS, RHEL 7.9 또는 RHEL 8.4 [3]	2	8GB	100GB	300

1. **SMT**(동시 멀티스레딩) 또는 하이퍼 스레딩이 활성화되지 않은 경우 하나의 **vCPU**는 하나의 물리적 코어와 동일합니다. 활성화하면 다음과 같은 공식을 사용하여 해당 비율을 계산합니다. **(**코어 당 스레드 수 **×** 코어 수**) ×** 소켓 수 **= vCPU** 수

2. **OpenShift Container Platform** 및 **Kubernetes**는 디스크 성능에 민감하며 특히 **10ms p99 fsync** 기간이 필요한 컨트롤 플레인 노드의 **etcd**에 더 빠른 스토리지가 권장됩니다. 많은 클라우

드 플랫폼에서 스토리지 크기와 IOPS를 함께 확장되므로 충분한 성능을 얻으려면 스토리지 볼륨을 과도하게 할당해야 할 수 있습니다.

3. 사용자가 프로비저닝한 모든 설치와 마찬가지로 클러스터에서 RHEL 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. RHEL 7 컴퓨팅 머신 사용은 더 이상 사용되지 않으며 향후 OpenShift Container Platform 4 릴리스에서 제거될 예정입니다.

17.5.4.3. 인증서 서명 요청 관리

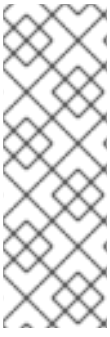
사용자가 프로비저닝하는 인프라를 사용하는 경우 자동 시스템 관리 기능으로 인해 클러스터의 액세스가 제한되므로 설치한 후 클러스터 인증서 서명 요청(CSR)을 승인하는 메커니즘을 제공해야 합니다. kube-controller-manager는 kubelet 클라이언트 CSR만 승인합니다. machine-approver는 올바른 시스템에서 발행한 요청인지 확인할 수 없기 때문에 kubelet 자격 증명을 사용하여 요청하는 서비스 인증서의 유효성을 보장할 수 없습니다. kubelet 서빙 인증서 요청의 유효성을 확인하고 요청을 승인하는 방법을 결정하여 구현해야 합니다.

17.5.4.4. 사용자 프로비저닝 인프라에 대한 네트워킹 요구사항

모든 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템이 부팅 중에 Ignition 구성 파일을 가져오려면 initramfs에 네트워킹을 구성해야 합니다.

초기 부팅 과정에서 시스템에 필요한 부팅 옵션을 제공하여 DHCP 서버를 통해 설정하거나 정적으로 설정하는 IP 주소 구성이 필요합니다. 네트워크 연결이 설정된 후 시스템은 HTTP 또는 HTTPS 서버에서 Ignition 구성 파일을 다운로드합니다. 그런 다음 Ignition 구성 파일을 사용하여 각 머신의 정확한 상태를 설정합니다. Machine Config Operator는 설치 후 새 인증서 또는 키 적용과 같은 머신에 대한 추가 변경을 완료합니다.

클러스터 시스템의 장기적인 관리를 위해 DHCP 서버를 사용하는 것이 좋습니다. DHCP 서버가 클러스터 시스템에 영구 IP 주소, DNS 서버 정보 및 호스트 이름을 제공하도록 구성되었는지 확인합니다.



참고

사용자 프로비저닝 인프라에 DHCP 서비스를 사용할 수 없는 경우 RHCOS 설치 시 노드에 IP 네트워킹 구성과 DNS 서버의 주소를 대신 제공할 수 있습니다. ISO 이미지에서 설치하는 경우 부팅 인수로 전달할 수 있습니다. 고정 IP 프로비저닝 *및 고급 네트워킹 옵션에 대한 자세한 내용은 RHCOS 설치 및 OpenShift Container Platform 부트스트랩 프로세스 시작* 섹션을 참조하십시오.

Kubernetes API 서버가 클러스터 시스템의 노드 이름을 확인할 수 있어야 합니다. API 서버와 작업자

노드가 서로 다른 영역에 있는 경우, **API** 서버가 노드 이름을 확인할 수 있도록 기본 **DNS** 검색 영역을 설정할 수 있습니다. 노드 개체와 모든 **DNS** 요청에서 항상 정규화된 도메인 이름으로 호스트를 가리키는 것도 지원되는 방법입니다

17.5.4.4.1. DHCP를 통해 클러스터 노드의 호스트 이름 설정

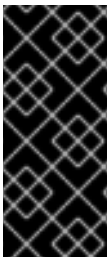
RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 호스트 이름은 **NetworkManager**를 통해 설정됩니다. 기본적으로 시스템은 **DHCP**를 통해 호스트 이름을 가져옵니다. **DHCP**에서 호스트 이름을 제공하지 않거나, 커널 인수를 통해 정적으로 설정하거나 다른 방법을 통해 설정하면 역방향 **DNS** 조회를 통해 얻을 수 있습니다. 역방향 **DNS** 조회는 노드에서 네트워크를 초기화한 후 수행되며 확인하는 데 시간이 걸릴 수 있습니다. 다른 시스템 서비스는 이 보다 먼저 시작하여 호스트 이름을 **localhost** 등으로 감지할 수 있습니다. **DHCP**를 사용하여 각 클러스터 노드의 호스트 이름을 제공하여 이 문제를 방지할 수 있습니다.

또한 **DHCP**를 통해 호스트 이름을 설정하면 **DNS** 분할 수평 구현 환경에서 수동으로 **DNS** 레코드 이름 구성 오류를 무시할 수 있습니다.

17.5.4.4.2. 네트워크 연결 요구사항

OpenShift Container Platform 클러스터 구성 요소가 통신할 수 있도록 시스템 간 네트워크 연결을 구성해야 합니다. 각 시스템에서 클러스터에 있는 다른 모든 시스템의 호스트 이름을 확인할 수 있어야 합니다.

이 섹션에서는 필요한 포트에 대해 자세히 설명합니다.



중요

연결된 **OpenShift Container Platform** 환경에서 모든 노드는 플랫폼 컨테이너의 이미지를 가져오고 **Red Hat**에 원격 측정 데이터를 제공하기 위해 인터넷에 액세스할 수 있어야 합니다.

표 17.41. 모든 시스템 간 통신에 사용되는 포트

프로토콜	포트	설명
ICMP	해당 없음	네트워크 연결성 테스트
TCP	**1936**	메트릭
	9000-**9999**	**9100**-**9101** 포트의 노드 내보내기 및 **9099** 포트의 Cluster Version Operator를 포함한 호스트 수준 서비스.

프로토콜	포트	설명
	10250-10259	Kubernetes에서 예약하는 기본 포트
	10256	openshift-sdn
UDP	**4789**	VXLAN 및 Geneve
	6081	VXLAN 및 Geneve
	9000-9999	**9100-9101** 포트의 노드 내보내기를 포함한 호스트 수준 서비스.
	500	IPsec IKE 패킷
	4500	IPsec NAT-T 패킷
TCP/UDP	**30000-32767**	Kubernetes 노드 포트
ESP	해당 없음	IPsec Encapsulating Security Payload (ESP)

표 17.42. 모든 시스템과 컨트롤 플레인 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**6443**	Kubernetes API

표 17.43. 컨트롤 플레인 머신 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**2379-2380**	etcd 서버 및 피어 포트

이더넷 어댑터 하드웨어 주소 요구사항

클러스터용 VM을 프로비저닝할 때 각 VM에 대해 구성된 이더넷 인터페이스는 VMware OUI(Organizationally Unique Identifier) 할당 범위의 MAC 주소를 사용해야 합니다.

- **00:05:69:00:00:00 ~ 00:05:69:FF:FF:FF**
- **00:0c:29:00:00:00 ~ 00:0c:29:FF:FF:FF**

- 00:1c:14:00:00:00 ~ 00:1c:14:FF:FF:FF
- 00:50:56:00:00:00 ~ 00:50:56:FF:FF:FF

VMware OUI 외부의 MAC 주소를 사용하면 클러스터 설치가 실패합니다.

사용자 프로비저닝 인프라에 대한 NTP 구성

OpenShift Container Platform 클러스터는 기본적으로 공용 NTP(Network Time Protocol) 서버를 사용하도록 구성되어 있습니다. 로컬 엔터프라이즈 NTP 서버를 사용하거나 클러스터가 연결이 끊긴 네트워크에 배포되는 경우 특정 시간 서버를 사용하도록 클러스터를 구성할 수 있습니다. 자세한 내용은 *chrony 타임 서비스 설정* 문서를 참조하십시오.

DHCP 서버가 NTP 서버 정보를 제공하는 경우 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템의 chrony 타임 서비스에서 정보를 읽고 NTP 서버와 클럭을 동기화할 수 있습니다.

추가 리소스

- chrony 타임 서비스 설정

17.5.4.5. 사용자 프로비저닝 DNS 요구사항

OpenShift Container Platform 배포의 경우 다음 구성 요소에 DNS 이름을 확인해야 합니다.

- Kubernetes API
- OpenShift Container Platform 애플리케이션 와일드카드
- 부트스트랩, 컨트롤 플레인 및 컴퓨팅 시스템

Kubernetes API, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템에 대한 역방향 DNS 확인이 필요합니다.

DNS A/AAAA 또는 CNAME 레코드는 이름 확인에 사용되며 PTR 레코드는 역방향 이름 확인에 사용

됩니다. RHCOS (Red Hat Enterprise Linux CoreOS)는 DHCP에서 호스트 이름을 제공하지 않는 한 모든 노드의 호스트 이름을 설정할 때 역방향 레코드를 사용하기 때문에 역방향 레코드가 중요합니다. 또한 역방향 레코드는 OpenShift Container Platform이 작동하는 데 필요한 인증서 서명 요청 (CSR)을 생성하는 데 사용됩니다.

참고

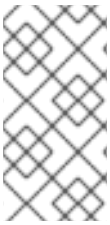
DHCP 서버를 사용하여 각 클러스터 노드에 호스트 이름을 제공하는 것이 좋습니다. 자세한 내용은 *사용자 프로비저닝 인프라 섹션에 대한 DHCP 권장 사항* 섹션을 참조하십시오.

사용자가 프로비저닝한 OpenShift Container Platform 클러스터에 대해 다음 DNS 레코드가 필요하며 설치 전에 있어야 합니다. 각 레코드에서 <cluster_name>은 클러스터 이름이고 <base_domain>은 install-config.yaml 파일에서 지정하는 기반 도메인입니다. 전체 DNS 레코드는 <component>.<cluster_name>.<base_domain> 형식입니다.

표 17.44. 필수 DNS 레코드

구성 요소	레코드	설명
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API 로드 밸런서를 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.
	api-int.<cluster_name>.<base_domain>.	내부적으로 API 로드 밸런서를 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.  **중요** API 서버는 Kubernetes에 기록된 호스트 이름으로 작업자 노드를 확인할 수 있어야 합니다. API 서버가 노드 이름을 확인할 수 없는 경우 프록시된 API 호출이 실패할 수 있으며 pod에서 로그를 검색할 수 없습니다.
라우트	***.apps.<cluster_name>.<base_domain>.**	애플리케이션 인그레스 로드 밸런서를 참조하는 와일드카드 DNS A/AAA 또는 CNAME 레코드입니다. 애플리케이션 인그레스 로드 밸런서는 Ingress 컨트롤러 Pod를 실행하는 머신을 대상으로 합니다. Ingress 컨트롤러 Pod는 기본적으로 컴퓨팅 머신에서 실행됩니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다. 예를 들어 **console-openshift-console.apps.<cluster_name>.<base_domain>**은 OpenShift Container Platform 콘솔의 와일드카드 경로로 사용됩니다.

구성 요소	레코드	설명
부트스트랩 시스템	**bootstrap.<cluster_name>.<base_domain>.**	부트스트랩 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.
컨트롤 플레인 머신	**<master><n>.<cluster_name>.<base_domain>.**	컨트롤 플레인 노드의 각 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.
컴퓨팅 머신	**<worker><n>.<cluster_name>.<base_domain>.**	작업자 노드의 각 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.



참고

OpenShift Container Platform 4.4 이상에서는 DNS 구성에서 etcd 호스트 및 SRV 레코드를 지정할 필요가 없습니다.

작은 정보

dig 명령을 사용하여 이름과 역방향 이름을 확인할 수 있습니다. 자세한 검증 단계는 *사용자 프로비저닝 인프라의 DNS 확인* 섹션을 참조하십시오.

17.5.4.5.1. 사용자 프로비저닝 클러스터의 DNS 구성 예

이 섹션에서는 사용자 프로비저닝 인프라에 OpenShift Container Platform을 배포하기 위한 DNS 요구 사항을 충족하는 A 및 PTR 레코드 구성 샘플을 제공합니다. 샘플은 하나의 DNS 솔루션을 선택하기 위한 조언을 제공하기 위한 것이 아닙니다.

이 예제에서 클러스터 이름은 ocp4이고 기본 도메인은 example.com입니다.

사용자 프로비저닝 클러스터의 DNS A 레코드 구성 예

다음 BIND 영역 파일의 예제에서는 사용자가 프로비저닝한 클러스터의 이름 확인을 위한 샘플 A 레코드를 보여줍니다.

예 17.10. 샘플 DNS 영역 데이터베이스

```
$TTL 1W
@ IN SOA ns1.example.com. root (
```

```
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
master0.ocp4.example.com. IN A 192.168.1.97 5
master1.ocp4.example.com. IN A 192.168.1.98 6
master2.ocp4.example.com. IN A 192.168.1.99 7
;
worker0.ocp4.example.com. IN A 192.168.1.11 8
worker1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1

Kubernetes API의 이름 확인을 제공합니다. 레코드는 **API** 로드 밸런서의 **IP** 주소를 나타냅니다.

2

Kubernetes API의 이름 확인을 제공합니다. 레코드는 **API** 로드 밸런서의 **IP** 주소를 참조하며 내부 클러스터 통신에 사용됩니다.

3

와일드카드 경로의 이름 확인을 제공합니다. 레코드는 애플리케이션 인그레스 로드 밸런서의 **IP** 주소를 나타냅니다. 애플리케이션 인그레스 로드 밸런서는 **Ingress** 컨트롤러 **Pod**를 실행하는 머신을 대상으로 합니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

참고

이 예제에서는 **Kubernetes API** 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 **API** 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

4

부트스트랩 시스템의 이름 확인을 제공합니다.

5 6 7

컨트롤 플레인 시스템의 이름 확인을 제공합니다.

8 9

컴퓨팅 시스템의 이름 확인을 제공합니다.

사용자 프로비저닝 클러스터의 **DNS PTR** 레코드 구성 예

다음 예제 **BIND** 영역 파일은 사용자 프로비저닝 클러스터의 역방향 이름 확인을 위한 샘플 **PTR** 레코드를 보여줍니다.

예 17.11. 역방향 레코드의 샘플 DNS 영역 데이터베이스

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR master0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR master1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR master2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 7
```

```
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com. 8
;
;EOF
```

1

Kubernetes API의 역방향 DNS 확인을 제공합니다. PTR 레코드는 API 로드 밸런서의 레코드 이름을 참조합니다.

2

Kubernetes API의 역방향 DNS 확인을 제공합니다. PTR 레코드는 API 로드 밸런서의 레코드 이름을 참조하며 내부 클러스터 통신에 사용됩니다.

3

부트스트랩 시스템의 역방향 DNS 확인을 제공합니다.

4 5 6

컨트롤 플레인 시스템의 역방향 DNS 확인을 제공합니다.

7 8

컴퓨팅 시스템의 역방향 DNS 확인을 제공합니다.

참고

OpenShift Container Platform 애플리케이션 와일드카드에는 PTR 레코드가 필요하지 않습니다.

17.5.4.6. 사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구사항

OpenShift Container Platform을 설치하기 전에 API 및 애플리케이션 인그레스 로드 밸런싱 인프라를 프로비저닝해야 합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 API 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

참고

RHEL(Red Hat Enterprise Linux) 인스턴스를 사용하여 API 및 애플리케이션 인그레스 로드 밸런서를 배포하려면 RHEL 서브스크립션을 별도로 구매해야 합니다.

로드 밸런서 인프라는 다음 요구 사항을 충족해야 합니다.

1. **API** 로드 밸런서: 플랫폼과 상호 작용하고 플랫폼을 구성할 수 있도록 사용자(인간과 시스템 모두)에게 공통 끝점을 제공합니다. 다음 조건을 설정합니다.

   - **Layer 4** 로드 밸런싱 전용입니다. 이를 **Raw TCP, SSL Passthrough** 또는 **SSL Bridge** 모드라고 합니다. **SSL Bridge** 모드를 사용하는 경우, **API** 경로에 대해 **SNI(Server Name Indication,** 서버 이름 표시)를 활성화해야 합니다.
   - 스테이트리스 로드 밸런싱 알고리즘입니다. 옵션은 로드 밸런서 구현에 따라 달라집니다.

   참고

   API 로드 밸런서가 제대로 작동하기 위해 세션 지속성이 필요하지 않습니다.

   로드 밸런서의 전면과 후면 모두에서 다음 포트를 구성하십시오.

표 17.45. API 로드 밸런서

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
6443	부트스트랩 및 컨트롤 플레인. 부트스트랩 시스템이 클러스터 컨트롤 플레인을 초기화한 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다. API 서버 상태 검사 프로브에 대한 **/readyz** 끝점을 구성해야 합니다.	X	X	Kubernetes API 서버
22623	부트스트랩 및 컨트롤 플레인. 부트스트랩 시스템이 클러스터 컨트롤 플레인을 초기화한 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다.	X		시스템 구성 서버

참고

API 서버가 /readyz 엔드포인트를 해제하는 시점부터 풀에서 API 서버 인스턴스가 제거되는 시점까지 시간이 30초를 넘지 않도록 로드 밸런서를 구성해야 합니다. /readyz가 오류를 반환하거나 정상 상태가 된 후 정해진 시간 안에 끝점이 제거 또는 추가되어야 합니다. 5초 또는 10초의 프로빙 주기(두 번의 성공적인 요청은 정상 상태, 세 번의 요청은 비정상 상태)는 충분한 테스트를 거친 값입니다.

2. 애플리케이션 인그레스 로드 밸런서: 클러스터 외부에서 유입되는 애플리케이션 트래픽에 대한 인그래스 포인트를 제공합니다. 다음 조건을 설정합니다.

- Layer 4 로드 밸런싱 전용입니다. 이를 Raw TCP, SSL Passthrough 또는 SSL Bridge 모드라고 합니다. SSL Bridge 모드를 사용하는 경우 인그레스 경로에 대해 SNI(Server Name Indication, 서버 이름 표시)를 활성화해야 합니다.

- 사용 가능한 옵션과 플랫폼에서 호스팅되는 애플리케이션 유형에 따라 연결 기반 또는 세션 기반 지속성이 권장됩니다.

작은 정보

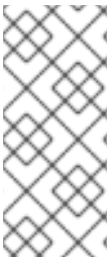
애플리케이션 인그레스 로드 밸런서에서 클라이언트의 실제 IP 주소를 확인할 수 있는 경우 소스 IP 기반 세션 지속성을 활성화하면 엔드 투 엔드 TLS 암호화를 사용하는 애플리케이션의 성능을 향상시킬 수 있습니다.

로드 밸런서의 전면과 후면 모두에서 다음 포트를 구성하십시오.

표 17.46. 애플리케이션 인그레스 로드 밸런서

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
443	기본적으로 인그레스 컨트롤러 pod, 컴퓨팅 또는 작업자를 실행하는 시스템입니다.	X	X	HTTPS 트래픽
80	기본적으로 인그레스 컨트롤러 pod, 컴퓨팅 또는 작업자를 실행하는 시스템입니다.	X	X	HTTP 트래픽

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
1936	기본적으로 Ingress 컨트롤러 Pod를 실행하는 작업자 노드입니다. 수신 상태 점검 프로브에 대해 **/healthz/ready** 끝점을 구성해야 합니다.	X	X	HTTP 트래픽



참고

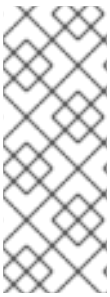
컴퓨팅 노드가 0인 3-노드 클러스터를 배포하는 경우 Ingress 컨트롤러 Pod는 컨트롤 플레인 노드에서 실행됩니다. 3-노드 클러스터 배포에서 HTTP 및 HTTPS 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다.

참고

인그레스 라우터에 대한 작업 구성이 OpenShift Container Platform 클러스터에 필요합니다. 컨트롤 플레인 초기화 후 인그레스 라우터를 설정해야 합니다.

17.5.4.6.1. 사용자 프로비저닝 클러스터의 로드 밸런서 구성 예

이 섹션에서는 사용자 프로비저닝 클러스터의 로드 밸런싱 요구 사항을 충족하는 API 및 애플리케이션 수신 로드 밸런서 구성 예를 제공합니다. 샘플은 HAProxy 로드 밸런서에 대한 **/etc/haproxy/haproxy.cfg** 구성입니다. 이 예제에서는 하나의 로드 밸런싱 솔루션을 선택하기 위한 조언을 제공하는 것을 목적으로 하지 않습니다.



참고

이 예제에서는 Kubernetes API 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 API 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

예 17.12. API 및 애플리케이션 인그레스 로드 밸런서 구성 샘플

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
```

```
    option                  redispatch
    retries                 3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s
    maxconn                 3000
  frontend stats
    bind *:1936
    mode            http
    log             global
    maxconn 10
    stats enable
    stats hide-version
    stats refresh 30s
    stats show-node
    stats show-desc Stats for ocp4 cluster (1)
    stats auth admin:ocp4
    stats uri /stats
  listen api-server-6443 (2)
    bind *:6443
    mode tcp
    server bootstrap bootstrap.ocp4.example.com:6443 check inter 1s backup (3)
    server master0 master0.ocp4.example.com:6443 check inter 1s
    server master1 master1.ocp4.example.com:6443 check inter 1s
    server master2 master2.ocp4.example.com:6443 check inter 1s
  listen machine-config-server-22623 (4)
    bind *:22623
    mode tcp
    server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup (5)
    server master0 master0.ocp4.example.com:22623 check inter 1s
    server master1 master1.ocp4.example.com:22623 check inter 1s
    server master2 master2.ocp4.example.com:22623 check inter 1s
  listen ingress-router-443 (6)
    bind *:443
    mode tcp
    balance source
    server worker0 worker0.ocp4.example.com:443 check inter 1s
    server worker1 worker1.ocp4.example.com:443 check inter 1s
  listen ingress-router-80 (7)
    bind *:80
    mode tcp
    balance source
    server worker0 worker0.ocp4.example.com:80 check inter 1s
    server worker1 worker1.ocp4.example.com:80 check inter 1s
```

(1)

이 예에서 클러스터 이름은 **ocp4** 입니다.

(2)

3 5

부트스트랩 항목은 **OpenShift Container Platform** 클러스터 설치 전에 있어야 하며 부트스트랩 프로세스가 완료된 후 제거해야 합니다.

4

포트 **22623**은 머신 구성 서버 트래픽을 처리하고 컨트롤 플레인 시스템을 가리킵니다.

6

포트 **443**은 **HTTPS** 트래픽을 처리하고 **Ingress** 컨트롤러 **Pod**를 실행하는 시스템을 가리킵니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

7

포트 **80**은 **HTTP** 트래픽을 처리하고 **Ingress** 컨트롤러 **Pod**를 실행하는 머신을 가리킵니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

참고

컴퓨팅 노드가 **0**인 **3-**노드 클러스터를 배포하는 경우 **Ingress** 컨트롤러 **Pod**는 컨트롤 플레인 노드에서 실행됩니다. **3-**노드 클러스터 배포에서 **HTTP** 및 **HTTPS** 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다.

작은 정보

HAProxy를 로드 밸런서로 사용하는 경우 **HAProxy** 노드에서 **netstat -nltupe**를 실행하여 **haproxy** 프로세스가 포트 **6443, 22623, 443** 및 **80**에서 수신 대기 중인지 확인할 수 있습니다.

참고

HAProxy를 로드 밸런서로 사용하고 **SELinux**가 **enforcing**으로 설정된 경우 **HAProxy** 서비스가 **setsebool -P haproxy_connect_any=1**을 실행하여 구성된 **TCP** 포트에 바인딩할 수 있는지 확인해야 합니다.

17.5.5. 사용자 프로비저닝 인프라 준비

사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치하기 전에 기본 인프라를 준비해야 합니다.

이 섹션에서는 OpenShift Container Platform 설치를 준비하기 위해 클러스터 인프라를 설정하는 데 필요한 높은 수준의 단계에 대해 자세히 설명합니다. 여기에는 클러스터 노드에 대한 IP 네트워킹 및 네트워크 연결 구성, 방화벽을 통해 필요한 포트 활성화, 필수 DNS 및 로드 밸런싱 인프라를 설정하는 작업이 포함됩니다.

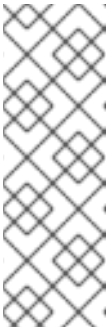
준비 후 클러스터 인프라는 *사용자 프로비저닝 인프라가 있는 클러스터의 요구 사항* 섹션에 설명된 요구 사항을 충족해야 합니다.

사전 요구 사항

- OpenShift Container Platform 4.x 테스트된 통합 페이지를 검토했습니다.
- *사용자 프로비저닝 인프라가 있는 클러스터의 요구 사항* 섹션에 자세히 설명된 인프라 요구 사항을 검토했습니다.

프로세스

1. DHCP를 사용하여 클러스터 노드에 IP 네트워킹 구성을 제공하는 경우 DHCP 서비스를 구성합니다.
   a. 노드의 영구 IP 주소를 DHCP 서버 구성에 추가합니다. 구성에서 관련 네트워크 인터페이스의 MAC 주소를 각 노드의 의도한 IP 주소와 일치시킵니다.
   b. DHCP를 사용하여 클러스터 시스템의 IP 주소 지정을 구성하는 경우 시스템은 DHCP를 통해 DNS 서버 정보도 가져옵니다. DHCP 서버 구성을 통해 클러스터 노드에서 사용하는 영구 DNS 서버 주소를 정의합니다.



참고

DHCP 서비스를 사용하지 않는 경우 RHCOS 설치 시 IP 네트워킹 구성과 DNS 서버의 주소를 노드에 제공해야 합니다. ISO 이미지에서 설치하는 경우 부팅 인수로 전달할 수 있습니다. 고정 IP 프로비저닝 *및 고급 네트워킹 옵션에 대한 자세한 내용은 RHCOS 설치 및 OpenShift Container Platform 부트스트랩 프로세스 시작* 섹션을 참조하십시오.

c.

DHCP 서버 구성에 클러스터 노드의 호스트 이름을 정의합니다. 호스트 이름 고려 사항에 대한 자세한 내용은 ***DHCP****를 통해 클러스터 노드 호스트 이름 설정* 섹션을 참조하십시오.

참고

DHCP 서비스를 사용하지 않는 경우 클러스터 노드는 역방향 **DNS** 조회를 통해 호스트 이름을 가져옵니다.

2.

네트워크 인프라가 클러스터 구성 요소 간 필수 네트워크 연결을 제공하는지 확인합니다. 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 인프라 섹션의 네트워킹 요구 사항* 섹션을 참조하십시오.

3.

OpenShift Container Platform 클러스터 구성 요소가 통신하는 데 필요한 포트를 활성화하도록 방화벽을 구성합니다. 필요한 포트에 대한 자세한 내용은 *사용자 프로비저닝 인프라 섹션의 네트워킹 요구 사항* 섹션을 참조하십시오.

4.

클러스터에 필요한 **DNS** 인프라를 설정합니다.

a.

Kubernetes API, 애플리케이션 와일드카드, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템의 **DNS** 이름 확인을 구성합니다.

b.

Kubernetes API, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템에 대한 역방향 **DNS** 확인을 구성합니다.

OpenShift Container Platform DNS 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 **DNS** 요구 사항* 섹션을 참조하십시오.

5.

DNS 구성을 확인합니다.

a.

설치 노드에서 **Kubernetes API**의 레코드 이름, 와일드카드 경로 및 클러스터 노드에 대해 **DNS** 조회를 실행합니다. 응답의 **IP** 주소가 올바른 구성 요소에 해당하는지 확인합니다.

b.

설치 노드에서 로드 밸런서 및 클러스터 노드의 **IP** 주소에 대해 역방향 **DNS** 조회를 실행합니다. 응답의 레코드 이름이 올바른 구성 요소에 해당하는지 확인합니다.

자세한 DNS 검증 단계는 *사용자 프로비저닝 인프라에 대한 DNS 확인* 섹션을 참조하십시오.

6. 필요한 API 및 애플리케이션 수신 로드 밸런싱 인프라를 프로비저닝합니다. 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구 사항* 섹션을 참조하십시오.

참고

일부 로드 밸런싱 솔루션에는 로드 밸런싱을 초기화하기 전에 클러스터 노드의 DNS 이름을 확인해야 합니다.

17.5.6. 사용자 프로비저닝 인프라에 대한 DNS 확인 검증

사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치하기 전에 DNS 구성을 확인할 수 있습니다.

중요

클러스터를 설치하기 전에 이 섹션에 설명된 검증 단계를 성공해야 합니다.

사전 요구 사항

- 사용자 프로비저닝 인프라에 필요한 DNS 레코드를 구성했습니다.

프로세스

1. 설치 노드에서 Kubernetes API의 레코드 이름, 와일드카드 경로 및 클러스터 노드에 대해 DNS 조회를 실행합니다. 응답에 포함된 IP 주소가 올바른 구성 요소에 해당하는지 확인합니다.

   a. Kubernetes API 레코드 이름을 조회합니다. 결과가 API 로드 밸런서의 IP 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
```

1 <nameserver_ip>를 네임서버의 IP 주소로, <cluster_name>을 클러스터 이름으로, <base_domain>을 기본 도메인 이름으로 바꿉니다.

출력 예

```
api.ocp4.example.com.  0 IN A 192.168.1.5
```

b. Kubernetes 내부 API 레코드 이름을 조회합니다. 결과가 API 로드 밸런서의 IP 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
```

출력 예

```
api-int.ocp4.example.com.  0 IN A 192.168.1.5
```

c. 예제 *.apps.<cluster_name>.<base_domain>을 테스트합니다. DNS 와일드카드를 조회합니다. 모든 애플리케이션 와일드카드 조회는 애플리케이션 인그레스 로드 밸런서의 IP 주소로 확인되어야 합니다.

```
$ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.
<base_domain>
```

출력 예

```
random.apps.ocp4.example.com.  0 IN A 192.168.1.5
```

참고

예제 출력에서는 **Kubernetes API** 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 **API** 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

random 항목을 다른 와일드카드 값으로 교체할 수 있습니다. 예를 들어 **OpenShift Container Platform** 콘솔의 경로를 쿼리할 수 있습니다.

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

출력 예

```
console-openshift-console.apps.ocp4.example.com. 0 IN A 192.168.1.5
```

d.

부트스트랩 **DNS** 레코드 이름에 대해 조회를 실행합니다. 결과가 부트스트랩 노드의 **IP** 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.
<base_domain>
```

출력 예

```
bootstrap.ocp4.example.com.  0 IN A 192.168.1.96
```

e.

이 방법을 사용하여 컨트롤 플레인 및 컴퓨팅 노드의 **DNS** 레코드 이름에 대해 조회를 수행합니다. 결과가 각 노드의 **IP** 주소에 해당하는지 확인합니다.

2.

설치 노드에서 로드 밸런서 및 클러스터 노드의 **IP** 주소에 대해 역방향 **DNS** 조회를 실행합니다. 응답에 포함된 레코드 이름이 올바른 구성 요소에 해당하는지 확인합니다.

a.

API 로드 밸런서의 IP 주소에 대해 역방향 조회를 수행합니다. 응답에 Kubernetes API 및 Kubernetes 내부 API의 레코드 이름이 포함되어 있는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

출력 예

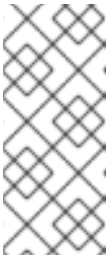
```
5.1.168.192.in-addr.arpa. 0 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 0 IN PTR api.ocp4.example.com. 2
```

1

Kubernetes 내부 API의 레코드 이름을 제공합니다.

2

Kubernetes API의 레코드 이름을 제공합니다.



참고

OpenShift Container Platform 애플리케이션 와일드카드에는 PTR 레코드가 필요하지 않습니다. 애플리케이션 인그레스 로드 밸런서의 IP 주소에 대한 역방향 DNS 확인에는 유효성 검사 단계가 필요하지 않습니다.

b.

부트스트랩 노드의 IP 주소에 대해 역방향 조회를 수행합니다. 결과가 부트스트랩 노드의 DNS 레코드 이름을 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

출력 예

```
96.1.168.192.in-addr.arpa. 0 IN PTR bootstrap.ocp4.example.com.
```

c.

이 방법을 사용하여 컨트롤 플레인 및 컴퓨팅 노드의 IP 주소에 대해 역방향 조회를 수행합니다. 결과가 각 노드의 DNS 레코드 이름과 일치하는지 확인합니다.

17.5.7. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

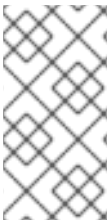
OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 core 사용자의 ~/.ssh/authorized_keys 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 core로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. ./openshift-install gather 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.



참고

AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

프로세스

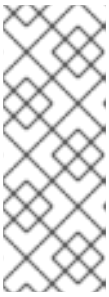
1.

로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1

새 SSH 키의 경로 및 파일 이름(예: ~/.ssh/id_ed25519)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 '~/.ssh 디렉터리에 있는지 확인하십시오.



참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 ed25519 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 rsa 또는 ecdsa 알고리즘을 사용하는 키를 생성합니다.

2.

공개 SSH 키를 확인합니다.

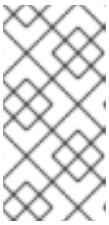
```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 ~/.ssh/id_ed25519.pub 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3.

아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 ./openshift-install gather 명령을 사용하려는 경우 필요합니다.



참고

일부 배포에서는 ~/.ssh/id_rsa 및 ~/.ssh/id_dsa와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a.

ssh-agent 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. ssh-agent에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: ~/.ssh/id_ed25519).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다. 프로비저닝하는 인프라에 클러스터를 설치하는 경우 설치 프로그램에 키를 제공해야 합니다.

17.5.8. 설치 프로그램 받기

OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

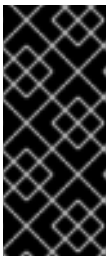
- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

프로세스

1. **OpenShift Cluster Manager** 사이트의 인프라 공급자 페이지에 액세스합니다. **Red Hat** 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.

2. 인프라 공급자를 선택합니다.

3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.



중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.

중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 **OpenShift Container Platform** 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 **Linux** 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 **OpenShift Container Platform** 구성 요소에 대한 컨테이너 이미지를 제공하는 **Quay.io**를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

17.5.9. 수동으로 설치 구성 파일 만들기

OpenShift Container Platform을 사용자가 프로비저닝한 설치의 경우 설치 구성 파일을 수동으로 생성합니다.

사전 요구 사항

- 로컬 시스템에 설치 프로그램에 제공할 **SSH** 공개 키가 있습니다. 키는 디버깅 및 재해 복구

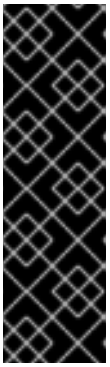
를 위해 클러스터 노드에 대한 **SSH** 인증에 사용됩니다.

- **OpenShift Container Platform** 설치 프로그램과 클러스터의 풀 시크릿이 있습니다.

프로세스

1. 필요한 설치 자산을 저장할 설치 디렉터리를 만듭니다.

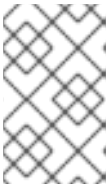
```
$ mkdir <installation_directory>
```



중요

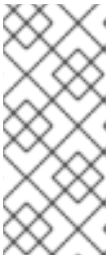
디렉터리를 만들어야 합니다. 부트스트랩 **X.509** 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 **OpenShift Container Platform** 버전에서 설치 파일을 복사할 때는 주의하십시오.

2. 샘플 **install-config.yaml** 파일 템플릿을 사용자 지정하여 **<installation_directory>**에 저장합니다.



참고

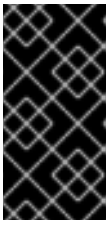
이 설정 파일의 이름을 **install-config.yaml**로 지정해야 합니다.



참고

일부 플랫폼 유형의 경우 대체로 **./openshift-install create install-config --dir <installation_directory>** 를 실행하여 **install-config.yaml** 파일을 생성할 수 있습니다. 프롬프트에서 클러스터 구성에 대한 세부 정보를 제공할 수 있습니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.



중요

install-config.yaml 파일은 설치 과정의 다음 단계에서 사용됩니다. 이 시점에서 이를 백업해야 합니다.

17.5.9.1. VMware vSphere용 샘플 install-config.yaml 파일

install-config.yaml 파일을 사용자 지정하여 **OpenShift Container Platform** 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.

```
apiVersion: v1
baseDomain: example.com 1
compute:
- hyperthreading: Enabled 2 3
  name: worker
  replicas: 0 4
controlPlane:
  hyperthreading: Enabled 5 6
  name: master
  replicas: 3 7
metadata:
  name: test 8
platform:
  vsphere:
    vcenter: your.vcenter.server 9
    username: username 10
    password: password 11
    datacenter: datacenter 12
    defaultDatastore: datastore 13
    folder: "/<datacenter_name>/vm/<folder_name>/<subfolder_name>" 14
fips: false 15
pullSecret: '{"auths": ...}' 16
sshKey: 'ssh-ed25519 AAAA...' 17
```

1

클러스터의 기본 도메인입니다. 모든 **DNS** 레코드는 이 기본 도메인의 하위 도메인이어야 하며 클러스터 이름을 포함해야 합니다.

2 5

controlPlane 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 현재 두 섹션이 모두 단일 시스템 풀을 정의하지만 향후 출시되는 **OpenShift Container Platform** 버전은 설치 과정에서 여러 컴퓨팅 풀 정의를 지원할 수 있습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

3 6

동시 멀티스레딩 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. 일부 클러스터 시스템에서 동시 멀티스레딩을 비활성화할 경우에는 해당 멀티스레딩을 모든 클러스터 시스템에서 비활성화해야 합니다.

중요

동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다. 동시 멀티스레딩을 비활성화하는 경우 시스템은 8개 이상의 **CPU**와 **32GB**의 **RAM**을 사용해야 합니다.

4

replicas 매개변수의 값을 **0**으로 설정해야 합니다. 이 매개변수는 클러스터를 생성하고 관리하는 작업자 수를 제어합니다. 이는 사용자 프로비저닝 인프라를 사용할 때 클러스터가 실행하지 않는 기능입니다. **OpenShift Container Platform** 설치를 완료하기 전에 클러스터에서 사용할 작업자 시스템을 수동으로 배포해야 합니다.

7

클러스터에 추가하는 컨트롤 플레인 시스템의 수입니다. 클러스터에서 이 값을 클러스터의 **etcd** 끝점 수로 사용하므로 이 값은 배포하는 컨트롤 플레인 시스템의 수와 일치해야 합니다.

8

DNS 레코드에 지정한 클러스터 이름입니다.

9

vCenter 서버의 정규화된 호스트 이름 또는 **IP** 주소입니다.

10

서버에 액세스하기 위한 사용자의 이름입니다. 이 사용자에게는 최소한 **vSphere**에서 정적 또는 동적 영구 볼륨 프로비저닝 에 필요한 역할과 권한이 있어야 합니다.

11

vSphere 사용자와 연결된 암호입니다.

12

vSphere 데이터 센터입니다.

13

사용할 기본 **vSphere** 데이터 저장소입니다.

14

선택사항: 설치 관리자 프로비저닝 인프라의 경우 설치 프로그램이 가상 머신을 생성하는 기존 폴더의 절대 경로(예: **/<datacenter_name>/vm/<folder_name>/<subfolder_name>**). 이 값을 제공하지 않으면 설치 프로그램이 인프라 **ID**로 이름이 지정된 데이터 센터 가상 머신 폴더에 최상위 폴더를 만듭니다. 클러스터에 인프라를 제공하는 경우에는 이 매개변수를 생략합니다.

15

FIPS 모드 활성화 또는 비활성화 여부입니다. 기본적으로 **FIPS** 모드는 비활성화됩니다. **FIPS** 모드가 활성화되면 **OpenShift Container Platform**이 실행되는 **RHCOS(Red Hat Enterprise Linux CoreOS)** 시스템에서 기본 **Kubernetes** 암호화 제품군은 우회하고 **RHCOS**와 함께 제공되는 암호화 모듈을 대신 사용합니다.

중요

FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 **OpenShift Container Platform** 배포에서만 지원됩니다.

16

OpenShift Cluster Manager 에서 얻은 풀 시크릿입니다. 이 풀 시크릿을 사용하면 **OpenShift Container Platform** 구성 요소에 대한 컨테이너 이미지를 제공하는 **Quay.io**를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

17

RHCOS(Red Hat Enterprise Linux CoreOS)의 **core** 사용자에 대한 기본 **SSH** 키의 공용 부분입니다.

17.5.9.2. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 **HTTP** 또는 **HTTPS** 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 **OpenShift Container Platform** 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.

- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 **API**에 대한 호출을 포함하여 모든 클러스터 발신(**Egress**) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.

참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 **Red Hat OpenStack Platform (RHOSP)**에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. 클러스터 외부에서 **HTTP** 연결을 구축하는 데 사용할 프록시 **URL**입니다. **URL** 스키마는 **http**여야 합니다.

2. 클러스터 외부에서 **HTTPS** 연결을 구축하는 데 사용할 프록시 **URL**입니다.

3.

4

이 값을 제공하면 설치 프로그램에서 추가 CA 인증서를 보유할 openshift -config 네임스페이스에 user-ca- bundle 이라는 구성 맵을 생성합니다. additionalTrustBundle 및 하나 이상의 프록시 설정을 제공하는 경우 프록시 오브젝트는 trustedCA 필드의 user-ca-bundle 구성 맵을 참조하도록 구성됩니다. 그러면 Cluster Network Operator에서 trustedCA 매개변수에 대해 지정된 콘텐츠를 RHCOS 신뢰 번들과 병합하는 trusted-ca-bundle 구성 맵을 생성합니다. 프록시의 ID 인증서를 RHCOS 트러스트 번들에 있는 기관에서 서명하지 않은 경우 additionalTrustBundle 필드가 있어야 합니다.

참고

설치 프로그램에서 프록시 adinessEndpoints 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 OpenShift Container Platform을 설치할 때 참조하십시오.

제공되는 install-config.yaml 파일의 프록시 설정을 사용하는 cluster라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 cluster Proxy 오브젝트는 계속 생성되지만 spec은 nil이 됩니다.

참고

cluster라는 Proxy 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

17.5.10. Kubernetes 매니페스트 및 Ignition 설정 파일 생성

일부 클러스터 정의 파일을 수정하고 클러스터 시스템을 수동으로 시작해야 하므로 클러스터가 시스템을 구성하는 데 필요한 Kubernetes 매니페스트 및 Ignition 구성 파일을 사용자가 생성해야 합니다.

설치 구성 파일은 Kubernetes 매니페스트로 변환됩니다. 매니페스트는 나중에 클러스터 머신을 구성하는 데 사용되는 Ignition 구성 파일로 래핑됩니다.

중요

- OpenShift Container Platform 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 node-bootstrapper 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램을 가져오셨습니다.
- install-config.yaml 설치 구성 파일을 생성하셨습니다.

프로세스

1. OpenShift Container Platform 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터에 대한 Kubernetes 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 <installation_directory>는 사용자가 만든 install-config.yaml 파일이 포함된 설치 디렉터리를 지정합니다.

2. 컨트롤 플레인 시스템 및 컴퓨팅 머신 세트를 정의하는 Kubernetes 매니페스트 파일을 제거합니다.

```
$ rm -f openshift/99_openshift-cluster-api_master-machines-*.yaml
openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

이러한 리소스는 사용자가 직접 생성하고 관리하기 때문에 초기화할 필요가 없습니다.

- 시스템 API로 머신 세트 파일을 보존하여 컴퓨팅 시스템을 생성할 수 있지만 사용자 환경과 일치하도록 해당 참조를 업데이트해야 합니다.

3. <installation_directory>/manifests/cluster-scheduler-02-config.yml Kubernetes 매니페스트 파일의 mastersSchedulable 매개변수가 false로 설정되어 있는지 확인합니다. 이 설정으로 인해 컨트롤 플레인 머신에서 포드가 예약되지 않습니다.

   a. <installation_directory>/manifests/cluster-scheduler-02-config.yml 파일을 엽니다.

   b. mastersSchedulable 매개변수를 찾아서 값을 False로 설정되어 있는지 확인합니다.

   c. 파일을 저장하고 종료합니다.

4. Ignition 구성 파일을 생성하려면 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행합니다.

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 <installation_directory>는 동일한 설치 디렉터리를 지정합니다.

설치 디렉터리의 부트스트랩, 컨트롤 플레인 및 컴퓨팅 노드에 대한 Ignition 구성 파일이 생성됩니다. kubeadmin-password 및 kubeconfig 파일은 ./<installation_directory>/auth 디렉터리에 생성됩니다.

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

17.5.11. 인프라 이름 추출

Ignition 구성 파일에는 VMware vSpher에서 클러스터를 고유하게 식별하는 데 사용할 수 있는 고유한 클러스터 ID가 포함되어 있습니다. 클러스터 ID를 가상 머신 폴더의 이름으로 사용하려면 해당 ID를 추출해야 합니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받으셨습니다.
- 클러스터에 대한 Ignition 구성 파일을 생성하셨습니다.
- jq CLI를 설치하셨습니다.

프로세스

- Ignition 구성 파일 메타데이터에서 인프라 이름을 추출하여 확인하려면 다음 명령을 실행하십시오.

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

출력 예

```
openshift-vw9j6 1
```

1 이 명령의 출력은 클러스터 이름과 임의의 문자열입니다.

17.5.12. RHCOS 설치 및 OpenShift Container Platform 부트스트랩 프로세스 시작

VMware vSphere의 사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치하려면 vSphere 호스트에 RHCOS(Red Hat Enterprise Linux Core OS)를 설치해야 합니다. RHCOS를 설치할 때 설치 중인 머신 유형에 대해 OpenShift Container Platform 설치 프로그램에서 생성한 Ignition 구성 파일을 제공해야 합니다. 적합한 네트워킹, DNS 및 로드 밸런싱 인프라를 구성한 경우 RHCOS 머신이 재부팅된 후 OpenShift Container Platform 부트스트랩 프로세스가 자동으로 시작됩니다.

사전 요구 사항

- 클러스터에 대한 Ignition 구성 파일이 있어야 합니다.
- 사용자 컴퓨터에서 액세스할 수 있고 생성한 시스템이 액세스할 수 있는 HTTP 서버에 액세스할 수 있습니다.
- vSphere 클러스터 를 생성했습니다.

프로세스

1. 설치 프로그램에서 생성된 부트스트랩 Ignition 구성 파일 (<installation_directory>/bootstrap.ign)을 HTTP 서버에 업로드합니다. 이 파일의 URL을 기록해 둡니다.
2. 부트스트랩 노드의 다음 보조 Ignition 구성 파일을 <installation_directory>/merge-bootstrap.ign으로 컴퓨터에 저장합니다.

```
{
  "ignition": {
    "config": {
      "merge": [
        {
          "source": "<bootstrap_ignition_config_url>", 1
          "verification": {}
        }
      ]
    },
    "timeouts": {},
    "version": "3.2.0"
  },
  "networkd": {},
  "passwd": {},
  "storage": {},
  "systemd": {}
}
```

1

호스팅한 부트스트랩 Ignition 구성 파일의 URL을 지정합니다.

부트스트랩 머신에 대한 VM(가상 머신)을 생성할 때 이 Ignition 구성 파일을 사용합니다.

3. 설치 프로그램이 생성한 다음 Ignition 구성 파일을 찾습니다.

- <installation_directory>/master.ign
- <installation_directory>/worker.ign
- <installation_directory>/merge-bootstrap.ign

4. Ignition 구성 파일을 Base64 인코딩으로 변환합니다. 이 절차의 뒷부분에서는 이러한 파일을 VM의 추가 구성 매개변수 guestinfo.ignition.config.data 에 추가해야 합니다.

예를 들어 Linux 운영 체제를 사용하는 경우 base64 명령을 사용하여 파일을 인코딩할 수 있습니다.

```
$ base64 -w0 <installation_directory>/master.ign > <installation_directory>/master.64
```

```
$ base64 -w0 <installation_directory>/worker.ign > <installation_directory>/worker.64
```

```
$ base64 -w0 <installation_directory>/merge-bootstrap.ign >
<installation_directory>/merge-bootstrap.64
```

중요

설치를 마친 후 클러스터에 컴퓨팅 시스템을 더 추가하려면 Ignition 구성 파일을 삭제하지 마십시오.

5. RHCOS OVA 이미지를 가져옵니다. RHCOS 이미지 미러 페이지에서 이미지를 사용할 수 있습니다.

중요

RHCOS 이미지는 OpenShift Container Platform 릴리스에 따라 변경되지 않을 수 있습니다. 설치하는 OpenShift Container Platform 버전과 같거나 그 이하의 버전 중 가장 최신 버전의 이미지를 다운로드해야 합니다. 지원되는 경우 OpenShift Container Platform 버전과 일치하는 이미지 버전을 사용합니다.

파일 이름에는 rhcos-vmware.<architecture>.ova 형식의 OpenShift Container Platform 버전 번호가 포함됩니다.

6. vSphere Client에서 VM을 저장할 데이터 센터 폴더를 생성합니다.

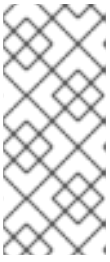
   a. VMs and Templates 보기를 클릭합니다.

   b. 데이터 센터 이름을 마우스 오른쪽 버튼으로 클릭합니다.

   c. New Folder → New VM and Template Folder를 클릭합니다.

   d. 표시되는 창에서 폴더 이름을 입력합니다. install-config.yaml 파일에서 기존 폴더를 지정하지 않은 경우 인프라 ID와 동일한 이름으로 폴더를 생성합니다. vCenter는 이 폴더 이름을 사용하여 Workspace 구성에 적절한 위치에 스토리지를 동적으로 프로비저닝합니다.

7. vSphere Client에서 OVA 이미지에 대한 템플릿을 생성한 다음 필요에 따라 템플릿을 복제합니다.



참고

다음 단계에서 템플릿을 생성한 다음 모든 클러스터 시스템에 대한 템플릿을 복제합니다. 그런 다음 VM을 프로비저닝할 때 복제된 머신 유형의 Ignition 구성 파일의 위치를 제공합니다.

   a. Hosts and Clusters 탭에서 클러스터 이름을 마우스 오른쪽 버튼으로 클릭하고 Deploy OVF Template을 선택합니다.

   .

b. **Select an OVF** 탭에서 다운로드한 **RHCOS OVA** 파일의 이름을 지정합니다.

c. 이름 및 폴더 선택 탭에서 템플릿의 가상 시스템 이름 (예: **Template-RHCOS**)을 설정합니다. **vSphere** 클러스터의 이름을 클릭하고 이전 단계에서 생성한 폴더를 선택합니다.

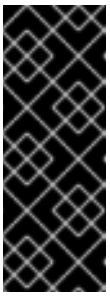
d. **Select a compute resource** 탭에서 **vSphere** 클러스터 이름을 클릭합니다.

e. **Select storage** 탭에서 **VM**의 스토리지 옵션을 구성합니다.

- 스토리지 기본 설정에 따라 **Thin Provision** 또는 **Thick Provision**을 선택합니다.
- **install-config.yaml** 파일에서 지정한 데이터 저장소를 선택합니다.

f. **Select network** 탭에서 사용 가능한 경우 클러스터에 대해 구성한 네트워크를 지정합니다.

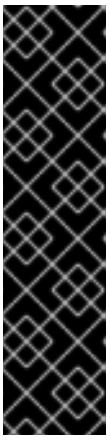
g. **OVF** 템플릿을 생성할 때 템플릿 사용자 지정 탭에 값을 지정하지 않거나 템플릿을 추가로 구성하지 마십시오.



중요

원래 **VM** 템플릿을 시작하지 마십시오. **VM** 템플릿이 꺼져 있어야 하며 새 **RHCOS** 머신에 대해 복제해야 합니다. **VM** 템플릿을 시작하면 **VM** 템플릿이 플랫폼의 **VM**으로 구성되므로 시스템 세트가 구성을 적용할 수 있는 템플릿으로 사용되지 않습니다.

8. 선택 사항: 필요한 경우 **VM** 템플릿에서 구성된 가상 하드웨어 버전을 업데이트합니다. 자세한 내용은 가상 머신을 **VMware** 설명서의 최신 하드웨어 버전으로 업그레이드를 참조하십시오.



중요

필요한 경우 VM 템플릿의 하드웨어 버전을 버전 15로 업데이트하는 것이 좋습니다. 이제 vSphere에서 실행 중인 클러스터 노드에 하드웨어 버전 13을 사용하는 것이 더 이상 사용되지 않습니다. 가져온 템플릿의 기본값이 하드웨어 버전 13인 경우 VM 템플릿을 하드웨어 버전 15로 업그레이드하기 전에 ESXi 호스트가 6.7U3 이상인지 확인해야 합니다. vSphere 버전이 6.7U3 미만인 경우 이 업그레이드 단계를 건너뛸 수 있습니다. 그러나 향후 OpenShift Container Platform 버전은 6.7U3 미만의 하드웨어 버전 13 및 vSphere 버전에 대한 지원을 제거하도록 예약되어 있습니다.

9. 템플릿이 배포된 후 클러스터에서 시스템의 가상 머신을 배포합니다.

a. 템플릿 이름을 마우스 오른쪽 버튼으로 클릭하고 Clone → Clone to Virtual Machine을 클릭합니다.

b. Select a name and folder 탭에서 가상 머신의 이름을 지정합니다. control-plane-0 또는 compute-1과 같은 시스템 유형을 이름에 포함할 수 있습니다.

c. Select a name and folder 탭에서 클러스터에 대해 생성한 폴더의 이름을 선택합니다.

d. Select a compute resource 탭에서 데이터 센터의 호스트 이름을 선택합니다.

e. 선택사항: Select storage 탭에서 스토리지 옵션을 사용자 지정합니다.

f. Select clone options에서 Customize this virtual machine's hardware를 선택합니다.

g. Customize hardware 탭에서 VM Options → Advanced를 클릭합니다.

- 선택 사항: vSphere에서 기본 DHCP 네트워킹을 재정의합니다. 고정 IP 네트워킹을 활성화하려면 다음을 수행합니다.

  i. 고정 IP 구성을 설정합니다.

```
$ export IPCFG="ip=<ip>::<gateway>:<netmask>:<hostname>:
<iface>:none nameserver=srv1 [nameserver=srv2 [nameserver=srv3 [...]]]"
```

명령 예

```
$ export IPCFG="ip=192.168.100.101::192.168.100.254:255.255.255.0:::none
nameserver=8.8.8.8"
```

ii. guestinfo.afterburn.initrd.network-kargs 속성을 설정한 후 vSphere의 OVA에서 VM을 부팅합니다.

```
$ govc vm.change -vm "<vm_name>" -e
"guestinfo.afterburn.initrd.network-kargs=${IPCFG}"
```

- 선택사항: 클러스터 성능 문제가 발생하는 경우 Latency Sensitivity 목록에서 High를 선택합니다. VM의 CPU 및 메모리 예약에 다음과 같은 값이 있는지 확인합니다.
  - 메모리 예약 값은 구성된 메모리 크기와 같아야 합니다.
  - CPU 예약 값은 측정된 물리적 CPU 속도를 곱한 최소 대기 시간이 짧은 가상 CPU 수여야 합니다.
- 구성 편집을 클릭하고 Configuration Parameters 창에서 사용 가능한 매개 변수 목록을 검색하여 스틸 클럭 회계 (stealclock.enable)를 검색합니다. 사용 가능한 경우 해당 값을 TRUE 로 설정합니다. 스틸링 클럭 계정을 사용하면 클러스터 문제 해결에 도움이 될 수 있습니다.
- 구성 추가 매개변수를 클릭합니다. 다음 매개변수 이름 및 값을 정의합니다.
  - guestinfo.ignition.config.data: 이 절차에서 이전에 생성한 base-64 인코딩 파일을 찾고 이 머신 유형에 대해 base64로 인코딩된 Ignition 구성 파일의 내용을 붙여넣습니다.
  -

- guestinfo.ignition.config.data.encoding: base64를 지정합니다.
- disk.EnableUUID: TRUE를 지정합니다.
- stealclock.enable:이 매개 변수가 정의되지 않은 경우 추가하고 TRUE 를 지정합니다.

h. Customize hardware 탭의 Virtual Hardware 패널에서 지정된 값을 필요에 따라 수정합니다. RAM, CPU 및 디스크 스토리지의 양이 시스템 유형에 대한 최소 요구사항을 충족하는지 확인합니다.

i. 구성을 완료하고 VM의 전원을 켭니다.

10. 각 시스템에 대해 이전 단계에 따라 클러스터의 나머지 시스템을 생성합니다.

중요

이때 부트스트랩 및 컨트롤 플레인 시스템을 생성해야 합니다. 일부 Pod는 기본적으로 컴퓨팅 시스템에 배포되므로 클러스터를 설치하기 전에 컴퓨팅 시스템을 두 개 이상 생성합니다.

17.5.13. vSphere의 클러스터에 더 많은 컴퓨팅 머신 추가

VMware vSphere에서 사용자가 프로비저닝한 OpenShift Container Platform 클러스터에 더 많은 컴퓨팅 머신을 추가할 수 있습니다.

사전 요구 사항

- 컴퓨팅 머신의 base64로 인코딩된 Ignition 파일을 가져옵니다.
- 클러스터에 생성한 vSphere 템플릿에 액세스할 수 있습니다.

절차

1. 템플릿이 배포된 후 클러스터에서 시스템의 가상 머신을 배포합니다.

   a. 템플릿 이름을 마우스 오른쪽 버튼으로 클릭하고 **Clone → Clone to Virtual Machine**을 클릭합니다.

   b. **Select a name and folder** 탭에서 가상 머신의 이름을 지정합니다. **compute-1**과 같은 머신 유형을 이름에 포함할 수 있습니다.

   c. **Select a name and folder** 탭에서 클러스터에 대해 생성한 폴더의 이름을 선택합니다.

   d. **Select a compute resource** 탭에서 데이터 센터의 호스트 이름을 선택합니다.

   e. 선택사항: **Select storage** 탭에서 스토리지 옵션을 사용자 지정합니다.

   f. **Select clone options**에서 **Customize this virtual machine's hardware**를 선택합니다.

   g. **Customize hardware** 탭에서 **VM Options → Advanced**를 클릭합니다.

      - **Latency Sensitivity** 목록에서 **High**를 선택합니다.
      - **Edit Configuration**을 클릭하고 **Configuration Parameters** 창에서 **Add Configuration Params**를 클릭합니다. 다음 매개변수 이름 및 값을 정의합니다.
        - **guestinfo.ignition.config.data:** 이 머신 유형에 대해 **base64**로 인코딩된 컴퓨팅 **Ignition** 구성 파일의 내용을 붙여넣습니다.
        - **guestinfo.ignition.config.data.encoding: base64**를 지정합니다.
        - **disk.EnableUUID: TRUE**를 지정합니다.

   h.

Customize hardware 탭의 **Virtual Hardware** 패널에서 지정된 값을 필요에 따라 수정합니다. **RAM, CPU** 및 디스크 스토리지의 양이 시스템 유형에 대한 최소 요구사항을 충족하는지 확인합니다. 또한 사용 가능한 네트워크가 여러 개인 경우 **Add network adapter**에서 올바른 네트워크를 선택해야 합니다.

i. 구성을 완료하고 **VM**의 전원을 켭니다.

2. 계속해서 클러스터에 추가 컴퓨팅 머신을 만듭니다.

17.5.14. 디스크 파티션 설정

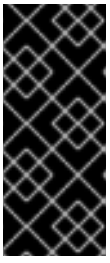
대부분의 경우 데이터 파티션은 원래 다른 운영 체제를 설치하는 대신 **RHCOS**를 설치하여 생성됩니다. 이러한 경우 **OpenShift Container Platform** 설치 프로그램은 디스크 파티션을 설정할 수 있어야합니다.

그러나 **OpenShift Container Platform** 노드를 설치할 때 기본 파티션 설정을 덮어 쓰기하기 위해 개입이 필요한 두 가지 경우가 있습니다.

- 별도의 파티션 생성: 빈 디스크에 그린 필드 설치의 경우 파티션에 별도의 스토리지를 추가할 수 있습니다. 이는 **/var** 또는 **/var**의 하위 디렉터리 (예: **/var/lib/etcd**)중 하나를 별도의 파티션으로 만드는 경우에 공식적으로 지원됩니다.

중요

100GB 이상의 디스크 크기 및 특히 **1TB**보다 큰 디스크 크기의 경우 별도의 **/var** 파티션을 만듭니다. 자세한 내용은 " 별도의 **/var** 파티션 생성" 및 이 Red Hat 지식베이스 문서 를 참조하십시오.



중요

Kubernetes는 두 개의 파일 시스템 파티션만 지원합니다. 원래 구성에 둘 이상의 파티션을 추가하는 경우 **Kubernetes**는 모든 파티션을 모니터링할 수 없습니다.

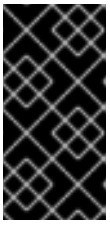
- 기존 파티션 유지: 브라운 필드 설치에서 기존 노드에 **OpenShift Container Platform**을 재설치하고 이전 운영 체제에 설치된 데이터 파티션을 유지해야 하는 경우 기존 데이터 파티션을 유지할 수 있는 **coreos-installer**에 부팅 인수와 옵션이 모두 있습니다.

별도의 /var 파티션 만들기

일반적으로 OpenShift Container Platform의 디스크 파티션 설정은 설치 프로그램에 맡겨야합니다. 그러나 확장하려는 파일 시스템의 일부에 별도의 파티션을 생성해야 하는 경우가 있습니다.

OpenShift 컨테이너 플랫폼은 /var 파티션 또는 /var의 하위 디렉터리 중 하나에 스토리지를 연결하는 단일 파티션의 추가를 지원합니다. 예를 들면 다음과 같습니다.

- /var/lib/containers: 시스템에 더 많은 이미지와 컨테이너가 추가됨에 따라 확장될 수 있는 컨테이너 관련 콘텐츠를 보관합니다.
- /var/lib/etcd: etcd 스토리지의 성능 최적화와 같은 목적으로 별도로 보관할 데이터를 보관합니다.
- /var: 감사 등의 목적에 맞게 별도로 분리하여 보관해야 하는 데이터를 보관합니다.



중요

100GB보다 큰 디스크 크기 및 특히 1TB보다 큰 디스크의 경우 별도의 /var 파티션을 만듭니다.

/var 디렉터리의 콘텐츠를 별도로 저장하면 필요에 따라 해당 영역에 대한 스토리지 확장을 보다 용이하게 하고 나중에 OpenShift Container Platform을 다시 설치하여 해당 데이터를 그대로 보존할 수 있습니다. 이 방법을 사용하면 모든 컨테이너를 다시 가져올 필요가 없으며 시스템을 업데이트할 때 대용량 로그 파일을 복사할 필요도 없습니다.

RHCOS(Red Hat Enterprise Linux CoreOS)를 새로 설치하기 전에 /var가 있어야 하므로 다음 절차에서는 OpenShift Container Platform 설치의 openshift-install 준비 단계 중에 삽입되는 머신 구성 매니페스트를 생성하여 별도의 /var 파티션을 설정합니다.

프로세스

1. OpenShift Container Platform 설치 파일을 저장할 디렉터리를 만듭니다.

```
$ mkdir $HOME/clusterconfig
```

2. openshift-install을 실행하여 manifest 및 openshift 하위 디렉터리에 파일 세트를 만듭니

다. 프롬프트가 표시되면 시스템 질문에 대답합니다.

```
$ openshift-install create manifests --dir $HOME/clusterconfig
? SSH Public Key ...
$ ls $HOME/clusterconfig/openshift/
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

3. 추가 파티션을 구성하는 Butane 구성을 생성합니다. 예를 들어 파일 이름을 $HOME/clusterconfig/98-var-partition.bu 로 지정하고, 디스크 장치 이름을 worker 시스템의 스토리지 장치 이름으로 변경하고 스토리지 크기를 적절하게 설정합니다. 이 예에서는 /var 디렉토리를 별도의 파티션에 배치합니다.

```
variant: openshift
version: 4.9.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1 파티션을 설정해야하는 디스크 저장 장치 이름입니다.

2 데이터 파티션을 부트 디스크에 추가할 때 최소 25000 메비 바이트가 권장됩니다. 루트 파일 시스템은 지정된 오프셋까지 사용 가능한 모든 공간을 채우기 위해 자동으로 크기가 조정됩니다. 값이 지정되지 않거나 지정된 값이 권장 최소값보다 작으면 생성되는 루트 파일 시스템의 크기가 너무 작아지고 RHCOS를 나중에 다시 설치할 때 데이터 파티션의 첫 번째 부분을 덮어 쓸 수 있습니다.

3

데이터 파티션의 크기(MB)입니다.

4

컨테이너 스토리지에 사용되는 파일 시스템에 prjquota 마운트 옵션을 활성화해야 합니다.

참고

별도의 /var 파티션을 만들 때 다른 인스턴스 유형에 동일한 장치 이름이 없는 경우 작업자 노드에 다른 인스턴스 유형을 사용할 수 없습니다.

4. Butane 구성에서 매니페스트를 생성하여 clusterconfig/openshift 디렉터리에 저장합니다. 예를 들어 다음 명령을 실행합니다.

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o
$HOME/clusterconfig/openshift/98-var-partition.yaml
```

5. openshift-install을 다시 실행하여 manifest 및 openshift 하위 디렉터리의 파일 세트에서 Ignition 구성을 만듭니다.

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

이제 Ignition 구성 파일을 vSphere 설치 절차에 대한 입력으로 사용하여 RHCOS (Red Hat Enterprise Linux CoreOS) 시스템을 설치할 수 있습니다.

17.5.15. bootupd를 사용하여 부트로더 업데이트

bootupd를 사용하여 부트로더를 업데이트하려면 RHCOS 머신에 수동으로 bootupd를 설치하거나 systemd 유닛이 활성화된 머신 구성을 제공해야 합니다. grubby 또는 기타 부트로더 툴과는 달리 bootupd는 커널 인수 전달과 같은 커널 공간 구성을 관리하지 않습니다.

bootupd를 설치한 후 OpenShift Container Platform 클러스터에서 원격으로 관리할 수 있습니다.

참고

bootupd는 **BootHole** 취약점에 대한 보호와 같이 베어 메탈 또는 가상화 하이퍼바이저 설치에서만 사용하는 것이 좋습니다.

수동 설치 방법

bootctl 명령줄 툴을 사용하여 **bootupd**를 수동으로 설치할 수 있습니다.

1. 시스템 상태를 검사합니다.

```
# bootupctl status
```

출력 예

```
Component EFI
  Installed: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
  Update: At latest version
```

2. **bootupd**를 설치하지 않고 생성된 **RHCOS** 이미지에는 명시적 채택 단계가 필요합니다.

   시스템 상태가 **Adoptable**인 경우 채택을 수행합니다.

```
# bootupctl adopt-and-update
```

출력 예

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

3. 업데이트를 사용할 수 있는 경우 다음 재부팅에 변경 사항이 적용되도록 업데이트를 적용합니다.

```
# bootupctl update
```

출력 예

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

시스템 구성 방법

bootupd를 활성화하는 또 다른 방법은 머신 구성을 제공하는 것입니다.

- 다음 예와 같이 systemd 단위가 활성화된 머신 구성 파일을 제공합니다.

출력 예

```
variant: rhcos
version: 1.1.0
systemd:
  units:
    - name: custom-bootupd-auto.service
      enabled: true
      contents: |
        [Unit]
        Description=Bootupd automatic update

        [Service]
        ExecStart=/usr/bin/bootupctl update
        RemainAfterExit=yes

        [Install]
        WantedBy=multi-user.target
```

17.5.16. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift

CLI(oc)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 oc를 설치할 수 있습니다.

중요

이전 버전의 oc를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 oc를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 Linux에서 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

절차

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.
2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. OpenShift v4.9 Linux Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. oc 바이너리를 PATH에 있는 디렉터리에 배치합니다.

   PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치

다음 절차에 따라 Windows에 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

절차

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.

2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.

3. OpenShift v4.9 Windows Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.

4. ZIP 프로그램으로 아카이브의 압축을 풉니다.

5. oc 바이너리를 PATH에 있는 디렉터리로 이동합니다.

   PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

   ```
   C:\> path
   ```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치

다음 절차에 따라 macOS에서 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

절차

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.

2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.

3. OpenShift v4.9 MacOSX Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.

4.

4. 아카이브의 압축을 해제하고 압축을 풉니다.

5. **oc** 바이너리 **PATH**의 디렉터리로 이동합니다.

   PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

   ```
   $ echo $PATH
   ```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

17.5.17. 부트스트랩 프로세스가 완료될 때까지 대기 중

OpenShift Container Platform 부트스트랩 프로세스는 클러스터 노드가 먼저 디스크에 설치된 영구 **RHCOS** 환경으로 부팅된 후에 시작됩니다. **Ignition** 구성 파일을 통해 제공되는 구성 정보는 부트스트랩 프로세스를 초기화하고 머신에 **OpenShift Container Platform**을 설치하는 데 사용됩니다. 부트스트랩 프로세스가 완료될 때까지 기다려야 합니다.

사전 요구 사항

- 클러스터에 대한 **Ignition** 구성 파일이 생성되어 있습니다.
- 적합한 네트워크, **DNS** 및 로드 밸런싱 인프라가 구성되어 있습니다.
- 설치 프로그램을 받아서 클러스터의 **Ignition** 구성 파일을 생성했습니다.
- 클러스터 머신에 **RHCOS**를 설치하고 **OpenShift Container Platform** 설치 프로그램에서 생성된 **Ignition** 구성 파일을 제공했습니다.
- 사용자 시스템에서 직접 인터넷에 액세스하거나 **HTTP** 또는 **HTTPS** 프록시를 사용할 수 있습니다.

절차

1. 부트스트랩 프로세스를 모니터링합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
  --log-level=info 2
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

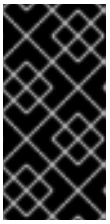
2 다른 설치 세부 사항을 보려면 info 대신 warn, debug 또는 error를 지정합니다.

출력 예

```
INFO Waiting up to 30m0s for the Kubernetes API at
https://api.test.example.com:6443...
INFO API v1.22.1 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

이 명령은 Kubernetes API 서버가 컨트롤 플레인 시스템에서 부트스트랩되었다는 신호를 보낼 때 성공합니다.

2. 부트스트랩 프로세스가 완료된 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다.



중요

이 시점에 로드 밸런서에서 부트스트랩 시스템을 제거해야 합니다. 부트스트랩 머신 자체를 제거하거나 다시 포맷할 수도 있습니다.

17.5.18. CLI를 사용하여 클러스터에 로그인

클러스터 kubeconfig 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. kubeconfig 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러

스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- oc CLI를 설치했습니다.

절차

1. kubeadmin 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 oc 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

17.5.19. 머신의 인증서 서명 요청 승인

클러스터에 시스템을 추가하면 추가한 시스템별로 보류 중인 인증서 서명 요청(CSR)이 두 개씩 생성됩니다. 이러한 CSR이 승인되었는지 확인해야 하며, 필요한 경우 이를 직접 승인해야 합니다. 클라이언트 요청을 먼저 승인한 다음 서버 요청을 승인해야 합니다.

사전 요구 사항

- 클러스터에 시스템을 추가했습니다.

프로세스

1. 클러스터가 시스템을 인식하는지 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0  Ready    master  63m  v1.22.1
master-1  Ready    master  63m  v1.22.1
master-2  Ready    master  64m  v1.22.1
```

출력에 생성된 모든 시스템이 나열됩니다.

참고

이전 출력에는 일부 **CSR**이 승인될 때까지 컴퓨팅 노드(작업자 노드라고도 함)가 포함되지 않을 수 있습니다.

2. 보류 중인 **CSR**을 검토하고 클러스터에 추가한 각 시스템에 대해 **Pending** 또는 **Approved** 상태의 클라이언트 및 서버 요청이 표시되는지 확인합니다.

```
$ oc get csr
```

출력 예

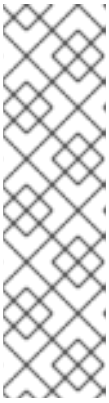
```
NAME        AGE    REQUESTOR                                                   CONDITION
csr-8b2br   15m    system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
```

```
csr-8vnps   15m    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

예에서는 두 시스템이 클러스터에 참여하고 있습니다. 목록에는 승인된 CSR이 더 많이 나타날 수도 있습니다.

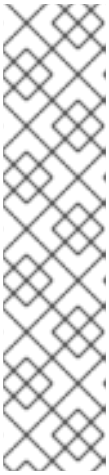
3.

CSR이 승인되지 않은 경우, 추가된 시스템에 대한 모든 보류 중인 CSR이 Pending 상태로 전환된 후 클러스터 시스템의 CSR을 승인합니다.



참고

CSR은 교체 주기가 자동으로 만료되므로 클러스터에 시스템을 추가한 후 1 시간 이내에 CSR을 승인하십시오. 한 시간 내에 승인하지 않으면 인증서가 교체되고 각 노드에 대해 두 개 이상의 인증서가 표시됩니다. 이러한 인증서를 모두 승인해야 합니다. 클라이언트 CSR이 승인되면 Kubelet은 인증서에 대한 보조 CSR을 생성하므로 수동 승인이 필요합니다. 그러면 Kubelet에서 동일한 매개변수를 사용하여 새 인증서를 요청하는 경우 인증서 갱신 요청은 machine-approver에 의해 자동으로 승인됩니다.



참고

베어 메탈 및 기타 사용자 프로비저닝 인프라와 같이 머신 API를 사용하도록 활성화되지 않는 플랫폼에서 실행되는 클러스터의 경우 CSR(Kubelet service Certificate Request)을 자동으로 승인하는 방법을 구현해야 합니다. 요청이 승인되지 않으면 API 서버가 kubelet에 연결될 때 서비스 인증서가 필요하므로 oc exec, oc rsh, oc logs 명령을 성공적으로 수행할 수 없습니다. Kubelet 엔드 포인트에 연결하는 모든 작업을 수행하려면 이 인증서 승인이 필요합니다. 이 방법은 새 CSR을 감시하고 CSR이 system:node 또는 system:admin 그룹의 node-bootstrapper 서비스 계정에 의해 제출되었는지 확인하고 노드의 ID를 확인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

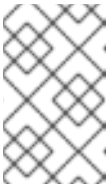
```
$ oc adm certificate approve <csr_name> 1
```

1

<csr_name>은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```



참고

일부 Operator는 일부 CSR이 승인될 때까지 사용할 수 없습니다.

4. 이제 클라이언트 요청이 승인되었으므로 클러스터에 추가한 각 머신의 서버 요청을 검토해야 합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 나머지 CSR이 승인되지 않고 Pending 상태인 경우 클러스터 머신의 CSR을 승인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 <csr_name>은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

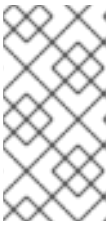
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs oc adm certificate approve
```

6. 모든 클라이언트 및 서버 CSR이 승인된 후 머신은 Ready 상태가 됩니다. 다음 명령을 실행하여 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0  Ready    master  73m  v1.22.1
master-1  Ready    master  73m  v1.22.1
master-2  Ready    master  74m  v1.22.1
worker-0  Ready    worker  11m  v1.22.1
worker-1  Ready    worker  11m  v1.22.1
```



참고

머신이 Ready 상태로 전환하는 데 서버 CSR의 승인 후 몇 분이 걸릴 수 있습니다.

추가 정보

- CSR에 대한 자세한 내용은 인증서 서명 요청을 참조하십시오.

17.5.20. Operator의 초기 설정

컨트롤 플레인이 초기화된 후 일부 Operator를 즉시 구성하여 모두 사용 가능하도록 해야 합니다.

사전 요구 사항

- 컨트롤 플레인이 초기화되어 있습니다.

프로세스

1. 클러스터 구성 요소가 온라인 상태인지 확인합니다.

```
$ watch -n5 oc get clusteroperators
```

출력 예

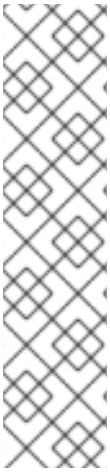
```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.9.0     True        False         False      19m
baremetal                                  4.9.0     True        False         False      37m
cloud-credential                           4.9.0     True        False         False      40m
cluster-autoscaler                         4.9.0     True        False         False      37m
config-operator                            4.9.0     True        False         False      38m
console                                    4.9.0     True        False         False      26m
csi-snapshot-controller                    4.9.0     True        False         False      37m
dns                                        4.9.0     True        False         False      37m
etcd                                       4.9.0     True        False         False      36m
image-registry                             4.9.0     True        False         False      31m
ingress                                    4.9.0     True        False         False      30m
insights                                   4.9.0     True        False         False      31m
kube-apiserver                             4.9.0     True        False         False      26m
kube-controller-manager                    4.9.0     True        False         False      36m
kube-scheduler                             4.9.0     True        False         False      36m
kube-storage-version-migrator              4.9.0     True        False         False      37m
machine-api                                4.9.0     True        False         False      29m
machine-approver                           4.9.0     True        False         False      37m
machine-config                             4.9.0     True        False         False      36m
marketplace                                4.9.0     True        False         False      37m
monitoring                                 4.9.0     True        False         False      29m
network                                    4.9.0     True        False         False      38m
node-tuning                                4.9.0     True        False         False      37m
openshift-apiserver                        4.9.0     True        False         False      32m
openshift-controller-manager               4.9.0     True        False         False      30m
openshift-samples                          4.9.0     True        False         False      32m
operator-lifecycle-manager                 4.9.0     True        False         False      37m
operator-lifecycle-manager-catalog         4.9.0     True        False         False      37m
operator-lifecycle-manager-packageserver   4.9.0     True        False         False      32m
service-ca                                 4.9.0     True        False         False      38m
storage                                    4.9.0     True        False         False      37m
```

2. 사용할 수 없는 Operator를 구성합니다.

17.5.20.1. 설치 중 제거된 이미지 레지스트리

공유 가능한 개체 스토리지를 제공하지 않는 플랫폼에서 **OpenShift Image Registry Operator**는 자체적으로 **Removed**로 부트스트랩합니다. 이를 통해 **openshift-installer**가 이러한 플랫폼 유형에서 설치를 완료할 수 있습니다.

설치 후 **managementState**를 **Removed**에서 **Managed**로 전환하도록 **Image Registry Operator** 구성을 편집해야 합니다.



참고

Prometheus 콘솔은 **ImageRegistryRemoved** 경고를 제공합니다. 다음은 예시 경고입니다.

"이미지 레지스트리가 제거되었습니다. **ImageStreamTags**를 참조하는 **ImageStreamTags, BuildConfigs** 및 **DeploymentConfigs**가 예상대로 작동하지 않을 수 있습니다. 스토리지를 구성하고 **configs.imageregistry.operator.openshift.io**를 편집하여 구성을 **Managed** 상태로 업데이트하십시오."

17.5.20.2. 이미지 레지스트리 스토리지 구성

기본 스토리지를 제공하지 않는 플랫폼에서는 처음에 **Image Registry Operator**를 사용할 수 없습니다. 설치한 후에 스토리지를 사용하도록 레지스트리를 구성하여 **Registry Operator**를 사용 가능하도록 만들어야 합니다.

프로덕션 클러스터에 필요한 영구 볼륨을 구성하는 과정의 지침이 표시됩니다. 해당하는 경우, 프로덕션 환경 외 클러스터에서만 사용할 수 있는 저장 위치로서 빈 디렉터리를 구성하는 과정의 지침이 표시됩니다.

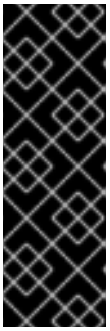
업그레이드 중에 **Recreate** 롤아웃 전략을 사용하여 이미지 레지스트리의 블록 스토리지 유형 사용을 허용하기 위한 추가 지침이 제공됩니다.

17.5.20.2.1. VMware vSphere용 레지스트리 스토리지 구성

클러스터 관리자는 설치한 후 스토리지를 사용하도록 레지스트리를 구성해야 합니다.

사전 요구 사항

- 클러스터 관리자 권한이 있어야 합니다.
- VMware vSphere에 클러스터가 있어야 합니다.
- Red Hat OpenShift Container Storage와 같이 클러스터용 영구 스토리지 프로비저닝.



중요

OpenShift Container Platform은 복제본이 하나만 있는 경우 이미지 레지스트리 스토리지에 대한 ReadWriteOnce 액세스를 지원합니다. ReadWriteOnce 액세스에서는 레지스트리가 Recreate 롤아웃 전략을 사용해야 합니다. 두 개 이상의 복제본으로 고 가용성을 지원하는 이미지 레지스트리를 배포하려면 ReadWriteMany 액세스가 필요합니다.

- "100Gi" 용량이 필요합니다.

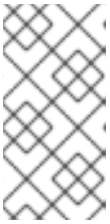
중요

테스트 결과, RHEL의 NFS 서버를 핵심 서비스용 스토리지 백엔드로 사용하는 데 문제가 있는 것으로 나타납니다. 여기에는 OpenShift Container Registry and Quay, 스토리지 모니터링을 위한 Prometheus, 로깅 스토리지를 위한 Elasticsearch가 포함됩니다. 따라서 RHEL NFS를 사용하여 핵심 서비스에서 사용하는 PV를 백업하는 것은 권장되지 않습니다.

마켓플레이스의 다른 NFS 구현에는 이러한 문제가 나타나지 않을 수 있습니다. 이러한 OpenShift Container Platform 핵심 구성 요소에 대해 완료된 테스트에 대한 자세한 내용은 개별 NFS 구현 공급업체에 문의하십시오.

절차

1. 스토리지를 사용하도록 레지스트리를 구성하기 위해 configs.imageregistry/cluster 리소스에서 spec.storage.pvc를 변경합니다.



참고

공유 스토리지를 사용할 때 보안 설정을 확인하여 외부에서의 액세스를 방지합니다.

2. 레지스트리 pod가 없는지 확인합니다.

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

출력 예

```
No resourses found in openshift-image-registry namespace
```

참고

출력에 레지스트리 Pod가 있는 경우 이 절차를 계속할 필요가 없습니다.

3. 레지스트리 구성을 확인합니다.

```
$ oc edit configs.imageregistry.operator.openshift.io
```

출력 예

```
storage:
  pvc:
    claim: 1
```

1 image-registry-storage 영구 볼륨 클레임 (PVC)을 자동으로 생성할 수 있도록 claim 필드를 비워 둡니다. PVC는 기본 스토리지 클래스를 기반으로 생성됩니다. 그러나 기본 스토리지 클래스에서 RADOS 블록 장치(RBD)와 같은 ReadWriteOnce(ReadWriteOnce) 볼륨을 제공할 수 있으므로 복제본이 두 개 이상될 때 문제가 발생할 수 있습니다.

4. clusteroperator 상태를 확인합니다.

```
$ oc get clusteroperator image-registry
```

출력 예

```
NAME             VERSION                             AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                                 True        False         False     6h50m
```

17.5.20.2.2. 프로덕션 환경 외 클러스터에서 이미지 레지스트리의 스토리지 구성

이미지 레지스트리 Operator에 대한 스토리지를 구성해야 합니다. 프로덕션 환경 외 클러스터의 경우, 이미지 레지스트리를 빈 디렉터리로 설정할 수 있습니다. 이렇게 하는 경우 레지스트리를 다시 시작하면 모든 이미지가 손실됩니다.

프로세스

- 이미지 레지스트리 스토리지를 빈 디렉터리로 설정하려면 다음을 수행하십시오.

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch
'{"spec":{"storage":{"emptyDir":{}}}}'
```

주의

프로덕션 환경 외 클러스터에 대해서만 이 옵션을 구성하십시오.

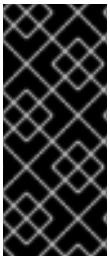
Image Registry Operator가 구성 요소를 초기화하기 전에 이 명령을 실행하면 oc patch 명령이 실패하며 다음 오류가 발생합니다.

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster"
not found
```

몇 분 후에 명령을 다시 실행하십시오.

17.5.20.2.3. VMware vSphere용 블록 레지스트리 스토리지 구성

클러스터 관리자로서 업그레이드 중에 이미지 레지스트리가 **vSphere VMDK(Virtual Machine Disk)**와 같은 블록 스토리지 유형을 사용할 수 있도록 허용하기 위해 **Recreate** 롤아웃 전략을 사용할 수 있습니다.



중요

블록 스토리지 볼륨이 지원되지만 프로덕션 클러스터에서 이미지 레지스트리와 함께 사용하는 것은 권장되지 않습니다. 레지스트리가 블록 스토리지에 구성된 설치는 레지스트리가 둘 이상의 복제본을 가질 수 없기 때문에 가용성이 높지 않습니다.

프로세스

1. 이미지 레지스트리 스토리지를 블록 스토리지 유형으로 설정하려면 **Recreate** 롤아웃 전략을 사용하고 복제본 **1**개 만으로 실행되도록 레지스트리를 패치합니다.

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. 블록 스토리지 장치에 **PV**를 프로비저닝하고 해당 볼륨의 **PVC**를 생성합니다. 요청된 블록 볼륨은 **RWO(ReadWriteOnce)** 액세스 모드를 사용합니다.

   a. **VMware vSphere PersistentVolumeClaim** 개체를 정의하려면 다음 내용이 포함된 **pvc.yaml** 파일을 생성합니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage 1
  namespace: openshift-image-registry 2
spec:
  accessModes:
  - ReadWriteOnce 3
  resources:
    requests:
      storage: 100Gi 4
```

1

PersistentVolumeClaim 개체를 표시하는 고유한 이름입니다.

2

PersistentVolumeClaim 오브젝트의 네임스페이스로 openshift-image-registry입니다.

3

영구 볼륨 클레임의 액세스 모드입니다. ReadWriteOnce를 사용하면 단일 노드에서 읽기 및 쓰기 권한으로 볼륨을 마운트할 수 있습니다.

4

영구 볼륨 클레임의 크기입니다.

b. 파일에서 PersistentVolumeClaim 오브젝트를 만듭니다.

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 올바른 PVC를 참조하도록 레지스트리 설정을 편집합니다.

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

출력 예

```
storage:
  pvc:
    claim: 1
```

1

사용자 지정 PVC를 만들면 image-registry-storage PVC의 기본 자동 생성을 위해 claim 필드를 비워둘 수 있습니다.

올바른 PVC를 참조하도록 레지스트리 스토리지를 구성하는 방법은 vSphere 용 레지스트리 구성을

참조하십시오.

17.5.21. 사용자 프로비저닝 인프라에 설치 완료

Operator 구성을 완료한 후 제공하는 인프라에 클러스터 설치를 완료할 수 있습니다.

사전 요구 사항

- 컨트롤 플레인이 초기화되어 있습니다.
- 초기 Operator 구성을 완료해야 합니다.

프로세스

1. 다음 명령을 사용하여 모든 클러스터 구성 요소가 온라인 상태인지 확인합니다.

```
$ watch -n5 oc get clusteroperators
```

출력 예

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.9.0     True        False         False      19m
baremetal                                  4.9.0     True        False         False      37m
cloud-credential                           4.9.0     True        False         False      40m
cluster-autoscaler                         4.9.0     True        False         False      37m
config-operator                            4.9.0     True        False         False      38m
console                                    4.9.0     True        False         False      26m
csi-snapshot-controller                    4.9.0     True        False         False      37m
dns                                        4.9.0     True        False         False      37m
etcd                                       4.9.0     True        False         False      36m
image-registry                             4.9.0     True        False         False      31m
ingress                                    4.9.0     True        False         False      30m
insights                                   4.9.0     True        False         False      31m
kube-apiserver                             4.9.0     True        False         False      26m
kube-controller-manager                    4.9.0     True        False         False      36m
kube-scheduler                             4.9.0     True        False         False      36m
kube-storage-version-migrator              4.9.0     True        False         False      37m
machine-api                                4.9.0     True        False         False      29m
machine-approver                           4.9.0     True        False         False      37m
machine-config                             4.9.0     True        False         False      36m
marketplace                                4.9.0     True        False         False      37m
monitoring                                 4.9.0     True        False         False      29m
```

```
network                                    4.9.0     True        False         False      38m
node-tuning                                4.9.0     True        False         False      37m
openshift-apiserver                        4.9.0     True        False         False      32m
openshift-controller-manager               4.9.0     True        False         False      30m
openshift-samples                          4.9.0     True        False         False      32m
operator-lifecycle-manager                 4.9.0     True        False         False      37m
operator-lifecycle-manager-catalog         4.9.0     True        False         False      37m
operator-lifecycle-manager-packageserver   4.9.0     True        False         False      32m
service-ca                                 4.9.0     True        False         False      38m
storage                                    4.9.0     True        False         False      37m
```

또는 다음 명령은 모든 클러스터를 사용할 수 있을 때 알립니다. 또한 인증 정보를 검색하고 표시합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

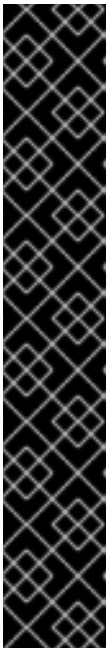
1

<installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

출력 예

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator가 Kubernetes API 서버에서 OpenShift Container Platform 클러스터 배포를 완료하면 명령이 성공합니다.



중요

- 설치 프로그램에서 생성하는 **Ignition** 구성 파일에 **24**시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 **24**시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 **kubelet** 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(**CSR**)을 수동으로 승인해야 합니다. 자세한 내용은 ***Recovering from expired control plane certificates*** 문서를 참조하십시오.

- 클러스터를 설치한 후 **24**시간에서 **22**시간까지의 인증서가 교체되기 때문에 생성된 후 **12**시간 이내에 **Ignition** 구성 파일을 사용하는 것이 좋습니다. **12**시간 이내에 **Ignition** 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

2. **Kubernetes API** 서버가 **Pod**와 통신하고 있는지 확인합니다.

   a. 모든 **Pod** 목록을 보려면 다음 명령을 사용하십시오.

   ```
   $ oc get pods --all-namespaces
   ```

   출력 예

   ```
   NAMESPACE                       NAME                                        READY   STATUS
   RESTARTS   AGE
   openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8
   1/1     Running   1          9m
   openshift-apiserver             apiserver-67b9g                             1/1     Running   0
   3m
   openshift-apiserver             apiserver-ljcmx                             1/1     Running   0
   1m
   openshift-apiserver             apiserver-z25h4                             1/1     Running   0
   2m
   openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8
   1/1     Running   0          5m
   ...
   ```

   b. 다음 명령을 사용하여 이전 명령의 출력에 나열된 **Pod**의 로그를 표시합니다.

   -

```
$ oc logs <pod_name> -n <namespace> 1
```

1

이전 명령의 출력에 표시된 대로 **Pod** 이름과 네임스페이스를 지정합니다.

Pod 로그가 표시되면 **Kubernetes API** 서버는 클러스터 시스템과 통신할 수 있습니다.

3. **FCP(Fibre Channel Protocol)**를 사용하는 설치에는 다중 경로를 활성화하기 위해 추가 단계가 필요합니다. 설치 중에 멀티패스를 활성화하지 마십시오.

   자세한 내용은 *설치 후 머신 구성 작업* 설명서에서 **"RHCOS에서 커널 인수를 사용하여 멀티패스 활성화"**를 참조하십시오.

vSphere에 컴퓨팅 머신 추가에 따라 클러스터 설치가 완료된 후 추가 컴퓨팅 머신을 추가할 수 있습니다.

17.5.22. VMware vSphere 볼륨 백업

OpenShift Container Platform은 새 볼륨을 독립 영구 디스크로 프로비저닝하여 클러스터의 모든 노드에서 볼륨을 자유롭게 연결 및 분리합니다. 결과적으로 스냅샷을 사용하는 볼륨을 백업하거나 스냅샷에서 볼륨을 복원할 수 없습니다. 자세한 내용은 스냅 샷 제한 을 참조하십시오.

프로세스

영구 볼륨의 백업을 생성하려면 다음을 수행합니다.

1. 영구 볼륨을 사용중인 애플리케이션을 중지합니다.
2. 영구 볼륨을 복제합니다.
3. 애플리케이션을 다시 시작합니다.
4. 복제된 볼륨의 백업을 만듭니다.

5. 복제된 볼륨을 삭제합니다.

17.5.23. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

17.5.24. 다음 단계

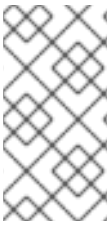
- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.
- 레지스트리를 설정하고 레지스트리 스토리지를 구성합니다.
- 선택 사항: vSphere Problem Detector Operator에서 이벤트를 보고 클러스터에 권한 또는 스토리지 구성 문제가 있는지 확인합니다.

17.6. 네트워크 사용자 지정으로 VSPHERE에 클러스터 설치

OpenShift Container Platform 4.9 버전에서는 사용자 지정 네트워크 구성 옵션으로 프로비저닝하는 클러스터를 VMware vSphere 인프라에 설치할 수 있습니다. 네트워크 구성을 사용자 지정할 경우, 클러스터가 사용자 환경의 기존 IP 주소 할당과 공존하고 기존 MTU 및 VXLAN 구성과 통합될 수 있습니다.

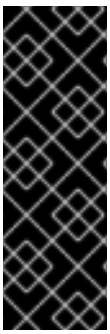
설치 과정에서 네트워크 구성 매개변수를 대부분 설정해야 하며, 실행 중인 클러스터에서는

kubeProxy 구성 매개변수만 수정할 수 있습니다.



참고

OpenShift Container Platform은 단일 VMware vCenter에만 클러스터 배포를 지원합니다. 여러 vCenter에 머신/머신 세트가 있는 클러스터 배포는 지원되지 않습니다.



중요

사용자가 프로비저닝한 인프라 설치를 수행하는 단계는 예시용으로만 제공됩니다. 사용자가 제공하는 인프라를 사용하여 클러스터를 설치하려면 vSphere 플랫폼 및 OpenShift Container Platform 설치 프로세스에 대한 정보가 필요합니다. 사용자 프로비저닝 인프라 설치 지침을 가이드로 사용하십시오. 다른 방법을 통해 필요한 리소스를 자유롭게 생성할 수 있습니다.

17.6.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.
- 설치를 완료하려면 vSphere 호스트에 RHCOS(Red Hat Enterprise Linux CoreOS) OVA를 업로드해야 합니다. 이 프로세스를 완료하는 시스템에는 vCenter 및 ESXi 호스트의 포트 443에 액세스해야 합니다. 포트 443에 액세스할 수 있는지 확인합니다.
- 방화벽을 사용하는 경우 관리자가 포트 443에 액세스할 수 있음을 확인합니다. 설치에 성공하려면 포트 443에서 컨트롤 플레인 노드가 vCenter 및 ESXi 호스트에 연결할 수 있어야 합니다.
- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성해야 합니다.

17.6.2. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.

중요

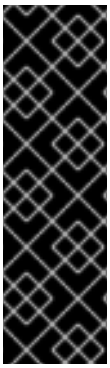
클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

17.6.3. VMware vSphere 인프라 요구사항

사용하는 구성 요소의 요구사항을 충족하는 VMware vSphere 버전 6 또는 7 인스턴스에 OpenShift Container Platform 클러스터를 설치해야 합니다.

표 17.47. vSphere 가상 환경의 버전 요구 사항

가상 환경 제품	필요한 버전
VM 하드웨어 버전	13 이상
vSphere ESXi hosts	6.5 이상
vCenter 호스트	6.5 이상



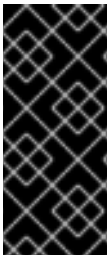
중요

이제 VMware vSphere 버전 6.7U2 또는 이전 버전에 클러스터를 설치하는 것이 더 이상 사용되지 않으며 가상 하드웨어 버전 13은 더 이상 사용되지 않습니다. 이러한 버전은 여전히 완전히 지원되지만 향후 OpenShift Container Platform 버전에서는 지원이 제거됩니다. 이제 OpenShift Container Platform의 vSphere 가상 머신의 하드웨어 버전 15가 기본값이 되었습니다. vSphere 노드의 하드웨어 버전을 업데이트하려면 " vSphere에서 실행되는 노드에서 하드웨어 업데이트" 문서를 참조하십시오.

표 17.48. VMware 구성 요소에 지원되는 최소 vSphere 버전

구성 요소	지원되는 최소 버전	설명
하이퍼바이저	HW 버전 13이 포함된 vSphere 6.5 이상	이 버전은 RHCOS(Red Hat Enterprise Linux CoreOS)가 지원하는 최소 버전입니다. Red Hat Enterprise Linux 8 지원 하이퍼바이저 목록을 참조하십시오.
In-tree 드라이버가 있는 스토리지	vSphere 6.5 이상	이 플러그인은 OpenShift Container Platform에 포함된 vSphere용 인트리 스토리지 드라이버를 사용하여 vSphere 스토리지를 생성합니다.
선택사항: 네트워킹(NSX-T)	vSphere 6.5U3 또는 vSphere 6.7U2 이상	OpenShift Container Platform에는 vSphere 6.5U3 또는 vSphere 6.7U2+가 필요합니다. NSX 및 OpenShift Container Platform의 호환성에 대한 자세한 내용은 VMware의 NSX 컨테이너 플러그인 설명서의 릴리스 노트 섹션을 참조하십시오.

vSphere 버전 6.5 인스턴스를 사용하는 경우 OpenShift Container Platform을 설치하기 전에 6.7U3 또는 7.0로 업그레이드하는 것이 좋습니다.



중요

OpenShift Container Platform을 설치하기 전에 ESXi 호스트의 시간이 동기화되었는지 확인해야 합니다. VMware 문서에서 Edit Time Configuration for a Host를 참조하십시오.

추가 리소스

- **vSphere 노드의 하드웨어 버전을 업데이트하려면 vSphere에서 실행 중인 노드에서 하드웨**

어 업데이트를 참조하십시오.

17.6.4. 사용자 프로비저닝 인프라를 포함한 클러스터의 시스템 요구사항

사용자 프로비저닝 인프라가 포함된 클러스터의 경우, 필요한 모든 시스템을 배포해야 합니다.

이 섹션에서는 사용자 프로비저닝 인프라에 **OpenShift Container Platform**을 배포해야 하는 요구 사항에 대해 설명합니다.

17.6.4.1. 클러스터 설치에 필요한 시스템

최소 **OpenShift Container Platform** 클러스터에 다음과 같은 호스트가 필요합니다.

표 17.49. 최소 필수 호스트

호스트	설명
임시 부트스트랩 시스템 한 개	컨트롤 플레인 시스템 세 개에 OpenShift Container Platform 클러스터를 배포하기 위한 부트스트랩 시스템이 클러스터에 필요합니다. 클러스터를 설치한 후 부트스트랩 시스템을 제거할 수 있습니다.
컨트롤 플레인 시스템 세 개	컨트롤 플레인 시스템은 컨트롤 플레인을 구성하는 Kubernetes 및 OpenShift Container Platform 서비스를 실행합니다.
두 개 이상의 컴퓨팅 시스템(작업자 시스템이라고도 함).	OpenShift Container Platform 사용자가 요청한 워크로드는 컴퓨팅 머신에서 실행됩니다.

중요

클러스터의 고가용성을 유지하려면 이러한 클러스터 시스템에 대해 별도의 물리적 호스트를 사용하십시오.

부트스트랩, 컨트롤 플레인 시스템은 운영 체제로 **RHCOS (Red Hat Enterprise Linux CoreOS)**를 사용해야 합니다. 그러나 컴퓨팅 머신은 **RHCOS(Red Hat Enterprise Linux CoreOS), RHEL (Red Hat Enterprise Linux) 7.9** 또는 **RHEL 8.4** 중에서 선택할 수 있습니다.

RHCOS는 **RHEL 8(Red Hat Enterprise Linux)**을 기반으로하며 모든 하드웨어 인증 및 요구사항을 모두 이어받습니다. Red Hat Enterprise Linux 기술 기능 및 제한을 참조하십시오.

중요

모든 가상 머신은 설치 프로그램과 동일한 데이터 저장소 및 폴더에 있어야 합니다.

17.6.4.2. 클러스터 설치를 위한 최소 리소스 요구 사항

각 클러스터 시스템이 다음과 같은 최소 요구사항을 충족해야 합니다.

표 17.50. 최소 리소스 요구사항

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
부트스트랩	RHCOS	4	16GB	100GB	300
컨트롤 플레인	RHCOS	4	16GB	100GB	300
컴퓨팅	RHCOS, RHEL 7.9 또는 RHEL 8.4 [3]	2	8GB	100GB	300

1. SMT(동시 멀티스레딩) 또는 하이퍼 스레딩이 활성화되지 않은 경우 하나의 vCPU는 하나의 물리적 코어와 동일합니다. 활성화하면 다음과 같은 공식을 사용하여 해당 비율을 계산합니다. (코어 당 스레드 수 × 코어 수) × 소켓 수 = vCPU 수

2. OpenShift Container Platform 및 Kubernetes는 디스크 성능에 민감하며 특히 10ms p99 fsync 기간이 필요한 컨트롤 플레인 노드의 etcd에 더 빠른 스토리지가 권장됩니다. 많은 클라우드 플랫폼에서 스토리지 크기와 IOPS를 함께 확장되므로 충분한 성능을 얻으려면 스토리지 볼륨을 과도하게 할당해야 할 수 있습니다.

3. 사용자가 프로비저닝한 모든 설치와 마찬가지로 클러스터에서 RHEL 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. RHEL 7 컴퓨팅 머신 사용은 더 이상 사용되지 않으며 향후 OpenShift Container Platform 4 릴리스에서 제거될 예정입니다.

17.6.4.3. 인증서 서명 요청 관리

사용자가 프로비저닝하는 인프라를 사용하는 경우 자동 시스템 관리 기능으로 인해 클러스터의 액세스가 제한되므로 설치한 후 클러스터 인증서 서명 요청(CSR)을 승인하는 메커니즘을 제공해야 합니다. kube-controller-manager는 kubelet 클라이언트 CSR만 승인합니다. machine-approver는 올바른 시스템에서 발행한 요청인지 확인할 수 없기 때문에 kubelet 자격 증명을 사용하여 요청하는 서비스 인증서의

유효성을 보장할 수 없습니다. **kubelet** 서빙 인증서 요청의 유효성을 확인하고 요청을 승인하는 방법을 결정하여 구현해야 합니다.

17.6.4.4. 사용자 프로비저닝 인프라에 대한 네트워킹 요구사항

모든 **RHCOS(Red Hat Enterprise Linux CoreOS)** 시스템이 부팅 중에 **Ignition** 구성 파일을 가져오려면 **initramfs**에 네트워킹을 구성해야 합니다.

초기 부팅 과정에서 시스템에 필요한 부팅 옵션을 제공하여 **DHCP** 서버를 통해 설정하거나 정적으로 설정하는 **IP** 주소 구성이 필요합니다. 네트워크 연결이 설정된 후 시스템은 **HTTP** 또는 **HTTPS** 서버에서 **Ignition** 구성 파일을 다운로드합니다. 그런 다음 **Ignition** 구성 파일을 사용하여 각 머신의 정확한 상태를 설정합니다. **Machine Config Operator**는 설치 후 새 인증서 또는 키 적용과 같은 머신에 대한 추가 변경을 완료합니다.

클러스터 시스템의 장기적인 관리를 위해 **DHCP** 서버를 사용하는 것이 좋습니다. **DHCP** 서버가 클러스터 시스템에 영구 **IP** 주소, **DNS** 서버 정보 및 호스트 이름을 제공하도록 구성되었는지 확인합니다.

참고

사용자 프로비저닝 인프라에 **DHCP** 서비스를 사용할 수 없는 경우 **RHCOS** 설치 시 노드에 **IP** 네트워킹 구성과 **DNS** 서버의 주소를 대신 제공할 수 있습니다. **ISO** 이미지에서 설치하는 경우 부팅 인수로 전달할 수 있습니다. 고정 **IP** 프로비저닝 *및 고급 네트워킹 옵션에 대한 자세한 내용은 **RHCOS** 설치 및 **OpenShift Container Platform** 부트스트랩 프로세스 시작* 섹션을 참조하십시오.

Kubernetes API 서버가 클러스터 시스템의 노드 이름을 확인할 수 있어야 합니다. **API** 서버와 작업자 노드가 서로 다른 영역에 있는 경우, **API** 서버가 노드 이름을 확인할 수 있도록 기본 **DNS** 검색 영역을 설정할 수 있습니다. 노드 개체와 모든 **DNS** 요청에서 항상 정규화된 도메인 이름으로 호스트를 가리키는 것도 지원되는 방법입니다

17.6.4.4.1. DHCP를 통해 클러스터 노드의 호스트 이름 설정

RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 호스트 이름은 **NetworkManager**를 통해 설정됩니다. 기본적으로 시스템은 **DHCP**를 통해 호스트 이름을 가져옵니다. **DHCP**에서 호스트 이름을 제공하지 않거나, 커널 인수를 통해 정적으로 설정하거나 다른 방법을 통해 설정하면 역방향 **DNS** 조회를 통해 얻을 수 있습니다. 역방향 **DNS** 조회는 노드에서 네트워크를 초기화한 후 수행되며 확인하는 데 시간이 걸릴 수 있습니다. 다른 시스템 서비스는 이 보다 먼저 시작하여 호스트 이름을 **localhost** 등으로 감지할 수 있습니다. **DHCP**를 사용하여 각 클러스터 노드의 호스트 이름을 제공하여 이 문제를 방지할 수 있습니다.

또한 **DHCP**를 통해 호스트 이름을 설정하면 **DNS** 분할 수평 구현 환경에서 수동으로 **DNS** 레코드 이

름 구성 오류를 무시할 수 있습니다.

17.6.4.4.2. 네트워크 연결 요구사항

OpenShift Container Platform 클러스터 구성 요소가 통신할 수 있도록 시스템 간 네트워크 연결을 구성해야 합니다. 각 시스템에서 클러스터에 있는 다른 모든 시스템의 호스트 이름을 확인할 수 있어야 합니다.

이 섹션에서는 필요한 포트에 대해 자세히 설명합니다.

중요

연결된 **OpenShift Container Platform** 환경에서 모든 노드는 플랫폼 컨테이너의 이미지를 가져오고 **Red Hat**에 원격 측정 데이터를 제공하기 위해 인터넷에 액세스할 수 있어야 합니다.

표 17.51. 모든 시스템 간 통신에 사용되는 포트

프로토콜	포트	설명
ICMP	해당 없음	네트워크 연결성 테스트
TCP	**1936**	메트릭
	9000-**9999**	**9100**-**9101** 포트의 노드 내보내기 및 **9099** 포트의 Cluster Version Operator를 포함한 호스트 수준 서비스.
	10250-**10259**	Kubernetes에서 예약하는 기본 포트
	10256	openshift-sdn
UDP	**4789**	VXLAN 및 Geneve
	6081	VXLAN 및 Geneve
	9000-**9999**	**9100**-**9101** 포트의 노드 내보내기를 포함한 호스트 수준 서비스.
	500	IPsec IKE 패킷
	4500	IPsec NAT-T 패킷
TCP/UDP	**30000**-**32767**	Kubernetes 노드 포트

프로토콜	포트	설명
ESP	해당 없음	IPsec Encapsulating Security Payload (ESP)

표 17.52. 모든 시스템과 컨트롤 플레인 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**6443**	Kubernetes API

표 17.53. 컨트롤 플레인 머신 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**2379-2380**	etcd 서버 및 피어 포트

이더넷 어댑터 하드웨어 주소 요구사항

클러스터용 VM을 프로비저닝할 때 각 VM에 대해 구성된 이더넷 인터페이스는 VMware OUI(Organizationally Unique Identifier) 할당 범위의 MAC 주소를 사용해야 합니다.

- **00:05:69:00:00:00 ~ 00:05:69:FF:FF:FF**
- **00:0c:29:00:00:00 ~ 00:0c:29:FF:FF:FF**
- **00:1c:14:00:00:00 ~ 00:1c:14:FF:FF:FF**
- **00:50:56:00:00:00 ~ 00:50:56:FF:FF:FF**

VMware OUI 외부의 MAC 주소를 사용하면 클러스터 설치가 실패합니다.

사용자 프로비저닝 인프라에 대한 NTP 구성

OpenShift Container Platform 클러스터는 기본적으로 공용 NTP(Network Time Protocol) 서버를 사용하도록 구성되어 있습니다. 로컬 엔터프라이즈 NTP 서버를 사용하거나 클러스터가 연결이 끊긴 네트워크에 배포되는 경우 특정 시간 서버를 사용하도록 클러스터를 구성할 수 있습니다. 자세한 내용은 *chrony 타임 서비스 설정* 문서를 참조하십시오.

DHCP 서버가 NTP 서버 정보를 제공하는 경우 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템

의 chrony 타임 서비스에서 정보를 읽고 NTP 서버와 클럭을 동기화할 수 있습니다.

추가 리소스

- chrony 타임 서비스 설정

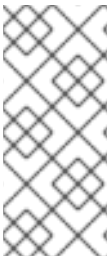
17.6.4.5. 사용자 프로비저닝 DNS 요구사항

OpenShift Container Platform 배포의 경우 다음 구성 요소에 DNS 이름을 확인해야 합니다.

- Kubernetes API
- OpenShift Container Platform 애플리케이션 와일드카드
- 부트스트랩, 컨트롤 플레인 및 컴퓨팅 시스템

Kubernetes API, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템에 대한 역방향 DNS 확인이 필요합니다.

DNS A/AAAA 또는 CNAME 레코드는 이름 확인에 사용되며 PTR 레코드는 역방향 이름 확인에 사용됩니다. RHCOS (Red Hat Enterprise Linux CoreOS)는 DHCP에서 호스트 이름을 제공하지 않는 한 모든 노드의 호스트 이름을 설정할 때 역방향 레코드를 사용하기 때문에 역방향 레코드가 중요합니다. 또한 역방향 레코드는 OpenShift Container Platform이 작동하는 데 필요한 인증서 서명 요청 (CSR)을 생성하는 데 사용됩니다.



참고

DHCP 서버를 사용하여 각 클러스터 노드에 호스트 이름을 제공하는 것이 좋습니다. 자세한 내용은 *사용자 프로비저닝 인프라 섹션에 대한 DHCP 권장 사항* 섹션을 참조하십시오.

사용자가 프로비저닝한 OpenShift Container Platform 클러스터에 대해 다음 DNS 레코드가 필요하며 설치 전에 있어야 합니다. 각 레코드에서 <cluster_name>은 클러스터 이름이고 <base_domain>은 install-config.yaml 파일에서 지정하는 기반 도메인입니다. 전체 DNS 레코드는 <component>.<cluster_name>.<base_domain> 형식입니다.

표 17.54. 필수 DNS 레코드

구성 요소	레코드	설명
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API 로드 밸런서를 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.
	api-int.<cluster_name>.<base_domain>.	내부적으로 API 로드 밸런서를 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다. **중요** API 서버는 Kubernetes에 기록된 호스트 이름으로 작업자 노드를 확인할 수 있어야 합니다. API 서버가 노드 이름을 확인할 수 없는 경우 프록시된 API 호출이 실패할 수 있으며 pod에서 로그를 검색할 수 없습니다.
라우트	***.apps.<cluster_name>.<base_domain>.**	애플리케이션 인그레스 로드 밸런서를 참조하는 와일드카드 DNS A/AAA 또는 CNAME 레코드입니다. 애플리케이션 인그레스 로드 밸런서는 Ingress 컨트롤러 Pod를 실행하는 머신을 대상으로 합니다. Ingress 컨트롤러 Pod는 기본적으로 컴퓨팅 머신에서 실행됩니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다. 예를 들어 **console-openshift-console.apps.<cluster_name>.<base_domain>**은 OpenShift Container Platform 콘솔의 와일드카드 경로로 사용됩니다.
부트스트랩 시스템	**bootstrap.<cluster_name>.<base_domain>.**	부트스트랩 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.
컨트롤 플레인 머신	**<master><n>.<cluster_name>.<base_domain>.**	컨트롤 플레인 노드의 각 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.
컴퓨팅 머신	**<worker><n>.<cluster_name>.<base_domain>.**	작업자 노드의 각 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.

참고

OpenShift Container Platform 4.4 이상에서는 DNS 구성에서 etcd 호스트 및 SRV 레코드를 지정할 필요가 없습니다.

작은 정보

dig 명령을 사용하여 이름과 역방향 이름을 확인할 수 있습니다. 자세한 검증 단계는 *사용자 프로비저닝 인프라의 DNS 확인* 섹션을 참조하십시오.

17.6.4.5.1. 사용자 프로비저닝 클러스터의 DNS 구성 예

이 섹션에서는 사용자 프로비저닝 인프라에 OpenShift Container Platform을 배포하기 위한 DNS 요구 사항을 충족하는 A 및 PTR 레코드 구성 샘플을 제공합니다. 샘플은 하나의 DNS 솔루션을 선택하기 위한 조언을 제공하기 위한 것이 아닙니다.

이 예제에서 클러스터 이름은 ocp4이고 기본 도메인은 example.com입니다.

사용자 프로비저닝 클러스터의 DNS A 레코드 구성 예

다음 BIND 영역 파일의 예제에서는 사용자가 프로비저닝한 클러스터의 이름 확인을 위한 샘플 A 레코드를 보여줍니다.

예 17.13. 샘플 DNS 영역 데이터베이스

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
master0.ocp4.example.com. IN A 192.168.1.97 5
master1.ocp4.example.com. IN A 192.168.1.98 6
master2.ocp4.example.com. IN A 192.168.1.99 7
```

```
;
worker0.ocp4.example.com. IN A 192.168.1.11 8
worker1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1

Kubernetes API의 이름 확인을 제공합니다. 레코드는 **API** 로드 밸런서의 **IP** 주소를 나타냅니다.

2

Kubernetes API의 이름 확인을 제공합니다. 레코드는 **API** 로드 밸런서의 **IP** 주소를 참조하며 내부 클러스터 통신에 사용됩니다.

3

와일드카드 경로의 이름 확인을 제공합니다. 레코드는 애플리케이션 인그레스 로드 밸런서의 **IP** 주소를 나타냅니다. 애플리케이션 인그레스 로드 밸런서는 **Ingress** 컨트롤러 **Pod**를 실행하는 머신을 대상으로 합니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

참고

이 예제에서는 **Kubernetes API** 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 **API** 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

4

부트스트랩 시스템의 이름 확인을 제공합니다.

5 6 7

컨트롤 플레인 시스템의 이름 확인을 제공합니다.

8 9

컴퓨팅 시스템의 이름 확인을 제공합니다.

사용자 프로비저닝 클러스터의 **DNS PTR** 레코드 구성 예

다음 예제 BIND 영역 파일은 사용자 프로비저닝 클러스터의 역방향 이름 확인을 위한 샘플 PTR 레코드를 보여줍니다.

예 17.14. 역방향 레코드의 샘플 DNS 영역 데이터베이스

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR master0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR master1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR master2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com. 8
;
;EOF
```

1 Kubernetes API의 역방향 DNS 확인을 제공합니다. PTR 레코드는 API 로드 밸런서의 레코드 이름을 참조합니다.

2 Kubernetes API의 역방향 DNS 확인을 제공합니다. PTR 레코드는 API 로드 밸런서의 레코드 이름을 참조하며 내부 클러스터 통신에 사용됩니다.

3 부트스트랩 시스템의 역방향 DNS 확인을 제공합니다.

4 5 6 컨트롤 플레인 시스템의 역방향 DNS 확인을 제공합니다.

7 8 컴퓨팅 시스템의 역방향 DNS 확인을 제공합니다.

참고

OpenShift Container Platform 애플리케이션 와일드카드에는 **PTR** 레코드가 필요하지 않습니다.

17.6.4.6. 사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구사항

OpenShift Container Platform을 설치하기 전에 **API** 및 애플리케이션 인그레스 로드 밸런싱 인프라를 프로비저닝해야 합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 **API** 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

참고

RHEL(Red Hat Enterprise Linux) 인스턴스를 사용하여 **API** 및 애플리케이션 인그레스 로드 밸런서를 배포하려면 **RHEL** 서브스크립션을 별도로 구매해야 합니다.

로드 밸런서 인프라는 다음 요구 사항을 충족해야 합니다.

1. **API** 로드 밸런서: 플랫폼과 상호 작용하고 플랫폼을 구성할 수 있도록 사용자(인간과 시스템 모두)에게 공통 끝점을 제공합니다. 다음 조건을 설정합니다.

   - **Layer 4** 로드 밸런싱 전용입니다. 이를 **Raw TCP, SSL Passthrough** 또는 **SSL Bridge** 모드라고 합니다. **SSL Bridge** 모드를 사용하는 경우, **API** 경로에 대해 **SNI(Server Name Indication,** 서버 이름 표시)를 활성화해야 합니다.
   - 스테이트리스 로드 밸런싱 알고리즘입니다. 옵션은 로드 밸런서 구현에 따라 달라집니다.

     참고

     API 로드 밸런서가 제대로 작동하기 위해 세션 지속성이 필요하지 않습니다.

   로드 밸런서의 전면과 후면 모두에서 다음 포트를 구성하십시오.

표 17.55. API 로드 밸런서

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
6443	부트스트랩 및 컨트롤 플레인. 부트스트랩 시스템이 클러스터 컨트롤 플레인을 초기화한 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다. API 서버 상태 검사 프로브에 대한 **/readyz** 끝점을 구성해야 합니다.	X	X	Kubernetes API 서버
22623	부트스트랩 및 컨트롤 플레인. 부트스트랩 시스템이 클러스터 컨트롤 플레인을 초기화한 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다.	X		시스템 구성 서버

참고

API 서버가 /readyz 엔드포인트를 해제하는 시점부터 풀에서 API 서버 인스턴스가 제거되는 시점까지 시간이 30초를 넘지 않도록 로드 밸런서를 구성해야 합니다. /readyz가 오류를 반환하거나 정상 상태가 된 후 정해진 시간 안에 끝점이 제거 또는 추가되어야 합니다. 5초 또는 10초의 프로빙 주기(두 번의 성공적인 요청은 정상 상태, 세 번의 요청은 비정상 상태)는 충분한 테스트를 거친 값입니다.

2. **애플리케이션 인그레스 로드 밸런서: 클러스터 외부에서 유입되는 애플리케이션 트래픽에 대한 인그래스 포인트를 제공합니다. 다음 조건을 설정합니다.**

- **Layer 4 로드 밸런싱 전용입니다. 이를 Raw TCP, SSL Passthrough 또는 SSL Bridge 모드라고 합니다. SSL Bridge 모드를 사용하는 경우 인그레스 경로에 대해 SNI(Server Name Indication, 서버 이름 표시)를 활성화해야 합니다.**

- **사용 가능한 옵션과 플랫폼에서 호스팅되는 애플리케이션 유형에 따라 연결 기반 또는 세션 기반 지속성이 권장됩니다.**

작은 정보

애플리케이션 인그레스 로드 밸런서에서 클라이언트의 실제 IP 주소를 확인할 수 있는 경우 소스 IP 기반 세션 지속성을 활성화하면 엔드 투 엔드 TLS 암호화를 사용하는 애플리케이션의 성능을 향상시킬 수 있습니다.

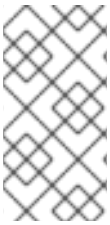
로드 밸런서의 전면과 후면 모두에서 다음 포트를 구성하십시오.

표 17.56. 애플리케이션 인그레스 로드 밸런서

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
443	기본적으로 인그레스 컨트롤러 pod, 컴퓨팅 또는 작업자를 실행하는 시스템입니다.	X	X	HTTPS 트래픽
80	기본적으로 인그레스 컨트롤러 pod, 컴퓨팅 또는 작업자를 실행하는 시스템입니다.	X	X	HTTP 트래픽
1936	기본적으로 Ingress 컨트롤러 Pod를 실행하는 작업자 노드입니다. 수신 상태 점검 프로브에 대해 **/healthz/ready** 끝점을 구성해야 합니다.	X	X	HTTP 트래픽

참고

컴퓨팅 노드가 0인 3-노드 클러스터를 배포하는 경우 Ingress 컨트롤러 Pod는 컨트롤 플레인 노드에서 실행됩니다. 3-노드 클러스터 배포에서 HTTP 및 HTTPS 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다.



참고

인그레스 라우터에 대한 작업 구성이 OpenShift Container Platform 클러스터에 필요합니다. 컨트롤 플레인 초기화 후 인그레스 라우터를 설정해야 합니다.

17.6.4.6.1. 사용자 프로비저닝 클러스터의 로드 밸런서 구성 예

이 섹션에서는 사용자 프로비저닝 클러스터의 로드 밸런싱 요구 사항을 충족하는 API 및 애플리케이션 수신 로드 밸런서 구성 예를 제공합니다. 샘플은 HAProxy 로드 밸런서에 대한 /etc/haproxy/haproxy.cfg 구성입니다. 이 예제에서는 하나의 로드 밸런싱 솔루션을 선택하기 위한 조언을 제공하는 것을 목적으로 하지 않습니다.

참고

이 예제에서는 Kubernetes API 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 API 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

예 17.15. API 및 애플리케이션 인그레스 로드 밸런서 구성 샘플

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
frontend stats
  bind *:1936
  mode            http
  log             global
  maxconn 10
  stats enable
  stats hide-version
  stats refresh 30s
  stats show-node
  stats show-desc Stats for ocp4 cluster (1)
  stats auth admin:ocp4
  stats uri /stats
listen api-server-6443 (2)
  bind *:6443
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:6443 check inter 1s backup (3)
  server master0 master0.ocp4.example.com:6443 check inter 1s
  server master1 master1.ocp4.example.com:6443 check inter 1s
  server master2 master2.ocp4.example.com:6443 check inter 1s
listen machine-config-server-22623 (4)
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup (5)
  server master0 master0.ocp4.example.com:22623 check inter 1s
  server master1 master1.ocp4.example.com:22623 check inter 1s
  server master2 master2.ocp4.example.com:22623 check inter 1s
listen ingress-router-443 (6)
  bind *:443
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:443 check inter 1s
  server worker1 worker1.ocp4.example.com:443 check inter 1s
listen ingress-router-80 (7)
  bind *:80
  mode tcp
```

```
balance source
server worker0 worker0.ocp4.example.com:80 check inter 1s
server worker1 worker1.ocp4.example.com:80 check inter 1s
```

1

이 예에서 클러스터 이름은 **ocp4** 입니다.

2

포트 **6443**은 **Kubernetes API** 트래픽을 처리하고 컨트롤 플레인 시스템을 가리킵니다.

3 5

부트스트랩 항목은 **OpenShift Container Platform** 클러스터 설치 전에 있어야 하며 부트스트랩 프로세스가 완료된 후 제거해야 합니다.

4

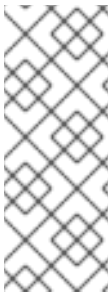
포트 **22623**은 머신 구성 서버 트래픽을 처리하고 컨트롤 플레인 시스템을 가리킵니다.

6

포트 **443**은 **HTTPS** 트래픽을 처리하고 **Ingress** 컨트롤러 **Pod**를 실행하는 시스템을 가리킵니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

7

포트 **80**은 **HTTP** 트래픽을 처리하고 **Ingress** 컨트롤러 **Pod**를 실행하는 머신을 가리킵니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

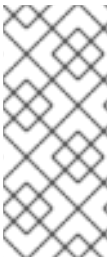


참고

컴퓨팅 노드가 **0**인 **3-**노드 클러스터를 배포하는 경우 **Ingress** 컨트롤러 **Pod**는 컨트롤 플레인 노드에서 실행됩니다. **3-**노드 클러스터 배포에서 **HTTP** 및 **HTTPS** 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다.

작은 정보

HAProxy를 로드 밸런서로 사용하는 경우 **HAProxy** 노드에서 **netstat -nltupe**를 실행하여 **haproxy** 프로세스가 포트 **6443, 22623, 443** 및 **80**에서 수신 대기 중인지 확인할 수 있습니다.



참고

HAProxy를 로드 밸런서로 사용하고 SELinux가 enforcing으로 설정된 경우 HAProxy 서비스가 setsebool -P haproxy_connect_any=1을 실행하여 구성된 TCP 포트에 바인딩할 수 있는지 확인해야 합니다.

17.6.5. 사용자 프로비저닝 인프라 준비

사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치하기 전에 기본 인프라를 준비해야 합니다.

이 섹션에서는 OpenShift Container Platform 설치를 준비하기 위해 클러스터 인프라를 설정하는 데 필요한 높은 수준의 단계에 대해 자세히 설명합니다. 여기에는 클러스터 노드에 대한 IP 네트워킹 및 네트워크 연결 구성, 방화벽을 통해 필요한 포트 활성화, 필수 DNS 및 로드 밸런싱 인프라를 설정하는 작업이 포함됩니다.

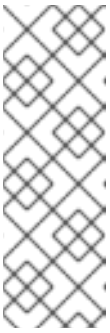
준비 후 클러스터 인프라는 *사용자 프로비저닝 인프라가 있는 클러스터의 요구 사항* 섹션에 설명된 요구 사항을 충족해야 합니다.

사전 요구 사항

- OpenShift Container Platform 4.x 테스트된 통합 페이지를 검토했습니다.
- *사용자 프로비저닝 인프라가 있는 클러스터의 요구 사항* 섹션에 자세히 설명된 인프라 요구 사항을 검토했습니다.

프로세스

1. DHCP를 사용하여 클러스터 노드에 IP 네트워킹 구성을 제공하는 경우 DHCP 서비스를 구성합니다.
   a. 노드의 영구 IP 주소를 DHCP 서버 구성에 추가합니다. 구성에서 관련 네트워크 인터페이스의 MAC 주소를 각 노드의 의도한 IP 주소와 일치시킵니다.
   b. DHCP를 사용하여 클러스터 시스템의 IP 주소 지정을 구성하는 경우 시스템은 DHCP를 통해 DNS 서버 정보도 가져옵니다. DHCP 서버 구성을 통해 클러스터 노드에서 사용하는 영구 DNS 서버 주소를 정의합니다.

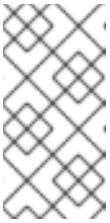


참고

DHCP 서비스를 사용하지 않는 경우 **RHCOS** 설치 시 **IP** 네트워킹 구성과 **DNS** 서버의 주소를 노드에 제공해야 합니다. **ISO** 이미지에서 설치하는 경우 부팅 인수로 전달할 수 있습니다. 고정 **IP** 프로비저닝 *및 고급 네트워킹 옵션에 대한 자세한 내용은 **RHCOS** 설치 및 **OpenShift Container Platform** 부트스트랩 프로세스 시작* 섹션을 참조하십시오.

c.

DHCP 서버 구성에 클러스터 노드의 호스트 이름을 정의합니다. 호스트 이름 고려 사항에 대한 자세한 내용은 ***DHCP**를 통해 클러스터 노드 호스트 이름 설정* 섹션을 참조하십시오.



참고

DHCP 서비스를 사용하지 않는 경우 클러스터 노드는 역방향 **DNS** 조회를 통해 호스트 이름을 가져옵니다.

2.

네트워크 인프라가 클러스터 구성 요소 간 필수 네트워크 연결을 제공하는지 확인합니다. 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 인프라 섹션의 네트워킹 요구 사항* 섹션을 참조하십시오.

3.

OpenShift Container Platform 클러스터 구성 요소가 통신하는 데 필요한 포트를 활성화하도록 방화벽을 구성합니다. 필요한 포트에 대한 자세한 내용은 *사용자 프로비저닝 인프라 섹션의 네트워킹 요구 사항* 섹션을 참조하십시오.

4.

클러스터에 필요한 **DNS** 인프라를 설정합니다.

a.

Kubernetes API, 애플리케이션 와일드카드, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템의 **DNS** 이름 확인을 구성합니다.

b.

Kubernetes API, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템에 대한 역방향 **DNS** 확인을 구성합니다.

OpenShift Container Platform DNS 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 **DNS** 요구 사항* 섹션을 참조하십시오.

5.

DNS 구성을 확인합니다.

a.

설치 노드에서 **Kubernetes API**의 레코드 이름, 와일드카드 경로 및 클러스터 노드에 대해 **DNS** 조회를 실행합니다. 응답의 **IP** 주소가 올바른 구성 요소에 해당하는지 확인합니다.

b.

설치 노드에서 로드 밸런서 및 클러스터 노드의 **IP** 주소에 대해 역방향 **DNS** 조회를 실행합니다. 응답의 레코드 이름이 올바른 구성 요소에 해당하는지 확인합니다.

자세한 **DNS** 검증 단계는 *사용자 프로비저닝 인프라에 대한 DNS 확인* 섹션을 참조하십시오.

6.

필요한 **API** 및 애플리케이션 수신 로드 밸런싱 인프라를 프로비저닝합니다. 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구 사항* 섹션을 참조하십시오.

참고

일부 로드 밸런싱 솔루션에는 로드 밸런싱을 초기화하기 전에 클러스터 노드의 **DNS** 이름을 확인해야 합니다.

17.6.6. 사용자 프로비저닝 인프라에 대한 DNS 확인 검증

사용자 프로비저닝 인프라에 **OpenShift Container Platform**을 설치하기 전에 **DNS** 구성을 확인할 수 있습니다.

중요

클러스터를 설치하기 전에 이 섹션에 설명된 검증 단계를 성공해야 합니다.

사전 요구 사항

- 사용자 프로비저닝 인프라에 필요한 **DNS** 레코드를 구성했습니다.

프로세스

1.

설치 노드에서 **Kubernetes API**의 레코드 이름, 와일드카드 경로 및 클러스터 노드에 대해 **DNS** 조회를 실행합니다. 응답에 포함된 **IP** 주소가 올바른 구성 요소에 해당하는지 확인합니다.

a.

Kubernetes API 레코드 이름을 조회합니다. 결과가 **API** 로드 밸런서의 **IP** 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
```

1

<nameserver_ip>를 네임서버의 IP 주소로, <cluster_name>을 클러스터 이름으로, <base_domain>을 기본 도메인 이름으로 바꿉니다.

출력 예

```
api.ocp4.example.com.  0 IN A 192.168.1.5
```

b.

Kubernetes 내부 API 레코드 이름을 조회합니다. 결과가 API 로드 밸런서의 IP 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
```

출력 예

```
api-int.ocp4.example.com.  0 IN A 192.168.1.5
```

c.

예제 *.apps.<cluster_name>.<base_domain>을 테스트합니다. DNS 와일드카드를 조회합니다. 모든 애플리케이션 와일드카드 조회는 애플리케이션 인그레스 로드 밸런서의 IP 주소로 확인되어야 합니다.

```
$ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.
<base_domain>
```

출력 예

```
random.apps.ocp4.example.com.  0 IN A 192.168.1.5
```

참고

예제 출력에서는 **Kubernetes API** 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 **API** 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

random 항목을 다른 와일드카드 값으로 교체할 수 있습니다. 예를 들어 **OpenShift Container Platform** 콘솔의 경로를 쿼리할 수 있습니다.

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

출력 예

```
console-openshift-console.apps.ocp4.example.com. 0 IN A 192.168.1.5
```

d.

부트스트랩 **DNS** 레코드 이름에 대해 조회를 실행합니다. 결과가 부트스트랩 노드의 **IP** 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.
<base_domain>
```

출력 예

```
bootstrap.ocp4.example.com.  0 IN A 192.168.1.96
```

e.

이 방법을 사용하여 컨트롤 플레인 및 컴퓨팅 노드의 **DNS** 레코드 이름에 대해 조회를 수행합니다. 결과가 각 노드의 **IP** 주소에 해당하는지 확인합니다.

2. 설치 노드에서 로드 밸런서 및 클러스터 노드의 IP 주소에 대해 역방향 DNS 조회를 실행합니다. 응답에 포함된 레코드 이름이 올바른 구성 요소에 해당하는지 확인합니다.

a. API 로드 밸런서의 IP 주소에 대해 역방향 조회를 수행합니다. 응답에 Kubernetes API 및 Kubernetes 내부 API의 레코드 이름이 포함되어 있는지 확인합니다.

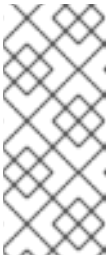
```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

출력 예

```
5.1.168.192.in-addr.arpa. 0 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 0 IN PTR api.ocp4.example.com. 2
```

1 Kubernetes 내부 API의 레코드 이름을 제공합니다.

2 Kubernetes API의 레코드 이름을 제공합니다.



참고

OpenShift Container Platform 애플리케이션 와일드카드에는 PTR 레코드가 필요하지 않습니다. 애플리케이션 인그레스 로드 밸런서의 IP 주소에 대한 역방향 DNS 확인에는 유효성 검사 단계가 필요하지 않습니다.

b. 부트스트랩 노드의 IP 주소에 대해 역방향 조회를 수행합니다. 결과가 부트스트랩 노드의 DNS 레코드 이름을 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

출력 예

```
96.1.168.192.in-addr.arpa. 0 IN PTR bootstrap.ocp4.example.com.
```

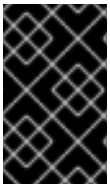
c. 이 방법을 사용하여 컨트롤 플레인 및 컴퓨팅 노드의 IP 주소에 대해 역방향 조회를 수행합니다. 결과가 각 노드의 DNS 레코드 이름과 일치하는지 확인합니다.

17.6.7. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 core 사용자의 ~/.ssh/authorized_keys 목록에 추가되어 암호 없는 인증을 활성화합니다.

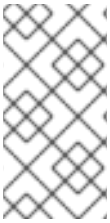
키가 노드에 전달되면 키 쌍을 사용하여 사용자 core로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. ./openshift-install gather 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.



중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.



참고

AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

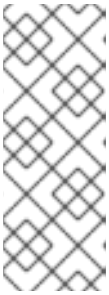
프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1

새 SSH 키의 경로 및 파일 이름(예: ~/.ssh/id_ed25519)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 '~/.ssh 디렉터리에 있는지 확인하십시오.



참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 ed25519 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 rsa 또는 ecdsa 알고리즘을 사용하는 키를 생성합니다.

2.

공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 ~/.ssh/id_ed25519.pub 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3.

아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 ./openshift-install gather 명령을 사용하려는 경우 필요합니다.

참고

일부 배포에서는 ~/.ssh/id_rsa 및 ~/.ssh/id_dsa와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a.

ssh-agent 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. ssh-agent에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: ~/.ssh/id_ed25519).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

17.6.8. 설치 프로그램 받기

OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

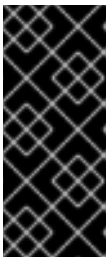
- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

프로세스

1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.

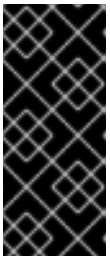
2. 인프라 공급자를 선택합니다.

3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.



중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.



중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 OpenShift Container Platform 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

17.6.9. 수동으로 설치 구성 파일 만들기

OpenShift Container Platform을 사용자가 프로비저닝한 설치의 경우 설치 구성 파일을 수동으로 생성합니다.

사전 요구 사항

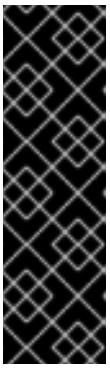
- 로컬 시스템에 설치 프로그램에 제공할 **SSH** 공개 키가 있습니다. 키는 디버깅 및 재해 복구를 위해 클러스터 노드에 대한 **SSH** 인증에 사용됩니다.

- **OpenShift Container Platform** 설치 프로그램과 클러스터의 풀 시크릿이 있습니다.

프로세스

1. 필요한 설치 자산을 저장할 설치 디렉터리를 만듭니다.

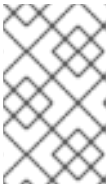
```
$ mkdir <installation_directory>
```



중요

디렉터리를 만들어야 합니다. 부트스트랩 **X.509** 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 **OpenShift Container Platform** 버전에서 설치 파일을 복사할 때는 주의하십시오.

2. 샘플 **install-config.yaml** 파일 템플릿을 사용자 지정하여 **<installation_directory>**에 저장합니다.



참고

이 설정 파일의 이름을 **install-config.yaml**로 지정해야 합니다.

참고

일부 플랫폼 유형의 경우 대체로 **./openshift-install create install-config --dir <installation_directory>** 를 실행하여 **install-config.yaml** 파일을 생성할 수 있습니다. 프롬프트에서 클러스터 구성에 대한 세부 정보를 제공할 수 있습니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.

중요

install-config.yaml 파일은 설치 과정의 다음 단계에서 사용됩니다. 이 시점에서 이를 백업해야 합니다.

17.6.9.1. VMware vSphere용 샘플 install-config.yaml 파일

install-config.yaml 파일을 사용자 지정하여 **OpenShift Container Platform** 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.

```
apiVersion: v1
baseDomain: example.com 1
compute:
- hyperthreading: Enabled 2 3
  name: worker
  replicas: 0 4
controlPlane:
  hyperthreading: Enabled 5 6
  name: master
  replicas: 3 7
metadata:
  name: test 8
platform:
  vsphere:
    vcenter: your.vcenter.server 9
    username: username 10
    password: password 11
    datacenter: datacenter 12
    defaultDatastore: datastore 13
    folder: "/<datacenter_name>/vm/<folder_name>/<subfolder_name>" 14
fips: false 15
pullSecret: '{"auths": ...}' 16
sshKey: 'ssh-ed25519 AAAA...' 17
```

1 클러스터의 기본 도메인입니다. 모든 **DNS** 레코드는 이 기본 도메인의 하위 도메인이어야 하며 클러스터 이름을 포함해야 합니다.

2 5 **controlPlane** 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 현재 두 섹션이 모두 단일 시스

템 풀을 정의하지만 향후 출시되는 **OpenShift Container Platform** 버전은 설치 과정에서 여러 컴퓨팅 풀 정의를 지원할 수 있습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

3 6

동시 멀티스레딩 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. 일부 클러스터 시스템에서 동시 멀티스레딩을 비활성화할 경우에는 해당 멀티스레딩을 모든 클러스터 시스템에서 비활성화해야 합니다.

중요

동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다. 동시 멀티스레딩을 비활성화하는 경우 시스템은 **8**개 이상의 **CPU**와 **32GB**의 **RAM**을 사용해야 합니다.

4

replicas 매개변수의 값을 **0**으로 설정해야 합니다. 이 매개변수는 클러스터를 생성하고 관리하는 작업자 수를 제어합니다. 이는 사용자 프로비저닝 인프라를 사용할 때 클러스터가 실행하지 않는 기능입니다. **OpenShift Container Platform** 설치를 완료하기 전에 클러스터에서 사용할 작업자 시스템을 수동으로 배포해야 합니다.

7

클러스터에 추가하는 컨트롤 플레인 시스템의 수입니다. 클러스터에서 이 값을 클러스터의 **etcd** 끝점 수로 사용하므로 이 값은 배포하는 컨트롤 플레인 시스템의 수와 일치해야 합니다.

8

DNS 레코드에 지정한 클러스터 이름입니다.

9

vCenter 서버의 정규화된 호스트 이름 또는 **IP** 주소입니다.

10

서버에 액세스하기 위한 사용자의 이름입니다. 이 사용자에게는 최소한 **vSphere**에서 정적 또는 동적 영구 볼륨 프로비저닝 에 필요한 역할과 권한이 있어야 합니다.

11

vSphere 사용자와 연결된 암호입니다.

12

13

사용할 기본 **vSphere** 데이터 저장소입니다.

14

선택사항: 설치 관리자 프로비저닝 인프라의 경우 설치 프로그램이 가상 머신을 생성하는 기존 폴더의 절대 경로(예: **/<datacenter_name>/vm/<folder_name>/<subfolder_name>**). 이 값을 제공하지 않으면 설치 프로그램이 인프라 **ID**로 이름이 지정된 데이터 센터 가상 머신 폴더에 최상위 폴더를 만듭니다. 클러스터에 인프라를 제공하는 경우에는 이 매개변수를 생략합니다.

15

FIPS 모드 활성화 또는 비활성화 여부입니다. 기본적으로 **FIPS** 모드는 비활성화됩니다. **FIPS** 모드가 활성화되면 **OpenShift Container Platform**이 실행되는 **RHCOS(Red Hat Enterprise Linux CoreOS)** 시스템에서 기본 **Kubernetes** 암호화 제품군은 우회하고 **RHCOS**와 함께 제공되는 암호화 모듈을 대신 사용합니다.

중요

FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 **OpenShift Container Platform** 배포에서만 지원됩니다.

16

OpenShift Cluster Manager 에서 얻은 풀 시크릿입니다. 이 풀 시크릿을 사용하면 **OpenShift Container Platform** 구성 요소에 대한 컨테이너 이미지를 제공하는 **Quay.io**를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

17

RHCOS(Red Hat Enterprise Linux CoreOS)의 **core** 사용자에 대한 기본 **SSH** 키의 공용 부분입니다.

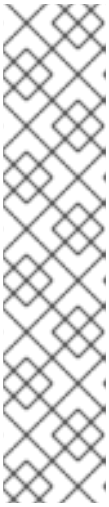
17.6.9.2. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 **HTTP** 또는 **HTTPS** 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 **OpenShift Container Platform** 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.

- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 **API**에 대한 호출을 포함하여 모든 클러스터 발신(**Egress**) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.



참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), **Google Cloud Platform (GCP)**, **Microsoft Azure** 및 **Red Hat OpenStack Platform (RHOSP)**에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. 클러스터 외부에서 **HTTP** 연결을 구축하는 데 사용할 프록시 **URL**입니다. **URL** 스키마는 **http**여야 합니다.

2. 클러스터 외부에서 **HTTPS** 연결을 구축하는 데 사용할 프록시 **URL**입니다.

3

대상 도메인 이름, IP 주소 또는 프록시에서 제외할 기타 네트워크 CIDR로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 .을 입력합니다. 예를 들어, .y.com은 x.y.com과 일치하지만 y.com은 일치하지 않습니다. *를 사용하여 모든 대상에 대해 프록시를 바이패스합니다. vCenter의 IP 주소와 해당 시스템에 사용하는 IP 범위를 포함해야 합니다.

4

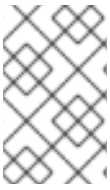
이 값을 제공하면 설치 프로그램에서 추가 CA 인증서를 보유할 openshift -config 네임스페이스에 user-ca- bundle 이라는 구성 맵을 생성합니다. additionalTrustBundle 및 하나 이상의 프록시 설정을 제공하는 경우 프록시 오브젝트는 trustedCA 필드의 user-ca-bundle 구성 맵을 참조하도록 구성됩니다. 그러면 Cluster Network Operator에서 trustedCA 매개변수에 대해 지정된 콘텐츠를 RHCOS 신뢰 번들과 병합하는 trusted-ca-bundle 구성 맵을 생성합니다. 프록시의 ID 인증서를 RHCOS 트러스트 번들에 있는 기관에서 서명하지 않은 경우 additionalTrustBundle 필드가 있어야 합니다.

참고

설치 프로그램에서 프록시 adinessEndpoints 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 OpenShift Container Platform을 설치할 때 참조하십시오.

제공되는 install-config.yaml 파일의 프록시 설정을 사용하는 cluster라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 cluster Proxy 오브젝트는 계속 생성되지만 spec은 nil이 됩니다.



참고

cluster라는 Proxy 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

17.6.10. 네트워크 구성 단계

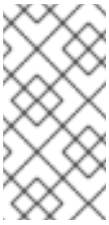
OpenShift Container Platform을 설치하기 전에 네트워크 구성을 사용자 지정할 수 있습니다.

1 단계

매니페스트 파일을 생성하기 전에 install-config.yaml 파일에서 다음 네트워크 관련 필드를 사용자 지정할 수 있습니다.

- **networking.networkType**
- **networking.clusterNetwork**
- **networking.serviceNetwork**
- **networking.machineNetwork**

이러한 필드에 대한 자세한 내용은 *설치 구성 매개 변수*에서 참조하십시오.



참고

기본 **NIC**가 상주하는 **CIDR**과 일치하도록 **networking.machineNetwork**를 설정합니다.

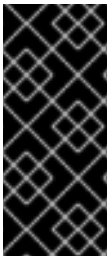
2 단계

openshift-install create manifests 를 실행하여 매니페스트 파일을 생성한 후 수정할 필드로 사용자 지정된 **Cluster Network Operator** 매니페스트를 정의할 수 있습니다. 매니페스트를 사용하여 고급 네트워크 구성을 지정할 수 있습니다.

install-config.yaml 파일에서 **1** 단계에 지정된 값을 **2** 단계에서 재정의할 수 없습니다. 그러나 **2** 단계에서 클러스터 네트워크 공급자를 추가로 사용자 지정할 수 있습니다.

17.6.11. 고급 네트워크 구성 지정

클러스터 네트워크 제공자의 고급 네트워크 구성을 사용하여 클러스터를 기존 네트워크 환경에 통합할 수 있습니다. 클러스터를 설치하기 전에만 고급 네트워크 구성을 지정할 수 있습니다.



중요

설치 프로그램에서 생성한 **OpenShift Container Platform** 매니페스트 파일을 수정하여 네트워크 구성을 사용자 정의하는 것은 지원되지 않습니다. 다음 절차에서와 같이 생성한 매니페스트 파일을 적용할 수 있습니다.

사전 요구 사항

- install-config.yaml 파일을 생성하고 수정 작업을 완료했습니다.

절차

1. 설치 프로그램이 포함된 디렉터리로 변경하고 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 <installation_directory>는 클러스터의 install-config.yaml 파일이 포함된 디렉터리의 이름을 지정합니다.

2. <installation_directory>/ manifests/ 디렉토리에 cluster-network-03-config.yml이라는 stub 매니페스트 파일을 만듭니다.

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
```

3. 다음과 같이 cluster-network-03-config.yml 파일에서 클러스터의 고급 네트워크 구성을 지정합니다.

OpenShift SDN 네트워크 공급자에 대한 다른 VXLAN 포트 지정

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    openshiftSDNConfig:
      vxlanPort: 4800
```

OVN-Kubernetes 네트워크 공급자의 IPsec 활성화

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    ovnKubernetesConfig:
      ipsecConfig: {}
```

4. 선택사항: manifests/cluster-network-03-config.yml 파일을 백업합니다. 설치 프로그램은 Ignition 구성 파일을 생성할 때 manifests/ 디렉터리를 사용합니다.

5. 컨트롤 플레인 시스템을 정의하는 Kubernetes 매니페스트 파일을 제거하고 machineSets 를 컴퓨팅합니다.

```
$ rm -f openshift/99_openshift-cluster-api_master-machines-*.yaml
openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

이러한 리소스는 사용자가 직접 생성하고 관리하기 때문에 초기화할 필요가 없습니다.

- 시스템 API로 MachineSet 파일을 보존하여 컴퓨팅 시스템을 생성할 수 있지만 사용자 환경과 일치하도록 해당 참조를 업데이트해야 합니다.

17.6.12. CNO(Cluster Network Operator) 구성

클러스터 네트워크의 구성은 CNO(Cluster Network Operator) 구성의 일부로 지정되며 cluster라는 이름의 CR(사용자 정의 리소스) 오브젝트에 저장됩니다. CR은 operator.openshift.io API 그룹에서 Network API의 필드를 지정합니다.

CNO 구성은 Network.config.openshift.io API 그룹의 Network API에서 클러스터 설치 중에 다음 필드를 상속하며 이러한 필드는 변경할 수 없습니다.

clusterNetwork

Pod IP 주소가 할당되는 IP 주소 풀입니다.

serviceNetwork

서비스를 위한 IP 주소 풀입니다.

defaultNetwork.type

OpenShift SDN 또는 OVN-Kubernetes와 같은 클러스터 네트워크 공급자입니다.

cluster라는 CNO 오브젝트에서 defaultNetwork 오브젝트의 필드를 설정하여 클러스터의 클러스터 네트워크 공급자 구성을 지정할 수 있습니다.

17.6.12.1. CNO(Cluster Network Operator) 구성 오브젝트

CNO(Cluster Network Operator)의 필드는 다음 표에 설명되어 있습니다.

표 17.57. CNO(Cluster Network Operator) 구성 오브젝트

필드	유형	설명
metadata.name	**string**	CNO 개체 이름입니다. 이 이름은 항상 **cluster**입니다.

spec.clusterNetwork	**array**	Pod IP 주소가 할당되는 IP 주소 블록과 클러스터의 각 개별 노드에 할당된 서브넷 접두사 길이를 지정하는 목록입니다. 예를 들면 다음과 같습니다.
		`spec:`
		`  clusterNetwork:`
		`  - cidr: 10.128.0.0/19`
		`  hostPrefix: 23`
		`  - cidr: 10.128.32.0/19`
		`  hostPrefix: 23`
		매니페스트를 생성하기 전에 **install-config.yaml** 파일에서만 이 필드를 사용자 지정할 수 있습니다. 값은 매니페스트 파일에서 읽기 전용입니다.

필드	유형	설명
spec.serviceNetwork	**array**	서비스의 IP 주소 블록입니다. OpenShift SDN 및 OVN-Kubernetes CNI(Container Network Interface) 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다. 예를 들면 다음과 같습니다. spec: serviceNetwork: - 172.30.0.0/14 매니페스트를 생성하기 전에 **install-config.yaml** 파일에서만 이 필드를 사용자 지정할 수 있습니다. 값은 매니페스트 파일에서 읽기 전용입니다.
spec.defaultNetwork	**object**	클러스터 네트워크의 CNI(Container Network Interface) 클러스터 네트워크 공급자를 구성합니다.
spec.kubeProxyConfig	**object**	이 개체의 필드는 kube-proxy 구성을 지정합니다. OVN-Kubernetes 클러스터 네트워크 공급자를 사용하는 경우 kube-proxy 구성이 적용되지 않습니다.

defaultNetwork 오브젝트 구성

defaultNetwork 오브젝트의 값은 다음 표에 정의되어 있습니다.

표 17.58. defaultNetwork 오브젝트

필드	유형	설명
type	**string**	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, 클러스터 네트워크 공급자가 설치 중에 선택됩니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다. **참고** OpenShift Container Platform은 기본적으로 OpenShift SDN CNI(Container Network Interface) 클러스터 네트워크 공급자를 사용합니다.
openshiftSDNConfig	**object**	이 오브젝트는 OpenShift SDN 클러스터 네트워크 공급자에만 유효합니다.
ovnKubernetesConfig	**object**	이 오브젝트는 OVN-Kubernetes 클러스터 네트워크 공급자에만 유효합니다.

OpenShift SDN CNI 네트워크 공급자에 대한 구성

다음 표에서는 **OpenShift SDN Container Network Interface (CNI) 클러스터 네트워크 공급자의 구성 필드를 설명합니다.**

표 17.59. openshiftSDNConfig 오브젝트

필드	유형	설명
mode	**string**	OpenShift SDN의 네트워크 격리 모드를 구성합니다. 기본값은 **NetworkPolicy**입니다. **Multitenant** 및 **Subnet** 값은 OpenShift Container Platform 3.x 버전과의 역호환할 수 있지만 권장되지는 않습니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
mtu	**integer**	VXLAN 오버레이 네트워크의 최대 전송 단위(MTU)입니다. 이는 기본 네트워크 인터페이스의 MTU를 기준으로 자동 탐지됩니다. 일반적으로 감지된 MTU를 재정의할 필요는 없습니다. 자동 감지 값이 예상 밖인 경우 노드의 기본 네트워크 인터페이스의 MTU가 올바른지 확인합니다. 이 옵션을 사용하여 노드의 기본 네트워크 인터페이스의 MTU 값을 변경할 수 없습니다. 클러스터에 다른 노드에 대한 다른 MTU 값이 필요한 경우 이 값을 클러스터의 가장 낮은 MTU 값보다 **50** 미만으로 설정해야 합니다. 예를 들어 클러스터의 일부 노드에는 **9001**의 MTU가 있고 일부에는 **1500**의 MTU가 있는 경우 이 값을 **1450**으로 설정해야 합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
vxlanPort	**integer**	모든 VXLAN 패킷에 사용할 포트입니다. 기본값은 **4789**입니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다. 기존 노드가 다른 VXLAN 네트워크에 속하는 가상 환경에서 실행 중인 경우에는 기본값을 변경해야 할 수도 있습니다. 예를 들어 VMware NSX-T 위에서 OpenShift SDN 오버레이를 실행할 때 두 SDN이 동일한 기본 VXLAN 포트 번호를 사용하므로 VXLAN의 대체 포트를 선택해야 합니다. AWS(Amazon Web Services)에서는 포트 **9000**과 포트 **9999** 사이에서 VXLAN의 대체 포트를 선택할 수 있습니다.

OpenShift SDN 구성 예

```
defaultNetwork:
  type: OpenShiftSDN
  openshiftSDNConfig:
    mode: NetworkPolicy
    mtu: 1450
    vxlanPort: 4789
```

OVN-Kubernetes CNI 클러스터 네트워크 공급자에 대한 구성

다음 표에서는 OVN-Kubernetes CNI 클러스터 네트워크 공급자의 구성 필드를 설명합니다.

표 17.60. ovnKubernetesConfig object

필드	유형	설명
mtu	**integer**	Geneve(Generic Network Virtualization Encapsulation) 오버레이 네트워크의 MTU(최대 전송 단위)입니다. 이는 기본 네트워크 인터페이스의 MTU를 기준으로 자동 탐지됩니다. 일반적으로 감지된 MTU를 재정의할 필요는 없습니다. 자동 감지 값이 예상 밖인 경우 노드의 기본 네트워크 인터페이스의 MTU가 올바른지 확인합니다. 이 옵션을 사용하여 노드의 기본 네트워크 인터페이스의 MTU 값을 변경할 수 없습니다. 클러스터에 다른 노드에 대한 다른 MTU 값이 필요한 경우, 이 값을 클러스터의 가장 낮은 MTU 값보다 **100** 미만으로 설정해야 합니다. 예를 들어, 클러스터의 일부 노드에 **9001**의 MTU가 있고 일부에는 **1500**의 MTU가 있는 경우 이 값을**1400**으로 설정해야 합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
genevePort	**integer**	모든 Geneve 패킷에 사용할 포트입니다. 기본값은 **6081**입니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
ipsecConfig	**object**	IPsec 암호화를 활성화하려면 빈 오브젝트를 지정합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
policyAuditConfig	**object**	네트워크 정책 감사 로깅을 사용자 정의할 구성 오브젝트를 지정합니다. 설정되지 않으면 기본값 감사 로그 설정이 사용됩니다.

표 17.61. policyAuditConfig object

필드	유형	설명
rateLimit	integer	노드당 1초마다 생성할 최대 메시지 수입니다. 기본값은 초당 **20** 개의 메시지입니다.
maxFileSize	integer	감사 로그의 최대 크기(바이트)입니다. 기본값은 **50000000 또는 50** MB입니다.

필드	유형	설명
대상	string	다음 추가 감사 로그 대상 중 하나입니다. **libc** 호스트에서 journald 프로세스의 libc **syslog()** 함수입니다. **udp:<host>:<port>** syslog 서버입니다. **<host>:<port>**를 syslog 서버의 호스트 및 포트로 바꿉니다. **unix:<file>** **<file>**로 지정된 Unix Domain Socket 파일입니다. **null** 감사 로그를 추가 대상으로 보내지 마십시오.
syslogFacility	string	RFC5424에 정의된 **kern**과 같은 syslog 기능입니다. 기본값은 **local0**입니다.

OVN-Kubernetes 구성 예

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
    ipsecConfig: {}
```

kubeProxyConfig 오브젝트 구성

kubeProxyConfig 오브젝트의 값은 다음 표에 정의되어 있습니다.

표 17.62. kubeProxyConfig object

필드	유형	설명

필드	유형	설명
iptablesSyncPeriod	**string**	**iptables** 규칙의 새로 고침 간격입니다. 기본값은 **30s**입니다. 유효 접미사로 **s**, **m**, **h**가 있으며, 자세한 설명은 Go **time** 패키지 문서를 참조하십시오. **참고** OpenShift Container Platform 4.3 이상에서는 성능이 개선되어 더 이상 **iptablesSyncPeriod** 매개변수를 조정할 필요가 없습니다.

proxyArguments.iptables-min-sync-period	**array**	**iptables** 규칙을 새로 고치기 전 최소 기간입니다. 이 필드를 통해 새로 고침 간격이 너무 짧지 않도록 조정할 수 있습니다. 유효 접미사로 **s**, **m**, **h**가 있으며, 자세한 설명은 Go **time** 패키지를 참조하십시오. 기본값은 다음과 같습니다.
		
		`kubeProxyConfig:`
		`  proxyArguments:`
		`  iptables-min-sync-period:`
		`  - 0s`

17.6.13. Ignition 구성 파일 생성

클러스터 시스템을 수동으로 시작해야 하므로 클러스터가 시스템을 생성하는 데 필요한 **Ignition** 구성 파일을 사용자가 생성해야 합니다.

중요

- 설치 프로그램에서 생성하는 **Ignition** 구성 파일에 **24**시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 **24**시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 **kubelet** 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(**CSR**)을 수동으로 승인해야 합니다. 자세한 내용은 ***Recovering from expired control plane certificates*** 문서를 참조하십시오.
- 클러스터를 설치한 후 **24**시간에서 **22**시간까지의 인증서가 교체되기 때문에 생성된 후 **12**시간 이내에 **Ignition** 구성 파일을 사용하는 것이 좋습니다. **12**시간 이내에 **Ignition** 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

사전 요구 사항

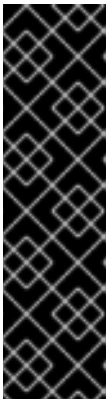
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

프로세스

- Ignition 구성 파일을 가져옵니다.

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 <installation_directory>는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.



중요

install-config.yaml 파일을 생성한 경우 파일이 포함된 디렉터리를 지정하십시오. 그렇지 않으면 빈 디렉터리를 지정합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

다음 파일이 디렉터리에 생성됩니다.

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

17.6.14. 인프라 이름 추출

Ignition 구성 파일에는 VMware vSpher에서 클러스터를 고유하게 식별하는 데 사용할 수 있는 고유한 클러스터 ID가 포함되어 있습니다. 클러스터 ID를 가상 머신 폴더의 이름으로 사용하려면 해당 ID를 추출해야 합니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받으셨습니다.
- 클러스터에 대한 Ignition 구성 파일을 생성하셨습니다.
- jq CLI를 설치하셨습니다.

프로세스

- Ignition 구성 파일 메타데이터에서 인프라 이름을 추출하여 확인하려면 다음 명령을 실행하십시오.

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

출력 예

```
openshift-vw9j6 1
```

1 이 명령의 출력은 클러스터 이름과 임의의 문자열입니다.

17.6.15. RHCOS 설치 및 OpenShift Container Platform 부트스트랩 프로세스 시작

VMware vSphere의 사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치하려면 vSphere 호스트에 RHCOS(Red Hat Enterprise Linux Core OS)를 설치해야 합니다. RHCOS를 설치할 때 설치 중인 머신 유형에 대해 OpenShift Container Platform 설치 프로그램에서 생성한 Ignition 구성 파일을 제공해야 합니다. 적합한 네트워킹, DNS 및 로드 밸런싱 인프라를 구성한 경우 RHCOS 머신이 재부팅된 후 OpenShift Container Platform 부트스트랩 프로세스가 자동으로 시작됩니다.

사전 요구 사항

- 클러스터에 대한 Ignition 구성 파일이 있어야 합니다.

- 사용자 컴퓨터에서 액세스할 수 있고 생성한 시스템이 액세스할 수 있는 HTTP 서버에 액세스할 수 있습니다.

- vSphere 클러스터 를 생성했습니다.

프로세스

1. 설치 프로그램에서 생성된 부트스트랩 Ignition 구성 파일 (<installation_directory>/bootstrap.ign)을 HTTP 서버에 업로드합니다. 이 파일의 URL을 기록해 둡니다.

2. 부트스트랩 노드의 다음 보조 Ignition 구성 파일을 <installation_directory>/merge-bootstrap.ign으로 컴퓨터에 저장합니다.

```
{
  "ignition": {
    "config": {
      "merge": [
        {
          "source": "<bootstrap_ignition_config_url>", 1
          "verification": {}
        }
      ]
    },
    "timeouts": {},
    "version": "3.2.0"
  },
  "networkd": {},
  "passwd": {},
  "storage": {},
  "systemd": {}
}
```

1. 호스팅한 부트스트랩 Ignition 구성 파일의 URL을 지정합니다.

부트스트랩 머신에 대한 VM(가상 머신)을 생성할 때 이 Ignition 구성 파일을 사용합니다.

3. 설치 프로그램이 생성한 다음 **Ignition** 구성 파일을 찾습니다.

- **<installation_directory>/master.ign**
- **<installation_directory>/worker.ign**
- **<installation_directory>/merge-bootstrap.ign**

4. **Ignition** 구성 파일을 **Base64** 인코딩으로 변환합니다. 이 절차의 뒷부분에서는 이러한 파일을 **VM**의 추가 구성 매개변수 **guestinfo.ignition.config.data** 에 추가해야 합니다.

예를 들어 **Linux** 운영 체제를 사용하는 경우 **base64** 명령을 사용하여 파일을 인코딩할 수 있습니다.

```
$ base64 -w0 <installation_directory>/master.ign > <installation_directory>/master.64
```

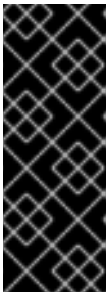
```
$ base64 -w0 <installation_directory>/worker.ign > <installation_directory>/worker.64
```

```
$ base64 -w0 <installation_directory>/merge-bootstrap.ign >
<installation_directory>/merge-bootstrap.64
```

중요

설치를 마친 후 클러스터에 컴퓨팅 시스템을 더 추가하려면 **Ignition** 구성 파일을 삭제하지 마십시오.

5. **RHCOS OVA** 이미지를 가져옵니다. **RHCOS** 이미지 미러 페이지에서 이미지를 사용할 수 있습니다.



중요

RHCOS 이미지는 **OpenShift Container Platform** 릴리스에 따라 변경되지 않을 수 있습니다. 설치하는 **OpenShift Container Platform** 버전과 같거나 그 이하의 버전 중 가장 최신 버전의 이미지를 다운로드해야 합니다. 지원되는 경우 **OpenShift Container Platform** 버전과 일치하는 이미지 버전을 사용합니다.

파일 이름에는 rhcos-vmware.<architecture>.ova 형식의 OpenShift Container Platform 버전 번호가 포함됩니다.

6. vSphere Client에서 VM을 저장할 데이터 센터 폴더를 생성합니다.

   a. VMs and Templates 보기를 클릭합니다.

   b. 데이터 센터 이름을 마우스 오른쪽 버튼으로 클릭합니다.

   c. New Folder → New VM and Template Folder를 클릭합니다.

   d. 표시되는 창에서 폴더 이름을 입력합니다. install-config.yaml 파일에서 기존 폴더를 지정하지 않은 경우 인프라 ID와 동일한 이름으로 폴더를 생성합니다. vCenter는 이 폴더 이름을 사용하여 Workspace 구성에 적절한 위치에 스토리지를 동적으로 프로비저닝합니다.

7. vSphere Client에서 OVA 이미지에 대한 템플릿을 생성한 다음 필요에 따라 템플릿을 복제합니다.

참고

다음 단계에서 템플릿을 생성한 다음 모든 클러스터 시스템에 대한 템플릿을 복제합니다. 그런 다음 VM을 프로비저닝할 때 복제된 머신 유형의 Ignition 구성 파일의 위치를 제공합니다.

   a. Hosts and Clusters 탭에서 클러스터 이름을 마우스 오른쪽 버튼으로 클릭하고 Deploy OVF Template을 선택합니다.

   b. Select an OVF 탭에서 다운로드한 RHCOS OVA 파일의 이름을 지정합니다.

   c. 이름 및 폴더 선택 탭에서 템플릿의 가상 시스템 이름 (예: Template-RHCOS)을 설정합니다. vSphere 클러스터의 이름을 클릭하고 이전 단계에서 생성한 폴더를 선택합니다.

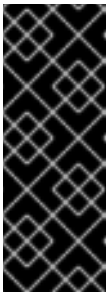
d. **Select a compute resource** 탭에서 **vSphere** 클러스터 이름을 클릭합니다.

e. **Select storage** 탭에서 **VM**의 스토리지 옵션을 구성합니다.

- 스토리지 기본 설정에 따라 **Thin Provision** 또는 **Thick Provision**을 선택합니다.
- **install-config.yaml** 파일에서 지정한 데이터 저장소를 선택합니다.

f. **Select network** 탭에서 사용 가능한 경우 클러스터에 대해 구성한 네트워크를 지정합니다.

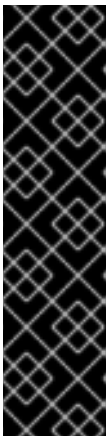
g. **OVF** 템플릿을 생성할 때 템플릿 사용자 지정 탭에 값을 지정하지 않거나 템플릿을 추가로 구성하지 마십시오.



중요

원래 **VM** 템플릿을 시작하지 마십시오. **VM** 템플릿이 꺼져 있어야 하며 새 **RHCOS** 머신에 대해 복제해야 합니다. **VM** 템플릿을 시작하면 **VM** 템플릿이 플랫폼의 **VM**으로 구성되므로 시스템 세트가 구성을 적용할 수 있는 템플릿으로 사용되지 않습니다.

8. 선택 사항: 필요한 경우 **VM** 템플릿에서 구성된 가상 하드웨어 버전을 업데이트합니다. 자세한 내용은 가상 머신을 VMware 설명서의 최신 하드웨어 버전으로 업그레이드를 참조하십시오.



중요

필요한 경우 **VM** 템플릿의 하드웨어 버전을 버전 **15**로 업데이트하는 것이 좋습니다. 이제 **vSphere**에서 실행 중인 클러스터 노드에 하드웨어 버전 **13**을 사용하는 것이 더 이상 사용되지 않습니다. 가져온 템플릿의 기본값이 하드웨어 버전 **13**인 경우 **VM** 템플릿을 하드웨어 버전 **15**로 업그레이드하기 전에 **ESXi** 호스트가 **6.7U3** 이상인지 확인해야 합니다. **vSphere** 버전이 **6.7U3** 미만인 경우 이 업그레이드 단계를 건너뛸 수 있습니다. 그러나 향후 **OpenShift Container Platform** 버전은 **6.7U3** 미만의 하드웨어 버전 **13** 및 **vSphere** 버전에 대한 지원을 제거하도록 예약되어 있습니다.

9. 템플릿이 배포된 후 클러스터에서 시스템의 가상 머신을 배포합니다.

a. 템플릿 이름을 마우스 오른쪽 버튼으로 클릭하고 **Clone → Clone to Virtual Machine**을 클릭합니다.

b. **Select a name and folder** 탭에서 가상 머신의 이름을 지정합니다. **control-plane-0** 또는 **compute-1**과 같은 시스템 유형을 이름에 포함할 수 있습니다.

c. **Select a name and folder** 탭에서 클러스터에 대해 생성한 폴더의 이름을 선택합니다.

d. **Select a compute resource** 탭에서 데이터 센터의 호스트 이름을 선택합니다.

e. 선택사항: **Select storage** 탭에서 스토리지 옵션을 사용자 지정합니다.

f. **Select clone options**에서 **Customize this virtual machine's hardware**를 선택합니다.

g. **Customize hardware** 탭에서 **VM Options → Advanced**를 클릭합니다.

- 선택 사항: **vSphere**에서 기본 **DHCP** 네트워킹을 재정의합니다. 고정 **IP** 네트워킹을 활성화하려면 다음을 수행합니다.

  i. 고정 **IP** 구성을 설정합니다.

```
$ export IPCFG="ip=<ip>::<gateway>:<netmask>:<hostname>:
<iface>:none nameserver=srv1 [nameserver=srv2 [nameserver=srv3 [...]]]"
```

명령 예

```
$ export IPCFG="ip=192.168.100.101::192.168.100.254:255.255.255.0:::none
nameserver=8.8.8.8"
```

ii.

guestinfo.afterburn.initrd.network-kargs 속성을 설정한 후 **vSphere**의 **OVA**에서 **VM**을 부팅합니다.

```
$ govc vm.change -vm "<vm_name>" -e
"guestinfo.afterburn.initrd.network-kargs=${IPCFG}"
```

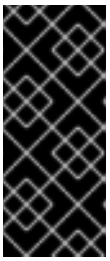
- 선택사항: 클러스터 성능 문제가 발생하는 경우 **Latency Sensitivity** 목록에서 **High**를 선택합니다. **VM**의 **CPU** 및 메모리 예약에 다음과 같은 값이 있는지 확인합니다.
  - 메모리 예약 값은 구성된 메모리 크기와 같아야 합니다.
  - **CPU** 예약 값은 측정된 물리적 **CPU** 속도를 곱한 최소 대기 시간이 짧은 가상 **CPU** 수여야 합니다.
- 구성 편집을 클릭하고 **Configuration Parameters** 창에서 사용 가능한 매개 변수 목록을 검색하여 스틸 클럭 회계 **(stealclock.enable)**를 검색합니다. 사용 가능한 경우 해당 값을 **TRUE** 로 설정합니다. 스틸링 클럭 계정을 사용하면 클러스터 문제 해결에 도움이 될 수 있습니다.
- 구성 추가 매개변수를 클릭합니다. 다음 매개변수 이름 및 값을 정의합니다.
  - **guestinfo.ignition.config.data:** 이 절차에서 이전에 생성한 **base-64** 인코딩 파일을 찾고 이 머신 유형에 대해 **base64**로 인코딩된 **Ignition** 구성 파일의 내용을 붙여넣습니다.
  - **guestinfo.ignition.config.data.encoding: base64**를 지정합니다.
  - **disk.EnableUUID: TRUE**를 지정합니다.
  - **stealclock.enable:**이 매개 변수가 정의되지 않은 경우 추가하고 **TRUE** 를 지정합니다.

h.

Customize hardware 탭의 **Virtual Hardware** 패널에서 지정된 값을 필요에 따라 수정합니다. **RAM, CPU** 및 디스크 스토리지의 양이 시스템 유형에 대한 최소 요구사항을 충족하는지 확인합니다.

i. 구성을 완료하고 **VM**의 전원을 켭니다.

10. 각 시스템에 대해 이전 단계에 따라 클러스터의 나머지 시스템을 생성합니다.



중요

이때 부트스트랩 및 컨트롤 플레인 시스템을 생성해야 합니다. 일부 **Pod**는 기본적으로 컴퓨팅 시스템에 배포되므로 클러스터를 설치하기 전에 컴퓨팅 시스템을 두 개 이상 생성합니다.

17.6.16. vSphere의 클러스터에 더 많은 컴퓨팅 머신 추가

VMware vSphere에서 사용자가 프로비저닝한 **OpenShift Container Platform** 클러스터에 더 많은 컴퓨팅 머신을 추가할 수 있습니다.

사전 요구 사항

- 컴퓨팅 머신의 **base64**로 인코딩된 **Ignition** 파일을 가져옵니다.
- 클러스터에 생성한 **vSphere** 템플릿에 액세스할 수 있습니다.

절차

1. 템플릿이 배포된 후 클러스터에서 시스템의 가상 머신을 배포합니다.

a. 템플릿 이름을 마우스 오른쪽 버튼으로 클릭하고 **Clone → Clone to Virtual Machine**을 클릭합니다.

b. **Select a name and folder** 탭에서 가상 머신의 이름을 지정합니다. **compute-1**과 같은 머신 유형을 이름에 포함할 수 있습니다.

c. **Select a name and folder** 탭에서 클러스터에 대해 생성한 폴더의 이름을 선택합니다.

d. **Select a compute resource** 탭에서 데이터 센터의 호스트 이름을 선택합니다.

e. 선택사항: **Select storage** 탭에서 스토리지 옵션을 사용자 지정합니다.

f. **Select clone options**에서 **Customize this virtual machine's hardware**를 선택합니다.

g. **Customize hardware** 탭에서 **VM Options → Advanced**를 클릭합니다.

- **Latency Sensitivity** 목록에서 **High**를 선택합니다.
- **Edit Configuration**을 클릭하고 **Configuration Parameters** 창에서 **Add Configuration Params**를 클릭합니다. 다음 매개변수 이름 및 값을 정의합니다.
  - **guestinfo.ignition.config.data:** 이 머신 유형에 대해 **base64**로 인코딩된 컴퓨팅 **Ignition** 구성 파일의 내용을 붙여넣습니다.
  - **guestinfo.ignition.config.data.encoding: base64**를 지정합니다.
  - **disk.EnableUUID: TRUE**를 지정합니다.

h. **Customize hardware** 탭의 **Virtual Hardware** 패널에서 지정된 값을 필요에 따라 수정합니다. **RAM, CPU** 및 디스크 스토리지의 양이 시스템 유형에 대한 최소 요구사항을 충족하는지 확인합니다. 또한 사용 가능한 네트워크가 여러 개인 경우 **Add network adapter**에서 올바른 네트워크를 선택해야 합니다.

i. 구성을 완료하고 **VM**의 전원을 켭니다.

2. 계속해서 클러스터에 추가 컴퓨팅 머신을 만듭니다.

17.6.17. 디스크 파티션 설정

대부분의 경우 데이터 파티션은 원래 다른 운영 체제를 설치하는 대신 **RHCOS**를 설치하여 생성됩니다. 이러한 경우 **OpenShift Container Platform** 설치 프로그램은 디스크 파티션을 설정할 수 있어야합니다.

그러나 **OpenShift Container Platform** 노드를 설치할 때 기본 파티션 설정을 덮어 쓰기하기 위해 개입이 필요한 두 가지 경우가 있습니다.

- 별도의 파티션 생성: 빈 디스크에 그린 필드 설치의 경우 파티션에 별도의 스토리지를 추가할 수 있습니다. 이는 **/var** 또는 **/var**의 하위 디렉터리 (예: **/var/lib/etcd**)중 하나를 별도의 파티션으로 만드는 경우에 공식적으로 지원됩니다.

중요

100GB 이상의 디스크 크기 및 특히 **1TB**보다 큰 디스크 크기의 경우 별도의 **/var** 파티션을 만듭니다. 자세한 내용은 " 별도의 **/var** 파티션 생성" 및 이 Red Hat 지식베이스 문서 를 참조하십시오.

중요

Kubernetes는 두 개의 파일 시스템 파티션만 지원합니다. 원래 구성에 둘 이상의 파티션을 추가하는 경우 **Kubernetes**는 모든 파티션을 모니터링할 수 없습니다.

- 기존 파티션 유지: 브라운 필드 설치에서 기존 노드에 **OpenShift Container Platform**을 재설치하고 이전 운영 체제에 설치된 데이터 파티션을 유지해야 하는 경우 기존 데이터 파티션을 유지할 수 있는 **coreos-installer**에 부팅 인수와 옵션이 모두 있습니다.

별도의 **/var** 파티션 만들기

일반적으로 **OpenShift Container Platform**의 디스크 파티션 설정은 설치 프로그램에 맡겨야합니다. 그러나 확장하려는 파일 시스템의 일부에 별도의 파티션을 생성해야 하는 경우가 있습니다.

OpenShift 컨테이너 플랫폼은 **/var** 파티션 또는 **/var**의 하위 디렉터리 중 하나에 스토리지를 연결하는 단일 파티션의 추가를 지원합니다. 예를 들면 다음과 같습니다.

- **/var/lib/containers:** 시스템에 더 많은 이미지와 컨테이너가 추가됨에 따라 확장될 수 있는 컨테이너 관련 콘텐츠를 보관합니다.

- /var/lib/etcd: etcd 스토리지의 성능 최적화와 같은 목적으로 별도로 보관할 데이터를 보관합니다.

- /var: 감사 등의 목적에 맞게 별도로 분리하여 보관해야 하는 데이터를 보관합니다.

중요

100GB보다 큰 디스크 크기 및 특히 1TB보다 큰 디스크의 경우 별도의 /var 파티션을 만듭니다.

/var 디렉터리의 콘텐츠를 별도로 저장하면 필요에 따라 해당 영역에 대한 스토리지 확장을 보다 용이하게 하고 나중에 OpenShift Container Platform을 다시 설치하여 해당 데이터를 그대로 보존할 수 있습니다. 이 방법을 사용하면 모든 컨테이너를 다시 가져올 필요가 없으며 시스템을 업데이트할 때 대용량 로그 파일을 복사할 필요도 없습니다.

RHCOS(Red Hat Enterprise Linux CoreOS)를 새로 설치하기 전에 /var가 있어야 하므로 다음 절차에서는 OpenShift Container Platform 설치의 openshift-install 준비 단계 중에 삽입되는 머신 구성 매니페스트를 생성하여 별도의 /var 파티션을 설정합니다.

프로세스

1. OpenShift Container Platform 설치 파일을 저장할 디렉터리를 만듭니다.

```
$ mkdir $HOME/clusterconfig
```

2. openshift-install을 실행하여 manifest 및 openshift 하위 디렉터리에 파일 세트를 만듭니다. 프롬프트가 표시되면 시스템 질문에 대답합니다.

```
$ openshift-install create manifests --dir $HOME/clusterconfig
? SSH Public Key ...
$ ls $HOME/clusterconfig/openshift/
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

3.

추가 파티션을 구성하는 Butane 구성을 생성합니다. 예를 들어 파일 이름을 $HOME/clusterconfig/98-var-partition.bu 로 지정하고, 디스크 장치 이름을 worker 시스템의 스토리지 장치 이름으로 변경하고 스토리지 크기를 적절하게 설정합니다. 이 예에서는 /var 디렉토리를 별도의 파티션에 배치합니다.

```
variant: openshift
version: 4.9.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1

파티션을 설정해야하는 디스크 저장 장치 이름입니다.

2

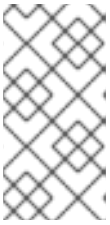
데이터 파티션을 부트 디스크에 추가할 때 최소 25000 메비 바이트가 권장됩니다. 루트 파일 시스템은 지정된 오프셋까지 사용 가능한 모든 공간을 채우기 위해 자동으로 크기가 조정됩니다. 값이 지정되지 않거나 지정된 값이 권장 최소값보다 작으면 생성되는 루트 파일 시스템의 크기가 너무 작아지고 RHCOS를 나중에 다시 설치할 때 데이터 파티션의 첫 번째 부분을 덮어 쓸 수 있습니다.

3

데이터 파티션의 크기(MB)입니다.

4

컨테이너 스토리지에 사용되는 파일 시스템에 prjquota 마운트 옵션을 활성화해야 합니다.



참고

별도의 /var 파티션을 만들 때 다른 인스턴스 유형에 동일한 장치 이름이 없는 경우 작업자 노드에 다른 인스턴스 유형을 사용할 수 없습니다.

4. Butane 구성에서 매니페스트를 생성하여 clusterconfig/openshift 디렉터리에 저장합니다. 예를 들어 다음 명령을 실행합니다.

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o
$HOME/clusterconfig/openshift/98-var-partition.yaml
```

5. openshift-install을 다시 실행하여 manifest 및 openshift 하위 디렉터리의 파일 세트에서 Ignition 구성을 만듭니다.

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

이제 Ignition 구성 파일을 vSphere 설치 절차에 대한 입력으로 사용하여 RHCOS (Red Hat Enterprise Linux CoreOS) 시스템을 설치할 수 있습니다.

17.6.18. bootupd를 사용하여 부트로더 업데이트

bootupd를 사용하여 부트로더를 업데이트하려면 RHCOS 머신에 수동으로 bootupd를 설치하거나 systemd 유닛이 활성화된 머신 구성을 제공해야 합니다. grubby 또는 기타 부트로더 툴과는 달리 bootupd는 커널 인수 전달과 같은 커널 공간 구성을 관리하지 않습니다.

bootupd를 설치한 후 OpenShift Container Platform 클러스터에서 원격으로 관리할 수 있습니다.

참고

bootupd는 BootHole 취약점에 대한 보호와 같이 베어 메탈 또는 가상화 하이퍼바이저 설치에서만 사용하는 것이 좋습니다.

수동 설치 방법

bootctl 명령줄 툴을 사용하여 bootupd를 수동으로 설치할 수 있습니다.

1. 시스템 상태를 검사합니다.

```
# bootupctl status
```

출력 예

```
Component EFI
  Installed: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
  Update: At latest version
```

2. bootupd를 설치하지 않고 생성된 RHCOS 이미지에는 명시적 채택 단계가 필요합니다.

   시스템 상태가 Adoptable인 경우 채택을 수행합니다.

```
# bootupctl adopt-and-update
```

출력 예

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

3. 업데이트를 사용할 수 있는 경우 다음 재부팅에 변경 사항이 적용되도록 업데이트를 적용합니다.

```
# bootupctl update
```

출력 예

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

시스템 구성 방법

bootupd를 활성화하는 또 다른 방법은 머신 구성을 제공하는 것입니다.

- 다음 예와 같이 **systemd** 단위가 활성화된 머신 구성 파일을 제공합니다.

출력 예

```
variant: rhcos
version: 1.1.0
systemd:
  units:
    - name: custom-bootupd-auto.service
      enabled: true
      contents: |
        [Unit]
        Description=Bootupd automatic update

        [Service]
        ExecStart=/usr/bin/bootupctl update
        RemainAfterExit=yes

        [Install]
        WantedBy=multi-user.target
```

17.6.19. 부트스트랩 프로세스가 완료될 때까지 대기 중

OpenShift Container Platform 부트스트랩 프로세스는 클러스터 노드가 먼저 디스크에 설치된 영구 **RHCOS** 환경으로 부팅된 후에 시작됩니다. **Ignition** 구성 파일을 통해 제공되는 구성 정보는 부트스트랩 프로세스를 초기화하고 머신에 **OpenShift Container Platform**을 설치하는 데 사용됩니다. 부트스트랩 프로세스가 완료될 때까지 기다려야 합니다.

사전 요구 사항

- 클러스터에 대한 **Ignition** 구성 파일이 생성되어 있습니다.

- 적합한 네트워크, DNS 및 로드 밸런싱 인프라가 구성되어 있습니다.
- 설치 프로그램을 받아서 클러스터의 Ignition 구성 파일을 생성했습니다.
- 클러스터 머신에 RHCOS를 설치하고 OpenShift Container Platform 설치 프로그램에서 생성된 Ignition 구성 파일을 제공했습니다.
- 사용자 시스템에서 직접 인터넷에 액세스하거나 HTTP 또는 HTTPS 프록시를 사용할 수 있습니다.

프로세스

1. 부트스트랩 프로세스를 모니터링합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2 다른 설치 세부 사항을 보려면 info 대신 warn, debug 또는 error를 지정합니다.

출력 예

```
INFO Waiting up to 30m0s for the Kubernetes API at
https://api.test.example.com:6443...
INFO API v1.22.1 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

이 명령은 Kubernetes API 서버가 컨트롤 플레인 시스템에서 부트스트랩되었다는 신호를 보낼 때 성공합니다.

2. 부트스트랩 프로세스가 완료된 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다.

중요

이 시점에 로드 밸런서에서 부트스트랩 시스템을 제거해야 합니다. 부트스트랩 머신 자체를 제거하거나 다시 포맷할 수도 있습니다.

17.6.20. CLI를 사용하여 클러스터에 로그인

클러스터 kubeconfig 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. kubeconfig 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- oc CLI를 설치했습니다.

프로세스

1. kubeadmin 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 oc 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

17.6.21. 머신의 인증서 서명 요청 승인

클러스터에 시스템을 추가하면 추가한 시스템별로 보류 중인 인증서 서명 요청(CSR)이 두 개씩 생성됩니다. 이러한 CSR이 승인되었는지 확인해야 하며, 필요한 경우 이를 직접 승인해야 합니다. 클라이언트 요청을 먼저 승인한 다음 서버 요청을 승인해야 합니다.

사전 요구 사항

- 클러스터에 시스템을 추가했습니다.

절차

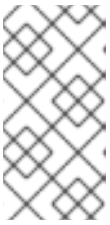
1. 클러스터가 시스템을 인식하는지 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0  Ready    master  63m  v1.22.1
master-1  Ready    master  63m  v1.22.1
master-2  Ready    master  64m  v1.22.1
```

출력에 생성된 모든 시스템이 나열됩니다.



참고

이전 출력에는 일부 **CSR**이 승인될 때까지 컴퓨팅 노드(작업자 노드라고도 함)가 포함되지 않을 수 있습니다.

2.

보류 중인 **CSR**을 검토하고 클러스터에 추가한 각 시스템에 대해 **Pending** 또는 **Approved** 상태의 클라이언트 및 서버 요청이 표시되는지 확인합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE    REQUESTOR                                                                   CONDITION
csr-8b2br   15m    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

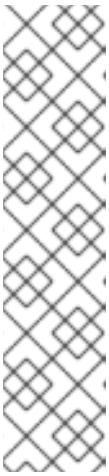
예에서는 두 시스템이 클러스터에 참여하고 있습니다. 목록에는 승인된 **CSR**이 더 많이 나타날 수도 있습니다.

3.

CSR이 승인되지 않은 경우, 추가된 시스템에 대한 모든 보류 중인 **CSR**이 **Pending** 상태로 전환된 후 클러스터 시스템의 **CSR**을 승인합니다.

참고

CSR은 교체 주기가 자동으로 만료되므로 클러스터에 시스템을 추가한 후 **1** 시간 이내에 **CSR**을 승인하십시오. 한 시간 내에 승인하지 않으면 인증서가 교체되고 각 노드에 대해 두 개 이상의 인증서가 표시됩니다. 이러한 인증서를 모두 승인해야 합니다. 클라이언트 **CSR**이 승인되면 **Kubelet**은 인증서에 대한 보조 **CSR**을 생성하므로 수동 승인이 필요합니다. 그러면 **Kubelet**에서 동일한 매개변수를 사용하여 새 인증서를 요청하는 경우 인증서 갱신 요청은 **machine-approver**에 의해 자동으로 승인됩니다.



참고

베어 메탈 및 기타 사용자 프로비저닝 인프라와 같이 머신 **API**를 사용하도록 활성화되지 않는 플랫폼에서 실행되는 클러스터의 경우 **CSR(Kubelet service Certificate Request)**을 자동으로 승인하는 방법을 구현해야 합니다. 요청이 승인되지 않으면 **API** 서버가 **kubelet**에 연결될 때 서비스 인증서가 필요하므로 **oc exec, oc rsh, oc logs** 명령을 성공적으로 수행할 수 없습니다. **Kubelet** 엔드 포인트에 연결하는 모든 작업을 수행하려면 이 인증서 승인이 필요합니다. 이 방법은 새 **CSR**을 감시하고 **CSR**이 **system:node** 또는 **system:admin** 그룹의 **node-bootstrapper** 서비스 계정에 의해 제출되었는지 확인하고 노드의 **ID**를 확인합니다.

- 개별적으로 승인하려면 유효한 **CSR** 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1

<csr_name>은 현재 **CSR** 목록에 있는 **CSR**의 이름입니다.

- 보류 중인 **CSR**을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

참고

일부 **Operator**는 일부 **CSR**이 승인될 때까지 사용할 수 없습니다.

4. 이제 클라이언트 요청이 승인되었으므로 클러스터에 추가한 각 머신의 서버 요청을 검토해야 합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
```

```
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 나머지 CSR이 승인되지 않고 Pending 상태인 경우 클러스터 머신의 CSR을 승인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 <csr_name>은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

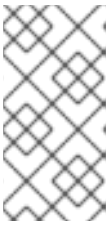
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs oc adm certificate approve
```

6. 모든 클라이언트 및 서버 CSR이 승인된 후 머신은 Ready 상태가 됩니다. 다음 명령을 실행하여 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0  Ready    master  73m  v1.22.1
master-1  Ready    master  73m  v1.22.1
master-2  Ready    master  74m  v1.22.1
worker-0  Ready    worker  11m  v1.22.1
worker-1  Ready    worker  11m  v1.22.1
```



참고

머신이 **Ready** 상태로 전환하는 데 서버 **CSR**의 승인 후 몇 분이 걸릴 수 있습니다.

추가 정보

- **CSR**에 대한 자세한 내용은 인증서 서명 요청을 참조하십시오.

17.6.21.1. Operator의 초기 설정

컨트롤 플레인이 초기화된 후 일부 **Operator**를 즉시 구성하여 모두 사용 가능하도록 해야 합니다.

사전 요구 사항

- 컨트롤 플레인이 초기화되어 있습니다.

절차

1. 클러스터 구성 요소가 온라인 상태인지 확인합니다.

```
$ watch -n5 oc get clusteroperators
```

출력 예

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.9.0     True        False         False      19m
baremetal                                  4.9.0     True        False         False      37m
cloud-credential                           4.9.0     True        False         False      40m
cluster-autoscaler                         4.9.0     True        False         False      37m
config-operator                            4.9.0     True        False         False      38m
console                                    4.9.0     True        False         False      26m
csi-snapshot-controller                    4.9.0     True        False         False      37m
dns                                        4.9.0     True        False         False      37m
etcd                                       4.9.0     True        False         False      36m
image-registry                             4.9.0     True        False         False      31m
ingress                                    4.9.0     True        False         False      30m
insights                                   4.9.0     True        False         False      31m
kube-apiserver                             4.9.0     True        False         False      26m
kube-controller-manager                    4.9.0     True        False         False      36m
kube-scheduler                             4.9.0     True        False         False      36m
```

```
kube-storage-version-migrator              4.9.0     True        False         False      37m
machine-api                                4.9.0     True        False         False      29m
machine-approver                           4.9.0     True        False         False      37m
machine-config                             4.9.0     True        False         False      36m
marketplace                                4.9.0     True        False         False      37m
monitoring                                 4.9.0     True        False         False      29m
network                                    4.9.0     True        False         False      38m
node-tuning                                4.9.0     True        False         False      37m
openshift-apiserver                        4.9.0     True        False         False      32m
openshift-controller-manager               4.9.0     True        False         False      30m
openshift-samples                          4.9.0     True        False         False      32m
operator-lifecycle-manager                 4.9.0     True        False         False      37m
operator-lifecycle-manager-catalog         4.9.0     True        False         False      37m
operator-lifecycle-manager-packageserver   4.9.0     True        False         False      32m
service-ca                                 4.9.0     True        False         False      38m
storage                                    4.9.0     True        False         False      37m
```

2. 사용할 수 없는 **Operator**를 구성합니다.

17.6.21.2. 설치 중 제거된 이미지 레지스트리

공유 가능한 개체 스토리지를 제공하지 않는 플랫폼에서 **OpenShift Image Registry Operator**는 자체적으로 **Removed**로 부트스트랩합니다. 이를 통해 **openshift-installer**가 이러한 플랫폼 유형에서 설치를 완료할 수 있습니다.

설치 후 **managementState**를 **Removed**에서 **Managed**로 전환하도록 **Image Registry Operator** 구성을 편집해야 합니다.

참고

Prometheus 콘솔은 **ImageRegistryRemoved** 경고를 제공합니다. 다음은 예시 경고입니다.

"이미지 레지스트리가 제거되었습니다. **ImageStreamTags**를 참조하는 **ImageStreamTags, BuildConfigs** 및 **DeploymentConfigs**가 예상대로 작동하지 않을 수 있습니다. 스토리지를 구성하고 **configs.imageregistry.operator.openshift.io**를 편집하여 구성을 **Managed** 상태로 업데이트하십시오."

17.6.21.3. 이미지 레지스트리 스토리지 구성

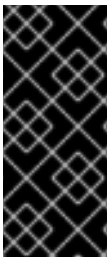
기본 스토리지를 제공하지 않는 플랫폼에서는 처음에 **Image Registry Operator**를 사용할 수 없습니다. 설치한 후에 스토리지를 사용하도록 레지스트리를 구성하여 **Registry Operator**를 사용 가능하도록 만들어야 합니다.

프로덕션 클러스터에 필요한 영구 볼륨을 구성하는 과정의 지침이 표시됩니다. 해당하는 경우, 프로덕션 환경 외 클러스터에서만 사용할 수 있는 저장 위치로서 빈 디렉터리를 구성하는 과정의 지침이 표시됩니다.

업그레이드 중에 **Recreate** 롤아웃 전략을 사용하여 이미지 레지스트리의 블록 스토리지 유형 사용을 허용하기 위한 추가 지침이 제공됩니다.

17.6.21.3.1. VMware vSphere용 블록 레지스트리 스토리지 구성

클러스터 관리자로서 업그레이드 중에 이미지 레지스트리가 **vSphere VMDK(Virtual Machine Disk)**와 같은 블록 스토리지 유형을 사용할 수 있도록 허용하기 위해 **Recreate** 롤아웃 전략을 사용할 수 있습니다.



중요

블록 스토리지 볼륨이 지원되지만 프로덕션 클러스터에서 이미지 레지스트리와 함께 사용하는 것은 권장되지 않습니다. 레지스트리가 블록 스토리지에 구성된 설치는 레지스트리가 둘 이상의 복제본을 가질 수 없기 때문에 가용성이 높지 않습니다.

절차

1. 이미지 레지스트리 스토리지를 블록 스토리지 유형으로 설정하려면 **Recreate** 롤아웃 전략을 사용하고 복제본 **1**개 만으로 실행되도록 레지스트리를 패치합니다.

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. 블록 스토리지 장치에 **PV**를 프로비저닝하고 해당 볼륨의 **PVC**를 생성합니다. 요청된 블록 볼륨은 **RWO(ReadWriteOnce)** 액세스 모드를 사용합니다.

   a. **VMware vSphere PersistentVolumeClaim** 개체를 정의하려면 다음 내용이 포함된 **pvc.yaml** 파일을 생성합니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
```

```
  name: image-registry-storage (1)
  namespace: openshift-image-registry (2)
spec:
  accessModes:
  - ReadWriteOnce (3)
  resources:
    requests:
      storage: 100Gi (4)
```

(1) PersistentVolumeClaim 개체를 표시하는 고유한 이름입니다.

(2) PersistentVolumeClaim 오브젝트의 네임스페이스로 openshift-image-registry입니다.

(3) 영구 볼륨 클레임의 액세스 모드입니다. ReadWriteOnce를 사용하면 단일 노드에서 읽기 및 쓰기 권한으로 볼륨을 마운트할 수 있습니다.

(4) 영구 볼륨 클레임의 크기입니다.

b. 파일에서 PersistentVolumeClaim 오브젝트를 만듭니다.

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 올바른 PVC를 참조하도록 레지스트리 설정을 편집합니다.

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

출력 예

```
storage:
  pvc:
    claim: (1)
```

1. 사용자 지정 PVC를 만들면 image-registry-storage PVC의 기본 자동 생성을 위해 claim 필드를 비워둘 수 있습니다.

올바른 PVC를 참조하도록 레지스트리 스토리지를 구성하는 방법은 vSphere 용 레지스트리 구성을 참조하십시오.

17.6.22. 사용자 프로비저닝 인프라에 설치 완료

Operator 구성을 완료한 후 제공하는 인프라에 클러스터 설치를 완료할 수 있습니다.

사전 요구 사항

- 컨트롤 플레인이 초기화되어 있습니다.
- 초기 Operator 구성을 완료해야 합니다.

프로세스

1. 다음 명령을 사용하여 모든 클러스터 구성 요소가 온라인 상태인지 확인합니다.

```
$ watch -n5 oc get clusteroperators
```

출력 예

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.9.0     True        False         False      19m
baremetal                                  4.9.0     True        False         False      37m
cloud-credential                           4.9.0     True        False         False      40m
cluster-autoscaler                         4.9.0     True        False         False      37m
config-operator                            4.9.0     True        False         False      38m
console                                    4.9.0     True        False         False      26m
csi-snapshot-controller                    4.9.0     True        False         False      37m
dns                                        4.9.0     True        False         False      37m
```

```
etcd                                       4.9.0     True        False         False      36m
image-registry                             4.9.0     True        False         False      31m
ingress                                    4.9.0     True        False         False      30m
insights                                   4.9.0     True        False         False      31m
kube-apiserver                             4.9.0     True        False         False      26m
kube-controller-manager                    4.9.0     True        False         False      36m
kube-scheduler                             4.9.0     True        False         False      36m
kube-storage-version-migrator              4.9.0     True        False         False      37m
machine-api                                4.9.0     True        False         False      29m
machine-approver                           4.9.0     True        False         False      37m
machine-config                             4.9.0     True        False         False      36m
marketplace                                4.9.0     True        False         False      37m
monitoring                                 4.9.0     True        False         False      29m
network                                    4.9.0     True        False         False      38m
node-tuning                                4.9.0     True        False         False      37m
openshift-apiserver                        4.9.0     True        False         False      32m
openshift-controller-manager               4.9.0     True        False         False      30m
openshift-samples                          4.9.0     True        False         False      32m
operator-lifecycle-manager                 4.9.0     True        False         False      37m
operator-lifecycle-manager-catalog         4.9.0     True        False         False      37m
operator-lifecycle-manager-packageserver   4.9.0     True        False         False      32m
service-ca                                 4.9.0     True        False         False      38m
storage                                    4.9.0     True        False         False      37m
```

또는 다음 명령은 모든 클러스터를 사용할 수 있을 때 알립니다. 또한 인증 정보를 검색하고 표시합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1

<installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

출력 예

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator가 Kubernetes API 서버에서 OpenShift Container Platform 클러스터 배포를 완료하면 명령이 성공합니다.

중요

- 설치 프로그램에서 생성하는 **Ignition** 구성 파일에 **24**시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 **24**시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 **kubelet** 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청**(CSR)**을 수동으로 승인해야 합니다. 자세한 내용은 *만료된 컨트롤 플레인 인증서에서 복구* 문서를 참조하십시오.

- 클러스터를 설치한 후 **24**시간에서 **22**시간까지의 인증서가 교체되기 때문에 생성된 후 **12**시간 이내에 **Ignition** 구성 파일을 사용하는 것이 좋습니다. **12**시간 이내에 **Ignition** 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

2. **Kubernetes API** 서버가 **Pod**와 통신하고 있는지 확인합니다.

   a. 모든 **Pod** 목록을 보려면 다음 명령을 사용하십시오.

   ```
   $ oc get pods --all-namespaces
   ```

   출력 예

   ```
   NAMESPACE                       NAME                                        READY   STATUS
   RESTARTS   AGE
   openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8
   1/1     Running   1          9m
   openshift-apiserver             apiserver-67b9g                             1/1     Running   0
   3m
   openshift-apiserver             apiserver-ljcmx                             1/1     Running   0
   1m
   openshift-apiserver             apiserver-z25h4                             1/1     Running   0
   2m
   openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8
   1/1     Running   0          5m
   ...
   ```

   b. 다음 명령을 사용하여 이전 명령의 출력에 나열된 **Pod**의 로그를 표시합니다.

```
$ oc logs <pod_name> -n <namespace> 1
```

1

이전 명령의 출력에 표시된 대로 **Pod** 이름과 네임스페이스를 지정합니다.

Pod 로그가 표시되면 **Kubernetes API** 서버는 클러스터 시스템과 통신할 수 있습니다.

3. **FCP(Fibre Channel Protocol)**를 사용하는 설치에는 다중 경로를 활성화하기 위해 추가 단계가 필요합니다. 설치 중에 멀티패스를 활성화하지 마십시오.

   자세한 내용은 *설치 후 머신 구성 작업* 설명서에서 **"RHCOS에서 커널 인수를 사용하여 멀티패스 활성화"**를 참조하십시오.

vSphere에 컴퓨팅 머신 추가에 따라 클러스터 설치가 완료된 후 추가 컴퓨팅 머신을 추가할 수 있습니다.

17.6.23. VMware vSphere 볼륨 백업

OpenShift Container Platform은 새 볼륨을 독립 영구 디스크로 프로비저닝하여 클러스터의 모든 노드에서 볼륨을 자유롭게 연결 및 분리합니다. 결과적으로 스냅샷을 사용하는 볼륨을 백업하거나 스냅샷에서 볼륨을 복원할 수 없습니다. 자세한 내용은 **스냅 샷 제한** 을 참조하십시오.

프로세스

영구 볼륨의 백업을 생성하려면 다음을 수행합니다.

1. 영구 볼륨을 사용중인 애플리케이션을 중지합니다.
2. 영구 볼륨을 복제합니다.
3. 애플리케이션을 다시 시작합니다.
4. 복제된 볼륨의 백업을 만듭니다.

5. 복제된 볼륨을 삭제합니다.

17.6.24. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

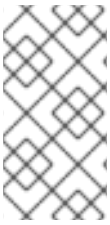
- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

17.6.25. 다음 단계

- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.
- 레지스트리를 설정하고 레지스트리 스토리지를 구성합니다.
- 선택 사항: vSphere Problem Detector Operator에서 이벤트를 보고 클러스터에 권한 또는 스토리지 구성 문제가 있는지 확인합니다.

17.7. 제한된 네트워크에서 VSPHERE에 클러스터 설치

OpenShift Container Platform 4.9에서는 설치 릴리스 콘텐츠의 내부 미러를 생성하여 제한된 네트워크의 VMware vSphere 인프라에 클러스터를 설치할 수 있습니다.

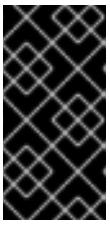


참고

OpenShift Container Platform은 단일 VMware vCenter에만 클러스터 배포를 지원합니다. 여러 vCenter에 머신/머신 세트가 있는 클러스터 배포는 지원되지 않습니다.

17.7.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.
- 미러 호스트에 레지스트리를 생성하고 사용 중인 OpenShift Container Platform 버전의 imageContentSources 데이터를 가져옵니다.



중요

미러 호스트에 설치 미디어가 있으므로 해당 컴퓨터를 사용하여 모든 설치 단계를 완료하십시오.

- 클러스터용 영구 스토리지를 프로비저닝합니다. 프라이빗 이미지 레지스트리를 배포하려면 스토리지에서 ReadWriteMany 액세스 모드를 제공해야 합니다.
- OpenShift Container Platform 설치 프로그램은 vCenter 및 ESXi 호스트의 포트 443에 액세스해야 합니다. 포트 443에 액세스할 수 있는지 확인했습니다.
- 방화벽을 사용하는 경우 관리자가 포트 443에 액세스할 수 있음을 확인합니다. 설치에 성공하려면 포트 443에서 컨트롤 플레인 노드가 vCenter 및 ESXi 호스트에 연결할 수 있어야 합니다.
- 방화벽을 사용하며 Telemetry 서비스를 사용할 예정인 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성해야 합니다.

참고

프록시를 구성하는 경우 해당 사이트 목록도 검토해야 합니다.

17.7.2. 네트워크가 제한된 환경에서의 설치 정보

OpenShift Container Platform 4.9에서는 소프트웨어 구성 요소를 받기 위한 인터넷 접속이 필요하지 않은 설치를 수행할 수 있습니다. 제한된 네트워크 설치는 클러스터를 설치하는 클라우드 플랫폼에 따라 설치 관리자 프로비저닝 인프라 또는 사용자 프로비저닝 인프라를 사용하여 완료할 수 있습니다.

클라우드 플랫폼에 제한된 네트워크 설치를 수행하는 방법을 선택해도 클라우드 **API**에 액세스는 가능해야 합니다. **Amazon Web Service**의 **Route 53 DNS** 및 **IAM** 서비스와 같은 일부 클라우드 기능에는 인터넷 액세스가 필요합니다. 사용 중인 네트워크에 따라 베어메탈 하드웨어 또는 **VMware vSphere**에 설치하기 위해 필요한 인터넷 액세스가 줄어들 수 있습니다.

제한된 네트워크 설치를 완료하려면 **OpenShift Container Platform** 레지스트리의 내용을 미러링하고 설치 미디어를 포함할 레지스트리를 생성해야 합니다. 인터넷과 폐쇄 네트워크에 모두 액세스하거나 제한 사항을 따르는 다른 방법을 통해 미러 호스트에 레지스트리를 생성할 수 있습니다.

17.7.2.1. 추가 제한

제한된 네트워크의 클러스터에는 다음과 같은 추가 제한이 있습니다.

- **ClusterVersion** 상태에 사용 가능한 업데이트를 검색할 수 없음 오류가 포함되어 있습니다.
- 기본적으로 필요한 이미지 스트림 태그에 액세스할 수 없기 때문에 개발자 카탈로그의 내용을 사용할 수 없습니다.

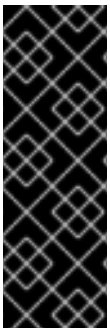
17.7.3. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하기 위해 필요한 이미지를 받으려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 **Telemetry** 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.

- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.



중요

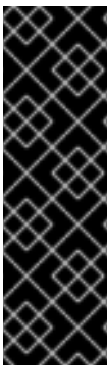
클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

17.7.4. VMware vSphere 인프라 요구사항

사용하는 구성 요소의 요구사항을 충족하는 VMware vSphere 버전 6 또는 7 인스턴스에 OpenShift Container Platform 클러스터를 설치해야 합니다.

표 17.63. vSphere 가상 환경의 버전 요구 사항

가상 환경 제품	필요한 버전
VM 하드웨어 버전	13 이상
vSphere ESXi hosts	6.5 이상
vCenter 호스트	6.5 이상



중요

이제 VMware vSphere 버전 6.7U2 또는 이전 버전에 클러스터를 설치하는 것이 더 이상 사용되지 않으며 가상 하드웨어 버전 13은 더 이상 사용되지 않습니다. 이러한 버전은 여전히 완전히 지원되지만 향후 OpenShift Container Platform 버전에서는 지원이 제거됩니다. 이제 OpenShift Container Platform의 vSphere 가상 머신의 하드웨어 버전 15가 기본값이 되었습니다. vSphere 노드의 하드웨어 버전을 업데이트하려면 " vSphere에서 실행되는 노드에서 하드웨어 업데이트" 문서를 참조하십시오.

표 17.64. VMware 구성 요소에 지원되는 최소 vSphere 버전

구성 요소	지원되는 최소 버전	설명
하이퍼바이저	HW 버전 13이 포함된 vSphere 6.5 이상	이 버전은 RHCOS(Red Hat Enterprise Linux CoreOS)가 지원하는 최소 버전입니다. Red Hat Enterprise Linux 8 지원 하이퍼바이저 목록을 참조하십시오.

구성 요소	지원되는 최소 버전	설명
In-tree 드라이버가 있는 스토리지	vSphere 6.5 이상	이 플러그인은 OpenShift Container Platform에 포함된 vSphere용 인트리 스토리지 드라이버를 사용하여 vSphere 스토리지를 생성합니다.
선택사항: 네트워킹(NSX-T)	vSphere 6.5U3 또는 vSphere 6.7U2 이상	OpenShift Container Platform에는 vSphere 6.5U3 또는 vSphere 6.7U2+가 필요합니다. NSX 및 OpenShift Container Platform의 호환성에 대한 자세한 내용은 VMware의 NSX 컨테이너 플러그인 설명서의 릴리스 노트 섹션을 참조하십시오.

vSphere 버전 6.5 인스턴스를 사용하는 경우 OpenShift Container Platform을 설치하기 전에 6.7U3 또는 7.0로 업그레이드하는 것이 좋습니다.

중요

OpenShift Container Platform을 설치하기 전에 ESXi 호스트의 시간이 동기화되었는지 확인해야 합니다. VMware 문서에서 Edit Time Configuration for a Host를 참조하십시오.

17.7.5. 네트워크 연결 요구사항

OpenShift Container Platform 클러스터 구성 요소가 통신할 수 있도록 시스템 간 네트워크 연결을 구성해야 합니다.

필요한 네트워크 포트에 대한 다음 세부 정보를 검토합니다.

표 17.65. 모든 시스템 간 통신에 사용되는 포트

프로토콜	포트	설명
ICMP	해당 없음	네트워크 연결성 테스트
TCP	**1936**	메트릭
	9000-9999	**9100-9101** 포트의 노드 내보내기 및 **9099** 포트의 Cluster Version Operator를 포함한 호스트 수준 서비스.

프로토콜	포트	설명
	10250-10259	Kubernetes에서 예약하는 기본 포트
	10256	openshift-sdn
UDP	**4789**	VXLAN(Virtual extensible LAN)
	6081	Geneve
	9000-9999	**9100-9101** 포트의 노드 내보내기를 포함한 호스트 수준 서비스.
	500	IPsec IKE 패킷
	4500	IPsec NAT-T 패킷
TCP/UDP	**30000-32767**	Kubernetes 노드 포트
ESP	해당 없음	IPsec Encapsulating Security Payload (ESP)

표 17.66. 모든 시스템과 컨트롤 플레인 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**6443**	Kubernetes API

표 17.67. 컨트롤 플레인 머신 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**2379-2380**	etcd 서버 및 피어 포트

추가 리소스

- **vSphere 노드의 하드웨어 버전을 업데이트하려면 vSphere에서 실행 중인 노드에서 하드웨어 업데이트를 참조하십시오.**

17.7.6. vCenter 요구사항

설치 관리자가 프로비저닝하는 인프라를 사용하는 vCenter에 OpenShift Container Platform 클러스터를 설치하기 전에 환경을 준비해야 합니다.

필요한 vCenter 계정 권한

vCenter에 OpenShift Container Platform 클러스터를 설치하려면 설치 프로그램에서 필요한 리소스를 읽고 생성할 수 있는 권한이 있는 계정에 액세스해야 합니다. 필요한 모든 권한에 액세스할 수 있는 가장 간단한 방법은 글로벌 관리 권한이 있는 계정을 사용하는 것입니다.

글로벌 관리 권한이 있는 계정을 사용할 수 없는 경우 **OpenShift Container Platform** 클러스터 설치에 필요한 권한을 부여하려면 역할을 생성해야 합니다. 대부분의 권한이 항상 필요하지만 기본 동작인 **vCenter** 인스턴스에 **OpenShift Container Platform** 클러스터를 포함하는 폴더를 프로비저닝할 설치 프로그램인 경우에만 필요합니다. 필요한 권한을 부여하려면 지정된 오브젝트에 대해 **vSphere** 역할을 생성하거나 수정해야 합니다.

설치 프로그램이 **vSphere** 가상 머신 폴더를 생성하는 경우 추가 역할이 필요합니다.

예 17.16. vSphere API에 설치에 필요한 역할 및 권한

역할의 vSphere 개체	필요한 경우	vSphere API에서 필요한 권한
vSphere vCenter	Always	CNS **.Searchable InventoryService.Tagging.AttachTag InventoryService.Tagging.CreateCategory InventoryService.Tagging.CreateTag InventoryService.Tagging.Delete**Tag **InventoryService. tagged.DeleteTag InventoryService.Tagging.EditCategory InventoryService.Tagging.EditTag**ECDHE**Session ValidateSession**Forwarded**StorageProfile.Update**Forwarded**StorageProfile.View**
vSphere vCenter Cluster	Always	**Host.Config.Storage Resource.AssignVMToPool VApp.AssignResourcePool VApp.Import VirtualMachine.Config.AddNewDisk**

역할의 vSphere 개체	필요한 경우	vSphere API에서 필요한 권한
vSphere Datastore	Always	**Datastore.AllocateSpace Datastore.LoadBalancer Datastore.FileManagement InventoryService.Tagging.ObjectAttachable**
vSphere Port Group	Always	**Network.Assign**

역할의 vSphere 개체	필요한 경우	vSphere API에서 필요한 권한
가상 머신 폴더	Always	**InventoryService.ume.ObjectAttachable Resource.AssignVMToPool VApp.Import VirtualMachine.Config.Add ExistingDisk VirtualMachine.Config.Add NewDisk VirtualMachine.Config.Add RemoveDevice VirtualMachine.Config.Advanced Config VirtualMachine.Config.Annotation ECDHEVirtualMachine .Config.DiskExtend** Forwarded VirtualMachine**.Config.** DiskExtend Forwarded**VirtualMachine.Config.DiskLease VirtualMachine.Config.ReCreate** VolumeClaim.Config**.MemoryDriver.**

역할의 vSphere 개체	필요한 경우	vSphere API에서 필요한 권한
vSphere vCenter Datacenter	설치 프로그램이 가상 머신 폴더를 생성하는 경우	**InventoryService.Tagging.ObjectAttachable Resource.AssignVMToPool VApp.Import VirtualMachine.Config.Add ExistingDisk VirtualMachine.Config.Add NewDisk VirtualMachine.Config.Add RemoveDevice VirtualMachine.Config.Adva ncedConfig VirtualMachine.Config.Anno tationECDHEVirtualMachine .Config.DiskExtend**Forwarde d VirtualMachine**.**Config**.DiskExt end ForwardedVirtualMachine.C onfig.DiskLeaseVirtualMach ine.Config.ReCreate VolumeClaim.Config.Reset**defined**.**

예 17.17. vCenter 그래픽 사용자 인터페이스(GUI)에 설치하는 데 필요한 역할 및 권한

역할의 vSphere 개체	필요한 경우	vCenter GUI에서 필요한 권한
vSphere vCenter	Always	CNS .**Searchable "vSphere agged""Assign or unassign vSphere Tag" "vSphere tagging" "vSphere" "vSphere" "vSphere"CreatevSphere Tag" vSphere 태그""vSphere" 삭제 태그 지정" "vSphere Assignment""Edit vSphere Tag" "vSphere Tagging" "vSphere 태그" 세션" "Profile 기반 스토리지""Profile 기반 스토리지 업데이트""Profile 기반 스토리지"Profile 중심 스토리지 업데이트"Profile 기반 스토리지 업데이트"Profile 기반 스토리지 업데이트 스토리지 보기"**
vSphere vCenter Cluster	클러스터 루트에서 VM이 생성되는 경우	**Host.Configuration"Storage partition configuration" 리소스" 리소스 풀에 가상 머신을 서명" VApp"Assign resource pool" VApp.Import"Virtual machine""Change Configuration"."새 디스크 추가"**
vSphere vCenter 리소스 풀	기존 리소스 풀이 제공되는 경우	**Host.Configuration"Storage partition configuration" 리소스" 리소스 풀에 가상 머신을 서명" VApp"Assign resource pool" VApp.Import"Virtual machine""Change Configuration"."새 디스크 추가"**

역할의 vSphere 개체	필요한 경우	vCenter GUI에서 필요한 권한
vSphere Datastore	Always	**Datastore"Allocate space"** 데이터**저장소"** **데이터 저장소"** **Datastore"Low level file operations""vSphere Assignment"Assign or Unassign vSphere Tag on Object"**
vSphere Port Group	Always	**네트워크 "Assign network"**
가상 머신 폴더	Always	**"vSphere Assignment""Assign or unassign vSphere Tag on Object"** **Resource"Assign virtual machine to resource pool"**: **VApp.Import** **"Virtual machine""Change Configuration","Add existing disk""**Add existing disk**""Add new disk""Add new disk" "Virtual""** **머신 "시스템 변경", "가상 머신""** **"가상 머신""시스템 변경"** **"가상 머신""변경 구성",** **"가상 머신", 변경, CPU 수 변경"** **"가상 머신", 가상 디스크 변경", 가상 디스크 변경"** **"가상머신". 머신 변경 "구성 변경"** **"가상 머신""마이크 구성 변경"** **"가상 머신""마이크 구성 변경" "구성 변경",** **"가상 머신", 변경" 디스크 삭제"** **"가상 머신".변경 구성".Rename** **"가상 머신 변경" VIX API**로가상 머신 변경**"Gregradevirtual machine"Guestoperating system management by VIX API**

역할의 vSphere 개체	필요한 경우	vCenter GUI에서 필요한 권한
		machine". interaction" "Virtual machine". Interconnection"Power on" "Virtual machine". interactionion.Reset "Virtual machine"."Edit Inventory"."Create new" "Virtual machine"."Create existing" "Virtual machine"Edit Inventory""Create from existing" "Virtual machine"Edit Inventory""Create from existing" virtual machine".Edit Inventory""Create from existing" virtual machine".Edit Inventory""Create from existing""Virtual machine" machine".Provisioning."Clone 가상 머신" "가상 머신".Provisioning." "가상 머신".Provisioning."Deploy 템플릿"
vSphere vCenter Datacenter	설치 프로그램이 가상 머신 폴더를 생성하는 경우	"vSphere Assignment""Assign or unassign vSphere Tag on Object" Resource"Assign virtual machine to resource pool": VApp.Import "Virtual machine""Change Configuration","Add existing disk""Add existing disk""Add new disk""Add new disk" "Virtual"" 머신 "시스템 변경", "가상 머신"" "가상 머신""시스템 변경" "가상 머신""변경 구성", "가상 머신", 변경, CPU 수 변경" "가상 머신", 가상 디스크 변경", 가상 디스크 변경" "가상머신". 머신 변경 "구성 변경" "가상 머신""마이크 구성 변경" "가상 머신""마이크 구성 변경" "구성 변경", "가상 머신", 변경" 디스크 삭

역할의 vSphere 개체	필요한 경우	vCenter GUI에서 필요한 권한
		제" 가상 머신 .권한 부 정".Rename "가상 머신 변경" VIX API로가상 머신 변 경"Gregradevirtual machine"Guestoperating system management by VIX API machine". interaction" "Virtual machine". Interconnection"Power on" "Virtual machine". interactionion.Reset "Virtual machine"."Edit Inventory"."Create new" "Virtual machine"."Create existing" "Virtual machine"Edit Inventory""Create from existing" "Virtual machine"Edit Inventory""Create from existing" virtual machine".Edit Inventory""Create from existing" virtual machine".Edit Inventory""Create from existing""Virtual machine" machine".Provisioning."Clone 가상 머신" "가상 머 신".Provisioning."Deploy 템플릿" "가상 머신".Provisioning" 폴더 만들기" 폴더 생성"폴더 삭제"

또한 사용자에게 일부 **ReadOnly** 권한이 필요하며 일부 역할에는 하위 오브젝트에 권한을 부여하는 사용 권한이 필요합니다. 이러한 설정은 클러스터를 기존 폴더에 설치할지 여부에 따라 달라집니다.

예 17.18. 필수 권한 및 권한 부여 설정

vSphere 오브젝트	폴더 유형	하위 항목으로 권한 부여	권한 필요
vSphere vCenter	Always	False	나열된 필수 권한
vSphere vCenter Datacenter	기존 폴더	False	**ReadOnly** 권한
	설치 프로그램은 폴더를 생성	True	나열된 필수 권한
vSphere vCenter Cluster	Always	True	나열된 필수 권한
vSphere vCenter Datastore	Always	False	나열된 필수 권한
vSphere Switch	Always	False	**ReadOnly** 권한
vSphere Port Group	Always	False	나열된 필수 권한
vSphere vCenter Virtual Machine Folder	기존 폴더	True	나열된 필수 권한

필요한 권한만으로 계정을 생성하는 방법에 대한 자세한 내용은 **vSphere** 문서에서 **vSphere 권한 및 사용자 관리 작업**을 참조하십시오.

vMotion으로 OpenShift Container Platform 사용

vSphere 환경에서 **vMotion**을 사용하려는 경우 **OpenShift Container Platform** 클러스터를 설치하기 전에 다음을 고려하십시오.

- **OpenShift Container Platform**은 일반적으로 **compute-only vMotion** 을 지원합니다. 스토리지 **vMotion**을 사용하면 문제가 발생할 수 있으며 지원되지 않습니다.

  컴퓨팅 및 컨트롤 플레인 노드의 가동 시간을 보장하기 위해 **vMotion**에 대한 **VMware** 모범 사례를 따르는 것이 좋습니다. 또한 **VMware** 유사성 방지 규칙을 사용하여 유지 관리 또는 하드웨어 문제 중에 **OpenShift Container Platform**의 가용성을 개선하는 것이 좋습니다.

  vMotion 및 유사성 방지 규칙에 대한 자세한 내용은 **vMotion 네트워킹 요구 사항** 및 **VM 유사성 방지 규칙에** 대한 **VMware vSphere** 설명서를 참조하십시오.

- **Pod**에서 **vSphere** 볼륨을 사용하는 경우 데이터 저장소를 통해 **VM**을 마이그레이션하면 수동으로 또는 스토리지 **vMotion**을 통해 **VM**을 마이그레이션하면 **OpenShift Container Platform**

PV(영구 볼륨) 오브젝트 내에서 잘못된 참조가 표시됩니다. 이러한 참조는 영향을 받는 포드가 시작되지 않게 하고 데이터 손실을 초래할 수 있습니다.

- 마찬가지로 OpenShift Container Platform은 데이터 저장소에서 VMDK의 선택적 마이그레이션을 지원하지 않으며 VM 프로비저닝 또는 PV의 동적 또는 정적 프로비저닝을 위해 데이터 저장소 클러스터의 일부 또는 PV의 동적 또는 정적 프로비저닝을 위해 데이터 저장소 클러스터의 일부인 데이터 저장소를 사용하는 경우도 있습니다.

클러스터 리소스

설치 관리자 프로비저닝 인프라를 사용하는 OpenShift Container Platform 클러스터를 배포할 때는 설치 프로그램이 vCenter 인스턴스에서 여러 리소스를 생성할 수 있어야 합니다.

표준 OpenShift Container Platform을 설치하면 다음 vCenter 리소스가 생성됩니다.

- 폴더 한 개
- 태그 카테고리 한 개
- 태그 한 개
- 가상 머신:
  - 템플릿 한 개
  - 임시 부트스트랩 노드 한 개
  - 컨트롤 플레인 노드 세 개
  - 컴퓨팅 시스템 세 개

이러한 리소스는 856GB의 스토리지를 사용하지만 부트스트랩 노드는 클러스터 설치 프로세스 중에 제거됩니다. 표준 클러스터를 사용하려면 최소 800GB의 스토리지가 필요합니다.

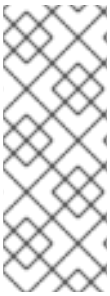
컴퓨팅 시스템을 더 많이 배포할수록 **OpenShift Container Platform** 클러스터는 더 많은 스토리지를 사용합니다.

클러스터 제한

사용 가능한 리소스는 클러스터마다 다릅니다. **vCenter** 내에서 가능한 클러스터 수는 주로 사용 가능한 스토리지 공간과 필요한 리소스 수에 대한 제한으로 제한됩니다. 클러스터가 생성하는 **vCenter** 리소스와 **IP** 주소 및 네트워크와 같이 클러스터를 배포하는 데 필요한 리소스에 대한 제한 사항을 모두 고려해야 합니다.

네트워킹 요구사항

네트워크에 **DHCP**를 사용하고 **DHCP** 서버가 클러스터 시스템에 영구 **IP** 주소를 제공하도록 구성되어 있는지 확인합니다. **DHCP** 서버를 사용하도록 기본 게이트웨이를 구성해야 합니다. 모든 노드는 동일한 **VLAN**에 있어야 합니다. 두 번째 **VLAN**을 **Day 2** 작업으로 사용하여 클러스터를 확장할 수 없습니다. 네트워크가 제한된 환경에서 **VM**은 노드, **PVC**(영구 볼륨 클레임) 및 기타 리소스를 프로비저닝하고 관리할 수 있도록 **vCenter**에 액세스할 수 있어야 합니다. 또한 **OpenShift Container Platform** 클러스터를 설치하기 전에 다음 네트워킹 리소스를 생성해야 합니다.



참고

클러스터의 각 **OpenShift Container Platform** 노드는 **DHCP**를 통해 검색할 수 있는 **NTP(Network Time Protocol)** 서버에 액세스할 수 있어야 합니다. **NTP** 서버 없이도 설치할 수 있습니다. 그러나 비동기 서버 클록으로 인해 **NTP** 서버가 차단하는 오류가 발생합니다.

필요한 IP 주소

설치 관리자 프로비저닝 **vSphere** 설치에는 다음 두 개의 고정 **IP** 주소가 필요합니다.

- **API** 주소는 클러스터 **API**에 액세스하는 데 사용됩니다.
- **Ingress** 주소는 클러스터 인그레스 트래픽에 사용됩니다.

OpenShift Container Platform 클러스터를 설치할 때 설치 프로그램에 이 **IP** 주소를 제공해야 합니다.

DNS 레코드

OpenShift Container Platform 클러스터를 호스팅하는 **vCenter** 인스턴스에 적절한 **DNS** 서버에서 고정 **IP** 주소 두 개의 **DNS** 레코드를 생성해야 합니다. 각 레코드에서 **<cluster_name>**은 클러스터 이름

이고 <base_domain>은 클러스터를 설치할 때 지정하는 클러스터 기본 도메인입니다. 전체 DNS 레코드는 <component>.<cluster_name>.<base_domain> 형식입니다.

표 17.68. 필수 DNS 레코드

구성 요소	레코드	설명
API VIP	api.<cluster_name>.<base_domain>.	이 DNS A/AAAA 또는 CNAME 레코드는 컨트롤 플레인 시스템의 로드 밸런서를 가리켜야 합니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.
Ingress VIP	*.apps.<cluster_name>.<base_domain>.	기본적으로 작업자 노드인 인그레스 라우터 Pod를 실행하는 시스템을 대상으로 하는 로드 밸런서를 가리키는 와일드카드 DNS A/AAAA 또는 CNAME 레코드입니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.

17.7.7. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

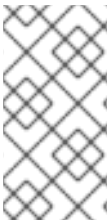
OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 core 사용자의 ~/.ssh/authorized_keys 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 core로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. ./openshift-install gather 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.



참고

AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

절차

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: ~/.ssh/id_ed25519)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 '~/.ssh 디렉터리에 있는지 확인하십시오.

참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 ed25519 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 rsa 또는 ecdsa 알고리즘을 사용하는 키를 생성합니다.

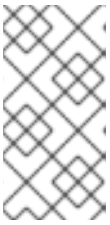
2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 ~/.ssh/id_ed25519.pub 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 ./openshift-install gather 명령을 사용하려는 경우 필요합니다.



참고

일부 배포에서는 ~/.ssh/id_rsa 및 ~/.ssh/id_dsa와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. ssh-agent 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. ssh-agent에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1

SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: ~/.ssh/id_ed25519).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- **OpenShift Container Platform**을 설치할 때 **SSH** 공개 키를 설치 프로그램에 지정합니다.

17.7.8. 시스템 신뢰에 vCenter 루트 CA 인증서 추가

설치 프로그램이 **vCenter API**에 액세스해야 하므로 **OpenShift Container Platform** 클러스터를 설치하기 전에 **vCenter**의 신뢰할 수 있는 루트 **CA** 인증서를 시스템 신뢰에 추가해야 합니다.

절차

1. **vCenter** 홈 페이지에서 **vCenter**의 루트 **CA** 인증서를 다운로드합니다. **vSphere Web Services SDK** 섹션에서 **Download trusted root CA certificates**를 클릭합니다. **<vCenter>/certs/download.zip** 파일이 다운로드됩니다.

2. **vCenter** 루트 **CA** 인증서가 포함된 압축 파일의 압축을 풉니다. 압축 파일의 내용은 다음 파일 구조와 유사합니다.

```
certs
├── lin
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
├── mac
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
└── win
    ├── 108f4d17.0.crt
    ├── 108f4d17.r1.crl
    ├── 7e757f6a.0.crt
    ├── 8e4f8471.0.crt
    └── 8e4f8471.r0.crl

3 directories, 15 files
```

3. 운영 체제 파일을 시스템 신뢰에 추가합니다. 예를 들어 **Fedora** 운영 체제에서는 다음 명령을 실행합니다.

```
# cp certs/lin/* /etc/pki/ca-trust/source/anchors
```

4. 시스템 신뢰를 업데이트합니다. 예를 들어 Fedora 운영 체제에서는 다음 명령을 실행합니다.

```
# update-ca-trust extract
```

17.7.9. 제한된 네트워크 설치를 위한 RHCOS 이미지 생성

Red Hat Enterprise Linux CoreOS(RHCOS) 이미지를 다운로드하여 제한된 네트워크 VMware vSphere 환경에 OpenShift Container Platform을 설치하십시오.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램을 가져옵니다. 제한된 네트워크 설치의 경우 프로그램은 미러 레지스트리 호스트에 있습니다.

절차

1. Red Hat Customer Portal의 제품 다운로드 페이지에 로그인합니다.

2. Version 에서 RHEL 8용 최신 OpenShift Container Platform 4.9 릴리스를 선택합니다.

중요

RHCOS 이미지는 OpenShift Container Platform 릴리스에 따라 변경되지 않을 수 있습니다. 설치하는 OpenShift Container Platform 버전과 같거나 그 이하의 버전 중 가장 최신 버전의 이미지를 다운로드해야 합니다. 사용 가능한 경우 OpenShift Container Platform 버전과 일치하는 이미지 버전을 사용합니다.

3. Red Hat Enterprise Linux CoreOS (RHCOS) - vSphere 이미지를 다운로드합니다.

4. 다운로드한 이미지를 bastion 서버에서 액세스할 수 있는 위치에 업로드합니다.

이제 이미지를 제한된 설치에 사용할 수 있습니다. OpenShift Container Platform 배포에 사용할 이미지 이름 또는 위치를 기록해 둡니다.

17.7.10. 설치 구성 파일 만들기

VMware vSphere에 설치하는 OpenShift Container Platform 클러스터를 사용자 지정할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다. 제한된 네트워크 설치의 경우, 해당 파일은 미러 호스트에 있습니다.
- 미러 레지스트리 작성 중에 생성된 imageContentSources 값이 있어야 합니다.
- 미러 레지스트리에 대한 인증서의 내용을 가져옵니다.
- RHCOS (Red Hat Enterprise Linux CoreOS) 이미지를 검색하여 액세스 가능한 위치에 업로드합니다.
- 서브스크립션 수준에서 서비스 주체 권한을 획득합니다.

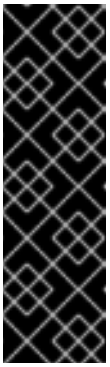
절차

1. install-config.yaml 파일을 생성합니다.

   a. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

   1 <installation_directory>는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.



중요

비어 있는 디렉터리를 지정합니다. 부트스트랩 **X.509** 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 **OpenShift Container Platform** 버전에서 설치 파일을 복사할 때는 주의하십시오.

b. 화면에 나타나는 지시에 따라 클라우드에 대한 구성 세부 사항을 입력합니다.

i. 선택사항: 클러스터 시스템에 액세스하는 데 사용할 **SSH** 키를 선택합니다.

참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 **OpenShift Container Platform** 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 **SSH** 키를 지정합니다.

ii. 대상 플랫폼으로 **vsphere**를 선택합니다.

iii. **vCenter** 인스턴스의 이름을 지정합니다.

iv. 클러스터를 생성하는 데 필요한 권한이 있는 **vCenter** 계정의 사용자 이름과 암호를 지정합니다.

설치 프로그램이 **vCenter** 인스턴스에 연결됩니다.

v. 연결할 **vCenter** 인스턴스에서 데이터 센터를 선택합니다.

vi. 사용할 기본 **vCenter** 데이터 저장소를 선택합니다.

vii. **OpenShift Container Platform** 클러스터를 설치할 **vCenter** 클러스터를 선택합니다. 설치 프로그램은 **vSphere** 클러스터의 루트 리소스 풀을 기본 리소스 풀로 사용합

니다.

viii. vCenter 인스턴스에서 구성한 가상 IP 주소 및 DNS 레코드가 포함된 네트워크를 선택합니다.

ix. 컨트롤 플레인 API 액세스를 위해 구성한 가상 IP 주소를 입력합니다.

x. 클러스터 인그레스용으로 구성한 가상 IP 주소를 입력합니다.

xi. 기본 도메인을 입력합니다. 이 기본 도메인은 구성한 DNS 레코드에서 사용한 것과 동일해야 합니다.

xii. 클러스터를 설명할 수 있는 이름을 입력합니다. 클러스터 이름은 구성한 DNS 레코드에서 사용한 것과 동일해야 합니다.

xiii. Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 붙여넣습니다.

2. install-config.yaml 파일에서 platform.vsphere.clusterOSImage 값을 이미지 위치 또는 이름으로 설정합니다. 예를 들면 다음과 같습니다.

```
platform:
  vsphere:
    clusterOSImage: http://mirror.example.com/images/rhcos-43.81.201912131630.0-
vmware.x86_64.ova?
sha256=ffebbd68e8a1f2a245ca19522c16c86f67f9ac8e4e0c1f0a812b068b16f7265d
```

3. 제한된 네트워크에서의 설치에 필요한 추가 정보를 제공하려면 install-config.yaml 파일을 편집합니다.

a. 레지스트리의 인증 정보를 포함하도록 pullSecret 값을 업데이트합니다.

```
pullSecret: '{"auths":{"<mirror_host_name>:5000": {"auth": "
<credentials>","email": "you@example.com"}}}'
```

<mirror_host_name>의 경우 미러 레지스트리의 인증서에 지정한 레지스트리 도메인 이름을 지정하고 <credentials>의 경우 미러 레지스트리에 base64로 인코딩된 사용자 이름

및 암호를 지정합니다.

b. additionalTrustBundle 매개변수와 값을 추가합니다.

```
additionalTrustBundle: |
  -----BEGIN CERTIFICATE-----

ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
```

값은 미러 레지스트리에 사용한 인증서 파일의 내용이어야 하며, 신뢰할 수 있는 기존 인증 기관 또는 미러 레지스트리에 대해 생성한 자체 서명 인증서일 수 있습니다.

c. 다음과 같은 이미지 콘텐츠 리소스를 추가합니다.

```
imageContentSources:
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: registry.redhat.io/ocp/release
```

이러한 값을 완료하려면 미러 레지스트리 생성 중에 기록한 **imageContentSources**를 사용하십시오.

4. 필요한 **install-config.yaml** 파일을 수정합니다. 사용 가능한 매개변수에 대한 자세한 정보는 **Installation configuration parameters** 섹션에서 확인할 수 있습니다.

5. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.

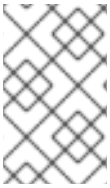
중요

install-config.yaml 파일은 설치 과정에서 사용됩니다. 이 파일을 재사용하려면 지금 백업해야 합니다.

17.7.10.1. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 매개변수 값을 제공하여 클러스터를 호스

팅할 클라우드 플랫폼에서 사용자 계정을 설명하고 선택사항으로 클러스터의 플랫폼을 사용자 지정합니다. install-config.yaml 설치 구성 파일을 생성할 때 명령줄을 통해 필요한 매개변수 값을 제공합니다. 클러스터를 사용자 지정하면 install-config.yaml 파일을 수정하여 플랫폼에 대한 세부 정보를 제공할 수 있습니다.



참고

설치한 후에는 install-config.yaml 파일에서 이러한 매개변수를 수정할 수 없습니다.

중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

17.7.10.1.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 17.69. 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체

매개변수	설명	값
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	**dev** 와 같은 소문자 및 하이픈(-)의 문자열입니다.
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

17.7.10.1.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 17.70. 네트워크 매개변수

매개변수	설명	값

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체  **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.

networking.clusterNetwork	Pod의 IP 주소 블록입니다.
	기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다.
	여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.
		개체의 배열입니다. 예를 들면 다음과 같습니다.
		`networking:`
		`  clusterNetwork:`
		`  - cidr: 10.128.0.0/14`
		`  hostPrefix: 23`

networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된 **cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.

매개변수	설명	값
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16** **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

17.7.10.1.3. 선택적 구성 매개변수

선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

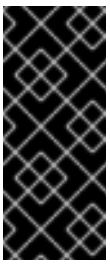
표 17.71. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.

매개변수	설명	값
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은**3**입니다.
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열

매개변수	설명	값
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 기본값은 **External**입니다. 이 필드를 **Internal** 로 설정하는 것은 클라우드 이외의 플랫폼에서는 지원되지 않습니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

17.7.10.1.4. 추가 VMware vSphere 구성 매개변수

추가 VMware vSphere 구성 매개변수는 다음 표에 설명되어 있습니다.

표 17.72. 추가 VMware vSphere 클러스터 매개변수

매개변수	설명	값
platform.vsphere.vCenter	vCenter 서버의 정규화된 호스트 이름 또는 IP 주소입니다.	문자열
platform.vsphere.username	vCenter 인스턴스에 연결하는 데 사용할 사용자 이름입니다. 이 사용자에게는 최소한 vSphere에서 정적 또는 동적 영구 볼륨 프로비저닝 에 필요한 역할과 권한이 있어야 합니다.	문자열
platform.vsphere.password	vCenter 사용자 이름의 암호입니다.	문자열
platform.vsphere.datacenter	vCenter 인스턴스에서 사용할 데이터 센터의 이름입니다.	문자열
platform.vsphere.defaultDatastore	프로비저닝 볼륨에 사용할 기본 데이터 저장소의 이름입니다.	문자열
platform.vsphere.folder	*선택사항*입니다. 설치 프로그램이 가상 머신을 생성하는 기존 폴더의 절대 경로입니다. 이 값을 제공하지 않으면 설치 프로그램이 데이터 센터 가상 머신 폴더에 인프라 ID로 이름이 지정된 폴더를 생성합니다.	문자열(예: **/<datacenter_name>/vm/<folder_name>/<subfolder_name>**).

매개변수	설명	값
platform.vsphere.network	vCenter 인스턴스에서 구성한 가상 IP 주소 및 DNS 레코드가 포함된 네트워크입니다.	문자열
platform.vsphere.cluster	OpenShift Container Platform 클러스터를 설치할 vCenter 클러스터입니다.	문자열
platform.vsphere.apiVIP	컨트롤 플레인 API 액세스를 위해 구성한 가상 IP(VIP) 주소입니다.	IP 주소 (예: **128.0.0.1**)
platform.vsphere.ingressVIP	클러스터 인그레스용으로 구성한 가상 IP(VIP) 주소입니다.	IP 주소 (예: **128.0.0.1**)

17.7.10.1.5. 선택적 VMware vSphere 시스템 풀 구성 매개변수

다음 표에는 선택적 VMware vSphere 시스템 풀 구성 매개 변수가 설명되어 있습니다.

표 17.73. 선택적 VMware vSphere 시스템 풀 매개변수

매개변수	설명	값
platform.vsphere.clusterOSImage	설치 프로그램이 RHCOS 이미지를 다운로드하는 위치입니다. 네트워크가 제한된 환경에서 설치를 수행하려면이 매개변수를 설정해야 합니다.	HTTP 또는 HTTPS URL (선택 옵션으로 SHA-256 형식의 체크섬을 사용) 예: **https://mirror.openshift.com/images/rhcos-<version>-vmware.<architecture>.ova**.
platform.vsphere.osDisk.diskSizeGB	디스크 크기(GB)입니다.	정수
platform.vsphere.cpus	가상 머신을 할당할 총 가상 프로세서 코어 수입니다. **platform.vsphere.cpus** 의 값은 **platform.vsphere.coresPerSocket** 값의 여러 개여야 합니다.	정수
platform.vsphere.coresPerSocket	가상 머신의 소켓당 코어 수입니다. 가상 머신의 가상 소켓 수는 **platform.vsphere.cpus/platform.vsphere.coresPerSocket** 입니다. 컨트롤 플레인 노드 및 작업자 노드의 기본값은 각각 **4** 및 **2** 입니다.	정수
platform.vsphere.memoryMB	가상 머신의 메모리 크기(MB)입니다.	정수

17.7.10.2. 설치 관리자 프로비저닝 VMware vSphere 클러스터의 샘플 install-config.yaml 파일

install-config.yaml 파일을 사용자 지정하여 OpenShift Container Platform 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.

```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 3
  platform:
    vsphere: (4)
      cpus: 2
      coresPerSocket: 2
      memoryMB: 8192
      osDisk:
        diskSizeGB: 120
controlPlane: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3
  platform:
    vsphere: (7)
      cpus: 4
      coresPerSocket: 2
      memoryMB: 16384
      osDisk:
        diskSizeGB: 120
metadata:
  name: cluster (8)
platform:
  vsphere:
    vcenter: your.vcenter.server
    username: username
    password: password
    datacenter: datacenter
    defaultDatastore: datastore
    folder: folder
    network: VM_Network
    cluster: vsphere_cluster_name (9)
    apiVIP: api_vip
    ingressVIP: ingress_vip
    clusterOSImage: http://mirror.example.com/images/rhcos-48.83.202103221318-0-
vmware.x86_64.ova (10)
fips: false
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email":
"you@example.com"}}}' (11)
sshKey: 'ssh-ed25519 AAAA...'
additionalTrustBundle: | (12)
  -----BEGIN CERTIFICATE-----
  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
```

```
  -----END CERTIFICATE-----
imageContentSources: 13
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

1

클러스터의 기본 도메인입니다. 모든 **DNS** 레코드는 이 기본 도메인의 하위 도메인이어야 하며 클러스터 이름을 포함해야 합니다.

2 5

controlPlane 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

3 6

동시 멀티스레딩 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. 일부 클러스터 시스템에서 동시 멀티스레딩을 비활성화할 경우에는 해당 멀티스레딩을 모든 클러스터 시스템에서 비활성화해야 합니다.

중요

동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다. 동시 멀티스레딩을 비활성화하는 경우 시스템은 **8**개 이상의 **CPU**와 **32GB**의 **RAM**을 사용해야 합니다.

4 7

선택사항: 컴퓨팅 및 컨트롤 플레인 시스템의 시스템 풀 매개변수에 대한 추가 구성을 제공하십시오.

8

DNS 레코드에 지정한 클러스터 이름입니다.

9

OpenShift Container Platform 클러스터를 설치할 **vSphere** 클러스터입니다. 설치 프로그램은 **vSphere** 클러스터의 루트 리소스 풀을 기본 리소스 풀로 사용합니다.

10

bastion 서버에서 액세스할 수 있는 RHCOS(Red Hat Enterprise Linux CoreOS) 이미지의 위치입니다.

11

<local_registry>는 미러 레지스트리가 해당 내용을 제공하는 데 사용하는 레지스트리 도메인 이름과 포트(선택사항)를 지정합니다. 예: registry.example.com 또는 registry.example.com:5000. <credentials>는 미러 레지스트리의 base64 인코딩 사용자 이름과 암호를 지정합니다.

12

미러 레지스트리에 사용한 인증서 파일의 내용을 제공하십시오.

13

명령 출력에서 imageContentSources 섹션을 제공하여 리포지토리를 미러링하십시오.

17.7.10.3. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 HTTP 또는 HTTPS 프록시를 사용할 수 있습니다. install-config.yaml 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 OpenShift Container Platform 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 install-config.yaml 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 API에 대한 호출을 포함하여 모든 클러스터 발신(Egress) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 Proxy 오브젝트의 spec.noProxy 필드에 사이트를 추가했습니다.

참고

Proxy 오브젝트의 status.noProxy 필드는 설치 구성에 있는 networking.machineNetwork[].cidr, networking.clusterNetwork[].cidr, networking.serviceNetwork[] 필드의 값으로 채워집니다.

Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 Red Hat OpenStack Platform (RHOSP)에 설치하는 경우 Proxy 오브젝트 status.noProxy 필드도 인스턴스 메타데이터 끝점(169.254.169.254)로 채워집니다.

절차

1. install-config.yaml 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1 클러스터 외부에서 HTTP 연결을 구축하는 데 사용할 프록시 URL입니다. URL 스키마는 http여야 합니다.

2 클러스터 외부에서 HTTPS 연결을 구축하는 데 사용할 프록시 URL입니다.

3 대상 도메인 이름, IP 주소 또는 프록시에서 제외할 기타 네트워크 CIDR로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 .을 입력합니다. 예를 들어, .y.com은 x.y.com과 일치하지만 y.com은 일치하지 않습니다. *를 사용하여 모든 대상에 대해 프록시를 바이패스합니다. vCenter의 IP 주소와 해당 시스템에 사용하는 IP 범위를 포함해야 합니다.

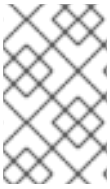
4

참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 **OpenShift Container Platform**을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 **nil**이 됩니다.

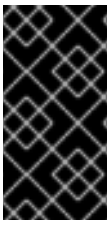


참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

17.7.11. 클러스터 배포

호환되는 클라우드 플랫폼에 **OpenShift Container Platform**을 설치할 수 있습니다.



중요

최초 설치 과정에서 설치 프로그램의 **create cluster** 명령을 한 번만 실행할 수 있습니다.

사전 요구 사항

- **OpenShift Container Platform** 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

절차

1. 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터 배포를 초기화합니다.

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
  --log-level=info 2
```

1

<installation_directory> 값으로 사용자 지정한 **./install-config.yaml** 파일의 위치를 지정합니다.

2

다른 설치 세부 사항을 보려면 **info** 대신 **warn, debug** 또는 **error**를 지정합니다.

클러스터 배포가 완료되면 웹 콘솔로 연결되는 링크와 **kubeadmin** 사용자의 인증 정보가 포함된 클러스터 액세스 지침이 사용자 터미널에 표시됩니다.

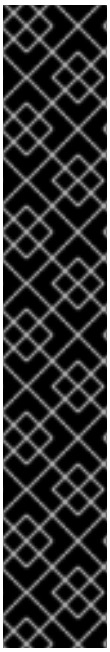
```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-Wt5AL"
INFO Time elapsed: 36m22s
```

+

참고

설치에 성공하면 클러스터 액세스 및 인증 정보도 **<installation_directory>/.openshift_install.log**로 출력됩니다.

+



중요

- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 node-bootstrapper 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.

- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

+

중요

설치 프로그램에서 생성되는 파일이나 설치 프로그램을 삭제해서는 안 됩니다. 클러스터를 삭제하려면 두 가지가 모두 필요합니다.

17.7.12. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift CLI(oc)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 oc를 설치할 수 있습니다.

중요

이전 버전의 oc를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 oc를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 Linux에서 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

절차

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.

2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.

3. **OpenShift v4.9 Linux Client** 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.

4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. **oc** 바이너리를 **PATH**에 있는 디렉터리에 배치합니다.

   PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 **OpenSfhit CLI** 설치

다음 절차에 따라 **Windows**에 **OpenShift CLI(oc)** 바이너리를 설치할 수 있습니다.

절차

1. **Red Hat** 고객 포털에서 **OpenShift Container Platform** 다운로드 페이지로 이동합니다.

2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.

3. **OpenShift v4.9 Windows Client** 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.

4. **ZIP** 프로그램으로 아카이브의 압축을 풉니다.

5. oc 바이너리를 PATH에 있는 디렉터리로 이동합니다.

   PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치

다음 절차에 따라 macOS에서 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

절차

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.
2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. OpenShift v4.9 MacOSX Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 해제하고 압축을 풉니다.
5. oc 바이너리 PATH의 디렉터리로 이동합니다.

   PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
$ oc <command>
```

17.7.13. CLI를 사용하여 클러스터에 로그인

클러스터 kubeconfig 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. kubeconfig 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- oc CLI를 설치했습니다.

절차

1. kubeadmin 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 oc 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

17.7.14. 기본 OperatorHub 소스 비활성화

Red Hat 및 커뮤니티 프로젝트에서 제공하는 콘텐츠를 소싱하는 Operator 카탈로그는 OpenShift

Container Platform을 설치하는 동안 기본적으로 OperatorHub용으로 구성됩니다. 제한된 네트워크 환경에서는 클러스터 관리자로서 기본 카탈로그를 비활성화해야 합니다.

절차

- OperatorHub 오브젝트에 disableAllDefaultSources: true를 추가하여 기본 카탈로그의 소스를 비활성화합니다.

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

작은 정보

또는 웹 콘솔을 사용하여 카탈로그 소스를 관리할 수 있습니다. 관리 → 클러스터 설정 → 구성 → OperatorHub 페이지에서 개별 소스를 생성, 삭제, 비활성화 및 활성화할 수 있는 소스 탭을 클릭합니다.

17.7.15. 레지스트리 스토리지 생성

클러스터를 설치한 후 Registry Operator를 위한 스토리지를 생성해야 합니다.

17.7.15.1. 설치 중 제거된 이미지 레지스트리

공유 가능한 개체 스토리지를 제공하지 않는 플랫폼에서 OpenShift Image Registry Operator는 자체적으로 Removed로 부트스트랩합니다. 이를 통해 openshift-installer가 이러한 플랫폼 유형에서 설치를 완료할 수 있습니다.

설치 후 managementState를 Removed에서 Managed로 전환하도록 Image Registry Operator 구성을 편집해야 합니다.

참고

Prometheus 콘솔은 ImageRegistryRemoved 경고를 제공합니다. 다음은 예시 경고입니다.

"이미지 레지스트리가 제거되었습니다. ImageStreamTags를 참조하는 ImageStreamTags, BuildConfigs 및 DeploymentConfigs가 예상대로 작동하지 않을 수 있습니다. 스토리지를 구성하고 configs.imageregistry.operator.openshift.io를 편집하여 구성을 Managed 상태로 업데이트하십시오."

17.7.15.2. 이미지 레지스트리 스토리지 구성

기본 스토리지를 제공하지 않는 플랫폼에서는 처음에 **Image Registry Operator**를 사용할 수 없습니다. 설치한 후에 스토리지를 사용하도록 레지스트리를 구성하여 **Registry Operator**를 사용 가능하도록 만들어야 합니다.

프로덕션 클러스터에 필요한 영구 볼륨을 구성하는 과정의 지침이 표시됩니다. 해당하는 경우, 프로덕션 환경 외 클러스터에서만 사용할 수 있는 저장 위치로서 빈 디렉터리를 구성하는 과정의 지침이 표시됩니다.

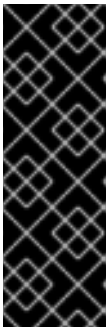
업그레이드 중에 **Recreate** 롤아웃 전략을 사용하여 이미지 레지스트리의 블록 스토리지 유형 사용을 허용하기 위한 추가 지침이 제공됩니다.

17.7.15.2.1. VMware vSphere용 레지스트리 스토리지 구성

클러스터 관리자는 설치한 후 스토리지를 사용하도록 레지스트리를 구성해야 합니다.

사전 요구 사항

- 클러스터 관리자 권한이 있어야 합니다.
- **VMware vSphere**에 클러스터가 있어야 합니다.
- **Red Hat OpenShift Container Storage**와 같이 클러스터용 영구 스토리지 프로비저닝.



중요

OpenShift Container Platform은 복제본이 하나만 있는 경우 이미지 레지스트리 스토리지에 대한 **ReadWriteOnce** 액세스를 지원합니다. **ReadWriteOnce** 액세스에서는 레지스트리가 **Recreate** 롤아웃 전략을 사용해야 합니다. 두 개 이상의 복제본으로 고 가용성을 지원하는 이미지 레지스트리를 배포하려면 **ReadWriteMany** 액세스가 필요합니다.

- **"100Gi"** 용량이 필요합니다.

중요

테스트 결과, **RHEL**의 **NFS** 서버를 핵심 서비스용 스토리지 백엔드로 사용하는 데 문제가 있는 것으로 나타납니다. 여기에는 **OpenShift Container Registry and Quay**, 스토리지 모니터링을 위한 **Prometheus**, 로깅 스토리지를 위한 **Elasticsearch**가 포함됩니다. 따라서 **RHEL NFS**를 사용하여 핵심 서비스에서 사용하는 **PV**를 백업하는 것은 권장되지 않습니다.

마켓플레이스의 다른 **NFS** 구현에는 이러한 문제가 나타나지 않을 수 있습니다. 이러한 **OpenShift Container Platform** 핵심 구성 요소에 대해 완료된 테스트에 대한 자세한 내용은 개별 **NFS** 구현 공급업체에 문의하십시오.

절차

1. 스토리지를 사용하도록 레지스트리를 구성하기 위해 **configs.imageregistry/cluster** 리소스에서 **spec.storage.pvc**를 변경합니다.

참고

공유 스토리지를 사용할 때 보안 설정을 확인하여 외부에서의 액세스를 방지합니다.

2. 레지스트리 **pod**가 없는지 확인합니다.

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

출력 예

```
No resourses found in openshift-image-registry namespace
```

참고

출력에 레지스트리 **Pod**가 있는 경우 이 절차를 계속할 필요가 없습니다.

3. 레지스트리 구성을 확인합니다.

```
$ oc edit configs.imageregistry.operator.openshift.io
```

출력 예

```
storage:
  pvc:
    claim: 1
```

1 image-registry-storage 영구 볼륨 클레임 (PVC)을 자동으로 생성할 수 있도록 claim 필드를 비워 둡니다. PVC는 기본 스토리지 클래스를 기반으로 생성됩니다. 그러나 기본 스토리지 클래스에서 RADOS 블록 장치(RBD)와 같은 ReadWriteOnce(ReadWriteOnce) 볼륨을 제공할 수 있으므로 복제본이 두 개 이상될 때 문제가 발생할 수 있습니다.

4. clusteroperator 상태를 확인합니다.

```
$ oc get clusteroperator image-registry
```

출력 예

```
NAME             VERSION                             AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                                 True        False         False     6h50m
```

17.7.16. 스틸 클럭 계정

기본적으로 설치 프로그램은 스틸 클럭 회계 매개 변수 (stealclock.enabled)를 활성화하지 않고 클러스터의 가상 머신을 프로비저닝합니다. 스틸링 클럭 계정을 사용하면 클러스터 문제 해결에 도움이 될 수

있습니다. 클러스터가 배포된 후 vSphere Client를 사용하여 각 가상 머신에서 이 매개변수를 활성화합니다.

자세한 내용은 이 Red Hat 기술 자료 문서 를 참조하십시오.

17.7.17. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

17.7.18. 다음 단계

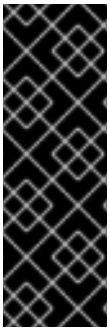
- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.
- 레지스트리를 설정하고 레지스트리 스토리지를 구성합니다.

17.8. 사용자 프로비저닝 인프라가 있는 제한된 네트워크에서 VSPHERE에 클러스터 설치

OpenShift Container Platform 버전 4.9에서는 네트워크가 제한된 환경에서 사용자가 프로비저닝하는 VMware vSphere 인프라에 클러스터를 설치할 수 있습니다.

참고

OpenShift Container Platform은 단일 **VMware vCenter**에만 클러스터 배포를 지원합니다. 여러 **vCenter**에 머신/머신 세트가 있는 클러스터 배포는 지원되지 않습니다.

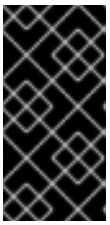


중요

사용자가 프로비저닝한 인프라 설치를 수행하는 단계는 예시용으로만 제공됩니다. 사용자가 제공하는 인프라를 사용하여 클러스터를 설치하려면 **vSphere** 플랫폼 및 **OpenShift Container Platform** 설치 프로세스에 대한 정보가 필요합니다. 사용자 프로비저닝 인프라 설치 지침을 가이드로 사용하십시오. 다른 방법을 통해 필요한 리소스를 자유롭게 생성할 수 있습니다.

17.8.1. 사전 요구 사항

- **OpenShift Container Platform** 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.
- 미러 호스트에 레지스트리를 생성하고 사용 중인 **OpenShift Container Platform** 버전의 **imageContentSources** 데이터를 가져옵니다.

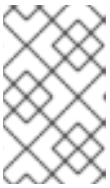
  

  중요

  미러 호스트에 설치 미디어가 있으므로 해당 컴퓨터를 사용하여 모든 설치 단계를 완료하십시오.

- 클러스터용 영구 스토리지를 프로비저닝합니다. 프라이빗 이미지 레지스트리를 배포하려면 스토리지에서 **ReadWriteMany** 액세스 모드를 제공해야 합니다.
- 설치를 완료하려면 **vSphere** 호스트에 **RHCOS(Red Hat Enterprise Linux CoreOS) OVA**를 업로드해야 합니다. 이 프로세스를 완료하는 시스템에는 **vCenter** 및 **ESXi** 호스트의 포트 **443**에 액세스해야 합니다. 포트 **443**에 액세스할 수 있는지 확인했습니다.
- 방화벽을 사용하는 경우 관리자가 포트 **443**에 액세스할 수 있음을 확인합니다. 설치에 성공하려면 포트 **443**에서 컨트롤 플레인 노드가 **vCenter** 및 **ESXi** 호스트에 연결할 수 있어야 합니다.

- 방화벽을 사용하며 **Telemetry** 서비스를 사용할 예정인 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성해야 합니다.



참고

프록시를 구성하는 경우에도 해당 사이트 목록을 검토하십시오.

17.8.2. 네트워크가 제한된 환경에서의 설치 정보

OpenShift Container Platform 4.9에서는 소프트웨어 구성 요소를 받기 위한 인터넷 접속이 필요하지 않은 설치를 수행할 수 있습니다. 제한된 네트워크 설치는 클러스터를 설치하는 클라우드 플랫폼에 따라 설치 관리자 프로비저닝 인프라 또는 사용자 프로비저닝 인프라를 사용하여 완료할 수 있습니다.

클라우드 플랫폼에 제한된 네트워크 설치를 수행하는 방법을 선택해도 클라우드 **API**에 액세스는 가능해야 합니다. **Amazon Web Service**의 **Route 53 DNS** 및 **IAM** 서비스와 같은 일부 클라우드 기능에는 인터넷 액세스가 필요합니다. 사용 중인 네트워크에 따라 베어메탈 하드웨어 또는 **VMware vSphere**에 설치하기 위해 필요한 인터넷 액세스가 줄어들 수 있습니다.

제한된 네트워크 설치를 완료하려면 **OpenShift Container Platform** 레지스트리의 내용을 미러링하고 설치 미디어를 포함할 레지스트리를 생성해야 합니다. 인터넷과 폐쇄 네트워크에 모두 액세스하거나 제한 사항을 따르는 다른 방법을 통해 미러 호스트에 레지스트리를 생성할 수 있습니다.

중요

사용자 프로비저닝 설치의 구성이 복잡하므로 사용자 프로비저닝 인프라를 사용하여 제한된 네트워크 설치를 시도하기 전에 표준 사용자 프로비저닝 인프라 설치를 완료하는 것이 좋습니다. 이 테스트 설치를 완료하면 제한된 네트워크에 설치하는 동안 발생할 수 있는 문제를 보다 쉽게 파악 및 해결할 수 있습니다.

17.8.2.1. 추가 제한

제한된 네트워크의 클러스터에는 다음과 같은 추가 제한이 있습니다.

- **ClusterVersion** 상태에 사용 가능한 업데이트를 검색할 수 없음 오류가 포함되어 있습니다.
- 기본적으로 필요한 이미지 스트림 태그에 액세스할 수 없기 때문에 개발자 카탈로그의 내용을 사용할 수 없습니다.

17.8.3. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하기 위해 필요한 이미지를 받으려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 **Telemetry** 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.

중요

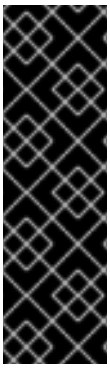
클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

17.8.4. VMware vSphere 인프라 요구사항

사용하는 구성 요소의 요구사항을 충족하는 **VMware vSphere** 버전 **6** 또는 **7** 인스턴스에 **OpenShift Container Platform** 클러스터를 설치해야 합니다.

표 **17.74. vSphere** 가상 환경의 버전 요구 사항

가상 환경 제품	필요한 버전
VM 하드웨어 버전	13 이상
vSphere ESXi hosts	6.5 이상
vCenter 호스트	6.5 이상



중요

이제 VMware vSphere 버전 6.7U2 또는 이전 버전에 클러스터를 설치하는 것이 더 이상 사용되지 않으며 가상 하드웨어 버전 13은 더 이상 사용되지 않습니다. 이러한 버전은 여전히 완전히 지원되지만 향후 OpenShift Container Platform 버전에서는 지원이 제거됩니다. 이제 OpenShift Container Platform의 vSphere 가상 머신의 하드웨어 버전 15가 기본값이 되었습니다. vSphere 노드의 하드웨어 버전을 업데이트하려면 " vSphere에서 실행되는 노드에서 하드웨어 업데이트" 문서를 참조하십시오.

표 17.75. VMware 구성 요소에 지원되는 최소 vSphere 버전

구성 요소	지원되는 최소 버전	설명
하이퍼바이저	HW 버전 13이 포함된 vSphere 6.5 이상	이 버전은 RHCOS(Red Hat Enterprise Linux CoreOS)가 지원하는 최소 버전입니다. Red Hat Enterprise Linux 8 지원 하이퍼바이저 목록을 참조하십시오.
In-tree 드라이버가 있는 스토리지	vSphere 6.5 이상	이 플러그인은 OpenShift Container Platform에 포함된 vSphere용 인트리 스토리지 드라이버를 사용하여 vSphere 스토리지를 생성합니다.
선택사항: 네트워킹(NSX-T)	vSphere 6.5U3 또는 vSphere 6.7U2 이상	OpenShift Container Platform에는 vSphere 6.5U3 또는 vSphere 6.7U2+가 필요합니다. NSX 및 OpenShift Container Platform의 호환성에 대한 자세한 내용은 VMware의 NSX 컨테이너 플러그인 설명서의 릴리스 노트 섹션을 참조하십시오.

vSphere 버전 6.5 인스턴스를 사용하는 경우 OpenShift Container Platform을 설치하기 전에 6.7U3 또는 7.0로 업그레이드하는 것이 좋습니다.

중요

OpenShift Container Platform을 설치하기 전에 ESXi 호스트의 시간이 동기화되었는지 확인해야 합니다. VMware 문서에서 Edit Time Configuration for a Host를 참조하십시오.

추가 리소스

- **vSphere 노드의 하드웨어 버전을 업데이트하려면 vSphere에서 실행 중인 노드에서 하드웨**

어 업데이트를 참조하십시오.

17.8.5. 사용자 프로비저닝 인프라를 포함한 클러스터의 시스템 요구사항

사용자 프로비저닝 인프라가 포함된 클러스터의 경우, 필요한 모든 시스템을 배포해야 합니다.

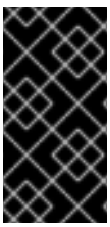
이 섹션에서는 사용자 프로비저닝 인프라에 OpenShift Container Platform을 배포해야 하는 요구 사항에 대해 설명합니다.

17.8.5.1. 클러스터 설치에 필요한 시스템

최소 OpenShift Container Platform 클러스터에 다음과 같은 호스트가 필요합니다.

표 17.76. 최소 필수 호스트

호스트	설명
임시 부트스트랩 시스템 한 개	컨트롤 플레인 시스템 세 개에 OpenShift Container Platform 클러스터를 배포하기 위한 부트스트랩 시스템이 클러스터에 필요합니다. 클러스터를 설치한 후 부트스트랩 시스템을 제거할 수 있습니다.
컨트롤 플레인 시스템 세 개	컨트롤 플레인 시스템은 컨트롤 플레인을 구성하는 Kubernetes 및 OpenShift Container Platform 서비스를 실행합니다.
두 개 이상의 컴퓨팅 시스템(작업자 시스템이라고도 함).	OpenShift Container Platform 사용자가 요청한 워크로드는 컴퓨팅 머신에서 실행됩니다.

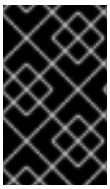


중요

클러스터의 고가용성을 유지하려면 이러한 클러스터 시스템에 대해 별도의 물리적 호스트를 사용하십시오.

부트스트랩, 컨트롤 플레인 시스템은 운영 체제로 RHCOS (Red Hat Enterprise Linux CoreOS)를 사용해야 합니다. 그러나 컴퓨팅 머신은 RHCOS(Red Hat Enterprise Linux CoreOS), RHEL (Red Hat Enterprise Linux) 7.9 또는 RHEL 8.4 중에서 선택할 수 있습니다.

RHCOS는 RHEL 8(Red Hat Enterprise Linux)을 기반으로하며 모든 하드웨어 인증 및 요구사항을 모두 이어받습니다. Red Hat Enterprise Linux 기술 기능 및 제한을 참조하십시오.



중요

모든 가상 머신은 설치 프로그램과 동일한 데이터 저장소 및 폴더에 있어야 합니다.

17.8.5.2. 클러스터 설치를 위한 최소 리소스 요구 사항

각 클러스터 시스템이 다음과 같은 최소 요구사항을 충족해야 합니다.

표 17.77. 최소 리소스 요구사항

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
부트스트랩	RHCOS	4	16GB	100GB	300
컨트롤 플레인	RHCOS	4	16GB	100GB	300
컴퓨팅	RHCOS, RHEL 7.9 또는 RHEL 8.4 [3]	2	8GB	100GB	300

1. SMT(동시 멀티스레딩) 또는 하이퍼 스레딩이 활성화되지 않은 경우 하나의 vCPU는 하나의 물리적 코어와 동일합니다. 활성화하면 다음과 같은 공식을 사용하여 해당 비율을 계산합니다. (코어 당 스레드 수 × 코어 수) × 소켓 수 = vCPU 수

2. OpenShift Container Platform 및 Kubernetes는 디스크 성능에 민감하며 특히 10ms p99 fsync 기간이 필요한 컨트롤 플레인 노드의 etcd에 더 빠른 스토리지가 권장됩니다. 많은 클라우드 플랫폼에서 스토리지 크기와 IOPS를 함께 확장되므로 충분한 성능을 얻으려면 스토리지 볼륨을 과도하게 할당해야 할 수 있습니다.

3. 사용자가 프로비저닝한 모든 설치와 마찬가지로 클러스터에서 RHEL 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. RHEL 7 컴퓨팅 머신 사용은 더 이상 사용되지 않으며 향후 OpenShift Container Platform 4 릴리스에서 제거될 예정입니다.

17.8.5.3. 인증서 서명 요청 관리

사용자가 프로비저닝하는 인프라를 사용하는 경우 자동 시스템 관리 기능으로 인해 클러스터의 액세스가 제한되므로 설치한 후 클러스터 인증서 서명 요청(CSR)을 승인하는 메커니즘을 제공해야 합니다. kube-controller-manager는 kubelet 클라이언트 CSR만 승인합니다. machine-approver는 올바른 시스템에서 발행한 요청인지 확인할 수 없기 때문에 kubelet 자격 증명을 사용하여 요청하는 서비스 인증서의

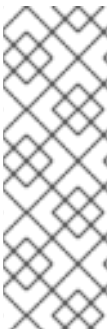
유효성을 보장할 수 없습니다. kubelet 서빙 인증서 요청의 유효성을 확인하고 요청을 승인하는 방법을 결정하여 구현해야 합니다.

17.8.5.4. 사용자 프로비저닝 인프라에 대한 네트워킹 요구사항

모든 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템이 부팅 중에 Ignition 구성 파일을 가져오려면 initramfs에 네트워킹을 구성해야 합니다.

초기 부팅 과정에서 시스템에 필요한 부팅 옵션을 제공하여 DHCP 서버를 통해 설정하거나 정적으로 설정하는 IP 주소 구성이 필요합니다. 네트워크 연결이 설정된 후 시스템은 HTTP 또는 HTTPS 서버에서 Ignition 구성 파일을 다운로드합니다. 그런 다음 Ignition 구성 파일을 사용하여 각 머신의 정확한 상태를 설정합니다. Machine Config Operator는 설치 후 새 인증서 또는 키 적용과 같은 머신에 대한 추가 변경을 완료합니다.

클러스터 시스템의 장기적인 관리를 위해 DHCP 서버를 사용하는 것이 좋습니다. DHCP 서버가 클러스터 시스템에 영구 IP 주소, DNS 서버 정보 및 호스트 이름을 제공하도록 구성되었는지 확인합니다.



참고

사용자 프로비저닝 인프라에 DHCP 서비스를 사용할 수 없는 경우 RHCOS 설치 시 노드에 IP 네트워킹 구성과 DNS 서버의 주소를 대신 제공할 수 있습니다. ISO 이미지에서 설치하는 경우 부팅 인수로 전달할 수 있습니다. 고정 IP 프로비저닝 *및 고급 네트워킹 옵션에 대한 자세한 내용은 RHCOS 설치 및 OpenShift Container Platform 부트스트랩 프로세스 시작* 섹션을 참조하십시오.

Kubernetes API 서버가 클러스터 시스템의 노드 이름을 확인할 수 있어야 합니다. API 서버와 작업자 노드가 서로 다른 영역에 있는 경우, API 서버가 노드 이름을 확인할 수 있도록 기본 DNS 검색 영역을 설정할 수 있습니다. 노드 개체와 모든 DNS 요청에서 항상 정규화된 도메인 이름으로 호스트를 가리키는 것도 지원되는 방법입니다

17.8.5.4.1. DHCP를 통해 클러스터 노드의 호스트 이름 설정

RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 호스트 이름은 NetworkManager를 통해 설정됩니다. 기본적으로 시스템은 DHCP를 통해 호스트 이름을 가져옵니다. DHCP에서 호스트 이름을 제공하지 않거나, 커널 인수를 통해 정적으로 설정하거나 다른 방법을 통해 설정하면 역방향 DNS 조회를 통해 얻을 수 있습니다. 역방향 DNS 조회는 노드에서 네트워크를 초기화한 후 수행되며 확인하는 데 시간이 걸릴 수 있습니다. 다른 시스템 서비스는 이 보다 먼저 시작하여 호스트 이름을 localhost 등으로 감지할 수 있습니다. DHCP를 사용하여 각 클러스터 노드의 호스트 이름을 제공하여 이 문제를 방지할 수 있습니다.

또한 DHCP를 통해 호스트 이름을 설정하면 DNS 분할 수평 구현 환경에서 수동으로 DNS 레코드 이

름 구성 오류를 무시할 수 있습니다.

17.8.5.4.2. 네트워크 연결 요구사항

OpenShift Container Platform 클러스터 구성 요소가 통신할 수 있도록 시스템 간 네트워크 연결을 구성해야 합니다. 각 시스템에서 클러스터에 있는 다른 모든 시스템의 호스트 이름을 확인할 수 있어야 합니다.

이 섹션에서는 필요한 포트에 대해 자세히 설명합니다.

중요

연결된 **OpenShift Container Platform** 환경에서 모든 노드는 플랫폼 컨테이너의 이미지를 가져오고 **Red Hat**에 원격 측정 데이터를 제공하기 위해 인터넷에 액세스할 수 있어야 합니다.

표 **17.78.** 모든 시스템 간 통신에 사용되는 포트

프로토콜	포트	설명
ICMP	해당 없음	네트워크 연결성 테스트
TCP	**1936**	메트릭
	9000-**9999**	**9100**-**9101** 포트의 노드 내보내기 및 **9099** 포트의 Cluster Version Operator를 포함한 호스트 수준 서비스.
	10250-**10259**	Kubernetes에서 예약하는 기본 포트
	10256	openshift-sdn
UDP	**4789**	VXLAN 및 Geneve
	6081	VXLAN 및 Geneve
	9000-**9999**	**9100**-**9101** 포트의 노드 내보내기를 포함한 호스트 수준 서비스.
	500	IPsec IKE 패킷
	4500	IPsec NAT-T 패킷
TCP/UDP	**30000**-**32767**	Kubernetes 노드 포트

프로토콜	포트	설명
ESP	해당 없음	IPsec Encapsulating Security Payload (ESP)

표 17.79. 모든 시스템과 컨트롤 플레인 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**6443**	Kubernetes API

표 17.80. 컨트롤 플레인 머신 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**2379-2380**	etcd 서버 및 피어 포트

이더넷 어댑터 하드웨어 주소 요구사항

클러스터용 VM을 프로비저닝할 때 각 VM에 대해 구성된 이더넷 인터페이스는 VMware OUI(Organizationally Unique Identifier) 할당 범위의 MAC 주소를 사용해야 합니다.

- **00:05:69:00:00:00 ~ 00:05:69:FF:FF:FF**
- **00:0c:29:00:00:00 ~ 00:0c:29:FF:FF:FF**
- **00:1c:14:00:00:00 ~ 00:1c:14:FF:FF:FF**
- **00:50:56:00:00:00 ~ 00:50:56:FF:FF:FF**

VMware OUI 외부의 MAC 주소를 사용하면 클러스터 설치가 실패합니다.

사용자 프로비저닝 인프라에 대한 NTP 구성

OpenShift Container Platform 클러스터는 기본적으로 공용 NTP(Network Time Protocol) 서버를 사용하도록 구성되어 있습니다. 로컬 엔터프라이즈 NTP 서버를 사용하거나 클러스터가 연결이 끊긴 네트워크에 배포되는 경우 특정 시간 서버를 사용하도록 클러스터를 구성할 수 있습니다. 자세한 내용은 *chrony 타임 서비스 설정* 문서를 참조하십시오.

DHCP 서버가 NTP 서버 정보를 제공하는 경우 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템

의 chrony 타임 서비스에서 정보를 읽고 NTP 서버와 클럭을 동기화할 수 있습니다.

추가 리소스

- chrony 타임 서비스 설정

17.8.5.5. 사용자 프로비저닝 DNS 요구사항

OpenShift Container Platform 배포의 경우 다음 구성 요소에 DNS 이름을 확인해야 합니다.

- Kubernetes API
- OpenShift Container Platform 애플리케이션 와일드카드
- 부트스트랩, 컨트롤 플레인 및 컴퓨팅 시스템

Kubernetes API, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템에 대한 역방향 DNS 확인이 필요합니다.

DNS A/AAAA 또는 CNAME 레코드는 이름 확인에 사용되며 PTR 레코드는 역방향 이름 확인에 사용됩니다. RHCOS (Red Hat Enterprise Linux CoreOS)는 DHCP에서 호스트 이름을 제공하지 않는 한 모든 노드의 호스트 이름을 설정할 때 역방향 레코드를 사용하기 때문에 역방향 레코드가 중요합니다. 또한 역방향 레코드는 OpenShift Container Platform이 작동하는 데 필요한 인증서 서명 요청 (CSR)을 생성하는 데 사용됩니다.

참고

DHCP 서버를 사용하여 각 클러스터 노드에 호스트 이름을 제공하는 것이 좋습니다. 자세한 내용은 *사용자 프로비저닝 인프라 섹션에 대한 DHCP 권장 사항* 섹션을 참조하십시오.

사용자가 프로비저닝한 OpenShift Container Platform 클러스터에 대해 다음 DNS 레코드가 필요하며 설치 전에 있어야 합니다. 각 레코드에서 <cluster_name>은 클러스터 이름이고 <base_domain>은 install-config.yaml 파일에서 지정하는 기반 도메인입니다. 전체 DNS 레코드는 <component>.<cluster_name>.<base_domain> 형식입니다.

표 17.81. 필수 DNS 레코드

구성 요소	레코드	설명
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API 로드 밸런서를 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.
	api-int.<cluster_name>.<base_domain>.	내부적으로 API 로드 밸런서를 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다. **중요** API 서버는 Kubernetes에 기록된 호스트 이름으로 작업자 노드를 확인할 수 있어야 합니다. API 서버가 노드 이름을 확인할 수 없는 경우 프록시된 API 호출이 실패할 수 있으며 pod에서 로그를 검색할 수 없습니다.
라우트	***.apps.<cluster_name>.<base_domain>.**	애플리케이션 인그레스 로드 밸런서를 참조하는 와일드카드 DNS A/AAA 또는 CNAME 레코드입니다. 애플리케이션 인그레스 로드 밸런서는 Ingress 컨트롤러 Pod를 실행하는 머신을 대상으로 합니다. Ingress 컨트롤러 Pod는 기본적으로 컴퓨팅 머신에서 실행됩니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다. 예를 들어 **console-openshift-console.apps.<cluster_name>.<base_domain>**은 OpenShift Container Platform 콘솔의 와일드카드 경로로 사용됩니다.
부트스트랩 시스템	**bootstrap.<cluster_name>.<base_domain>.**	부트스트랩 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.
컨트롤 플레인 머신	**<master><n>.<cluster_name>.<base_domain>.**	컨트롤 플레인 노드의 각 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.
컴퓨팅 머신	**<worker><n>.<cluster_name>.<base_domain>.**	작업자 노드의 각 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.

참고

OpenShift Container Platform 4.4 이상에서는 DNS 구성에서 etcd 호스트 및 SRV 레코드를 지정할 필요가 없습니다.

작은 정보

dig 명령을 사용하여 이름과 역방향 이름을 확인할 수 있습니다. 자세한 검증 단계는 *사용자 프로비저닝 인프라의 DNS 확인* 섹션을 참조하십시오.

17.8.5.5.1. 사용자 프로비저닝 클러스터의 DNS 구성 예

이 섹션에서는 사용자 프로비저닝 인프라에 OpenShift Container Platform을 배포하기 위한 DNS 요구 사항을 충족하는 A 및 PTR 레코드 구성 샘플을 제공합니다. 샘플은 하나의 DNS 솔루션을 선택하기 위한 조언을 제공하기 위한 것이 아닙니다.

이 예제에서 클러스터 이름은 ocp4이고 기본 도메인은 example.com입니다.

사용자 프로비저닝 클러스터의 DNS A 레코드 구성 예

다음 BIND 영역 파일의 예제에서는 사용자가 프로비저닝한 클러스터의 이름 확인을 위한 샘플 A 레코드를 보여줍니다.

예 17.19. 샘플 DNS 영역 데이터베이스

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
master0.ocp4.example.com. IN A 192.168.1.97 5
master1.ocp4.example.com. IN A 192.168.1.98 6
master2.ocp4.example.com. IN A 192.168.1.99 7
```

```
;
worker0.ocp4.example.com. IN A 192.168.1.11 8
worker1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1

Kubernetes API의 이름 확인을 제공합니다. 레코드는 **API** 로드 밸런서의 **IP** 주소를 나타냅니다.

2

Kubernetes API의 이름 확인을 제공합니다. 레코드는 **API** 로드 밸런서의 **IP** 주소를 참조하며 내부 클러스터 통신에 사용됩니다.

3

와일드카드 경로의 이름 확인을 제공합니다. 레코드는 애플리케이션 인그레스 로드 밸런서의 **IP** 주소를 나타냅니다. 애플리케이션 인그레스 로드 밸런서는 **Ingress** 컨트롤러 **Pod**를 실행하는 머신을 대상으로 합니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

참고

이 예제에서는 **Kubernetes API** 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 **API** 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

4

부트스트랩 시스템의 이름 확인을 제공합니다.

5 6 7

컨트롤 플레인 시스템의 이름 확인을 제공합니다.

8 9

컴퓨팅 시스템의 이름 확인을 제공합니다.

사용자 프로비저닝 클러스터의 **DNS PTR** 레코드 구성 예

다음 예제 BIND 영역 파일은 사용자 프로비저닝 클러스터의 역방향 이름 확인을 위한 샘플 PTR 레코드를 보여줍니다.

예 17.20. 역방향 레코드의 샘플 DNS 영역 데이터베이스

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR master0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR master1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR master2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com. 8
;
;EOF
```

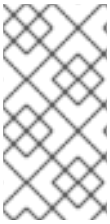
1 Kubernetes API의 역방향 DNS 확인을 제공합니다. PTR 레코드는 API 로드 밸런서의 레코드 이름을 참조합니다.

2 Kubernetes API의 역방향 DNS 확인을 제공합니다. PTR 레코드는 API 로드 밸런서의 레코드 이름을 참조하며 내부 클러스터 통신에 사용됩니다.

3 부트스트랩 시스템의 역방향 DNS 확인을 제공합니다.

4 5 6 컨트롤 플레인 시스템의 역방향 DNS 확인을 제공합니다.

7 8 컴퓨팅 시스템의 역방향 DNS 확인을 제공합니다.



참고

OpenShift Container Platform 애플리케이션 와일드카드에는 PTR 레코드가 필요하지 않습니다.

17.8.5.6. 사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구사항

OpenShift Container Platform을 설치하기 전에 API 및 애플리케이션 인그레스 로드 밸런싱 인프라를 프로비저닝해야 합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 API 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

참고

RHEL(Red Hat Enterprise Linux) 인스턴스를 사용하여 API 및 애플리케이션 인그레스 로드 밸런서를 배포하려면 RHEL 서브스크립션을 별도로 구매해야 합니다.

로드 밸런서 인프라는 다음 요구 사항을 충족해야 합니다.

1. API 로드 밸런서: 플랫폼과 상호 작용하고 플랫폼을 구성할 수 있도록 사용자(인간과 시스템 모두)에게 공통 끝점을 제공합니다. 다음 조건을 설정합니다.

   - Layer 4 로드 밸런싱 전용입니다. 이를 Raw TCP, SSL Passthrough 또는 SSL Bridge 모드라고 합니다. SSL Bridge 모드를 사용하는 경우, API 경로에 대해 SNI(Server Name Indication, 서버 이름 표시)를 활성화해야 합니다.
   - 스테이트리스 로드 밸런싱 알고리즘입니다. 옵션은 로드 밸런서 구현에 따라 달라집니다.

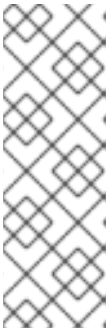
   참고

   API 로드 밸런서가 제대로 작동하기 위해 세션 지속성이 필요하지 않습니다.

   로드 밸런서의 전면과 후면 모두에서 다음 포트를 구성하십시오.

표 17.82. API 로드 밸런서

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
6443	부트스트랩 및 컨트롤 플레인. 부트스트랩 시스템이 클러스터 컨트롤 플레인을 초기화한 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다. API 서버 상태 검사 프로브에 대한 **/readyz** 끝점을 구성해야 합니다.	X	X	Kubernetes API 서버
22623	부트스트랩 및 컨트롤 플레인. 부트스트랩 시스템이 클러스터 컨트롤 플레인을 초기화한 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다.	X		시스템 구성 서버



참고

API 서버가 /readyz 엔드포인트를 해제하는 시점부터 풀에서 API 서버 인스턴스가 제거되는 시점까지 시간이 30초를 넘지 않도록 로드 밸런서를 구성해야 합니다. /readyz가 오류를 반환하거나 정상 상태가 된 후 정해진 시간 안에 끝점이 제거 또는 추가되어야 합니다. 5초 또는 10초의 프로빙 주기(두 번의 성공적인 요청은 정상 상태, 세 번의 요청은 비정상 상태)는 충분한 테스트를 거친 값입니다.

2. **애플리케이션 인그레스 로드 밸런서: 클러스터 외부에서 유입되는 애플리케이션 트래픽에 대한 인그래스 포인트를 제공합니다. 다음 조건을 설정합니다.**

- **Layer 4 로드 밸런싱 전용입니다. 이를 Raw TCP, SSL Passthrough 또는 SSL Bridge 모드라고 합니다. SSL Bridge 모드를 사용하는 경우 인그레스 경로에 대해 SNI(Server Name Indication, 서버 이름 표시)를 활성화해야 합니다.**

- **사용 가능한 옵션과 플랫폼에서 호스팅되는 애플리케이션 유형에 따라 연결 기반 또는 세션 기반 지속성이 권장됩니다.**

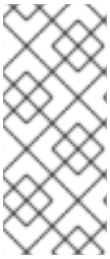
작은 정보

애플리케이션 인그레스 로드 밸런서에서 클라이언트의 실제 IP 주소를 확인할 수 있는 경우 소스 IP 기반 세션 지속성을 활성화하면 엔드 투 엔드 TLS 암호화를 사용하는 애플리케이션의 성능을 향상시킬 수 있습니다.

로드 밸런서의 전면과 후면 모두에서 다음 포트를 구성하십시오.

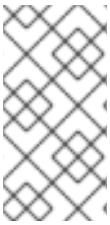
표 17.83. 애플리케이션 인그레스 로드 밸런서

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
443	기본적으로 인그레스 컨트롤러 pod, 컴퓨팅 또는 작업자를 실행하는 시스템입니다.	X	X	HTTPS 트래픽
80	기본적으로 인그레스 컨트롤러 pod, 컴퓨팅 또는 작업자를 실행하는 시스템입니다.	X	X	HTTP 트래픽
1936	기본적으로 Ingress 컨트롤러 Pod를 실행하는 작업자 노드입니다. 수신 상태 점검 프로브에 대해 **/healthz/ready** 끝점을 구성해야 합니다.	X	X	HTTP 트래픽



참고

컴퓨팅 노드가 0인 3-노드 클러스터를 배포하는 경우 Ingress 컨트롤러 Pod는 컨트롤 플레인 노드에서 실행됩니다. 3-노드 클러스터 배포에서 HTTP 및 HTTPS 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다.

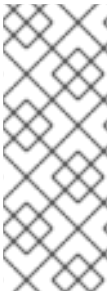


참고

인그레스 라우터에 대한 작업 구성이 OpenShift Container Platform 클러스터에 필요합니다. 컨트롤 플레인 초기화 후 인그레스 라우터를 설정해야 합니다.

17.8.5.6.1. 사용자 프로비저닝 클러스터의 로드 밸런서 구성 예

이 섹션에서는 사용자 프로비저닝 클러스터의 로드 밸런싱 요구 사항을 충족하는 API 및 애플리케이션 수신 로드 밸런서 구성 예를 제공합니다. 샘플은 HAProxy 로드 밸런서에 대한 /etc/haproxy/haproxy.cfg 구성입니다. 이 예제에서는 하나의 로드 밸런싱 솔루션을 선택하기 위한 조언을 제공하는 것을 목적으로 하지 않습니다.



참고

이 예제에서는 Kubernetes API 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 API 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

예 17.21. API 및 애플리케이션 인그레스 로드 밸런서 구성 샘플

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
frontend stats
  bind *:1936
  mode            http
  log             global
  maxconn 10
  stats enable
  stats hide-version
  stats refresh 30s
  stats show-node
  stats show-desc Stats for ocp4 cluster 1
  stats auth admin:ocp4
  stats uri /stats
listen api-server-6443 2
  bind *:6443
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:6443 check inter 1s backup 3
  server master0 master0.ocp4.example.com:6443 check inter 1s
  server master1 master1.ocp4.example.com:6443 check inter 1s
  server master2 master2.ocp4.example.com:6443 check inter 1s
listen machine-config-server-22623 4
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup 5
  server master0 master0.ocp4.example.com:22623 check inter 1s
  server master1 master1.ocp4.example.com:22623 check inter 1s
  server master2 master2.ocp4.example.com:22623 check inter 1s
listen ingress-router-443 6
  bind *:443
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:443 check inter 1s
  server worker1 worker1.ocp4.example.com:443 check inter 1s
listen ingress-router-80 7
  bind *:80
  mode tcp
```

```
balance source
server worker0 worker0.ocp4.example.com:80 check inter 1s
server worker1 worker1.ocp4.example.com:80 check inter 1s
```

1 이 예에서 클러스터 이름은 **ocp4** 입니다.

2 포트 **6443**은 **Kubernetes API** 트래픽을 처리하고 컨트롤 플레인 시스템을 가리킵니다.

3 5 부트스트랩 항목은 **OpenShift Container Platform** 클러스터 설치 전에 있어야 하며 부트스트랩 프로세스가 완료된 후 제거해야 합니다.

4 포트 **22623**은 머신 구성 서버 트래픽을 처리하고 컨트롤 플레인 시스템을 가리킵니다.

6 포트 **443**은 **HTTPS** 트래픽을 처리하고 **Ingress** 컨트롤러 **Pod**를 실행하는 시스템을 가리킵니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

7 포트 **80**은 **HTTP** 트래픽을 처리하고 **Ingress** 컨트롤러 **Pod**를 실행하는 머신을 가리킵니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

참고

컴퓨팅 노드가 **0**인 **3-**노드 클러스터를 배포하는 경우 **Ingress** 컨트롤러 **Pod**는 컨트롤 플레인 노드에서 실행됩니다. **3-**노드 클러스터 배포에서 **HTTP** 및 **HTTPS** 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다.

작은 정보

HAProxy를 로드 밸런서로 사용하는 경우 **HAProxy** 노드에서 **netstat -nltupe**를 실행하여 **haproxy** 프로세스가 포트 **6443, 22623, 443** 및 **80**에서 수신 대기 중인지 확인할 수 있습니다.

참고

HAProxy를 로드 밸런서로 사용하고 SELinux가 enforcing으로 설정된 경우 HAProxy 서비스가 setsebool -P haproxy_connect_any=1을 실행하여 구성된 TCP 포트에 바인딩할 수 있는지 확인해야 합니다.

17.8.6. 사용자 프로비저닝 인프라 준비

사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치하기 전에 기본 인프라를 준비해야 합니다.

이 섹션에서는 OpenShift Container Platform 설치를 준비하기 위해 클러스터 인프라를 설정하는 데 필요한 높은 수준의 단계에 대해 자세히 설명합니다. 여기에는 클러스터 노드에 대한 IP 네트워킹 및 네트워크 연결 구성, 방화벽을 통해 필요한 포트 활성화, 필수 DNS 및 로드 밸런싱 인프라를 설정하는 작업이 포함됩니다.

준비 후 클러스터 인프라는 *사용자 프로비저닝 인프라가 있는 클러스터의 요구 사항* 섹션에 설명된 요구 사항을 충족해야 합니다.

사전 요구 사항

- OpenShift Container Platform 4.x 테스트된 통합 페이지를 검토했습니다.
- *사용자 프로비저닝 인프라가 있는 클러스터의 요구 사항* 섹션에 자세히 설명된 인프라 요구 사항을 검토했습니다.

프로세스

1. DHCP를 사용하여 클러스터 노드에 IP 네트워킹 구성을 제공하는 경우 DHCP 서비스를 구성합니다.
   a. 노드의 영구 IP 주소를 DHCP 서버 구성에 추가합니다. 구성에서 관련 네트워크 인터페이스의 MAC 주소를 각 노드의 의도한 IP 주소와 일치시킵니다.
   b. DHCP를 사용하여 클러스터 시스템의 IP 주소 지정을 구성하는 경우 시스템은 DHCP를 통해 DNS 서버 정보도 가져옵니다. DHCP 서버 구성을 통해 클러스터 노드에서 사용하는 영구 DNS 서버 주소를 정의합니다.

참고

DHCP 서비스를 사용하지 않는 경우 **RHCOS** 설치 시 **IP** 네트워킹 구성과 **DNS** 서버의 주소를 노드에 제공해야 합니다. **ISO** 이미지에서 설치하는 경우 부팅 인수로 전달할 수 있습니다. 고정 **IP** 프로비저닝 *및 고급 네트워킹 옵션에 대한 자세한 내용은 **RHCOS** 설치 및 **OpenShift Container Platform** 부트스트랩 프로세스 시작* 섹션을 참조하십시오.

c.

DHCP 서버 구성에 클러스터 노드의 호스트 이름을 정의합니다. 호스트 이름 고려 사항에 대한 자세한 내용은 ***DHCP**를 통해 클러스터 노드 호스트 이름 설정* 섹션을 참조하십시오.

참고

DHCP 서비스를 사용하지 않는 경우 클러스터 노드는 역방향 **DNS** 조회를 통해 호스트 이름을 가져옵니다.

2.

네트워크 인프라가 클러스터 구성 요소 간 필수 네트워크 연결을 제공하는지 확인합니다. 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 인프라 섹션의 네트워킹 요구 사항* 섹션을 참조하십시오.

3.

OpenShift Container Platform 클러스터 구성 요소가 통신하는 데 필요한 포트를 활성화하도록 방화벽을 구성합니다. 필요한 포트에 대한 자세한 내용은 *사용자 프로비저닝 인프라 섹션의 네트워킹 요구 사항* 섹션을 참조하십시오.

4.

클러스터에 필요한 **DNS** 인프라를 설정합니다.

a.

Kubernetes API, 애플리케이션 와일드카드, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템의 **DNS** 이름 확인을 구성합니다.

b.

Kubernetes API, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템에 대한 역방향 **DNS** 확인을 구성합니다.

OpenShift Container Platform DNS 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 **DNS** 요구 사항* 섹션을 참조하십시오.

5.

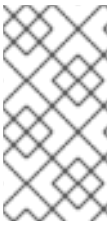
DNS 구성을 확인합니다.

a.
설치 노드에서 **Kubernetes API**의 레코드 이름, 와일드카드 경로 및 클러스터 노드에 대해 **DNS** 조회를 실행합니다. 응답의 **IP** 주소가 올바른 구성 요소에 해당하는지 확인합니다.

b.
설치 노드에서 로드 밸런서 및 클러스터 노드의 **IP** 주소에 대해 역방향 **DNS** 조회를 실행합니다. 응답의 레코드 이름이 올바른 구성 요소에 해당하는지 확인합니다.

자세한 **DNS** 검증 단계는 *사용자 프로비저닝 인프라에 대한 DNS 확인* 섹션을 참조하십시오.

6.
필요한 **API** 및 애플리케이션 수신 로드 밸런싱 인프라를 프로비저닝합니다. 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구 사항* 섹션을 참조하십시오.



참고

일부 로드 밸런싱 솔루션에는 로드 밸런싱을 초기화하기 전에 클러스터 노드의 **DNS** 이름을 확인해야 합니다.

17.8.7. 사용자 프로비저닝 인프라에 대한 DNS 확인 검증

사용자 프로비저닝 인프라에 **OpenShift Container Platform**을 설치하기 전에 **DNS** 구성을 확인할 수 있습니다.

중요

클러스터를 설치하기 전에 이 섹션에 설명된 검증 단계를 성공해야 합니다.

사전 요구 사항

- 사용자 프로비저닝 인프라에 필요한 **DNS** 레코드를 구성했습니다.

프로세스

1.
설치 노드에서 **Kubernetes API**의 레코드 이름, 와일드카드 경로 및 클러스터 노드에 대해 **DNS** 조회를 실행합니다. 응답에 포함된 **IP** 주소가 올바른 구성 요소에 해당하는지 확인합니다.

a.
Kubernetes API 레코드 이름을 조회합니다. 결과가 **API** 로드 밸런서의 **IP** 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
```

1

<nameserver_ip>를 네임서버의 IP 주소로, <cluster_name>을 클러스터 이름으로, <base_domain>을 기본 도메인 이름으로 바꿉니다.

출력 예

```
api.ocp4.example.com.  0 IN A 192.168.1.5
```

b.

Kubernetes 내부 API 레코드 이름을 조회합니다. 결과가 API 로드 밸런서의 IP 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
```

출력 예

```
api-int.ocp4.example.com.  0 IN A 192.168.1.5
```

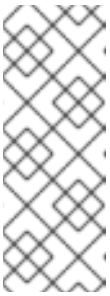
c.

예제 *.apps.<cluster_name>.<base_domain>을 테스트합니다. DNS 와일드카드를 조회합니다. 모든 애플리케이션 와일드카드 조회는 애플리케이션 인그레스 로드 밸런서의 IP 주소로 확인되어야 합니다.

```
$ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.
<base_domain>
```

출력 예

```
random.apps.ocp4.example.com.  0 IN A 192.168.1.5
```



참고

예제 출력에서는 **Kubernetes API** 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 **API** 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

random 항목을 다른 와일드카드 값으로 교체할 수 있습니다. 예를 들어 **OpenShift Container Platform** 콘솔의 경로를 쿼리할 수 있습니다.

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

출력 예

```
console-openshift-console.apps.ocp4.example.com. 0 IN A 192.168.1.5
```

d.

부트스트랩 **DNS** 레코드 이름에 대해 조회를 실행합니다. 결과가 부트스트랩 노드의 **IP** 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.
<base_domain>
```

출력 예

```
bootstrap.ocp4.example.com.  0 IN A 192.168.1.96
```

e.

이 방법을 사용하여 컨트롤 플레인 및 컴퓨팅 노드의 **DNS** 레코드 이름에 대해 조회를 수행합니다. 결과가 각 노드의 **IP** 주소에 해당하는지 확인합니다.

2. 설치 노드에서 로드 밸런서 및 클러스터 노드의 IP 주소에 대해 역방향 DNS 조회를 실행합니다. 응답에 포함된 레코드 이름이 올바른 구성 요소에 해당하는지 확인합니다.

a. API 로드 밸런서의 IP 주소에 대해 역방향 조회를 수행합니다. 응답에 Kubernetes API 및 Kubernetes 내부 API의 레코드 이름이 포함되어 있는지 확인합니다.

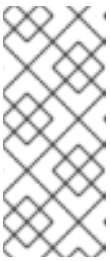
```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

출력 예

```
5.1.168.192.in-addr.arpa. 0 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 0 IN PTR api.ocp4.example.com. 2
```

1 Kubernetes 내부 API의 레코드 이름을 제공합니다.

2 Kubernetes API의 레코드 이름을 제공합니다.



참고

OpenShift Container Platform 애플리케이션 와일드카드에는 PTR 레코드가 필요하지 않습니다. 애플리케이션 인그레스 로드 밸런서의 IP 주소에 대한 역방향 DNS 확인에는 유효성 검사 단계가 필요하지 않습니다.

b. 부트스트랩 노드의 IP 주소에 대해 역방향 조회를 수행합니다. 결과가 부트스트랩 노드의 DNS 레코드 이름을 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

출력 예

```
96.1.168.192.in-addr.arpa. 0 IN PTR bootstrap.ocp4.example.com.
```

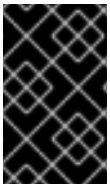
c. 이 방법을 사용하여 컨트롤 플레인 및 컴퓨팅 노드의 IP 주소에 대해 역방향 조회를 수행합니다. 결과가 각 노드의 DNS 레코드 이름과 일치하는지 확인합니다.

17.8.8. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 core 사용자의 ~/.ssh/authorized_keys 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 core로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. ./openshift-install gather 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.



중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

참고

AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

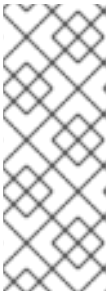
프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1

새 SSH 키의 경로 및 파일 이름(예: ~/.ssh/id_ed25519)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 '~/.ssh 디렉터리에 있는지 확인하십시오.



참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 ed25519 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 rsa 또는 ecdsa 알고리즘을 사용하는 키를 생성합니다.

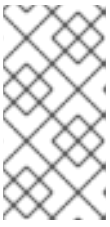
2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 ~/.ssh/id_ed25519.pub 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 ./openshift-install gather 명령을 사용하려는 경우 필요합니다.



참고

일부 배포에서는 ~/.ssh/id_rsa 및 ~/.ssh/id_dsa와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. ssh-agent 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 **FIPS** 모드인 경우 **FIPS** 호환 알고리즘만 사용하여 **SSH** 키를 생성합니다. 키는 **RSA** 또는 **ECDSA**여야 합니다.

4. **ssh-agent**에 **SSH** 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 **SSH** 개인 키의 경로 및 파일 이름을 지정합니다(예: ~/.ssh/id_ed25519).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- **OpenShift Container Platform**을 설치할 때 **SSH** 공개 키를 설치 프로그램에 지정합니다. 프로비저닝하는 인프라에 클러스터를 설치하는 경우 설치 프로그램에 키를 제공해야 합니다.

17.8.9. 수동으로 설치 구성 파일 만들기

OpenShift Container Platform을 사용자가 프로비저닝한 설치의 경우 설치 구성 파일을 수동으로 생성합니다.

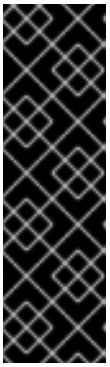
사전 요구 사항

- 로컬 시스템에 설치 프로그램에 제공할 **SSH** 공개 키가 있습니다. 키는 디버깅 및 재해 복구를 위해 클러스터 노드에 대한 **SSH** 인증에 사용됩니다.
- **OpenShift Container Platform** 설치 프로그램과 클러스터의 풀 시크릿이 있습니다.
- 명령 출력에서 **imageContentSources** 섹션을 가져와서 리포지토리를 미러링합니다.
- 미러 레지스트리에 대한 인증서의 내용을 가져옵니다.

프로세스

1. 필요한 설치 자산을 저장할 설치 디렉터리를 만듭니다.

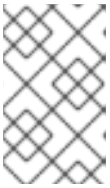
```
$ mkdir <installation_directory>
```



중요

디렉터리를 만들어야 합니다. 부트스트랩 **X.509** 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 **OpenShift Container Platform** 버전에서 설치 파일을 복사할 때는 주의하십시오.

2. 샘플 **install-config.yaml** 파일 템플릿을 사용자 지정하여 **<installation_directory>**에 저장합니다.



참고

이 설정 파일의 이름을 **install-config.yaml**로 지정해야 합니다.

- **docker.io**와 같이 **RHCOS**가 기본적으로 신뢰하는 레지스트리를 사용하는 경우를 제외하고, **additionalTrustBundle** 섹션에 있는 미러 리포지토리에 대한 인증서 내용을 제공해야 합니다. 대부분의 경우 미러 인증서를 제공해야 합니다.

- 리포지토리를 미러링하려면 명령 출력의 **imageContentSources** 섹션을 삽입해야 합니다.

참고

일부 플랫폼 유형의 경우 대체로 **./openshift-install create install-config --dir <installation_directory>** 를 실행하여 **install-config.yaml** 파일을 생성할 수 있습니다. 프롬프트에서 클러스터 구성에 대한 세부 정보를 제공할 수 있습니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.

중요

install-config.yaml 파일은 설치 과정의 다음 단계에서 사용됩니다. 이 시점에서 이를 백업해야 합니다.

17.8.9.1. VMware vSphere용 샘플 install-config.yaml 파일

install-config.yaml 파일을 사용자 지정하여 **OpenShift Container Platform** 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.

```
apiVersion: v1
baseDomain: example.com 1
compute:
- hyperthreading: Enabled 2 3
  name: worker
  replicas: 0 4
controlPlane:
  hyperthreading: Enabled 5 6
  name: master
  replicas: 3 7
metadata:
  name: test 8
platform:
  vsphere:
    vcenter: your.vcenter.server 9
    username: username 10
    password: password 11
    datacenter: datacenter 12
    defaultDatastore: datastore 13
    folder: "/<datacenter_name>/vm/<folder_name>/<subfolder_name>" 14
```

```
fips: false 15
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email":
"you@example.com"}}}' 16
sshKey: 'ssh-ed25519 AAAA...' 17
additionalTrustBundle: | 18
  -----BEGIN CERTIFICATE-----
  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
imageContentSources: 19
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

1

클러스터의 기본 도메인입니다. 모든 **DNS** 레코드는 이 기본 도메인의 하위 도메인이어야 하며 클러스터 이름을 포함해야 합니다.

2 5

controlPlane 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 현재 두 섹션이 모두 단일 시스템 풀을 정의하지만 향후 출시되는 **OpenShift Container Platform** 버전은 설치 과정에서 여러 컴퓨팅 풀 정의를 지원할 수 있습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

3 6

동시 멀티스레딩 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. 일부 클러스터 시스템에서 동시 멀티스레딩을 비활성화할 경우에는 해당 멀티스레딩을 모든 클러스터 시스템에서 비활성화해야 합니다.

중요

동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다. 동시 멀티스레딩을 비활성화하는 경우 시스템은 **8**개 이상의 **CPU**와 **32GB**의 **RAM**을 사용해야 합니다.

4

replicas 매개변수의 값을 **0**으로 설정해야 합니다. 이 매개변수는 클러스터를 생성하고 관리하는 작업자 수를 제어합니다. 이는 사용자 프로비저닝 인프라를 사용할 때 클러스터가 실행하지 않는 기능입니다. **OpenShift Container Platform** 설치를 완료하기 전에 클러스터에서 사용할 작업자 시스템을 수동으로 배포해야 합니다.

7

클러스터에 추가하는 컨트롤 플레인 시스템의 수입니다. 클러스터에서 이 값을 클러스터의 **etcd** 끝점 수로 사용하므로 이 값은 배포하는 컨트롤 플레인 시스템의 수와 일치해야 합니다.

8

DNS 레코드에 지정한 클러스터 이름입니다.

9

vCenter 서버의 정규화된 호스트 이름 또는 **IP** 주소입니다.

10

서버에 액세스하기 위한 사용자의 이름입니다. 이 사용자에게는 최소한 **vSphere**에서 정적 또는 동적 영구 볼륨 프로비저닝 에 필요한 역할과 권한이 있어야 합니다.

11

vSphere 사용자와 연결된 암호입니다.

12

vSphere 데이터 센터입니다.

13

사용할 기본 **vSphere** 데이터 저장소입니다.

14

선택사항: 설치 관리자 프로비저닝 인프라의 경우 설치 프로그램이 가상 머신을 생성하는 기존 폴더의 절대 경로(예: /<datacenter_name>/vm/<folder_name>/<subfolder_name>). 이 값을 제공하지 않으면 설치 프로그램이 인프라 **ID**로 이름이 지정된 데이터 센터 가상 머신 폴더에 최상위 폴더를 만듭니다. 클러스터에 인프라를 제공하는 경우에는 이 매개변수를 생략합니다.

15

FIPS 모드 활성화 또는 비활성화 여부입니다. 기본적으로 **FIPS** 모드는 비활성화됩니다. **FIPS** 모드가 활성화되면 **OpenShift Container Platform**이 실행되는 **RHCOS(Red Hat Enterprise Linux CoreOS)** 시스템에서 기본 **Kubernetes** 암호화 제품군은 우회하고 **RHCOS**와 함께 제공되는 암호화 모듈을 대신 사용합니다.

중요

FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 **OpenShift Container Platform** 배포에서만 지원됩니다.

16

17

RHCOS(Red Hat Enterprise Linux CoreOS)의 **core** 사용자에 대한 기본 **SSH** 키의 공용 부분입니다.

참고

설치 디버깅 또는 재해 복구 수행을 원하는 프로덕션 환경 **OpenShift Container Platform** 클러스터의 경우 **ssh-agent** 프로세스에 사용할 **SSH** 키를 지정하십시오.

18

미러 레지스트리에 사용한 인증서 파일의 내용을 제공하십시오.

19

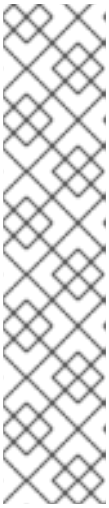
명령 출력에서 **imageContentSources** 섹션을 제공하여 리포지토리를 미러링하십시오.

17.8.9.2. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 **HTTP** 또는 **HTTPS** 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 **OpenShift Container Platform** 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 **API**에 대한 호출을 포함하여 모든 클러스터 발신(**Egress**) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.



참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 **Red Hat OpenStack Platform (RHOSP)**에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. 클러스터 외부에서 **HTTP** 연결을 구축하는 데 사용할 프록시 **URL**입니다. **URL** 스키마는 **http**여야 합니다.

2. 클러스터 외부에서 **HTTPS** 연결을 구축하는 데 사용할 프록시 **URL**입니다.

3. 대상 도메인 이름, **IP** 주소 또는 프록시에서 제외할 기타 네트워크 **CIDR**로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. ***를 사용하여 모든 대상에 대해 프록시를 바이패스합니다. **vCenter**의 **IP** 주소와 해당 시스템에 사용하는 **IP** 범위를 포함해야 합니다.

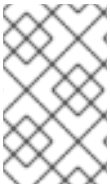
4.

참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 **OpenShift Container Platform**을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 **nil**이 됩니다.



참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

17.8.10. Kubernetes 매니페스트 및 Ignition 설정 파일 생성

일부 클러스터 정의 파일을 수정하고 클러스터 시스템을 수동으로 시작해야 하므로 클러스터가 시스템을 구성하는 데 필요한 **Kubernetes** 매니페스트 및 **Ignition** 구성 파일을 사용자가 생성해야 합니다.

설치 구성 파일은 **Kubernetes** 매니페스트로 변환됩니다. 매니페스트는 나중에 클러스터 머신을 구성하는 데 사용되는 **Ignition** 구성 파일로 래핑됩니다.

중요

- **OpenShift Container Platform** 설치 프로그램에서 생성하는 **Ignition** 구성 파일에 **24**시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 **24**시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 **kubelet** 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(**CSR**)을 수동으로 승인해야 합니다. 자세한 내용은 ***Recovering from expired control plane certificates*** 문서를 참조하십시오.

- 클러스터를 설치한 후 **24**시간에서 **22**시간까지의 인증서가 교체되기 때문에 생성된 후 **12**시간 이내에 **Ignition** 구성 파일을 사용하는 것이 좋습니다. **12**시간 이내에 **Ignition** 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

사전 요구 사항

- **OpenShift Container Platform** 설치 프로그램을 가져오셨습니다. 제한된 네트워크 설치의 경우, 해당 파일은 미러 호스트에 있습니다.

- **install-config.yaml** 설치 구성 파일을 생성하셨습니다.

프로세스

1. **OpenShift Container Platform** 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터에 대한 **Kubernetes** 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>**는 사용자가 만든 **install-config.yaml** 파일이 포함된 설치 디렉터리를 지정합니다.

2. 컨트롤 플레인 시스템 및 컴퓨팅 머신 세트를 정의하는 **Kubernetes** 매니페스트 파일을 제거합니다.

```
$ rm -f openshift/99_openshift-cluster-api_master-machines-*.yaml
openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

이러한 리소스는 사용자가 직접 생성하고 관리하기 때문에 초기화할 필요가 없습니다.

- 시스템 API로 머신 세트 파일을 보존하여 컴퓨팅 시스템을 생성할 수 있지만 사용자 환경과 일치하도록 해당 참조를 업데이트해야 합니다.

3. <installation_directory>/manifests/cluster-scheduler-02-config.yml Kubernetes 매니페스트 파일의 mastersSchedulable 매개변수가 false로 설정되어 있는지 확인합니다. 이 설정으로 인해 컨트롤 플레인 머신에서 포드가 예약되지 않습니다.

   a. <installation_directory>/manifests/cluster-scheduler-02-config.yml 파일을 엽니다.

   b. mastersSchedulable 매개변수를 찾아서 값을 False로 설정되어 있는지 확인합니다.

   c. 파일을 저장하고 종료합니다.

4. Ignition 구성 파일을 생성하려면 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행합니다.

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 <installation_directory>는 동일한 설치 디렉터리를 지정합니다.

설치 디렉터리의 부트스트랩, 컨트롤 플레인 및 컴퓨팅 노드에 대한 Ignition 구성 파일이 생성됩니다. kubeadmin-password 및 kubeconfig 파일은 ./<installation_directory>/auth 디렉터리에 생성됩니다.

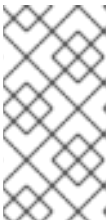
```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

17.8.11. chrony 타임 서비스 설정

chrony.conf 파일의 내용을 수정하고 해당 내용을 머신 구성으로 노드에 전달하여 chrony 타임 서비스 (chronyd)에서 사용하는 시간 서버 및 관련 구성을 설정해야 합니다.

프로세스

1. chrony.conf 파일의 내용을 포함하여 Butane config를 만듭니다. 예를 들어 작업자 노드에 chrony를 구성하려면 99-worker-chrony.bu 파일을 만듭니다.



참고

Butane에 대한 자세한 내용은 “Butane 을 사용하여 머신 구성 생성”을 참조하십시오.

```
variant: openshift
version: 4.9.0
metadata:
  name: 99-worker-chrony (1)
  labels:
    machineconfiguration.openshift.io/role: worker (2)
storage:
  files:
  - path: /etc/chrony.conf
    mode: 0644 (3)
    overwrite: true
    contents:
      inline: |
        pool 0.rhel.pool.ntp.org iburst (4)
        driftfile /var/lib/chrony/drift
        makestep 1.0 3
        rtcsync
        logdir /var/log/chrony
```

(1) (2) 컨트롤 플레인 노드에서 두 위치에 있는 master를 worker로 대체합니다.

(3) 시스템 구성 파일에서 mode 필드의 8진수 값 모드를 지정합니다. 파일을 만들고 변경 사항을 적용하면 모드가 10진수 값으로 변환됩니다. oc get mc <mc-name> -o yaml 명령을 사용하여 YAML 파일을 확인할 수 있습니다.

(4)

2. Butane을 사용하여 노드에 전달할 구성이 포함된 MachineConfig 파일 99-worker-chrony.yaml을 생성합니다.

```
$ butane 99-worker-chrony.bu -o 99-worker-chrony.yaml
```

3. 다음 두 가지 방법 중 하나로 설정을 적용하십시오.

- 클러스터가 아직 실행되지 않은 경우 매니페스트 파일을 생성한 후 <installation_directory>/openshift 디렉터리에 MachineConfig 개체 파일을 추가한 다음 클러스터를 계속 작성합니다.
- 클러스터가 이미 실행중인 경우 다음과 같은 파일을 적용합니다.

```
$ oc apply -f ./99-worker-chrony.yaml
```

17.8.12. 인프라 이름 추출

Ignition 구성 파일에는 VMware vSpher에서 클러스터를 고유하게 식별하는 데 사용할 수 있는 고유한 클러스터 ID가 포함되어 있습니다. 클러스터 ID를 가상 머신 폴더의 이름으로 사용하려면 해당 ID를 추출해야 합니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받으셨습니다.
- 클러스터에 대한 Ignition 구성 파일을 생성하셨습니다.
- jq CLI를 설치하셨습니다.

프로세스

- Ignition 구성 파일 메타데이터에서 인프라 이름을 추출하여 확인하려면 다음 명령을 실행하십시오.

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

출력 예

```
openshift-vw9j6 1
```

1 이 명령의 출력은 클러스터 이름과 임의의 문자열입니다.

17.8.13. RHCOS 설치 및 OpenShift Container Platform 부트스트랩 프로세스 시작

VMware vSphere의 사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치하려면 vSphere 호스트에 RHCOS(Red Hat Enterprise Linux Core OS)를 설치해야 합니다. RHCOS를 설치할 때 설치 중인 머신 유형에 대해 OpenShift Container Platform 설치 프로그램에서 생성한 Ignition 구성 파일을 제공해야 합니다. 적합한 네트워킹, DNS 및 로드 밸런싱 인프라를 구성한 경우 RHCOS 머신이 재부팅된 후 OpenShift Container Platform 부트스트랩 프로세스가 자동으로 시작됩니다.

사전 요구 사항

- 클러스터에 대한 Ignition 구성 파일이 있어야 합니다.
- 사용자 컴퓨터에서 액세스할 수 있고 생성한 시스템이 액세스할 수 있는 HTTP 서버에 액세스할 수 있습니다.
- vSphere 클러스터 를 생성했습니다.

프로세스

1. 설치 프로그램에서 생성된 부트스트랩 Ignition 구성 파일 (<installation_directory>/bootstrap.ign)을 HTTP 서버에 업로드합니다. 이 파일의 URL을 기록

해 둡니다.

2. 부트스트랩 노드의 다음 보조 Ignition 구성 파일을 <installation_directory>/merge-bootstrap.ign으로 컴퓨터에 저장합니다.

```
{
  "ignition": {
    "config": {
      "merge": [
        {
          "source": "<bootstrap_ignition_config_url>", 1
          "verification": {}
        }
      ]
    },
    "timeouts": {},
    "version": "3.2.0"
  },
  "networkd": {},
  "passwd": {},
  "storage": {},
  "systemd": {}
}
```

1 호스팅한 부트스트랩 Ignition 구성 파일의 URL을 지정합니다.

부트스트랩 머신에 대한 VM(가상 머신)을 생성할 때 이 Ignition 구성 파일을 사용합니다.

3. 설치 프로그램이 생성한 다음 Ignition 구성 파일을 찾습니다.

- <installation_directory>/master.ign
- <installation_directory>/worker.ign
- <installation_directory>/merge-bootstrap.ign

4. Ignition 구성 파일을 Base64 인코딩으로 변환합니다. 이 절차의 뒷부분에서는 이러한 파일을 VM의 추가 구성 매개변수 guestinfo.ignition.config.data 에 추가해야 합니다.

예를 들어 **Linux** 운영 체제를 사용하는 경우 **base64** 명령을 사용하여 파일을 인코딩할 수 있습니다.

```
$ base64 -w0 <installation_directory>/master.ign > <installation_directory>/master.64
```

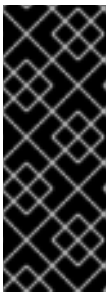
```
$ base64 -w0 <installation_directory>/worker.ign > <installation_directory>/worker.64
```

```
$ base64 -w0 <installation_directory>/merge-bootstrap.ign >
<installation_directory>/merge-bootstrap.64
```

중요

설치를 마친 후 클러스터에 컴퓨팅 시스템을 더 추가하려면 **Ignition** 구성 파일을 삭제하지 마십시오.

5. **RHCOS OVA** 이미지를 가져옵니다. **RHCOS** 이미지 미러 페이지에서 이미지를 사용할 수 있습니다.



중요

RHCOS 이미지는 **OpenShift Container Platform** 릴리스에 따라 변경되지 않을 수 있습니다. 설치하는 **OpenShift Container Platform** 버전과 같거나 그 이하의 버전 중 가장 최신 버전의 이미지를 다운로드해야 합니다. 지원되는 경우 **OpenShift Container Platform** 버전과 일치하는 이미지 버전을 사용합니다.

파일 이름에는 **rhcos-vmware.<architecture>.ova** 형식의 **OpenShift Container Platform** 버전 번호가 포함됩니다.

6. **vSphere Client**에서 **VM**을 저장할 데이터 센터 폴더를 생성합니다.

   a. **VMs and Templates** 보기를 클릭합니다.

   b. 데이터 센터 이름을 마우스 오른쪽 버튼으로 클릭합니다.

   c. **New Folder** → **New VM and Template Folder**를 클릭합니다.

d.

표시되는 창에서 폴더 이름을 입력합니다. **install-config.yaml** 파일에서 기존 폴더를 지정하지 않은 경우 인프라 **ID**와 동일한 이름으로 폴더를 생성합니다. **vCenter**는 이 폴더 이름을 사용하여 **Workspace** 구성에 적절한 위치에 스토리지를 동적으로 프로비저닝합니다.

7.

vSphere Client에서 **OVA** 이미지에 대한 템플릿을 생성한 다음 필요에 따라 템플릿을 복제합니다.

참고

다음 단계에서 템플릿을 생성한 다음 모든 클러스터 시스템에 대한 템플릿을 복제합니다. 그런 다음 **VM**을 프로비저닝할 때 복제된 머신 유형의 **Ignition** 구성 파일의 위치를 제공합니다.

a.

Hosts and Clusters 탭에서 클러스터 이름을 마우스 오른쪽 버튼으로 클릭하고 **Deploy OVF Template**을 선택합니다.

b.

Select an OVF 탭에서 다운로드한 **RHCOS OVA** 파일의 이름을 지정합니다.

c.

이름 및 폴더 선택 탭에서 템플릿의 가상 시스템 이름 (예: **Template-RHCOS**)을 설정합니다. **vSphere** 클러스터의 이름을 클릭하고 이전 단계에서 생성한 폴더를 선택합니다.

d.

Select a compute resource 탭에서 **vSphere** 클러스터 이름을 클릭합니다.

e.

Select storage 탭에서 **VM**의 스토리지 옵션을 구성합니다.

- 스토리지 기본 설정에 따라 **Thin Provision** 또는 **Thick Provision**을 선택합니다.
- **install-config.yaml** 파일에서 지정한 데이터 저장소를 선택합니다.

f.

Select network 탭에서 사용 가능한 경우 클러스터에 대해 구성한 네트워크를 지정합니다.

g.

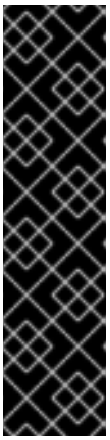
OVF 템플릿을 생성할 때 템플릿 사용자 지정 탭에 값을 지정하지 않거나 템플릿을 추가

로 구성하지 마십시오.

중요

원래 VM 템플릿을 시작하지 마십시오. VM 템플릿이 꺼져 있어야 하며 새 RHCOS 머신에 대해 복제해야 합니다. VM 템플릿을 시작하면 VM 템플릿이 플랫폼의 VM으로 구성되므로 시스템 세트가 구성을 적용할 수 있는 템플릿으로 사용되지 않습니다.

8. 선택 사항: 필요한 경우 VM 템플릿에서 구성된 가상 하드웨어 버전을 업데이트합니다. 자세한 내용은 가상 머신을 VMware 설명서의 최신 하드웨어 버전으로 업그레이드를 참조하십시오.



중요

필요한 경우 VM 템플릿의 하드웨어 버전을 버전 15로 업데이트하는 것이 좋습니다. 이제 vSphere에서 실행 중인 클러스터 노드에 하드웨어 버전 13을 사용하는 것이 더 이상 사용되지 않습니다. 가져온 템플릿의 기본값이 하드웨어 버전 13인 경우 VM 템플릿을 하드웨어 버전 15로 업그레이드하기 전에 ESXi 호스트가 6.7U3 이상인지 확인해야 합니다. vSphere 버전이 6.7U3 미만인 경우 이 업그레이드 단계를 건너뛸 수 있습니다. 그러나 향후 OpenShift Container Platform 버전은 6.7U3 미만의 하드웨어 버전 13 및 vSphere 버전에 대한 지원을 제거하도록 예약되어 있습니다.

9. 템플릿이 배포된 후 클러스터에서 시스템의 가상 머신을 배포합니다.

a. 템플릿 이름을 마우스 오른쪽 버튼으로 클릭하고 Clone → Clone to Virtual Machine을 클릭합니다.

b. Select a name and folder 탭에서 가상 머신의 이름을 지정합니다. control-plane-0 또는 compute-1과 같은 시스템 유형을 이름에 포함할 수 있습니다.

c. Select a name and folder 탭에서 클러스터에 대해 생성한 폴더의 이름을 선택합니다.

d. Select a compute resource 탭에서 데이터 센터의 호스트 이름을 선택합니다.

e. 선택사항: Select storage 탭에서 스토리지 옵션을 사용자 지정합니다.

f. Select clone options에서 Customize this virtual machine's hardware를 선택합니다.

g. Customize hardware 탭에서 VM Options → Advanced를 클릭합니다.

- 선택 사항: vSphere에서 기본 DHCP 네트워킹을 재정의합니다. 고정 IP 네트워킹을 활성화하려면 다음을 수행합니다.

  i. 고정 IP 구성을 설정합니다.

  ```
  $ export IPCFG="ip=<ip>::<gateway>:<netmask>:<hostname>:
  <iface>:none nameserver=srv1 [nameserver=srv2 [nameserver=srv3 [...]]]"
  ```

  명령 예

  ```
  $ export IPCFG="ip=192.168.100.101::192.168.100.254:255.255.255.0:::none
  nameserver=8.8.8.8"
  ```

  ii. guestinfo.afterburn.initrd.network-kargs 속성을 설정한 후 vSphere의 OVA에서 VM을 부팅합니다.

  ```
  $ govc vm.change -vm "<vm_name>" -e
  "guestinfo.afterburn.initrd.network-kargs=${IPCFG}"
  ```

- 선택사항: 클러스터 성능 문제가 발생하는 경우 Latency Sensitivity 목록에서 High를 선택합니다. VM의 CPU 및 메모리 예약에 다음과 같은 값이 있는지 확인합니다.

  - 메모리 예약 값은 구성된 메모리 크기와 같아야 합니다.

  - CPU 예약 값은 측정된 물리적 CPU 속도를 곱한 최소 대기 시간이 짧은 가상 CPU 수여야 합니다.

-

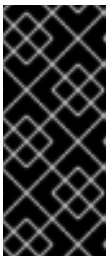
구성 편집을 클릭하고 **Configuration Parameters** 창에서 사용 가능한 매개 변수 목록을 검색하여 스틸 클럭 회계 **(stealclock.enable)**를 검색합니다. 사용 가능한 경우 해당 값을 **TRUE** 로 설정합니다. 스틸링 클럭 계정을 사용하면 클러스터 문제 해결에 도움이 될 수 있습니다.

- 구성 추가 매개변수를 클릭합니다. 다음 매개변수 이름 및 값을 정의합니다.
  - **guestinfo.ignition.config.data:** 이 절차에서 이전에 생성한 **base-64** 인코딩 파일을 찾고 이 머신 유형에 대해 **base64**로 인코딩된 **Ignition** 구성 파일의 내용을 붙여넣습니다.
  - **guestinfo.ignition.config.data.encoding: base64**를 지정합니다.
  - **disk.EnableUUID: TRUE**를 지정합니다.
  - **stealclock.enable:**이 매개 변수가 정의되지 않은 경우 추가하고 **TRUE** 를 지정합니다.

h. **Customize hardware** 탭의 **Virtual Hardware** 패널에서 지정된 값을 필요에 따라 수정합니다. **RAM, CPU** 및 디스크 스토리지의 양이 시스템 유형에 대한 최소 요구사항을 충족하는지 확인합니다.

i. 구성을 완료하고 **VM**의 전원을 켭니다.

10. 각 시스템에 대해 이전 단계에 따라 클러스터의 나머지 시스템을 생성합니다.



중요

이때 부트스트랩 및 컨트롤 플레인 시스템을 생성해야 합니다. 일부 **Pod**는 기본적으로 컴퓨팅 시스템에 배포되므로 클러스터를 설치하기 전에 컴퓨팅 시스템을 두 개 이상 생성합니다.

17.8.14. vSphere의 클러스터에 더 많은 컴퓨팅 머신 추가

VMware vSphere에서 사용자가 프로비저닝한 **OpenShift Container Platform** 클러스터에 더 많은 컴퓨팅 머신을 추가할 수 있습니다.

사전 요구 사항

- 컴퓨팅 머신의 base64로 인코딩된 Ignition 파일을 가져옵니다.
- 클러스터에 생성한 vSphere 템플릿에 액세스할 수 있습니다.

절차

1. 템플릿이 배포된 후 클러스터에서 시스템의 가상 머신을 배포합니다.
   a. 템플릿 이름을 마우스 오른쪽 버튼으로 클릭하고 Clone → Clone to Virtual Machine을 클릭합니다.
   b. Select a name and folder 탭에서 가상 머신의 이름을 지정합니다. compute-1과 같은 머신 유형을 이름에 포함할 수 있습니다.
   c. Select a name and folder 탭에서 클러스터에 대해 생성한 폴더의 이름을 선택합니다.
   d. Select a compute resource 탭에서 데이터 센터의 호스트 이름을 선택합니다.
   e. 선택사항: Select storage 탭에서 스토리지 옵션을 사용자 지정합니다.
   f. Select clone options에서 Customize this virtual machine's hardware를 선택합니다.
   g. Customize hardware 탭에서 VM Options → Advanced를 클릭합니다.
      - Latency Sensitivity 목록에서 High를 선택합니다.
      - Edit Configuration을 클릭하고 Configuration Parameters 창에서 Add Configuration Params를 클릭합니다. 다음 매개변수 이름 및 값을 정의합니다.

- guestinfo.ignition.config.data: 이 머신 유형에 대해 base64로 인코딩된 컴퓨팅 Ignition 구성 파일의 내용을 붙여넣습니다.
- guestinfo.ignition.config.data.encoding: base64를 지정합니다.
- disk.EnableUUID: TRUE를 지정합니다.

h. Customize hardware 탭의 Virtual Hardware 패널에서 지정된 값을 필요에 따라 수정합니다. RAM, CPU 및 디스크 스토리지의 양이 시스템 유형에 대한 최소 요구사항을 충족하는지 확인합니다. 또한 사용 가능한 네트워크가 여러 개인 경우 Add network adapter에서 올바른 네트워크를 선택해야 합니다.

i. 구성을 완료하고 VM의 전원을 켭니다.

2. 계속해서 클러스터에 추가 컴퓨팅 머신을 만듭니다.

17.8.15. 디스크 파티션 설정

대부분의 경우 데이터 파티션은 원래 다른 운영 체제를 설치하는 대신 RHCOS를 설치하여 생성됩니다. 이러한 경우 OpenShift Container Platform 설치 프로그램은 디스크 파티션을 설정할 수 있어야합니다.

그러나 OpenShift Container Platform 노드를 설치할 때 기본 파티션 설정을 덮어 쓰기하기 위해 개입이 필요한 두 가지 경우가 있습니다.

- 별도의 파티션 생성: 빈 디스크에 그린 필드 설치의 경우 파티션에 별도의 스토리지를 추가할 수 있습니다. 이는 /var 또는 /var의 하위 디렉터리 (예: /var/lib/etcd)중 하나를 별도의 파티션으로 만드는 경우에 공식적으로 지원됩니다.

중요

100GB 이상의 디스크 크기 및 특히 1TB보다 큰 디스크 크기의 경우 별도의 /var 파티션을 만듭니다. 자세한 내용은 " 별도의 /var 파티션 생성" 및 이 Red Hat 지식베이스 문서 를 참조하십시오.

중요

Kubernetes는 두 개의 파일 시스템 파티션만 지원합니다. 원래 구성에 둘 이상의 파티션을 추가하는 경우 **Kubernetes**는 모든 파티션을 모니터링할 수 없습니다.

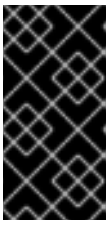
- 기존 파티션 유지: 브라운 필드 설치에서 기존 노드에 **OpenShift Container Platform**을 재설치하고 이전 운영 체제에 설치된 데이터 파티션을 유지해야 하는 경우 기존 데이터 파티션을 유지할 수 있는 **coreos-installer**에 부팅 인수와 옵션이 모두 있습니다.

별도의 /var 파티션 만들기

일반적으로 **OpenShift Container Platform**의 디스크 파티션 설정은 설치 프로그램에 맡겨야합니다. 그러나 확장하려는 파일 시스템의 일부에 별도의 파티션을 생성해야 하는 경우가 있습니다.

OpenShift 컨테이너 플랫폼은 **/var** 파티션 또는 **/var**의 하위 디렉터리 중 하나에 스토리지를 연결하는 단일 파티션의 추가를 지원합니다. 예를 들면 다음과 같습니다.

- **/var/lib/containers:** 시스템에 더 많은 이미지와 컨테이너가 추가됨에 따라 확장될 수 있는 컨테이너 관련 콘텐츠를 보관합니다.
- **/var/lib/etcd: etcd** 스토리지의 성능 최적화와 같은 목적으로 별도로 보관할 데이터를 보관합니다.
- **/var:** 감사 등의 목적에 맞게 별도로 분리하여 보관해야 하는 데이터를 보관합니다.



중요

100GB보다 큰 디스크 크기 및 특히 **1TB**보다 큰 디스크의 경우 별도의 **/var** 파티션을 만듭니다.

/var 디렉터리의 콘텐츠를 별도로 저장하면 필요에 따라 해당 영역에 대한 스토리지 확장을 보다 용이하게 하고 나중에 **OpenShift Container Platform**을 다시 설치하여 해당 데이터를 그대로 보존할 수 있습니다. 이 방법을 사용하면 모든 컨테이너를 다시 가져올 필요가 없으며 시스템을 업데이트할 때 대용량 로그 파일을 복사할 필요도 없습니다.

RHCOS(Red Hat Enterprise Linux CoreOS)를 새로 설치하기 전에 **/var**가 있어야 하므로 다음 절차

에서는 **OpenShift Container Platform** 설치의 **openshift-install** 준비 단계 중에 삽입되는 머신 구성 매니페스트를 생성하여 별도의 **/var** 파티션을 설정합니다.

절차

1. **OpenShift Container Platform** 설치 파일을 저장할 디렉터리를 만듭니다.

```
$ mkdir $HOME/clusterconfig
```

2. **openshift-install**을 실행하여 **manifest** 및 **openshift** 하위 디렉터리에 파일 세트를 만듭니다. 프롬프트가 표시되면 시스템 질문에 대답합니다.

```
$ openshift-install create manifests --dir $HOME/clusterconfig
? SSH Public Key ...
$ ls $HOME/clusterconfig/openshift/
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

3. 추가 파티션을 구성하는 **Butane** 구성을 생성합니다. 예를 들어 파일 이름을 **$HOME/clusterconfig/98-var-partition.bu** 로 지정하고, 디스크 장치 이름을 **worker** 시스템의 스토리지 장치 이름으로 변경하고 스토리지 크기를 적절하게 설정합니다. 이 예에서는 **/var** 디렉토리를 별도의 파티션에 배치합니다.

```
variant: openshift
version: 4.9.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1

2

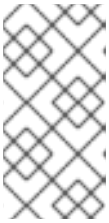
데이터 파티션을 부트 디스크에 추가할 때 최소 **25000** 메비 바이트가 권장됩니다. 루트 파일 시스템은 지정된 오프셋까지 사용 가능한 모든 공간을 채우기 위해 자동으로 크기가 조정됩니다. 값이 지정되지 않거나 지정된 값이 권장 최소값보다 작으면 생성되는 루트 파일 시스템의 크기가 너무 작아지고 **RHCOS**를 나중에 다시 설치할 때 데이터 파티션의 첫 번째 부분을 덮어 쓸 수 있습니다.

3

데이터 파티션의 크기(**MB**)입니다.

4

컨테이너 스토리지에 사용되는 파일 시스템에 **prjquota** 마운트 옵션을 활성화해야 합니다.



참고

별도의 **/var** 파티션을 만들 때 다른 인스턴스 유형에 동일한 장치 이름이 없는 경우 작업자 노드에 다른 인스턴스 유형을 사용할 수 없습니다.

4. **Butane** 구성에서 매니페스트를 생성하여 **clusterconfig/openshift** 디렉터리에 저장합니다. 예를 들어 다음 명령을 실행합니다.

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o
$HOME/clusterconfig/openshift/98-var-partition.yaml
```

5. **openshift-install**을 다시 실행하여 **manifest** 및 **openshift** 하위 디렉터리의 파일 세트에서 **Ignition** 구성을 만듭니다.

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

이제 **Ignition** 구성 파일을 **vSphere** 설치 절차에 대한 입력으로 사용하여 **RHCOS (Red Hat Enterprise Linux CoreOS)** 시스템을 설치할 수 있습니다.

17.8.16. bootupd를 사용하여 부트로더 업데이트

bootupd를 사용하여 부트로더를 업데이트하려면 **RHCOS** 머신에 수동으로 **bootupd**를 설치하거나 **systemd** 유닛이 활성화된 머신 구성을 제공해야 합니다. **grubby** 또는 기타 부트로더 툴과는 달리 **bootupd**는 커널 인수 전달과 같은 커널 공간 구성을 관리하지 않습니다.

bootupd를 설치한 후 **OpenShift Container Platform** 클러스터에서 원격으로 관리할 수 있습니다.

참고

bootupd는 **BootHole** 취약점에 대한 보호와 같이 베어 메탈 또는 가상화 하이퍼바이저 설치에서만 사용하는 것이 좋습니다.

수동 설치 방법

bootctl 명령줄 툴을 사용하여 **bootupd**를 수동으로 설치할 수 있습니다.

1. 시스템 상태를 검사합니다.

```
# bootupctl status
```

출력 예

```
Component EFI
  Installed: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
  Update: At latest version
```

2. **bootupd**를 설치하지 않고 생성된 **RHCOS** 이미지에는 명시적 채택 단계가 필요합니다.

   시스템 상태가 **Adoptable**인 경우 채택을 수행합니다.

```
# bootupctl adopt-and-update
```

출력 예

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

3. 업데이트를 사용할 수 있는 경우 다음 재부팅에 변경 사항이 적용되도록 업데이트를 적용합니다.

```
# bootupctl update
```

출력 예

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

시스템 구성 방법

bootupd를 활성화하는 또 다른 방법은 머신 구성을 제공하는 것입니다.

- 다음 예와 같이 **systemd** 단위가 활성화된 머신 구성 파일을 제공합니다.

출력 예

```
variant: rhcos
version: 1.1.0
systemd:
  units:
    - name: custom-bootupd-auto.service
      enabled: true
      contents: |
        [Unit]
        Description=Bootupd automatic update

        [Service]
        ExecStart=/usr/bin/bootupctl update
        RemainAfterExit=yes

        [Install]
        WantedBy=multi-user.target
```

17.8.17. 부트스트랩 프로세스가 완료될 때까지 대기 중

OpenShift Container Platform 부트스트랩 프로세스는 클러스터 노드가 먼저 디스크에 설치된 영구 RHCOS 환경으로 부팅된 후에 시작됩니다. Ignition 구성 파일을 통해 제공되는 구성 정보는 부트스트랩 프로세스를 초기화하고 머신에 OpenShift Container Platform을 설치하는 데 사용됩니다. 부트스트랩 프로세스가 완료될 때까지 기다려야 합니다.

사전 요구 사항

- 클러스터에 대한 Ignition 구성 파일이 생성되어 있습니다.
- 적합한 네트워크, DNS 및 로드 밸런싱 인프라가 구성되어 있습니다.
- 설치 프로그램을 받아서 클러스터의 Ignition 구성 파일을 생성했습니다.
- 클러스터 머신에 RHCOS를 설치하고 OpenShift Container Platform 설치 프로그램에서 생성된 Ignition 구성 파일을 제공했습니다.

프로세스

1. 부트스트랩 프로세스를 모니터링합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2 다른 설치 세부 사항을 보려면 info 대신 warn, debug 또는 error를 지정합니다.

출력 예

```
INFO Waiting up to 30m0s for the Kubernetes API at
https://api.test.example.com:6443...
INFO API v1.22.1 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

이 명령은 **Kubernetes API** 서버가 컨트롤 플레인 시스템에서 부트스트랩되었다는 신호를 보낼 때 성공합니다.

2. 부트스트랩 프로세스가 완료된 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다.

중요

이 시점에 로드 밸런서에서 부트스트랩 시스템을 제거해야 합니다. 부트스트랩 머신 자체를 제거하거나 다시 포맷할 수도 있습니다.

17.8.18. CLI를 사용하여 클러스터에 로그인

클러스터 **kubeconfig** 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. **kubeconfig** 파일에는 **CLI**에서 올바른 클러스터 및 **API** 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 **OpenShift Container Platform** 설치 과정에서 생성됩니다.

사전 요구 사항

- **OpenShift Container Platform** 클러스터를 배포했습니다.
- **oc CLI**를 설치했습니다.

프로세스

1. **kubeadmin** 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 oc 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

17.8.19. 머신의 인증서 서명 요청 승인

클러스터에 시스템을 추가하면 추가한 시스템별로 보류 중인 인증서 서명 요청(CSR)이 두 개씩 생성됩니다. 이러한 CSR이 승인되었는지 확인해야 하며, 필요한 경우 이를 직접 승인해야 합니다. 클라이언트 요청을 먼저 승인한 다음 서버 요청을 승인해야 합니다.

사전 요구 사항

- 클러스터에 시스템을 추가했습니다.

절차

1. 클러스터가 시스템을 인식하는지 확인합니다.

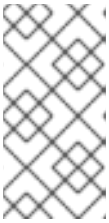
```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0 Ready    master  63m  v1.22.1
```

```
master-1  Ready    master  63m  v1.22.1
master-2  Ready    master  64m  v1.22.1
```

출력에 생성된 모든 시스템이 나열됩니다.



참고

이전 출력에는 일부 **CSR**이 승인될 때까지 컴퓨팅 노드(작업자 노드라고도 함)가 포함되지 않을 수 있습니다.

2. 보류 중인 **CSR**을 검토하고 클러스터에 추가한 각 시스템에 대해 **Pending** 또는 **Approved** 상태의 클라이언트 및 서버 요청이 표시되는지 확인합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE    REQUESTOR                                                   CONDITION
csr-8b2br   15m    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

예에서는 두 시스템이 클러스터에 참여하고 있습니다. 목록에는 승인된 **CSR**이 더 많이 나타날 수도 있습니다.

3. **CSR**이 승인되지 않은 경우, 추가된 시스템에 대한 모든 보류 중인 **CSR**이 **Pending** 상태로 전환된 후 클러스터 시스템의 **CSR**을 승인합니다.

참고

CSR은 교체 주기가 자동으로 만료되므로 클러스터에 시스템을 추가한 후 1 시간 이내에 CSR을 승인하십시오. 한 시간 내에 승인하지 않으면 인증서가 교체되고 각 노드에 대해 두 개 이상의 인증서가 표시됩니다. 이러한 인증서를 모두 승인해야 합니다. 클라이언트 CSR이 승인되면 Kubelet은 인증서에 대한 보조 CSR을 생성하므로 수동 승인이 필요합니다. 그러면 Kubelet에서 동일한 매개변수를 사용하여 새 인증서를 요청하는 경우 인증서 갱신 요청은 machine-approver에 의해 자동으로 승인됩니다.

참고

베어 메탈 및 기타 사용자 프로비저닝 인프라와 같이 머신 API를 사용하도록 활성화되지 않는 플랫폼에서 실행되는 클러스터의 경우 CSR(Kubelet service Certificate Request)을 자동으로 승인하는 방법을 구현해야 합니다. 요청이 승인되지 않으면 API 서버가 kubelet에 연결될 때 서비스 인증서가 필요하므로 oc exec, oc rsh, oc logs 명령을 성공적으로 수행할 수 없습니다. Kubelet 엔드 포인트에 연결하는 모든 작업을 수행하려면 이 인증서 승인이 필요합니다. 이 방법은 새 CSR을 감시하고 CSR이 system:node 또는 system:admin 그룹의 node-bootstrapper 서비스 계정에 의해 제출되었는지 확인하고 노드의 ID를 확인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 <csr_name>은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

참고

일부 Operator는 일부 CSR이 승인될 때까지 사용할 수 없습니다.

4. 이제 클라이언트 요청이 승인되었으므로 클러스터에 추가한 각 머신의 서버 요청을 검토해야 합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal        Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal        Pending
...
```

5. 나머지 CSR이 승인되지 않고 Pending 상태인 경우 클러스터 머신의 CSR을 승인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 <csr_name>은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs oc adm certificate approve
```

6. 모든 클라이언트 및 서버 CSR이 승인된 후 머신은 Ready 상태가 됩니다. 다음 명령을 실행하여 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0  Ready    master  73m  v1.22.1
master-1  Ready    master  73m  v1.22.1
master-2  Ready    master  74m  v1.22.1
worker-0  Ready    worker  11m  v1.22.1
worker-1  Ready    worker  11m  v1.22.1
```

참고

머신이 **Ready** 상태로 전환하는 데 서버 **CSR**의 승인 후 몇 분이 걸릴 수 있습니다.

추가 정보

- **CSR**에 대한 자세한 내용은 인증서 서명 요청을 참조하십시오.

17.8.20. Operator의 초기 설정

컨트롤 플레인이 초기화된 후 일부 **Operator**를 즉시 구성하여 모두 사용 가능하도록 해야 합니다.

사전 요구 사항

- 컨트롤 플레인이 초기화되어 있습니다.

절차

1. 클러스터 구성 요소가 온라인 상태인지 확인합니다.

```
$ watch -n5 oc get clusteroperators
```

출력 예

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.9.0     True        False         False     19m
baremetal                                  4.9.0     True        False         False     37m
cloud-credential                           4.9.0     True        False         False     40m
```

```
cluster-autoscaler                         4.9.0     True        False         False      37m
config-operator                            4.9.0     True        False         False      38m
console                                    4.9.0     True        False         False      26m
csi-snapshot-controller                    4.9.0     True        False         False      37m
dns                                        4.9.0     True        False         False      37m
etcd                                       4.9.0     True        False         False      36m
image-registry                             4.9.0     True        False         False      31m
ingress                                    4.9.0     True        False         False      30m
insights                                   4.9.0     True        False         False      31m
kube-apiserver                             4.9.0     True        False         False      26m
kube-controller-manager                    4.9.0     True        False         False      36m
kube-scheduler                             4.9.0     True        False         False      36m
kube-storage-version-migrator              4.9.0     True        False         False      37m
machine-api                                4.9.0     True        False         False      29m
machine-approver                           4.9.0     True        False         False      37m
machine-config                             4.9.0     True        False         False      36m
marketplace                                4.9.0     True        False         False      37m
monitoring                                 4.9.0     True        False         False      29m
network                                    4.9.0     True        False         False      38m
node-tuning                                4.9.0     True        False         False      37m
openshift-apiserver                        4.9.0     True        False         False      32m
openshift-controller-manager               4.9.0     True        False         False      30m
openshift-samples                          4.9.0     True        False         False      32m
operator-lifecycle-manager                 4.9.0     True        False         False      37m
operator-lifecycle-manager-catalog         4.9.0     True        False         False      37m
operator-lifecycle-manager-packageserver   4.9.0     True        False         False      32m
service-ca                                 4.9.0     True        False         False      38m
storage                                    4.9.0     True        False         False      37m
```

2. 사용할 수 없는 **Operator**를 구성합니다.

17.8.20.1. 기본 OperatorHub 소스 비활성화

Red Hat 및 커뮤니티 프로젝트에서 제공하는 콘텐츠를 소싱하는 **Operator** 카탈로그는 **OpenShift Container Platform**을 설치하는 동안 기본적으로 **OperatorHub**용으로 구성됩니다. 제한된 네트워크 환경에서는 클러스터 관리자로서 기본 카탈로그를 비활성화해야 합니다.

절차

- **OperatorHub** 오브젝트에 **disableAllDefaultSources: true**를 추가하여 기본 카탈로그의 소스를 비활성화합니다.

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

작은 정보

또는 웹 콘솔을 사용하여 카탈로그 소스를 관리할 수 있습니다. 관리 → 클러스터 설정 → 구성 → **OperatorHub** 페이지에서 개별 소스를 생성, 삭제, 비활성화 및 활성화할 수 있는 소스 탭을 클릭합니다.

17.8.20.2. 이미지 레지스트리 스토리지 구성

기본 스토리지를 제공하지 않는 플랫폼에서는 처음에 **Image Registry Operator**를 사용할 수 없습니다. 설치한 후에 스토리지를 사용하도록 레지스트리를 구성하여 **Registry Operator**를 사용 가능하도록 만들어야 합니다.

프로덕션 클러스터에 필요한 영구 볼륨을 구성하는 과정의 지침이 표시됩니다. 해당하는 경우, 프로덕션 환경 외 클러스터에서만 사용할 수 있는 저장 위치로서 빈 디렉터리를 구성하는 과정의 지침이 표시됩니다.

업그레이드 중에 **Recreate** 롤아웃 전략을 사용하여 이미지 레지스트리의 블록 스토리지 유형 사용을 허용하기 위한 추가 지침이 제공됩니다.

17.8.20.2.1. VMware vSphere용 레지스트리 스토리지 구성

클러스터 관리자는 설치한 후 스토리지를 사용하도록 레지스트리를 구성해야 합니다.

사전 요구 사항

- 클러스터 관리자 권한이 있어야 합니다.
- **VMware vSphere**에 클러스터가 있어야 합니다.
- **Red Hat OpenShift Container Storage**와 같이 클러스터용 영구 스토리지 프로비저닝.

중요

OpenShift Container Platform은 복제본이 하나만 있는 경우 이미지 레지스트리 스토리지에 대한 **ReadWriteOnce** 액세스를 지원합니다. **ReadWriteOnce** 액세스에서는 레지스트리가 **Recreate** 롤아웃 전략을 사용해야 합니다. 두 개 이상의 복제본으로 고 가용성을 지원하는 이미지 레지스트리를 배포하려면 **ReadWriteMany** 액세스가 필요합니다.

- "100Gi" 용량이 필요합니다.

중요

테스트 결과, RHEL의 NFS 서버를 핵심 서비스용 스토리지 백엔드로 사용하는 데 문제가 있는 것으로 나타납니다. 여기에는 OpenShift Container Registry and Quay, 스토리지 모니터링을 위한 Prometheus, 로깅 스토리지를 위한 Elasticsearch가 포함됩니다. 따라서 RHEL NFS를 사용하여 핵심 서비스에서 사용하는 PV를 백업하는 것은 권장되지 않습니다.

마켓플레이스의 다른 NFS 구현에는 이러한 문제가 나타나지 않을 수 있습니다. 이러한 OpenShift Container Platform 핵심 구성 요소에 대해 완료된 테스트에 대한 자세한 내용은 개별 NFS 구현 공급업체에 문의하십시오.

프로세스

1. 스토리지를 사용하도록 레지스트리를 구성하기 위해 configs.imageregistry/cluster 리소스에서 spec.storage.pvc를 변경합니다.

참고

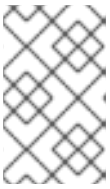
공유 스토리지를 사용할 때 보안 설정을 확인하여 외부에서의 액세스를 방지합니다.

2. 레지스트리 pod가 없는지 확인합니다.

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

출력 예

```
No resourses found in openshift-image-registry namespace
```



참고

출력에 레지스트리 Pod가 있는 경우 이 절차를 계속할 필요가 없습니다.

3. 레지스트리 구성을 확인합니다.

```
$ oc edit configs.imageregistry.operator.openshift.io
```

출력 예

```
storage:
  pvc:
    claim: 1
```

1 image-registry-storage 영구 볼륨 클레임 (PVC)을 자동으로 생성할 수 있도록 claim 필드를 비워 둡니다. PVC는 기본 스토리지 클래스를 기반으로 생성됩니다. 그러나 기본 스토리지 클래스에서 RADOS 블록 장치(RBD)와 같은 ReadWriteOnce(ReadWriteOnce) 볼륨을 제공할 수 있으므로 복제본이 두 개 이상될 때 문제가 발생할 수 있습니다.

4. clusteroperator 상태를 확인합니다.

```
$ oc get clusteroperator image-registry
```

출력 예

```
NAME             VERSION                             AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                                 True        False         False     6h50m
```

17.8.20.2.2. 프로덕션 환경 외 클러스터에서 이미지 레지스트리의 스토리지 구성

이미지 레지스트리 Operator에 대한 스토리지를 구성해야 합니다. 프로덕션 환경 외 클러스터의 경우, 이미지 레지스트리를 빈 디렉터리로 설정할 수 있습니다. 이렇게 하는 경우 레지스트리를 다시 시작하면 모든 이미지가 손실됩니다.

프로세스

- 이미지 레지스트리 스토리지를 빈 디렉터리로 설정하려면 다음을 수행하십시오.

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch
'{"spec":{"storage":{"emptyDir":{}}}}'
```

주의

프로덕션 환경 외 클러스터에 대해서만 이 옵션을 구성하십시오.

Image Registry Operator가 구성 요소를 초기화하기 전에 이 명령을 실행하면 oc patch 명령이 실패하며 다음 오류가 발생합니다.

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster"
not found
```

몇 분 후에 명령을 다시 실행하십시오.

17.8.20.2.3. VMware vSphere용 블록 레지스트리 스토리지 구성

클러스터 관리자로서 업그레이드 중에 이미지 레지스트리가 vSphere VMDK(Virtual Machine Disk)와 같은 블록 스토리지 유형을 사용할 수 있도록 허용하기 위해 Recreate 롤아웃 전략을 사용할 수 있습니다.

중요

블록 스토리지 볼륨이 지원되지만 프로덕션 클러스터에서 이미지 레지스트리와 함께 사용하는 것은 권장되지 않습니다. 레지스트리가 블록 스토리지에 구성된 설치는 레지스트리가 둘 이상의 복제본을 가질 수 없기 때문에 가용성이 높지 않습니다.

프로세스

1. 이미지 레지스트리 스토리지를 블록 스토리지 유형으로 설정하려면 **Recreate** 롤아웃 전략을 사용하고 복제본 **1**개 만으로 실행되도록 레지스트리를 패치합니다.

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. 블록 스토리지 장치에 **PV**를 프로비저닝하고 해당 볼륨의 **PVC**를 생성합니다. 요청된 블록 볼륨은 **RWO(ReadWriteOnce)** 액세스 모드를 사용합니다.

   a. **VMware vSphere PersistentVolumeClaim** 개체를 정의하려면 다음 내용이 포함된 **pvc.yaml** 파일을 생성합니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage 1
  namespace: openshift-image-registry 2
spec:
  accessModes:
  - ReadWriteOnce 3
  resources:
    requests:
      storage: 100Gi 4
```

1. **PersistentVolumeClaim** 개체를 표시하는 고유한 이름입니다.

2. **PersistentVolumeClaim** 오브젝트의 네임스페이스로 **openshift-image-registry**입니다.

3. 영구 볼륨 클레임의 액세스 모드입니다. **ReadWriteOnce**를 사용하면 단일 노드에서 읽기 및 쓰기 권한으로 볼륨을 마운트할 수 있습니다.

4

영구 볼륨 클레임의 크기입니다.

b. 파일에서 **PersistentVolumeClaim** 오브젝트를 만듭니다.

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 올바른 **PVC**를 참조하도록 레지스트리 설정을 편집합니다.

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

출력 예

```
storage:
  pvc:
    claim: 1
```

1

사용자 지정 **PVC**를 만들면 **image-registry-storage PVC**의 기본 자동 생성을 위해 **claim** 필드를 비워둘 수 있습니다.

올바른 **PVC**를 참조하도록 레지스트리 스토리지를 구성하는 방법은 **vSphere** 용 레지스트리 구성을 참조하십시오.

17.8.21. 사용자 프로비저닝 인프라에 설치 완료

Operator 구성을 완료한 후 제공하는 인프라에 클러스터 설치를 완료할 수 있습니다.

사전 요구 사항

- 컨트롤 플레인이 초기화되어 있습니다.
- 초기 Operator 구성을 완료해야 합니다.

프로세스

1. 다음 명령을 사용하여 모든 클러스터 구성 요소가 온라인 상태인지 확인합니다.

```
$ watch -n5 oc get clusteroperators
```

출력 예

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.9.0     True        False         False      19m
baremetal                                  4.9.0     True        False         False      37m
cloud-credential                           4.9.0     True        False         False      40m
cluster-autoscaler                         4.9.0     True        False         False      37m
config-operator                            4.9.0     True        False         False      38m
console                                    4.9.0     True        False         False      26m
csi-snapshot-controller                    4.9.0     True        False         False      37m
dns                                        4.9.0     True        False         False      37m
etcd                                       4.9.0     True        False         False      36m
image-registry                             4.9.0     True        False         False      31m
ingress                                    4.9.0     True        False         False      30m
insights                                   4.9.0     True        False         False      31m
kube-apiserver                             4.9.0     True        False         False      26m
kube-controller-manager                    4.9.0     True        False         False      36m
kube-scheduler                             4.9.0     True        False         False      36m
kube-storage-version-migrator              4.9.0     True        False         False      37m
machine-api                                4.9.0     True        False         False      29m
machine-approver                           4.9.0     True        False         False      37m
machine-config                             4.9.0     True        False         False      36m
marketplace                                4.9.0     True        False         False      37m
monitoring                                 4.9.0     True        False         False      29m
network                                    4.9.0     True        False         False      38m
node-tuning                                4.9.0     True        False         False      37m
openshift-apiserver                        4.9.0     True        False         False      32m
openshift-controller-manager               4.9.0     True        False         False      30m
openshift-samples                          4.9.0     True        False         False      32m
operator-lifecycle-manager                 4.9.0     True        False         False      37m
operator-lifecycle-manager-catalog         4.9.0     True        False         False      37m
operator-lifecycle-manager-packageserver   4.9.0     True        False         False      32m
service-ca                                 4.9.0     True        False         False      38m
storage                                    4.9.0     True        False         False      37m
```

또는 다음 명령은 모든 클러스터를 사용할 수 있을 때 알립니다. 또한 인증 정보를 검색하고 표시합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1

<installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

출력 예

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator가 Kubernetes API 서버에서 OpenShift Container Platform 클러스터 배포를 완료하면 명령이 성공합니다.

중요

- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 node-bootstrapper 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

2. Kubernetes API 서버가 Pod와 통신하고 있는지 확인합니다.

a.

모든 **Pod** 목록을 보려면 다음 명령을 사용하십시오.

```
$ oc get pods --all-namespaces
```

출력 예

```
NAMESPACE                      NAME                                        READY   STATUS
RESTARTS   AGE
openshift-apiserver-operator   openshift-apiserver-operator-85cb746d55-zqhs8
1/1     Running   1          9m
openshift-apiserver            apiserver-67b9g                             1/1     Running   0
3m
openshift-apiserver            apiserver-ljcmx                             1/1     Running   0
1m
openshift-apiserver            apiserver-z25h4                             1/1     Running   0
2m
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8
1/1     Running   0          5m
...
```

b.

다음 명령을 사용하여 이전 명령의 출력에 나열된 **Pod**의 로그를 표시합니다.

```
$ oc logs <pod_name> -n <namespace> 1
```

1

이전 명령의 출력에 표시된 대로 **Pod** 이름과 네임스페이스를 지정합니다.

Pod 로그가 표시되면 **Kubernetes API** 서버는 클러스터 시스템과 통신할 수 있습니다.

3.

FCP(Fibre Channel Protocol)를 사용하는 설치에는 다중 경로를 활성화하기 위해 추가 단계가 필요합니다. 설치 중에 멀티패스를 활성화하지 마십시오.

자세한 내용은 *설치 후 머신 구성 작업* 설명서에서 **"RHCOS**에서 커널 인수를 사용하여 멀티패스 활성화**"**를 참조하십시오.

4. 클러스터 등록 페이지에서 클러스터를 등록합니다.

vSphere에 컴퓨팅 머신 추가에 따라 클러스터 설치가 완료된 후 추가 컴퓨팅 머신을 추가할 수 있습니다.

17.8.22. VMware vSphere 볼륨 백업

OpenShift Container Platform은 새 볼륨을 독립 영구 디스크로 프로비저닝하여 클러스터의 모든 노드에서 볼륨을 자유롭게 연결 및 분리합니다. 결과적으로 스냅샷을 사용하는 볼륨을 백업하거나 스냅샷에서 볼륨을 복원할 수 없습니다. 자세한 내용은 스냅 샷 제한 을 참조하십시오.

프로세스

영구 볼륨의 백업을 생성하려면 다음을 수행합니다.

1. 영구 볼륨을 사용중인 애플리케이션을 중지합니다.
2. 영구 볼륨을 복제합니다.
3. 애플리케이션을 다시 시작합니다.
4. 복제된 볼륨의 백업을 만듭니다.
5. 복제된 볼륨을 삭제합니다.

17.8.23. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription

watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

17.8.24. 다음 단계

- 클러스터를 사용자 지정합니다.
- 클러스터를 설치하는 데 사용한 미러 레지스트리에 신뢰할 수 있는 CA가 있는 경우 추가 신뢰 저장소를 구성하여 클러스터에 추가합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.
- 선택 사항: vSphere Problem Detector Operator에서 이벤트를 보고 클러스터에 권한 또는 스토리지 구성 문제가 있는지 확인합니다.

17.9. VSPHERE에서 설치 관리자 프로비저닝 인프라를 사용하는 클러스터 설치 제거

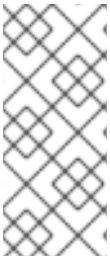
설치 관리자 프로비저닝 인프라를 사용하여 VMware vSphere 인스턴스에 배포한 클러스터를 제거할 수 있습니다.

참고

openshift-install destroy cluster 명령을 실행하여 OpenShift Container Platform을 제거하면 vSphere 볼륨이 자동으로 삭제되지 않습니다. 클러스터 관리자는 vSphere 볼륨을 수동으로 찾아서 삭제해야 합니다.

17.9.1. 설치 관리자가 프로비저닝한 인프라를 사용하는 클러스터 제거

클라우드에서 설치 관리자 프로비저닝 인프라를 사용하는 클러스터를 제거할 수 있습니다.



참고

설치 제거 후 특히 **UPI(User Provisioned Infrastructure)** 클러스터에서 제거되지 않은 리소스에 대해 클라우드 공급자를 확인합니다. 설치 관리자가 생성하지 않았거나 설치 프로그램이 액세스할 수 없는 리소스가 있을 수 있습니다.

사전 요구 사항

- 클러스터를 배포하는 데 사용한 설치 프로그램의 사본을 준비합니다.
- 클러스터를 만들 때 설치 프로그램에 의해 생성된 파일을 준비합니다.

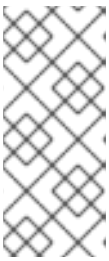
프로세스

1. 클러스터를 설치하는 데 사용한 컴퓨터에 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행합니다.

```
$ ./openshift-install destroy cluster \
--dir <installation_directory> --log-level info 1 2
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2 다른 세부 사항을 보려면 **info** 대신 **warn, debug** 또는 **error**를 지정합니다.



참고

클러스터의 클러스터 정의 파일이 포함되어 있는 디렉터리를 지정해야 합니다. 설치 프로그램이 클러스터를 삭제하려면 이 디렉터리에 있는 **metadata.json** 파일이 필요합니다.

2. 선택사항: **<installation_directory>** 디렉터리와 **OpenShift Container Platform** 설치 프로그램을 삭제합니다.

17.10. VSPHERE PROBLEM DETECTOR OPERATOR 사용

17.10.1. vSphere Problem Detector Operator 정보

vSphere Problem Detector Operator는 스토리지와 관련된 일반적인 설치 및 잘못된 구성 문제를 위해 vSphere에 배포된 클러스터를 확인합니다.

Operator는 openshift-cluster-storage-operator 네임 스페이스에서 실행되며 Cluster Storage Operator가 vSphere에 클러스터를 배포하는 것을 탐지하면 Cluster Storage Operator에 의해 시작됩니다. vSphere Problem Detector Operator는 vSphere vCenter Server와 통신하여 클러스터의 가상 머신, 기본 데이터 저장소 및 vSphere vCenter 서버 구성에 대한 기타 정보를 확인합니다. Operator는 Cloud Credential Operator의 인증 정보를 사용하여 vSphere에 연결합니다.

Operator는 다음 일정에 따라 검사를 실행합니다.

- 검사는 8시간마다 실행됩니다.
- 검사에 실패하면 Operator는 1분, 2분, 4분, 8분 등의 간격으로 다시 검사를 실행합니다. Operator는 간격을 최대 8시간까지 두 배로 늘립니다.
- 모든 검사가 통과되면 일정이 8시간 간격으로 돌아갑니다.

Operator는 오류 후 검사 빈도를 늘리므로 실패 조건이 수정된 후 Operator에서 빠르게 성공적으로 완료를 보고할 수 있습니다. 즉시 문제 해결 정보를 위해 Operator를 수동으로 실행할 수 있습니다.

17.10.2. vSphere Problem Detector Operator 검사 실행

vSphere Problem Detector Operator 검사를 실행하기 위한 일정을 재정의하고 즉시 검사를 실행할 수 있습니다.

vSphere Problem Detector Operator는 8시간마다 검사를 자동으로 실행합니다. 그러나 Operator가 시작되면 즉시 검사를 실행합니다. Cluster Storage Operator가 시작될 때 Operator는 Cluster Storage Operator에 의해 시작하여 vSphere에서 클러스터가 실행 중인지 확인합니다. 검사를 즉시 실행하려면 vSphere Problem Detector Operator를 0 으로 스케일링하고 1로 돌아가 vSphere Problem Detector Operator를 다시 시작할 수 있습니다.

사전 요구 사항

- cluster-admin 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

프로세스

1. Operator를 0으로 스케일링합니다.

```
$ oc scale deployment/vsphere-problem-detector-operator --replicas=0 \
   -n openshift-cluster-storage-operator
```

배포가 즉시 0으로 스케일링되지 않으면 다음 명령을 실행하여 Pod가 종료될 때까지 기다릴 수 있습니다.

```
$ oc wait pods -l name=vsphere-problem-detector-operator \
   --for=delete --timeout=5m -n openshift-cluster-storage-operator
```

2. Operator를 1로 다시 스케일링합니다.

```
$ oc scale deployment/vsphere-problem-detector-operator --replicas=1 \
   -n openshift-cluster-storage-operator
```

3. 이전 리더 잠금을 삭제하여 Cluster Storage Operator의 새 리더 선택을 가속화합니다.

```
$ oc delete -n openshift-cluster-storage-operator \
   cm vsphere-problem-detector-lock
```

검증

- vSphere Problem Detector Operator에서 생성된 이벤트 또는 로그를 표시합니다. 이벤트 또는 로그에 최근 타임 스탬프가 있는지 확인합니다.

17.10.3. vSphere Problem Detector Operator에서 이벤트 보기

vSphere Problem Detector Operator가 실행되고 구성 검사를 수행한 후 명령줄 또는 OpenShift Container Platform 웹 콘솔에서 볼 수 있는 이벤트를 생성합니다.

프로세스

- 명령줄을 사용하여 이벤트를 보려면 다음 명령을 실행합니다.

```
$ oc get event -n openshift-cluster-storage-operator \
  --sort-by={.metadata.creationTimestamp}
```

출력 예

```
16m     Normal   Started         pod/vsphere-problem-detector-operator-xxxxx
Started container vsphere-problem-detector
16m     Normal   Created         pod/vsphere-problem-detector-operator-xxxxx
Created container vsphere-problem-detector
16m     Normal   LeaderElection  configmap/vsphere-problem-detector-lock
vsphere-problem-detector-operator-xxxxx became leader
```

- OpenShift Container Platform 웹 콘솔을 사용하여 이벤트를 보려면 다음 프로젝트 메뉴에서 홈 → 이벤트로 이동하여 **openshift-cluster-storage-operator**를 선택합니다.

17.10.4. vSphere Problem Detector Operator에서 로그 보기

vSphere Problem Detector Operator가 실행되고 구성 검사를 수행한 후 명령줄 또는 OpenShift Container Platform 웹 콘솔에서 볼 수 있는 로그 레코드를 생성합니다.

프로세스

- 명령줄을 사용하여 로그를 보려면 다음 명령을 실행합니다.

```
$ oc logs deployment/vsphere-problem-detector-operator \
  -n openshift-cluster-storage-operator
```

출력 예

```
I0108 08:32:28.445696       1 operator.go:209] ClusterInfo passed
I0108 08:32:28.451029       1 datastore.go:57] CheckStorageClasses checked 1 storage
classes, 0 problems found
I0108 08:32:28.451047       1 operator.go:209] CheckStorageClasses passed
I0108 08:32:28.452160       1 operator.go:209] CheckDefaultDatastore passed
I0108 08:32:28.480648       1 operator.go:271] CheckNodeDiskUUID:<host_name>
```

```
passed
I0108 08:32:28.480685       1 operator.go:271] CheckNodeProviderID:<host_name>
passed
```

- **OpenShift Container Platform 웹 콘솔을 사용하여 Operator 로그를 보려면 다음 단계를 수행합니다.**

  a. **워크로드 → 포드로 이동합니다.**

  b. **프로젝트 메뉴에서 openshift-cluster-storage-operator를 선택합니다**

  c. **vsphere-problem-detector-operator 포드에 대한 링크를 클릭합니다.**

  d. **포드 세부 정보 페이지에서 로그 탭을 클릭하여 로그를 확인합니다.**

17.10.5. vSphere Problem Detector Operator에서 실행한 설정 검사

다음 표에서는 vSphere Problem Detector Operator가 실행하는 설정 검사를 식별합니다. 일부 검사에서는 클러스터 설정을 확인합니다. 다른 검사에서는 클러스터에서 각 노드의 설정을 확인합니다.

표 17.84. 클러스터 설정 검사

이름	설명
CheckDefaultDatastore	vSphere 설정의 기본 데이터 저장소 이름이 동적 프로비저닝에 사용할 수 있을 만큼 짧은지 확인합니다. 이 검사에 실패하는 경우 다음을 예상할 수 있습니다. ● **마운트 단위 설정 실패: 부적절한 인수**와 같은 저널에 대한 **systemd** 로그 오류. ● 가상 머신이 종료되거나 재부팅되면 노드에서 모든 포드를 드레이닝하지 않고도 **systemd**가 볼륨을 마운트 해제하지 않습니다. 이 검사에 실패하면 기본 데이터 저장소에 대해 이름이 더 짧은 vSphere를 재구성하십시오.

이름	설명
CheckFolderPermissions	기본 데이터 저장소에서 볼륨을 나열할 수 있는 권한을 확인합니다. 이 권한은 볼륨을 생성하는 데 필요합니다. Operator는 / 및 **/kubevols** 디렉터리를 나열하여 권한을 확인합니다. root 디렉터리가 있어야 합니다. 검사가 실행될 때 **/kubevols** 디렉터리가 존재하지 않는 경우 허용됩니다. 디렉터리가 아직 존재하지 않는 경우 데이터 저장소가 동적 프로비저닝과 함께 사용되면 **/kubevols** 디렉터리가 생성됩니다. 이 검사가 실패하면 OpenShift Container Platform 설치 중에 지정된 vCenter 계정에 필요한 권한을 검토하십시오.
CheckStorageClasses	다음을 확인합니다. ● 이 스토리지 클래스에서 프로비저닝하는 각 영구 볼륨에 대한 정규화된 경로는 255자 미만입니다. ● 스토리지 클래스가 스토리지 정책을 사용하는 경우 스토리지 클래스는 하나의 정책만 사용해야 하며 해당 정책을 정의해야 합니다.
CheckTaskPermissions	최근 작업 및 데이터 저장소를 나열할 수 있는 권한을 확인합니다.
ClusterInfo	vSphere vCenter에서 클러스터 버전 및 UUID를 수집합니다.

표 17.85. 노드 설정 검사

이름	설명
CheckNodeDiskUUID	모든 vSphere 가상 머신이 **disk.enableUUID=TRUE**로 구성되어 있는지 확인합니다. 이 검사에 실패하는 경우 vSphere Red Hat 지식베이스 정보 솔루션의 VM에서 'disk.EnableUUID' 매개변수를 확인하는 방법을 참조하십시오.
CheckNodeProviderID	모든 노드가 vSphere vCenter의 **ProviderID**로 구성되었는지 확인합니다. 다음 명령의 출력에 각 노드의 공급자 ID가 포함되지 않으면 이 검사가 실패합니다. `$ oc get nodes -o custom-columns=NAME:.metadata.name,PROVIDER_ID:.spec.providerID,UUID:.status.nodeInfo.systemUUID` 이 검사에 실패하는 경우 클러스터의 각 노드의 공급자 ID 설정에 대한 정보는 vSphere 제품 설명서를 참조하십시오.
CollectNodeESXiVersion	노드를 실행하는 ESXi 호스트 버전을 보고합니다.
CollectNodeHWVersion	노드의 가상 머신 하드웨어 버전을 보고합니다.

17.10.6. 스토리지 클래스 구성 검사 정보

vSphere 스토리지를 사용하는 영구 볼륨의 이름은 데이터 저장소 이름 및 클러스터 ID와 관련이 있습니다.

영구 볼륨이 생성되면 systemd는 영구 볼륨의 마운트 장치를 생성합니다. systemd 프로세스에는 영구 볼륨에 사용되는 VDMK 파일의 정규화된 경로 길이에 대한 255자 제한이 있습니다.

정규화된 경로는 systemd 및 vSphere에 대한 명명 규칙을 기반으로 합니다. 명명 규칙은 다음 패턴을 사용합니다.

```
/var/lib/kubelet/plugins/kubernetes.io/vsphere-volume/mounts/[<datastore>] 00000000-0000-0000-0000-000000000000/<cluster_id>-dynamic-pvc-00000000-0000-0000-0000-000000000000.vmdk
```

- 명명 규칙에는 255자 제한 중 205자가 필요합니다.
- 데이터 저장소 이름과 클러스터 ID는 배포에서 결정됩니다.
- 데이터 저장소 이름과 클러스터 ID는 이전 패턴으로 대체됩니다. 그러면 특수 문자를 피하기 위해 systemd-escape 명령을 사용하여 경로가 처리됩니다. 예를 들어 하이픈 문자는 이스케이프 후 4자를 사용합니다. 추출된 값은 \x2d입니다.
- systemd-escape를 사용하여 systemd에서 VDMK 파일에 대한 정규화된 경로에 액세스할 수 있도록 한 후 경로 길이는 255자 미만이어야 합니다.

17.10.7. vSphere Problem Detector Operator의 지표

vSphere Problem Detector Operator는 OpenShift Container Platform 모니터링 스택에서 사용할 다음 지표를 노출합니다.

표 17.86. vSphere Problem Detector Operator가 노출하는 지표

이름	설명
vsphere_cluster_check_total	누적 수의 클러스터 수준에서 vSphere Problem Detector Operator가 수행되었는지 확인합니다. 이 카운트에는 성공 및 실패가 모두 포함됩니다.
vsphere_cluster_check_errors	vSphere Problem Detector Operator가 수행한 실패한 클러스터 수준 검사의 수입니다. 예를 들어 1의 값은 하나의 클러스터 수준 검사에 실패했음을 나타냅니다.

이름	설명
vsphere_esxi_version_total	특정 버전이 있는 ESXi 호스트의 수입니다. 호스트가 두 개 이상의 노드를 실행하는 경우 호스트는 한 번만 계산됩니다.
vsphere_node_check_total	누적 수의 노드 수준에서 vSphere Problem Detector Operator가 수행되었는지 확인합니다. 이 카운트에는 성공 및 실패가 모두 포함됩니다.
vsphere_node_check_errors	vSphere Problem Detector Operator가 수행한 실패한 노드 수준 검사의 수입니다. 예를 들어 **1**의 값은 하나의 노드 수준 검사에 실패했음을 나타냅니다.
vsphere_node_hw_version_total	특정 하드웨어 버전이 있는 vSphere 노드의 수입니다.
vsphere_vcenter_info	vSphere vCenter Server에 대한 정보입니다.

17.10.8. 추가 리소스

- 모니터링 개요

18장. VMC에 설치

18.1. VMC에 설치할 준비

18.1.1. 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.
- 방화벽을 사용하며 Telemetry를 사용할 예정이면 클러스터에 필요한 사이트를 허용하도록 방화벽을 구성해야 합니다.

18.1.2. VMC에 OpenShift Container Platform을 설치할 방법 선택

설치 관리자 프로비저닝 또는 사용자 프로비저닝 인프라를 사용하여 VMC에 OpenShift Container Platform을 설치할 수 있습니다. 기본 설치 유형은 설치 프로그램이 클러스터의 기본 인프라를 프로비저닝하는 설치 관리자 프로비저닝 인프라를 사용합니다. 제공하는 인프라에 OpenShift Container Platform도 설치할 수 있습니다. 설치 프로그램에서 프로비저닝한 인프라를 사용하지 않는 경우 클러스터 리소스를 직접 관리하고 유지보수해야 합니다.

설치 관리자 프로비저닝 및 사용자 프로비저닝 설치 프로세스에 대한 자세한 내용은 설치 프로세스를 참조하십시오.

중요

사용자가 프로비저닝한 인프라 설치를 수행하는 단계는 예시용으로만 제공됩니다. 사용자가 제공하는 인프라를 사용하여 클러스터를 설치하려면 VMC 플랫폼 및 OpenShift Container Platform 설치 프로세스에 대한 정보가 필요합니다. 사용자 프로비저닝 인프라 설치 지침을 가이드로 사용하십시오. 다른 방법을 통해 필요한 리소스를 자유롭게 생성할 수 있습니다.

18.1.2.1. 설치 관리자가 프로비저닝한 VMC에 OpenShift Container Platform을 설치

설치 프로그램에서 프로비저닝한 인프라를 사용하면 설치 프로그램에서 OpenShift Container Platform에 필요한 리소스의 프로비저닝을 사전 구성하고 자동화할 수 있습니다.

- VMC에 클러스터 설치: 사용자 지정없이 설치 관리자 프로비저닝 인프라 설치를 사용하여

VMC에 OpenShift Container Platform을 설치할 수 있습니다.

- 사용자 지정으로 VMC에 클러스터 설치: 기본 사용자 지정 옵션으로 설치 관리자 프로비저닝 인프라 설치를 사용하여 VMC에 OpenShift Container Platform을 설치할 수 있습니다.
- 네트워크 사용자 지정으로 VMC에 클러스터 설치: 설치 관리자 프로비저닝 VMC 인프라에 네트워크 사용자 지정을 사용하여 OpenShift Container Platform을 설치할 수 있습니다. 설치 중에 OpenShift Container Platform 네트워크 구성을 사용자 지정할 수 있으므로 클러스터가 기존 IP 주소 할당과 공존하고 네트워크 요구 사항을 준수할 수 있습니다.
- 제한된 네트워크의 VMC에 클러스터 설치: 설치 릴리스 컨텐츠의 내부 미러를 생성하여 제한된 네트워크의 VMC 인프라에 클러스터를 설치할 수 있습니다. 이 방법을 사용하여 인터넷에 표시되지 않는 내부 네트워크에 OpenShift Container Platform을 배포할 수 있습니다.

18.1.2.2. VMC에서 OpenShift Container Platform의 사용자 프로비저닝 인프라 설치

사용자 프로비저닝 인프라를 사용하려면 OpenShift Container Platform에 필요한 모든 리소스를 프로비저닝해야 합니다.

- 사용자 프로비저닝 인프라로 VMC에 클러스터 설치: 사용자가 제공하는 VMC 인프라에 OpenShift Container Platform을 설치할 수 있습니다.
- 사용자 프로비저닝 인프라 및 네트워크 사용자 지정으로 VMC에 클러스터 설치: 사용자 지정 네트워크 구성 옵션으로 프로비저닝하는 VMC 인프라에 OpenShift Container Platform을 설치할 수 있습니다.
- 사용자 프로비저닝 인프라가 있는 제한된 네트워크의 VMC에 클러스터 설치: 제한된 네트워크에서 프로비저닝하는 VMC 인프라에 OpenShift Container Platform을 설치할 수 있습니다.

18.1.3. VMware vSphere 인프라 요구사항

사용하는 구성 요소의 요구사항을 충족하는 VMware vSphere 버전 6 또는 7 인스턴스에 OpenShift Container Platform 클러스터를 설치해야 합니다.

표 18.1. vSphere 가상 환경의 버전 요구 사항

가상 환경 제품	필요한 버전
VM 하드웨어 버전	13 이상
vSphere ESXi hosts	6.5 이상
vCenter 호스트	6.5 이상

중요

이제 VMware vSphere 버전 6.7U2 또는 이전 버전에 클러스터를 설치하는 것이 더 이상 사용되지 않으며 가상 하드웨어 버전 13은 더 이상 사용되지 않습니다. 이러한 버전은 여전히 완전히 지원되지만 향후 OpenShift Container Platform 버전에서는 지원이 제거됩니다. 이제 OpenShift Container Platform의 vSphere 가상 머신의 하드웨어 버전 15가 기본값이 되었습니다. vSphere 노드의 하드웨어 버전을 업데이트하려면 " vSphere에서 실행되는 노드에서 하드웨어 업데이트" 문서를 참조하십시오.

표 18.2. VMware 구성 요소에 지원되는 최소 vSphere 버전

구성 요소	지원되는 최소 버전	설명
하이퍼바이저	HW 버전 13이 포함된 vSphere 6.5 이상	이 버전은 RHCOS(Red Hat Enterprise Linux CoreOS)가 지원하는 최소 버전입니다. Red Hat Enterprise Linux 8 지원 하이퍼바이저 목록을 참조하십시오.
In-tree 드라이버가 있는 스토리지	vSphere 6.5 이상	이 플러그인은 OpenShift Container Platform에 포함된 vSphere용 인트리 스토리지 드라이버를 사용하여 vSphere 스토리지를 생성합니다.

vSphere 버전 6.5 인스턴스를 사용하는 경우 OpenShift Container Platform을 설치하기 전에 6.7U3 또는 7.0로 업그레이드하는 것이 좋습니다.

중요

OpenShift Container Platform을 설치하기 전에 ESXi 호스트의 시간이 동기화되었는지 확인해야 합니다. VMware 문서에서 Edit Time Configuration for a Host를 참조하십시오.

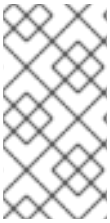
18.1.4. VMC에서 OpenShift Container Platform의 설치 관리자 프로비저닝된 인프라 설치를 제거

- VMC에서 설치 관리자 프로비저닝 인프라를 사용하는 클러스터 설치: 설치 관리자 프로비저닝 인프라를 사용하는 VMC 인프라에 배포한 클러스터를 제거할 수 있습니다.

18.2. VMC에 클러스터 설치

OpenShift Container Platform 버전 4.9에서는 AWS의 VMC(VMware Cloud)에 클러스터를 배포하여 VMware vSphere에 설치할 수 있습니다.

OpenShift Container Platform 배포를 위해 VMC 환경을 구성한 후 VMC 환경에 공동 배치된 bastion 관리 호스트에서 OpenShift Container Platform 설치 프로그램을 사용합니다. 설치 프로그램 및 컨트롤 플레인은 OpenShift Container Platform 클러스터에 필요한 리소스 배포 및 관리 프로세스를 자동화합니다.

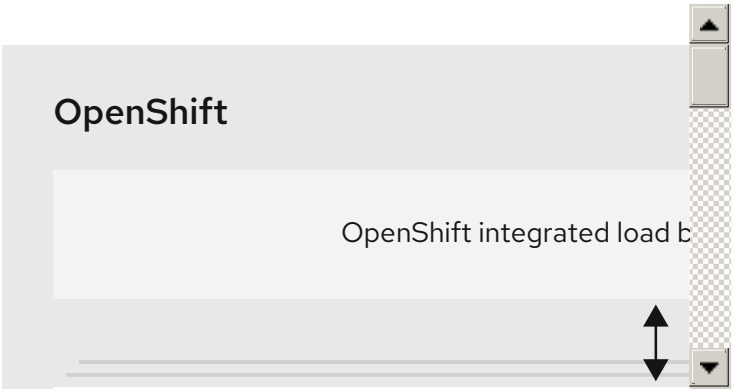


참고

OpenShift Container Platform은 단일 VMware vCenter에만 클러스터 배포를 지원합니다. 여러 vCenter에 머신/머신 세트가 있는 클러스터 배포는 지원되지 않습니다.

18.2.1. vSphere용 VMC 설정

AWS 호스팅된 vSphere 클러스터에 VMware Cloud (VMC)의 OpenShift Container Platform을 설치하여 하이브리드 클라우드 전반에 애플리케이션을 온프레미스 및 오프프레미스 모두 배포 및 관리할 수 있습니다.



VMware vSphere에 OpenShift Container Platform을 설치하기 전에 VMC 환경에서 여러 옵션을 구성해야 합니다. VMC 환경에 다음과 같은 사전 요구 사항이 있는지 확인합니다.

- 비독점 DHCP 사용, NSX-T 네트워크 세그먼트 및 서브넷을 생성합니다. 서브넷에서 다른 VM(가상 머신)을 호스팅할 수 있지만 OpenShift Container Platform 배포에서는 8개 이상의 IP

주소를 사용할 수 있어야 합니다.

- DHCP 범위 외부에서 두 개의 IP 주소를 할당하고 역방향 DNS 레코드로 구성합니다.
  - 할당된 IP 주소를 가리키는 api.<cluster_name>.<base_domain>의 DNS 레코드입니다.
  - 할당된 IP 주소를 가리키는 *.apps.<cluster_name>.<base_domain>의 DNS 레코드입니다.
- 다음 방화벽 규칙을 설정합니다.
  - OpenShift Container Platform 컴퓨팅 네트워크와 인터넷 간의 ANY:ANY 방화벽 규칙입니다. 이는 노드와 애플리케이션에서 컨테이너 이미지를 다운로드하는 데 사용됩니다.
  - 포트 443의 설치 호스트와 소프트웨어 정의 데이터 센터(SDDC) 관리 네트워크 간의 ANY:ANY 방화벽 규칙입니다. 이를 통해 배포 중에 RHCOS(Red Hat Enterprise Linux CoreOS) OVA를 업로드할 수 있습니다.
  - OpenShift Container Platform 컴퓨팅 네트워크와 인터넷 간의 HTTPS 방화벽 규칙입니다. 이 연결을 통해 OpenShift Container Platform은 노드, PVC(영구 볼륨 클레임) 및 기타 리소스를 프로비저닝 및 관리하기 위해 vCenter과 통신할 수 있습니다.
- OpenShift Container Platform을 배포하려면 다음 정보가 있어야 합니다.
  - OpenShift Container Platform 클러스터에 vmc-prod-1로 로그인합니다.
  - companyname.com과 같은 기본 DNS 이름입니다.
  - 기본값을 사용하지 않는 경우 Pod 네트워크 CIDR 및 서비스 네트워크 CIDR을 식별해야 하며, 기본적으로 10.128.0.0/14 및 172.30.0.0/16으로 설정됩니다. 이러한 CIDR은 pod-to-pod 및 pod-to-service 통신에 사용되며 외부에서 액세스할 수 없지만 조직의 기존 서브넷과 중복되어서는 안 됩니다.

- 다음 vCenter 정보를 참조하십시오:
  - vCenter 호스트 이름, 사용자 이름 및 암호
  - 데이터 센터 이름 (예: SDDC-Datacenter)
  - Cluster-1과 같은 클러스터 이름
  - 네트워크 이름
  - WorkloadDatastore와 같은 데이터 저장소 이름

참고

클러스터 설치가 완료된 후 vSphere 클러스터를 VMC Compute-ResourcePool 리소스 풀로 이동하는 것이 좋습니다.

- VMC에 bastion으로 배포된 Linux 기반 호스트입니다.
  - bastion 호스트는 RHEL(Red Hat Enterprise Linux) 또는 기타 Linux 기반 호스트일 수 있습니다. 인터넷에 연결되어 있어야 하고 OVA를 ESXi 호스트에 업로드하는 기능이 있어야 합니다.
  - OpenShift CLI 툴을 bastion 호스트에 다운로드하여 설치합니다.
    - openshift-install 설치 프로그램
    - OpenShift CLI(oc) 툴

참고

Kubernetes 용 VMware NCP (NSX Conrainer Plugin)dOpen에는 사용할 수 없으며 NSX는 OpenShift SDN으로 사용되지 않습니다. 현재 VMC에서 사용할 수 있는 NSX 버전은 OpenShift Container Platform에서 인증된 NCP 버전과 호환되지 않습니다.

그러나 NSX DHCP 서비스는 전체 스택 자동화된 OpenShift Container Platform 배포와 vSphere와의 Machine API 통합을 통해 수동 또는 자동으로 프로비저닝된 노드를 사용하여 가상 머신 IP 관리에 사용됩니다. 또한 OpenShift Container Platform 클러스터와 bastion 호스트와 VMC vSphere 호스트 사이에서 액세스할 수 있도록 NSX 방화벽 규칙이 생성됩니다.

18.2.1.1. VMC Sizer 툴

AWS의 VMware Cloud는 AWS 베어메탈 인프라 상단에 구축됩니다. 이는 AWS 네이티브 서비스를 실행하는 것과 동일한 베어 메탈 인프라입니다. AWS 소프트웨어 정의 데이터 센터(SDDC)의 VMware 클라우드가 배포되면 이러한 물리적 서버 노드를 사용하고 단일 테넌트 방식으로 VMware ESXi 하이퍼바이저를 실행할 수 있습니다. 즉, VMC를 사용하는 모든 사용자가 물리적 인프라에 액세스할 수 없습니다. 가상 인프라를 호스팅하는데 필요한 물리적 호스트 수를 고려해야 합니다.

이를 확인하기 위해 VMware는 AWS Sizer에서 VMC를 제공합니다. 이 툴을 사용하면 VMC에서 호스팅할 리소스를 정의할 수 있습니다.

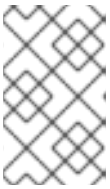
- 워크로드 유형
- 총 가상 머신 수
- 사양 정보는 다음과 같습니다:
  - 스토리지 요구사항
  - vCPU
  - vRAM

- 오버커밋 비율

이러한 세부 정보를 사용하여 VMware 모범 사례를 기반으로 보고서를 생성하고 클러스터 구성과 필요한 호스트 수를 권장합니다.

18.2.2. vSphere 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.
- 블록 레지스트리 스토리지가 프로비저닝되어 있습니다. 자세한 내용은 영구 저장 장치 이해를 참조하십시오.
- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성해야 합니다.



참고

프록시를 구성하는 경우에도 해당 사이트 목록을 검토하십시오.

18.2.3. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.

- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.

중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

18.2.4. VMware vSphere 인프라 요구사항

사용하는 구성 요소의 요구사항을 충족하는 VMware vSphere 버전 6 또는 7 인스턴스에 OpenShift Container Platform 클러스터를 설치해야 합니다.

표 18.3. vSphere 가상 환경의 버전 요구 사항

가상 환경 제품	필요한 버전
VM 하드웨어 버전	13 이상
vSphere ESXi hosts	6.5 이상
vCenter 호스트	6.5 이상

중요

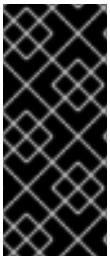
이제 VMware vSphere 버전 6.7U2 또는 이전 버전에 클러스터를 설치하는 것이 더 이상 사용되지 않으며 가상 하드웨어 버전 13은 더 이상 사용되지 않습니다. 이러한 버전은 여전히 완전히 지원되지만 향후 OpenShift Container Platform 버전에서는 지원이 제거됩니다. 이제 OpenShift Container Platform의 vSphere 가상 머신의 하드웨어 버전 15가 기본값이 되었습니다. vSphere 노드의 하드웨어 버전을 업데이트하려면 " vSphere에서 실행되는 노드에서 하드웨어 업데이트" 문서를 참조하십시오.

표 18.4. VMware 구성 요소에 지원되는 최소 vSphere 버전

구성 요소	지원되는 최소 버전	설명
하이퍼바이저	HW 버전 13이 포함된 vSphere 6.5 이상	이 버전은 RHCOS(Red Hat Enterprise Linux CoreOS)가 지원하는 최소 버전입니다. Red Hat Enterprise Linux 8 지원 하이퍼바이저 목록을 참조하십시오.

구성 요소	지원되는 최소 버전	설명
In-tree 드라이버가 있는 스토리지	vSphere 6.5 이상	이 플러그인은 OpenShift Container Platform에 포함된 vSphere용 인트리 스토리지 드라이버를 사용하여 vSphere 스토리지를 생성합니다.

vSphere 버전 6.5 인스턴스를 사용하는 경우 OpenShift Container Platform을 설치하기 전에 6.7U3 또는 7.0로 업그레이드하는 것이 좋습니다.



중요

OpenShift Container Platform을 설치하기 전에 ESXi 호스트의 시간이 동기화되었는지 확인해야 합니다. VMware 문서에서 Edit Time Configuration for a Host를 참조하십시오.

18.2.5. 네트워크 연결 요구사항

OpenShift Container Platform 클러스터 구성 요소가 통신할 수 있도록 시스템 간 네트워크 연결을 구성해야 합니다.

필요한 네트워크 포트에 대한 다음 세부 정보를 검토합니다.

표 18.5. 모든 시스템 간 통신에 사용되는 포트

프로토콜	포트	설명
ICMP	해당 없음	네트워크 연결성 테스트
TCP	**1936**	메트릭
	9000-9999	**9100-9101** 포트의 노드 내보내기 및 **9099** 포트의 Cluster Version Operator를 포함한 호스트 수준 서비스.
	10250-10259	Kubernetes에서 예약하는 기본 포트
	10256	openshift-sdn
UDP	**4789**	VXLAN(Virtual extensible LAN)
	6081	Geneve

프로토콜	포트	설명
	9000-9999	**9100-9101** 포트의 노드 내보내기를 포함한 호스트 수준 서비스.
	500	IPsec IKE 패킷
	4500	IPsec NAT-T 패킷
TCP/UDP	**30000-32767**	Kubernetes 노드 포트
ESP	해당 없음	IPsec Encapsulating Security Payload (ESP)

표 18.6. 모든 시스템과 컨트롤 플레인 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**6443**	Kubernetes API

표 18.7. 컨트롤 플레인 머신 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**2379-2380**	etcd 서버 및 피어 포트

추가 리소스

- **vSphere** 노드의 하드웨어 버전을 업데이트하려면 **vSphere에서 실행 중인 노드에서 하드웨어 업데이트를 참조하십시오.**

18.2.6. vCenter 요구사항

설치 관리자가 프로비저닝하는 인프라를 사용하는 vCenter에 OpenShift Container Platform 클러스터를 설치하기 전에 환경을 준비해야 합니다.

필요한 vCenter 계정 권한

vCenter에 OpenShift Container Platform 클러스터를 설치하려면 설치 프로그램에서 필요한 리소스를 읽고 생성할 수 있는 권한이 있는 계정에 액세스해야 합니다. 필요한 모든 권한에 액세스할 수 있는 가장 간단한 방법은 글로벌 관리 권한이 있는 계정을 사용하는 것입니다.

글로벌 관리 권한이 있는 계정을 사용할 수 없는 경우 OpenShift Container Platform 클러스터 설치에 필요한 권한을 부여하려면 역할을 생성해야 합니다. 대부분의 권한이 항상 필요하지만 기본 동작인

vCenter 인스턴스에 **OpenShift Container Platform** 클러스터를 포함하는 폴더를 프로비저닝할 설치 프로그램인 경우에만 필요합니다. 필요한 권한을 부여하려면 지정된 오브젝트에 대해 **vSphere** 역할을 생성하거나 수정해야 합니다.

설치 프로그램이 **vSphere** 가상 머신 폴더를 생성하는 경우 추가 역할이 필요합니다.

예 18.1. vSphere API에 설치에 필요한 역할 및 권한

역할의 vSphere 개체	필요한 경우	vSphere API에서 필요한 권한
vSphere vCenter	Always	CNS **.Searchable InventoryService.Tagging.AttachTag InventoryService.Tagging.CreateCategory InventoryService.Tagging.CreateTag InventoryService.Tagging.Delete**Tag **InventoryService. tagged.DeleteTag InventoryService.Tagging.EditCategory InventoryService.Tagging.EditTag**ECDHE**Session ValidateSession**Forwarded**StorageProfile.Update**Forwarded**StorageProfile.View**
vSphere vCenter Cluster	Always	**Host.Config.Storage Resource.AssignVMToPool VApp.AssignResourcePool VApp.Import VirtualMachine.Config.Add NewDisk**
vSphere Datastore	Always	**Datastore.AllocateSpace Datastore.LoadBalancer Datastore.FileManagement InventoryService.Tagging.O bjectAttachable**
vSphere Port Group	Always	**Network.Assign**

역할의 vSphere 개체	필요한 경우	vSphere API에서 필요한 권한
가상 머신 폴더	Always	**InventoryService.ume.ObjectAttachable Resource.AssignVMToPool VApp.Import VirtualMachine.Config.Add ExistingDisk VirtualMachine.Config.Add NewDisk VirtualMachine.Config.Add RemoveDevice VirtualMachine.Config.Advanced Config VirtualMachine.Config.Anno tationECDHEVirtualMachine .Config.DiskExtend**Forwarde d VirtualMachine**.Config.**DiskExte nd Forwarded**VirtualMachine.Co nfig.DiskLeaseVirtualMachi ne.Config.ReCreate**VolumeCl aim.Config**.MemoryDriver.**

역할의 vSphere 개체	필요한 경우	vSphere API에서 필요한 권한
vSphere vCenter Datacenter	설치 프로그램이 가상 머신 폴더를 생성하는 경우	**InventoryService.Tagging.ObjectAttachable Resource.AssignVMToPool VApp.Import VirtualMachine.Config.Add ExistingDisk VirtualMachine.Config.Add NewDisk VirtualMachine.Config.Add RemoveDevice VirtualMachine.Config.Advanced Config VirtualMachine.Config.Anno tationECDHEVirtualMachine .Config.DiskExtend**Forwarde d VirtualMachine**.**Config**.DiskExt end ForwardedVirtualMachine.C onfig.DiskLeaseVirtualMach ine.Config.ReCreate VolumeClaim.Config.Reset**d efined**.**

예 18.2. vCenter 그래픽 사용자 인터페이스(GUI)에 설치하는 데 필요한 역할 및 권한

역할의 vSphere 개체	필요한 경우	vCenter GUI에서 필요한 권한
vSphere vCenter	Always	CNS **.Searchable "vSphere agged""Assign or unassign vSphere Tag" "vSphere tagging" "vSphere" "vSphere" "vSphere"CreatevSphere Tag" vSphere 태그""vSphere" 삭제 태그 지정" "vSphere Assignment""Edit vSphere Tag" "vSphere Tagging" "vSphere 태그" 세션" "Profile 기반 스토리지""Profile 기반 스토리지 업데이트""Profile 기반 스토리지"Profile 중심 스토리지 업데이트"Profile 기반 스토리지 업데이트"Profile 기반 스토리지 업데이트 스토리지 보기"**
vSphere vCenter Cluster	클러스터 루트에서 VM이 생성되는 경우	**Host.Configuration"Storage partition configuration" 리소스" 리소스 풀에 가상 머신을 서명" VApp"Assign resource pool" VApp.Import"Virtual machine""Change Configuration"."새 디스크 추가"**
vSphere vCenter 리소스 풀	기존 리소스 풀이 제공되는 경우	**Host.Configuration"Storage partition configuration" 리소스" 리소스 풀에 가상 머신을 서명" VApp"Assign resource pool" VApp.Import"Virtual machine""Change Configuration"."새 디스크 추가"**

역할의 vSphere 개체	필요한 경우	vCenter GUI에서 필요한 권한
vSphere Datastore	Always	**Datastore"Allocate space"** 데이터**저장소"** **데이터 저장소"** **Datastore"Low level file operations""vSphere Assignment"Assign or Unassign vSphere Tag on Object"**
vSphere Port Group	Always	**네트워크 "Assign network"**
가상 머신 폴더	Always	**"vSphere Assignment""Assign or unassign vSphere Tag on Object"** **Resource"Assign virtual machine to resource pool"**: **VApp.Import** **"Virtual machine""Change Configuration","Add existing disk""**Add existing disk**""Add new disk""Add new disk" "Virtual""** **머신 "시스템 변경", "가상 머신""** **"가상 머신""시스템 변경"** **"가상 머신""변경 구성",** **"가상 머신", 변경, CPU 수 변경"** **"가상 머신", 가상 디스크 변경", 가상 디스크 변경"** **"가상머신". 머신 변경 "구성 변경"** **"가상 머신""마이크 구성 변경"** **"가상 머신""마이크 구성 변경" "구성 변경",** **"가상 머신", 변경" 디스크 삭제"** **"가상 머신".변경 구성".Rename** **"가상 머신 변경" VIX API**로가상 머신 변경**"Gregradevirtual machine"Guestoperating system management by VIX API**

역할의 vSphere 개체	필요한 경우	vCenter GUI에서 필요한 권한
		machine". interaction" "Virtual machine". Interconnection"Power on" "Virtual machine". interactionion.Reset "Virtual machine"."Edit Inventory"."Create new" "Virtual machine"."Create existing" "Virtual machine"Edit Inventory""Create from existing" "Virtual machine"Edit Inventory""Create from existing" virtual machine".Edit Inventory""Create from existing" virtual machine".Edit Inventory""Create from existing""Virtual machine" machine".Provisioning."Clone 가상 머신" "가상 머신".Provisioning." "가상 머신".Provisioning."Deploy 템플릿"
vSphere vCenter Datacenter	설치 프로그램이 가상 머신 폴더를 생성하는 경우	"vSphere Assignment""Assign or unassign vSphere Tag on Object" Resource"Assign virtual machine to resource pool": VApp.Import "Virtual machine""Change Configuration","Add existing disk""Add existing disk""Add new disk""Add new disk" "Virtual"" 머신 "시스템 변경", "가상 머신"" "가상 머신""시스템 변경" "가상 머신""변경 구성", "가상 머신", 변경, CPU 수 변경" "가상 머신", 가상 디스크 변경", 가상 디스크 변경" "가상머신". 머신 변경 "구성 변경" "가상 머신""마이크 구성 변경" "가상 머신""마이크 구성 변경" "구성 변경", "가상 머신", 변경" 디스크 삭

역할의 vSphere 개체	필요한 경우	vCenter GUI에서 필요한 권한
		제" "가상 머신 .권한 부 여".Rename "가상 머신 변경" VIX API로가상 머신 변 경"Gregradevirtual machine"Guestoperating system management by VIX API machine". interaction" "Virtual machine". Interconnection"Power on" "Virtual machine". interactionion.Reset "Virtual machine"."Edit Inventory"."Create new" "Virtual machine"."Create existing" "Virtual machine"Edit Inventory""Create from existing" "Virtual machine"Edit Inventory""Create from existing" virtual machine".Edit Inventory""Create from existing" virtual machine".Edit Inventory""Create from existing""Virtual machine" machine".Provisioning."Clone 가상 머신" "가상 머 신".Provisioning."Deploy 템플릿" "가상 머신".Provisioning" 폴더 만들기" 폴더 생성"폴더 삭제"

또한 사용자에게 일부 **ReadOnly** 권한이 필요하며 일부 역할에는 하위 오브젝트에 권한을 부여하는 사용 권한이 필요합니다. 이러한 설정은 클러스터를 기존 폴더에 설치할지 여부에 따라 달라집니다.

예 18.3. 필수 권한 및 권한 부여 설정

vSphere 오브젝트	폴더 유형	하위 항목으로 권한 부여	권한 필요
vSphere vCenter	Always	False	나열된 필수 권한
vSphere vCenter Datacenter	기존 폴더	False	**ReadOnly** 권한
	설치 프로그램은 폴더를 생성	True	나열된 필수 권한
vSphere vCenter Cluster	Always	True	나열된 필수 권한
vSphere vCenter Datastore	Always	False	나열된 필수 권한
vSphere Switch	Always	False	**ReadOnly** 권한
vSphere Port Group	Always	False	나열된 필수 권한
vSphere vCenter Virtual Machine Folder	기존 폴더	True	나열된 필수 권한

필요한 권한만으로 계정을 생성하는 방법에 대한 자세한 내용은 **vSphere** 문서에서 **vSphere 권한 및 사용자 관리 작업**을 참조하십시오.

vMotion으로 OpenShift Container Platform 사용

vSphere 환경에서 **vMotion**을 사용하려는 경우 **OpenShift Container Platform** 클러스터를 설치하기 전에 다음을 고려하십시오.

- **OpenShift Container Platform**은 일반적으로 **compute-only vMotion** 을 지원합니다. 스토리지 **vMotion**을 사용하면 문제가 발생할 수 있으며 지원되지 않습니다.

  컴퓨팅 및 컨트롤 플레인 노드의 가동 시간을 보장하기 위해 **vMotion**에 대한 **VMware** 모범 사례를 따르는 것이 좋습니다. 또한 **VMware** 유사성 방지 규칙을 사용하여 유지 관리 또는 하드웨어 문제 중에 **OpenShift Container Platform**의 가용성을 개선하는 것이 좋습니다.

  vMotion 및 유사성 방지 규칙에 대한 자세한 내용은 **vMotion 네트워킹 요구 사항** 및 **VM 유사성 방지 규칙에** 대한 **VMware vSphere** 설명서를 참조하십시오.

- **Pod**에서 **vSphere** 볼륨을 사용하는 경우 데이터 저장소를 통해 **VM**을 마이그레이션하면 수동으로 또는 스토리지 **vMotion**을 통해 **VM**을 마이그레이션하면 **OpenShift Container Platform**

PV(영구 볼륨) 오브젝트 내에서 잘못된 참조가 표시됩니다. 이러한 참조는 영향을 받는 포드가 시작되지 않게 하고 데이터 손실을 초래할 수 있습니다.

- 마찬가지로 OpenShift Container Platform은 데이터 저장소에서 VMDK의 선택적 마이그레이션을 지원하지 않으며 VM 프로비저닝 또는 PV의 동적 또는 정적 프로비저닝을 위해 데이터 저장소 클러스터의 일부 또는 PV의 동적 또는 정적 프로비저닝을 위해 데이터 저장소 클러스터의 일부인 데이터 저장소를 사용하는 경우도 있습니다.

클러스터 리소스

설치 관리자 프로비저닝 인프라를 사용하는 OpenShift Container Platform 클러스터를 배포할 때는 설치 프로그램이 vCenter 인스턴스에서 여러 리소스를 생성할 수 있어야 합니다.

표준 OpenShift Container Platform을 설치하면 다음 vCenter 리소스가 생성됩니다.

- 폴더 한 개
- 태그 카테고리 한 개
- 태그 한 개
- 가상 머신:
  - 템플릿 한 개
  - 임시 부트스트랩 노드 한 개
  - 컨트롤 플레인 노드 세 개
  - 컴퓨팅 시스템 세 개

이러한 리소스는 856GB의 스토리지를 사용하지만 부트스트랩 노드는 클러스터 설치 프로세스 중에 제거됩니다. 표준 클러스터를 사용하려면 최소 800GB의 스토리지가 필요합니다.

컴퓨팅 시스템을 더 많이 배포할수록 **OpenShift Container Platform** 클러스터는 더 많은 스토리지를 사용합니다.

클러스터 제한

사용 가능한 리소스는 클러스터마다 다릅니다. **vCenter** 내에서 가능한 클러스터 수는 주로 사용 가능한 스토리지 공간과 필요한 리소스 수에 대한 제한으로 제한됩니다. 클러스터가 생성하는 **vCenter** 리소스와 **IP** 주소 및 네트워크와 같이 클러스터를 배포하는 데 필요한 리소스에 대한 제한 사항을 모두 고려해야 합니다.

네트워킹 요구사항

네트워크에 **DHCP**를 사용하고 **DHCP** 서버가 클러스터 시스템에 영구 **IP** 주소를 제공하도록 구성되어 있는지 확인합니다. **DHCP** 서버를 사용하도록 기본 게이트웨이를 구성해야 합니다. 모든 노드는 동일한 **VLAN**에 있어야 합니다. 두 번째 **VLAN**을 **Day 2** 작업으로 사용하여 클러스터를 확장할 수 없습니다. 또한 **OpenShift Container Platform** 클러스터를 설치하기 전에 다음 네트워킹 리소스를 생성해야 합니다.

참고

클러스터의 각 **OpenShift Container Platform** 노드는 **DHCP**를 통해 검색할 수 있는 **NTP(Network Time Protocol)** 서버에 액세스할 수 있어야 합니다. **NTP** 서버 없이도 설치할 수 있습니다. 그러나 비동기 서버 클록으로 인해 **NTP** 서버가 차단하는 오류가 발생합니다.

필요한 IP 주소

설치 관리자 프로비저닝 **vSphere** 설치에는 다음 두 개의 고정 **IP** 주소가 필요합니다.

- **API** 주소는 클러스터 **API**에 액세스하는 데 사용됩니다.
- **Ingress** 주소는 클러스터 인그레스 트래픽에 사용됩니다.

OpenShift Container Platform 클러스터를 설치할 때 설치 프로그램에 이 **IP** 주소를 제공해야 합니다.

DNS 레코드

OpenShift Container Platform 클러스터를 호스팅하는 **vCenter** 인스턴스에 적절한 **DNS** 서버에서 고정 **IP** 주소 두 개의 **DNS** 레코드를 생성해야 합니다. 각 레코드에서 **<cluster_name>**은 클러스터 이름이고 **<base_domain>**은 클러스터를 설치할 때 지정하는 클러스터 기본 도메인입니다. 전체 **DNS** 레코드는 **<component>.<cluster_name>.<base_domain>** 형식입니다.

표 18.8. 필수 DNS 레코드

구성 요소	레코드	설명
API VIP	**api.<cluster_name>.<base_domain>.**	이 DNS A/AAAA 또는 CNAME 레코드는 컨트롤 플레인 시스템의 로드 밸런서를 가리켜야 합니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.
Ingress VIP	***.apps.<cluster_name>.<base_domain>.**	기본적으로 작업자 노드인 인그레스 라우터 Pod를 실행하는 시스템을 대상으로 하는 로드 밸런서를 가리키는 와일드카드 DNS A/AAAA 또는 CNAME 레코드입니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.

18.2.7. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 **SSH** 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 **Ignition** 구성 파일을 통해 **RHCOS(Red Hat Enterprise Linux CoreOS)** 노드에 전달되며 노드에 대한 **SSH** 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 **core** 사용자의 **~/.ssh/authorized_keys** 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 **core**로 **RHCOS** 노드에 **SSH**로 **SSH** 연결을 수행할 수 있습니다 . **SSH**를 통해 노드에 액세스하려면 로컬 사용자의 **SSH**에서 개인 키 **ID**를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 **SSH**를 실행하려면 설치 프로세스 중에 **SSH** 공용 키를 지정해야 합니다. **./openshift-install gather** 명령에도 **SSH** 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

참고

AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

절차

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: ~/.ssh/id_ed25519)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 '~/.ssh 디렉터리에 있는지 확인하십시오.

참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 ed25519 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 rsa 또는 ecdsa 알고리즘을 사용하는 키를 생성합니다.

2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 ~/.ssh/id_ed25519.pub 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 ./openshift-install gather 명령을 사용하려는 경우 필요합니다.

참고

일부 배포에서는 ~/.ssh/id_rsa 및 ~/.ssh/id_dsa와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. ssh-agent 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. ssh-agent에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: ~/.ssh/id_ed25519).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

18.2.8. 설치 프로그램 받기

OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

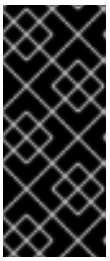
- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

절차

1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.

2. 인프라 공급자를 선택합니다.

3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.



중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.

중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 OpenShift Container Platform 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

18.2.9. 시스템 신뢰에 vCenter 루트 CA 인증서 추가

설치 프로그램이 vCenter API에 액세스해야 하므로 OpenShift Container Platform 클러스터를 설치하기 전에 vCenter의 신뢰할 수 있는 루트 CA 인증서를 시스템 신뢰에 추가해야 합니다.

절차

1. vCenter 홈 페이지에서 vCenter의 루트 CA 인증서를 다운로드합니다. vSphere Web Services SDK 섹션에서 Download trusted root CA certificates를 클릭합니다. <vCenter>/certs/download.zip 파일이 다운로드됩니다.

2. vCenter 루트 CA 인증서가 포함된 압축 파일의 압축을 풉니다. 압축 파일의 내용은 다음 파일 구조와 유사합니다.

```
certs
├── lin
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
├── mac
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
└── win
    ├── 108f4d17.0.crt
    ├── 108f4d17.r1.crl
    ├── 7e757f6a.0.crt
    ├── 8e4f8471.0.crt
    └── 8e4f8471.r0.crl

3 directories, 15 files
```

3. 운영 체제 파일을 시스템 신뢰에 추가합니다. 예를 들어 Fedora 운영 체제에서는 다음 명령을 실행합니다.

```
# cp certs/lin/* /etc/pki/ca-trust/source/anchors
```

4. 시스템 신뢰를 업데이트합니다. 예를 들어 Fedora 운영 체제에서는 다음 명령을 실행합니다.

```
# update-ca-trust extract
```

18.2.10. 클러스터 배포

호환되는 클라우드 플랫폼에 **OpenShift Container Platform**을 설치할 수 있습니다.

중요

최초 설치 과정에서 설치 프로그램의 **create cluster** 명령을 한 번만 실행할 수 있습니다.

사전 요구 사항

- 클러스터를 호스팅하는 클라우드 플랫폼으로 계정을 구성합니다.
- **OpenShift Container Platform** 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

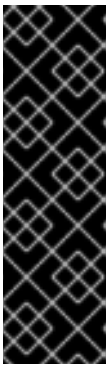
절차

1. 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터 배포를 초기화합니다.

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
  --log-level=info 2
```

1 **<installation_directory>**는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.

2 다른 설치 세부 사항을 보려면 **info** 대신 **warn, debug** 또는 **error**를 지정합니다.

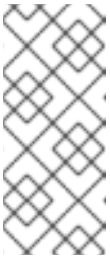


중요

비어 있는 디렉터리를 지정합니다. 부트스트랩 **X.509** 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 **OpenShift Container Platform** 버전에서 설치 파일을 복사할 때는 주의하십시오.

화면에 나타나는 지시에 따라 필요한 값을 입력합니다.

a. 선택사항: 클러스터 시스템에 액세스하는 데 사용할 SSH 키를 선택합니다.



참고

설치 디버깅 또는 재해 복구 수행을 원하는 프로덕션 환경 OpenShift Container Platform 클러스터의 경우 ssh-agent 프로세스에 사용할 SSH 키를 지정하십시오.

b. 대상 플랫폼으로 vsphere를 선택합니다.

c. vCenter 인스턴스의 이름을 지정합니다.

d. 클러스터를 생성하는 데 필요한 권한이 있는 vCenter 계정의 사용자 이름과 암호를 지정합니다.

설치 프로그램이 vCenter 인스턴스에 연결됩니다.

e. 연결할 vCenter 인스턴스에서 데이터 센터를 선택합니다.

f. 사용할 기본 vCenter 데이터 저장소를 선택합니다.

참고

Datastore 및 클러스터 이름은 60자를 초과할 수 없으므로 결합된 문자열 길이가 60자 제한을 초과하지 않도록 합니다.

g. OpenShift Container Platform 클러스터를 설치할 vCenter 클러스터를 선택합니다. 설치 프로그램은 vSphere 클러스터의 루트 리소스 풀을 기본 리소스 풀로 사용합니다.

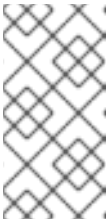
h. vCenter 인스턴스에서 구성한 가상 IP 주소 및 DNS 레코드가 포함된 네트워크를 선택합니다.

i. 컨트롤 플레인 **API** 액세스를 위해 구성한 가상 **IP** 주소를 입력합니다.

j. 클러스터 인그레스용으로 구성한 가상 **IP** 주소를 입력합니다.

k. 기본 도메인을 입력합니다. 이 기본 도메인은 구성한 **DNS** 레코드에서 사용한 것과 동일해야 합니다.

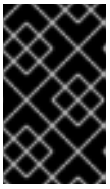
l. 클러스터를 설명할 수 있는 이름을 입력합니다. 클러스터 이름은 구성한 **DNS** 레코드에서 사용한 것과 동일해야 합니다.



참고

Datastore 및 클러스터 이름은 **60**자를 초과할 수 없으므로 결합된 문자열 길이가 **60**자 제한을 초과하지 않도록 합니다.

m. Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 붙여넣습니다.



중요

VMC 환경에서 호스팅되는 **bastion**의 **openshift-install** 명령을 사용합니다.

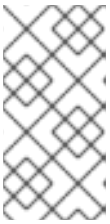
참고

호스트에 구성된 클라우드 공급자 계정에 클러스터를 배포하기에 충분한 권한이 없는 경우, 설치 프로세스가 중단되고 누락된 권한을 알리는 메시지가 표시됩니다.

클러스터 배포가 완료되면 웹 콘솔로 연결되는 링크와 **kubeadmin** 사용자의 인증 정보가 포함된 클러스터 액세스 지침이 사용자 터미널에 표시됩니다.

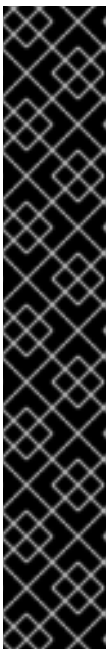
출력 예

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-
ymBZj-Wt5AL"
INFO Time elapsed: 36m22s
```



참고

설치에 성공하면 클러스터 액세스 및 인증 정보도 **<installation_directory>/.openshift_install.log**로 출력됩니다.



중요

- 설치 프로그램에서 생성하는 **Ignition** 구성 파일에 **24**시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 **24**시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 **kubelet** 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(**CSR**)을 수동으로 승인해야 합니다. 자세한 내용은 ***Recovering from expired control plane certificates*** 문서를 참조하십시오.

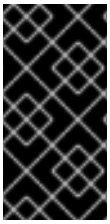
- 클러스터를 설치한 후 **24**시간에서 **22**시간까지의 인증서가 교체되기 때문에 생성된 후 **12**시간 이내에 **Ignition** 구성 파일을 사용하는 것이 좋습니다. **12**시간 이내에 **Ignition** 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

중요

설치 프로그램에서 생성되는 파일이나 설치 프로그램을 삭제해서는 안 됩니다. 클러스터를 삭제하려면 두 가지가 모두 필요합니다.

18.2.11. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 **OpenShift Container Platform**과 상호 작용하기 위해 **OpenShift CLI(oc)**를 설치할 수 있습니다. **Linux, Windows** 또는 **macOS**에 **oc**를 설치할 수 있습니다.



중요

이전 버전의 oc를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 oc를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 Linux에서 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

절차

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.
2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. OpenShift v4.9 Linux Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. oc 바이너리를 PATH에 있는 디렉터리에 배치합니다.

   PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치

다음 절차에 따라 Windows에 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

절차

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.

2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.

3. OpenShift v4.9 Windows Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.

4. ZIP 프로그램으로 아카이브의 압축을 풉니다.

5. oc 바이너리를 PATH에 있는 디렉터리로 이동합니다.

   PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치

다음 절차에 따라 macOS에서 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

절차

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.

2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.

3. OpenShift v4.9 MacOSX Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.

4. 아카이브의 압축을 해제하고 압축을 풉니다.

5. oc 바이너리 PATH의 디렉터리로 이동합니다.

   PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
$ oc <command>
```

18.2.12. CLI를 사용하여 클러스터에 로그인

클러스터 kubeconfig 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. kubeconfig 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- oc CLI를 설치했습니다.

절차

1. kubeadmin 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 oc 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

18.2.13. 레지스트리 스토리지 생성

클러스터를 설치한 후 레지스트리 Operator를 위한 스토리지를 생성해야 합니다.

18.2.13.1. 설치 중 제거된 이미지 레지스트리

공유 가능한 개체 스토리지를 제공하지 않는 플랫폼에서 OpenShift Image Registry Operator는 자체적으로 Removed로 부트스트랩합니다. 이를 통해 openshift-installer가 이러한 플랫폼 유형에서 설치를 완료할 수 있습니다.

설치 후 managementState를 Removed에서 Managed로 전환하도록 Image Registry Operator 구성을 편집해야 합니다.

참고

Prometheus 콘솔은 ImageRegistryRemoved 경고를 제공합니다. 다음은 예시 경고입니다.

"이미지 레지스트리가 제거되었습니다. ImageStreamTags를 참조하는 ImageStreamTags, BuildConfigs 및 DeploymentConfigs가 예상대로 작동하지 않을 수 있습니다. 스토리지를 구성하고 configs.imageregistry.operator.openshift.io를 편집하여 구성을 Managed 상태로 업데이트하십시오."

18.2.13.2. 이미지 레지스트리 스토리지 구성

기본 스토리지를 제공하지 않는 플랫폼에서는 처음에 Image Registry Operator를 사용할 수 없습니다. 설치한 후에 스토리지를 사용하도록 레지스트리를 구성하여 Registry Operator를 사용 가능하도록 만들어야 합니다.

프로덕션 클러스터에 필요한 영구 볼륨을 구성하는 과정의 지침이 표시됩니다. 해당하는 경우, 프로덕션 환경 외 클러스터에서만 사용할 수 있는 저장 위치로서 빈 디렉터리를 구성하는 과정의 지침이 표시됩

니다.

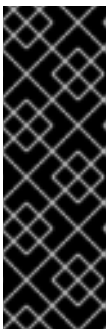
업그레이드 중에 **Recreate** 롤아웃 전략을 사용하여 이미지 레지스트리의 블록 스토리지 유형 사용을 허용하기 위한 추가 지침이 제공됩니다.

18.2.13.2.1. VMware vSphere용 레지스트리 스토리지 구성

클러스터 관리자는 설치한 후 스토리지를 사용하도록 레지스트리를 구성해야 합니다.

사전 요구 사항

- 클러스터 관리자 권한이 있어야 합니다.
- VMware vSphere에 클러스터가 있어야 합니다.
- Red Hat OpenShift Container Storage와 같이 클러스터용 영구 스토리지 프로비저닝.



중요

OpenShift Container Platform은 복제본이 하나만 있는 경우 이미지 레지스트리 스토리지에 대한 **ReadWriteOnce** 액세스를 지원합니다. **ReadWriteOnce** 액세스에서는 레지스트리가 **Recreate** 롤아웃 전략을 사용해야 합니다. 두 개 이상의 복제본으로 고 가용성을 지원하는 이미지 레지스트리를 배포하려면 **ReadWriteMany** 액세스가 필요합니다.

- "100Gi" 용량이 필요합니다.

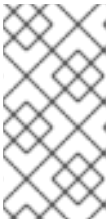
중요

테스트 결과, RHEL의 NFS 서버를 핵심 서비스용 스토리지 백엔드로 사용하는 데 문제가 있는 것으로 나타납니다. 여기에는 OpenShift Container Registry and Quay, 스토리지 모니터링을 위한 Prometheus, 로깅 스토리지를 위한 Elasticsearch가 포함됩니다. 따라서 RHEL NFS를 사용하여 핵심 서비스에서 사용하는 PV를 백업하는 것은 권장되지 않습니다.

마켓플레이스의 다른 NFS 구현에는 이러한 문제가 나타나지 않을 수 있습니다. 이러한 OpenShift Container Platform 핵심 구성 요소에 대해 완료된 테스트에 대한 자세한 내용은 개별 NFS 구현 공급업체에 문의하십시오.

프로세스

1. 스토리지를 사용하도록 레지스트리를 구성하기 위해 configs.imageregistry/cluster 리소스에서 spec.storage.pvc를 변경합니다.



참고

공유 스토리지를 사용할 때 보안 설정을 확인하여 외부에서의 액세스를 방지합니다.

2. 레지스트리 pod가 없는지 확인합니다.

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

출력 예

```
No resourses found in openshift-image-registry namespace
```

참고

출력에 레지스트리 Pod가 있는 경우 이 절차를 계속할 필요가 없습니다.

3.
레지스트리 구성을 확인합니다.

```
$ oc edit configs.imageregistry.operator.openshift.io
```

출력 예

```
storage:
  pvc:
    claim: 1
```

1
image-registry-storage 영구 볼륨 클레임 (PVC)을 자동으로 생성할 수 있도록 claim 필드를 비워 둡니다. PVC는 기본 스토리지 클래스를 기반으로 생성됩니다. 그러나 기본 스토리지 클래스에서 RADOS 블록 장치(RBD)와 같은 ReadWriteOnce(ReadWriteOnce) 볼륨을 제공할 수 있으므로 복제본이 두 개 이상될 때 문제가 발생할 수 있습니다.

4.
clusteroperator 상태를 확인합니다.

```
$ oc get clusteroperator image-registry
```

출력 예

```
NAME             VERSION                       AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                           True        False         False     6h50m
```

18.2.13.2.2. VMware vSphere용 블록 레지스트리 스토리지 구성

클러스터 관리자로서 업그레이드 중에 이미지 레지스트리가 vSphere VMDK(Virtual Machine Disk)와 같은 블록 스토리지 유형을 사용할 수 있도록 허용하기 위해 Recreate 롤아웃 전략을 사용할 수 있습니다.

중요

블록 스토리지 볼륨이 지원되지만 프로덕션 클러스터에서 이미지 레지스트리와 함께 사용하는 것은 권장되지 않습니다. 레지스트리가 블록 스토리지에 구성된 설치는 레지스트리가 둘 이상의 복제본을 가질 수 없기 때문에 가용성이 높지 않습니다.

프로세스

1. 이미지 레지스트리 스토리지를 블록 스토리지 유형으로 설정하려면 **Recreate** 롤아웃 전략을 사용하고 복제본 **1**개 만으로 실행되도록 레지스트리를 패치합니다.

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. 블록 스토리지 장치에 **PV**를 프로비저닝하고 해당 볼륨의 **PVC**를 생성합니다. 요청된 블록 볼륨은 **RWO(ReadWriteOnce)** 액세스 모드를 사용합니다.

   a. **VMware vSphere PersistentVolumeClaim** 개체를 정의하려면 다음 내용이 포함된 **pvc.yaml** 파일을 생성합니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage 1
  namespace: openshift-image-registry 2
spec:
  accessModes:
  - ReadWriteOnce 3
  resources:
    requests:
      storage: 100Gi 4
```

1. **PersistentVolumeClaim** 개체를 표시하는 고유한 이름입니다.

2. **PersistentVolumeClaim** 오브젝트의 네임스페이스로 **openshift-image-registry**입니다.

3. 영구 볼륨 클레임의 액세스 모드입니다. **ReadWriteOnce**를 사용하면 단일 노드에서 읽기 및 쓰기 권한으로 볼륨을 마운트할 수 있습니다.

4

영구 볼륨 클레임의 크기입니다.

b. 파일에서 **PersistentVolumeClaim** 오브젝트를 만듭니다.

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 올바른 **PVC**를 참조하도록 레지스트리 설정을 편집합니다.

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

출력 예

```
storage:
  pvc:
    claim: 1
```

1

사용자 지정 **PVC**를 만들면 **image-registry-storage PVC**의 기본 자동 생성을 위해 **claim** 필드를 비워둘 수 있습니다.

올바른 **PVC**를 참조하도록 레지스트리 스토리지를 구성하는 방법은 **vSphere** 용 레지스트리 구성을 참조하십시오.

18.2.14. VMware vSphere 볼륨 백업

OpenShift Container Platform은 새 볼륨을 독립 영구 디스크로 프로비저닝하여 클러스터의 모든 노드에서 볼륨을 자유롭게 연결 및 분리합니다. 결과적으로 스냅샷을 사용하는 볼륨을 백업하거나 스냅샷에서 볼륨을 복원할 수 없습니다. 자세한 내용은 스냅 샷 제한 을 참조하십시오.

프로세스

영구 볼륨의 백업을 생성하려면 다음을 수행합니다.

1. 영구 볼륨을 사용중인 애플리케이션을 중지합니다.
2. 영구 볼륨을 복제합니다.
3. 애플리케이션을 다시 시작합니다.
4. 복제된 볼륨의 백업을 만듭니다.
5. 복제된 볼륨을 삭제합니다.

18.2.15. 스틸 클럭 계정

기본적으로 설치 프로그램은 스틸 클럭 회계 매개 변수 (**stealclock.enabled**)를 활성화하지 않고 클러스터의 가상 머신을 프로비저닝합니다. 스틸링 클럭 계정을 사용하면 클러스터 문제 해결에 도움이 될 수 있습니다. 클러스터가 배포된 후 **vSphere Client**를 사용하여 각 가상 머신에서 이 매개변수를 활성화합니다.

자세한 내용은 이 **Red Hat** 기술 자료 문서 를 참조하십시오.

18.2.16. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 **Telemetry** 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 **Telemetry**가 자동으로 실행되고 **OpenShift Cluster Manager** 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 **OpenShift Cluster Manager**를 사용하여 자동으로 또는 **OpenShift Cluster Manager**를 사용하여 수동으로 유지 관리되는지 확인한 후 **subscription watch**를 사용하여 계정 또는 다중 클러스터 수준에서 **OpenShift Container Platform** 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

18.2.17. 다음 단계

- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.
- 레지스트리를 설정하고 레지스트리 스토리지를 구성합니다.
- 선택 사항: vSphere Problem Detector Operator에서 이벤트를 보고 클러스터에 권한 또는 스토리지 구성 문제가 있는지 확인합니다.

18.3. 사용자 지정 설정으로 VMC에 클러스터 설치

OpenShift Container Platform 4.9 버전에서는 AWS의 VMC (VMware Cloud)에 배포하여 설치 관리자 프로비저닝 인프라를 사용하는 VMware vSphere 인스턴스에 클러스터를 설치할 수 있습니다.

OpenShift Container Platform 배포를 위해 VMC 환경을 구성한 후 VMC 환경에 공동 배치된 bastion 관리 호스트에서 OpenShift Container Platform 설치 프로그램을 사용합니다. 설치 프로그램 및 컨트롤 플레인은 OpenShift Container Platform 클러스터에 필요한 리소스 배포 및 관리 프로세스를 자동화합니다.

OpenShift Container Platform 설치를 사용자 지정하려면 클러스터를 설치하기 전에 install-config.yaml 파일에서 매개변수를 수정합니다.

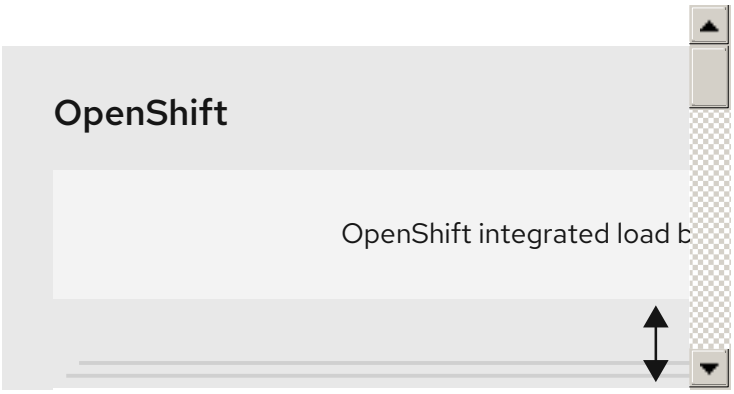
참고

OpenShift Container Platform은 단일 VMware vCenter에만 클러스터 배포를 지원합니다. 여러 vCenter에 머신/머신 세트가 있는 클러스터 배포는 지원되지 않습니다.

18.3.1. vSphere용 VMC 설정

AWS 호스팅된 vSphere 클러스터에 VMware Cloud (VMC)의 OpenShift Container Platform을 설치하여 하이브리드 클라우드 전반에 애플리케이션을 온프레미스 및 오프프레미스 모두 배포 및 관리할 수

있습니다.



VMware vSphere에 OpenShift Container Platform을 설치하기 전에 VMC 환경에서 여러 옵션을 구성해야 합니다. VMC 환경에 다음과 같은 사전 요구 사항이 있는지 확인합니다.

- 비독점 DHCP 사용, NSX-T 네트워크 세그먼트 및 서브넷을 생성합니다. 서브넷에서 다른 VM(가상 머신)을 호스팅할 수 있지만 OpenShift Container Platform 배포에서는 8개 이상의 IP 주소를 사용할 수 있어야 합니다.
- DHCP 범위 외부에서 두 개의 IP 주소를 할당하고 역방향 DNS 레코드로 구성합니다.
  - 할당된 IP 주소를 가리키는 api.<cluster_name>.<base_domain>의 DNS 레코드입니다.
  - 할당된 IP 주소를 가리키는 *.apps.<cluster_name>.<base_domain>의 DNS 레코드입니다.
- 다음 방화벽 규칙을 설정합니다.
  - OpenShift Container Platform 컴퓨팅 네트워크와 인터넷 간의 ANY:ANY 방화벽 규칙입니다. 이는 노드와 애플리케이션에서 컨테이너 이미지를 다운로드하는 데 사용됩니다.
  - 포트 443의 설치 호스트와 소프트웨어 정의 데이터 센터(SDDC) 관리 네트워크 간의 ANY:ANY 방화벽 규칙입니다. 이를 통해 배포 중에 RHCOS(Red Hat Enterprise Linux CoreOS) OVA를 업로드할 수 있습니다.
  - OpenShift Container Platform 컴퓨팅 네트워크와 인터넷 간의 HTTPS 방화벽 규칙입

니다. 이 연결을 통해 OpenShift Container Platform은 노드, PVC(영구 볼륨 클레임) 및 기타 리소스를 프로비저닝 및 관리하기 위해 vCenter과 통신할 수 있습니다.

- OpenShift Container Platform을 배포하려면 다음 정보가 있어야 합니다.
  - OpenShift Container Platform 클러스터에 vmc-prod-1로 로그인합니다.
  - companyname.com과 같은 기본 DNS 이름입니다.
  - 기본값을 사용하지 않는 경우 Pod 네트워크 CIDR 및 서비스 네트워크 CIDR을 식별해야 하며, 기본적으로 10.128.0.0/14 및 172.30.0.0/16으로 설정됩니다. 이러한 CIDR은 pod-to-pod 및 pod-to-service 통신에 사용되며 외부에서 액세스할 수 없지만 조직의 기존 서브넷과 중복되어서는 안 됩니다.
  - 다음 vCenter 정보를 참조하십시오:
    - vCenter 호스트 이름, 사용자 이름 및 암호
    - 데이터 센터 이름 (예: SDDC-Datacenter)
    - Cluster-1과 같은 클러스터 이름
    - 네트워크 이름
    - WorkloadDatastore와 같은 데이터 저장소 이름

참고

클러스터 설치가 완료된 후 vSphere 클러스터를 VMC Compute-ResourcePool 리소스 풀로 이동하는 것이 좋습니다.

- VMC에 bastion으로 배포된 Linux 기반 호스트입니다.
  - bastion 호스트는 RHEL(Red Hat Enterprise Linux) 또는 기타 Linux 기반 호스트일 수 있습니다. 인터넷에 연결되어 있어야 하고 OVA를 ESXi 호스트에 업로드하는 기능이 있어야 합니다.
  - OpenShift CLI 툴을 bastion 호스트에 다운로드하여 설치합니다.
    - openshift-install 설치 프로그램
    - OpenShift CLI(oc) 툴

참고

Kubernetes 용 VMware NCP (NSX Conrainer Plugin)dOpen에는 사용할 수 없으며 NSX는 OpenShift SDN으로 사용되지 않습니다. 현재 VMC에서 사용할 수 있는 NSX 버전은 OpenShift Container Platform에서 인증된 NCP 버전과 호환되지 않습니다.

그러나 NSX DHCP 서비스는 전체 스택 자동화된 OpenShift Container Platform 배포와 vSphere와의 Machine API 통합을 통해 수동 또는 자동으로 프로비저닝된 노드를 사용하여 가상 머신 IP 관리에 사용됩니다. 또한 OpenShift Container Platform 클러스터와 bastion 호스트와 VMC vSphere 호스트 사이에서 액세스할 수 있도록 NSX 방화벽 규칙이 생성됩니다.

18.3.1.1. VMC Sizer 툴

AWS의 VMware Cloud는 AWS 베어메탈 인프라 상단에 구축됩니다. 이는 AWS 네이티브 서비스를 실행하는 것과 동일한 베어 메탈 인프라입니다. AWS 소프트웨어 정의 데이터 센터(SDDC)의 VMware 클라우드가 배포되면 이러한 물리적 서버 노드를 사용하고 단일 테넌트 방식으로 VMware ESXi 하이퍼바이저를 실행할 수 있습니다. 즉, VMC를 사용하는 모든 사용자가 물리적 인프라에 액세스할 수 없습니다. 가상 인프라를 호스팅하는데 필요한 물리적 호스트 수를 고려해야 합니다.

이를 확인하기 위해 VMware는 AWS Sizer에서 VMC를 제공합니다. 이 툴을 사용하면 VMC에서 호스팅할 리소스를 정의할 수 있습니다.

- 워크로드 유형

- 총 가상 머신 수
- 사양 정보는 다음과 같습니다:
  - 스토리지 요구사항
  - **vCPU**
  - **vRAM**
  - 오버커밋 비율

이러한 세부 정보를 사용하여 **VMware** 모범 사례를 기반으로 보고서를 생성하고 클러스터 구성과 필요한 호스트 수를 권장합니다.

18.3.2. vSphere 사전 요구 사항

- **OpenShift Container Platform** 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.
- 블록 레지스트리 스토리지가 프로비저닝되어 있습니다. 자세한 내용은 영구 저장 장치 이해를 참조하십시오.
- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성해야 합니다.

참고

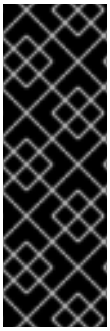
프록시를 구성하는 경우에도 해당 사이트 목록을 검토하십시오.

18.3.3. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 **Telemetry** 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.



중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

18.3.4. VMware vSphere 인프라 요구사항

사용하는 구성 요소의 요구사항을 충족하는 **VMware vSphere** 버전 **6** 또는 **7** 인스턴스에 **OpenShift Container Platform** 클러스터를 설치해야 합니다.

표 **18.9. vSphere** 가상 환경의 버전 요구 사항

가상 환경 제품	필요한 버전
VM 하드웨어 버전	13 이상
vSphere ESXi hosts	6.5 이상
vCenter 호스트	6.5 이상

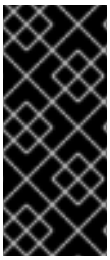
중요

이제 VMware vSphere 버전 6.7U2 또는 이전 버전에 클러스터를 설치하는 것이 더 이상 사용되지 않으며 가상 하드웨어 버전 13은 더 이상 사용되지 않습니다. 이러한 버전은 여전히 완전히 지원되지만 향후 OpenShift Container Platform 버전에서는 지원이 제거됩니다. 이제 OpenShift Container Platform의 vSphere 가상 머신의 하드웨어 버전 15가 기본값이 되었습니다. vSphere 노드의 하드웨어 버전을 업데이트하려면 " vSphere에서 실행되는 노드에서 하드웨어 업데이트" 문서를 참조하십시오.

표 18.10. VMware 구성 요소에 지원되는 최소 vSphere 버전

구성 요소	지원되는 최소 버전	설명
하이퍼바이저	HW 버전 13이 포함된 vSphere 6.5 이상	이 버전은 RHCOS(Red Hat Enterprise Linux CoreOS)가 지원하는 최소 버전입니다. Red Hat Enterprise Linux 8 지원 하이퍼바이저 목록을 참조하십시오.
In-tree 드라이버가 있는 스토리지	vSphere 6.5 이상	이 플러그인은 OpenShift Container Platform에 포함된 vSphere용 인트리 스토리지 드라이버를 사용하여 vSphere 스토리지를 생성합니다.

vSphere 버전 6.5 인스턴스를 사용하는 경우 OpenShift Container Platform을 설치하기 전에 6.7U3 또는 7.0로 업그레이드하는 것이 좋습니다.



중요

OpenShift Container Platform을 설치하기 전에 ESXi 호스트의 시간이 동기화되었는지 확인해야 합니다. VMware 문서에서 Edit Time Configuration for a Host를 참조하십시오.

18.3.5. 네트워크 연결 요구사항

OpenShift Container Platform 클러스터 구성 요소가 통신할 수 있도록 시스템 간 네트워크 연결을 구성해야 합니다.

필요한 네트워크 포트에 대한 다음 세부 정보를 검토합니다.

표 18.11. 모든 시스템 간 통신에 사용되는 포트

프로토콜	포트	설명
ICMP	해당 없음	네트워크 연결성 테스트
TCP	**1936**	메트릭
	9000-9999	**9100-9101** 포트의 노드 내보내기 및 **9099** 포트의 Cluster Version Operator를 포함한 호스트 수준 서비스.
	10250-10259	Kubernetes에서 예약하는 기본 포트
	10256	openshift-sdn
UDP	**4789**	VXLAN(Virtual extensible LAN)
	6081	Geneve
	9000-9999	**9100-9101** 포트의 노드 내보내기를 포함한 호스트 수준 서비스.
	500	IPsec IKE 패킷
	4500	IPsec NAT-T 패킷
TCP/UDP	**30000-32767**	Kubernetes 노드 포트
ESP	해당 없음	IPsec Encapsulating Security Payload (ESP)

표 18.12. 모든 시스템과 컨트롤 플레인 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**6443**	Kubernetes API

표 18.13. 컨트롤 플레인 머신 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**2379-2380**	etcd 서버 및 피어 포트

추가 리소스

- **vSphere 노드의 하드웨어 버전을 업데이트하려면 vSphere에서 실행 중인 노드에서 하드웨어 업데이트를 참조하십시오.**

18.3.6. vCenter 요구사항

설치 관리자가 프로비저닝하는 인프라를 사용하는 **vCenter**에 **OpenShift Container Platform** 클러스터를 설치하기 전에 환경을 준비해야 합니다.

필요한 **vCenter** 계정 권한

vCenter에 **OpenShift Container Platform** 클러스터를 설치하려면 설치 프로그램에서 필요한 리소스를 읽고 생성할 수 있는 권한이 있는 계정에 액세스해야 합니다. 필요한 모든 권한에 액세스할 수 있는 가장 간단한 방법은 글로벌 관리 권한이 있는 계정을 사용하는 것입니다.

글로벌 관리 권한이 있는 계정을 사용할 수 없는 경우 **OpenShift Container Platform** 클러스터 설치에 필요한 권한을 부여하려면 역할을 생성해야 합니다. 대부분의 권한이 항상 필요하지만 기본 동작인 **vCenter** 인스턴스에 **OpenShift Container Platform** 클러스터를 포함하는 폴더를 프로비저닝할 설치 프로그램인 경우에만 필요합니다. 필요한 권한을 부여하려면 지정된 오브젝트에 대해 **vSphere** 역할을 생성하거나 수정해야 합니다.

설치 프로그램이 **vSphere** 가상 머신 폴더를 생성하는 경우 추가 역할이 필요합니다.

예 18.4. vSphere API에 설치에 필요한 역할 및 권한

역할의 vSphere 개체	필요한 경우	vSphere API에서 필요한 권한
vSphere vCenter	Always	CNS.**Searchable** **InventoryService.Tagging.AttachTag** **InventoryService.Tagging.CreateCategory** **InventoryService.Tagging.CreateTag** **InventoryService.Tagging.Delete**Tag **InventoryService.tagged.DeleteTag** **InventoryService.Tagging.EditCategory** **InventoryService.Tagging.EditTag**ECDHE**Session** **ValidateSession**Forwarded**StorageProfile.Update**Forwarded**StorageProfile.View**

역할의 vSphere 개체	필요한 경우	vSphere API에서 필요한 권한
vSphere vCenter Cluster	Always	**Host.Config.Storage** **Resource.AssignVMToPool** **VApp.AssignResourcePool** **VApp.Import** **VirtualMachine.Config.Add NewDisk**
vSphere Datastore	Always	**Datastore.AllocateSpace** **Datastore.LoadBalancer** **Datastore.FileManagement** **InventoryService.Tagging.O bjectAttachable**
vSphere Port Group	Always	**Network.Assign**

역할의 vSphere 개체	필요한 경우	vSphere API에서 필요한 권한
가상 머신 폴더	Always	**InventoryService.ume.ObjectAttachable Resource.AssignVMToPool VApp.Import VirtualMachine.Config.Add ExistingDisk VirtualMachine.Config.Add NewDisk VirtualMachine.Config.Add RemoveDevice VirtualMachine.Config.Advanced Config VirtualMachine.Config.Anno tationECDHEVirtualMachine .Config.DiskExtend**Forwarde d VirtualMachine**.Config.**DiskExte nd Forwarded**VirtualMachine.Co nfig.DiskLeaseVirtualMachi ne.Config.ReCreate**VolumeCl aim.Config**.MemoryDriver.**

역할의 vSphere 개체	필요한 경우	vSphere API에서 필요한 권한
vSphere vCenter Datacenter	설치 프로그램이 가상 머신 폴더를 생성하는 경우	**InventoryService.Tagging.ObjectAttachable Resource.AssignVMToPool VApp.Import VirtualMachine.Config.Add ExistingDisk VirtualMachine.Config.Add NewDisk VirtualMachine.Config.Add RemoveDevice VirtualMachine.Config.Advanced Config VirtualMachine.Config.Anno tationECDHEVirtualMachine .Config.DiskExtend**Forwarde d VirtualMachine**.**Config**.DiskExt end ForwardedVirtualMachine.C onfig.DiskLeaseVirtualMach ine.Config.ReCreate VolumeClaim.Config.Reset**d efined**.**

예 18.5. vCenter 그래픽 사용자 인터페이스(GUI)에 설치하는 데 필요한 역할 및 권한

역할의 vSphere 개체	필요한 경우	vCenter GUI에서 필요한 권한
vSphere vCenter	Always	CNS **.Searchable "vSphere agged""Assign or unassign vSphere Tag" "vSphere tagging" "vSphere" "vSphere" "vSphere"CreatevSphere Tag" vSphere 태그""vSphere" 삭제 태그 지정" "vSphere Assignment""Edit vSphere Tag" "vSphere Tagging" "vSphere 태그" 세션" "Profile 기반 스토리지""Profile 기반 스토리지 업데이트""Profile 기반 스토리지"Profile 중심 스토리지 업데이트"Profile 기반 스토리지 업데이트"Profile 기반 스토리지 업데이트 스토리지 보기"**
vSphere vCenter Cluster	클러스터 루트에서 VM이 생성되는 경우	**Host.Configuration"Storage partition configuration" 리소스" 리소스 풀에 가상 머신을 서명" VApp"Assign resource pool" VApp.Import"Virtual machine""Change Configuration"."새 디스크 추가"**
vSphere vCenter 리소스 풀	기존 리소스 풀이 제공되는 경우	**Host.Configuration"Storage partition configuration" 리소스" 리소스 풀에 가상 머신을 서명" VApp"Assign resource pool" VApp.Import"Virtual machine""Change Configuration"."새 디스크 추가"**

역할의 vSphere 개체	필요한 경우	vCenter GUI에서 필요한 권한
vSphere Datastore	Always	**Datastore"Allocate space"** 데이터**저장소"** **데이터 저장소"** **Datastore"Low level file operations""vSphere Assignment"Assign or Unassign vSphere Tag on Object"**
vSphere Port Group	Always	**네트워크 "Assign network"**
가상 머신 폴더	Always	**"vSphere Assignment""Assign or unassign vSphere Tag on Object"** **Resource"Assign virtual machine to resource pool"**: **VApp.Import** **"Virtual machine""Change Configuration","Add existing disk""**Add existing disk**""Add new disk""Add new disk" "Virtual""** **머신 "시스템 변경", "가상 머신""** **"가상 머신""시스템 변경"** **"가상 머신""변경 구성",** **"가상 머신", 변경, CPU 수 변경"** **"가상 머신", 가상 디스크 변경", 가상 디스크 변경"** **"가상머신". 머신 변경 "구성 변경"** **"가상 머신""마이크 구성 변경"** **"가상 머신""마이크 구성 변경" "구성 변경",** **"가상 머신", 변경" 디스크 삭제"** **"가상 머신".변경 구성".Rename** **"가상 머신 변경" VIX API**로가상 머신 변경**"Gregradevirtual machine"Guestoperating system management by VIX API**

역할의 vSphere 개체	필요한 경우	vCenter GUI에서 필요한 권한
		"Virtual machine". Interconnection"Power on" "Virtual machine". interactionion.Reset "Virtual machine"."Edit Inventory"."Create new" "Virtual machine"."Create existing" "Virtual machine"Edit Inventory""Create from existing" "Virtual machine"Edit Inventory""Create from existing" virtual machine".Edit Inventory""Create from existing" virtual machine".Edit Inventory""Create from existing""Virtual machine" machine".Provisioning."Clone 가상 머신" "가상 머신".Provisioning." "가상 머신".Provisioning."Deploy 템플릿"
vSphere vCenter Datacenter	설치 프로그램이 가상 머신 폴더를 생성하는 경우	**"vSphere Assignment""Assign or unassign vSphere Tag on Object" Resource"Assign virtual machine to resource pool"**: **VApp.Import "Virtual machine""Change Configuration","Add existing disk""**Add existing disk""**Add new disk""Add new disk" "Virtual""** **머신 "시스템 변경", "가상 머신"" "가상 머신""시스템 변경" "가상 머신""변경 구성", "가상 머신", 변경, CPU 수 변경" "가상 머신", 가상 디스크 변경", 가상 디스크 변경" "가상머신". 머신 변경 "구성 변경" "가상 머신""마이크 구성 변경" "가상 머신""마이크 구성 변경" "구성 변경",**

역할의 vSphere 개체	필요한 경우	"가상 머신", 변경" 디스크 삭제 vCenter GUI에서 필요한 권한
		"가상 머신".변경 구성".Rename "가상 머신 변경" VIX API로가상 머신 변경"Gregradevirtual machine"Guestoperating system management by VIX API machine". interaction" "Virtual machine". Interconnection"Power on" "Virtual machine". interactionion.Reset "Virtual machine"."Edit Inventory"."Create new" "Virtual machine"."Create existing" "Virtual machine"Edit Inventory""Create from existing" "Virtual machine"Edit Inventory""Create from existing" virtual machine".Edit Inventory""Create from existing" virtual machine".Edit Inventory""Create from existing""Virtual machine" machine".Provisioning."Clone 가상 머신" "가상 머신".Provisioning."Deploy 템플릿" "가상 머신".Provisioning" 폴더 만들기" 폴더 생성"폴더 삭제"

또한 사용자에게 일부 **ReadOnly** 권한이 필요하며 일부 역할에는 하위 오브젝트에 권한을 부여하는 사용 권한이 필요합니다. 이러한 설정은 클러스터를 기존 폴더에 설치할지 여부에 따라 달라집니다.

예 18.6. 필수 권한 및 권한 부여 설정

vSphere 오브젝트	폴더 유형	하위 항목으로 권한 부여	권한 필요
vSphere vCenter	Always	False	나열된 필수 권한
vSphere vCenter Datacenter	기존 폴더	False	**ReadOnly** 권한
	설치 프로그램은 폴더를 생성	True	나열된 필수 권한
vSphere vCenter Cluster	Always	True	나열된 필수 권한
vSphere vCenter Datastore	Always	False	나열된 필수 권한
vSphere Switch	Always	False	**ReadOnly** 권한
vSphere Port Group	Always	False	나열된 필수 권한
vSphere vCenter Virtual Machine Folder	기존 폴더	True	나열된 필수 권한

필요한 권한만으로 계정을 생성하는 방법에 대한 자세한 내용은 **vSphere** 문서에서 **vSphere 권한 및 사용자 관리 작업**을 참조하십시오.

vMotion으로 OpenShift Container Platform 사용

vSphere 환경에서 **vMotion**을 사용하려는 경우 **OpenShift Container Platform** 클러스터를 설치하기 전에 다음을 고려하십시오.

- **OpenShift Container Platform**은 일반적으로 **compute-only vMotion** 을 지원합니다. 스토리지 **vMotion**을 사용하면 문제가 발생할 수 있으며 지원되지 않습니다.

  컴퓨팅 및 컨트롤 플레인 노드의 가동 시간을 보장하기 위해 **vMotion**에 대한 **VMware** 모범 사례를 따르는 것이 좋습니다. 또한 **VMware** 유사성 방지 규칙을 사용하여 유지 관리 또는 하드웨어 문제 중에 **OpenShift Container Platform**의 가용성을 개선하는 것이 좋습니다.

  vMotion 및 유사성 방지 규칙에 대한 자세한 내용은 **vMotion 네트워킹 요구 사항** 및 **VM 유사성 방지 규칙에** 대한 **VMware vSphere** 설명서를 참조하십시오.

- **Pod**에서 **vSphere** 볼륨을 사용하는 경우 데이터 저장소를 통해 **VM**을 마이그레이션하면 수

동으로 또는 스토리지 **vMotion**을 통해 **VM**을 마이그레이션하면 **OpenShift Container Platform PV(**영구 볼륨**)** 오브젝트 내에서 잘못된 참조가 표시됩니다**.** 이러한 참조는 영향을 받는 포드가 시작되지 않게 하고 데이터 손실을 초래할 수 있습니다**.**

- 마찬가지로 **OpenShift Container Platform**은 데이터 저장소에서 **VMDK**의 선택적 마이그레이션을 지원하지 않으며 **VM** 프로비저닝 또는 **PV**의 동적 또는 정적 프로비저닝을 위해 데이터 저장소 클러스터의 일부 또는 **PV**의 동적 또는 정적 프로비저닝을 위해 데이터 저장소 클러스터의 일부인 데이터 저장소를 사용하는 경우도 있습니다**.**

클러스터 리소스

설치 관리자 프로비저닝 인프라를 사용하는 **OpenShift Container Platform** 클러스터를 배포할 때는 설치 프로그램이 **vCenter** 인스턴스에서 여러 리소스를 생성할 수 있어야 합니다**.**

표준 **OpenShift Container Platform**을 설치하면 다음 **vCenter** 리소스가 생성됩니다**.**

- 폴더 한 개
- 태그 카테고리 한 개
- 태그 한 개
- 가상 머신**:**
  - 템플릿 한 개
  - 임시 부트스트랩 노드 한 개
  - 컨트롤 플레인 노드 세 개
  - 컴퓨팅 시스템 세 개

이러한 리소스는 **856GB**의 스토리지를 사용하지만 부트스트랩 노드는 클러스터 설치 프로세스 중에 제거됩니다**.** 표준 클러스터를 사용하려면 최소 **800GB**의 스토리지가 필요합니다**.**

컴퓨팅 시스템을 더 많이 배포할수록 **OpenShift Container Platform** 클러스터는 더 많은 스토리지를 사용합니다.

클러스터 제한

사용 가능한 리소스는 클러스터마다 다릅니다. **vCenter** 내에서 가능한 클러스터 수는 주로 사용 가능한 스토리지 공간과 필요한 리소스 수에 대한 제한으로 제한됩니다. 클러스터가 생성하는 **vCenter** 리소스와 **IP** 주소 및 네트워크와 같이 클러스터를 배포하는 데 필요한 리소스에 대한 제한 사항을 모두 고려해야 합니다.

네트워킹 요구사항

네트워크에 **DHCP**를 사용하고 **DHCP** 서버가 클러스터 시스템에 영구 **IP** 주소를 제공하도록 구성되어 있는지 확인합니다. **DHCP** 서버를 사용하도록 기본 게이트웨이를 구성해야 합니다. 모든 노드는 동일한 **VLAN**에 있어야 합니다. 두 번째 **VLAN**을 **Day 2** 작업으로 사용하여 클러스터를 확장할 수 없습니다. 또한 **OpenShift Container Platform** 클러스터를 설치하기 전에 다음 네트워킹 리소스를 생성해야 합니다.

참고

클러스터의 각 **OpenShift Container Platform** 노드는 **DHCP**를 통해 검색할 수 있는 **NTP(Network Time Protocol)** 서버에 액세스할 수 있어야 합니다. **NTP** 서버 없이도 설치할 수 있습니다. 그러나 비동기 서버 클록으로 인해 **NTP** 서버가 차단하는 오류가 발생합니다.

필요한 IP 주소

설치 관리자 프로비저닝 **vSphere** 설치에는 다음 두 개의 고정 **IP** 주소가 필요합니다.

- **API** 주소는 클러스터 **API**에 액세스하는 데 사용됩니다.
- **Ingress** 주소는 클러스터 인그레스 트래픽에 사용됩니다.

OpenShift Container Platform 클러스터를 설치할 때 설치 프로그램에 이 **IP** 주소를 제공해야 합니다.

DNS 레코드

OpenShift Container Platform 클러스터를 호스팅하는 **vCenter** 인스턴스에 적절한 **DNS** 서버에서 고정 **IP** 주소 두 개의 **DNS** 레코드를 생성해야 합니다. 각 레코드에서 **<cluster_name>**은 클러스터 이름

이고 <base_domain>은 클러스터를 설치할 때 지정하는 클러스터 기본 도메인입니다. 전체 DNS 레코드는 <component>.<cluster_name>.<base_domain> 형식입니다.

표 18.14. 필수 DNS 레코드

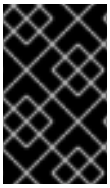
구성 요소	레코드	설명
API VIP	api.<cluster_name>.<base_domain>.	이 DNS A/AAAA 또는 CNAME 레코드는 컨트롤 플레인 시스템의 로드 밸런서를 가리켜야 합니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.
Ingress VIP	*.apps.<cluster_name>.<base_domain>.	기본적으로 작업자 노드인 인그레스 라우터 Pod를 실행하는 시스템을 대상으로 하는 로드 밸런서를 가리키는 와일드카드 DNS A/AAAA 또는 CNAME 레코드입니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.

18.3.7. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 core 사용자의 ~/.ssh/authorized_keys 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 core로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. ./openshift-install gather 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.



중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

참고

AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

절차

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: ~/.ssh/id_ed25519)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 '~/.ssh 디렉터리에 있는지 확인하십시오.

참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 ed25519 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 rsa 또는 ecdsa 알고리즘을 사용하는 키를 생성합니다.

2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 ~/.ssh/id_ed25519.pub 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 ./openshift-install gather 명령을 사용하려는 경우 필요합니다.

참고

일부 배포에서는 ~/.ssh/id_rsa 및 ~/.ssh/id_dsa와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

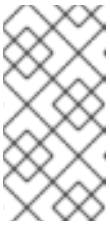
a.

ssh-agent 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```



참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4.

ssh-agent에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1

SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: ~/.ssh/id_ed25519).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- **OpenShift Container Platform**을 설치할 때 **SSH** 공개 키를 설치 프로그램에 지정합니다.

18.3.8. 설치 프로그램 받기

OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

- **500MB**의 로컬 디스크 공간이 있는 **Linux** 또는 **macOS**를 실행하는 컴퓨터가 있습니다.

절차

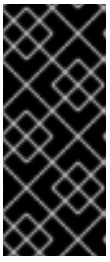
1. **OpenShift Cluster Manager** 사이트의 인프라 공급자 페이지에 액세스합니다. **Red Hat** 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.

2. 인프라 공급자를 선택합니다.

3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.

중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.



중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 **OpenShift Container Platform** 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 **Linux** 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

18.3.9. 시스템 신뢰에 vCenter 루트 CA 인증서 추가

설치 프로그램이 vCenter API에 액세스해야 하므로 OpenShift Container Platform 클러스터를 설치하기 전에 vCenter의 신뢰할 수 있는 루트 CA 인증서를 시스템 신뢰에 추가해야 합니다.

절차

1. vCenter 홈 페이지에서 vCenter의 루트 CA 인증서를 다운로드합니다. vSphere Web Services SDK 섹션에서 Download trusted root CA certificates를 클릭합니다. <vCenter>/certs/download.zip 파일이 다운로드됩니다.

2. vCenter 루트 CA 인증서가 포함된 압축 파일의 압축을 풉니다. 압축 파일의 내용은 다음 파일 구조와 유사합니다.

```
certs
├── lin
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
├── mac
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
└── win
    ├── 108f4d17.0.crt
    ├── 108f4d17.r1.crl
    ├── 7e757f6a.0.crt
    ├── 8e4f8471.0.crt
    └── 8e4f8471.r0.crl

3 directories, 15 files
```

3. 운영 체제 파일을 시스템 신뢰에 추가합니다. 예를 들어 Fedora 운영 체제에서는 다음 명령을 실행합니다.

```
# cp certs/lin/* /etc/pki/ca-trust/source/anchors
```

4. 시스템 신뢰를 업데이트합니다. 예를 들어 Fedora 운영 체제에서는 다음 명령을 실행합니다.

```
# update-ca-trust extract
```

18.3.10. 설치 구성 파일 만들기

VMware vSphere에 설치하는 OpenShift Container Platform 클러스터를 사용자 지정할 수 있습니다.

사전 요구 사항

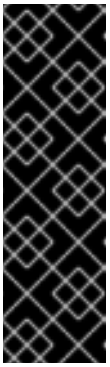
- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.
- 서브스크립션 수준에서 서비스 주체 권한을 획득합니다.

절차

1. install-config.yaml 파일을 생성합니다.
   a. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 <installation_directory>는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.



중요

비어 있는 디렉터리를 지정합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

b.

화면에 나타나는 지시에 따라 클라우드에 대한 구성 세부 사항을 입력합니다.

i.

선택사항: 클러스터 시스템에 액세스하는 데 사용할 SSH 키를 선택합니다.

참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 ssh-agent 프로세스가 사용하는 SSH 키를 지정합니다.

ii.

대상 플랫폼으로 vsphere를 선택합니다.

iii.

vCenter 인스턴스의 이름을 지정합니다.

iv.

클러스터를 생성하는 데 필요한 권한이 있는 vCenter 계정의 사용자 이름과 암호를 지정합니다.

설치 프로그램이 vCenter 인스턴스에 연결됩니다.

v.

연결할 vCenter 인스턴스에서 데이터 센터를 선택합니다.

vi.

사용할 기본 vCenter 데이터 저장소를 선택합니다.

vii.

OpenShift Container Platform 클러스터를 설치할 vCenter 클러스터를 선택합니다. 설치 프로그램은 vSphere 클러스터의 루트 리소스 풀을 기본 리소스 풀로 사용합니다.

viii.

vCenter 인스턴스에서 구성한 가상 IP 주소 및 DNS 레코드가 포함된 네트워크를 선택합니다.

ix.

컨트롤 플레인 API 액세스를 위해 구성한 가상 IP 주소를 입력합니다.

x.

x.
클러스터 인그레스용으로 구성한 가상 **IP** 주소를 입력합니다.

xi.
기본 도메인을 입력합니다. 이 기본 도메인은 구성한 **DNS** 레코드에서 사용한 것과 동일해야 합니다.

xii.
클러스터를 설명할 수 있는 이름을 입력합니다. 클러스터 이름은 구성한 **DNS** 레코드에서 사용한 것과 동일해야 합니다.

xiii.
Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 붙여넣습니다.

2.
install-config.yaml 파일을 수정합니다. 사용 가능한 매개변수에 대한 자세한 정보는 “설치 구성 매개변수” 섹션에서 확인할 수 있습니다.

3.
여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.

중요

install-config.yaml 파일은 설치 과정에서 사용됩니다. 이 파일을 재사용하려면 지금 백업해야 합니다.

18.3.10.1. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 매개변수 값을 제공하여 클러스터를 호스팅할 클라우드 플랫폼에서 사용자 계정을 설명하고 선택사항으로 클러스터의 플랫폼을 사용자 지정합니다. **install-config.yaml** 설치 구성 파일을 생성할 때 명령줄을 통해 필요한 매개변수 값을 제공합니다. 클러스터를 사용자 지정하면 **install-config.yaml** 파일을 수정하여 플랫폼에 대한 세부 정보를 제공할 수 있습니다.

참고

설치한 후에는 **install-config.yaml** 파일에서 이러한 매개변수를 수정할 수 없습니다.

중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

18.3.10.1.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 18.15. 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	**dev** 와 같은 소문자 및 하이픈(**-**)의 문자열입니다.

매개변수	설명	값
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

18.3.10.1.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 18.16. 네트워크 매개변수

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체 **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.

매개변수	설명	값
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.
networking.clusterNetwork	Pod의 IP 주소 블록입니다. 기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된 **cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16

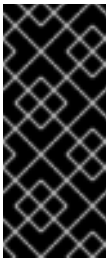
매개변수	설명	값
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16**  **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

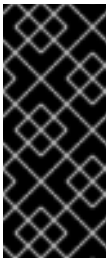
18.3.10.1.3. 선택적 구성 매개변수

선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 18.17. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열

매개변수	설명	값
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은 **3**입니다.
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열

매개변수	설명	값
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 기본값은 **External**입니다. 이 필드를 **Internal** 로 설정하는 것은 클라우드 이외의 플랫폼에서는 지원되지 않습니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

18.3.10.1.4. 추가 VMware vSphere 구성 매개변수

추가 VMware vSphere 구성 매개변수는 다음 표에 설명되어 있습니다.

표 18.18. 추가 VMware vSphere 클러스터 매개변수

매개변수	설명	값
platform.vsphere.vCenter	vCenter 서버의 정규화된 호스트 이름 또는 IP 주소입니다.	문자열
platform.vsphere.username	vCenter 인스턴스에 연결하는 데 사용할 사용자 이름입니다. 이 사용자에게는 최소한 vSphere에서 정적 또는 동적 영구 볼륨 프로비저닝 에 필요한 역할과 권한이 있어야 합니다.	문자열
platform.vsphere.password	vCenter 사용자 이름의 암호입니다.	문자열
platform.vsphere.datacenter	vCenter 인스턴스에서 사용할 데이터 센터의 이름입니다.	문자열
platform.vsphere.defaultDatastore	프로비저닝 볼륨에 사용할 기본 데이터 저장소의 이름입니다.	문자열
platform.vsphere.folder	*선택사항*입니다. 설치 프로그램이 가상 머신을 생성하는 기존 폴더의 절대 경로입니다. 이 값을 제공하지 않으면 설치 프로그램이 데이터 센터 가상 머신 폴더에 인프라 ID로 이름이 지정된 폴더를 생성합니다.	문자열(예: **/<datacenter_name>/vm/<folder_name>/<subfolder_name>**).

매개변수	설명	값
platform.vsphere.network	vCenter 인스턴스에서 구성한 가상 IP 주소 및 DNS 레코드가 포함된 네트워크입니다.	문자열
platform.vsphere.cluster	OpenShift Container Platform 클러스터를 설치할 vCenter 클러스터입니다.	문자열
platform.vsphere.apiVIP	컨트롤 플레인 API 액세스를 위해 구성한 가상 IP(VIP) 주소입니다.	IP 주소 (예: **128.0.0.1**)
platform.vsphere.ingressVIP	클러스터 인그레스용으로 구성한 가상 IP(VIP) 주소입니다.	IP 주소 (예: **128.0.0.1**)

18.3.10.1.5. 선택적 VMware vSphere 시스템 풀 구성 매개변수

다음 표에는 선택적 VMware vSphere 시스템 풀 구성 매개 변수가 설명되어 있습니다.

표 18.19. 선택적 VMware vSphere 시스템 풀 매개변수

매개변수	설명	값
platform.vsphere.clusterOSImage	설치 프로그램이 RHCOS 이미지를 다운로드하는 위치입니다. 네트워크가 제한된 환경에서 설치를 수행하려면이 매개변수를 설정해야 합니다.	HTTP 또는 HTTPS URL (선택 옵션으로 SHA-256 형식의 체크섬을 사용) 예: **https://mirror.openshift.com/images/rhcos-<version>-vmware.<architecture>.ova**.
platform.vsphere.osDisk.diskSizeGB	디스크 크기(GB)입니다.	정수
platform.vsphere.cpus	가상 머신을 할당할 총 가상 프로세서 코어 수입니다. **platform.vsphere.cpus** 의 값은 **platform.vsphere.coresPerSocket** 값의 여러 개여야 합니다.	정수
platform.vsphere.coresPerSocket	가상 머신의 소켓당 코어 수입니다. 가상 머신의 가상 소켓 수는 **platform.vsphere.cpus/platform.vsphere.coresPerSocket** 입니다. 컨트롤 플레인 노드 및 작업자 노드의 기본값은 각각 **4** 및 **2** 입니다.	정수
platform.vsphere.memoryMB	가상 머신의 메모리 크기(MB)입니다.	정수

18.3.10.2. 설치 관리자 프로비저닝 VMware vSphere 클러스터의 샘플 install-config.yaml 파일

install-config.yaml 파일을 사용자 지정하여 OpenShift Container Platform 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.

```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 3
  platform:
    vsphere: (4)
      cpus: 2
      coresPerSocket: 2
      memoryMB: 8192
      osDisk:
        diskSizeGB: 120
controlPlane: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3
  platform:
    vsphere: (7)
      cpus: 4
      coresPerSocket: 2
      memoryMB: 16384
      osDisk:
        diskSizeGB: 120
metadata:
  name: cluster (8)
platform:
  vsphere:
    vcenter: your.vcenter.server
    username: username
    password: password
    datacenter: datacenter
    defaultDatastore: datastore
    folder: folder
    network: VM_Network
    cluster: vsphere_cluster_name (9)
    apiVIP: api_vip
    ingressVIP: ingress_vip
fips: false
pullSecret: '{"auths": ...}'
sshKey: 'ssh-ed25519 AAAA...'
```

(1)

클러스터의 기본 도메인입니다. 모든 DNS 레코드는 이 기본 도메인의 하위 도메인이어야 하며 클러스터 이름을 포함해야 합니다.

2 5

controlPlane 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

3 6

동시 멀티스레딩 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. 일부 클러스터 시스템에서 동시 멀티스레딩을 비활성화할 경우에는 해당 멀티스레딩을 모든 클러스터 시스템에서 비활성화해야 합니다.

중요

동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다. 동시 멀티스레딩을 비활성화하는 경우 시스템은 8개 이상의 **CPU**와 **32GB**의 **RAM**을 사용해야 합니다.

4 7

선택사항: 컴퓨팅 및 컨트롤 플레인 시스템의 시스템 풀 매개변수에 대한 추가 구성을 제공하십시오.

8

DNS 레코드에 지정한 클러스터 이름입니다.

9

OpenShift Container Platform 클러스터를 설치할 **vSphere** 클러스터입니다. 설치 프로그램은 **vSphere** 클러스터의 루트 리소스 풀을 기본 리소스 풀로 사용합니다.

18.3.10.3. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 **HTTP** 또는 **HTTPS** 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 **OpenShift Container Platform** 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.

- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 **API**에 대한 호출을 포함하여 모든 클러스터 발신(**Egress**) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.

참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), **Google Cloud Platform (GCP)**, **Microsoft Azure** 및 **Red Hat OpenStack Platform (RHOSP)**에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

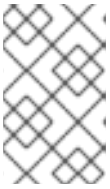
1. 클러스터 외부에서 **HTTP** 연결을 구축하는 데 사용할 프록시 **URL**입니다. **URL** 스키마는 **http**여야 합니다.

2. 클러스터 외부에서 **HTTPS** 연결을 구축하는 데 사용할 프록시 **URL**입니다.

3.

4

이 값을 제공하면 설치 프로그램에서 추가 CA 인증서를 보유할 openshift -config 네임스페이스에 user-ca- bundle 이라는 구성 맵을 생성합니다. additionalTrustBundle 및 하나 이상의 프록시 설정을 제공하는 경우 프록시 오브젝트는 trustedCA 필드의 user-ca-bundle 구성 맵을 참조하도록 구성됩니다. 그러면 Cluster Network Operator에서 trustedCA 매개변수에 대해 지정된 콘텐츠를 RHCOS 신뢰 번들과 병합하는 trusted-ca-bundle 구성 맵을 생성합니다. 프록시의 ID 인증서를 RHCOS 트러스트 번들에 있는 기관에서 서명하지 않은 경우 additionalTrustBundle 필드가 있어야 합니다.

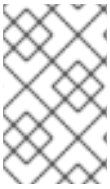


참고

설치 프로그램에서 프록시 adinessEndpoints 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 OpenShift Container Platform을 설치할 때 참조하십시오.

제공되는 install-config.yaml 파일의 프록시 설정을 사용하는 cluster라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 cluster Proxy 오브젝트는 계속 생성되지만 spec은 nil이 됩니다.



참고

cluster라는 Proxy 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

18.3.11. 클러스터 배포

호환되는 클라우드 플랫폼에 OpenShift Container Platform을 설치할 수 있습니다.

중요

최초 설치 과정에서 설치 프로그램의 create cluster 명령을 한 번만 실행할 수 있습니다.

사전 요구 사항

- 클러스터를 호스팅하는 클라우드 플랫폼으로 계정을 구성합니다.

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

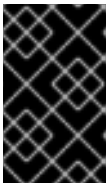
절차

1. 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터 배포를 초기화합니다.

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
  --log-level=info 2
```

1 <installation_directory> 값으로 사용자 지정한 ./install-config.yaml 파일의 위치를 지정합니다.

2 다른 설치 세부 사항을 보려면 info 대신 warn, debug 또는 error를 지정합니다.



중요

VMC 환경에서 호스팅되는 bastion의 openshift-install 명령을 사용합니다.

참고

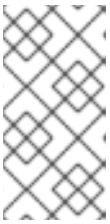
호스트에 구성된 클라우드 공급자 계정에 클러스터를 배포하기에 충분한 권한이 없는 경우, 설치 프로세스가 중단되고 누락된 권한을 알리는 메시지가 표시됩니다.

클러스터 배포가 완료되면 웹 콘솔로 연결되는 링크와 kubeadmin 사용자의 인증 정보가 포함된 클러스터 액세스 지침이 사용자 터미널에 표시됩니다.

출력 예

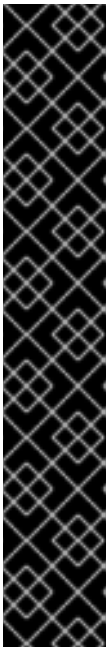
```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
```

```
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-Wt5AL"
INFO Time elapsed: 36m22s
```



참고

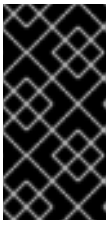
설치에 성공하면 클러스터 액세스 및 인증 정보도 **<installation_directory>/.openshift_install.log**로 출력됩니다.



중요

- 설치 프로그램에서 생성하는 **Ignition** 구성 파일에 **24**시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 **24**시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 **kubelet** 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청**(CSR)**을 수동으로 승인해야 합니다. 자세한 내용은 ***Recovering from expired control plane certificates*** 문서를 참조하십시오.

- 클러스터를 설치한 후 **24**시간에서 **22**시간까지의 인증서가 교체되기 때문에 생성된 후 **12**시간 이내에 **Ignition** 구성 파일을 사용하는 것이 좋습니다. **12**시간 이내에 **Ignition** 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.



중요

설치 프로그램에서 생성되는 파일이나 설치 프로그램을 삭제해서는 안 됩니다. 클러스터를 삭제하려면 두 가지가 모두 필요합니다.

18.3.12. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 **OpenShift Container Platform**과 상호 작용하기 위해 **OpenShift CLI(oc)**를 설치할 수 있습니다. **Linux, Windows** 또는 **macOS**에 **oc**를 설치할 수 있습니다.

중요

이전 버전의 oc를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 oc를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 Linux에서 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

절차

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.
2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. OpenShift v4.9 Linux Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. oc 바이너리를 PATH에 있는 디렉터리에 배치합니다.

   PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치

다음 절차에 따라 Windows에 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

절차

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.

2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.

3. OpenShift v4.9 Windows Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.

4. ZIP 프로그램으로 아카이브의 압축을 풉니다.

5. oc 바이너리를 PATH에 있는 디렉터리로 이동합니다.

   PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치

다음 절차에 따라 macOS에서 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

절차

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.

2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.

3. OpenShift v4.9 MacOSX Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.

4. 아카이브의 압축을 해제하고 압축을 풉니다.

5. oc 바이너리 PATH의 디렉터리로 이동합니다.

   PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
$ oc <command>
```

18.3.13. CLI를 사용하여 클러스터에 로그인

클러스터 kubeconfig 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. kubeconfig 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- oc CLI를 설치했습니다.

절차

1. kubeadmin 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 oc 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

18.3.14. 레지스트리 스토리지 생성

클러스터를 설치한 후 Registry Operator를 위한 스토리지를 생성해야 합니다.

18.3.14.1. 설치 중 제거된 이미지 레지스트리

공유 가능한 개체 스토리지를 제공하지 않는 플랫폼에서 OpenShift Image Registry Operator는 자체적으로 Removed로 부트스트랩합니다. 이를 통해 openshift-installer가 이러한 플랫폼 유형에서 설치를 완료할 수 있습니다.

설치 후 managementState를 Removed에서 Managed로 전환하도록 Image Registry Operator 구성을 편집해야 합니다.

참고

Prometheus 콘솔은 ImageRegistryRemoved 경고를 제공합니다. 다음은 예시 경고입니다.

"이미지 레지스트리가 제거되었습니다. ImageStreamTags를 참조하는 ImageStreamTags, BuildConfigs 및 DeploymentConfigs가 예상대로 작동하지 않을 수 있습니다. 스토리지를 구성하고 configs.imageregistry.operator.openshift.io를 편집하여 구성을 Managed 상태로 업데이트하십시오."

18.3.14.2. 이미지 레지스트리 스토리지 구성

기본 스토리지를 제공하지 않는 플랫폼에서는 처음에 Image Registry Operator를 사용할 수 없습니다. 설치한 후에 스토리지를 사용하도록 레지스트리를 구성하여 Registry Operator를 사용 가능하도록 만들어야 합니다.

프로덕션 클러스터에 필요한 영구 볼륨을 구성하는 과정의 지침이 표시됩니다. 해당하는 경우, 프로덕션 환경 외 클러스터에서만 사용할 수 있는 저장 위치로서 빈 디렉터리를 구성하는 과정의 지침이 표시됩

니다.

업그레이드 중에 **Recreate** 롤아웃 전략을 사용하여 이미지 레지스트리의 블록 스토리지 유형 사용을 허용하기 위한 추가 지침이 제공됩니다.

18.3.14.2.1. VMware vSphere용 레지스트리 스토리지 구성

클러스터 관리자는 설치한 후 스토리지를 사용하도록 레지스트리를 구성해야 합니다.

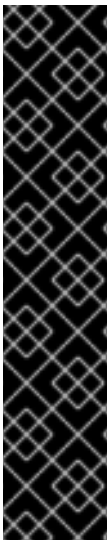
사전 요구 사항

- 클러스터 관리자 권한이 있어야 합니다.
- **VMware vSphere**에 클러스터가 있어야 합니다.
- **Red Hat OpenShift Container Storage**와 같이 클러스터용 영구 스토리지 프로비저닝.

중요

OpenShift Container Platform은 복제본이 하나만 있는 경우 이미지 레지스트리 스토리지에 대한 **ReadWriteOnce** 액세스를 지원합니다. **ReadWriteOnce** 액세스에서는 레지스트리가 **Recreate** 롤아웃 전략을 사용해야 합니다. 두 개 이상의 복제본으로 고 가용성을 지원하는 이미지 레지스트리를 배포하려면 **ReadWriteMany** 액세스가 필요합니다.

- **"100Gi"** 용량이 필요합니다.



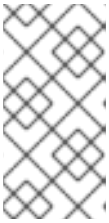
중요

테스트 결과, RHEL의 NFS 서버를 핵심 서비스용 스토리지 백엔드로 사용하는 데 문제가 있는 것으로 나타납니다. 여기에는 OpenShift Container Registry and Quay, 스토리지 모니터링을 위한 Prometheus, 로깅 스토리지를 위한 Elasticsearch가 포함됩니다. 따라서 RHEL NFS를 사용하여 핵심 서비스에서 사용하는 PV를 백업하는 것은 권장되지 않습니다.

마켓플레이스의 다른 NFS 구현에는 이러한 문제가 나타나지 않을 수 있습니다. 이러한 OpenShift Container Platform 핵심 구성 요소에 대해 완료된 테스트에 대한 자세한 내용은 개별 NFS 구현 공급업체에 문의하십시오.

프로세스

1. 스토리지를 사용하도록 레지스트리를 구성하기 위해 configs.imageregistry/cluster 리소스에서 spec.storage.pvc를 변경합니다.



참고

공유 스토리지를 사용할 때 보안 설정을 확인하여 외부에서의 액세스를 방지합니다.

2. 레지스트리 pod가 없는지 확인합니다.

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

출력 예

```
No resourses found in openshift-image-registry namespace
```

참고

출력에 레지스트리 Pod가 있는 경우 이 절차를 계속할 필요가 없습니다.

3. 레지스트리 구성을 확인합니다.

```
$ oc edit configs.imageregistry.operator.openshift.io
```

출력 예

```
storage:
  pvc:
    claim: 1
```

1 image-registry-storage 영구 볼륨 클레임 (PVC)을 자동으로 생성할 수 있도록 claim 필드를 비워 둡니다. PVC는 기본 스토리지 클래스를 기반으로 생성됩니다. 그러나 기본 스토리지 클래스에서 RADOS 블록 장치(RBD)와 같은 ReadWriteOnce(ReadWriteOnce) 볼륨을 제공할 수 있으므로 복제본이 두 개 이상될 때 문제가 발생할 수 있습니다.

4. clusteroperator 상태를 확인합니다.

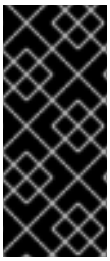
```
$ oc get clusteroperator image-registry
```

출력 예

```
NAME             VERSION                             AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                                 True        False         False     6h50m
```

18.3.14.2.2. VMware vSphere용 블록 레지스트리 스토리지 구성

클러스터 관리자로서 업그레이드 중에 이미지 레지스트리가 vSphere VMDK(Virtual Machine Disk)와 같은 블록 스토리지 유형을 사용할 수 있도록 허용하기 위해 Recreate 롤아웃 전략을 사용할 수 있습니다.



중요

블록 스토리지 볼륨이 지원되지만 프로덕션 클러스터에서 이미지 레지스트리와 함께 사용하는 것은 권장되지 않습니다. 레지스트리가 블록 스토리지에 구성된 설치는 레지스트리가 둘 이상의 복제본을 가질 수 없기 때문에 가용성이 높지 않습니다.

프로세스

1. 이미지 레지스트리 스토리지를 블록 스토리지 유형으로 설정하려면 **Recreate** 롤아웃 전략을 사용하고 복제본 **1**개 만으로 실행되도록 레지스트리를 패치합니다.

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. 블록 스토리지 장치에 **PV**를 프로비저닝하고 해당 볼륨의 **PVC**를 생성합니다. 요청된 블록 볼륨은 **RWO(ReadWriteOnce)** 액세스 모드를 사용합니다.

   a. **VMware vSphere PersistentVolumeClaim** 개체를 정의하려면 다음 내용이 포함된 **pvc.yaml** 파일을 생성합니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage (1)
  namespace: openshift-image-registry (2)
spec:
  accessModes:
  - ReadWriteOnce (3)
  resources:
    requests:
      storage: 100Gi (4)
```

(1) **PersistentVolumeClaim** 개체를 표시하는 고유한 이름입니다.

(2) **PersistentVolumeClaim** 오브젝트의 네임스페이스로 **openshift-image-registry**입니다.

(3) 영구 볼륨 클레임의 액세스 모드입니다. **ReadWriteOnce**를 사용하면 단일 노드에서 읽기 및 쓰기 권한으로 볼륨을 마운트할 수 있습니다.

4 영구 볼륨 클레임의 크기입니다.

b. 파일에서 **PersistentVolumeClaim** 오브젝트를 만듭니다.

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 올바른 **PVC**를 참조하도록 레지스트리 설정을 편집합니다.

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

출력 예

```
storage:
  pvc:
    claim: 1
```

1 사용자 지정 **PVC**를 만들면 **image-registry-storage PVC**의 기본 자동 생성을 위해 **claim** 필드를 비워둘 수 있습니다.

올바른 **PVC**를 참조하도록 레지스트리 스토리지를 구성하는 방법은 **vSphere** 용 레지스트리 구성을 참조하십시오.

18.3.15. VMware vSphere 볼륨 백업

OpenShift Container Platform은 새 볼륨을 독립 영구 디스크로 프로비저닝하여 클러스터의 모든 노드에서 볼륨을 자유롭게 연결 및 분리합니다. 결과적으로 스냅샷을 사용하는 볼륨을 백업하거나 스냅샷에서 볼륨을 복원할 수 없습니다. 자세한 내용은 스냅 샷 제한 을 참조하십시오.

프로세스

영구 볼륨의 백업을 생성하려면 다음을 수행합니다.

1. 영구 볼륨을 사용중인 애플리케이션을 중지합니다.
2. 영구 볼륨을 복제합니다.
3. 애플리케이션을 다시 시작합니다.
4. 복제된 볼륨의 백업을 만듭니다.
5. 복제된 볼륨을 삭제합니다.

18.3.16. 스틸 클럭 계정

기본적으로 설치 프로그램은 스틸 클럭 회계 매개 변수 (**stealclock.enabled**)를 활성화하지 않고 클러스터의 가상 머신을 프로비저닝합니다. 스틸링 클럭 계정을 사용하면 클러스터 문제 해결에 도움이 될 수 있습니다. 클러스터가 배포된 후 **vSphere Client**를 사용하여 각 가상 머신에서 이 매개변수를 활성화합니다.

자세한 내용은 이 **Red Hat** 기술 자료 문서 를 참조하십시오.

18.3.17. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 **Telemetry** 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 **Telemetry**가 자동으로 실행되고 **OpenShift Cluster Manager** 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 **OpenShift Cluster Manager**를 사용하여 자동으로 또는 **OpenShift Cluster Manager**를 사용하여 수동으로 유지 관리되는지 확인한 후 **subscription watch**를 사용하여 계정 또는 다중 클러스터 수준에서 **OpenShift Container Platform** 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

18.3.18. 다음 단계

- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.
- 레지스트리를 설정하고 레지스트리 스토리지를 구성합니다.
- 선택 사항: vSphere Problem Detector Operator에서 이벤트를 보고 클러스터에 권한 또는 스토리지 구성 문제가 있는지 확인합니다.

18.4. 네트워크 사용자 지정으로 VMC에 클러스터 설치

OpenShift Container Platform 4.9 버전에서는 AWS의 VMC (VMware Cloud)에 배포하여 사용자 지정 네트워크 구성 옵션이 있는 설치 관리자 프로비저닝 인프라를 사용하여 VMware vSphere 인스턴스에 클러스터를 설치할 수 있습니다.

OpenShift Container Platform 배포를 위해 VMC 환경을 구성한 후 VMC 환경에 공동 배치된 bastion 관리 호스트에서 OpenShift Container Platform 설치 프로그램을 사용합니다. 설치 프로그램 및 컨트롤 플레인은 OpenShift Container Platform 클러스터에 필요한 리소스 배포 및 관리 프로세스를 자동화합니다.

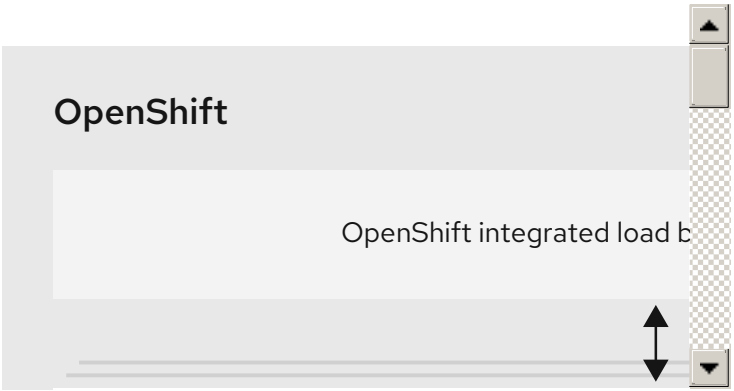
OpenShift Container Platform 네트워크 구성을 사용자 지정할 경우, 클러스터가 사용자 환경의 기존 IP 주소 할당과 공존하고 기존 MTU 및 VXLAN 구성과 통합될 수 있습니다. 설치를 사용자 지정하려면 클러스터를 설치하기 전에 install-config.yaml 파일에서 매개변수를 수정합니다. 설치 과정에서 네트워크 구성 매개변수를 대부분 설정해야 하며, 실행 중인 클러스터에서는 kubeProxy 구성 매개변수만 수정할 수 있습니다.

참고

OpenShift Container Platform은 단일 VMware vCenter에만 클러스터 배포를 지원합니다. 여러 vCenter에 머신/머신 세트가 있는 클러스터 배포는 지원되지 않습니다.

18.4.1. vSphere용 VMC 설정

AWS 호스팅된 vSphere 클러스터에 VMware Cloud (VMC)의 OpenShift Container Platform을 설치하여 하이브리드 클라우드 전반에 애플리케이션을 온프레미스 및 오프프레미스 모두 배포 및 관리할 수 있습니다.



VMware vSphere에 OpenShift Container Platform을 설치하기 전에 VMC 환경에서 여러 옵션을 구성해야 합니다. VMC 환경에 다음과 같은 사전 요구 사항이 있는지 확인합니다.

- 비독점 DHCP 사용, NSX-T 네트워크 세그먼트 및 서브넷을 생성합니다. 서브넷에서 다른 VM(가상 머신)을 호스팅할 수 있지만 OpenShift Container Platform 배포에서는 8개 이상의 IP 주소를 사용할 수 있어야 합니다.
- DHCP 범위 외부에서 두 개의 IP 주소를 할당하고 역방향 DNS 레코드로 구성합니다.
  - 할당된 IP 주소를 가리키는 api.<cluster_name>.<base_domain>의 DNS 레코드입니다.
  - 할당된 IP 주소를 가리키는 *.apps.<cluster_name>.<base_domain>의 DNS 레코드입니다.
- 다음 방화벽 규칙을 설정합니다.
  - OpenShift Container Platform 컴퓨팅 네트워크와 인터넷 간의 ANY:ANY 방화벽 규칙입니다. 이는 노드와 애플리케이션에서 컨테이너 이미지를 다운로드하는 데 사용됩니다.
  - 포트 443의 설치 호스트와 소프트웨어 정의 데이터 센터(SDDC) 관리 네트워크 간의 ANY:ANY 방화벽 규칙입니다. 이를 통해 배포 중에 RHCOS(Red Hat Enterprise Linux CoreOS) OVA를 업로드할 수 있습니다.

- OpenShift Container Platform 컴퓨팅 네트워크와 인터넷 간의 HTTPS 방화벽 규칙입니다. 이 연결을 통해 OpenShift Container Platform은 노드, PVC(영구 볼륨 클레임) 및 기타 리소스를 프로비저닝 및 관리하기 위해 vCenter과 통신할 수 있습니다.

- OpenShift Container Platform을 배포하려면 다음 정보가 있어야 합니다.

  - OpenShift Container Platform 클러스터에 vmc-prod-1로 로그인합니다.

  - companyname.com과 같은 기본 DNS 이름입니다.

  - 기본값을 사용하지 않는 경우 Pod 네트워크 CIDR 및 서비스 네트워크 CIDR을 식별해야 하며, 기본적으로 10.128.0.0/14 및 172.30.0.0/16으로 설정됩니다. 이러한 CIDR은 pod-to-pod 및 pod-to-service 통신에 사용되며 외부에서 액세스할 수 없지만 조직의 기존 서브넷과 중복되어서는 안 됩니다.

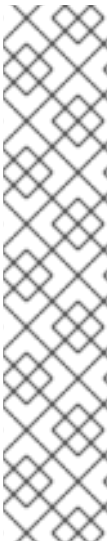
  - 다음 vCenter 정보를 참조하십시오:

    - vCenter 호스트 이름, 사용자 이름 및 암호
    - 데이터 센터 이름 (예: SDDC-Datacenter)
    - Cluster-1과 같은 클러스터 이름
    - 네트워크 이름
    - WorkloadDatastore와 같은 데이터 저장소 이름

참고

클러스터 설치가 완료된 후 vSphere 클러스터를 VMC Compute-ResourcePool 리소스 풀로 이동하는 것이 좋습니다.

- VMC에 bastion으로 배포된 Linux 기반 호스트입니다.
  - bastion 호스트는 RHEL(Red Hat Enterprise Linux) 또는 기타 Linux 기반 호스트일 수 있습니다. 인터넷에 연결되어 있어야 하고 OVA를 ESXi 호스트에 업로드하는 기능이 있어야 합니다.
  - OpenShift CLI 툴을 bastion 호스트에 다운로드하여 설치합니다.
    - openshift-install 설치 프로그램
    - OpenShift CLI(oc) 툴



참고

Kubernetes 용 VMware NCP (NSX Conrainer Plugin)dOpen에는 사용할 수 없으며 NSX는 OpenShift SDN으로 사용되지 않습니다. 현재 VMC에서 사용할 수 있는 NSX 버전은 OpenShift Container Platform에서 인증된 NCP 버전과 호환되지 않습니다.

그러나 NSX DHCP 서비스는 전체 스택 자동화된 OpenShift Container Platform 배포와 vSphere와의 Machine API 통합을 통해 수동 또는 자동으로 프로비저닝된 노드를 사용하여 가상 머신 IP 관리에 사용됩니다. 또한 OpenShift Container Platform 클러스터와 bastion 호스트와 VMC vSphere 호스트 사이에서 액세스할 수 있도록 NSX 방화벽 규칙이 생성됩니다.

18.4.1.1. VMC Sizer 툴

AWS의 VMware Cloud는 AWS 베어메탈 인프라 상단에 구축됩니다. 이는 AWS 네이티브 서비스를 실행하는 것과 동일한 베어 메탈 인프라입니다. AWS 소프트웨어 정의 데이터 센터(SDDC)의 VMware 클라우드가 배포되면 이러한 물리적 서버 노드를 사용하고 단일 테넌트 방식으로 VMware ESXi 하이퍼바이저를 실행할 수 있습니다. 즉, VMC를 사용하는 모든 사용자가 물리적 인프라에 액세스할 수 없습니다. 가상 인프라를 호스팅하는데 필요한 물리적 호스트 수를 고려해야 합니다.

이를 확인하기 위해 VMware는 AWS Sizer에서 VMC를 제공합니다. 이 툴을 사용하면 VMC에서 호스팅할 리소스를 정의할 수 있습니다.

- 워크로드 유형

- 총 가상 머신 수
- 사양 정보는 다음과 같습니다:
  - 스토리지 요구사항
  - **vCPU**
  - **vRAM**
  - 오버커밋 비율

이러한 세부 정보를 사용하여 **VMware** 모범 사례를 기반으로 보고서를 생성하고 클러스터 구성과 필요한 호스트 수를 권장합니다.

18.4.2. vSphere 사전 요구 사항

- **OpenShift Container Platform** 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.
- 블록 레지스트리 스토리지가 프로비저닝되어 있습니다. 자세한 내용은 영구 저장 장치 이해를 참조하십시오.
- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성해야 합니다.

참고

프록시를 구성하는 경우에도 해당 사이트 목록을 검토하십시오.

18.4.3. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.

중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

18.4.4. VMware vSphere 인프라 요구사항

사용하는 구성 요소의 요구사항을 충족하는 VMware vSphere 버전 6 또는 7 인스턴스에 OpenShift Container Platform 클러스터를 설치해야 합니다.

표 18.20. vSphere 가상 환경의 버전 요구 사항

가상 환경 제품	필요한 버전
VM 하드웨어 버전	13 이상
vSphere ESXi hosts	6.5 이상
vCenter 호스트	6.5 이상

중요

이제 VMware vSphere 버전 6.7U2 또는 이전 버전에 클러스터를 설치하는 것이 더 이상 사용되지 않으며 가상 하드웨어 버전 13은 더 이상 사용되지 않습니다. 이러한 버전은 여전히 완전히 지원되지만 향후 OpenShift Container Platform 버전에서는 지원이 제거됩니다. 이제 OpenShift Container Platform의 vSphere 가상 머신의 하드웨어 버전 15가 기본값이 되었습니다. vSphere 노드의 하드웨어 버전을 업데이트하려면 " vSphere에서 실행되는 노드에서 하드웨어 업데이트" 문서를 참조하십시오.

표 18.21. VMware 구성 요소에 지원되는 최소 vSphere 버전

구성 요소	지원되는 최소 버전	설명
하이퍼바이저	HW 버전 13이 포함된 vSphere 6.5 이상	이 버전은 RHCOS(Red Hat Enterprise Linux CoreOS)가 지원하는 최소 버전입니다. Red Hat Enterprise Linux 8 지원 하이퍼바이저 목록을 참조하십시오.
In-tree 드라이버가 있는 스토리지	vSphere 6.5 이상	이 플러그인은 OpenShift Container Platform에 포함된 vSphere용 인트리 스토리지 드라이버를 사용하여 vSphere 스토리지를 생성합니다.

vSphere 버전 6.5 인스턴스를 사용하는 경우 OpenShift Container Platform을 설치하기 전에 6.7U3 또는 7.0로 업그레이드하는 것이 좋습니다.

중요

OpenShift Container Platform을 설치하기 전에 ESXi 호스트의 시간이 동기화되었는지 확인해야 합니다. VMware 문서에서 Edit Time Configuration for a Host를 참조하십시오.

18.4.5. 네트워크 연결 요구사항

OpenShift Container Platform 클러스터 구성 요소가 통신할 수 있도록 시스템 간 네트워크 연결을 구성해야 합니다.

필요한 네트워크 포트에 대한 다음 세부 정보를 검토합니다.

표 18.22. 모든 시스템 간 통신에 사용되는 포트

프로토콜	포트	설명
ICMP	해당 없음	네트워크 연결성 테스트
TCP	**1936**	메트릭
	9000-9999	**9100-9101** 포트의 노드 내보내기 및 **9099** 포트의 Cluster Version Operator를 포함한 호스트 수준 서비스.
	10250-10259	Kubernetes에서 예약하는 기본 포트
	10256	openshift-sdn
UDP	**4789**	VXLAN(Virtual extensible LAN)
	6081	Geneve
	9000-9999	**9100-9101** 포트의 노드 내보내기를 포함한 호스트 수준 서비스.
	500	IPsec IKE 패킷
	4500	IPsec NAT-T 패킷
TCP/UDP	**30000-32767**	Kubernetes 노드 포트
ESP	해당 없음	IPsec Encapsulating Security Payload (ESP)

표 18.23. 모든 시스템과 컨트롤 플레인 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**6443**	Kubernetes API

표 18.24. 컨트롤 플레인 머신 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**2379-2380**	etcd 서버 및 피어 포트

추가 리소스

- **vSphere 노드의 하드웨어 버전을 업데이트하려면 vSphere에서 실행 중인 노드에서 하드웨어 업데이트를 참조하십시오.**

18.4.6. vCenter 요구사항

설치 관리자가 프로비저닝하는 인프라를 사용하는 **vCenter**에 **OpenShift Container Platform** 클러스터를 설치하기 전에 환경을 준비해야 합니다.

필요한 **vCenter** 계정 권한

vCenter에 **OpenShift Container Platform** 클러스터를 설치하려면 설치 프로그램에서 필요한 리소스를 읽고 생성할 수 있는 권한이 있는 계정에 액세스해야 합니다. 필요한 모든 권한에 액세스할 수 있는 가장 간단한 방법은 글로벌 관리 권한이 있는 계정을 사용하는 것입니다.

글로벌 관리 권한이 있는 계정을 사용할 수 없는 경우 **OpenShift Container Platform** 클러스터 설치에 필요한 권한을 부여하려면 역할을 생성해야 합니다. 대부분의 권한이 항상 필요하지만 기본 동작인 **vCenter** 인스턴스에 **OpenShift Container Platform** 클러스터를 포함하는 폴더를 프로비저닝할 설치 프로그램인 경우에만 필요합니다. 필요한 권한을 부여하려면 지정된 오브젝트에 대해 **vSphere** 역할을 생성하거나 수정해야 합니다.

설치 프로그램이 **vSphere** 가상 머신 폴더를 생성하는 경우 추가 역할이 필요합니다.

예 18.7. vSphere API에 설치에 필요한 역할 및 권한

역할의 vSphere 개체	필요한 경우	vSphere API에서 필요한 권한
vSphere vCenter	Always	CNS .**Searchable** **InventoryService.Tagging.AttachTag** **InventoryService.Tagging.CreateCategory** **InventoryService.Tagging.CreateTag** **InventoryService.Tagging.Delete**Tag **InventoryService.tagged.DeleteTag** **InventoryService.Tagging.EditCategory** **InventoryService.Tagging.EditTag**ECDHE**Session** **ValidateSession**Forwarded**StorageProfile.Update**Forwarded**StorageProfile.View**

역할의 vSphere 개체	필요한 경우	vSphere API에서 필요한 권한
vSphere vCenter Cluster	Always	**Host.Config.Storage** **Resource.AssignVMToPool** **VApp.AssignResourcePool** **VApp.Import** **VirtualMachine.Config.Add NewDisk**
vSphere Datastore	Always	**Datastore.AllocateSpace** **Datastore.LoadBalancer** **Datastore.FileManagement** **InventoryService.Tagging.O bjectAttachable**
vSphere Port Group	Always	**Network.Assign**

역할의 vSphere 개체	필요한 경우	vSphere API에서 필요한 권한
가상 머신 폴더	Always	**InventoryService.ume.ObjectAttachable Resource.AssignVMToPool VApp.Import VirtualMachine.Config.Add ExistingDisk VirtualMachine.Config.Add NewDisk VirtualMachine.Config.Add RemoveDevice VirtualMachine.Config.Advanced Config VirtualMachine.Config.Anno tationECDHEVirtualMachine .Config.DiskExtend**Forwarde d VirtualMachine**.Config.**DiskExte nd Forwarded**VirtualMachine.Co nfig.DiskLeaseVirtualMachi ne.Config.ReCreate**VolumeCl aim.Config**.MemoryDriver.**

역할의 vSphere 개체	필요한 경우	vSphere API에서 필요한 권한
vSphere vCenter Datacenter	설치 프로그램이 가상 머신 폴더를 생성하는 경우	**InventoryService.Tagging.ObjectAttachable Resource.AssignVMToPool VApp.Import VirtualMachine.Config.AddExistingDisk VirtualMachine.Config.AddNewDisk VirtualMachine.Config.AddRemoveDevice VirtualMachine.Config.AdvancedConfig VirtualMachine.Config.AnnotationECDHEVirtualMachine.Config.DiskExtend**Forwarded VirtualMachine**.**Config**.DiskExtend ForwardedVirtualMachine.Config.DiskLeaseVirtualMachine.Config.ReCreate VolumeClaim.Config.Reset**defined**.**

예 18.8. vCenter 그래픽 사용자 인터페이스(GUI)에 설치하는 데 필요한 역할 및 권한

역할의 vSphere 개체	필요한 경우	vCenter GUI에서 필요한 권한
vSphere vCenter	Always	CNS **.Searchable "vSphere agged""Assign or unassign vSphere Tag" "vSphere tagging" "vSphere" "vSphere" "vSphere"CreatevSphere Tag" vSphere 태그""vSphere" 삭제 태그 지정" "vSphere Assignment""Edit vSphere Tag" "vSphere Tagging" "vSphere 태그" 세션" "Profile 기반 스토리지""Profile 기반 스토리지 업데이트""Profile 기반 스토리지"Profile 중심 스토리지 업데이트"Profile 기반 스토리지 업데이트"Profile 기반 스토리지 업데이트 스토리지 보기"**
vSphere vCenter Cluster	클러스터 루트에서 VM이 생성되는 경우	**Host.Configuration"Storage partition configuration" 리소스" 리소스 풀에 가상 머신을 서명" VApp"Assign resource pool" VApp.Import"Virtual machine""Change Configuration"."새 디스크 추가"**
vSphere vCenter 리소스 풀	기존 리소스 풀이 제공되는 경우	**Host.Configuration"Storage partition configuration" 리소스" 리소스 풀에 가상 머신을 서명" VApp"Assign resource pool" VApp.Import"Virtual machine""Change Configuration"."새 디스크 추가"**

역할의 vSphere 개체	필요한 경우	vCenter GUI에서 필요한 권한
vSphere Datastore	Always	**Datastore"Allocate space"** 데이터**저장소"** **데이터 저장소"** **Datastore"Low level file operations""vSphere Assignment"Assign or Unassign vSphere Tag on Object"**
vSphere Port Group	Always	**네트워크 "Assign network"**
가상 머신 폴더	Always	**"vSphere Assignment""Assign or unassign vSphere Tag on Object"** **Resource"Assign virtual machine to resource pool"**: **VApp.Import** **"Virtual machine""Change Configuration","Add existing disk""**Add existing disk**""Add new disk""Add new disk" "Virtual""** **머신 "시스템 변경", "가상 머신""** **"가상 머신""시스템 변경"** **"가상 머신""변경 구성",** **"가상 머신", 변경, CPU 수 변경"** **"가상 머신", 가상 디스크 변경", 가상 디스크 변경"** **"가상머신". 머신 변경 "구성 변경"** **"가상 머신""마이크 구성 변경"** **"가상 머신""마이크 구성 변경" "구성 변경",** **"가상 머신", 변경" 디스크 삭제"** **"가상 머신".변경 구성".Rename** **"가상 머신 변경" VIX API**로가상 머신 변경**"Gregradevirtual machine"Guestoperating system management by VIX API**

역할의 vSphere 개체	필요한 경우	vCenter GUI에서 필요한 권한
		"Virtual machine". Interconnection"Power on" "Virtual machine". interactionion.Reset "Virtual machine"."Edit Inventory"."Create new" "Virtual machine"."Create existing" "Virtual machine"Edit Inventory""Create from existing" "Virtual machine"Edit Inventory""Create from existing" virtual machine".Edit Inventory""Create from existing" virtual machine".Edit Inventory""Create from existing""Virtual machine" machine".Provisioning."Clone 가상 머신" "가상 머신".Provisioning." "가상 머신".Provisioning."Deploy 템플릿"
vSphere vCenter Datacenter	설치 프로그램이 가상 머신 폴더를 생성하는 경우	"vSphere Assignment""Assign or unassign vSphere Tag on Object" Resource"Assign virtual machine to resource pool": VApp.Import "Virtual machine""Change Configuration","Add existing disk""Add existing disk""Add new disk""Add new disk" "Virtual"" 머신 "시스템 변경", "가상 머신"" "가상 머신""시스템 변경" "가상 머신""변경 구성", "가상 머신", 변경, CPU 수 변경" "가상 머신", 가상 디스크 변경", 가상 디스크 변경" "가상머신". 머신 변경 "구성 변경" "가상 머신""마이크 구성 변경" "가상 머신""마이크 구성 변경" "구성 변경",

역할의 vSphere 개체	필요한 경우	"가상 머신", 변경" 디스크 삭 vCenter GUI에서 필요한 권한
		"가상 머신".변경 구 성".Rename "가상 머신 변경" VIX API로가 상 머신 변 경"Gregradevirtual machine"Guestoperating system management by VIX API machine". interaction" "Virtual machine". Interconnection"Power on" "Virtual machine". interactionion.Reset "Virtual machine"."Edit Inventory"."Create new" "Virtual machine"."Create existing" "Virtual machine"Edit Inventory""Create from existing" "Virtual machine"Edit Inventory""Create from existing" virtual machine".Edit Inventory""Create from existing" virtual machine".Edit Inventory""Create from existing""Virtual machine" machine".Provisioning."Clo ne 가상 머신" "가상 머 신".Provisioning."Deploy 템 플릿" "가상 머신".Provisioning" 폴더 만들기" 폴더 생성"폴더 삭제"

또한 사용자에게 일부 **ReadOnly** 권한이 필요하며 일부 역할에는 하위 오브젝트에 권한을 부여하는 사용 권한이 필요합니다. 이러한 설정은 클러스터를 기존 폴더에 설치할지 여부에 따라 달라집니다.

예 18.9. 필수 권한 및 권한 부여 설정

vSphere 오브젝트	폴더 유형	하위 항목으로 권한 부여	권한 필요
vSphere vCenter	Always	False	나열된 필수 권한
vSphere vCenter Datacenter	기존 폴더	False	**ReadOnly** 권한
	설치 프로그램은 폴더를 생성	True	나열된 필수 권한
vSphere vCenter Cluster	Always	True	나열된 필수 권한
vSphere vCenter Datastore	Always	False	나열된 필수 권한
vSphere Switch	Always	False	**ReadOnly** 권한
vSphere Port Group	Always	False	나열된 필수 권한
vSphere vCenter Virtual Machine Folder	기존 폴더	True	나열된 필수 권한

필요한 권한만으로 계정을 생성하는 방법에 대한 자세한 내용은 **vSphere** 문서에서 **vSphere 권한 및 사용자 관리 작업**을 참조하십시오.

vMotion으로 OpenShift Container Platform 사용

vSphere 환경에서 **vMotion**을 사용하려는 경우 **OpenShift Container Platform** 클러스터를 설치하기 전에 다음을 고려하십시오.

- **OpenShift Container Platform**은 일반적으로 **compute-only vMotion** 을 지원합니다. 스토리지 **vMotion**을 사용하면 문제가 발생할 수 있으며 지원되지 않습니다.

  컴퓨팅 및 컨트롤 플레인 노드의 가동 시간을 보장하기 위해 **vMotion**에 대한 **VMware** 모범 사례를 따르는 것이 좋습니다. 또한 **VMware** 유사성 방지 규칙을 사용하여 유지 관리 또는 하드웨어 문제 중에 **OpenShift Container Platform**의 가용성을 개선하는 것이 좋습니다.

  vMotion 및 유사성 방지 규칙에 대한 자세한 내용은 **vMotion 네트워킹 요구 사항** 및 **VM 유사성 방지 규칙에** 대한 **VMware vSphere** 설명서를 참조하십시오.

- **Pod**에서 **vSphere** 볼륨을 사용하는 경우 데이터 저장소를 통해 **VM**을 마이그레이션하면 수

동으로 또는 스토리지 **vMotion**을 통해 **VM**을 마이그레이션하면 **OpenShift Container Platform PV(**영구 볼륨**)** 오브젝트 내에서 잘못된 참조가 표시됩니다**.** 이러한 참조는 영향을 받는 포드가 시작되지 않게 하고 데이터 손실을 초래할 수 있습니다**.**

- 마찬가지로 **OpenShift Container Platform**은 데이터 저장소에서 **VMDK**의 선택적 마이그레이션을 지원하지 않으며 **VM** 프로비저닝 또는 **PV**의 동적 또는 정적 프로비저닝을 위해 데이터 저장소 클러스터의 일부 또는 **PV**의 동적 또는 정적 프로비저닝을 위해 데이터 저장소 클러스터의 일부인 데이터 저장소를 사용하는 경우도 있습니다**.**

클러스터 리소스

설치 관리자 프로비저닝 인프라를 사용하는 **OpenShift Container Platform** 클러스터를 배포할 때는 설치 프로그램이 **vCenter** 인스턴스에서 여러 리소스를 생성할 수 있어야 합니다**.**

표준 **OpenShift Container Platform**을 설치하면 다음 **vCenter** 리소스가 생성됩니다**.**

- 폴더 한 개
- 태그 카테고리 한 개
- 태그 한 개
- 가상 머신**:**
  - 템플릿 한 개
  - 임시 부트스트랩 노드 한 개
  - 컨트롤 플레인 노드 세 개
  - 컴퓨팅 시스템 세 개

이러한 리소스는 **856GB**의 스토리지를 사용하지만 부트스트랩 노드는 클러스터 설치 프로세스 중에 제거됩니다**.** 표준 클러스터를 사용하려면 최소 **800GB**의 스토리지가 필요합니다**.**

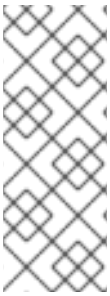
컴퓨팅 시스템을 더 많이 배포할수록 **OpenShift Container Platform** 클러스터는 더 많은 스토리지를 사용합니다.

클러스터 제한

사용 가능한 리소스는 클러스터마다 다릅니다. **vCenter** 내에서 가능한 클러스터 수는 주로 사용 가능한 스토리지 공간과 필요한 리소스 수에 대한 제한으로 제한됩니다. 클러스터가 생성하는 **vCenter** 리소스와 **IP** 주소 및 네트워크와 같이 클러스터를 배포하는 데 필요한 리소스에 대한 제한 사항을 모두 고려해야 합니다.

네트워킹 요구사항

네트워크에 **DHCP**를 사용하고 **DHCP** 서버가 클러스터 시스템에 영구 **IP** 주소를 제공하도록 구성되어 있는지 확인합니다. **DHCP** 서버를 사용하도록 기본 게이트웨이를 구성해야 합니다. 모든 노드는 동일한 **VLAN**에 있어야 합니다. 두 번째 **VLAN**을 **Day 2** 작업으로 사용하여 클러스터를 확장할 수 없습니다. 또한 **OpenShift Container Platform** 클러스터를 설치하기 전에 다음 네트워킹 리소스를 생성해야 합니다.



참고

클러스터의 각 **OpenShift Container Platform** 노드는 **DHCP**를 통해 검색할 수 있는 **NTP(Network Time Protocol)** 서버에 액세스할 수 있어야 합니다. **NTP** 서버 없이도 설치할 수 있습니다. 그러나 비동기 서버 클록으로 인해 **NTP** 서버가 차단하는 오류가 발생합니다.

필요한 IP 주소

설치 관리자 프로비저닝 **vSphere** 설치에는 다음 두 개의 고정 **IP** 주소가 필요합니다.

- **API** 주소는 클러스터 **API**에 액세스하는 데 사용됩니다.
- **Ingress** 주소는 클러스터 인그레스 트래픽에 사용됩니다.

OpenShift Container Platform 클러스터를 설치할 때 설치 프로그램에 이 **IP** 주소를 제공해야 합니다.

DNS 레코드

OpenShift Container Platform 클러스터를 호스팅하는 **vCenter** 인스턴스에 적절한 **DNS** 서버에서 고정 **IP** 주소 두 개의 **DNS** 레코드를 생성해야 합니다. 각 레코드에서 **<cluster_name>**은 클러스터 이름

이고 <base_domain>은 클러스터를 설치할 때 지정하는 클러스터 기본 도메인입니다. 전체 DNS 레코드는 <component>.<cluster_name>.<base_domain> 형식입니다.

표 18.25. 필수 DNS 레코드

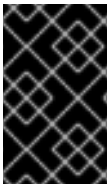
구성 요소	레코드	설명
API VIP	api.<cluster_name>.<base_domain>.	이 DNS A/AAAA 또는 CNAME 레코드는 컨트롤 플레인 시스템의 로드 밸런서를 가리켜야 합니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.
Ingress VIP	*.apps.<cluster_name>.<base_domain>.	기본적으로 작업자 노드인 인그레스 라우터 Pod를 실행하는 시스템을 대상으로 하는 로드 밸런서를 가리키는 와일드카드 DNS A/AAAA 또는 CNAME 레코드입니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.

18.4.7. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 core 사용자의 ~/.ssh/authorized_keys 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 core로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. ./openshift-install gather 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.



중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

참고

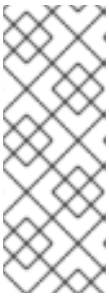
AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

절차

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: ~/.ssh/id_ed25519)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 '~/.ssh 디렉터리에 있는지 확인하십시오.



참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 ed25519 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 rsa 또는 ecdsa 알고리즘을 사용하는 키를 생성합니다.

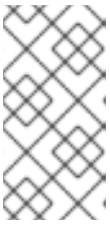
2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 ~/.ssh/id_ed25519.pub 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 ./openshift-install gather 명령을 사용하려는 경우 필요합니다.



참고

일부 배포에서는 ~/.ssh/id_rsa 및 ~/.ssh/id_dsa와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

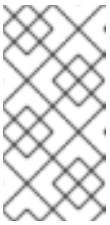
a.

ssh-agent 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```



참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4.

ssh-agent에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1

SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: ~/.ssh/id_ed25519).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

18.4.8. 설치 프로그램 받기

OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

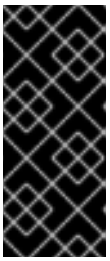
- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

절차

1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.

2. 인프라 공급자를 선택합니다.

3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.



중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.

중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 OpenShift Container Platform 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

18.4.9. 시스템 신뢰에 vCenter 루트 CA 인증서 추가

설치 프로그램이 vCenter API에 액세스해야 하므로 OpenShift Container Platform 클러스터를 설치하기 전에 vCenter의 신뢰할 수 있는 루트 CA 인증서를 시스템 신뢰에 추가해야 합니다.

절차

1. vCenter 홈 페이지에서 vCenter의 루트 CA 인증서를 다운로드합니다. vSphere Web Services SDK 섹션에서 Download trusted root CA certificates를 클릭합니다. <vCenter>/certs/download.zip 파일이 다운로드됩니다.

2. vCenter 루트 CA 인증서가 포함된 압축 파일의 압축을 풉니다. 압축 파일의 내용은 다음 파일 구조와 유사합니다.

```
certs
├── lin
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
├── mac
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
└── win
    ├── 108f4d17.0.crt
    ├── 108f4d17.r1.crl
    ├── 7e757f6a.0.crt
    ├── 8e4f8471.0.crt
    └── 8e4f8471.r0.crl

3 directories, 15 files
```

3. 운영 체제 파일을 시스템 신뢰에 추가합니다. 예를 들어 Fedora 운영 체제에서는 다음 명령을 실행합니다.

-

```
# cp certs/lin/* /etc/pki/ca-trust/source/anchors
```

4. 시스템 신뢰를 업데이트합니다. 예를 들어 **Fedora** 운영 체제에서는 다음 명령을 실행합니다.

```
# update-ca-trust extract
```

18.4.10. 설치 구성 파일 만들기

VMware vSphere에 설치하는 **OpenShift Container Platform** 클러스터를 사용자 지정할 수 있습니다.

사전 요구 사항

- **OpenShift Container Platform** 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.
- 서브스크립션 수준에서 서비스 주체 권한을 획득합니다.

절차

1. **install-config.yaml** 파일을 생성합니다.
   a. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 **<installation_directory>**는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.

중요

비어 있는 디렉터리를 지정합니다. 부트스트랩 **X.509** 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 **OpenShift Container Platform** 버전에서 설치 파일을 복사할 때는 주의하십시오.

b.

화면에 나타나는 지시에 따라 클라우드에 대한 구성 세부 사항을 입력합니다.

i.

선택사항: 클러스터 시스템에 액세스하는 데 사용할 SSH 키를 선택합니다.

참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 ssh-agent 프로세스가 사용하는 SSH 키를 지정합니다.

ii.

대상 플랫폼으로 vsphere를 선택합니다.

iii.

vCenter 인스턴스의 이름을 지정합니다.

iv.

클러스터를 생성하는 데 필요한 권한이 있는 vCenter 계정의 사용자 이름과 암호를 지정합니다.

설치 프로그램이 vCenter 인스턴스에 연결됩니다.

v.

연결할 vCenter 인스턴스에서 데이터 센터를 선택합니다.

vi.

사용할 기본 vCenter 데이터 저장소를 선택합니다.

vii.

OpenShift Container Platform 클러스터를 설치할 vCenter 클러스터를 선택합니다. 설치 프로그램은 vSphere 클러스터의 루트 리소스 풀을 기본 리소스 풀로 사용합니다.

viii.

vCenter 인스턴스에서 구성한 가상 IP 주소 및 DNS 레코드가 포함된 네트워크를 선택합니다.

ix.

컨트롤 플레인 API 액세스를 위해 구성한 가상 IP 주소를 입력합니다.

v

x.
클러스터 인그레스용으로 구성한 가상 **IP** 주소를 입력합니다.

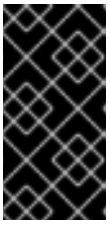
xi.
기본 도메인을 입력합니다. 이 기본 도메인은 구성한 **DNS** 레코드에서 사용한 것과 동일해야 합니다.

xii.
클러스터를 설명할 수 있는 이름을 입력합니다. 클러스터 이름은 구성한 **DNS** 레코드에서 사용한 것과 동일해야 합니다.

xiii.
Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 붙여넣습니다.

2. **install-config.yaml** 파일을 수정합니다. 사용 가능한 매개변수에 대한 자세한 정보는 “설치 구성 매개변수” 섹션에서 확인할 수 있습니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.

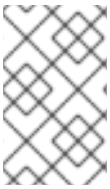


중요

install-config.yaml 파일은 설치 과정에서 사용됩니다. 이 파일을 재사용하려면 지금 백업해야 합니다.

18.4.10.1. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 매개변수 값을 제공하여 클러스터를 호스팅할 클라우드 플랫폼에서 사용자 계정을 설명하고 선택사항으로 클러스터의 플랫폼을 사용자 지정합니다. **install-config.yaml** 설치 구성 파일을 생성할 때 명령줄을 통해 필요한 매개변수 값을 제공합니다. 클러스터를 사용자 지정하면 **install-config.yaml** 파일을 수정하여 플랫폼에 대한 세부 정보를 제공할 수 있습니다.



참고

설치한 후에는 **install-config.yaml** 파일에서 이러한 매개변수를 수정할 수 없습니다.

중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

18.4.10.1.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 18.26. 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	**dev** 와 같은 소문자 및 하이픈(**-**)의 문자열입니다.

매개변수	설명	값
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

18.4.10.1.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 18.27. 네트워크 매개변수

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체 **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.

매개변수	설명	값
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.
networking.clusterNetwork	Pod의 IP 주소 블록입니다. 기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된 **cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16

매개변수	설명	값
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16**  **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

18.4.10.1.3. 선택적 구성 매개변수

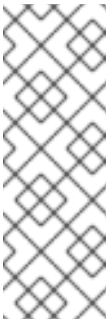
선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 18.28. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열

매개변수	설명	값
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은 **3**입니다.
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열

매개변수	설명	값
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 기본값은 **External**입니다. 이 필드를 **Internal** 로 설정하는 것은 클라우드 이외의 플랫폼에서는 지원되지 않습니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

18.4.10.1.4. 추가 VMware vSphere 구성 매개변수

추가 VMware vSphere 구성 매개변수는 다음 표에 설명되어 있습니다.

표 18.29. 추가 VMware vSphere 클러스터 매개변수

매개변수	설명	값
platform.vsphere.vCenter	vCenter 서버의 정규화된 호스트 이름 또는 IP 주소입니다.	문자열
platform.vsphere.username	vCenter 인스턴스에 연결하는 데 사용할 사용자 이름입니다. 이 사용자에게는 최소한 vSphere에서 정적 또는 동적 영구 볼륨 프로비저닝 에 필요한 역할과 권한이 있어야 합니다.	문자열
platform.vsphere.password	vCenter 사용자 이름의 암호입니다.	문자열
platform.vsphere.datacenter	vCenter 인스턴스에서 사용할 데이터 센터의 이름입니다.	문자열
platform.vsphere.defaultDatastore	프로비저닝 볼륨에 사용할 기본 데이터 저장소의 이름입니다.	문자열
platform.vsphere.folder	*선택사항*입니다. 설치 프로그램이 가상 머신을 생성하는 기존 폴더의 절대 경로입니다. 이 값을 제공하지 않으면 설치 프로그램이 데이터 센터 가상 머신 폴더에 인프라 ID로 이름이 지정된 폴더를 생성합니다.	문자열(예: **/<datacenter_name>/vm/<folder_name>/<subfolder_name>**).

매개변수	설명	값
platform.vsphere.network	vCenter 인스턴스에서 구성한 가상 IP 주소 및 DNS 레코드가 포함된 네트워크입니다.	문자열
platform.vsphere.cluster	OpenShift Container Platform 클러스터를 설치할 vCenter 클러스터입니다.	문자열
platform.vsphere.apiVIP	컨트롤 플레인 API 액세스를 위해 구성한 가상 IP(VIP) 주소입니다.	IP 주소 (예: **128.0.0.1**)
platform.vsphere.ingressVIP	클러스터 인그레스용으로 구성한 가상 IP(VIP) 주소입니다.	IP 주소 (예: **128.0.0.1**)

18.4.10.1.5. 선택적 VMware vSphere 시스템 풀 구성 매개변수

다음 표에는 선택적 VMware vSphere 시스템 풀 구성 매개 변수가 설명되어 있습니다.

표 18.30. 선택적 VMware vSphere 시스템 풀 매개변수

매개변수	설명	값
platform.vsphere.clusterOSImage	설치 프로그램이 RHCOS 이미지를 다운로드하는 위치입니다. 네트워크가 제한된 환경에서 설치를 수행하려면이 매개변수를 설정해야 합니다.	HTTP 또는 HTTPS URL (선택 옵션으로 SHA-256 형식의 체크섬을 사용) 예: **https://mirror.openshift.com/images/rhcos-<version>-vmware.<architecture>.ova**.
platform.vsphere.osDisk.diskSizeGB	디스크 크기(GB)입니다.	정수
platform.vsphere.cpus	가상 머신을 할당할 총 가상 프로세서 코어 수입니다. **platform.vsphere.cpus** 의 값은 **platform.vsphere.coresPerSocket** 값의 여러 개여야 합니다.	정수
platform.vsphere.coresPerSocket	가상 머신의 소켓당 코어 수입니다. 가상 머신의 가상 소켓 수는 **platform.vsphere.cpus/platform.vsphere.coresPerSocket** 입니다. 컨트롤 플레인 노드 및 작업자 노드의 기본값은 각각 **4** 및 **2** 입니다.	정수
platform.vsphere.memoryMB	가상 머신의 메모리 크기(MB)입니다.	정수

18.4.10.2. 설치 관리자 프로비저닝 VMware vSphere 클러스터의 샘플 install-config.yaml 파일

install-config.yaml 파일을 사용자 지정하여 OpenShift Container Platform 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.

```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 3
  platform:
    vsphere: (4)
      cpus: 2
      coresPerSocket: 2
      memoryMB: 8192
      osDisk:
        diskSizeGB: 120
controlPlane: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3
  platform:
    vsphere: (7)
      cpus: 4
      coresPerSocket: 2
      memoryMB: 16384
      osDisk:
        diskSizeGB: 120
metadata:
  name: cluster (8)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  vsphere:
    vcenter: your.vcenter.server
    username: username
    password: password
    datacenter: datacenter
    defaultDatastore: datastore
    folder: folder
    network: VM_Network
    cluster: vsphere_cluster_name (9)
    apiVIP: api_vip
    ingressVIP: ingress_vip
```

```
fips: false
pullSecret: '{"auths": ...}'
sshKey: 'ssh-ed25519 AAAA...'
```

1

클러스터의 기본 도메인입니다. 모든 **DNS** 레코드는 이 기본 도메인의 하위 도메인이어야 하며 클러스터 이름을 포함해야 합니다.

2 5

controlPlane 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

3 6

동시 멀티스레딩 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. 일부 클러스터 시스템에서 동시 멀티스레딩을 비활성화할 경우에는 해당 멀티스레딩을 모든 클러스터 시스템에서 비활성화해야 합니다.

중요

동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다. 동시 멀티스레딩을 비활성화하는 경우 시스템은 **8**개 이상의 **CPU**와 **32GB**의 **RAM**을 사용해야 합니다.

4 7

선택사항: 컴퓨팅 및 컨트롤 플레인 시스템의 시스템 풀 매개변수에 대한 추가 구성을 제공하십시오.

8

DNS 레코드에 지정한 클러스터 이름입니다.

9

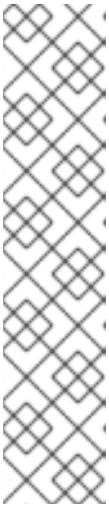
OpenShift Container Platform 클러스터를 설치할 **vSphere** 클러스터입니다. 설치 프로그램은 **vSphere** 클러스터의 루트 리소스 풀을 기본 리소스 풀로 사용합니다.

18.4.10.3. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 **HTTP** 또는 **HTTPS** 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 **OpenShift Container Platform** 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 **API**에 대한 호출을 포함하여 모든 클러스터 발신(**Egress**) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.



참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), **Google Cloud Platform (GCP)**, **Microsoft Azure** 및 **Red Hat OpenStack Platform (RHOSP)**에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1

2

클러스터 외부에서 **HTTPS** 연결을 구축하는 데 사용할 프록시 **URL**입니다.

3

대상 도메인 이름, **IP** 주소 또는 프록시에서 제외할 기타 네트워크 **CIDR**로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. ***를 사용하여 모든 대상에 대해 프록시를 바이패스합니다. **vCenter**의 **IP** 주소와 해당 시스템에 사용하는 **IP** 범위를 포함해야 합니다.

4

이 값을 제공하면 설치 프로그램에서 추가 **CA** 인증서를 보유할 **openshift -config** 네임스페이스에 **user-ca- bundle** 이라는 구성 맵을 생성합니다. **additionalTrustBundle** 및 하나 이상의 프록시 설정을 제공하는 경우 프록시 오브젝트는 **trustedCA** 필드의 **user-ca-bundle** 구성 맵을 참조하도록 구성됩니다. 그러면 **Cluster Network Operator**에서 **trustedCA** 매개변수에 대해 지정된 콘텐츠를 **RHCOS** 신뢰 번들과 병합하는 **trusted-ca-bundle** 구성 맵을 생성합니다. 프록시의 **ID** 인증서를 **RHCOS** 트러스트 번들에 있는 기관에서 서명하지 않은 경우 **additionalTrustBundle** 필드가 있어야 합니다.

참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 **OpenShift Container Platform**을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 **nil**이 됩니다.

참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

18.4.11. 네트워크 구성 단계

OpenShift Container Platform을 설치하기 전에 네트워크 구성을 사용자 지정할 수 있습니다.

1 단계

매니페스트 파일을 생성하기 전에 **install-config.yaml** 파일에서 다음 네트워크 관련 필드를 사용자 지정할 수 있습니다.

- **networking.networkType**
- **networking.clusterNetwork**
- **networking.serviceNetwork**
- **networking.machineNetwork**

이러한 필드에 대한 자세한 내용은 *설치 구성 매개 변수*에서 참조하십시오.

참고

기본 **NIC**가 상주하는 **CIDR**과 일치하도록 **networking.machineNetwork**를 설정합니다.

2 단계

openshift-install create manifests 를 실행하여 매니페스트 파일을 생성한 후 수정할 필드로 사용자 지정된 **Cluster Network Operator** 매니페스트를 정의할 수 있습니다. 매니페스트를 사용하여 고급 네트워크 구성을 지정할 수 있습니다.

install-config.yaml 파일에서 **1** 단계에 지정된 값을 **2** 단계에서 재정의할 수 없습니다. 그러나 **2** 단계에서 클러스터 네트워크 공급자를 추가로 사용자 지정할 수 있습니다.

18.4.12. 고급 네트워크 구성 지정

클러스터 네트워크 제공자의 고급 네트워크 구성을 사용하여 클러스터를 기존 네트워크 환경에 통합할 수 있습니다. 클러스터를 설치하기 전에만 고급 네트워크 구성을 지정할 수 있습니다.

중요

설치 프로그램에서 생성한 **OpenShift Container Platform** 매니페스트 파일을 수정하여 네트워크 구성을 사용자 정의하는 것은 지원되지 않습니다. 다음 절차에서와 같이 생성한 매니페스트 파일을 적용할 수 있습니다.

사전 요구 사항

- **install-config.yaml** 파일을 생성하고 수정 작업을 완료했습니다.

절차

1. 설치 프로그램이 포함된 디렉터리로 변경하고 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>**는 클러스터의 **install-config.yaml** 파일이 포함된 디렉터리의 이름을 지정합니다.

2. **<installation_directory>/ manifests/** 디렉토리에 **cluster-network-03-config.yml**이라는 **stub** 매니페스트 파일을 만듭니다.

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
```

3. 다음과 같이 **cluster-network-03-config.yml** 파일에서 클러스터의 고급 네트워크 구성을 지정합니다.

OpenShift SDN 네트워크 공급자에 대한 다른 **VXLAN** 포트 지정

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
```

```
  defaultNetwork:
    openshiftSDNConfig:
      vxlanPort: 4800
```

OVN-Kubernetes 네트워크 공급자의 IPsec 활성화

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    ovnKubernetesConfig:
      ipsecConfig: {}
```

4. 선택사항: manifests/cluster-network-03-config.yml 파일을 백업합니다. 설치 프로그램은 Ignition 구성 파일을 생성할 때 manifests/ 디렉터리를 사용합니다.

18.4.13. CNO(Cluster Network Operator) 구성

클러스터 네트워크의 구성은 CNO(Cluster Network Operator) 구성의 일부로 지정되며 cluster라는 이름의 CR(사용자 정의 리소스) 오브젝트에 저장됩니다. CR은 operator.openshift.io API 그룹에서 Network API의 필드를 지정합니다.

CNO 구성은 Network.config.openshift.io API 그룹의 Network API에서 클러스터 설치 중에 다음 필드를 상속하며 이러한 필드는 변경할 수 없습니다.

clusterNetwork

Pod IP 주소가 할당되는 IP 주소 풀입니다.

serviceNetwork

서비스를 위한 IP 주소 풀입니다.

defaultNetwork.type

OpenShift SDN 또는 OVN-Kubernetes와 같은 클러스터 네트워크 공급자입니다.

cluster라는 CNO 오브젝트에서 defaultNetwork 오브젝트의 필드를 설정하여 클러스터의 클러스터 네트워크 공급자 구성을 지정할 수 있습니다.

18.4.13.1. CNO(Cluster Network Operator) 구성 오브젝트

CNO(Cluster Network Operator)의 필드는 다음 표에 설명되어 있습니다.

표 18.31. CNO(Cluster Network Operator) 구성 오브젝트

필드	유형	설명
metadata.name	**string**	CNO 개체 이름입니다. 이 이름은 항상 **cluster**입니다.
spec.clusterNetwork	**array**	Pod IP 주소가 할당되는 IP 주소 블록과 클러스터의 각 개별 노드에 할당된 서브넷 접두사 길이를 지정하는 목록입니다. 예를 들면 다음과 같습니다. spec: clusterNetwork: - cidr: 10.128.0.0/19 hostPrefix: 23 - cidr: 10.128.32.0/19 hostPrefix: 23 매니페스트를 생성하기 전에 **install-config.yaml** 파일에서만 이 필드를 사용자 지정할 수 있습니다. 값은 매니페스트 파일에서 읽기 전용입니다.
spec.serviceNetwork	**array**	서비스의 IP 주소 블록입니다. OpenShift SDN 및 OVN-Kubernetes CNI(Container Network Interface) 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다. 예를 들면 다음과 같습니다. spec: serviceNetwork: - 172.30.0.0/14 매니페스트를 생성하기 전에 **install-config.yaml** 파일에서만 이 필드를 사용자 지정할 수 있습니다. 값은 매니페스트 파일에서 읽기 전용입니다.
spec.defaultNetwork	**object**	클러스터 네트워크의 CNI(Container Network Interface) 클러스터 네트워크 공급자를 구성합니다.

필드	유형	설명
spec.kubeProxyConfig	**object**	이 개체의 필드는 kube-proxy 구성을 지정합니다. OVN-Kubernetes 클러스터 네트워크 공급자를 사용하는 경우 kube-proxy 구성이 적용되지 않습니다.

defaultNetwork 오브젝트 구성

defaultNetwork 오브젝트의 값은 다음 표에 정의되어 있습니다.

표 18.32. defaultNetwork 오브젝트

필드	유형	설명
type	**string**	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, 클러스터 네트워크 공급자가 설치 중에 선택됩니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다. **참고** OpenShift Container Platform은 기본적으로 OpenShift SDN CNI(Container Network Interface) 클러스터 네트워크 공급자를 사용합니다.
openshiftSDNConfig	**object**	이 오브젝트는 OpenShift SDN 클러스터 네트워크 공급자에만 유효합니다.
ovnKubernetesConfig	**object**	이 오브젝트는 OVN-Kubernetes 클러스터 네트워크 공급자에만 유효합니다.

OpenShift SDN CNI 네트워크 공급자에 대한 구성

다음 표에서는 OpenShift SDN Container Network Interface (CNI) 클러스터 네트워크 공급자의 구성 필드를 설명합니다.

표 18.33. openshiftSDNConfig 오브젝트

필드	유형	설명

필드	유형	설명
mode	**string**	OpenShift SDN의 네트워크 격리 모드를 구성합니다. 기본값은 **NetworkPolicy**입니다. **Multitenant** 및 **Subnet** 값은 OpenShift Container Platform 3.x 버전과의 역호환할 수 있지만 권장되지는 않습니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
mtu	**integer**	VXLAN 오버레이 네트워크의 최대 전송 단위(MTU)입니다. 이는 기본 네트워크 인터페이스의 MTU를 기준으로 자동 탐지됩니다. 일반적으로 감지된 MTU를 재정의할 필요는 없습니다. 자동 감지 값이 예상 밖인 경우 노드의 기본 네트워크 인터페이스의 MTU가 올바른지 확인합니다. 이 옵션을 사용하여 노드의 기본 네트워크 인터페이스의 MTU 값을 변경할 수 없습니다. 클러스터에 다른 노드에 대한 다른 MTU 값이 필요한 경우 이 값을 클러스터의 가장 낮은 MTU 값보다 **50** 미만으로 설정해야 합니다. 예를 들어 클러스터의 일부 노드에는 **9001**의 MTU가 있고 일부에는 **1500**의 MTU가 있는 경우 이 값을 **1450**으로 설정해야 합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
vxlanPort	**integer**	모든 VXLAN 패킷에 사용할 포트입니다. 기본값은 **4789**입니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다. 기존 노드가 다른 VXLAN 네트워크에 속하는 가상 환경에서 실행 중인 경우에는 기본값을 변경해야 할 수도 있습니다. 예를 들어 VMware NSX-T 위에서 OpenShift SDN 오버레이를 실행할 때 두 SDN이 동일한 기본 VXLAN 포트 번호를 사용하므로 VXLAN의 대체 포트를 선택해야 합니다. AWS(Amazon Web Services)에서는 포트 **9000**과 포트 **9999** 사이에서 VXLAN의 대체 포트를 선택할 수 있습니다.

OpenShift SDN 구성 예

```
defaultNetwork:
  type: OpenShiftSDN
  openshiftSDNConfig:
    mode: NetworkPolicy
    mtu: 1450
    vxlanPort: 4789
```

OVN-Kubernetes CNI 클러스터 네트워크 공급자에 대한 구성

다음 표에서는 **OVN-Kubernetes CNI** 클러스터 네트워크 공급자의 구성 필드를 설명합니다.

표 18.34. ovnKubernetesConfig object

필드	유형	설명
mtu	**integer**	Geneve(Generic Network Virtualization Encapsulation) 오버레이 네트워크의 MTU(최대 전송 단위)입니다. 이는 기본 네트워크 인터페이스의 MTU를 기준으로 자동 탐지됩니다. 일반적으로 감지된 MTU를 재정의할 필요는 없습니다. 자동 감지 값이 예상 밖인 경우 노드의 기본 네트워크 인터페이스의 MTU가 올바른지 확인합니다. 이 옵션을 사용하여 노드의 기본 네트워크 인터페이스의 MTU 값을 변경할 수 없습니다. 클러스터에 다른 노드에 대한 다른 MTU 값이 필요한 경우, 이 값을 클러스터의 가장 낮은 MTU 값보다 **100** 미만으로 설정해야 합니다. 예를 들어, 클러스터의 일부 노드에 **9001**의 MTU가 있고 일부에는 **1500**의 MTU가 있는 경우 이 값을 **1400**으로 설정해야 합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
genevePort	**integer**	모든 Geneve 패킷에 사용할 포트입니다. 기본값은 **6081**입니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
ipsecConfig	**object**	IPsec 암호화를 활성화하려면 빈 오브젝트를 지정합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
policyAuditConfig	**object**	네트워크 정책 감사 로깅을 사용자 정의할 구성 오브젝트를 지정합니다. 설정되지 않으면 기본값 감사 로그 설정이 사용됩니다.

표 18.35. policyAuditConfig object

필드	유형	설명
rateLimit	integer	노드당 1초마다 생성할 최대 메시지 수입니다. 기본값은 초당 **20** 개의 메시지입니다.
maxFileSize	integer	감사 로그의 최대 크기(바이트)입니다. 기본값은 **50000000** 또는 **50** MB입니다.

필드	유형	설명
대상	string	다음 추가 감사 로그 대상 중 하나입니다. **libc** 호스트에서 journald 프로세스의 libc **syslog()** 함수입니다. **udp:<host>:<port>** syslog 서버입니다. **<host>:<port>**를 syslog 서버의 호스트 및 포트로 바꿉니다. **unix:<file>** **<file>**로 지정된 Unix Domain Socket 파일입니다. **null** 감사 로그를 추가 대상으로 보내지 마십시오.
syslogFacility	string	RFC5424에 정의된 **kern**과 같은 syslog 기능입니다. 기본값은 **local0**입니다.

OVN-Kubernetes 구성 예

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
    ipsecConfig: {}
```

kubeProxyConfig 오브젝트 구성

kubeProxyConfig 오브젝트의 값은 다음 표에 정의되어 있습니다.

표 18.36. kubeProxyConfig object

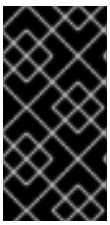
필드	유형	설명

필드	유형	설명
iptablesSyncPeriod	**string**	**iptables** 규칙의 새로 고침 간격입니다. 기본값은 **30s**입니다. 유효 접미사로 **s**, **m**, **h**가 있으며, 자세한 설명은 Go **time** 패키지 문서를 참조하십시오.  **참고** OpenShift Container Platform 4.3 이상에서는 성능이 개선되어 더 이상 **iptablesSyncPeriod** 매개변수를 조정할 필요가 없습니다.

proxyArguments.iptables-min-sync-period	**array**	**iptables** 규칙을 새로 고치기 전 최소 기간입니다. 이 필드를 통해 새로 고침 간격이 너무 짧지 않도록 조정할 수 있습니다. 유효 접미사로 **s**, **m**, **h**가 있으며, 자세한 설명은 Go **time** 패키지를 참조하십시오. 기본값은 다음과 같습니다.
		
		`kubeProxyConfig:`
		`  proxyArguments:`
		`  iptables-min-sync-period:`
		`  - 0s`

18.4.14. 클러스터 배포

호환되는 클라우드 플랫폼에 **OpenShift Container Platform**을 설치할 수 있습니다.



중요

최초 설치 과정에서 설치 프로그램의 **create cluster** 명령을 한 번만 실행할 수 있습니다.

사전 요구 사항

- 클러스터를 호스팅하는 클라우드 플랫폼으로 계정을 구성합니다.
- **OpenShift Container Platform** 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

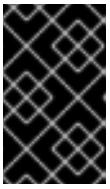
프로세스

1. 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터 배포를 초기화합니다.

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
  --log-level=info 2
```

1 **<installation_directory>** 값으로 사용자 지정한 **./install-config.yaml** 파일의 위치를 지정합니다.

2 다른 설치 세부 사항을 보려면 **info** 대신 **warn**, **debug** 또는 **error**를 지정합니다.



중요

VMC 환경에서 호스팅되는 **bastion**의 **openshift-install** 명령을 사용합니다.

참고

호스트에 구성된 클라우드 공급자 계정에 클러스터를 배포하기에 충분한 권한이 없는 경우, 설치 프로세스가 중단되고 누락된 권한을 알리는 메시지가 표시됩니다.

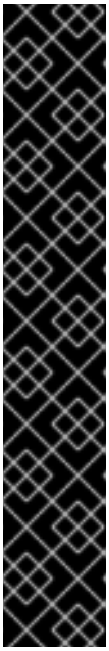
클러스터 배포가 완료되면 웹 콘솔로 연결되는 링크와 **kubeadmin** 사용자의 인증 정보가 포함된 클러스터 액세스 지침이 사용자 터미널에 표시됩니다.

출력 예

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-
ymBZj-Wt5AL"
INFO Time elapsed: 36m22s
```

참고

설치에 성공하면 클러스터 액세스 및 인증 정보도 **<installation_directory>/.openshift_install.log**로 출력됩니다.



중요

- 설치 프로그램에서 생성하는 **Ignition** 구성 파일에 **24**시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 **24**시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 **kubelet** 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(**CSR**)을 수동으로 승인해야 합니다. 자세한 내용은 ***Recovering from expired control plane certificates*** 문서를 참조하십시오.

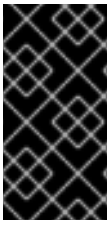
- 클러스터를 설치한 후 **24**시간에서 **22**시간까지의 인증서가 교체되기 때문에 생성된 후 **12**시간 이내에 **Ignition** 구성 파일을 사용하는 것이 좋습니다. **12**시간 이내에 **Ignition** 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

중요

설치 프로그램에서 생성되는 파일이나 설치 프로그램을 삭제해서는 안 됩니다. 클러스터를 삭제하려면 두 가지가 모두 필요합니다.

18.4.15. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 **OpenShift Container Platform**과 상호 작용하기 위해 **OpenShift CLI(oc)**를 설치할 수 있습니다. **Linux, Windows** 또는 **macOS**에 **oc**를 설치할 수 있습니다.



중요

이전 버전의 **oc**를 설치한 경우 **OpenShift Container Platform 4.9**의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 **oc**를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 Linux에서 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.
2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. OpenShift v4.9 Linux Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. oc 바이너리를 PATH에 있는 디렉터리에 배치합니다.

   PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치

다음 절차에 따라 Windows에 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.
2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3.

OpenShift v4.9 Windows Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.

4. ZIP 프로그램으로 아카이브의 압축을 풉니다.

5. oc 바이너리를 PATH에 있는 디렉터리로 이동합니다.

   PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치

다음 절차에 따라 macOS에서 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.

2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.

3. OpenShift v4.9 MacOSX Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.

4. 아카이브의 압축을 해제하고 압축을 풉니다.

5. oc 바이너리 PATH의 디렉터리로 이동합니다.

   PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
$ oc <command>
```

18.4.16. CLI를 사용하여 클러스터에 로그인

클러스터 kubeconfig 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. kubeconfig 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- oc CLI를 설치했습니다.

프로세스

1. kubeadmin 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 oc 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

18.4.17. 레지스트리 스토리지 생성

클러스터를 설치한 후 레지스트리 Operator를 위한 스토리지를 생성해야 합니다.

18.4.17.1. 설치 중 제거된 이미지 레지스트리

공유 가능한 개체 스토리지를 제공하지 않는 플랫폼에서 OpenShift Image Registry Operator는 자체적으로 Removed로 부트스트랩합니다. 이를 통해 openshift-installer가 이러한 플랫폼 유형에서 설치를 완료할 수 있습니다.

설치 후 managementState를 Removed에서 Managed로 전환하도록 Image Registry Operator 구성을 편집해야 합니다.

참고

Prometheus 콘솔은 ImageRegistryRemoved 경고를 제공합니다. 다음은 예시 경고입니다.

"이미지 레지스트리가 제거되었습니다. ImageStreamTags를 참조하는 ImageStreamTags, BuildConfigs 및 DeploymentConfigs가 예상대로 작동하지 않을 수 있습니다. 스토리지를 구성하고 configs.imageregistry.operator.openshift.io를 편집하여 구성을 Managed 상태로 업데이트하십시오."

18.4.17.2. 이미지 레지스트리 스토리지 구성

기본 스토리지를 제공하지 않는 플랫폼에서는 처음에 Image Registry Operator를 사용할 수 없습니다. 설치한 후에 스토리지를 사용하도록 레지스트리를 구성하여 Registry Operator를 사용 가능하도록 만들어야 합니다.

프로덕션 클러스터에 필요한 영구 볼륨을 구성하는 과정의 지침이 표시됩니다. 해당하는 경우, 프로덕션 환경 외 클러스터에서만 사용할 수 있는 저장 위치로서 빈 디렉터리를 구성하는 과정의 지침이 표시됩니다.

업그레이드 중에 Recreate 롤아웃 전략을 사용하여 이미지 레지스트리의 블록 스토리지 유형 사용을 허용하기 위한 추가 지침이 제공됩니다.

18.4.17.2.1. VMware vSphere용 레지스트리 스토리지 구성

클러스터 관리자는 설치한 후 스토리지를 사용하도록 레지스트리를 구성해야 합니다.

사전 요구 사항

- 클러스터 관리자 권한이 있어야 합니다.
- VMware vSphere에 클러스터가 있어야 합니다.
- Red Hat OpenShift Container Storage와 같이 클러스터용 영구 스토리지 프로비저닝.

중요

OpenShift Container Platform은 복제본이 하나만 있는 경우 이미지 레지스트리 스토리지에 대한 ReadWriteOnce 액세스를 지원합니다. ReadWriteOnce 액세스에서는 레지스트리가 Recreate 롤아웃 전략을 사용해야 합니다. 두 개 이상의 복제본으로 고 가용성을 지원하는 이미지 레지스트리를 배포하려면 ReadWriteMany 액세스가 필요합니다.

- "100Gi" 용량이 필요합니다.

중요

테스트 결과, RHEL의 NFS 서버를 핵심 서비스용 스토리지 백엔드로 사용하는 데 문제가 있는 것으로 나타납니다. 여기에는 OpenShift Container Registry and Quay, 스토리지 모니터링을 위한 Prometheus, 로깅 스토리지를 위한 Elasticsearch가 포함됩니다. 따라서 RHEL NFS를 사용하여 핵심 서비스에서 사용하는 PV를 백업하는 것은 권장되지 않습니다.

마켓플레이스의 다른 NFS 구현에는 이러한 문제가 나타나지 않을 수 있습니다. 이러한 OpenShift Container Platform 핵심 구성 요소에 대해 완료된 테스트에 대한 자세한 내용은 개별 NFS 구현 공급업체에 문의하십시오.

프로세스

1. 스토리지를 사용하도록 레지스트리를 구성하기 위해 configs.imageregistry/cluster 리소스에서 spec.storage.pvc를 변경합니다.

참고

공유 스토리지를 사용할 때 보안 설정을 확인하여 외부에서의 액세스를 방지합니다.

2. 레지스트리 **pod**가 없는지 확인합니다.

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

출력 예

```
No resourses found in openshift-image-registry namespace
```

참고

출력에 레지스트리 **Pod**가 있는 경우 이 절차를 계속할 필요가 없습니다.

3. 레지스트리 구성을 확인합니다.

```
$ oc edit configs.imageregistry.operator.openshift.io
```

출력 예

```
storage:
  pvc:
    claim: 1
```

1

4. clusteroperator 상태를 확인합니다.

```
$ oc get clusteroperator image-registry
```

출력 예

```
NAME             VERSION                             AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                                 True        False         False     6h50m
```

18.4.17.2.2. VMware vSphere용 블록 레지스트리 스토리지 구성

클러스터 관리자로서 업그레이드 중에 이미지 레지스트리가 vSphere VMDK(Virtual Machine Disk)와 같은 블록 스토리지 유형을 사용할 수 있도록 허용하기 위해 Recreate 롤아웃 전략을 사용할 수 있습니다.

중요

블록 스토리지 볼륨이 지원되지만 프로덕션 클러스터에서 이미지 레지스트리와 함께 사용하는 것은 권장되지 않습니다. 레지스트리가 블록 스토리지에 구성된 설치는 레지스트리가 둘 이상의 복제본을 가질 수 없기 때문에 가용성이 높지 않습니다.

프로세스

1. 이미지 레지스트리 스토리지를 블록 스토리지 유형으로 설정하려면 Recreate 롤아웃 전략을 사용하고 복제본 1개 만으로 실행되도록 레지스트리를 패치합니다.

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. 블록 스토리지 장치에 PV를 프로비저닝하고 해당 볼륨의 PVC를 생성합니다. 요청된 블록 볼륨은 RWO(ReadWriteOnce) 액세스 모드를 사용합니다.

   a. VMware vSphere PersistentVolumeClaim 개체를 정의하려면 다음 내용이 포함된 pvc.yaml 파일을 생성합니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage (1)
  namespace: openshift-image-registry (2)
spec:
  accessModes:
  - ReadWriteOnce (3)
  resources:
    requests:
      storage: 100Gi (4)
```

(1) PersistentVolumeClaim 개체를 표시하는 고유한 이름입니다.

(2) PersistentVolumeClaim 오브젝트의 네임스페이스로 openshift-image-registry입니다.

(3) 영구 볼륨 클레임의 액세스 모드입니다. ReadWriteOnce를 사용하면 단일 노드에서 읽기 및 쓰기 권한으로 볼륨을 마운트할 수 있습니다.

(4) 영구 볼륨 클레임의 크기입니다.

b. 파일에서 PersistentVolumeClaim 오브젝트를 만듭니다.

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 올바른 PVC를 참조하도록 레지스트리 설정을 편집합니다.

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

출력 예

```
storage:
  pvc:
    claim: (1)
```

-

1

사용자 지정 **PVC**를 만들면 **image-registry-storage PVC**의 기본 자동 생성을 위해 **claim** 필드를 비워둘 수 있습니다.

올바른 **PVC**를 참조하도록 레지스트리 스토리지를 구성하는 방법은 **vSphere** 용 레지스트리 구성을 참조하십시오.

18.4.18. VMware vSphere 볼륨 백업

OpenShift Container Platform은 새 볼륨을 독립 영구 디스크로 프로비저닝하여 클러스터의 모든 노드에서 볼륨을 자유롭게 연결 및 분리합니다. 결과적으로 스냅샷을 사용하는 볼륨을 백업하거나 스냅샷에서 볼륨을 복원할 수 없습니다. 자세한 내용은 스냅 샷 제한 을 참조하십시오.

프로세스

영구 볼륨의 백업을 생성하려면 다음을 수행합니다.

1. 영구 볼륨을 사용중인 애플리케이션을 중지합니다.
2. 영구 볼륨을 복제합니다.
3. 애플리케이션을 다시 시작합니다.
4. 복제된 볼륨의 백업을 만듭니다.
5. 복제된 볼륨을 삭제합니다.

18.4.19. 스틸 클럭 계정

기본적으로 설치 프로그램은 스틸 클럭 회계 매개 변수 (**stealclock.enabled**)를 활성화하지 않고 클러스터의 가상 머신을 프로비저닝합니다. 스틸링 클럭 계정을 사용하면 클러스터 문제 해결에 도움이 될 수

있습니다. 클러스터가 배포된 후 vSphere Client를 사용하여 각 가상 머신에서 이 매개변수를 활성화합니다.

자세한 내용은 이 Red Hat 기술 자료 문서 를 참조하십시오.

18.4.20. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

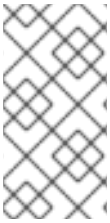
18.4.21. 다음 단계

- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.
- 레지스트리를 설정하고 레지스트리 스토리지를 구성합니다.
- 선택 사항: vSphere Problem Detector Operator에서 이벤트를 보고 클러스터에 권한 또는 스토리지 구성 문제가 있는지 확인합니다.

18.5. 제한된 네트워크에서 VMC에 클러스터 설치

OpenShift Container Platform 4.9 버전에서는 AWS의 VMC(VMware Cloud)에 배포하여 제한된 네트워크의 VMware vSphere 인프라에 클러스터를 설치할 수 있습니다.

OpenShift Container Platform 배포를 위해 VMC 환경을 구성한 후 VMC 환경에 공동 배치된 bastion 관리 호스트에서 OpenShift Container Platform 설치 프로그램을 사용합니다. 설치 프로그램 및 컨트롤 플레인은 OpenShift Container Platform 클러스터에 필요한 리소스 배포 및 관리 프로세스를 자동화합니다.

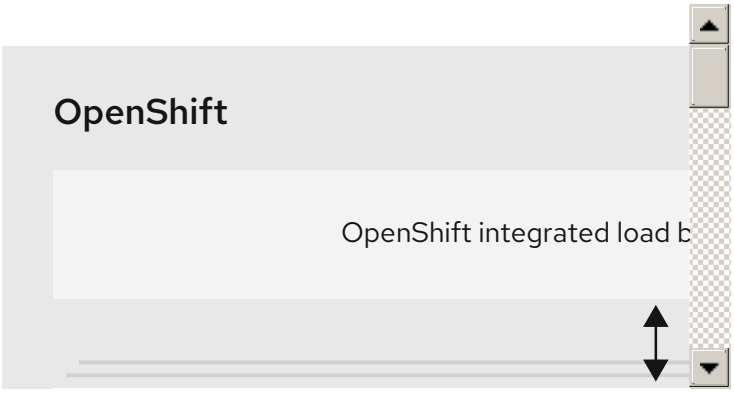


참고

OpenShift Container Platform은 단일 VMware vCenter에만 클러스터 배포를 지원합니다. 여러 vCenter에 머신/머신 세트가 있는 클러스터 배포는 지원되지 않습니다.

18.5.1. vSphere용 VMC 설정

AWS 호스팅된 vSphere 클러스터에 VMware Cloud (VMC)의 OpenShift Container Platform을 설치하여 하이브리드 클라우드 전반에 애플리케이션을 온프레미스 및 오프프레미스 모두 배포 및 관리할 수 있습니다.



VMware vSphere에 OpenShift Container Platform을 설치하기 전에 VMC 환경에서 여러 옵션을 구성해야 합니다. VMC 환경에 다음과 같은 사전 요구 사항이 있는지 확인합니다.

- 비독점 DHCP 사용, NSX-T 네트워크 세그먼트 및 서브넷을 생성합니다. 서브넷에서 다른 VM(가상 머신)을 호스팅할 수 있지만 OpenShift Container Platform 배포에서는 8개 이상의 IP 주소를 사용할 수 있어야 합니다.
- DHCP 범위 외부에서 두 개의 IP 주소를 할당하고 역방향 DNS 레코드로 구성합니다.
  - 할당된 IP 주소를 가리키는 api.<cluster_name>.<base_domain>의 DNS 레코드입니다.
  - 할당된 IP 주소를 가리키는 *.apps.<cluster_name>.<base_domain>의 DNS 레코드입

니다.

- 다음 방화벽 규칙을 설정합니다.
  - 포트 443의 설치 호스트와 소프트웨어 정의 데이터 센터(SDDC) 관리 네트워크 간의 ANY:ANY 방화벽 규칙입니다. 이를 통해 배포 중에 RHCOS(Red Hat Enterprise Linux CoreOS) OVA를 업로드할 수 있습니다.
  - OpenShift Container Platform 컴퓨팅 네트워크와 인터넷 간의 HTTPS 방화벽 규칙입니다. 이 연결을 통해 OpenShift Container Platform은 노드, PVC(영구 볼륨 클레임) 및 기타 리소스를 프로비저닝 및 관리하기 위해 vCenter과 통신할 수 있습니다.
- OpenShift Container Platform을 배포하려면 다음 정보가 있어야 합니다.
  - OpenShift Container Platform 클러스터에 vmc-prod-1로 로그인합니다.
  - companyname.com과 같은 기본 DNS 이름입니다.
  - 기본값을 사용하지 않는 경우 Pod 네트워크 CIDR 및 서비스 네트워크 CIDR을 식별해야 하며, 기본적으로 10.128.0.0/14 및 172.30.0.0/16으로 설정됩니다. 이러한 CIDR은 pod-to-pod 및 pod-to-service 통신에 사용되며 외부에서 액세스할 수 없지만 조직의 기존 서브넷과 중복되어서는 안 됩니다.
  - 다음 vCenter 정보를 참조하십시오:
    - vCenter 호스트 이름, 사용자 이름 및 암호
    - 데이터 센터 이름 (예: SDDC-Datacenter)
    - Cluster-1과 같은 클러스터 이름
    - 네트워크 이름

- WorkloadDatastore와 같은 데이터 저장소 이름

참고

클러스터 설치가 완료된 후 vSphere 클러스터를 VMC Compute-ResourcePool 리소스 풀로 이동하는 것이 좋습니다.

- VMC에 bastion으로 배포된 Linux 기반 호스트입니다.
  - bastion 호스트는 RHEL(Red Hat Enterprise Linux) 또는 기타 Linux 기반 호스트일 수 있습니다. 인터넷에 연결되어 있어야 하고 OVA를 ESXi 호스트에 업로드하는 기능이 있어야 합니다.
  - OpenShift CLI 툴을 bastion 호스트에 다운로드하여 설치합니다.
    - openshift-install 설치 프로그램
    - OpenShift CLI(oc) 툴

참고

Kubernetes 용 VMware NCP (NSX Conrainer Plugin)dOpen에는 사용할 수 없으며 NSX는 OpenShift SDN으로 사용되지 않습니다. 현재 VMC에서 사용할 수 있는 NSX 버전은 OpenShift Container Platform에서 인증된 NCP 버전과 호환되지 않습니다.

그러나 NSX DHCP 서비스는 전체 스택 자동화된 OpenShift Container Platform 배포와 vSphere와의 Machine API 통합을 통해 수동 또는 자동으로 프로비저닝된 노드를 사용하여 가상 머신 IP 관리에 사용됩니다. 또한 OpenShift Container Platform 클러스터와 bastion 호스트와 VMC vSphere 호스트 사이에서 액세스할 수 있도록 NSX 방화벽 규칙이 생성됩니다.

18.5.1.1. VMC Sizer 툴

AWS의 VMware Cloud는 AWS 베어메탈 인프라 상단에 구축됩니다. 이는 AWS 네이티브 서비스를 실행하는 것과 동일한 베어 메탈 인프라입니다. AWS 소프트웨어 정의 데이터 센터(SDDC)의 VMware 클라우드가 배포되면 이러한 물리적 서버 노드를 사용하고 단일 테넌트 방식으로 VMware ESXi 하이퍼바

이저를 실행할 수 있습니다. 즉, VMC를 사용하는 모든 사용자가 물리적 인프라에 액세스할 수 없습니다. 가상 인프라를 호스팅하는데 필요한 물리적 호스트 수를 고려해야 합니다.

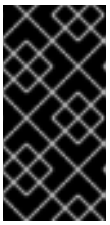
이를 확인하기 위해 VMware는 AWS Sizer에서 VMC를 제공합니다. 이 툴을 사용하면 VMC에서 호스팅할 리소스를 정의할 수 있습니다.

- 워크로드 유형
- 총 가상 머신 수
- 사양 정보는 다음과 같습니다:
  - 스토리지 요구사항
  - vCPU
  - vRAM
  - 오버커밋 비율

이러한 세부 정보를 사용하여 VMware 모범 사례를 기반으로 보고서를 생성하고 클러스터 구성과 필요한 호스트 수를 권장합니다.

18.5.2. vSphere 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.
- 미러 호스트에 레지스트리를 생성하고 사용 중인 OpenShift Container Platform 버전의 imageContentSources 데이터를 가져옵니다.

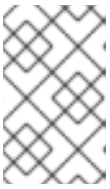


중요

미러 호스트에 설치 미디어가 있으므로 해당 컴퓨터를 사용하여 모든 설치 단계를 완료하십시오.

- 블록 레지스트리 스토리지가 프로비저닝되어 있습니다. 자세한 내용은 영구 저장 장치 이해를 참조하십시오.

- 방화벽을 사용하며 Telemetry 서비스를 사용할 예정인 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성해야 합니다.



참고

프록시를 구성하는 경우 해당 사이트 목록도 검토해야 합니다.

18.5.3. 네트워크가 제한된 환경에서의 설치 정보

OpenShift Container Platform 4.9에서는 소프트웨어 구성 요소를 받기 위한 인터넷 접속이 필요하지 않은 설치를 수행할 수 있습니다. 제한된 네트워크 설치는 클러스터를 설치하는 클라우드 플랫폼에 따라 설치 관리자 프로비저닝 인프라 또는 사용자 프로비저닝 인프라를 사용하여 완료할 수 있습니다.

클라우드 플랫폼에 제한된 네트워크 설치를 수행하는 방법을 선택해도 클라우드 API에 액세스는 가능해야 합니다. Amazon Web Service의 Route 53 DNS 및 IAM 서비스와 같은 일부 클라우드 기능에는 인터넷 액세스가 필요합니다. 사용 중인 네트워크에 따라 베어메탈 하드웨어 또는 VMware vSphere에 설치하기 위해 필요한 인터넷 액세스가 줄어들 수 있습니다.

제한된 네트워크 설치를 완료하려면 OpenShift Container Platform 레지스트리의 내용을 미러링하고 설치 미디어를 포함할 레지스트리를 생성해야 합니다. 인터넷과 폐쇄 네트워크에 모두 액세스하거나 제한 사항을 따르는 다른 방법을 통해 미러 호스트에 레지스트리를 생성할 수 있습니다.

18.5.3.1. 추가 제한

제한된 네트워크의 클러스터에는 다음과 같은 추가 제한이 있습니다.

- ClusterVersion 상태에 사용 가능한 업데이트를 검색할 수 없음 오류가 포함되어 있습니다.

-

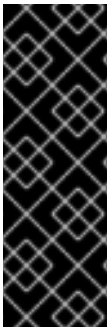
기본적으로 필요한 이미지 스트림 태그에 액세스할 수 없기 때문에 개발자 카탈로그의 내용을 사용할 수 없습니다.

18.5.4. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하기 위해 필요한 이미지를 받으려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 **Telemetry** 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.



중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

18.5.5. VMware vSphere 인프라 요구사항

사용하는 구성 요소의 요구사항을 충족하는 **VMware vSphere** 버전 **6** 또는 **7** 인스턴스에 **OpenShift Container Platform** 클러스터를 설치해야 합니다.

표 **18.37. vSphere** 가상 환경의 버전 요구 사항

가상 환경 제품	필요한 버전
VM 하드웨어 버전	13 이상
vSphere ESXi hosts	6.5 이상

가상 환경 제품	필요한 버전
vCenter 호스트	6.5 이상

중요

이제 VMware vSphere 버전 6.7U2 또는 이전 버전에 클러스터를 설치하는 것이 더 이상 사용되지 않으며 가상 하드웨어 버전 13은 더 이상 사용되지 않습니다. 이러한 버전은 여전히 완전히 지원되지만 향후 OpenShift Container Platform 버전에서는 지원이 제거됩니다. 이제 OpenShift Container Platform의 vSphere 가상 머신의 하드웨어 버전 15가 기본값이 되었습니다. vSphere 노드의 하드웨어 버전을 업데이트하려면 " vSphere에서 실행되는 노드에서 하드웨어 업데이트" 문서를 참조하십시오.

표 18.38. VMware 구성 요소에 지원되는 최소 vSphere 버전

구성 요소	지원되는 최소 버전	설명
하이퍼바이저	HW 버전 13이 포함된 vSphere 6.5 이상	이 버전은 RHCOS(Red Hat Enterprise Linux CoreOS)가 지원하는 최소 버전입니다. Red Hat Enterprise Linux 8 지원 하이퍼바이저 목록을 참조하십시오.
In-tree 드라이버가 있는 스토리지	vSphere 6.5 이상	이 플러그인은 OpenShift Container Platform에 포함된 vSphere용 인트리 스토리지 드라이버를 사용하여 vSphere 스토리지를 생성합니다.

vSphere 버전 6.5 인스턴스를 사용하는 경우 OpenShift Container Platform을 설치하기 전에 6.7U3 또는 7.0로 업그레이드하는 것이 좋습니다.

중요

OpenShift Container Platform을 설치하기 전에 ESXi 호스트의 시간이 동기화되었는지 확인해야 합니다. VMware 문서에서 Edit Time Configuration for a Host를 참조하십시오.

18.5.6. 네트워크 연결 요구사항

OpenShift Container Platform 클러스터 구성 요소가 통신할 수 있도록 시스템 간 네트워크 연결을 구성해야 합니다.

필요한 네트워크 포트에 대한 다음 세부 정보를 검토합니다.

표 18.39. 모든 시스템 간 통신에 사용되는 포트

프로토콜	포트	설명
ICMP	해당 없음	네트워크 연결성 테스트
TCP	**1936**	메트릭
	9000-9999	**9100-9101** 포트의 노드 내보내기 및 **9099** 포트의 Cluster Version Operator를 포함한 호스트 수준 서비스.
	10250-10259	Kubernetes에서 예약하는 기본 포트
	10256	openshift-sdn
UDP	**4789**	VXLAN(Virtual extensible LAN)
	6081	Geneve
	9000-9999	**9100-9101** 포트의 노드 내보내기를 포함한 호스트 수준 서비스.
	500	IPsec IKE 패킷
	4500	IPsec NAT-T 패킷
TCP/UDP	**30000-32767**	Kubernetes 노드 포트
ESP	해당 없음	IPsec Encapsulating Security Payload (ESP)

표 18.40. 모든 시스템과 컨트롤 플레인 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**6443**	Kubernetes API

표 18.41. 컨트롤 플레인 머신 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**2379-2380**	etcd 서버 및 피어 포트

추가 리소스

- vSphere 노드의 하드웨어 버전을 업데이트하려면 vSphere에서 실행 중인 노드에서 하드웨어 업데이트를 참조하십시오.

18.5.7. vCenter 요구사항

설치 관리자가 프로비저닝하는 인프라를 사용하는 vCenter에 OpenShift Container Platform 클러스터를 설치하기 전에 환경을 준비해야 합니다.

필요한 vCenter 계정 권한

vCenter에 OpenShift Container Platform 클러스터를 설치하려면 설치 프로그램에서 필요한 리소스를 읽고 생성할 수 있는 권한이 있는 계정에 액세스해야 합니다. 필요한 모든 권한에 액세스할 수 있는 가장 간단한 방법은 글로벌 관리 권한이 있는 계정을 사용하는 것입니다.

글로벌 관리 권한이 있는 계정을 사용할 수 없는 경우 OpenShift Container Platform 클러스터 설치에 필요한 권한을 부여하려면 역할을 생성해야 합니다. 대부분의 권한이 항상 필요하지만 기본 동작인 vCenter 인스턴스에 OpenShift Container Platform 클러스터를 포함하는 폴더를 프로비저닝할 설치 프로그램인 경우에만 필요합니다. 필요한 권한을 부여하려면 지정된 오브젝트에 대해 vSphere 역할을 생성하거나 수정해야 합니다.

설치 프로그램이 vSphere 가상 머신 폴더를 생성하는 경우 추가 역할이 필요합니다.

예 18.10. vSphere API에 설치에 필요한 역할 및 권한

역할의 vSphere 개체	필요한 경우	vSphere API에서 필요한 권한

역할의 vSphere 개체	필요한 경우	vSphere API에서 필요한 권한
vSphere vCenter	Always	CNS .**Searchable** **InventoryService.Tagging.AttachTag** **InventoryService.Tagging.CreateCategory** **InventoryService.Tagging.CreateTag** **InventoryService.Tagging.Delete**Tag **InventoryService.tagged.DeleteTag** **InventoryService.Tagging.EditCategory** **InventoryService.Tagging.EditTag**ECDHE**Session** **ValidateSession**Forwarded**StorageProfile.Update**Forwarded**StorageProfile.View**
vSphere vCenter Cluster	Always	**Host.Config.Storage** **Resource.AssignVMToPool** **VApp.AssignResourcePool** **VApp.Import** **VirtualMachine.Config.AddNewDisk**
vSphere Datastore	Always	**Datastore.AllocateSpace** **Datastore.LoadBalancer** **Datastore.FileManagement** **InventoryService.Tagging.ObjectAttachable**
vSphere Port Group	Always	**Network.Assign**

역할의 vSphere 개체	필요한 경우	vSphere API에서 필요한 권한
가상 머신 폴더	Always	**InventoryService.ume.ObjectAttachable Resource.AssignVMToPool VApp.Import VirtualMachine.Config.Add ExistingDisk VirtualMachine.Config.Add NewDisk VirtualMachine.Config.Add RemoveDevice VirtualMachine.Config.Advanced Config VirtualMachine.Config.Annotation ECDHEVirtualMachine .Config.DiskExtend** Forwarded VirtualMachine**.Config.** DiskExtend Forwarded**VirtualMachine.Config.DiskLease VirtualMachine.Config.ReCreate** VolumeClaim.Config**.MemoryDriver.**

역할의 vSphere 개체	필요한 경우	vSphere API에서 필요한 권한
vSphere vCenter Datacenter	설치 프로그램이 가상 머신 폴더를 생성하는 경우	**InventoryService.Tagging.ObjectAttachable Resource.AssignVMToPool VApp.Import VirtualMachine.Config.AddExistingDisk VirtualMachine.Config.AddNewDisk VirtualMachine.Config.AddRemoveDevice VirtualMachine.Config.AdvancedConfig VirtualMachine.Config.AnnotationECDHEVirtualMachine.Config.DiskExtend**Forwarded VirtualMachine**.**Config**.DiskExtend ForwardedVirtualMachine.Config.DiskLeaseVirtualMachine.Config.ReCreate VolumeClaim.Config.Reset**defined**.**

예 18.11. vCenter 그래픽 사용자 인터페이스(GUI)에 설치하는 데 필요한 역할 및 권한

역할의 vSphere 개체	필요한 경우	vCenter GUI에서 필요한 권한
vSphere vCenter	Always	CNS **.Searchable "vSphere agged""Assign or unassign vSphere Tag" "vSphere tagging" "vSphere" "vSphere" "vSphere"CreatevSphere Tag" vSphere 태그""vSphere" 삭제 태그 지정" "vSphere Assignment""Edit vSphere Tag" "vSphere Tagging" "vSphere 태그" 세션" "Profile 기반 스토리지""Profile 기반 스토리지 업데이트""Profile 기반 스토리지"Profile 중심 스토리지 업데이트"Profile 기반 스토리지 업데이트"Profile 기반 스토리지 업데이트 스토리지 보기"**
vSphere vCenter Cluster	클러스터 루트에서 VM이 생성되는 경우	**Host.Configuration"Storage partition configuration" 리소스" 리소스 풀에 가상 머신을 서명" VApp"Assign resource pool" VApp.Import"Virtual machine""Change Configuration"."새 디스크 추가"**
vSphere vCenter 리소스 풀	기존 리소스 풀이 제공되는 경우	**Host.Configuration"Storage partition configuration" 리소스" 리소스 풀에 가상 머신을 서명" VApp"Assign resource pool" VApp.Import"Virtual machine""Change Configuration"."새 디스크 추가"**

역할의 vSphere 개체	필요한 경우	vCenter GUI에서 필요한 권한
vSphere Datastore	Always	**Datastore"Allocate space"** 데이터**저장소"** **데이터 저장소"** **Datastore"Low level file operations""vSphere Assignment"Assign or Unassign vSphere Tag on Object"**
vSphere Port Group	Always	**네트워크 "Assign network"**
가상 머신 폴더	Always	**"vSphere Assignment""Assign or unassign vSphere Tag on Object"** **Resource"Assign virtual machine to resource pool"**: **VApp.Import** **"Virtual machine""Change Configuration","Add existing disk""**Add existing disk**""Add new disk""Add new disk" "Virtual""** **머신 "시스템 변경", "가상 머신""** **"가상 머신""시스템 변경"** **"가상 머신""변경 구성",** **"가상 머신", 변경, CPU 수 변경"** **"가상 머신", 가상 디스크 변경", 가상 디스크 변경"** **"가상머신". 머신 변경 "구성 변경"** **"가상 머신""마이크 구성 변경"** **"가상 머신""마이크 구성 변경" "구성 변경",** **"가상 머신", 변경" 디스크 삭제"** **"가상 머신".변경 구성".Rename** **"가상 머신 변경" VIX API**로가상 머신 변경**"Gregradevirtual machine"Guestoperating system management by VIX API**

역할의 vSphere 개체	필요한 경우	vCenter GUI에서 필요한 권한
		machine". interaction" "Virtual machine". Interconnection"Power on" "Virtual machine". interactionion.Reset "Virtual machine"."Edit Inventory"."Create new" "Virtual machine"."Create existing" "Virtual machine"Edit Inventory""Create from existing" "Virtual machine"Edit Inventory""Create from existing" virtual machine".Edit Inventory""Create from existing" virtual machine".Edit Inventory""Create from existing""Virtual machine" machine".Provisioning."Clone 가상 머신" "가상 머신".Provisioning." "가상 머신".Provisioning."Deploy 템플릿"
vSphere vCenter Datacenter	설치 프로그램이 가상 머신 폴더를 생성하는 경우	"vSphere Assignment""Assign or unassign vSphere Tag on Object" Resource"Assign virtual machine to resource pool": VApp.Import "Virtual machine""Change Configuration","Add existing disk""Add existing disk""Add new disk""Add new disk" "Virtual"" 머신 "시스템 변경", "가상 머신"" "가상 머신""시스템 변경" "가상 머신""변경 구성", "가상 머신", 변경, CPU 수 변경" "가상 머신", 가상 디스크 변경", 가상 디스크 변경" "가상머신". 머신 변경 "구성 변경" "가상 머신""마이크 구성 변경" "가상 머신""마이크 구성 변경" "구성 변경", "가상 머신", 변경" 디스크 삭

역할의 vSphere 개체	필요한 경우	vCenter GUI에서 필요한 권한
		제" "가상 머신 .관리 수정".Rename "가상 머신 변경" VIX API로가상 머신 변 경"Gregradevirtual machine"Guestoperating system management by VIX API machine". interaction" "Virtual machine". Interconnection"Power on" "Virtual machine". interactionion.Reset "Virtual machine"."Edit Inventory"."Create new" "Virtual machine"."Create existing" "Virtual machine"Edit Inventory""Create from existing" "Virtual machine"Edit Inventory""Create from existing" virtual machine".Edit Inventory""Create from existing" virtual machine".Edit Inventory""Create from existing""Virtual machine" machine".Provisioning."Clone 가상 머신" "가상 머 신".Provisioning."Deploy 템플릿" "가상 머신".Provisioning" 폴더 만들기" 폴더 생성"폴더 삭제"

또한 사용자에게 일부 **ReadOnly** 권한이 필요하며 일부 역할에는 하위 오브젝트에 권한을 부여하는 사용 권한이 필요합니다. 이러한 설정은 클러스터를 기존 폴더에 설치할지 여부에 따라 달라집니다.

예 18.12. 필수 권한 및 권한 부여 설정

vSphere 오브젝트	폴더 유형	하위 항목으로 권한 부여	권한 필요
vSphere vCenter	Always	False	나열된 필수 권한
vSphere vCenter Datacenter	기존 폴더	False	**ReadOnly** 권한
	설치 프로그램은 폴더를 생성	True	나열된 필수 권한
vSphere vCenter Cluster	Always	True	나열된 필수 권한
vSphere vCenter Datastore	Always	False	나열된 필수 권한
vSphere Switch	Always	False	**ReadOnly** 권한
vSphere Port Group	Always	False	나열된 필수 권한
vSphere vCenter Virtual Machine Folder	기존 폴더	True	나열된 필수 권한

필요한 권한만으로 계정을 생성하는 방법에 대한 자세한 내용은 **vSphere** 문서에서 **vSphere 권한 및 사용자 관리 작업**을 참조하십시오.

vMotion으로 OpenShift Container Platform 사용

vSphere 환경에서 **vMotion**을 사용하려는 경우 **OpenShift Container Platform** 클러스터를 설치하기 전에 다음을 고려하십시오.

- **OpenShift Container Platform**은 일반적으로 **compute-only vMotion** 을 지원합니다. 스토리지 **vMotion**을 사용하면 문제가 발생할 수 있으며 지원되지 않습니다.

  컴퓨팅 및 컨트롤 플레인 노드의 가동 시간을 보장하기 위해 **vMotion**에 대한 **VMware** 모범 사례를 따르는 것이 좋습니다. 또한 **VMware** 유사성 방지 규칙을 사용하여 유지 관리 또는 하드웨어 문제 중에 **OpenShift Container Platform**의 가용성을 개선하는 것이 좋습니다.

  vMotion 및 유사성 방지 규칙에 대한 자세한 내용은 **vMotion 네트워킹 요구 사항** 및 **VM 유사성 방지 규칙에** 대한 **VMware vSphere** 설명서를 참조하십시오.

- **Pod**에서 **vSphere** 볼륨을 사용하는 경우 데이터 저장소를 통해 **VM**을 마이그레이션하면 수동으로 또는 스토리지 **vMotion**을 통해 **VM**을 마이그레이션하면 **OpenShift Container Platform**

PV(영구 볼륨) 오브젝트 내에서 잘못된 참조가 표시됩니다. 이러한 참조는 영향을 받는 포드가 시작되지 않게 하고 데이터 손실을 초래할 수 있습니다.

- 마찬가지로 OpenShift Container Platform은 데이터 저장소에서 VMDK의 선택적 마이그레이션을 지원하지 않으며 VM 프로비저닝 또는 PV의 동적 또는 정적 프로비저닝을 위해 데이터 저장소 클러스터의 일부 또는 PV의 동적 또는 정적 프로비저닝을 위해 데이터 저장소 클러스터의 일부인 데이터 저장소를 사용하는 경우도 있습니다.

클러스터 리소스

설치 관리자 프로비저닝 인프라를 사용하는 OpenShift Container Platform 클러스터를 배포할 때는 설치 프로그램이 vCenter 인스턴스에서 여러 리소스를 생성할 수 있어야 합니다.

표준 OpenShift Container Platform을 설치하면 다음 vCenter 리소스가 생성됩니다.

- 폴더 한 개
- 태그 카테고리 한 개
- 태그 한 개
- 가상 머신:
  - 템플릿 한 개
  - 임시 부트스트랩 노드 한 개
  - 컨트롤 플레인 노드 세 개
  - 컴퓨팅 시스템 세 개

이러한 리소스는 856GB의 스토리지를 사용하지만 부트스트랩 노드는 클러스터 설치 프로세스 중에 제거됩니다. 표준 클러스터를 사용하려면 최소 800GB의 스토리지가 필요합니다.

컴퓨팅 시스템을 더 많이 배포할수록 **OpenShift Container Platform** 클러스터는 더 많은 스토리지를 사용합니다.

클러스터 제한

사용 가능한 리소스는 클러스터마다 다릅니다. **vCenter** 내에서 가능한 클러스터 수는 주로 사용 가능한 스토리지 공간과 필요한 리소스 수에 대한 제한으로 제한됩니다. 클러스터가 생성하는 **vCenter** 리소스와 **IP** 주소 및 네트워크와 같이 클러스터를 배포하는 데 필요한 리소스에 대한 제한 사항을 모두 고려해야 합니다.

네트워킹 요구사항

네트워크에 **DHCP**를 사용하고 **DHCP** 서버가 클러스터 시스템에 영구 **IP** 주소를 제공하도록 구성되어 있는지 확인합니다. **DHCP** 서버를 사용하도록 기본 게이트웨이를 구성해야 합니다. 모든 노드는 동일한 **VLAN**에 있어야 합니다. 두 번째 **VLAN**을 **Day 2** 작업으로 사용하여 클러스터를 확장할 수 없습니다. 네트워크가 제한된 환경에서 **VM**은 노드, **PVC**(영구 볼륨 클레임) 및 기타 리소스를 프로비저닝하고 관리할 수 있도록 **vCenter**에 액세스할 수 있어야 합니다. 또한 **OpenShift Container Platform** 클러스터를 설치하기 전에 다음 네트워킹 리소스를 생성해야 합니다.

참고

클러스터의 각 **OpenShift Container Platform** 노드는 **DHCP**를 통해 검색할 수 있는 **NTP(Network Time Protocol)** 서버에 액세스할 수 있어야 합니다. **NTP** 서버 없이도 설치할 수 있습니다. 그러나 비동기 서버 클록으로 인해 **NTP** 서버가 차단하는 오류가 발생합니다.

필요한 IP 주소

설치 관리자 프로비저닝 **vSphere** 설치에는 다음 두 개의 고정 **IP** 주소가 필요합니다.

- **API** 주소는 클러스터 **API**에 액세스하는 데 사용됩니다.
- **Ingress** 주소는 클러스터 인그레스 트래픽에 사용됩니다.

OpenShift Container Platform 클러스터를 설치할 때 설치 프로그램에 이 **IP** 주소를 제공해야 합니다.

DNS 레코드

OpenShift Container Platform 클러스터를 호스팅하는 **vCenter** 인스턴스에 적절한 **DNS** 서버에서 고정 **IP** 주소 두 개의 **DNS** 레코드를 생성해야 합니다. 각 레코드에서 **<cluster_name>**은 클러스터 이름

이고 <base_domain>은 클러스터를 설치할 때 지정하는 클러스터 기본 도메인입니다. 전체 DNS 레코드는 <component>.<cluster_name>.<base_domain> 형식입니다.

표 18.42. 필수 DNS 레코드

구성 요소	레코드	설명
API VIP	api.<cluster_name>.<base_domain>.	이 DNS A/AAAA 또는 CNAME 레코드는 컨트롤 플레인 시스템의 로드 밸런서를 가리켜야 합니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.
Ingress VIP	*.apps.<cluster_name>.<base_domain>.	기본적으로 작업자 노드인 인그레스 라우터 Pod를 실행하는 시스템을 대상으로 하는 로드 밸런서를 가리키는 와일드카드 DNS A/AAAA 또는 CNAME 레코드입니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.

18.5.8. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 core 사용자의 ~/.ssh/authorized_keys 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 core로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. ./openshift-install gather 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

참고

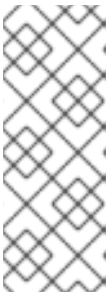
AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

절차

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 새 SSH 키의 경로 및 파일 이름(예: ~/.ssh/id_ed25519)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 '~/.ssh 디렉터리에 있는지 확인하십시오.



참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 ed25519 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 rsa 또는 ecdsa 알고리즘을 사용하는 키를 생성합니다.

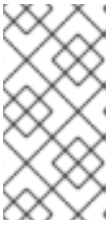
2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 ~/.ssh/id_ed25519.pub 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 ./openshift-install gather 명령을 사용하려는 경우 필요합니다.



참고

일부 배포에서는 ~/.ssh/id_rsa 및 ~/.ssh/id_dsa와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a.

ssh-agent 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4.

ssh-agent에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1

SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: ~/.ssh/id_ed25519).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- **OpenShift Container Platform**을 설치할 때 **SSH** 공개 키를 설치 프로그램에 지정합니다.

18.5.9. 시스템 신뢰에 vCenter 루트 CA 인증서 추가

설치 프로그램이 **vCenter API**에 액세스해야 하므로 **OpenShift Container Platform** 클러스터를 설치하기 전에 **vCenter**의 신뢰할 수 있는 루트 **CA** 인증서를 시스템 신뢰에 추가해야 합니다.

절차

1. **vCenter** 홈 페이지에서 **vCenter**의 루트 **CA** 인증서를 다운로드합니다. **vSphere Web Services SDK** 섹션에서 **Download trusted root CA certificates**를 클릭합니다. **<vCenter>/certs/download.zip** 파일이 다운로드됩니다.

2. **vCenter** 루트 **CA** 인증서가 포함된 압축 파일의 압축을 풉니다. 압축 파일의 내용은 다음 파일 구조와 유사합니다.

```
certs
├── lin
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
├── mac
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
└── win
    ├── 108f4d17.0.crt
    ├── 108f4d17.r1.crl
    ├── 7e757f6a.0.crt
    ├── 8e4f8471.0.crt
    └── 8e4f8471.r0.crl

3 directories, 15 files
```

3. 운영 체제 파일을 시스템 신뢰에 추가합니다. 예를 들어 **Fedora** 운영 체제에서는 다음 명령을 실행합니다.

```
# cp certs/lin/* /etc/pki/ca-trust/source/anchors
```

4. 시스템 신뢰를 업데이트합니다. 예를 들어 Fedora 운영 체제에서는 다음 명령을 실행합니다.

```
# update-ca-trust extract
```

18.5.10. 제한된 네트워크 설치를 위한 RHCOS 이미지 생성

Red Hat Enterprise Linux CoreOS(RHCOS) 이미지를 다운로드하여 제한된 네트워크 VMware vSphere 환경에 OpenShift Container Platform을 설치하십시오.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램을 가져옵니다. 제한된 네트워크 설치의 경우 프로그램은 미러 레지스트리 호스트에 있습니다.

절차

1. Red Hat Customer Portal의 제품 다운로드 페이지에 로그인합니다.

2. Version 에서 RHEL 8용 최신 OpenShift Container Platform 4.9 릴리스를 선택합니다.

중요

RHCOS 이미지는 OpenShift Container Platform 릴리스에 따라 변경되지 않을 수 있습니다. 설치하는 OpenShift Container Platform 버전과 같거나 그 이하의 버전 중 가장 최신 버전의 이미지를 다운로드해야 합니다. 사용 가능한 경우 OpenShift Container Platform 버전과 일치하는 이미지 버전을 사용합니다.

3. Red Hat Enterprise Linux CoreOS (RHCOS) - vSphere 이미지를 다운로드합니다.

4. 다운로드한 이미지를 bastion 서버에서 액세스할 수 있는 위치에 업로드합니다.

이제 이미지를 제한된 설치에 사용할 수 있습니다. OpenShift Container Platform 배포에 사용할 이미지 이름 또는 위치를 기록해 둡니다.

18.5.11. 설치 구성 파일 만들기

VMware vSphere에 설치하는 OpenShift Container Platform 클러스터를 사용자 지정할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다. 제한된 네트워크 설치의 경우, 해당 파일은 미러 호스트에 있습니다.
- 미러 레지스트리 작성 중에 생성된 imageContentSources 값이 있어야 합니다.
- 미러 레지스트리에 대한 인증서의 내용을 가져옵니다.
- RHCOS (Red Hat Enterprise Linux CoreOS) 이미지를 검색하여 액세스 가능한 위치에 업로드합니다.
- 서브스크립션 수준에서 서비스 주체 권한을 획득합니다.

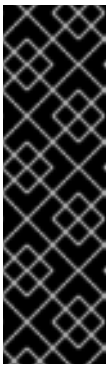
절차

1. install-config.yaml 파일을 생성합니다.

   a. 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 <installation_directory>는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.



중요

비어 있는 디렉터리를 지정합니다. 부트스트랩 **X.509** 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 **OpenShift Container Platform** 버전에서 설치 파일을 복사할 때는 주의하십시오.

b. 화면에 나타나는 지시에 따라 클라우드에 대한 구성 세부 사항을 입력합니다.

i. 선택사항: 클러스터 시스템에 액세스하는 데 사용할 **SSH** 키를 선택합니다.

참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 **OpenShift Container Platform** 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 **SSH** 키를 지정합니다.

ii. 대상 플랫폼으로 **vsphere**를 선택합니다.

iii. **vCenter** 인스턴스의 이름을 지정합니다.

iv. 클러스터를 생성하는 데 필요한 권한이 있는 **vCenter** 계정의 사용자 이름과 암호를 지정합니다.

설치 프로그램이 **vCenter** 인스턴스에 연결됩니다.

v. 연결할 **vCenter** 인스턴스에서 데이터 센터를 선택합니다.

vi. 사용할 기본 **vCenter** 데이터 저장소를 선택합니다.

vii. **OpenShift Container Platform** 클러스터를 설치할 **vCenter** 클러스터를 선택합니다. 설치 프로그램은 **vSphere** 클러스터의 루트 리소스 풀을 기본 리소스 풀로 사용합

니다.

viii. vCenter 인스턴스에서 구성한 가상 IP 주소 및 DNS 레코드가 포함된 네트워크를 선택합니다.

ix. 컨트롤 플레인 API 액세스를 위해 구성한 가상 IP 주소를 입력합니다.

x. 클러스터 인그레스용으로 구성한 가상 IP 주소를 입력합니다.

xi. 기본 도메인을 입력합니다. 이 기본 도메인은 구성한 DNS 레코드에서 사용한 것과 동일해야 합니다.

xii. 클러스터를 설명할 수 있는 이름을 입력합니다. 클러스터 이름은 구성한 DNS 레코드에서 사용한 것과 동일해야 합니다.

xiii. Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 붙여넣습니다.

2. install-config.yaml 파일에서 platform.vsphere.clusterOSImage 값을 이미지 위치 또는 이름으로 설정합니다. 예를 들면 다음과 같습니다.

```
platform:
  vsphere:
    clusterOSImage: http://mirror.example.com/images/rhcos-43.81.201912131630.0-
vmware.x86_64.ova?
sha256=ffebbd68e8a1f2a245ca19522c16c86f67f9ac8e4e0c1f0a812b068b16f7265d
```

3. 제한된 네트워크에서의 설치에 필요한 추가 정보를 제공하려면 install-config.yaml 파일을 편집합니다.

a. 레지스트리의 인증 정보를 포함하도록 pullSecret 값을 업데이트합니다.

```
pullSecret: '{"auths":{"<mirror_host_name>:5000": {"auth": "
<credentials>","email": "you@example.com"}}}'
```

<mirror_host_name>의 경우 미러 레지스트리의 인증서에 지정한 레지스트리 도메인 이름을 지정하고 <credentials>의 경우 미러 레지스트리에 base64로 인코딩된 사용자 이름

및 암호를 지정합니다.

b. additionalTrustBundle 매개변수와 값을 추가합니다.

```
additionalTrustBundle: |
  -----BEGIN CERTIFICATE-----

ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
```

값은 미러 레지스트리에 사용한 인증서 파일의 내용이어야 하며, 신뢰할 수 있는 기존 인증 기관 또는 미러 레지스트리에 대해 생성한 자체 서명 인증서일 수 있습니다.

c. 다음과 같은 이미지 콘텐츠 리소스를 추가합니다.

```
imageContentSources:
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: registry.redhat.io/ocp/release
```

이러한 값을 완료하려면 미러 레지스트리 생성 중에 기록한 **imageContentSources**를 사용하십시오.

4. 필요한 **install-config.yaml** 파일을 수정합니다. 사용 가능한 매개변수에 대한 자세한 정보는 **Installation configuration parameters** 섹션에서 확인할 수 있습니다.

5. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.

중요

install-config.yaml 파일은 설치 과정에서 사용됩니다. 이 파일을 재사용하려면 지금 백업해야 합니다.

18.5.11.1. 설치 구성 매개변수

OpenShift Container Platform 클러스터를 배포하기 전에 매개변수 값을 제공하여 클러스터를 호스

팅할 클라우드 플랫폼에서 사용자 계정을 설명하고 선택사항으로 클러스터의 플랫폼을 사용자 지정합니다. **install-config.yaml** 설치 구성 파일을 생성할 때 명령줄을 통해 필요한 매개변수 값을 제공합니다. 클러스터를 사용자 지정하면 **install-config.yaml** 파일을 수정하여 플랫폼에 대한 세부 정보를 제공할 수 있습니다.

참고

설치한 후에는 **install-config.yaml** 파일에서 이러한 매개변수를 수정할 수 없습니다.

중요

openshift-install 명령은 매개변수의 필드 이름을 검증하지 않습니다. 잘못된 이름이 지정되면 관련 파일 또는 오브젝트가 생성되지 않으며 오류가 보고되지 않습니다. 지정된 매개변수의 필드 이름이 올바른지 확인합니다.

18.5.11.1.1. 필수 구성 매개변수

필수 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 18.43. 필수 매개 변수

매개변수	설명	값
apiVersion	**install-config.yaml** 콘텐츠의 API 버전입니다. 현재 버전은 **v1**입니다. 설치 관리자가 이전 API 버전도 지원할 수 있습니다.	문자열
baseDomain	클라우드 공급자의 기본 도메인입니다. 기본 도메인은 OpenShift Container Platform 클러스터 구성 요소에 대한 경로를 생성하는 데 사용됩니다. 클러스터의 전체 DNS 이름은 **baseDomain**과 **metadata.name** 매개변수 값의 조합으로, **<metadata.name>.<baseDomain>** 형식입니다.	정규화된 도메인 또는 하위 도메인 이름(예: **example.com**).
metadata	Kubernetes 리소스 **ObjectMeta**로 **name** 매개변수만 사용합니다.	개체

매개변수	설명	값
metadata.name	클러스터의 이름입니다. 클러스터의 DNS 레코드는 **{{.metadata.name}}.{{.baseDomain}}** 형식의 모든 하위 도메인입니다.	**dev** 와 같은 소문자 및 하이픈(-)의 문자열입니다.
platform	설치를 수행할 특정 플랫폼에 대한 구성: **aws**,**baremetal**,**azure**,**gcp**, **openstack**,**ovirt**,**vsphere**, 또는 **{}**. **platform.<platform>** 매개변수에 대한 자세한 내용은 다음 표에서 사용자 플랫폼에 해당하는 정보를 참조하십시오.	개체

pullSecret	Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 가져와서 Quay.io와 같은 서비스에서 OpenShift Container Platform 구성 요소의 컨테이너 이미지 다운로드를 인증합니다.	{
		"auths":{
		"cloud.openshift.com":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		},
		"quay.io":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		}
		}
		}

18.5.11.1.2. 네트워크 구성 매개변수

기존 네트워크 인프라의 요구 사항에 따라 설치 구성을 사용자 지정할 수 있습니다. 예를 들어 클러스터 네트워크의 IP 주소 블록을 확장하거나 기본값과 다른 IP 주소 블록을 제공할 수 있습니다.

IPv4 주소만 지원됩니다.

표 18.44. 네트워크 매개변수

매개변수	설명	값

매개변수	설명	값
networking	클러스터의 네트워크의 구성입니다.	개체  **참고** 설치한 후에는 **networking** 오브젝트에서 지정된 매개변수를 수정할 수 없습니다.
networking.networkType	설치할 클러스터 네트워크 공급자 CNI(Container Network Interface) 플러그인입니다.	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, **OpenShiftSDN** 은 모든 Linux 네트워크의 CNI 공급자입니다. **OVNKubernetes** 는 Linux 및 Windows 서버를 모두 포함하는 Linux 네트워크 및 하이브리드 네트워크의 CNI 공급자입니다. 기본값은 **OpenShiftSDN**입니다.

networking.clusterNetwork	Pod의 IP 주소 블록입니다.
	기본값은 **10.128.0.0/14**이며, 호스트 접두사는 **/23**입니다.
	여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.
		개체의 배열입니다. 예를 들면 다음과 같습니다.
		`networking:`
		`  clusterNetwork:`
		`  - cidr: 10.128.0.0/14`
		`  hostPrefix: 23`

networking.clusterNetwork.cidr	**networking.clusterNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. IPv4 네트워크입니다.	CIDR(Classless Inter-Domain Routing) 표기법의 IP 주소 블록입니다. IPv4 블록의 접두사 길이는 **0** 에서 **32** 사이입니다.
networking.clusterNetwork.hostPrefix	개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하는 경우, 지정된 **cidr** 이외 **/23** 서브넷이 각 노드에 할당됩니다. **23**의 **hostPrefix** 값은 510(2^(32 - 23) - 2) Pod IP 주소를 제공합니다.	서브넷 접두사입니다. 기본값은 **23**입니다.

매개변수	설명	값
networking.serviceNetwork	서비스의 IP 주소 블록입니다. 기본값은 **172.30.0.0/16**입니다. OpenShift SDN 및 OVN-Kubernetes 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다.	CIDR 형식의 IP 주소 블록이 있는 어레이입니다. 예를 들면 다음과 같습니다. networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	시스템의 IP 주소 블록입니다. 여러 IP 주소 블록을 지정하는 경우 블록이 겹치지 않아야 합니다.	개체의 배열입니다. 예를 들면 다음과 같습니다. networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork**를 사용하는 경우 필수 항목입니다. IP 주소 블록입니다. libvirt를 제외한 모든 플랫폼의 기본값은 **10.0.0.0/16**입니다. libvirt의 기본값은 **192.168.126.0/24**입니다.	CIDR 표기법의 IP 네트워크 블록입니다. 예: **10.0.0.0/16** **참고** 기본 NIC가 상주하는 CIDR과 일치하도록 **networking.machineNetwork**를 설정합니다.

18.5.11.1.3. 선택적 구성 매개변수

선택적 설치 구성 매개변수는 다음 표에 설명되어 있습니다.

표 18.45. 선택적 매개변수

매개변수	설명	값
additionalTrustBundle	노드의 신뢰할 수 있는 인증서 스토리지에 추가되는 PEM 인코딩 X.509 인증서 번들입니다. 이 신뢰할 수 있는 번들은 프록시가 구성되었을 때에도 사용할 수 있습니다.	문자열
compute	컴퓨팅 노드를 구성하는 시스템의 구성입니다.	**MachinePool** 개체의 배열입니다.

매개변수	설명	값
compute.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 이기종 클러스터는 현재 지원되지 않으므로 모든 풀이 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열
compute.hyperthreading	컴퓨팅 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
compute.name	**compute**를 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**worker**
compute.platform	**compute**를 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 작업자 시스템을 호스팅할 클라우드 공급자를 지정합니다. 이 매개변수 값은 **controlPlane.platform** 매개변수 값과 일치해야 합니다	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
compute.replicas	프로비저닝할 컴퓨팅 시스템(작업자 시스템이라고도 함) 수입니다.	**2** 이상의 양의 정수이며, 기본값은**3**입니다.
controlPlane	컨트롤 플레인을 구성하는 시스템들의 구성입니다.	**MachinePool** 개체의 배열입니다.
controlPlane.architecture	풀에 있는 시스템의 명령어 집합 아키텍처를 결정합니다. 현재 이기종 클러스터는 지원되지 않으므로 모든 풀에서 동일한 아키텍처를 지정해야 합니다. 유효한 값은 **amd64**(기본값)입니다.	문자열

매개변수	설명	값
controlPlane.hyperthreading	컨트롤 플레인 시스템에서 동시 멀티스레딩 또는 **hyperthreading** 활성화 또는 비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다.  **중요** 동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.	**Enabled** 또는 **Disabled**
controlPlane.name	**controlPlane**을 사용하는 경우 필수 항목입니다. 시스템 풀의 이름입니다.	**master**
controlPlane.platform	**controlPlane**을 사용하는 경우 필수 항목입니다. 이 매개변수를 사용하여 컨트롤 플레인 시스템을 호스팅하는 클라우드 공급자를 지정합니다. 이 매개변수 값은 **compute.platform** 매개변수 값과 일치해야 합니다.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere** 또는 **{}**
controlPlane.replicas	프로비저닝하는 컨트롤 플레인 시스템의 수입니다.	지원되는 유일한 값은 기본값인 **3**입니다.

매개변수	설명	값
credentialsMode	Cloud Credential Operator (CCO) 모드입니다. 모드가 지정되지 않은 경우 CCO는 여러 모드가 지원되는 플랫폼에서 Mint 모드가 우선으로 되어 지정된 인증 정보의 기능을 동적으로 확인하려고 합니다.  **참고** 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다. CCO 모드에 대한 자세한 내용은 *Cluster Operators* 의 *Cloud Credential Operator* 를 참조하십시오.  **참고** AWS 계정에 SCP(서비스 제어 정책)가 활성화된 경우 **credentialsMode** 매개변수를 **Mint**,**Passthrough** 또는 **Manual** 로 구성해야 합니다.	**Mint**,**Passthrough**,**Manual** 또는 빈 문자열(**""**)

매개변수	설명	값
fips	FIPS 모드를 활성화 또는 비활성화합니다. 기본값은 **false**(비활성화)입니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.  **중요** FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.  **참고** Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.	**false** 또는 **true**
imageContentSources	릴리스 이미지 내용의 소스 및 리포지토리입니다.	개체의 배열입니다. 이 표의 다음 행에 설명된 대로 **source**와 **mirrors**(선택사항)가 포함됩니다.
imageContentSources.source	**imageContentSources**를 사용하는 경우 필수 항목입니다. 예를 들어 이미지 가져오기 사양에서 사용자가 가리키는 리포지토리를 지정합니다.	문자열
imageContentSources.mirrors	동일한 이미지를 포함할 수도 있는 하나 이상의 리포지토리를 지정합니다.	문자열 배열
publish	Kubernetes API, OpenShift 경로와 같이 클러스터의 사용자 끝점을 게시하거나 노출하는 방법입니다.	**Internal** 또는 **External**입니다. 기본값은 **External**입니다. 이 필드를 **Internal** 로 설정하는 것은 클라우드 이외의 플랫폼에서는 지원되지 않습니다.

매개변수	설명	값
sshKey	클러스터 시스템 액세스 인증에 필요한 하나 이상의 SSH 키입니다. **참고** 설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 SSH 키를 지정합니다.	하나 이상의 키입니다. 예를 들면 다음과 같습니다. sshKey: <key1> <key2> <key3>

18.5.11.1.4. 추가 VMware vSphere 구성 매개변수

추가 VMware vSphere 구성 매개변수는 다음 표에 설명되어 있습니다.

표 18.46. 추가 VMware vSphere 클러스터 매개변수

매개변수	설명	값
platform.vsphere.vCenter	vCenter 서버의 정규화된 호스트 이름 또는 IP 주소입니다.	문자열
platform.vsphere.username	vCenter 인스턴스에 연결하는 데 사용할 사용자 이름입니다. 이 사용자에게는 최소한 vSphere에서 정적 또는 동적 영구 볼륨 프로비저닝 에 필요한 역할과 권한이 있어야 합니다.	문자열
platform.vsphere.password	vCenter 사용자 이름의 암호입니다.	문자열
platform.vsphere.datacenter	vCenter 인스턴스에서 사용할 데이터 센터의 이름입니다.	문자열
platform.vsphere.defaultDatastore	프로비저닝 볼륨에 사용할 기본 데이터 저장소의 이름입니다.	문자열
platform.vsphere.folder	*선택사항*입니다. 설치 프로그램이 가상 머신을 생성하는 기존 폴더의 절대 경로입니다. 이 값을 제공하지 않으면 설치 프로그램이 데이터 센터 가상 머신 폴더에 인프라 ID로 이름이 지정된 폴더를 생성합니다.	문자열(예: **/<datacenter_name>/vm/<folder_name>/<subfolder_name>**).

매개변수	설명	값
platform.vsphere.network	vCenter 인스턴스에서 구성한 가상 IP 주소 및 DNS 레코드가 포함된 네트워크입니다.	문자열
platform.vsphere.cluster	OpenShift Container Platform 클러스터를 설치할 vCenter 클러스터입니다.	문자열
platform.vsphere.apiVIP	컨트롤 플레인 API 액세스를 위해 구성한 가상 IP(VIP) 주소입니다.	IP 주소 (예: **128.0.0.1**)
platform.vsphere.ingressVIP	클러스터 인그레스용으로 구성한 가상 IP(VIP) 주소입니다.	IP 주소 (예: **128.0.0.1**)

18.5.11.1.5. 선택적 VMware vSphere 시스템 풀 구성 매개변수

다음 표에는 선택적 VMware vSphere 시스템 풀 구성 매개 변수가 설명되어 있습니다.

표 18.47. 선택적 VMware vSphere 시스템 풀 매개변수

매개변수	설명	값
platform.vsphere.clusterOSImage	설치 프로그램이 RHCOS 이미지를 다운로드하는 위치입니다. 네트워크가 제한된 환경에서 설치를 수행하려면이 매개변수를 설정해야 합니다.	HTTP 또는 HTTPS URL (선택 옵션으로 SHA-256 형식의 체크섬을 사용) 예: **https://mirror.openshift.com/images/rhcos-<version>-vmware.<architecture>.ova**.
platform.vsphere.osDisk.diskSizeGB	디스크 크기(GB)입니다.	정수
platform.vsphere.cpus	가상 머신을 할당할 총 가상 프로세서 코어 수입니다. **platform.vsphere.cpus** 의 값은 **platform.vsphere.coresPerSocket** 값의 여러 개여야 합니다.	정수
platform.vsphere.coresPerSocket	가상 머신의 소켓당 코어 수입니다. 가상 머신의 가상 소켓 수는 **platform.vsphere.cpus/platform.vsphere.coresPerSocket** 입니다. 컨트롤 플레인 노드 및 작업자 노드의 기본값은 각각 **4** 및 **2** 입니다.	정수
platform.vsphere.memoryMB	가상 머신의 메모리 크기(MB)입니다.	정수

18.5.11.2. 설치 관리자 프로비저닝 VMware vSphere 클러스터의 샘플 install-config.yaml 파일

install-config.yaml 파일을 사용자 지정하여 **OpenShift Container Platform** 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.

```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 3
  platform:
    vsphere: (4)
      cpus: 2
      coresPerSocket: 2
      memoryMB: 8192
      osDisk:
        diskSizeGB: 120
controlPlane: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3
  platform:
    vsphere: (7)
      cpus: 4
      coresPerSocket: 2
      memoryMB: 16384
      osDisk:
        diskSizeGB: 120
metadata:
  name: cluster (8)
platform:
  vsphere:
    vcenter: your.vcenter.server
    username: username
    password: password
    datacenter: datacenter
    defaultDatastore: datastore
    folder: folder
    network: VM_Network
    cluster: vsphere_cluster_name (9)
    apiVIP: api_vip
    ingressVIP: ingress_vip
    clusterOSImage: http://mirror.example.com/images/rhcos-48.83.202103221318-0-
vmware.x86_64.ova (10)
fips: false
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email":
"you@example.com"}}}' (11)
sshKey: 'ssh-ed25519 AAAA...'
additionalTrustBundle: | (12)
  -----BEGIN CERTIFICATE-----
  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
```

```
  -----END CERTIFICATE-----
imageContentSources: 13
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

1

클러스터의 기본 도메인입니다. 모든 **DNS** 레코드는 이 기본 도메인의 하위 도메인이어야 하며 클러스터 이름을 포함해야 합니다.

2 5

controlPlane 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

3 6

동시 멀티스레딩 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. 일부 클러스터 시스템에서 동시 멀티스레딩을 비활성화할 경우에는 해당 멀티스레딩을 모든 클러스터 시스템에서 비활성화해야 합니다.

중요

동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다. 동시 멀티스레딩을 비활성화하는 경우 시스템은 **8**개 이상의 **CPU**와 **32GB**의 **RAM**을 사용해야 합니다.

4 7

선택사항: 컴퓨팅 및 컨트롤 플레인 시스템의 시스템 풀 매개변수에 대한 추가 구성을 제공하십시오.

8

DNS 레코드에 지정한 클러스터 이름입니다.

9

OpenShift Container Platform 클러스터를 설치할 **vSphere** 클러스터입니다. 설치 프로그램은 **vSphere** 클러스터의 루트 리소스 풀을 기본 리소스 풀로 사용합니다.

10

bastion 서버에서 액세스할 수 있는 RHCOS(Red Hat Enterprise Linux CoreOS) 이미지의 위치입니다.

11

<local_registry>는 미러 레지스트리가 해당 내용을 제공하는 데 사용하는 레지스트리 도메인 이름과 포트(선택사항)를 지정합니다. 예: registry.example.com 또는 registry.example.com:5000. <credentials>는 미러 레지스트리의 base64 인코딩 사용자 이름과 암호를 지정합니다.

12

미러 레지스트리에 사용한 인증서 파일의 내용을 제공하십시오.

13

명령 출력에서 imageContentSources 섹션을 제공하여 리포지토리를 미러링하십시오.

18.5.11.3. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 HTTP 또는 HTTPS 프록시를 사용할 수 있습니다. install-config.yaml 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 OpenShift Container Platform 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 install-config.yaml 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 API에 대한 호출을 포함하여 모든 클러스터 발신(Egress) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 Proxy 오브젝트의 spec.noProxy 필드에 사이트를 추가했습니다.

참고

Proxy 오브젝트의 status.noProxy 필드는 설치 구성에 있는 networking.machineNetwork[].cidr, networking.clusterNetwork[].cidr, networking.serviceNetwork[] 필드의 값으로 채워집니다.

Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 Red Hat OpenStack Platform (RHOSP)에 설치하는 경우 Proxy 오브젝트 status.noProxy 필드도 인스턴스 메타데이터 끝점(169.254.169.254)로 채워집니다.

절차

1. install-config.yaml 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

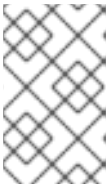
```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1 클러스터 외부에서 HTTP 연결을 구축하는 데 사용할 프록시 URL입니다. URL 스키마는 http여야 합니다.

2 클러스터 외부에서 HTTPS 연결을 구축하는 데 사용할 프록시 URL입니다.

3 대상 도메인 이름, IP 주소 또는 프록시에서 제외할 기타 네트워크 CIDR로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 .을 입력합니다. 예를 들어, .y.com은 x.y.com과 일치하지만 y.com은 일치하지 않습니다. *를 사용하여 모든 대상에 대해 프록시를 바이패스합니다. vCenter의 IP 주소와 해당 시스템에 사용하는 IP 범위를 포함해야 합니다.

4

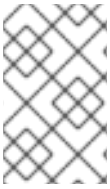


참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 **OpenShift Container Platform**을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 **nil**이 됩니다.

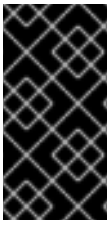


참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

18.5.12. 클러스터 배포

호환되는 클라우드 플랫폼에 **OpenShift Container Platform**을 설치할 수 있습니다.



중요

최초 설치 과정에서 설치 프로그램의 **create cluster** 명령을 한 번만 실행할 수 있습니다.

사전 요구 사항

- 클러스터를 호스팅하는 클라우드 플랫폼으로 계정을 구성합니다.
- **OpenShift Container Platform** 설치 프로그램과 클러스터의 풀 시크릿을 받습니다.

절차

1. 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터 배포를 초기화합니다.

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
  --log-level=info 2
```

1

<installation_directory> 값으로 사용자 지정한 ./install-config.yaml 파일의 위치를 지정합니다.

2

다른 설치 세부 사항을 보려면 info 대신 warn, debug 또는 error를 지정합니다.

중요

VMC 환경에서 호스팅되는 bastion의 openshift-install 명령을 사용합니다.

참고

호스트에 구성된 클라우드 공급자 계정에 클러스터를 배포하기에 충분한 권한이 없는 경우, 설치 프로세스가 중단되고 누락된 권한을 알리는 메시지가 표시됩니다.

클러스터 배포가 완료되면 웹 콘솔로 연결되는 링크와 kubeadmin 사용자의 인증 정보가 포함된 클러스터 액세스 지침이 사용자 터미널에 표시됩니다.

출력 예

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-
ymBZj-Wt5AL"
INFO Time elapsed: 36m22s
```

참고

설치에 성공하면 클러스터 액세스 및 인증 정보도 **<installation_directory>/.openshift_install.log**로 출력됩니다.

중요

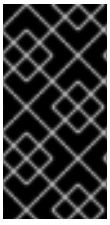
- 설치 프로그램에서 생성하는 **Ignition** 구성 파일에 **24**시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 **24**시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 **kubelet** 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청**(CSR)**을 수동으로 승인해야 합니다. 자세한 내용은 ***Recovering from expired control plane certificates*** 문서를 참조하십시오.
- 클러스터를 설치한 후 **24**시간에서 **22**시간까지의 인증서가 교체되기 때문에 생성된 후 **12**시간 이내에 **Ignition** 구성 파일을 사용하는 것이 좋습니다. **12**시간 이내에 **Ignition** 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

중요

설치 프로그램에서 생성되는 파일이나 설치 프로그램을 삭제해서는 안 됩니다. 클러스터를 삭제하려면 두 가지가 모두 필요합니다.

18.5.13. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 **OpenShift Container Platform**과 상호 작용하기 위해 **OpenShift CLI(oc)**를 설치할 수 있습니다. **Linux, Windows** 또는 **macOS**에 **oc**를 설치할 수 있습니다.



중요

이전 버전의 **oc**를 설치한 경우 **OpenShift Container Platform 4.9**의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 **oc**를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 **Linux**에서 **OpenShift CLI(oc)** 바이너리를 설치할 수 있습니다.

절차

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.

2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.

3. OpenShift v4.9 Linux Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.

4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. oc 바이너리를 PATH에 있는 디렉터리에 배치합니다.

   PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치

다음 절차에 따라 Windows에 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

절차

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.

2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.

3. OpenShift v4.9 Windows Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.

4. ZIP 프로그램으로 아카이브의 압축을 풉니다.

5. oc 바이너리를 PATH에 있는 디렉터리로 이동합니다.

   PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치

다음 절차에 따라 macOS에서 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

절차

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.

2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.

3. OpenShift v4.9 MacOSX Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.

4. 아카이브의 압축을 해제하고 압축을 풉니다.

5. oc 바이너리 PATH의 디렉터리로 이동합니다.

   PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
$ oc <command>
```

18.5.14. CLI를 사용하여 클러스터에 로그인

클러스터 kubeconfig 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. kubeconfig 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- oc CLI를 설치했습니다.

절차

1. kubeadmin 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 oc 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

18.5.15. 기본 OperatorHub 소스 비활성화

Red Hat 및 커뮤니티 프로젝트에서 제공하는 콘텐츠를 소싱하는 Operator 카탈로그는 OpenShift Container Platform을 설치하는 동안 기본적으로 OperatorHub용으로 구성됩니다. 제한된 네트워크 환경에서는 클러스터 관리자로서 기본 카탈로그를 비활성화해야 합니다.

절차

- OperatorHub 오브젝트에 disableAllDefaultSources: true를 추가하여 기본 카탈로그의 소스를 비활성화합니다.

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

작은 정보

또는 웹 콘솔을 사용하여 카탈로그 소스를 관리할 수 있습니다. 관리 → 클러스터 설정 → 구성 → OperatorHub 페이지에서 개별 소스를 생성, 삭제, 비활성화 및 활성화할 수 있는 소스 탭을 클릭합니다.

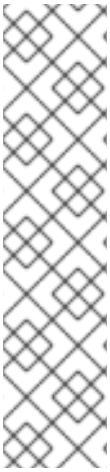
18.5.16. 레지스트리 스토리지 생성

클러스터를 설치한 후 Registry Operator를 위한 스토리지를 생성해야 합니다.

18.5.16.1. 설치 중 제거된 이미지 레지스트리

공유 가능한 개체 스토리지를 제공하지 않는 플랫폼에서 OpenShift Image Registry Operator는 자체적으로 Removed로 부트스트랩합니다. 이를 통해 openshift-installer가 이러한 플랫폼 유형에서 설치를 완료할 수 있습니다.

설치 후 managementState를 Removed에서 Managed로 전환하도록 Image Registry Operator 구성을 편집해야 합니다.



참고

Prometheus 콘솔은 **ImageRegistryRemoved** 경고를 제공합니다. 다음은 예시 경고입니다.

"이미지 레지스트리가 제거되었습니다. **ImageStreamTags**를 참조하는 **ImageStreamTags, BuildConfigs** 및 **DeploymentConfigs**가 예상대로 작동하지 않을 수 있습니다. 스토리지를 구성하고 **configs.imageregistry.operator.openshift.io**를 편집하여 구성을 **Managed** 상태로 업데이트하십시오."

18.5.16.2. 이미지 레지스트리 스토리지 구성

기본 스토리지를 제공하지 않는 플랫폼에서는 처음에 **Image Registry Operator**를 사용할 수 없습니다. 설치한 후에 스토리지를 사용하도록 레지스트리를 구성하여 **Registry Operator**를 사용 가능하도록 만들어야 합니다.

프로덕션 클러스터에 필요한 영구 볼륨을 구성하는 과정의 지침이 표시됩니다. 해당하는 경우, 프로덕션 환경 외 클러스터에서만 사용할 수 있는 저장 위치로서 빈 디렉터리를 구성하는 과정의 지침이 표시됩니다.

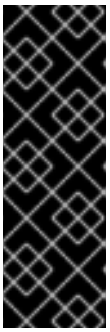
업그레이드 중에 **Recreate** 롤아웃 전략을 사용하여 이미지 레지스트리의 블록 스토리지 유형 사용을 허용하기 위한 추가 지침이 제공됩니다.

18.5.16.2.1. VMware vSphere용 레지스트리 스토리지 구성

클러스터 관리자는 설치한 후 스토리지를 사용하도록 레지스트리를 구성해야 합니다.

사전 요구 사항

- 클러스터 관리자 권한이 있어야 합니다.
- **VMware vSphere**에 클러스터가 있어야 합니다.
- **Red Hat OpenShift Container Storage**와 같이 클러스터용 영구 스토리지 프로비저닝.



중요

OpenShift Container Platform은 복제본이 하나만 있는 경우 이미지 레지스트리 스토리지에 대한 **ReadWriteOnce** 액세스를 지원합니다. **ReadWriteOnce** 액세스에서는 레지스트리가 **Recreate** 롤아웃 전략을 사용해야 합니다. 두 개 이상의 복제본으로 고 가용성을 지원하는 이미지 레지스트리를 배포하려면 **ReadWriteMany** 액세스가 필요합니다.

- **"100Gi"** 용량이 필요합니다.

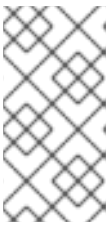
중요

테스트 결과, **RHEL**의 **NFS** 서버를 핵심 서비스용 스토리지 백엔드로 사용하는 데 문제가 있는 것으로 나타납니다. 여기에는 **OpenShift Container Registry and Quay**, 스토리지 모니터링을 위한 **Prometheus**, 로깅 스토리지를 위한 **Elasticsearch**가 포함됩니다. 따라서 **RHEL NFS**를 사용하여 핵심 서비스에서 사용하는 **PV**를 백업하는 것은 권장되지 않습니다.

마켓플레이스의 다른 **NFS** 구현에는 이러한 문제가 나타나지 않을 수 있습니다. 이러한 **OpenShift Container Platform** 핵심 구성 요소에 대해 완료된 테스트에 대한 자세한 내용은 개별 **NFS** 구현 공급업체에 문의하십시오.

절차

1. 스토리지를 사용하도록 레지스트리를 구성하기 위해 **configs.imageregistry/cluster** 리소스에서 **spec.storage.pvc**를 변경합니다.



참고

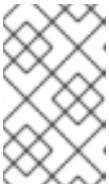
공유 스토리지를 사용할 때 보안 설정을 확인하여 외부에서의 액세스를 방지합니다.

2. 레지스트리 **pod**가 없는지 확인합니다.

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

출력 예

```
No resourses found in openshift-image-registry namespace
```



참고

출력에 레지스트리 Pod가 있는 경우 이 절차를 계속할 필요가 없습니다.

3. 레지스트리 구성을 확인합니다.

```
$ oc edit configs.imageregistry.operator.openshift.io
```

출력 예

```
storage:
  pvc:
    claim: 1
```

1 image-registry-storage 영구 볼륨 클레임 (PVC)을 자동으로 생성할 수 있도록 claim 필드를 비워 둡니다. PVC는 기본 스토리지 클래스를 기반으로 생성됩니다. 그러나 기본 스토리지 클래스에서 RADOS 블록 장치(RBD)와 같은 ReadWriteOnce(ReadWriteOnce) 볼륨을 제공할 수 있으므로 복제본이 두 개 이상될 때 문제가 발생할 수 있습니다.

4. clusteroperator 상태를 확인합니다.

```
$ oc get clusteroperator image-registry
```

출력 예

```
NAME              VERSION                              AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry    4.7                                  True        False         False     6h50m
```

18.5.17. 스틸 클럭 계정

기본적으로 설치 프로그램은 스틸 클럭 회계 매개 변수 (**stealclock.enabled**)를 활성화하지 않고 클러스터의 가상 머신을 프로비저닝합니다. 스틸링 클럭 계정을 사용하면 클러스터 문제 해결에 도움이 될 수 있습니다. 클러스터가 배포된 후 **vSphere Client**를 사용하여 각 가상 머신에서 이 매개변수를 활성화합니다.

자세한 내용은 이 **Red Hat 기술 자료 문서** 를 참조하십시오.

18.5.18. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 **Telemetry** 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 **Telemetry**가 자동으로 실행되고 **OpenShift Cluster Manager** 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 **OpenShift Cluster Manager**를 사용하여 자동으로 또는 **OpenShift Cluster Manager**를 사용하여 수동으로 유지 관리되는지 확인한 후 **subscription watch**를 사용하여 계정 또는 다중 클러스터 수준에서 **OpenShift Container Platform** 서브스크립션을 추적합니다.

추가 리소스

- **Telemetry** 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

18.5.19. 다음 단계

- 클러스터를 사용자 지정합니다.
- **Cluster Samples Operator** 및 **must-gather** 툴의 이미지 스트림을 구성합니다.

- 제한된 네트워크에서 Operator Lifecycle Manager (OLM) 사용 방법에 대해 살펴봅니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.
- 레지스트리를 설정하고 레지스트리 스토리지를 구성합니다.

18.6. 사용자 프로비저닝 인프라를 사용하여 VMC에 클러스터 설치

OpenShift Container Platform 4.9 버전에서는 프로비저닝하는 클러스터를 AWS의 VMC(VMware Cloud)에 배포하여 VMware vSphere 인프라에 설치할 수 있습니다.

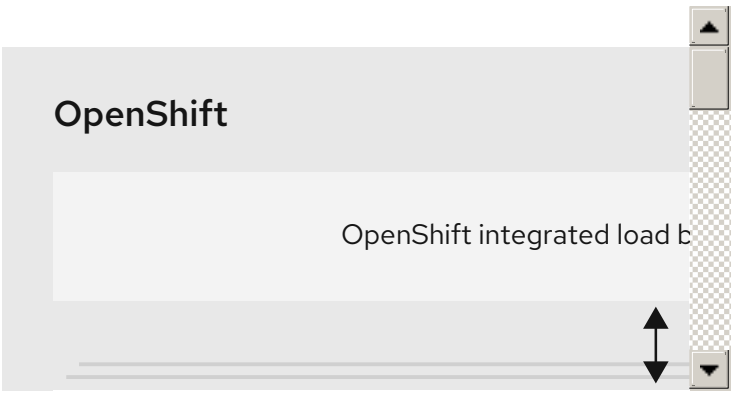
OpenShift Container Platform 배포를 위해 VMC 환경을 구성한 후 VMC 환경에 공동 배치된 bastion 관리 호스트에서 OpenShift Container Platform 설치 프로그램을 사용합니다. 설치 프로그램 및 컨트롤 플레인은 OpenShift Container Platform 클러스터에 필요한 리소스 배포 및 관리 프로세스를 자동화합니다.

참고

OpenShift Container Platform은 단일 VMware vCenter에만 클러스터 배포를 지원합니다. 여러 vCenter에 머신/머신 세트가 있는 클러스터 배포는 지원되지 않습니다.

18.6.1. vSphere용 VMC 설정

AWS 호스팅된 vSphere 클러스터에 VMware Cloud (VMC)의 OpenShift Container Platform을 설치하여 하이브리드 클라우드 전반에 애플리케이션을 온프레미스 및 오프프레미스 모두 배포 및 관리할 수 있습니다.



VMware vSphere에 OpenShift Container Platform을 설치하기 전에 VMC 환경에서 여러 옵션을 구

성해야 합니다. VMC 환경에 다음과 같은 사전 요구 사항이 있는지 확인합니다.

- 비독점 DHCP 사용, NSX-T 네트워크 세그먼트 및 서브넷을 생성합니다. 서브넷에서 다른 VM(가상 머신)을 호스팅할 수 있지만 OpenShift Container Platform 배포에서는 8개 이상의 IP 주소를 사용할 수 있어야 합니다.

- 다음 방화벽 규칙을 설정합니다.

  - OpenShift Container Platform 컴퓨팅 네트워크와 인터넷 간의 ANY:ANY 방화벽 규칙입니다. 이는 노드와 애플리케이션에서 컨테이너 이미지를 다운로드하는 데 사용됩니다.

  - 포트 443의 설치 호스트와 소프트웨어 정의 데이터 센터(SDDC) 관리 네트워크 간의 ANY:ANY 방화벽 규칙입니다. 이를 통해 배포 중에 RHCOS(Red Hat Enterprise Linux CoreOS) OVA를 업로드할 수 있습니다.

  - OpenShift Container Platform 컴퓨팅 네트워크와 인터넷 간의 HTTPS 방화벽 규칙입니다. 이 연결을 통해 OpenShift Container Platform은 노드, PVC(영구 볼륨 클레임) 및 기타 리소스를 프로비저닝 및 관리하기 위해 vCenter과 통신할 수 있습니다.

- OpenShift Container Platform을 배포하려면 다음 정보가 있어야 합니다.

  - OpenShift Container Platform 클러스터에 vmc-prod-1로 로그인합니다.

  - companyname.com과 같은 기본 DNS 이름입니다.

  - 기본값을 사용하지 않는 경우 Pod 네트워크 CIDR 및 서비스 네트워크 CIDR을 식별해야 하며, 기본적으로 10.128.0.0/14 및 172.30.0.0/16으로 설정됩니다. 이러한 CIDR은 pod-to-pod 및 pod-to-service 통신에 사용되며 외부에서 액세스할 수 없지만 조직의 기존 서브넷과 중복되어서는 안 됩니다.

  - 다음 vCenter 정보를 참조하십시오:

    - vCenter 호스트 이름, 사용자 이름 및 암호

- 데이터 센터 이름 (예: SDDC-Datacenter)
- Cluster-1과 같은 클러스터 이름
- 네트워크 이름
- WorkloadDatastore와 같은 데이터 저장소 이름

참고

클러스터 설치가 완료된 후 vSphere 클러스터를 VMC Compute-ResourcePool 리소스 풀로 이동하는 것이 좋습니다.

- VMC에 bastion으로 배포된 Linux 기반 호스트입니다.
  - bastion 호스트는 RHEL(Red Hat Enterprise Linux) 또는 기타 Linux 기반 호스트일 수 있습니다. 인터넷에 연결되어 있어야 하고 OVA를 ESXi 호스트에 업로드하는 기능이 있어야 합니다.
  - OpenShift CLI 툴을 bastion 호스트에 다운로드하여 설치합니다.
    - openshift-install 설치 프로그램
    - OpenShift CLI(oc) 툴

참고

Kubernetes 용 VMware NCP (NSX Conrainer Plugin)dOpen에는 사용할 수 없으며 NSX는 OpenShift SDN으로 사용되지 않습니다. 현재 VMC에서 사용할 수 있는 NSX 버전은 OpenShift Container Platform에서 인증된 NCP 버전과 호환되지 않습니다.

그러나 NSX DHCP 서비스는 전체 스택 자동화된 OpenShift Container Platform 배포와 vSphere와의 Machine API 통합을 통해 수동 또는 자동으로 프로비저닝된 노드를 사용하여 가상 머신 IP 관리에 사용됩니다. 또한 OpenShift Container Platform 클러스터와 bastion 호스트와 VMC vSphere 호스트 사이에서 액세스할 수 있도록 NSX 방화벽 규칙이 생성됩니다.

18.6.1.1. VMC Sizer 툴

AWS의 VMware Cloud는 AWS 베어메탈 인프라 상단에 구축됩니다. 이는 AWS 네이티브 서비스를 실행하는 것과 동일한 베어 메탈 인프라입니다. AWS 소프트웨어 정의 데이터 센터(SDDC)의 VMware 클라우드가 배포되면 이러한 물리적 서버 노드를 사용하고 단일 테넌트 방식으로 VMware ESXi 하이퍼바이저를 실행할 수 있습니다. 즉, VMC를 사용하는 모든 사용자가 물리적 인프라에 액세스할 수 없습니다. 가상 인프라를 호스팅하는데 필요한 물리적 호스트 수를 고려해야 합니다.

이를 확인하기 위해 VMware는 AWS Sizer에서 VMC를 제공합니다. 이 툴을 사용하면 VMC에서 호스팅할 리소스를 정의할 수 있습니다.

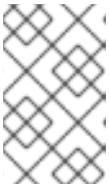
- 워크로드 유형
- 총 가상 머신 수
- 사양 정보는 다음과 같습니다:
  - 스토리지 요구사항
  - vCPU
  - vRAM

- 오버커밋 비율

이러한 세부 정보를 사용하여 VMware 모범 사례를 기반으로 보고서를 생성하고 클러스터 구성과 필요한 호스트 수를 권장합니다.

18.6.2. vSphere 사전 요구 사항

- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.
- 블록 레지스트리 스토리지가 프로비저닝되어 있습니다. 자세한 내용은 영구 저장 장치 이해를 참조하십시오.
- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성해야 합니다.



참고

프록시를 구성하는 경우에도 해당 사이트 목록을 검토하십시오.

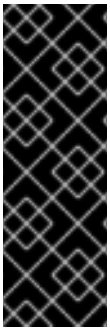
18.6.3. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 Telemetry 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.

- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.



중요

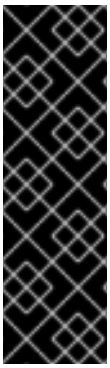
클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

18.6.4. VMware vSphere 인프라 요구사항

사용하는 구성 요소의 요구사항을 충족하는 VMware vSphere 버전 6 또는 7 인스턴스에 OpenShift Container Platform 클러스터를 설치해야 합니다.

표 18.48. vSphere 가상 환경의 버전 요구 사항

가상 환경 제품	필요한 버전
VM 하드웨어 버전	13 이상
vSphere ESXi hosts	6.5 이상
vCenter 호스트	6.5 이상



중요

이제 VMware vSphere 버전 6.7U2 또는 이전 버전에 클러스터를 설치하는 것이 더 이상 사용되지 않으며 가상 하드웨어 버전 13은 더 이상 사용되지 않습니다. 이러한 버전은 여전히 완전히 지원되지만 향후 OpenShift Container Platform 버전에서는 지원이 제거됩니다. 이제 OpenShift Container Platform의 vSphere 가상 머신의 하드웨어 버전 15가 기본값이 되었습니다. vSphere 노드의 하드웨어 버전을 업데이트하려면 " vSphere에서 실행되는 노드에서 하드웨어 업데이트" 문서를 참조하십시오.

표 18.49. VMware 구성 요소에 지원되는 최소 vSphere 버전

구성 요소	지원되는 최소 버전	설명
하이퍼바이저	HW 버전 13이 포함된 vSphere 6.5 이상	이 버전은 RHCOS(Red Hat Enterprise Linux CoreOS)가 지원하는 최소 버전입니다. Red Hat Enterprise Linux 8 지원 하이퍼바이저 목록을 참조하십시오.

구성 요소	지원되는 최소 버전	설명
In-tree 드라이버가 있는 스토리지	vSphere 6.5 이상	이 플러그인은 OpenShift Container Platform에 포함된 vSphere용 인트리 스토리지 드라이버를 사용하여 vSphere 스토리지를 생성합니다.

vSphere 버전 6.5 인스턴스를 사용하는 경우 OpenShift Container Platform을 설치하기 전에 6.7U3 또는 7.0로 업그레이드하는 것이 좋습니다.

중요

OpenShift Container Platform을 설치하기 전에 ESXi 호스트의 시간이 동기화되었는지 확인해야 합니다. VMware 문서에서 Edit Time Configuration for a Host를 참조하십시오.

추가 리소스

- **vSphere 노드의 하드웨어 버전을 업데이트하려면 vSphere에서 실행 중인 노드에서 하드웨어 업데이트를 참조하십시오.**

18.6.5. 사용자 프로비저닝 인프라를 포함한 클러스터의 시스템 요구사항

사용자 프로비저닝 인프라가 포함된 클러스터의 경우, 필요한 모든 시스템을 배포해야 합니다.

이 섹션에서는 사용자 프로비저닝 인프라에 OpenShift Container Platform을 배포해야 하는 요구 사항에 대해 설명합니다.

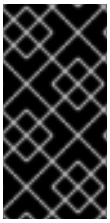
18.6.5.1. 클러스터 설치에 필요한 시스템

최소 OpenShift Container Platform 클러스터에 다음과 같은 호스트가 필요합니다.

표 18.50. 최소 필수 호스트

호스트	설명

호스트	설명
임시 부트스트랩 시스템 한 개	컨트롤 플레인 시스템 세 개에 OpenShift Container Platform 클러스터를 배포하기 위한 부트스트랩 시스템이 클러스터에 필요합니다. 클러스터를 설치한 후 부트스트랩 시스템을 제거할 수 있습니다.
컨트롤 플레인 시스템 세 개	컨트롤 플레인 시스템은 컨트롤 플레인을 구성하는 Kubernetes 및 OpenShift Container Platform 서비스를 실행합니다.
두 개 이상의 컴퓨팅 시스템(작업자 시스템이라고도 함).	OpenShift Container Platform 사용자가 요청한 워크로드는 컴퓨팅 머신에서 실행됩니다.



중요

클러스터의 고가용성을 유지하려면 이러한 클러스터 시스템에 대해 별도의 물리적 호스트를 사용하십시오.

부트스트랩, 컨트롤 플레인 시스템은 운영 체제로 RHCOS (Red Hat Enterprise Linux CoreOS)를 사용해야 합니다. 그러나 컴퓨팅 머신은 RHCOS(Red Hat Enterprise Linux CoreOS), RHEL (Red Hat Enterprise Linux) 7.9 또는 RHEL 8.4 중에서 선택할 수 있습니다.

RHCOS는 RHEL 8(Red Hat Enterprise Linux)을 기반으로하며 모든 하드웨어 인증 및 요구사항을 모두 이어받습니다. Red Hat Enterprise Linux 기술 기능 및 제한을 참조하십시오.

18.6.5.2. 클러스터 설치를 위한 최소 리소스 요구 사항

각 클러스터 시스템이 다음과 같은 최소 요구사항을 충족해야 합니다.

표 18.51. 최소 리소스 요구사항

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
부트스트랩	RHCOS	4	16GB	100GB	300
컨트롤 플레인	RHCOS	4	16GB	100GB	300
컴퓨팅	RHCOS, RHEL 7.9 또는 RHEL 8.4 [3]	2	8GB	100GB	300

1. SMT(동시 멀티스레딩) 또는 하이퍼 스레딩이 활성화되지 않은 경우 하나의 vCPU는 하나의 물리적 코어와 동일합니다. 활성화하면 다음과 같은 공식을 사용하여 해당 비율을 계산합니다. (코어 당 스레드 수 × 코어 수) × 소켓 수 = vCPU 수

2. OpenShift Container Platform 및 Kubernetes는 디스크 성능에 민감하며 특히 10ms p99 fsync 기간이 필요한 컨트롤 플레인 노드의 etcd에 더 빠른 스토리지가 권장됩니다. 많은 클라우드 플랫폼에서 스토리지 크기와 IOPS를 함께 확장되므로 충분한 성능을 얻으려면 스토리지 볼륨을 과도하게 할당해야 할 수 있습니다.

3. 사용자가 프로비저닝한 모든 설치와 마찬가지로 클러스터에서 RHEL 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. RHEL 7 컴퓨팅 머신 사용은 더 이상 사용되지 않으며 향후 OpenShift Container Platform 4 릴리스에서 제거될 예정입니다.

18.6.5.3. 인증서 서명 요청 관리

사용자가 프로비저닝하는 인프라를 사용하는 경우 자동 시스템 관리 기능으로 인해 클러스터의 액세스가 제한되므로 설치한 후 클러스터 인증서 서명 요청(CSR)을 승인하는 메커니즘을 제공해야 합니다. kube-controller-manager는 kubelet 클라이언트 CSR만 승인합니다. machine-approver는 올바른 시스템에서 발행한 요청인지 확인할 수 없기 때문에 kubelet 자격 증명을 사용하여 요청하는 서비스 인증서의 유효성을 보장할 수 없습니다. kubelet 서빙 인증서 요청의 유효성을 확인하고 요청을 승인하는 방법을 결정하여 구현해야 합니다.

18.6.5.4. 사용자 프로비저닝 인프라에 대한 네트워킹 요구사항

모든 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템이 부팅 중에 Ignition 구성 파일을 가져오려면 initramfs에 네트워킹을 구성해야 합니다.

초기 부팅 과정에서 시스템에 필요한 부팅 옵션을 제공하여 DHCP 서버를 통해 설정하거나 정적으로 설정하는 IP 주소 구성이 필요합니다. 네트워크 연결이 설정된 후 시스템은 HTTP 또는 HTTPS 서버에서 Ignition 구성 파일을 다운로드합니다. 그런 다음 Ignition 구성 파일을 사용하여 각 머신의 정확한 상태를 설정합니다. Machine Config Operator는 설치 후 새 인증서 또는 키 적용과 같은 머신에 대한 추가 변경을 완료합니다.

클러스터 시스템의 장기적인 관리를 위해 DHCP 서버를 사용하는 것이 좋습니다. DHCP 서버가 클러스터 시스템에 영구 IP 주소, DNS 서버 정보 및 호스트 이름을 제공하도록 구성되었는지 확인합니다.

참고

사용자 프로비저닝 인프라에 **DHCP** 서비스를 사용할 수 없는 경우 **RHCOS** 설치 시 노드에 **IP** 네트워킹 구성과 **DNS** 서버의 주소를 대신 제공할 수 있습니다. **ISO** 이미지에서 설치하는 경우 부팅 인수로 전달할 수 있습니다. 고정 **IP** 프로비저닝 *및 고급 네트워킹 옵션에 대한 자세한 내용은 **RHCOS** 설치 및 **OpenShift Container Platform** 부트스트랩 프로세스 시작* 섹션을 참조하십시오.

Kubernetes API 서버가 클러스터 시스템의 노드 이름을 확인할 수 있어야 합니다. **API** 서버와 작업자 노드가 서로 다른 영역에 있는 경우, **API** 서버가 노드 이름을 확인할 수 있도록 기본 **DNS** 검색 영역을 설정할 수 있습니다. 노드 개체와 모든 **DNS** 요청에서 항상 정규화된 도메인 이름으로 호스트를 가리키는 것도 지원되는 방법입니다

18.6.5.4.1. DHCP를 통해 클러스터 노드의 호스트 이름 설정

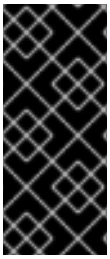
RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 호스트 이름은 **NetworkManager**를 통해 설정됩니다. 기본적으로 시스템은 **DHCP**를 통해 호스트 이름을 가져옵니다. **DHCP**에서 호스트 이름을 제공하지 않거나, 커널 인수를 통해 정적으로 설정하거나 다른 방법을 통해 설정하면 역방향 **DNS** 조회를 통해 얻을 수 있습니다. 역방향 **DNS** 조회는 노드에서 네트워크를 초기화한 후 수행되며 확인하는 데 시간이 걸릴 수 있습니다. 다른 시스템 서비스는 이 보다 먼저 시작하여 호스트 이름을 **localhost** 등으로 감지할 수 있습니다. **DHCP**를 사용하여 각 클러스터 노드의 호스트 이름을 제공하여 이 문제를 방지할 수 있습니다.

또한 **DHCP**를 통해 호스트 이름을 설정하면 **DNS** 분할 수평 구현 환경에서 수동으로 **DNS** 레코드 이름 구성 오류를 무시할 수 있습니다.

18.6.5.4.2. 네트워크 연결 요구사항

OpenShift Container Platform 클러스터 구성 요소가 통신할 수 있도록 시스템 간 네트워크 연결을 구성해야 합니다. 각 시스템에서 클러스터에 있는 다른 모든 시스템의 호스트 이름을 확인할 수 있어야 합니다.

이 섹션에서는 필요한 포트에 대해 자세히 설명합니다.



중요

연결된 **OpenShift Container Platform** 환경에서 모든 노드는 플랫폼 컨테이너의 이미지를 가져오고 **Red Hat**에 원격 측정 데이터를 제공하기 위해 인터넷에 액세스할 수 있어야 합니다.

표 **18.52.** 모든 시스템 간 통신에 사용되는 포트

프로토콜	포트	설명
ICMP	해당 없음	네트워크 연결성 테스트
TCP	**1936**	메트릭
	9000-9999	**9100-9101** 포트의 노드 내보내기 및 **9099** 포트의 Cluster Version Operator를 포함한 호스트 수준 서비스.
	10250-10259	Kubernetes에서 예약하는 기본 포트
	10256	openshift-sdn
UDP	**4789**	VXLAN 및 Geneve
	6081	VXLAN 및 Geneve
	9000-9999	**9100-9101** 포트의 노드 내보내기를 포함한 호스트 수준 서비스.
	500	IPsec IKE 패킷
	4500	IPsec NAT-T 패킷
TCP/UDP	**30000-32767**	Kubernetes 노드 포트
ESP	해당 없음	IPsec Encapsulating Security Payload (ESP)

표 18.53. 모든 시스템과 컨트롤 플레인 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**6443**	Kubernetes API

표 18.54. 컨트롤 플레인 머신 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**2379-2380**	etcd 서버 및 피어 포트

사용자 프로비저닝 인프라에 대한 NTP 구성

OpenShift Container Platform 클러스터는 기본적으로 공용 **NTP(Network Time Protocol)** 서버를 사용하도록 구성되어 있습니다. 로컬 엔터프라이즈 **NTP** 서버를 사용하거나 클러스터가 연결이 끊긴 네트워크에 배포되는 경우 특정 시간 서버를 사용하도록 클러스터를 구성할 수 있습니다. 자세한 내용은 ***chrony* 타임 서비스 설정** 문서를 참조하십시오.

DHCP 서버가 NTP 서버 정보를 제공하는 경우 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템의 chrony 타임 서비스에서 정보를 읽고 NTP 서버와 클럭을 동기화할 수 있습니다.

18.6.5.5. 사용자 프로비저닝 DNS 요구사항

OpenShift Container Platform 배포의 경우 다음 구성 요소에 DNS 이름을 확인해야 합니다.

- Kubernetes API
- OpenShift Container Platform 애플리케이션 와일드카드
- 부트스트랩, 컨트롤 플레인 및 컴퓨팅 시스템

Kubernetes API, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템에 대한 역방향 DNS 확인이 필요합니다.

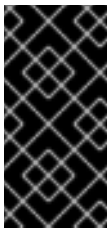
DNS A/AAAA 또는 CNAME 레코드는 이름 확인에 사용되며 PTR 레코드는 역방향 이름 확인에 사용됩니다. RHCOS (Red Hat Enterprise Linux CoreOS)는 DHCP에서 호스트 이름을 제공하지 않는 한 모든 노드의 호스트 이름을 설정할 때 역방향 레코드를 사용하기 때문에 역방향 레코드가 중요합니다. 또한 역방향 레코드는 OpenShift Container Platform이 작동하는 데 필요한 인증서 서명 요청 (CSR)을 생성하는 데 사용됩니다.

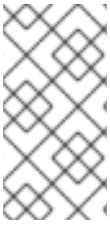
참고

DHCP 서버를 사용하여 각 클러스터 노드에 호스트 이름을 제공하는 것이 좋습니다. 자세한 내용은 *사용자 프로비저닝 인프라 섹션에 대한 DHCP 권장 사항* 섹션을 참조하십시오.

사용자가 프로비저닝한 OpenShift Container Platform 클러스터에 대해 다음 DNS 레코드가 필요하며 설치 전에 있어야 합니다. 각 레코드에서 <cluster_name>은 클러스터 이름이고 <base_domain>은 install-config.yaml 파일에서 지정하는 기반 도메인입니다. 전체 DNS 레코드는 <component>.<cluster_name>.<base_domain> 형식입니다.

표 18.55. 필수 DNS 레코드

구성 요소	레코드	설명
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API 로드 밸런서를 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.
	api-int.<cluster_name>.<base_domain>.	내부적으로 API 로드 밸런서를 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.  **중요** API 서버는 Kubernetes에 기록된 호스트 이름으로 작업자 노드를 확인할 수 있어야 합니다. API 서버가 노드 이름을 확인할 수 없는 경우 프록시된 API 호출이 실패할 수 있으며 pod에서 로그를 검색할 수 없습니다.
라우트	***.apps.<cluster_name>.<base_domain>.**	애플리케이션 인그레스 로드 밸런서를 참조하는 와일드카드 DNS A/AAA 또는 CNAME 레코드입니다. 애플리케이션 인그레스 로드 밸런서는 Ingress 컨트롤러 Pod를 실행하는 머신을 대상으로 합니다. Ingress 컨트롤러 Pod는 기본적으로 컴퓨팅 머신에서 실행됩니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다. 예를 들어 **console-openshift-console.apps.<cluster_name>.<base_domain>**은 OpenShift Container Platform 콘솔의 와일드카드 경로로 사용됩니다.
부트스트랩 시스템	**bootstrap.<cluster_name>.<base_domain>.**	부트스트랩 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.
컨트롤 플레인 머신	**<master><n>.<cluster_name>.<base_domain>.**	컨트롤 플레인 노드의 각 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.
컴퓨팅 머신	**<worker><n>.<cluster_name>.<base_domain>.**	작업자 노드의 각 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.



참고

OpenShift Container Platform 4.4 이상에서는 DNS 구성에서 etcd 호스트 및 SRV 레코드를 지정할 필요가 없습니다.

작은 정보

dig 명령을 사용하여 이름과 역방향 이름을 확인할 수 있습니다. 자세한 검증 단계는 *사용자 프로비저닝 인프라의 DNS 확인* 섹션을 참조하십시오.

18.6.5.5.1. 사용자 프로비저닝 클러스터의 DNS 구성 예

이 섹션에서는 사용자 프로비저닝 인프라에 OpenShift Container Platform을 배포하기 위한 DNS 요구 사항을 충족하는 A 및 PTR 레코드 구성 샘플을 제공합니다. 샘플은 하나의 DNS 솔루션을 선택하기 위한 조언을 제공하기 위한 것이 아닙니다.

이 예제에서 클러스터 이름은 ocp4이고 기본 도메인은 example.com입니다.

사용자 프로비저닝 클러스터의 DNS A 레코드 구성 예

다음 BIND 영역 파일의 예제에서는 사용자가 프로비저닝한 클러스터의 이름 확인을 위한 샘플 A 레코드를 보여줍니다.

예 18.13. 샘플 DNS 영역 데이터베이스

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
master0.ocp4.example.com. IN A 192.168.1.97 5
master1.ocp4.example.com. IN A 192.168.1.98 6
master2.ocp4.example.com. IN A 192.168.1.99 7
```

```
;
worker0.ocp4.example.com. IN A 192.168.1.11 8
worker1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1

Kubernetes API의 이름 확인을 제공합니다. 레코드는 **API** 로드 밸런서의 **IP** 주소를 나타냅니다.

2

Kubernetes API의 이름 확인을 제공합니다. 레코드는 **API** 로드 밸런서의 **IP** 주소를 참조하며 내부 클러스터 통신에 사용됩니다.

3

와일드카드 경로의 이름 확인을 제공합니다. 레코드는 애플리케이션 인그레스 로드 밸런서의 **IP** 주소를 나타냅니다. 애플리케이션 인그레스 로드 밸런서는 **Ingress** 컨트롤러 **Pod**를 실행하는 머신을 대상으로 합니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

참고

이 예제에서는 **Kubernetes API** 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 **API** 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

4

부트스트랩 시스템의 이름 확인을 제공합니다.

5 6 7

컨트롤 플레인 시스템의 이름 확인을 제공합니다.

8 9

컴퓨팅 시스템의 이름 확인을 제공합니다.

사용자 프로비저닝 클러스터의 **DNS PTR** 레코드 구성 예

다음 예제 BIND 영역 파일은 사용자 프로비저닝 클러스터의 역방향 이름 확인을 위한 샘플 PTR 레코드를 보여줍니다.

예 18.14. 역방향 레코드의 샘플 DNS 영역 데이터베이스

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR master0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR master1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR master2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com. 8
;
;EOF
```

1 Kubernetes API의 역방향 DNS 확인을 제공합니다. PTR 레코드는 API 로드 밸런서의 레코드 이름을 참조합니다.

2 Kubernetes API의 역방향 DNS 확인을 제공합니다. PTR 레코드는 API 로드 밸런서의 레코드 이름을 참조하며 내부 클러스터 통신에 사용됩니다.

3 부트스트랩 시스템의 역방향 DNS 확인을 제공합니다.

4 5 6 컨트롤 플레인 시스템의 역방향 DNS 확인을 제공합니다.

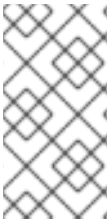
7 8 컴퓨팅 시스템의 역방향 DNS 확인을 제공합니다.

참고

OpenShift Container Platform 애플리케이션 와일드카드에는 **PTR** 레코드가 필요하지 않습니다.

18.6.5.6. 사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구사항

OpenShift Container Platform을 설치하기 전에 **API** 및 애플리케이션 인그레스 로드 밸런싱 인프라를 프로비저닝해야 합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 **API** 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.



참고

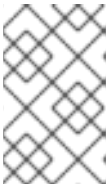
RHEL(Red Hat Enterprise Linux) 인스턴스를 사용하여 **API** 및 애플리케이션 인그레스 로드 밸런서를 배포하려면 **RHEL** 서브스크립션을 별도로 구매해야 합니다.

로드 밸런서 인프라는 다음 요구 사항을 충족해야 합니다.

1. **API** 로드 밸런서**:** 플랫폼과 상호 작용하고 플랫폼을 구성할 수 있도록 사용자(인간과 시스템 모두)에게 공통 끝점을 제공합니다. 다음 조건을 설정합니다.

   - **Layer 4** 로드 밸런싱 전용입니다. 이를 **Raw TCP, SSL Passthrough** 또는 **SSL Bridge** 모드라고 합니다. **SSL Bridge** 모드를 사용하는 경우**, API** 경로에 대해 **SNI(Server Name Indication,** 서버 이름 표시)를 활성화해야 합니다.

   - 스테이트리스 로드 밸런싱 알고리즘입니다. 옵션은 로드 밸런서 구현에 따라 달라집니다.

     참고

     API 로드 밸런서가 제대로 작동하기 위해 세션 지속성이 필요하지 않습니다.

   로드 밸런서의 전면과 후면 모두에서 다음 포트를 구성하십시오.

표 18.56. API 로드 밸런서

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
6443	부트스트랩 및 컨트롤 플레인. 부트스트랩 시스템이 클러스터 컨트롤 플레인을 초기화한 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다. API 서버 상태 검사 프로브에 대한 **/readyz** 끝점을 구성해야 합니다.	X	X	Kubernetes API 서버
22623	부트스트랩 및 컨트롤 플레인. 부트스트랩 시스템이 클러스터 컨트롤 플레인을 초기화한 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다.	X		시스템 구성 서버

참고

API 서버가 /readyz 엔드포인트를 해제하는 시점부터 풀에서 API 서버 인스턴스가 제거되는 시점까지 시간이 30초를 넘지 않도록 로드 밸런서를 구성해야 합니다. /readyz가 오류를 반환하거나 정상 상태가 된 후 정해진 시간 안에 끝점이 제거 또는 추가되어야 합니다. 5초 또는 10초의 프로빙 주기(두 번의 성공적인 요청은 정상 상태, 세 번의 요청은 비정상 상태)는 충분한 테스트를 거친 값입니다.

2. **애플리케이션 인그레스 로드 밸런서: 클러스터 외부에서 유입되는 애플리케이션 트래픽에 대한 인그래스 포인트를 제공합니다. 다음 조건을 설정합니다.**

- **Layer 4 로드 밸런싱 전용입니다. 이를 Raw TCP, SSL Passthrough 또는 SSL Bridge 모드라고 합니다. SSL Bridge 모드를 사용하는 경우 인그레스 경로에 대해 SNI(Server Name Indication, 서버 이름 표시)를 활성화해야 합니다.**

- **사용 가능한 옵션과 플랫폼에서 호스팅되는 애플리케이션 유형에 따라 연결 기반 또는 세션 기반 지속성이 권장됩니다.**

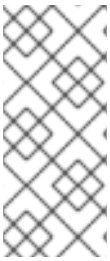
작은 정보

애플리케이션 인그레스 로드 밸런서에서 클라이언트의 실제 IP 주소를 확인할 수 있는 경우 소스 IP 기반 세션 지속성을 활성화하면 엔드 투 엔드 TLS 암호화를 사용하는 애플리케이션의 성능을 향상시킬 수 있습니다.

로드 밸런서의 전면과 후면 모두에서 다음 포트를 구성하십시오.

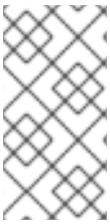
표 18.57. 애플리케이션 인그레스 로드 밸런서

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
443	기본적으로 인그레스 컨트롤러 pod, 컴퓨팅 또는 작업자를 실행하는 시스템입니다.	X	X	HTTPS 트래픽
80	기본적으로 인그레스 컨트롤러 pod, 컴퓨팅 또는 작업자를 실행하는 시스템입니다.	X	X	HTTP 트래픽
1936	기본적으로 Ingress 컨트롤러 Pod를 실행하는 작업자 노드입니다. 수신 상태 점검 프로브에 대해 **/healthz/ready** 끝점을 구성해야 합니다.	X	X	HTTP 트래픽



참고

컴퓨팅 노드가 0인 3-노드 클러스터를 배포하는 경우 Ingress 컨트롤러 Pod는 컨트롤 플레인 노드에서 실행됩니다. 3-노드 클러스터 배포에서 HTTP 및 HTTPS 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다.



참고

인그레스 라우터에 대한 작업 구성이 OpenShift Container Platform 클러스터에 필요합니다. 컨트롤 플레인 초기화 후 인그레스 라우터를 설정해야 합니다.

18.6.5.6.1. 사용자 프로비저닝 클러스터의 로드 밸런서 구성 예

이 섹션에서는 사용자 프로비저닝 클러스터의 로드 밸런싱 요구 사항을 충족하는 API 및 애플리케이션 수신 로드 밸런서 구성 예를 제공합니다. 샘플은 HAProxy 로드 밸런서에 대한 /etc/haproxy/haproxy.cfg 구성입니다. 이 예제에서는 하나의 로드 밸런싱 솔루션을 선택하기 위한 조언을 제공하는 것을 목적으로 하지 않습니다.

참고

이 예제에서는 Kubernetes API 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 API 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

예 18.15. API 및 애플리케이션 인그레스 로드 밸런서 구성 샘플

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
frontend stats
  bind *:1936
  mode            http
  log             global
  maxconn 10
  stats enable
  stats hide-version
  stats refresh 30s
  stats show-node
  stats show-desc Stats for ocp4 cluster (1)
  stats auth admin:ocp4
  stats uri /stats
listen api-server-6443 (2)
  bind *:6443
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:6443 check inter 1s backup (3)
  server master0 master0.ocp4.example.com:6443 check inter 1s
  server master1 master1.ocp4.example.com:6443 check inter 1s
  server master2 master2.ocp4.example.com:6443 check inter 1s
listen machine-config-server-22623 (4)
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup (5)
  server master0 master0.ocp4.example.com:22623 check inter 1s
  server master1 master1.ocp4.example.com:22623 check inter 1s
  server master2 master2.ocp4.example.com:22623 check inter 1s
listen ingress-router-443 (6)
  bind *:443
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:443 check inter 1s
  server worker1 worker1.ocp4.example.com:443 check inter 1s
listen ingress-router-80 (7)
  bind *:80
  mode tcp
```

```
balance source
server worker0 worker0.ocp4.example.com:80 check inter 1s
server worker1 worker1.ocp4.example.com:80 check inter 1s
```

1

이 예에서 클러스터 이름은 **ocp4** 입니다.

2

포트 **6443**은 **Kubernetes API** 트래픽을 처리하고 컨트롤 플레인 시스템을 가리킵니다.

3 5

부트스트랩 항목은 **OpenShift Container Platform** 클러스터 설치 전에 있어야 하며 부트스트랩 프로세스가 완료된 후 제거해야 합니다.

4

포트 **22623**은 머신 구성 서버 트래픽을 처리하고 컨트롤 플레인 시스템을 가리킵니다.

6

포트 **443**은 **HTTPS** 트래픽을 처리하고 **Ingress** 컨트롤러 **Pod**를 실행하는 시스템을 가리킵니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

7

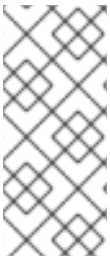
포트 **80**은 **HTTP** 트래픽을 처리하고 **Ingress** 컨트롤러 **Pod**를 실행하는 머신을 가리킵니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

참고

컴퓨팅 노드가 **0**인 **3-**노드 클러스터를 배포하는 경우 **Ingress** 컨트롤러 **Pod**는 컨트롤 플레인 노드에서 실행됩니다. **3-**노드 클러스터 배포에서 **HTTP** 및 **HTTPS** 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다.

작은 정보

HAProxy를 로드 밸런서로 사용하는 경우 **HAProxy** 노드에서 **netstat -nltupe**를 실행하여 **haproxy** 프로세스가 포트 **6443, 22623, 443** 및 **80**에서 수신 대기 중인지 확인할 수 있습니다.



참고

HAProxy를 로드 밸런서로 사용하고 SELinux가 enforcing으로 설정된 경우 HAProxy 서비스가 setsebool -P haproxy_connect_any=1을 실행하여 구성된 TCP 포트에 바인딩할 수 있는지 확인해야 합니다.

18.6.6. 사용자 프로비저닝 인프라 준비

사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치하기 전에 기본 인프라를 준비해야 합니다.

이 섹션에서는 OpenShift Container Platform 설치를 준비하기 위해 클러스터 인프라를 설정하는 데 필요한 높은 수준의 단계에 대해 자세히 설명합니다. 여기에는 클러스터 노드에 대한 IP 네트워킹 및 네트워크 연결 구성, 방화벽을 통해 필요한 포트 활성화, 필수 DNS 및 로드 밸런싱 인프라를 설정하는 작업이 포함됩니다.

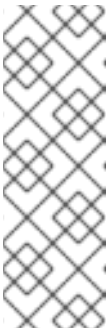
준비 후 클러스터 인프라는 *사용자 프로비저닝 인프라가 있는 클러스터의 요구 사항* 섹션에 설명된 요구 사항을 충족해야 합니다.

사전 요구 사항

- OpenShift Container Platform 4.x 테스트된 통합 페이지를 검토했습니다.
- *사용자 프로비저닝 인프라가 있는 클러스터의 요구 사항* 섹션에 자세히 설명된 인프라 요구 사항을 검토했습니다.

프로세스

1. DHCP를 사용하여 클러스터 노드에 IP 네트워킹 구성을 제공하는 경우 DHCP 서비스를 구성합니다.
   a. 노드의 영구 IP 주소를 DHCP 서버 구성에 추가합니다. 구성에서 관련 네트워크 인터페이스의 MAC 주소를 각 노드의 의도한 IP 주소와 일치시킵니다.
   b. DHCP를 사용하여 클러스터 시스템의 IP 주소 지정을 구성하는 경우 시스템은 DHCP를 통해 DNS 서버 정보도 가져옵니다. DHCP 서버 구성을 통해 클러스터 노드에서 사용하는 영구 DNS 서버 주소를 정의합니다.

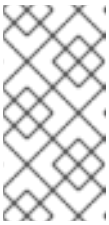


참고

DHCP 서비스를 사용하지 않는 경우 **RHCOS** 설치 시 **IP** 네트워킹 구성과 **DNS** 서버의 주소를 노드에 제공해야 합니다. **ISO** 이미지에서 설치하는 경우 부팅 인수로 전달할 수 있습니다. 고정 **IP** 프로비저닝 *및 고급 네트워킹 옵션에 대한 자세한 내용은 **RHCOS** 설치 및 **OpenShift Container Platform** 부트스트랩 프로세스 시작* 섹션을 참조하십시오.

c.

DHCP 서버 구성에 클러스터 노드의 호스트 이름을 정의합니다. 호스트 이름 고려 사항에 대한 자세한 내용은 ***DHCP**를 통해 클러스터 노드 호스트 이름 설정* 섹션을 참조하십시오.



참고

DHCP 서비스를 사용하지 않는 경우 클러스터 노드는 역방향 **DNS** 조회를 통해 호스트 이름을 가져옵니다.

2.

네트워크 인프라가 클러스터 구성 요소 간 필수 네트워크 연결을 제공하는지 확인합니다. 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 인프라 섹션의 네트워킹 요구 사항* 섹션을 참조하십시오.

3.

OpenShift Container Platform 클러스터 구성 요소가 통신하는 데 필요한 포트를 활성화하도록 방화벽을 구성합니다. 필요한 포트에 대한 자세한 내용은 *사용자 프로비저닝 인프라 섹션의 네트워킹 요구 사항* 섹션을 참조하십시오.

4.

클러스터에 필요한 **DNS** 인프라를 설정합니다.

a.

Kubernetes API, 애플리케이션 와일드카드, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템의 **DNS** 이름 확인을 구성합니다.

b.

Kubernetes API, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템에 대한 역방향 **DNS** 확인을 구성합니다.

OpenShift Container Platform DNS 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 **DNS** 요구 사항* 섹션을 참조하십시오.

5.

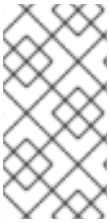
DNS 구성을 확인합니다.

a.
설치 노드에서 **Kubernetes API**의 레코드 이름, 와일드카드 경로 및 클러스터 노드에 대해 **DNS** 조회를 실행합니다. 응답의 **IP** 주소가 올바른 구성 요소에 해당하는지 확인합니다.

b.
설치 노드에서 로드 밸런서 및 클러스터 노드의 **IP** 주소에 대해 역방향 **DNS** 조회를 실행합니다. 응답의 레코드 이름이 올바른 구성 요소에 해당하는지 확인합니다.

자세한 **DNS** 검증 단계는 *사용자 프로비저닝 인프라에 대한 DNS 확인* 섹션을 참조하십시오.

6.
필요한 **API** 및 애플리케이션 수신 로드 밸런싱 인프라를 프로비저닝합니다. 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구 사항* 섹션을 참조하십시오.



참고

일부 로드 밸런싱 솔루션에는 로드 밸런싱을 초기화하기 전에 클러스터 노드의 **DNS** 이름을 확인해야 합니다.

18.6.7. 사용자 프로비저닝 인프라에 대한 DNS 확인 검증

사용자 프로비저닝 인프라에 **OpenShift Container Platform**을 설치하기 전에 **DNS** 구성을 확인할 수 있습니다.

중요

클러스터를 설치하기 전에 이 섹션에 설명된 검증 단계를 성공해야 합니다.

사전 요구 사항

- 사용자 프로비저닝 인프라에 필요한 **DNS** 레코드를 구성했습니다.

프로세스

1.
설치 노드에서 **Kubernetes API**의 레코드 이름, 와일드카드 경로 및 클러스터 노드에 대해 **DNS** 조회를 실행합니다. 응답에 포함된 **IP** 주소가 올바른 구성 요소에 해당하는지 확인합니다.

a.
Kubernetes API 레코드 이름을 조회합니다. 결과가 **API** 로드 밸런서의 **IP** 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
```

1

<nameserver_ip>를 네임서버의 IP 주소로, <cluster_name>을 클러스터 이름으로, <base_domain>을 기본 도메인 이름으로 바꿉니다.

출력 예

```
api.ocp4.example.com.  0 IN A 192.168.1.5
```

b.

Kubernetes 내부 API 레코드 이름을 조회합니다. 결과가 API 로드 밸런서의 IP 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
```

출력 예

```
api-int.ocp4.example.com.  0 IN A 192.168.1.5
```

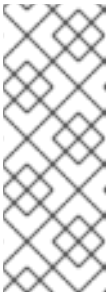
c.

예제 *.apps.<cluster_name>.<base_domain>을 테스트합니다. DNS 와일드카드를 조회합니다. 모든 애플리케이션 와일드카드 조회는 애플리케이션 인그레스 로드 밸런서의 IP 주소로 확인되어야 합니다.

```
$ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.
<base_domain>
```

출력 예

```
random.apps.ocp4.example.com.  0 IN A 192.168.1.5
```



참고

예제 출력에서는 **Kubernetes API** 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 **API** 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

random 항목을 다른 와일드카드 값으로 교체할 수 있습니다. 예를 들어 **OpenShift Container Platform** 콘솔의 경로를 쿼리할 수 있습니다.

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

출력 예

```
console-openshift-console.apps.ocp4.example.com. 0 IN A 192.168.1.5
```

d. 부트스트랩 **DNS** 레코드 이름에 대해 조회를 실행합니다. 결과가 부트스트랩 노드의 **IP** 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.
<base_domain>
```

출력 예

```
bootstrap.ocp4.example.com.  0 IN A 192.168.1.96
```

e. 이 방법을 사용하여 컨트롤 플레인 및 컴퓨팅 노드의 **DNS** 레코드 이름에 대해 조회를 수행합니다. 결과가 각 노드의 **IP** 주소에 해당하는지 확인합니다.

2.

설치 노드에서 로드 밸런서 및 클러스터 노드의 IP 주소에 대해 역방향 DNS 조회를 실행합니다. 응답에 포함된 레코드 이름이 올바른 구성 요소에 해당하는지 확인합니다.

a.

API 로드 밸런서의 IP 주소에 대해 역방향 조회를 수행합니다. 응답에 Kubernetes API 및 Kubernetes 내부 API의 레코드 이름이 포함되어 있는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

출력 예

```
5.1.168.192.in-addr.arpa. 0 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 0 IN PTR api.ocp4.example.com. 2
```

1

Kubernetes 내부 API의 레코드 이름을 제공합니다.

2

Kubernetes API의 레코드 이름을 제공합니다.

참고

OpenShift Container Platform 애플리케이션 와일드카드에는 PTR 레코드가 필요하지 않습니다. 애플리케이션 인그레스 로드 밸런서의 IP 주소에 대한 역방향 DNS 확인에는 유효성 검사 단계가 필요하지 않습니다.

b.

부트스트랩 노드의 IP 주소에 대해 역방향 조회를 수행합니다. 결과가 부트스트랩 노드의 DNS 레코드 이름을 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

출력 예

```
96.1.168.192.in-addr.arpa. 0 IN PTR bootstrap.ocp4.example.com.
```

c. 이 방법을 사용하여 컨트롤 플레인 및 컴퓨팅 노드의 IP 주소에 대해 역방향 조회를 수행합니다. 결과가 각 노드의 DNS 레코드 이름과 일치하는지 확인합니다.

18.6.8. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

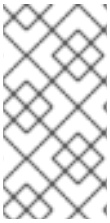
OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 core 사용자의 ~/.ssh/authorized_keys 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 core로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. ./openshift-install gather 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.



참고

AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1

새 SSH 키의 경로 및 파일 이름(예: ~/.ssh/id_ed25519)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 '~/.ssh 디렉터리에 있는지 확인하십시오.

참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 ed25519 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 rsa 또는 ecdsa 알고리즘을 사용하는 키를 생성합니다.

2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 ~/.ssh/id_ed25519.pub 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 ./openshift-install gather 명령을 사용하려는 경우 필요합니다.

참고

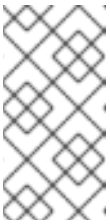
일부 배포에서는 ~/.ssh/id_rsa 및 ~/.ssh/id_dsa와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. ssh-agent 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```



참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. ssh-agent에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: ~/.ssh/id_ed25519).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다. 프로비저닝하는 인프라에 클러스터를 설치하는 경우 설치 프로그램에 키를 제공해야 합니다.

18.6.9. 설치 프로그램 받기

OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

프로세스

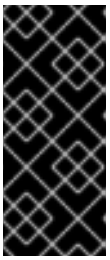
1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.

2. 인프라 공급자를 선택합니다.

3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.

중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.



중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 OpenShift Container Platform 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

18.6.10. 수동으로 설치 구성 파일 만들기

OpenShift Container Platform을 사용자가 프로비저닝한 설치의 경우 설치 구성 파일을 수동으로 생성합니다.

사전 요구 사항

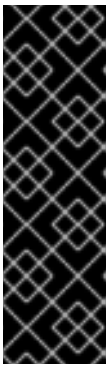
- 로컬 시스템에 설치 프로그램에 제공할 **SSH** 공개 키가 있습니다. 키는 디버깅 및 재해 복구를 위해 클러스터 노드에 대한 **SSH** 인증에 사용됩니다.

- **OpenShift Container Platform** 설치 프로그램과 클러스터의 풀 시크릿이 있습니다.

프로세스

1. 필요한 설치 자산을 저장할 설치 디렉터리를 만듭니다.

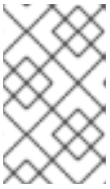
```
$ mkdir <installation_directory>
```



중요

디렉터리를 만들어야 합니다. 부트스트랩 **X.509** 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 **OpenShift Container Platform** 버전에서 설치 파일을 복사할 때는 주의하십시오.

2. 샘플 **install-config.yaml** 파일 템플릿을 사용자 지정하여 **<installation_directory>**에 저장합니다.



참고

이 설정 파일의 이름을 **install-config.yaml**로 지정해야 합니다.

참고

일부 플랫폼 유형의 경우 대체로 **./openshift-install create install-config --dir <installation_directory>** 를 실행하여 **install-config.yaml** 파일을 생성할 수 있습니다. 프롬프트에서 클러스터 구성에 대한 세부 정보를 제공할 수 있습니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.

중요

install-config.yaml 파일은 설치 과정의 다음 단계에서 사용됩니다. 이 시점에서 이를 백업해야 합니다.

18.6.10.1. VMware vSphere용 샘플 install-config.yaml 파일

install-config.yaml 파일을 사용자 지정하여 **OpenShift Container Platform** 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.

```
apiVersion: v1
baseDomain: example.com 1
compute:
- hyperthreading: Enabled 2 3
  name: worker
  replicas: 0 4
controlPlane:
  hyperthreading: Enabled 5 6
  name: master
  replicas: 3 7
metadata:
  name: test 8
platform:
  vsphere:
    vcenter: your.vcenter.server 9
    username: username 10
    password: password 11
    datacenter: datacenter 12
    defaultDatastore: datastore 13
    folder: "/<datacenter_name>/vm/<folder_name>/<subfolder_name>" 14
fips: false 15
pullSecret: '{"auths": ...}' 16
sshKey: 'ssh-ed25519 AAAA...' 17
```

1 클러스터의 기본 도메인입니다. 모든 **DNS** 레코드는 이 기본 도메인의 하위 도메인이어야 하며 클러스터 이름을 포함해야 합니다.

2 5 **controlPlane** 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 현재 두 섹션이 모두 단일 시스

템 풀을 정의하지만 향후 출시되는 **OpenShift Container Platform** 버전은 설치 과정에서 여러 컴퓨팅 풀 정의를 지원할 수 있습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

3 6

동시 멀티스레딩 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. 일부 클러스터 시스템에서 동시 멀티스레딩을 비활성화할 경우에는 해당 멀티스레딩을 모든 클러스터 시스템에서 비활성화해야 합니다.

중요

동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다. 동시 멀티스레딩을 비활성화하는 경우 시스템은 8개 이상의 **CPU**와 **32GB**의 **RAM**을 사용해야 합니다.

4

replicas 매개변수의 값을 **0**으로 설정해야 합니다. 이 매개변수는 클러스터를 생성하고 관리하는 작업자 수를 제어합니다. 이는 사용자 프로비저닝 인프라를 사용할 때 클러스터가 실행하지 않는 기능입니다. **OpenShift Container Platform** 설치를 완료하기 전에 클러스터에서 사용할 작업자 시스템을 수동으로 배포해야 합니다.

7

클러스터에 추가하는 컨트롤 플레인 시스템의 수입니다. 클러스터에서 이 값을 클러스터의 **etcd** 끝점 수로 사용하므로 이 값은 배포하는 컨트롤 플레인 시스템의 수와 일치해야 합니다.

8

DNS 레코드에 지정한 클러스터 이름입니다.

9

vCenter 서버의 정규화된 호스트 이름 또는 **IP** 주소입니다.

10

서버에 액세스하기 위한 사용자의 이름입니다. 이 사용자에게는 최소한 **vSphere**에서 정적 또는 동적 영구 볼륨 프로비저닝 에 필요한 역할과 권한이 있어야 합니다.

11

vSphere 사용자와 연결된 암호입니다.

12

13

사용할 기본 **vSphere** 데이터 저장소입니다.

14

선택사항: 설치 관리자 프로비저닝 인프라의 경우 설치 프로그램이 가상 머신을 생성하는 기존 폴더의 절대 경로(예: **/<datacenter_name>/vm/<folder_name>/<subfolder_name>**). 이 값을 제공하지 않으면 설치 프로그램이 인프라 **ID**로 이름이 지정된 데이터 센터 가상 머신 폴더에 최상위 폴더를 만듭니다. 클러스터에 인프라를 제공하는 경우에는 이 매개변수를 생략합니다.

15

FIPS 모드 활성화 또는 비활성화 여부입니다. 기본적으로 **FIPS** 모드는 비활성화됩니다. **FIPS** 모드가 활성화되면 **OpenShift Container Platform**이 실행되는 **RHCOS(Red Hat Enterprise Linux CoreOS)** 시스템에서 기본 **Kubernetes** 암호화 제품군은 우회하고 **RHCOS**와 함께 제공되는 암호화 모듈을 대신 사용합니다.

중요

FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 **OpenShift Container Platform** 배포에서만 지원됩니다.

16

OpenShift Cluster Manager 에서 얻은 풀 시크릿입니다. 이 풀 시크릿을 사용하면 **OpenShift Container Platform** 구성 요소에 대한 컨테이너 이미지를 제공하는 **Quay.io**를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

17

RHCOS(Red Hat Enterprise Linux CoreOS)의 **core** 사용자에 대한 기본 **SSH** 키의 공용 부분입니다.

18.6.10.2. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 **HTTP** 또는 **HTTPS** 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 **OpenShift Container Platform** 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.

- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 **API**에 대한 호출을 포함하여 모든 클러스터 발신(**Egress**) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.

참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 **Red Hat OpenStack Platform (RHOSP)**에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. 클러스터 외부에서 **HTTP** 연결을 구축하는 데 사용할 프록시 **URL**입니다. **URL** 스키마는 **http**여야 합니다.

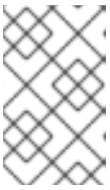
2. 클러스터 외부에서 **HTTPS** 연결을 구축하는 데 사용할 프록시 **URL**입니다.

3

대상 도메인 이름, IP 주소 또는 프록시에서 제외할 기타 네트워크 CIDR로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 .을 입력합니다. 예를 들어, .y.com은 x.y.com과 일치하지만 y.com은 일치하지 않습니다. *를 사용하여 모든 대상에 대해 프록시를 바이패스합니다. vCenter의 IP 주소와 해당 시스템에 사용하는 IP 범위를 포함해야 합니다.

4

이 값을 제공하면 설치 프로그램에서 추가 CA 인증서를 보유할 openshift -config 네임스페이스에 user-ca- bundle 이라는 구성 맵을 생성합니다. additionalTrustBundle 및 하나 이상의 프록시 설정을 제공하는 경우 프록시 오브젝트는 trustedCA 필드의 user-ca-bundle 구성 맵을 참조하도록 구성됩니다. 그러면 Cluster Network Operator에서 trustedCA 매개변수에 대해 지정된 콘텐츠를 RHCOS 신뢰 번들과 병합하는 trusted-ca-bundle 구성 맵을 생성합니다. 프록시의 ID 인증서를 RHCOS 트러스트 번들에 있는 기관에서 서명하지 않은 경우 additionalTrustBundle 필드가 있어야 합니다.

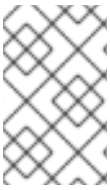


참고

설치 프로그램에서 프록시 adinessEndpoints 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 OpenShift Container Platform을 설치할 때 참조하십시오.

제공되는 install-config.yaml 파일의 프록시 설정을 사용하는 cluster라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 cluster Proxy 오브젝트는 계속 생성되지만 spec은 nil이 됩니다.



참고

cluster라는 Proxy 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

18.6.11. Kubernetes 매니페스트 및 Ignition 설정 파일 생성

일부 클러스터 정의 파일을 수정하고 클러스터 시스템을 수동으로 시작해야 하므로 클러스터가 시스템을 구성하는 데 필요한 Kubernetes 매니페스트 및 Ignition 구성 파일을 사용자가 생성해야 합니다.

설치 구성 파일은 Kubernetes 매니페스트로 변환됩니다. 매니페스트는 나중에 클러스터 머신을 구성하는 데 사용되는 Ignition 구성 파일로 래핑됩니다.

중요

- **OpenShift Container Platform** 설치 프로그램에서 생성하는 **Ignition** 구성 파일에 **24**시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 **24**시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 **kubelet** 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(**CSR**)을 수동으로 승인해야 합니다. 자세한 내용은 ***Recovering from expired control plane certificates*** 문서를 참조하십시오.

- 클러스터를 설치한 후 **24**시간에서 **22**시간까지의 인증서가 교체되기 때문에 생성된 후 **12**시간 이내에 **Ignition** 구성 파일을 사용하는 것이 좋습니다. **12**시간 이내에 **Ignition** 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

사전 요구 사항

- **OpenShift Container Platform** 설치 프로그램을 가져오셨습니다.

- **install-config.yaml** 설치 구성 파일을 생성하셨습니다.

프로세스

1. **OpenShift Container Platform** 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터에 대한 **Kubernetes** 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>**는 사용자가 만든 **install-config.yaml** 파일이 포함된 설치 디렉터리를 지정합니다.

2. 컨트롤 플레인 시스템 및 컴퓨팅 머신 세트를 정의하는 **Kubernetes** 매니페스트 파일을 제거합니다.

```
$ rm -f openshift/99_openshift-cluster-api_master-machines-*.yaml
openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

이러한 리소스는 사용자가 직접 생성하고 관리하기 때문에 초기화할 필요가 없습니다.

- 시스템 API로 머신 세트 파일을 보존하여 컴퓨팅 시스템을 생성할 수 있지만 사용자 환경과 일치하도록 해당 참조를 업데이트해야 합니다.

3. <installation_directory>/manifests/cluster-scheduler-02-config.yml Kubernetes 매니페스트 파일의 mastersSchedulable 매개변수가 false로 설정되어 있는지 확인합니다. 이 설정으로 인해 컨트롤 플레인 머신에서 포드가 예약되지 않습니다.

   a. <installation_directory>/manifests/cluster-scheduler-02-config.yml 파일을 엽니다.

   b. mastersSchedulable 매개변수를 찾아서 값을 False로 설정되어 있는지 확인합니다.

   c. 파일을 저장하고 종료합니다.

4. Ignition 구성 파일을 생성하려면 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행합니다.

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 <installation_directory>는 동일한 설치 디렉터리를 지정합니다.

설치 디렉터리의 부트스트랩, 컨트롤 플레인 및 컴퓨팅 노드에 대한 Ignition 구성 파일이 생성됩니다. kubeadmin-password 및 kubeconfig 파일은 ./<installation_directory>/auth 디렉터리에 생성됩니다.

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

18.6.12. 인프라 이름 추출

Ignition 구성 파일에는 AWS의 VMware Cloud에서 클러스터를 고유하게 식별하는 데 사용할 수 있는 고유한 클러스터 식별가 포함되어 있습니다. 클러스터 ID를 가상 머신 폴더의 이름으로 사용하려면 해당 ID를 추출해야 합니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받으셨습니다.
- 클러스터에 대한 Ignition 구성 파일을 생성하셨습니다.
- jq CLI를 설치하셨습니다.

프로세스

- Ignition 구성 파일 메타데이터에서 인프라 이름을 추출하여 확인하려면 다음 명령을 실행하십시오.

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

출력 예

```
openshift-vw9j6 1
```

1 이 명령의 출력은 클러스터 이름과 임의의 문자열입니다.

18.6.13. RHCOS 설치 및 OpenShift Container Platform 부트스트랩 프로세스 시작

VMware vSphere의 사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치하려면 vSphere 호스트에 RHCOS(Red Hat Enterprise Linux Core OS)를 설치해야 합니다. RHCOS를 설치할 때 설치 중인 머신 유형에 대해 OpenShift Container Platform 설치 프로그램에서 생성한 Ignition 구성 파일을 제공해야 합니다. 적합한 네트워킹, DNS 및 로드 밸런싱 인프라를 구성한 경우 RHCOS 머신이 재부팅된 후 OpenShift Container Platform 부트스트랩 프로세스가 자동으로 시작됩니다.

사전 요구 사항

- 클러스터에 대한 Ignition 구성 파일이 있어야 합니다.
- 사용자 컴퓨터에서 액세스할 수 있고 생성한 시스템이 액세스할 수 있는 HTTP 서버에 액세스할 수 있습니다.
- vSphere 클러스터 를 생성했습니다.

프로세스

1. 설치 프로그램에서 생성된 부트스트랩 Ignition 구성 파일 (<installation_directory>/bootstrap.ign)을 HTTP 서버에 업로드합니다. 이 파일의 URL을 기록해 둡니다.
2. 부트스트랩 노드의 다음 보조 Ignition 구성 파일을 <installation_directory>/merge-bootstrap.ign으로 컴퓨터에 저장합니다.

```
{
  "ignition": {
    "config": {
      "merge": [
        {
          "source": "<bootstrap_ignition_config_url>", 1
          "verification": {}
        }
      ]
    },
    "timeouts": {},
    "version": "3.2.0"
  },
  "networkd": {},
  "passwd": {},
  "storage": {},
  "systemd": {}
}
```

1

호스팅한 부트스트랩 Ignition 구성 파일의 URL을 지정합니다.

부트스트랩 머신에 대한 VM(가상 머신)을 생성할 때 이 Ignition 구성 파일을 사용합니다.

3. 설치 프로그램이 생성한 다음 Ignition 구성 파일을 찾습니다.

- <installation_directory>/master.ign
- <installation_directory>/worker.ign
- <installation_directory>/merge-bootstrap.ign

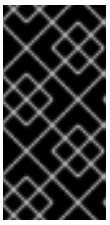
4. Ignition 구성 파일을 Base64 인코딩으로 변환합니다. 이 절차의 뒷부분에서는 이러한 파일을 VM의 추가 구성 매개변수 guestinfo.ignition.config.data 에 추가해야 합니다.

예를 들어 Linux 운영 체제를 사용하는 경우 base64 명령을 사용하여 파일을 인코딩할 수 있습니다.

```
$ base64 -w0 <installation_directory>/master.ign > <installation_directory>/master.64
```

```
$ base64 -w0 <installation_directory>/worker.ign > <installation_directory>/worker.64
```

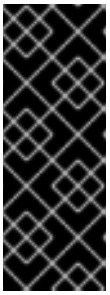
```
$ base64 -w0 <installation_directory>/merge-bootstrap.ign >
<installation_directory>/merge-bootstrap.64
```



중요

설치를 마친 후 클러스터에 컴퓨팅 시스템을 더 추가하려면 Ignition 구성 파일을 삭제하지 마십시오.

5. RHCOS OVA 이미지를 가져옵니다. RHCOS 이미지 미러 페이지에서 이미지를 사용할 수 있습니다.



중요

RHCOS 이미지는 **OpenShift Container Platform** 릴리스에 따라 변경되지 않을 수 있습니다. 설치하는 **OpenShift Container Platform** 버전과 같거나 그 이하의 버전 중 가장 최신 버전의 이미지를 다운로드해야 합니다. 지원되는 경우 **OpenShift Container Platform** 버전과 일치하는 이미지 버전을 사용합니다.

파일 이름에는 **rhcos-vmware.<architecture>.ova** 형식의 **OpenShift Container Platform** 버전 번호가 포함됩니다.

6. **vSphere Client**에서 **VM**을 저장할 데이터 센터 폴더를 생성합니다.

   a. **VMs and Templates** 보기를 클릭합니다.

   b. 데이터 센터 이름을 마우스 오른쪽 버튼으로 클릭합니다.

   c. **New Folder → New VM and Template Folder**를 클릭합니다.

   d. 표시되는 창에서 폴더 이름을 입력합니다. **install-config.yaml** 파일에서 기존 폴더를 지정하지 않은 경우 인프라 **ID**와 동일한 이름으로 폴더를 생성합니다. **vCenter**는 이 폴더 이름을 사용하여 **Workspace** 구성에 적절한 위치에 스토리지를 동적으로 프로비저닝합니다.

7. **vSphere Client**에서 **OVA** 이미지에 대한 템플릿을 생성한 다음 필요에 따라 템플릿을 복제합니다.

참고

다음 단계에서 템플릿을 생성한 다음 모든 클러스터 시스템에 대한 템플릿을 복제합니다. 그런 다음 **VM**을 프로비저닝할 때 복제된 머신 유형의 **Ignition** 구성 파일의 위치를 제공합니다.

   a. **Hosts and Clusters** 탭에서 클러스터 이름을 마우스 오른쪽 버튼으로 클릭하고 **Deploy OVF Template**을 선택합니다.

   .

b. **Select an OVF** 탭에서 다운로드한 **RHCOS OVA** 파일의 이름을 지정합니다.

c. 이름 및 폴더 선택 탭에서 템플릿의 가상 시스템 이름 (예: **Template-RHCOS**)을 설정합니다. **vSphere** 클러스터의 이름을 클릭하고 이전 단계에서 생성한 폴더를 선택합니다.

d. **Select a compute resource** 탭에서 **vSphere** 클러스터 이름을 클릭합니다.

e. **Select storage** 탭에서 **VM**의 스토리지 옵션을 구성합니다.

- 스토리지 기본 설정에 따라 **Thin Provision** 또는 **Thick Provision**을 선택합니다.
- **install-config.yaml** 파일에서 지정한 데이터 저장소를 선택합니다.

f. **Select network** 탭에서 사용 가능한 경우 클러스터에 대해 구성한 네트워크를 지정합니다.

g. **OVF** 템플릿을 생성할 때 템플릿 사용자 지정 탭에 값을 지정하지 않거나 템플릿을 추가로 구성하지 마십시오.

중요

원래 **VM** 템플릿을 시작하지 마십시오. **VM** 템플릿이 꺼져 있어야 하며 새 **RHCOS** 머신에 대해 복제해야 합니다. **VM** 템플릿을 시작하면 **VM** 템플릿이 플랫폼의 **VM**으로 구성되므로 시스템 세트가 구성을 적용할 수 있는 템플릿으로 사용되지 않습니다.

8. 선택 사항: 필요한 경우 **VM** 템플릿에서 구성된 가상 하드웨어 버전을 업데이트합니다. 자세한 내용은 가상 머신을 **VMware** 설명서의 최신 하드웨어 버전으로 업그레이드를 참조하십시오.

중요

필요한 경우 **VM** 템플릿의 하드웨어 버전을 버전 **15**로 업데이트하는 것이 좋습니다. 이제 **vSphere**에서 실행 중인 클러스터 노드에 하드웨어 버전 **13**을 사용하는 것이 더 이상 사용되지 않습니다. 가져온 템플릿의 기본값이 하드웨어 버전 **13**인 경우 **VM** 템플릿을 하드웨어 버전 **15**로 업그레이드하기 전에 **ESXi** 호스트가 **6.7U3** 이상인지 확인해야 합니다. **vSphere** 버전이 **6.7U3** 미만인 경우 이 업그레이드 단계를 건너뛸 수 있습니다. 그러나 향후 **OpenShift Container Platform** 버전은 **6.7U3** 미만의 하드웨어 버전 **13** 및 **vSphere** 버전에 대한 지원을 제거하도록 예약되어 있습니다.

9. 템플릿이 배포된 후 클러스터에서 시스템의 가상 머신을 배포합니다.

a. 템플릿 이름을 마우스 오른쪽 버튼으로 클릭하고 **Clone → Clone to Virtual Machine**을 클릭합니다.

b. **Select a name and folder** 탭에서 가상 머신의 이름을 지정합니다. **control-plane-0** 또는 **compute-1**과 같은 시스템 유형을 이름에 포함할 수 있습니다.

c. **Select a name and folder** 탭에서 클러스터에 대해 생성한 폴더의 이름을 선택합니다.

d. **Select a compute resource** 탭에서 데이터 센터의 호스트 이름을 선택합니다.

e. 선택사항: **Select storage** 탭에서 스토리지 옵션을 사용자 지정합니다.

f. **Select clone options**에서 **Customize this virtual machine's hardware**를 선택합니다.

g. **Customize hardware** 탭에서 **VM Options → Advanced**를 클릭합니다.

- 선택 사항: **vSphere**에서 기본 **DHCP** 네트워킹을 재정의합니다. 고정 **IP** 네트워킹을 활성화하려면 다음을 수행합니다.

  i. 고정 **IP** 구성을 설정합니다.

```
$ export IPCFG="ip=<ip>::<gateway>:<netmask>:<hostname>:
<iface>:none nameserver=srv1 [nameserver=srv2 [nameserver=srv3 [...]]]"
```

명령 예

```
$ export IPCFG="ip=192.168.100.101::192.168.100.254:255.255.255.0:::none
nameserver=8.8.8.8"
```

ii. guestinfo.afterburn.initrd.network-kargs 속성을 설정한 후 vSphere의 OVA에서 VM을 부팅합니다.

```
$ govc vm.change -vm "<vm_name>" -e
"guestinfo.afterburn.initrd.network-kargs=${IPCFG}"
```

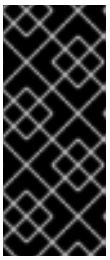
- 선택사항: 클러스터 성능 문제가 발생하는 경우 Latency Sensitivity 목록에서 High를 선택합니다. VM의 CPU 및 메모리 예약에 다음과 같은 값이 있는지 확인합니다.
  - 메모리 예약 값은 구성된 메모리 크기와 같아야 합니다.
  - CPU 예약 값은 측정된 물리적 CPU 속도를 곱한 최소 대기 시간이 짧은 가상 CPU 수여야 합니다.
- 구성 편집을 클릭하고 Configuration Parameters 창에서 사용 가능한 매개 변수 목록을 검색하여 스틸 클럭 회계 (stealclock.enable)를 검색합니다. 사용 가능한 경우 해당 값을 TRUE 로 설정합니다. 스틸링 클럭 계정을 사용하면 클러스터 문제 해결에 도움이 될 수 있습니다.
- 구성 추가 매개변수를 클릭합니다. 다음 매개변수 이름 및 값을 정의합니다.
  - guestinfo.ignition.config.data: 이 절차에서 이전에 생성한 base-64 인코딩 파일을 찾고 이 머신 유형에 대해 base64로 인코딩된 Ignition 구성 파일의 내용을 붙여넣습니다.
  -

- guestinfo.ignition.config.data.encoding: base64를 지정합니다.
- disk.EnableUUID: TRUE를 지정합니다.
- stealclock.enable:이 매개 변수가 정의되지 않은 경우 추가하고 TRUE 를 지정합니다.

h. Customize hardware 탭의 Virtual Hardware 패널에서 지정된 값을 필요에 따라 수정합니다. RAM, CPU 및 디스크 스토리지의 양이 시스템 유형에 대한 최소 요구사항을 충족하는지 확인합니다.

i. 구성을 완료하고 VM의 전원을 켭니다.

10. 각 시스템에 대해 이전 단계에 따라 클러스터의 나머지 시스템을 생성합니다.



중요

이때 부트스트랩 및 컨트롤 플레인 시스템을 생성해야 합니다. 일부 Pod는 기본적으로 컴퓨팅 시스템에 배포되므로 클러스터를 설치하기 전에 컴퓨팅 시스템을 두 개 이상 생성합니다.

18.6.14. vSphere의 클러스터에 더 많은 컴퓨팅 머신 추가

VMware vSphere에서 사용자가 프로비저닝한 OpenShift Container Platform 클러스터에 더 많은 컴퓨팅 머신을 추가할 수 있습니다.

사전 요구 사항

- 컴퓨팅 머신의 base64로 인코딩된 Ignition 파일을 가져옵니다.
- 클러스터에 생성한 vSphere 템플릿에 액세스할 수 있습니다.

프로세스

1. 템플릿이 배포된 후 클러스터에서 시스템의 가상 머신을 배포합니다.

a. 템플릿 이름을 마우스 오른쪽 버튼으로 클릭하고 **Clone → Clone to Virtual Machine**을 클릭합니다.

b. **Select a name and folder** 탭에서 가상 머신의 이름을 지정합니다. **compute-1**과 같은 머신 유형을 이름에 포함할 수 있습니다.

c. **Select a name and folder** 탭에서 클러스터에 대해 생성한 폴더의 이름을 선택합니다.

d. **Select a compute resource** 탭에서 데이터 센터의 호스트 이름을 선택합니다.

e. 선택사항: **Select storage** 탭에서 스토리지 옵션을 사용자 지정합니다.

f. **Select clone options**에서 **Customize this virtual machine's hardware**를 선택합니다.

g. **Customize hardware** 탭에서 **VM Options → Advanced**를 클릭합니다.

- **Latency Sensitivity** 목록에서 **High**를 선택합니다.
- **Edit Configuration**을 클릭하고 **Configuration Parameters** 창에서 **Add Configuration Params**를 클릭합니다. 다음 매개변수 이름 및 값을 정의합니다.
  - **guestinfo.ignition.config.data:** 이 머신 유형에 대해 **base64**로 인코딩된 컴퓨팅 **Ignition** 구성 파일의 내용을 붙여넣습니다.
  - **guestinfo.ignition.config.data.encoding: base64**를 지정합니다.
  - **disk.EnableUUID: TRUE**를 지정합니다.

h.

Customize hardware 탭의 Virtual Hardware 패널에서 지정된 값을 필요에 따라 수정합니다. RAM, CPU 및 디스크 스토리지의 양이 시스템 유형에 대한 최소 요구사항을 충족하는지 확인합니다. 또한 사용 가능한 네트워크가 여러 개인 경우 Add network adapter에서 올바른 네트워크를 선택해야 합니다.

i. 구성을 완료하고 VM의 전원을 켭니다.

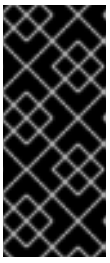
2. 계속해서 클러스터에 추가 컴퓨팅 머신을 만듭니다.

18.6.15. 디스크 파티션 설정

대부분의 경우 데이터 파티션은 원래 다른 운영 체제를 설치하는 대신 RHCOS를 설치하여 생성됩니다. 이러한 경우 OpenShift Container Platform 설치 프로그램은 디스크 파티션을 설정할 수 있어야합니다.

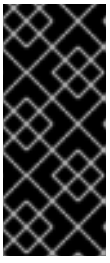
그러나 OpenShift Container Platform 노드를 설치할 때 기본 파티션 설정을 덮어 쓰기하기 위해 개입이 필요한 두 가지 경우가 있습니다.

- 별도의 파티션 생성: 빈 디스크에 그린 필드 설치의 경우 파티션에 별도의 스토리지를 추가할 수 있습니다. 이는 /var 또는 /var의 하위 디렉터리 (예: /var/lib/etcd)중 하나를 별도의 파티션으로 만드는 경우에 공식적으로 지원됩니다.



중요

100GB 이상의 디스크 크기 및 특히 1TB보다 큰 디스크 크기의 경우 별도의 /var 파티션을 만듭니다. 자세한 내용은 " 별도의 /var 파티션 생성" 및 이 Red Hat 지식베이스 문서 를 참조하십시오.



중요

Kubernetes는 두 개의 파일 시스템 파티션만 지원합니다. 원래 구성에 둘 이상의 파티션을 추가하는 경우 Kubernetes는 모든 파티션을 모니터링할 수 없습니다.

- 기존 파티션 유지: 브라운 필드 설치에서 기존 노드에 OpenShift Container Platform을 재설치하고 이전 운영 체제에 설치된 데이터 파티션을 유지해야 하는 경우 기존 데이터 파티션을 유지할 수 있는 coreos-installer에 부팅 인수와 옵션이 모두 있습니다.

별도의 /var 파티션 만들기

일반적으로 OpenShift Container Platform의 디스크 파티션 설정은 설치 프로그램에 맡겨야합니다. 그러나 확장하려는 파일 시스템의 일부에 별도의 파티션을 생성해야 하는 경우가 있습니다.

OpenShift 컨테이너 플랫폼은 /var 파티션 또는 /var의 하위 디렉터리 중 하나에 스토리지를 연결하는 단일 파티션의 추가를 지원합니다. 예를 들면 다음과 같습니다.

- /var/lib/containers: 시스템에 더 많은 이미지와 컨테이너가 추가됨에 따라 확장될 수 있는 컨테이너 관련 콘텐츠를 보관합니다.
- /var/lib/etcd: etcd 스토리지의 성능 최적화와 같은 목적으로 별도로 보관할 데이터를 보관합니다.
- /var: 감사 등의 목적에 맞게 별도로 분리하여 보관해야 하는 데이터를 보관합니다.

중요

100GB보다 큰 디스크 크기 및 특히 1TB보다 큰 디스크의 경우 별도의 /var 파티션을 만듭니다.

/var 디렉터리의 콘텐츠를 별도로 저장하면 필요에 따라 해당 영역에 대한 스토리지 확장을 보다 용이하게 하고 나중에 OpenShift Container Platform을 다시 설치하여 해당 데이터를 그대로 보존할 수 있습니다. 이 방법을 사용하면 모든 컨테이너를 다시 가져올 필요가 없으며 시스템을 업데이트할 때 대용량 로그 파일을 복사할 필요도 없습니다.

RHCOS(Red Hat Enterprise Linux CoreOS)를 새로 설치하기 전에 /var가 있어야 하므로 다음 절차에서는 OpenShift Container Platform 설치의 openshift-install 준비 단계 중에 삽입되는 머신 구성 매니페스트를 생성하여 별도의 /var 파티션을 설정합니다.

절차

1. OpenShift Container Platform 설치 파일을 저장할 디렉터리를 만듭니다.

```
$ mkdir $HOME/clusterconfig
```

2. openshift-install을 실행하여 manifest 및 openshift 하위 디렉터리에 파일 세트를 만듭니

다. 프롬프트가 표시되면 시스템 질문에 대답합니다.

```
$ openshift-install create manifests --dir $HOME/clusterconfig
? SSH Public Key ...
$ ls $HOME/clusterconfig/openshift/
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

3. 추가 파티션을 구성하는 **Butane** 구성을 생성합니다. 예를 들어 파일 이름을 **$HOME/clusterconfig/98-var-partition.bu** 로 지정하고, 디스크 장치 이름을 **worker** 시스템의 스토리지 장치 이름으로 변경하고 스토리지 크기를 적절하게 설정합니다. 이 예에서는 **/var** 디렉토리를 별도의 파티션에 배치합니다.

```
variant: openshift
version: 4.9.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1 파티션을 설정해야하는 디스크 저장 장치 이름입니다.

2 데이터 파티션을 부트 디스크에 추가할 때 최소 **25000** 메비 바이트가 권장됩니다. 루트 파일 시스템은 지정된 오프셋까지 사용 가능한 모든 공간을 채우기 위해 자동으로 크기가 조정됩니다. 값이 지정되지 않거나 지정된 값이 권장 최소값보다 작으면 생성되는 루트 파일 시스템의 크기가 너무 작아지고 **RHCOS**를 나중에 다시 설치할 때 데이터 파티션의 첫 번째 부분을 덮어 쓸 수 있습니다.

3

데이터 파티션의 크기(MB)입니다.

4

컨테이너 스토리지에 사용되는 파일 시스템에 **prjquota** 마운트 옵션을 활성화해야 합니다.

참고

별도의 **/var** 파티션을 만들 때 다른 인스턴스 유형에 동일한 장치 이름이 없는 경우 작업자 노드에 다른 인스턴스 유형을 사용할 수 없습니다.

4. **Butane** 구성에서 매니페스트를 생성하여 **clusterconfig/openshift** 디렉터리에 저장합니다. 예를 들어 다음 명령을 실행합니다.

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o
$HOME/clusterconfig/openshift/98-var-partition.yaml
```

5. **openshift-install**을 다시 실행하여 **manifest** 및 **openshift** 하위 디렉터리의 파일 세트에서 **Ignition** 구성을 만듭니다.

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

이제 **Ignition** 구성 파일을 **vSphere** 설치 절차에 대한 입력으로 사용하여 **RHCOS (Red Hat Enterprise Linux CoreOS)** 시스템을 설치할 수 있습니다.

18.6.16. bootupd를 사용하여 부트로더 업데이트

bootupd를 사용하여 부트로더를 업데이트하려면 **RHCOS** 머신에 수동으로 **bootupd**를 설치하거나 **systemd** 유닛이 활성화된 머신 구성을 제공해야 합니다. **grubby** 또는 기타 부트로더 툴과는 달리 **bootupd**는 커널 인수 전달과 같은 커널 공간 구성을 관리하지 않습니다.

bootupd를 설치한 후 **OpenShift Container Platform** 클러스터에서 원격으로 관리할 수 있습니다.

참고

bootupd는 **BootHole** 취약점에 대한 보호와 같이 베어 메탈 또는 가상화 하이퍼바이저 설치에서만 사용하는 것이 좋습니다.

수동 설치 방법

bootctl 명령줄 툴을 사용하여 **bootupd**를 수동으로 설치할 수 있습니다.

1. 시스템 상태를 검사합니다.

```
# bootupctl status
```

출력 예

```
Component EFI
  Installed: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
  Update: At latest version
```

2. **bootupd**를 설치하지 않고 생성된 **RHCOS** 이미지에는 명시적 채택 단계가 필요합니다.

   시스템 상태가 **Adoptable**인 경우 채택을 수행합니다.

```
# bootupctl adopt-and-update
```

출력 예

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

3. 업데이트를 사용할 수 있는 경우 다음 재부팅에 변경 사항이 적용되도록 업데이트를 적용합니다.

```
# bootupctl update
```

출력 예

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

시스템 구성 방법

bootupd를 활성화하는 또 다른 방법은 머신 구성을 제공하는 것입니다.

- 다음 예와 같이 systemd 단위가 활성화된 머신 구성 파일을 제공합니다.

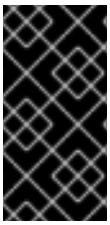
출력 예

```
variant: rhcos
version: 1.1.0
systemd:
  units:
    - name: custom-bootupd-auto.service
      enabled: true
      contents: |
        [Unit]
        Description=Bootupd automatic update

        [Service]
        ExecStart=/usr/bin/bootupctl update
        RemainAfterExit=yes

        [Install]
        WantedBy=multi-user.target
```

18.6.17. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift

CLI(oc)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 oc를 설치할 수 있습니다.



중요

이전 버전의 oc를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 oc를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 Linux에서 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.
2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. OpenShift v4.9 Linux Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. oc 바이너리를 PATH에 있는 디렉터리에 배치합니다.

   PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치

다음 절차에 따라 Windows에 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.
2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. OpenShift v4.9 Windows Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.
4. ZIP 프로그램으로 아카이브의 압축을 풉니다.
5. oc 바이너리를 PATH에 있는 디렉터리로 이동합니다.

   PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치

다음 절차에 따라 macOS에서 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.
2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. OpenShift v4.9 MacOSX Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.

4. 아카이브의 압축을 해제하고 압축을 풉니다.

5. **oc** 바이너리 **PATH**의 디렉터리로 이동합니다.

   PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

18.6.18. 부트스트랩 프로세스가 완료될 때까지 대기 중

OpenShift Container Platform 부트스트랩 프로세스는 클러스터 노드가 먼저 디스크에 설치된 영구 **RHCOS** 환경으로 부팅된 후에 시작됩니다. **Ignition** 구성 파일을 통해 제공되는 구성 정보는 부트스트랩 프로세스를 초기화하고 머신에 **OpenShift Container Platform**을 설치하는 데 사용됩니다. 부트스트랩 프로세스가 완료될 때까지 기다려야 합니다.

사전 요구 사항

- 클러스터에 대한 **Ignition** 구성 파일이 생성되어 있습니다.
- 적합한 네트워크, **DNS** 및 로드 밸런싱 인프라가 구성되어 있습니다.
- 설치 프로그램을 받아서 클러스터의 **Ignition** 구성 파일을 생성했습니다.
- 클러스터 머신에 **RHCOS**를 설치하고 **OpenShift Container Platform** 설치 프로그램에서 생성된 **Ignition** 구성 파일을 제공했습니다.
- 사용자 시스템에서 직접 인터넷에 액세스하거나 **HTTP** 또는 **HTTPS** 프록시를 사용할 수 있습니다.

프로세스

1. 부트스트랩 프로세스를 모니터링합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
  --log-level=info 2
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

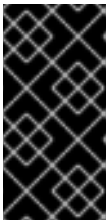
2 다른 설치 세부 사항을 보려면 info 대신 warn, debug 또는 error를 지정합니다.

출력 예

```
INFO Waiting up to 30m0s for the Kubernetes API at
https://api.test.example.com:6443...
INFO API v1.22.1 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

이 명령은 Kubernetes API 서버가 컨트롤 플레인 시스템에서 부트스트랩되었다는 신호를 보낼 때 성공합니다.

2. 부트스트랩 프로세스가 완료된 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다.



중요

이 시점에 로드 밸런서에서 부트스트랩 시스템을 제거해야 합니다. 부트스트랩 머신 자체를 제거하거나 다시 포맷할 수도 있습니다.

18.6.19. CLI를 사용하여 클러스터에 로그인

클러스터 kubeconfig 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. kubeconfig 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러

스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- oc CLI를 설치했습니다.

프로세스

1. kubeadmin 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 oc 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

18.6.20. 머신의 인증서 서명 요청 승인

클러스터에 시스템을 추가하면 추가한 시스템별로 보류 중인 인증서 서명 요청(CSR)이 두 개씩 생성됩니다. 이러한 CSR이 승인되었는지 확인해야 하며, 필요한 경우 이를 직접 승인해야 합니다. 클라이언트 요청을 먼저 승인한 다음 서버 요청을 승인해야 합니다.

사전 요구 사항

- 클러스터에 시스템을 추가했습니다.

절차

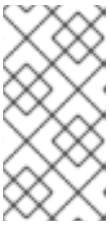
1. 클러스터가 시스템을 인식하는지 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0  Ready    master  63m  v1.22.1
master-1  Ready    master  63m  v1.22.1
master-2  Ready    master  64m  v1.22.1
```

출력에 생성된 모든 시스템이 나열됩니다.



참고

이전 출력에는 일부 **CSR**이 승인될 때까지 컴퓨팅 노드(작업자 노드라고도 함)가 포함되지 않을 수 있습니다.

2. 보류 중인 **CSR**을 검토하고 클러스터에 추가한 각 시스템에 대해 **Pending** 또는 **Approved** 상태의 클라이언트 및 서버 요청이 표시되는지 확인합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE    REQUESTOR                                                   CONDITION
csr-8b2br   15m    system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
```

```
csr-8vnps   15m    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

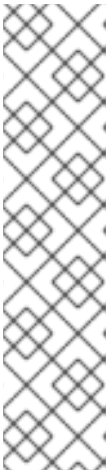
예에서는 두 시스템이 클러스터에 참여하고 있습니다. 목록에는 승인된 CSR이 더 많이 나타날 수도 있습니다.

3.

CSR이 승인되지 않은 경우, 추가된 시스템에 대한 모든 보류 중인 CSR이 Pending 상태로 전환된 후 클러스터 시스템의 CSR을 승인합니다.

참고

CSR은 교체 주기가 자동으로 만료되므로 클러스터에 시스템을 추가한 후 1 시간 이내에 CSR을 승인하십시오. 한 시간 내에 승인하지 않으면 인증서가 교체되고 각 노드에 대해 두 개 이상의 인증서가 표시됩니다. 이러한 인증서를 모두 승인해야 합니다. 클라이언트 CSR이 승인되면 Kubelet은 인증서에 대한 보조 CSR을 생성하므로 수동 승인이 필요합니다. 그러면 Kubelet에서 동일한 매개변수를 사용하여 새 인증서를 요청하는 경우 인증서 갱신 요청은 machine-approver에 의해 자동으로 승인됩니다.



참고

베어 메탈 및 기타 사용자 프로비저닝 인프라와 같이 머신 API를 사용하도록 활성화되지 않는 플랫폼에서 실행되는 클러스터의 경우 CSR(Kubelet service Certificate Request)을 자동으로 승인하는 방법을 구현해야 합니다. 요청이 승인되지 않으면 API 서버가 kubelet에 연결될 때 서비스 인증서가 필요하므로 oc exec, oc rsh, oc logs 명령을 성공적으로 수행할 수 없습니다. Kubelet 엔드 포인트에 연결하는 모든 작업을 수행하려면 이 인증서 승인이 필요합니다. 이 방법은 새 CSR을 감시하고 CSR이 system:node 또는 system:admin 그룹의 node-bootstrapper 서비스 계정에 의해 제출되었는지 확인하고 노드의 ID를 확인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1

<csr_name>은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

참고

일부 Operator는 일부 CSR이 승인될 때까지 사용할 수 없습니다.

4. 이제 클라이언트 요청이 승인되었으므로 클러스터에 추가한 각 머신의 서버 요청을 검토해야 합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 나머지 CSR이 승인되지 않고 Pending 상태인 경우 클러스터 머신의 CSR을 승인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1

<csr_name>은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs oc adm certificate approve
```

6. 모든 클라이언트 및 서버 CSR이 승인된 후 머신은 Ready 상태가 됩니다. 다음 명령을 실행하여 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0  Ready    master  73m  v1.22.1
master-1  Ready    master  73m  v1.22.1
master-2  Ready    master  74m  v1.22.1
worker-0  Ready    worker  11m  v1.22.1
worker-1  Ready    worker  11m  v1.22.1
```

참고

머신이 Ready 상태로 전환하는 데 서버 CSR의 승인 후 몇 분이 걸릴 수 있습니다.

추가 정보

- CSR에 대한 자세한 내용은 인증서 서명 요청을 참조하십시오.

18.6.21. Operator의 초기 설정

컨트롤 플레인이 초기화된 후 일부 Operator를 즉시 구성하여 모두 사용 가능하도록 해야 합니다.

사전 요구 사항

- 컨트롤 플레인이 초기화되어 있습니다.

프로세스

1. 클러스터 구성 요소가 온라인 상태인지 확인합니다.

```
$ watch -n5 oc get clusteroperators
```

출력 예

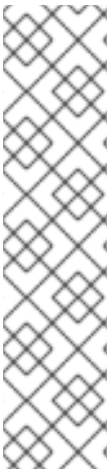
```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.9.0     True        False         False      19m
baremetal                                  4.9.0     True        False         False      37m
cloud-credential                           4.9.0     True        False         False      40m
cluster-autoscaler                         4.9.0     True        False         False      37m
config-operator                            4.9.0     True        False         False      38m
console                                    4.9.0     True        False         False      26m
csi-snapshot-controller                    4.9.0     True        False         False      37m
dns                                        4.9.0     True        False         False      37m
etcd                                       4.9.0     True        False         False      36m
image-registry                             4.9.0     True        False         False      31m
ingress                                    4.9.0     True        False         False      30m
insights                                   4.9.0     True        False         False      31m
kube-apiserver                             4.9.0     True        False         False      26m
kube-controller-manager                    4.9.0     True        False         False      36m
kube-scheduler                             4.9.0     True        False         False      36m
kube-storage-version-migrator              4.9.0     True        False         False      37m
machine-api                                4.9.0     True        False         False      29m
machine-approver                           4.9.0     True        False         False      37m
machine-config                             4.9.0     True        False         False      36m
marketplace                                4.9.0     True        False         False      37m
monitoring                                 4.9.0     True        False         False      29m
network                                    4.9.0     True        False         False      38m
node-tuning                                4.9.0     True        False         False      37m
openshift-apiserver                        4.9.0     True        False         False      32m
openshift-controller-manager               4.9.0     True        False         False      30m
openshift-samples                          4.9.0     True        False         False      32m
operator-lifecycle-manager                 4.9.0     True        False         False      37m
operator-lifecycle-manager-catalog         4.9.0     True        False         False      37m
operator-lifecycle-manager-packageserver   4.9.0     True        False         False      32m
service-ca                                 4.9.0     True        False         False      38m
storage                                    4.9.0     True        False         False      37m
```

2. 사용할 수 없는 **Operator**를 구성합니다.

18.6.21.1. 설치 중 제거된 이미지 레지스트리

공유 가능한 개체 스토리지를 제공하지 않는 플랫폼에서 **OpenShift Image Registry Operator**는 자체적으로 **Removed**로 부트스트랩합니다. 이를 통해 **openshift-installer**가 이러한 플랫폼 유형에서 설치를 완료할 수 있습니다.

설치 후 **managementState**를 **Removed**에서 **Managed**로 전환하도록 **Image Registry Operator** 구성을 편집해야 합니다.



참고

Prometheus 콘솔은 **ImageRegistryRemoved** 경고를 제공합니다. 다음은 예시 경고입니다.

"이미지 레지스트리가 제거되었습니다. **ImageStreamTags**를 참조하는 **ImageStreamTags, BuildConfigs** 및 **DeploymentConfigs**가 예상대로 작동하지 않을 수 있습니다. 스토리지를 구성하고 **configs.imageregistry.operator.openshift.io**를 편집하여 구성을 **Managed** 상태로 업데이트하십시오."

18.6.21.2. 이미지 레지스트리 스토리지 구성

기본 스토리지를 제공하지 않는 플랫폼에서는 처음에 **Image Registry Operator**를 사용할 수 없습니다. 설치한 후에 스토리지를 사용하도록 레지스트리를 구성하여 **Registry Operator**를 사용 가능하도록 만들어야 합니다.

프로덕션 클러스터에 필요한 영구 볼륨을 구성하는 과정의 지침이 표시됩니다. 해당하는 경우, 프로덕션 환경 외 클러스터에서만 사용할 수 있는 저장 위치로서 빈 디렉터리를 구성하는 과정의 지침이 표시됩니다.

업그레이드 중에 **Recreate** 롤아웃 전략을 사용하여 이미지 레지스트리의 블록 스토리지 유형 사용을 허용하기 위한 추가 지침이 제공됩니다.

18.6.21.2.1. VMware vSphere용 레지스트리 스토리지 구성

클러스터 관리자는 설치한 후 스토리지를 사용하도록 레지스트리를 구성해야 합니다.

사전 요구 사항

- 클러스터 관리자 권한이 있어야 합니다.
- VMware vSphere에 클러스터가 있어야 합니다.
- Red Hat OpenShift Container Storage와 같이 클러스터용 영구 스토리지 프로비저닝.

중요

OpenShift Container Platform은 복제본이 하나만 있는 경우 이미지 레지스트리 스토리지에 대한 ReadWriteOnce 액세스를 지원합니다. ReadWriteOnce 액세스에서는 레지스트리가 Recreate 롤아웃 전략을 사용해야 합니다. 두 개 이상의 복제본으로 고 가용성을 지원하는 이미지 레지스트리를 배포하려면 ReadWriteMany 액세스가 필요합니다.

- "100Gi" 용량이 필요합니다.

중요

테스트 결과, RHEL의 NFS 서버를 핵심 서비스용 스토리지 백엔드로 사용하는 데 문제가 있는 것으로 나타납니다. 여기에는 OpenShift Container Registry and Quay, 스토리지 모니터링을 위한 Prometheus, 로깅 스토리지를 위한 Elasticsearch가 포함됩니다. 따라서 RHEL NFS를 사용하여 핵심 서비스에서 사용하는 PV를 백업하는 것은 권장되지 않습니다.

마켓플레이스의 다른 NFS 구현에는 이러한 문제가 나타나지 않을 수 있습니다. 이러한 OpenShift Container Platform 핵심 구성 요소에 대해 완료된 테스트에 대한 자세한 내용은 개별 NFS 구현 공급업체에 문의하십시오.

프로세스

1. 스토리지를 사용하도록 레지스트리를 구성하기 위해 configs.imageregistry/cluster 리소스에서 spec.storage.pvc를 변경합니다.

참고

공유 스토리지를 사용할 때 보안 설정을 확인하여 외부에서의 액세스를 방지합니다.

2. 레지스트리 pod가 없는지 확인합니다.

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

출력 예

```
No resourses found in openshift-image-registry namespace
```

참고

출력에 레지스트리 Pod가 있는 경우 이 절차를 계속할 필요가 없습니다.

3. 레지스트리 구성을 확인합니다.

```
$ oc edit configs.imageregistry.operator.openshift.io
```

출력 예

```
storage:
  pvc:
    claim: 1
```

1 image-registry-storage 영구 볼륨 클레임 (PVC)을 자동으로 생성할 수 있도록 claim 필드를 비워 둡니다. PVC는 기본 스토리지 클래스를 기반으로 생성됩니다. 그러나 기본 스토리지 클래스에서 RADOS 블록 장치(RBD)와 같은 ReadWriteOnce(ReadWriteOnce) 볼륨을 제공할 수 있으므로 복제본이 두 개 이상될 때 문제가 발생할 수 있습니다.

4. clusteroperator 상태를 확인합니다.

```
$ oc get clusteroperator image-registry
```

출력 예

```
NAME             VERSION                             AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                                 True        False         False     6h50m
```

18.6.21.2.2. 프로덕션 환경 외 클러스터에서 이미지 레지스트리의 스토리지 구성

이미지 레지스트리 Operator에 대한 스토리지를 구성해야 합니다. 프로덕션 환경 외 클러스터의 경우, 이미지 레지스트리를 빈 디렉터리로 설정할 수 있습니다. 이렇게 하는 경우 레지스트리를 다시 시작하면 모든 이미지가 손실됩니다.

절차

- 이미지 레지스트리 스토리지를 빈 디렉터리로 설정하려면 다음을 수행하십시오.

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch
'{"spec":{"storage":{"emptyDir":{}}}}'
```

주의

프로덕션 환경 외 클러스터에 대해서만 이 옵션을 구성하십시오.

Image Registry Operator가 구성 요소를 초기화하기 전에 이 명령을 실행하면 oc patch 명령이 실패하며 다음 오류가 발생합니다.

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster"
not found
```

몇 분 후에 명령을 다시 실행하십시오.

18.6.21.2.3. VMware vSphere용 블록 레지스트리 스토리지 구성

클러스터 관리자로서 업그레이드 중에 이미지 레지스트리가 **vSphere VMDK(Virtual Machine Disk)**와 같은 블록 스토리지 유형을 사용할 수 있도록 허용하기 위해 **Recreate** 롤아웃 전략을 사용할 수 있습니다.

중요

블록 스토리지 볼륨이 지원되지만 프로덕션 클러스터에서 이미지 레지스트리와 함께 사용하는 것은 권장되지 않습니다. 레지스트리가 블록 스토리지에 구성된 설치는 레지스트리가 둘 이상의 복제본을 가질 수 없기 때문에 가용성이 높지 않습니다.

절차

1. 이미지 레지스트리 스토리지를 블록 스토리지 유형으로 설정하려면 **Recreate** 롤아웃 전략을 사용하고 복제본 **1**개 만으로 실행되도록 레지스트리를 패치합니다.

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. 블록 스토리지 장치에 **PV**를 프로비저닝하고 해당 볼륨의 **PVC**를 생성합니다. 요청된 블록 볼륨은 **RWO(ReadWriteOnce)** 액세스 모드를 사용합니다.

   a. **VMware vSphere PersistentVolumeClaim** 개체를 정의하려면 다음 내용이 포함된 **pvc.yaml** 파일을 생성합니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage 1
  namespace: openshift-image-registry 2
spec:
  accessModes:
  - ReadWriteOnce 3
  resources:
    requests:
      storage: 100Gi 4
```

1

PersistentVolumeClaim 개체를 표시하는 고유한 이름입니다.

2

PersistentVolumeClaim 오브젝트의 네임스페이스로 **openshift-image-registry**입니다.

3

영구 볼륨 클레임의 액세스 모드입니다. **ReadWriteOnce**를 사용하면 단일 노드에서 읽기 및 쓰기 권한으로 볼륨을 마운트할 수 있습니다.

4

영구 볼륨 클레임의 크기입니다.

b. 파일에서 **PersistentVolumeClaim** 오브젝트를 만듭니다.

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 올바른 **PVC**를 참조하도록 레지스트리 설정을 편집합니다.

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

출력 예

```
storage:
  pvc:
    claim: 1
```

1

사용자 지정 **PVC**를 만들면 **image-registry-storage PVC**의 기본 자동 생성을 위해 **claim** 필드를 비워둘 수 있습니다.

올바른 **PVC**를 참조하도록 레지스트리 스토리지를 구성하는 방법은 **vSphere** 용 레지스트리 구성을

참조하십시오.

18.6.22. 사용자 프로비저닝 인프라에 설치 완료

Operator 구성을 완료한 후 제공하는 인프라에 클러스터 설치를 완료할 수 있습니다.

사전 요구 사항

- 컨트롤 플레인이 초기화되어 있습니다.
- 초기 Operator 구성을 완료해야 합니다.

프로세스

1. 다음 명령을 사용하여 모든 클러스터 구성 요소가 온라인 상태인지 확인합니다.

```
$ watch -n5 oc get clusteroperators
```

출력 예

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.9.0     True        False         False      19m
baremetal                                  4.9.0     True        False         False      37m
cloud-credential                           4.9.0     True        False         False      40m
cluster-autoscaler                         4.9.0     True        False         False      37m
config-operator                            4.9.0     True        False         False      38m
console                                    4.9.0     True        False         False      26m
csi-snapshot-controller                    4.9.0     True        False         False      37m
dns                                        4.9.0     True        False         False      37m
etcd                                       4.9.0     True        False         False      36m
image-registry                             4.9.0     True        False         False      31m
ingress                                    4.9.0     True        False         False      30m
insights                                   4.9.0     True        False         False      31m
kube-apiserver                             4.9.0     True        False         False      26m
kube-controller-manager                    4.9.0     True        False         False      36m
kube-scheduler                             4.9.0     True        False         False      36m
kube-storage-version-migrator              4.9.0     True        False         False      37m
machine-api                                4.9.0     True        False         False      29m
machine-approver                           4.9.0     True        False         False      37m
machine-config                             4.9.0     True        False         False      36m
marketplace                                4.9.0     True        False         False      37m
monitoring                                 4.9.0     True        False         False      29m
```

```
network                                    4.9.0     True        False         False      38m
node-tuning                                4.9.0     True        False         False      37m
openshift-apiserver                        4.9.0     True        False         False      32m
openshift-controller-manager               4.9.0     True        False         False      30m
openshift-samples                          4.9.0     True        False         False      32m
operator-lifecycle-manager                 4.9.0     True        False         False      37m
operator-lifecycle-manager-catalog         4.9.0     True        False         False      37m
operator-lifecycle-manager-packageserver   4.9.0     True        False         False      32m
service-ca                                 4.9.0     True        False         False      38m
storage                                    4.9.0     True        False         False      37m
```

또는 다음 명령은 모든 클러스터를 사용할 수 있을 때 알립니다. 또한 인증 정보를 검색하고 표시합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

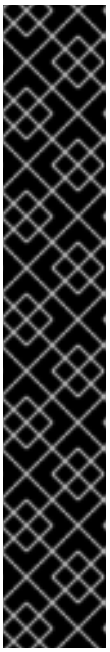
1

<installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

출력 예

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator가 Kubernetes API 서버에서 OpenShift Container Platform 클러스터 배포를 완료하면 명령이 성공합니다.



중요

- 설치 프로그램에서 생성하는 **Ignition** 구성 파일에 **24**시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 **24**시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 **kubelet** 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청(**CSR**)을 수동으로 승인해야 합니다. 자세한 내용은 ***Recovering from expired control plane certificates*** 문서를 참조하십시오.

- 클러스터를 설치한 후 **24**시간에서 **22**시간까지의 인증서가 교체되기 때문에 생성된 후 **12**시간 이내에 **Ignition** 구성 파일을 사용하는 것이 좋습니다. **12**시간 이내에 **Ignition** 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

2. **Kubernetes API** 서버가 **Pod**와 통신하고 있는지 확인합니다.

   a. 모든 **Pod** 목록을 보려면 다음 명령을 사용하십시오.

   ```
   $ oc get pods --all-namespaces
   ```

   출력 예

   ```
   NAMESPACE                  NAME                                        READY   STATUS
   RESTARTS   AGE
   openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8
   1/1    Running    1        9m
   openshift-apiserver        apiserver-67b9g                              1/1     Running    0
   3m
   openshift-apiserver        apiserver-ljcmx                              1/1     Running    0
   1m
   openshift-apiserver        apiserver-z25h4                              1/1     Running    0
   2m
   openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8
   1/1    Running    0        5m
   ...
   ```

   b. 다음 명령을 사용하여 이전 명령의 출력에 나열된 **Pod**의 로그를 표시합니다.

```
$ oc logs <pod_name> -n <namespace> 1
```

1

이전 명령의 출력에 표시된 대로 **Pod** 이름과 네임스페이스를 지정합니다.

Pod 로그가 표시되면 **Kubernetes API** 서버는 클러스터 시스템과 통신할 수 있습니다.

3. **FCP(Fibre Channel Protocol)**를 사용하는 설치에는 다중 경로를 활성화하기 위해 추가 단계가 필요합니다. 설치 중에 멀티패스를 활성화하지 마십시오.

   자세한 내용은 *설치 후 머신 구성 작업* 설명서에서 **"RHCOS에서 커널 인수를 사용하여 멀티패스 활성화"**를 참조하십시오.

vSphere에 컴퓨팅 머신 추가에 따라 클러스터 설치가 완료된 후 추가 컴퓨팅 머신을 추가할 수 있습니다.

18.6.23. VMware vSphere 볼륨 백업

OpenShift Container Platform은 새 볼륨을 독립 영구 디스크로 프로비저닝하여 클러스터의 모든 노드에서 볼륨을 자유롭게 연결 및 분리합니다. 결과적으로 스냅샷을 사용하는 볼륨을 백업하거나 스냅샷에서 볼륨을 복원할 수 없습니다. 자세한 내용은 **스냅 샷 제한** 을 참조하십시오.

프로세스

영구 볼륨의 백업을 생성하려면 다음을 수행합니다.

1. 영구 볼륨을 사용중인 애플리케이션을 중지합니다.
2. 영구 볼륨을 복제합니다.
3. 애플리케이션을 다시 시작합니다.
4. 복제된 볼륨의 백업을 만듭니다.

5. 복제된 볼륨을 삭제합니다.

18.6.24. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

18.6.25. 다음 단계

- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.
- 레지스트리를 설정하고 레지스트리 스토리지를 구성합니다.
- 선택 사항: vSphere Problem Detector Operator에서 이벤트를 보고 클러스터에 권한 또는 스토리지 구성 문제가 있는지 확인합니다.

18.7. 사용자 프로비저닝 인프라 및 네트워크 사용자 지정을 통해VMC에 클러스터 설치

OpenShift Container Platform 4.9 버전에서는 AWS의 VMC (VMware Cloud)에 배포하여 사용자 지정 네트워크 구성 옵션이 있는 프로비저닝하는 인프라를 사용하여 VMware vSphere 인스턴스에 클러스터를 설치할 수 있습니다.

OpenShift Container Platform 배포를 위해 VMC 환경을 구성한 후 VMC 환경에 공동 배치된 bastion

관리 호스트에서 **OpenShift Container Platform** 설치 프로그램을 사용합니다. 설치 프로그램 및 컨트롤 플레인은 **OpenShift Container Platform** 클러스터에 필요한 리소스 배포 및 관리 프로세스를 자동화합니다.

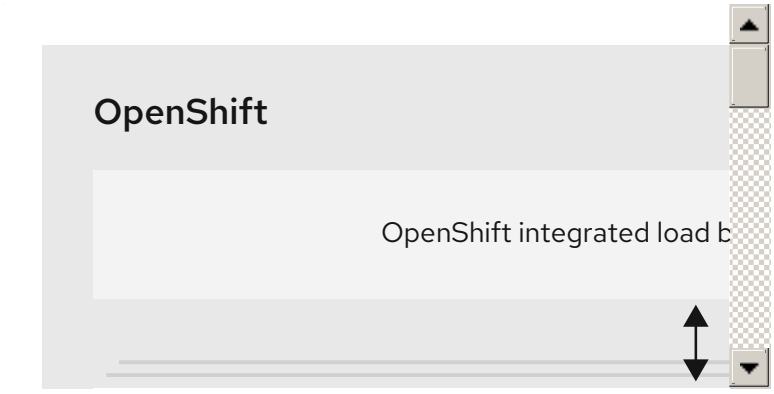
네트워크 구성을 사용자 지정할 경우, 클러스터가 사용자 환경의 기존 **IP** 주소 할당과 공존하고 기존 **VXLAN** 구성과 통합될 수 있습니다. 설치 과정에서 네트워크 구성 매개변수를 대부분 설정해야 하며, 실행 중인 클러스터에서는 **kubeProxy** 구성 매개변수만 수정할 수 있습니다.

참고

OpenShift Container Platform은 단일 **VMware vCenter**에만 클러스터 배포를 지원합니다. 여러 **vCenter**에 머신/머신 세트가 있는 클러스터 배포는 지원되지 않습니다.

18.7.1. vSphere용 VMC 설정

AWS 호스팅된 **vSphere** 클러스터에 **VMware Cloud (VMC)**의 **OpenShift Container Platform**을 설치하여 하이브리드 클라우드 전반에 애플리케이션을 온프레미스 및 오프프레미스 모두 배포 및 관리할 수 있습니다.



VMware vSphere에 **OpenShift Container Platform**을 설치하기 전에 **VMC** 환경에서 여러 옵션을 구성해야 합니다. **VMC** 환경에 다음과 같은 사전 요구 사항이 있는지 확인합니다.

- 비독점 **DHCP** 사용, **NSX-T** 네트워크 세그먼트 및 서브넷을 생성합니다. 서브넷에서 다른 **VM**(가상 머신)을 호스팅할 수 있지만 **OpenShift Container Platform** 배포에서는 **8**개 이상의 **IP** 주소를 사용할 수 있어야 합니다.
- 다음 방화벽 규칙을 설정합니다.
  - **OpenShift Container Platform** 컴퓨팅 네트워크와 인터넷 간의 **ANY:ANY** 방화벽 규칙입니다. 이는 노드와 애플리케이션에서 컨테이너 이미지를 다운로드하는 데 사용됩니다.

- 포트 443의 설치 호스트와 소프트웨어 정의 데이터 센터(SDDC) 관리 네트워크 간의 ANY:ANY 방화벽 규칙입니다. 이를 통해 배포 중에 RHCOS(Red Hat Enterprise Linux CoreOS) OVA를 업로드할 수 있습니다.

- OpenShift Container Platform 컴퓨팅 네트워크와 인터넷 간의 HTTPS 방화벽 규칙입니다. 이 연결을 통해 OpenShift Container Platform은 노드, PVC(영구 볼륨 클레임) 및 기타 리소스를 프로비저닝 및 관리하기 위해 vCenter과 통신할 수 있습니다.

- OpenShift Container Platform을 배포하려면 다음 정보가 있어야 합니다.

  - OpenShift Container Platform 클러스터에 vmc-prod-1로 로그인합니다.

  - companyname.com과 같은 기본 DNS 이름입니다.

  - 기본값을 사용하지 않는 경우 Pod 네트워크 CIDR 및 서비스 네트워크 CIDR을 식별해야 하며, 기본적으로 10.128.0.0/14 및 172.30.0.0/16으로 설정됩니다. 이러한 CIDR은 pod-to-pod 및 pod-to-service 통신에 사용되며 외부에서 액세스할 수 없지만 조직의 기존 서브넷과 중복되어서는 안 됩니다.

  - 다음 vCenter 정보를 참조하십시오:

    - vCenter 호스트 이름, 사용자 이름 및 암호

    - 데이터 센터 이름 (예: SDDC-Datacenter)

    - Cluster-1과 같은 클러스터 이름

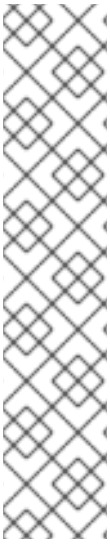
    - 네트워크 이름

    - WorkloadDatastore와 같은 데이터 저장소 이름

참고

클러스터 설치가 완료된 후 **vSphere** 클러스터를 **VMC Compute-ResourcePool** 리소스 풀로 이동하는 것이 좋습니다.

- **VMC**에 **bastion**으로 배포된 **Linux** 기반 호스트입니다.
  - **bastion** 호스트는 **RHEL(Red Hat Enterprise Linux)** 또는 기타 **Linux** 기반 호스트일 수 있습니다. 인터넷에 연결되어 있어야 하고 **OVA**를 **ESXi** 호스트에 업로드하는 기능이 있어야 합니다.
  - **OpenShift CLI** 툴을 **bastion** 호스트에 다운로드하여 설치합니다.
    - **openshift-install** 설치 프로그램
    - **OpenShift CLI(oc)** 툴



참고

Kubernetes 용 **VMware NCP (NSX Conrainer Plugin)dOpen**에는 사용할 수 없으며 **NSX**는 **OpenShift SDN**으로 사용되지 않습니다. 현재 **VMC**에서 사용할 수 있는 **NSX** 버전은 **OpenShift Container Platform**에서 인증된 **NCP** 버전과 호환되지 않습니다.

그러나 **NSX DHCP** 서비스는 전체 스택 자동화된 **OpenShift Container Platform** 배포와 **vSphere**와의 **Machine API** 통합을 통해 수동 또는 자동으로 프로비저닝된 노드를 사용하여 가상 머신 **IP** 관리에 사용됩니다. 또한 **OpenShift Container Platform** 클러스터와 **bastion** 호스트와 **VMC vSphere** 호스트 사이에서 액세스할 수 있도록 **NSX** 방화벽 규칙이 생성됩니다.

18.7.1.1. VMC Sizer 툴

AWS의 **VMware Cloud**는 **AWS** 베어메탈 인프라 상단에 구축됩니다. 이는 **AWS** 네이티브 서비스를 실행하는 것과 동일한 베어 메탈 인프라입니다. **AWS** 소프트웨어 정의 데이터 센터(**SDDC**)의 **VMware** 클라우드가 배포되면 이러한 물리적 서버 노드를 사용하고 단일 테넌트 방식으로 **VMware ESXi** 하이퍼바이저를 실행할 수 있습니다. 즉, **VMC**를 사용하는 모든 사용자가 물리적 인프라에 액세스할 수 없습니다. 가상 인프라를 호스팅하는데 필요한 물리적 호스트 수를 고려해야 합니다.

이를 확인하기 위해 VMware는 AWS Sizer에서 VMC를 제공합니다. 이 툴을 사용하면 VMC에서 호스팅할 리소스를 정의할 수 있습니다.

- 워크로드 유형
- 총 가상 머신 수
- 사양 정보는 다음과 같습니다:
  - 스토리지 요구사항
  - vCPU
  - vRAM
  - 오버커밋 비율

이러한 세부 정보를 사용하여 VMware 모범 사례를 기반으로 보고서를 생성하고 클러스터 구성과 필요한 호스트 수를 권장합니다.

18.7.2. vSphere 사전 요구 사항

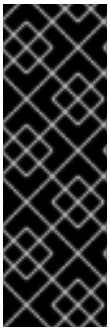
- OpenShift Container Platform 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.
- 블록 레지스트리 스토리지가 프로비저닝되어 있습니다. 자세한 내용은 영구 저장 장치 이해를 참조하십시오.
- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성해야 합니다.

18.7.3. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 **Telemetry** 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.



중요

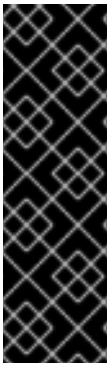
클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

18.7.4. VMware vSphere 인프라 요구사항

사용하는 구성 요소의 요구사항을 충족하는 **VMware vSphere** 버전 **6** 또는 **7** 인스턴스에 **OpenShift Container Platform** 클러스터를 설치해야 합니다.

표 **18.58. vSphere** 가상 환경의 버전 요구 사항

가상 환경 제품	필요한 버전
VM 하드웨어 버전	13 이상
vSphere ESXi hosts	6.5 이상
vCenter 호스트	6.5 이상



중요

이제 VMware vSphere 버전 6.7U2 또는 이전 버전에 클러스터를 설치하는 것이 더 이상 사용되지 않으며 가상 하드웨어 버전 13은 더 이상 사용되지 않습니다. 이러한 버전은 여전히 완전히 지원되지만 향후 OpenShift Container Platform 버전에서는 지원이 제거됩니다. 이제 OpenShift Container Platform의 vSphere 가상 머신의 하드웨어 버전 15가 기본값이 되었습니다. vSphere 노드의 하드웨어 버전을 업데이트하려면 " vSphere에서 실행되는 노드에서 하드웨어 업데이트" 문서를 참조하십시오.

표 18.59. VMware 구성 요소에 지원되는 최소 vSphere 버전

구성 요소	지원되는 최소 버전	설명
하이퍼바이저	HW 버전 13이 포함된 vSphere 6.5 이상	이 버전은 RHCOS(Red Hat Enterprise Linux CoreOS)가 지원하는 최소 버전입니다. Red Hat Enterprise Linux 8 지원 하이퍼바이저 목록을 참조하십시오.
In-tree 드라이버가 있는 스토리지	vSphere 6.5 이상	이 플러그인은 OpenShift Container Platform에 포함된 vSphere용 인트리 스토리지 드라이버를 사용하여 vSphere 스토리지를 생성합니다.

vSphere 버전 6.5 인스턴스를 사용하는 경우 OpenShift Container Platform을 설치하기 전에 6.7U3 또는 7.0로 업그레이드하는 것이 좋습니다.

중요

OpenShift Container Platform을 설치하기 전에 ESXi 호스트의 시간이 동기화되었는지 확인해야 합니다. VMware 문서에서 Edit Time Configuration for a Host를 참조하십시오.

추가 리소스

- vSphere 노드의 하드웨어 버전을 업데이트하려면 vSphere에서 실행 중인 노드에서 하드웨어 업데이트를 참조하십시오.

18.7.5. 사용자 프로비저닝 인프라를 포함한 클러스터의 시스템 요구사항

사용자 프로비저닝 인프라가 포함된 클러스터의 경우, 필요한 모든 시스템을 배포해야 합니다.

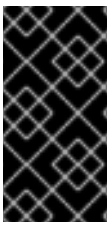
이 섹션에서는 사용자 프로비저닝 인프라에 OpenShift Container Platform을 배포해야 하는 요구 사항에 대해 설명합니다.

18.7.5.1. 클러스터 설치에 필요한 시스템

최소 OpenShift Container Platform 클러스터에 다음과 같은 호스트가 필요합니다.

표 18.60. 최소 필수 호스트

호스트	설명
임시 부트스트랩 시스템 한 개	컨트롤 플레인 시스템 세 개에 OpenShift Container Platform 클러스터를 배포하기 위한 부트스트랩 시스템이 클러스터에 필요합니다. 클러스터를 설치한 후 부트스트랩 시스템을 제거할 수 있습니다.
컨트롤 플레인 시스템 세 개	컨트롤 플레인 시스템은 컨트롤 플레인을 구성하는 Kubernetes 및 OpenShift Container Platform 서비스를 실행합니다.
두 개 이상의 컴퓨팅 시스템(작업자 시스템이라고도 함).	OpenShift Container Platform 사용자가 요청한 워크로드는 컴퓨팅 머신에서 실행됩니다.



중요

클러스터의 고가용성을 유지하려면 이러한 클러스터 시스템에 대해 별도의 물리적 호스트를 사용하십시오.

부트스트랩, 컨트롤 플레인 시스템은 운영 체제로 RHCOS (Red Hat Enterprise Linux CoreOS)를 사용해야 합니다. 그러나 컴퓨팅 머신은 RHCOS(Red Hat Enterprise Linux CoreOS), RHEL (Red Hat Enterprise Linux) 7.9 또는 RHEL 8.4 중에서 선택할 수 있습니다.

RHCOS는 RHEL 8(Red Hat Enterprise Linux)을 기반으로하며 모든 하드웨어 인증 및 요구사항을 모두 이어받습니다. Red Hat Enterprise Linux 기술 기능 및 제한을 참조하십시오.

18.7.5.2. 클러스터 설치를 위한 최소 리소스 요구 사항

각 클러스터 시스템이 다음과 같은 최소 요구사항을 충족해야 합니다.

표 18.61. 최소 리소스 요구사항

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
부트스트랩	RHCOS	4	16GB	100GB	300
컨트롤 플레인	RHCOS	4	16GB	100GB	300
컴퓨팅	RHCOS, RHEL 7.9 또는 RHEL 8.4 [3]	2	8GB	100GB	300

1. **SMT(동시 멀티스레딩) 또는 하이퍼 스레딩이 활성화되지 않은 경우 하나의 vCPU는 하나의 물리적 코어와 동일합니다. 활성화하면 다음과 같은 공식을 사용하여 해당 비율을 계산합니다. (코어 당 스레드 수 × 코어 수) × 소켓 수 = vCPU 수**

2. **OpenShift Container Platform 및 Kubernetes는 디스크 성능에 민감하며 특히 10ms p99 fsync 기간이 필요한 컨트롤 플레인 노드의 etcd에 더 빠른 스토리지가 권장됩니다. 많은 클라우드 플랫폼에서 스토리지 크기와 IOPS를 함께 확장되므로 충분한 성능을 얻으려면 스토리지 볼륨을 과도하게 할당해야 할 수 있습니다.**

3. **사용자가 프로비저닝한 모든 설치와 마찬가지로 클러스터에서 RHEL 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. RHEL 7 컴퓨팅 머신 사용은 더 이상 사용되지 않으며 향후 OpenShift Container Platform 4 릴리스에서 제거될 예정입니다.**

18.7.5.3. 인증서 서명 요청 관리

사용자가 프로비저닝하는 인프라를 사용하는 경우 자동 시스템 관리 기능으로 인해 클러스터의 액세스가 제한되므로 설치한 후 클러스터 인증서 서명 요청(CSR)을 승인하는 메커니즘을 제공해야 합니다. kube-controller-manager는 kubelet 클라이언트 CSR만 승인합니다. machine-approver는 올바른 시스템에서 발행한 요청인지 확인할 수 없기 때문에 kubelet 자격 증명을 사용하여 요청하는 서비스 인증서의 유효성을 보장할 수 없습니다. kubelet 서빙 인증서 요청의 유효성을 확인하고 요청을 승인하는 방법을 결정하여 구현해야 합니다.

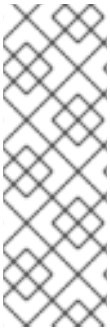
18.7.5.4. 사용자 프로비저닝 인프라에 대한 네트워킹 요구사항

모든 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템이 부팅 중에 Ignition 구성 파일을 가져오려면 initramfs에 네트워킹을 구성해야 합니다.

초기 부팅 과정에서 시스템에 필요한 부팅 옵션을 제공하여 DHCP 서버를 통해 설정하거나 정적으로 설정하는 IP 주소 구성이 필요합니다. 네트워크 연결이 설정된 후 시스템은 HTTP 또는 HTTPS 서버에서 Ignition 구성 파일을 다운로드합니다. 그런 다음 Ignition 구성 파일을 사용하여 각 머신의 정확한 상태를

설정합니다. **Machine Config Operator**는 설치 후 새 인증서 또는 키 적용과 같은 머신에 대한 추가 변경을 완료합니다.

클러스터 시스템의 장기적인 관리를 위해 **DHCP** 서버를 사용하는 것이 좋습니다. **DHCP** 서버가 클러스터 시스템에 영구 **IP** 주소, **DNS** 서버 정보 및 호스트 이름을 제공하도록 구성되었는지 확인합니다.



참고

사용자 프로비저닝 인프라에 **DHCP** 서비스를 사용할 수 없는 경우 **RHCOS** 설치 시 노드에 **IP** 네트워킹 구성과 **DNS** 서버의 주소를 대신 제공할 수 있습니다. **ISO** 이미지에서 설치하는 경우 부팅 인수로 전달할 수 있습니다. 고정 **IP** 프로비저닝 *및 고급 네트워킹 옵션에 대한 자세한 내용은 **RHCOS** 설치 및 **OpenShift Container Platform** 부트스트랩 프로세스 시작* 섹션을 참조하십시오.

Kubernetes API 서버가 클러스터 시스템의 노드 이름을 확인할 수 있어야 합니다. **API** 서버와 작업자 노드가 서로 다른 영역에 있는 경우, **API** 서버가 노드 이름을 확인할 수 있도록 기본 **DNS** 검색 영역을 설정할 수 있습니다. 노드 개체와 모든 **DNS** 요청에서 항상 정규화된 도메인 이름으로 호스트를 가리키는 것도 지원되는 방법입니다

18.7.5.4.1. DHCP를 통해 클러스터 노드의 호스트 이름 설정

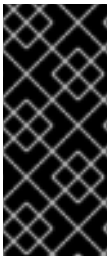
RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 호스트 이름은 **NetworkManager**를 통해 설정됩니다. 기본적으로 시스템은 **DHCP**를 통해 호스트 이름을 가져옵니다. **DHCP**에서 호스트 이름을 제공하지 않거나, 커널 인수를 통해 정적으로 설정하거나 다른 방법을 통해 설정하면 역방향 **DNS** 조회를 통해 얻을 수 있습니다. 역방향 **DNS** 조회는 노드에서 네트워크를 초기화한 후 수행되며 확인하는 데 시간이 걸릴 수 있습니다. 다른 시스템 서비스는 이 보다 먼저 시작하여 호스트 이름을 **localhost** 등으로 감지할 수 있습니다. **DHCP**를 사용하여 각 클러스터 노드의 호스트 이름을 제공하여 이 문제를 방지할 수 있습니다.

또한 **DHCP**를 통해 호스트 이름을 설정하면 **DNS** 분할 수평 구현 환경에서 수동으로 **DNS** 레코드 이름 구성 오류를 무시할 수 있습니다.

18.7.5.4.2. 네트워크 연결 요구사항

OpenShift Container Platform 클러스터 구성 요소가 통신할 수 있도록 시스템 간 네트워크 연결을 구성해야 합니다. 각 시스템에서 클러스터에 있는 다른 모든 시스템의 호스트 이름을 확인할 수 있어야 합니다.

이 섹션에서는 필요한 포트에 대해 자세히 설명합니다.



중요

연결된 OpenShift Container Platform 환경에서 모든 노드는 플랫폼 컨테이너의 이미지를 가져오고 Red Hat에 원격 측정 데이터를 제공하기 위해 인터넷에 액세스할 수 있어야 합니다.

표 18.62. 모든 시스템 간 통신에 사용되는 포트

프로토콜	포트	설명
ICMP	해당 없음	네트워크 연결성 테스트
TCP	**1936**	메트릭
	9000-9999	**9100-9101** 포트의 노드 내보내기 및 **9099** 포트의 Cluster Version Operator를 포함한 호스트 수준 서비스.
	10250-10259	Kubernetes에서 예약하는 기본 포트
	10256	openshift-sdn
UDP	**4789**	VXLAN 및 Geneve
	6081	VXLAN 및 Geneve
	9000-9999	**9100-9101** 포트의 노드 내보내기를 포함한 호스트 수준 서비스.
	500	IPsec IKE 패킷
	4500	IPsec NAT-T 패킷
TCP/UDP	**30000-32767**	Kubernetes 노드 포트
ESP	해당 없음	IPsec Encapsulating Security Payload (ESP)

표 18.63. 모든 시스템과 컨트롤 플레인 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**6443**	Kubernetes API

표 18.64. 컨트롤 플레인 머신 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**2379-2380**	etcd 서버 및 피어 포트

사용자 프로비저닝 인프라에 대한 NTP 구성

OpenShift Container Platform 클러스터는 기본적으로 공용 **NTP(Network Time Protocol)** 서버를 사용하도록 구성되어 있습니다. 로컬 엔터프라이즈 **NTP** 서버를 사용하거나 클러스터가 연결이 끊긴 네트워크에 배포되는 경우 특정 시간 서버를 사용하도록 클러스터를 구성할 수 있습니다. 자세한 내용은 ***chrony** 타임 서비스 설정* 문서를 참조하십시오.

DHCP 서버가 **NTP** 서버 정보를 제공하는 경우 **RHCOS(Red Hat Enterprise Linux CoreOS)** 시스템의 **chrony** 타임 서비스에서 정보를 읽고 **NTP** 서버와 클럭을 동기화할 수 있습니다.

18.7.5.5. 사용자 프로비저닝 DNS 요구사항

OpenShift Container Platform 배포의 경우 다음 구성 요소에 **DNS** 이름을 확인해야 합니다.

- **Kubernetes API**
- **OpenShift Container Platform** 애플리케이션 와일드카드
- 부트스트랩, 컨트롤 플레인 및 컴퓨팅 시스템

Kubernetes API, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템에 대한 역방향 **DNS** 확인이 필요합니다.

DNS A/AAAA 또는 **CNAME** 레코드는 이름 확인에 사용되며 **PTR** 레코드는 역방향 이름 확인에 사용됩니다. **RHCOS (Red Hat Enterprise Linux CoreOS)**는 **DHCP**에서 호스트 이름을 제공하지 않는 한 모든 노드의 호스트 이름을 설정할 때 역방향 레코드를 사용하기 때문에 역방향 레코드가 중요합니다. 또한 역방향 레코드는 **OpenShift Container Platform**이 작동하는 데 필요한 인증서 서명 요청 **(CSR)**을 생성하는 데 사용됩니다.

참고

DHCP 서버를 사용하여 각 클러스터 노드에 호스트 이름을 제공하는 것이 좋습니다. 자세한 내용은 *사용자 프로비저닝 인프라 섹션에 대한 DHCP 권장 사항* 섹션을 참조하십시오.

사용자가 프로비저닝한 OpenShift Container Platform 클러스터에 대해 다음 DNS 레코드가 필요하며 설치 전에 있어야 합니다. 각 레코드에서 <cluster_name>은 클러스터 이름이고 <base_domain>은 install-config.yaml 파일에서 지정하는 기반 도메인입니다. 전체 DNS 레코드는 <component>.<cluster_name>.<base_domain> 형식입니다.

표 18.65. 필수 DNS 레코드

구성 요소	레코드	설명
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API 로드 밸런서를 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.
	api-int.<cluster_name>.<base_domain>.	내부적으로 API 로드 밸런서를 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.  중요 API 서버는 Kubernetes에 기록된 호스트 이름으로 작업자 노드를 확인할 수 있어야 합니다. API 서버가 노드 이름을 확인할 수 없는 경우 프록시된 API 호출이 실패할 수 있으며 pod에서 로그를 검색할 수 없습니다.
라우트	***.apps.<cluster_name>.<base_domain>.**	애플리케이션 인그레스 로드 밸런서를 참조하는 와일드카드 DNS A/AAA 또는 CNAME 레코드입니다. 애플리케이션 인그레스 로드 밸런서는 Ingress 컨트롤러 Pod를 실행하는 머신을 대상으로 합니다. Ingress 컨트롤러 Pod는 기본적으로 컴퓨팅 머신에서 실행됩니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다. 예를 들어 **console-openshift-console.apps.<cluster_name>.<base_domain>**은 OpenShift Container Platform 콘솔의 와일드카드 경로로 사용됩니다.
부트스트랩 시스템	**bootstrap.<cluster_name>.<base_domain>.**	부트스트랩 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.

구성 요소	레코드	설명
컨트롤 플레인 머신	**<master><n>. <cluster_name>. <base_domain>.**	컨트롤 플레인 노드의 각 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.
컴퓨팅 머신	**<worker><n>. <cluster_name>. <base_domain>.**	작업자 노드의 각 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.

참고

OpenShift Container Platform 4.4 이상에서는 DNS 구성에서 etcd 호스트 및 SRV 레코드를 지정할 필요가 없습니다.

작은 정보

dig 명령을 사용하여 이름과 역방향 이름을 확인할 수 있습니다. 자세한 검증 단계는 *사용자 프로비저닝 인프라의 DNS 확인* 섹션을 참조하십시오.

18.7.5.5.1. 사용자 프로비저닝 클러스터의 DNS 구성 예

이 섹션에서는 사용자 프로비저닝 인프라에 OpenShift Container Platform을 배포하기 위한 DNS 요구 사항을 충족하는 A 및 PTR 레코드 구성 샘플을 제공합니다. 샘플은 하나의 DNS 솔루션을 선택하기 위한 조언을 제공하기 위한 것이 아닙니다.

이 예제에서 클러스터 이름은 ocp4이고 기본 도메인은 example.com입니다.

사용자 프로비저닝 클러스터의 DNS A 레코드 구성 예

다음 BIND 영역 파일의 예제에서는 사용자가 프로비저닝한 클러스터의 이름 확인을 위한 샘플 A 레코드를 보여줍니다.

예 18.16. 샘플 DNS 영역 데이터베이스

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
```

```
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
master0.ocp4.example.com. IN A 192.168.1.97 5
master1.ocp4.example.com. IN A 192.168.1.98 6
master2.ocp4.example.com. IN A 192.168.1.99 7
;
worker0.ocp4.example.com. IN A 192.168.1.11 8
worker1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1

Kubernetes API의 이름 확인을 제공합니다. 레코드는 **API** 로드 밸런서의 **IP** 주소를 나타냅니다.

2

Kubernetes API의 이름 확인을 제공합니다. 레코드는 **API** 로드 밸런서의 **IP** 주소를 참조하며 내부 클러스터 통신에 사용됩니다.

3

와일드카드 경로의 이름 확인을 제공합니다. 레코드는 애플리케이션 인그레스 로드 밸런서의 **IP** 주소를 나타냅니다. 애플리케이션 인그레스 로드 밸런서는 **Ingress** 컨트롤러 **Pod**를 실행하는 머신을 대상으로 합니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

참고

이 예제에서는 **Kubernetes API** 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 **API** 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

4

부트스트랩 시스템의 이름 확인을 제공합니다.

5 6 7

컨트롤 플레인 시스템의 이름 확인을 제공합니다.

8 9

컴퓨팅 시스템의 이름 확인을 제공합니다.

사용자 프로비저닝 클러스터의 **DNS PTR** 레코드 구성 예

다음 예제 **BIND** 영역 파일은 사용자 프로비저닝 클러스터의 역방향 이름 확인을 위한 샘플 **PTR** 레코드를 보여줍니다.

예 18.17. 역방향 레코드의 샘플 DNS 영역 데이터베이스

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR master0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR master1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR master2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 7
```

```
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com. 8
;
;EOF
```

1

Kubernetes API의 역방향 DNS 확인을 제공합니다. PTR 레코드는 API 로드 밸런서의 레코드 이름을 참조합니다.

2

Kubernetes API의 역방향 DNS 확인을 제공합니다. PTR 레코드는 API 로드 밸런서의 레코드 이름을 참조하며 내부 클러스터 통신에 사용됩니다.

3

부트스트랩 시스템의 역방향 DNS 확인을 제공합니다.

4 5 6

컨트롤 플레인 시스템의 역방향 DNS 확인을 제공합니다.

7 8

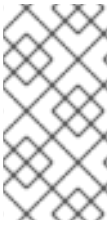
컴퓨팅 시스템의 역방향 DNS 확인을 제공합니다.

참고

OpenShift Container Platform 애플리케이션 와일드카드에는 PTR 레코드가 필요하지 않습니다.

18.7.5.6. 사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구사항

OpenShift Container Platform을 설치하기 전에 API 및 애플리케이션 인그레스 로드 밸런싱 인프라를 프로비저닝해야 합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 API 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.



참고

RHEL(Red Hat Enterprise Linux) 인스턴스를 사용하여 API 및 애플리케이션 인그레스 로드 밸런서를 배포하려면 RHEL 서브스크립션을 별도로 구매해야 합니다.

로드 밸런서 인프라는 다음 요구 사항을 충족해야 합니다.

1. **API 로드 밸런서:** 플랫폼과 상호 작용하고 플랫폼을 구성할 수 있도록 사용자(인간과 시스템 모두)에게 공통 끝점을 제공합니다. 다음 조건을 설정합니다.

   - **Layer 4** 로드 밸런싱 전용입니다. 이를 **Raw TCP, SSL Passthrough** 또는 **SSL Bridge** 모드라고 합니다. **SSL Bridge** 모드를 사용하는 경우, **API** 경로에 대해 **SNI(Server Name Indication,** 서버 이름 표시)를 활성화해야 합니다.

   - 스테이트리스 로드 밸런싱 알고리즘입니다. 옵션은 로드 밸런서 구현에 따라 달라집니다.

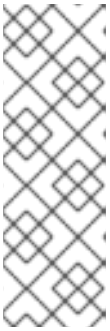
   참고

   API 로드 밸런서가 제대로 작동하기 위해 세션 지속성이 필요하지 않습니다.

   로드 밸런서의 전면과 후면 모두에서 다음 포트를 구성하십시오.

표 18.66. API 로드 밸런서

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
6443	부트스트랩 및 컨트롤 플레인. 부트스트랩 시스템이 클러스터 컨트롤 플레인을 초기화한 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다. API 서버 상태 검사 프로브에 대한 **/readyz** 끝점을 구성해야 합니다.	X	X	Kubernetes API 서버
22623	부트스트랩 및 컨트롤 플레인. 부트스트랩 시스템이 클러스터 컨트롤 플레인을 초기화한 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다.	X		시스템 구성 서버



참고

API 서버가 /readyz 엔드포인트를 해제하는 시점부터 풀에서 API 서버 인스턴스가 제거되는 시점까지 시간이 30초를 넘지 않도록 로드 밸런서를 구성해야 합니다. /readyz가 오류를 반환하거나 정상 상태가 된 후 정해진 시간 안에 끝점이 제거 또는 추가되어야 합니다. 5초 또는 10초의 프로빙 주기(두 번의 성공적인 요청은 정상 상태, 세 번의 요청은 비정상 상태)는 충분한 테스트를 거친 값입니다.

2.

애플리케이션 인그레스 로드 밸런서: 클러스터 외부에서 유입되는 애플리케이션 트래픽에 대한 인그래스 포인트를 제공합니다. 다음 조건을 설정합니다.

- Layer 4 로드 밸런싱 전용입니다. 이를 Raw TCP, SSL Passthrough 또는 SSL Bridge 모드라고 합니다. SSL Bridge 모드를 사용하는 경우 인그레스 경로에 대해 SNI(Server Name Indication, 서버 이름 표시)를 활성화해야 합니다.

- 사용 가능한 옵션과 플랫폼에서 호스팅되는 애플리케이션 유형에 따라 연결 기반 또는 세션 기반 지속성이 권장됩니다.

작은 정보

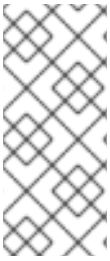
애플리케이션 인그레스 로드 밸런서에서 클라이언트의 실제 IP 주소를 확인할 수 있는 경우 소스 IP 기반 세션 지속성을 활성화하면 엔드 투 엔드 TLS 암호화를 사용하는 애플리케이션의 성능을 향상시킬 수 있습니다.

로드 밸런서의 전면과 후면 모두에서 다음 포트를 구성하십시오.

표 18.67. 애플리케이션 인그레스 로드 밸런서

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
443	기본적으로 인그레스 컨트롤러 pod, 컴퓨팅 또는 작업자를 실행하는 시스템입니다.	X	X	HTTPS 트래픽
80	기본적으로 인그레스 컨트롤러 pod, 컴퓨팅 또는 작업자를 실행하는 시스템입니다.	X	X	HTTP 트래픽

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
1936	기본적으로 Ingress 컨트롤러 Pod를 실행하는 작업자 노드입니다. 수신 상태 점검 프로브에 대해 **/healthz/ready** 끝점을 구성해야 합니다.	X	X	HTTP 트래픽



참고

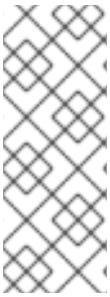
컴퓨팅 노드가 0인 3-노드 클러스터를 배포하는 경우 Ingress 컨트롤러 Pod는 컨트롤 플레인 노드에서 실행됩니다. 3-노드 클러스터 배포에서 HTTP 및 HTTPS 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다.

참고

인그레스 라우터에 대한 작업 구성이 OpenShift Container Platform 클러스터에 필요합니다. 컨트롤 플레인 초기화 후 인그레스 라우터를 설정해야 합니다.

18.7.5.6.1. 사용자 프로비저닝 클러스터의 로드 밸런서 구성 예

이 섹션에서는 사용자 프로비저닝 클러스터의 로드 밸런싱 요구 사항을 충족하는 API 및 애플리케이션 수신 로드 밸런서 구성 예를 제공합니다. 샘플은 HAProxy 로드 밸런서에 대한 **/etc/haproxy/haproxy.cfg** 구성입니다. 이 예제에서는 하나의 로드 밸런싱 솔루션을 선택하기 위한 조언을 제공하는 것을 목적으로 하지 않습니다.



참고

이 예제에서는 Kubernetes API 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 API 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

예 18.18. API 및 애플리케이션 인그레스 로드 밸런서 구성 샘플

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
```

```
    option                  redispatch
    retries                 3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s
    maxconn                 3000
  frontend stats
    bind *:1936
    mode            http
    log             global
    maxconn 10
    stats enable
    stats hide-version
    stats refresh 30s
    stats show-node
    stats show-desc Stats for ocp4 cluster (1)
    stats auth admin:ocp4
    stats uri /stats
  listen api-server-6443 (2)
    bind *:6443
    mode tcp
    server bootstrap bootstrap.ocp4.example.com:6443 check inter 1s backup (3)
    server master0 master0.ocp4.example.com:6443 check inter 1s
    server master1 master1.ocp4.example.com:6443 check inter 1s
    server master2 master2.ocp4.example.com:6443 check inter 1s
  listen machine-config-server-22623 (4)
    bind *:22623
    mode tcp
    server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup (5)
    server master0 master0.ocp4.example.com:22623 check inter 1s
    server master1 master1.ocp4.example.com:22623 check inter 1s
    server master2 master2.ocp4.example.com:22623 check inter 1s
  listen ingress-router-443 (6)
    bind *:443
    mode tcp
    balance source
    server worker0 worker0.ocp4.example.com:443 check inter 1s
    server worker1 worker1.ocp4.example.com:443 check inter 1s
  listen ingress-router-80 (7)
    bind *:80
    mode tcp
    balance source
    server worker0 worker0.ocp4.example.com:80 check inter 1s
    server worker1 worker1.ocp4.example.com:80 check inter 1s
```

(1) 이 예에서 클러스터 이름은 **ocp4** 입니다.

(2)

3 5

부트스트랩 항목은 **OpenShift Container Platform** 클러스터 설치 전에 있어야 하며 부트스트랩 프로세스가 완료된 후 제거해야 합니다.

4

포트 **22623**은 머신 구성 서버 트래픽을 처리하고 컨트롤 플레인 시스템을 가리킵니다.

6

포트 **443**은 **HTTPS** 트래픽을 처리하고 **Ingress** 컨트롤러 **Pod**를 실행하는 시스템을 가리킵니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

7

포트 **80**은 **HTTP** 트래픽을 처리하고 **Ingress** 컨트롤러 **Pod**를 실행하는 머신을 가리킵니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

참고

컴퓨팅 노드가 **0**인 **3-**노드 클러스터를 배포하는 경우 **Ingress** 컨트롤러 **Pod**는 컨트롤 플레인 노드에서 실행됩니다. **3-**노드 클러스터 배포에서 **HTTP** 및 **HTTPS** 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다.

작은 정보

HAProxy를 로드 밸런서로 사용하는 경우 **HAProxy** 노드에서 **netstat -nltupe**를 실행하여 **haproxy** 프로세스가 포트 **6443, 22623, 443** 및 **80**에서 수신 대기 중인지 확인할 수 있습니다.

참고

HAProxy를 로드 밸런서로 사용하고 **SELinux**가 **enforcing**으로 설정된 경우 **HAProxy** 서비스가 **setsebool -P haproxy_connect_any=1**을 실행하여 구성된 **TCP** 포트에 바인딩할 수 있는지 확인해야 합니다.

18.7.6. 사용자 프로비저닝 인프라 준비

사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치하기 전에 기본 인프라를 준비해야 합니다.

이 섹션에서는 OpenShift Container Platform 설치를 준비하기 위해 클러스터 인프라를 설정하는 데 필요한 높은 수준의 단계에 대해 자세히 설명합니다. 여기에는 클러스터 노드에 대한 IP 네트워킹 및 네트워크 연결 구성, 방화벽을 통해 필요한 포트 활성화, 필수 DNS 및 로드 밸런싱 인프라를 설정하는 작업이 포함됩니다.

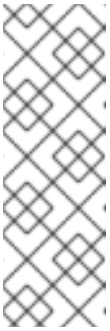
준비 후 클러스터 인프라는 *사용자 프로비저닝 인프라가 있는 클러스터의 요구 사항* 섹션에 설명된 요구 사항을 충족해야 합니다.

사전 요구 사항

- OpenShift Container Platform 4.x 테스트된 통합 페이지를 검토했습니다.
- *사용자 프로비저닝 인프라가 있는 클러스터의 요구 사항* 섹션에 자세히 설명된 인프라 요구 사항을 검토했습니다.

프로세스

1. DHCP를 사용하여 클러스터 노드에 IP 네트워킹 구성을 제공하는 경우 DHCP 서비스를 구성합니다.
   a. 노드의 영구 IP 주소를 DHCP 서버 구성에 추가합니다. 구성에서 관련 네트워크 인터페이스의 MAC 주소를 각 노드의 의도한 IP 주소와 일치시킵니다.
   b. DHCP를 사용하여 클러스터 시스템의 IP 주소 지정을 구성하는 경우 시스템은 DHCP를 통해 DNS 서버 정보도 가져옵니다. DHCP 서버 구성을 통해 클러스터 노드에서 사용하는 영구 DNS 서버 주소를 정의합니다.



참고

DHCP 서비스를 사용하지 않는 경우 RHCOS 설치 시 IP 네트워킹 구성과 DNS 서버의 주소를 노드에 제공해야 합니다. ISO 이미지에서 설치하는 경우 부팅 인수로 전달할 수 있습니다. 고정 IP 프로비저닝 *및 고급 네트워킹 옵션에 대한 자세한 내용은 RHCOS 설치 및 OpenShift Container Platform 부트스트랩 프로세스 시작* 섹션을 참조하십시오.

c.

DHCP 서버 구성에 클러스터 노드의 호스트 이름을 정의합니다. 호스트 이름 고려 사항에 대한 자세한 내용은 ***DHCP****를 통해 클러스터 노드 호스트 이름 설정* 섹션을 참조하십시오.

참고

DHCP 서비스를 사용하지 않는 경우 클러스터 노드는 역방향 **DNS** 조회를 통해 호스트 이름을 가져옵니다.

2.

네트워크 인프라가 클러스터 구성 요소 간 필수 네트워크 연결을 제공하는지 확인합니다. 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 인프라 섹션의 네트워킹 요구 사항* 섹션을 참조하십시오.

3.

OpenShift Container Platform 클러스터 구성 요소가 통신하는 데 필요한 포트를 활성화하도록 방화벽을 구성합니다. 필요한 포트에 대한 자세한 내용은 *사용자 프로비저닝 인프라 섹션의 네트워킹 요구 사항* 섹션을 참조하십시오.

4.

클러스터에 필요한 **DNS** 인프라를 설정합니다.

a.

Kubernetes API, 애플리케이션 와일드카드, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템의 **DNS** 이름 확인을 구성합니다.

b.

Kubernetes API, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템에 대한 역방향 **DNS** 확인을 구성합니다.

OpenShift Container Platform DNS 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 **DNS** 요구 사항* 섹션을 참조하십시오.

5.

DNS 구성을 확인합니다.

a.

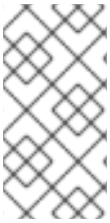
설치 노드에서 **Kubernetes API**의 레코드 이름, 와일드카드 경로 및 클러스터 노드에 대해 **DNS** 조회를 실행합니다. 응답의 **IP** 주소가 올바른 구성 요소에 해당하는지 확인합니다.

b.

설치 노드에서 로드 밸런서 및 클러스터 노드의 **IP** 주소에 대해 역방향 **DNS** 조회를 실행합니다. 응답의 레코드 이름이 올바른 구성 요소에 해당하는지 확인합니다.

자세한 DNS 검증 단계는 *사용자 프로비저닝 인프라에 대한 DNS 확인* 섹션을 참조하십시오.

6. 필요한 API 및 애플리케이션 수신 로드 밸런싱 인프라를 프로비저닝합니다. 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구 사항* 섹션을 참조하십시오.

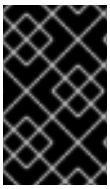


참고

일부 로드 밸런싱 솔루션에는 로드 밸런싱을 초기화하기 전에 클러스터 노드의 DNS 이름을 확인해야 합니다.

18.7.7. 사용자 프로비저닝 인프라에 대한 DNS 확인 검증

사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치하기 전에 DNS 구성을 확인할 수 있습니다.



중요

클러스터를 설치하기 전에 이 섹션에 설명된 검증 단계를 성공해야 합니다.

사전 요구 사항

- 사용자 프로비저닝 인프라에 필요한 DNS 레코드를 구성했습니다.

프로세스

1. 설치 노드에서 Kubernetes API의 레코드 이름, 와일드카드 경로 및 클러스터 노드에 대해 DNS 조회를 실행합니다. 응답에 포함된 IP 주소가 올바른 구성 요소에 해당하는지 확인합니다.

   a. Kubernetes API 레코드 이름을 조회합니다. 결과가 API 로드 밸런서의 IP 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
```

1 <nameserver_ip>를 네임서버의 IP 주소로, <cluster_name>을 클러스터 이름으로, <base_domain>을 기본 도메인 이름으로 바꿉니다.

출력 예

```
api.ocp4.example.com.  0 IN A 192.168.1.5
```

b. Kubernetes 내부 API 레코드 이름을 조회합니다. 결과가 API 로드 밸런서의 IP 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
```

출력 예

```
api-int.ocp4.example.com.  0 IN A 192.168.1.5
```

c. 예제 *.apps.<cluster_name>.<base_domain>을 테스트합니다. DNS 와일드카드를 조회합니다. 모든 애플리케이션 와일드카드 조회는 애플리케이션 인그레스 로드 밸런서의 IP 주소로 확인되어야 합니다.

```
$ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.
<base_domain>
```

출력 예

```
random.apps.ocp4.example.com.  0 IN A 192.168.1.5
```

참고

예제 출력에서는 **Kubernetes API** 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 **API** 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

random 항목을 다른 와일드카드 값으로 교체할 수 있습니다. 예를 들어 **OpenShift Container Platform** 콘솔의 경로를 쿼리할 수 있습니다.

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

출력 예

```
console-openshift-console.apps.ocp4.example.com. 0 IN A 192.168.1.5
```

d.

부트스트랩 **DNS** 레코드 이름에 대해 조회를 실행합니다. 결과가 부트스트랩 노드의 **IP** 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.
<base_domain>
```

출력 예

```
bootstrap.ocp4.example.com.  0 IN A 192.168.1.96
```

e.

이 방법을 사용하여 컨트롤 플레인 및 컴퓨팅 노드의 **DNS** 레코드 이름에 대해 조회를 수행합니다. 결과가 각 노드의 **IP** 주소에 해당하는지 확인합니다.

2.

설치 노드에서 로드 밸런서 및 클러스터 노드의 **IP** 주소에 대해 역방향 **DNS** 조회를 실행합니다. 응답에 포함된 레코드 이름이 올바른 구성 요소에 해당하는지 확인합니다.

a.

API 로드 밸런서의 IP 주소에 대해 역방향 조회를 수행합니다. 응답에 Kubernetes API 및 Kubernetes 내부 API의 레코드 이름이 포함되어 있는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

출력 예

```
5.1.168.192.in-addr.arpa. 0 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 0 IN PTR api.ocp4.example.com. 2
```

1

Kubernetes 내부 API의 레코드 이름을 제공합니다.

2

Kubernetes API의 레코드 이름을 제공합니다.

참고

OpenShift Container Platform 애플리케이션 와일드카드에는 PTR 레코드가 필요하지 않습니다. 애플리케이션 인그레스 로드 밸런서의 IP 주소에 대한 역방향 DNS 확인에는 유효성 검사 단계가 필요하지 않습니다.

b.

부트스트랩 노드의 IP 주소에 대해 역방향 조회를 수행합니다. 결과가 부트스트랩 노드의 DNS 레코드 이름을 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

출력 예

```
96.1.168.192.in-addr.arpa. 0 IN PTR bootstrap.ocp4.example.com.
```

c.

이 방법을 사용하여 컨트롤 플레인 및 컴퓨팅 노드의 IP 주소에 대해 역방향 조회를 수행합니다. 결과가 각 노드의 DNS 레코드 이름과 일치하는지 확인합니다.

18.7.8. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

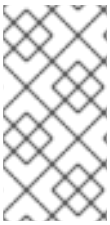
OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 core 사용자의 ~/.ssh/authorized_keys 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 core로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. ./openshift-install gather 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.



참고

AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

프로세스

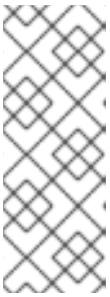
1.

로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1

새 SSH 키의 경로 및 파일 이름(예: ~/.ssh/id_ed25519)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 '~/.ssh 디렉터리에 있는지 확인하십시오.



참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 ed25519 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 rsa 또는 ecdsa 알고리즘을 사용하는 키를 생성합니다.

2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 ~/.ssh/id_ed25519.pub 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 ./openshift-install gather 명령을 사용하려는 경우 필요합니다.

참고

일부 배포에서는 ~/.ssh/id_rsa 및 ~/.ssh/id_dsa와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. ssh-agent 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. ssh-agent에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: ~/.ssh/id_ed25519).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다.

18.7.9. 설치 프로그램 받기

OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

프로세스

1. **OpenShift Cluster Manager** 사이트의 인프라 공급자 페이지에 액세스합니다. **Red Hat** 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.

2. 인프라 공급자를 선택합니다.

3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.

중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.

중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 **OpenShift Container Platform** 설치 제거 절차를 완료해야 합니다.

4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 **Linux** 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 **OpenShift Container Platform** 구성 요소에 대한 컨테이너 이미지를 제공하는 **Quay.io**를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

18.7.10. 수동으로 설치 구성 파일 만들기

OpenShift Container Platform을 사용자가 프로비저닝한 설치의 경우 설치 구성 파일을 수동으로 생성합니다.

사전 요구 사항

- 로컬 시스템에 설치 프로그램에 제공할 **SSH** 공개 키가 있습니다. 키는 디버깅 및 재해 복구

를 위해 클러스터 노드에 대한 **SSH** 인증에 사용됩니다.

- **OpenShift Container Platform** 설치 프로그램과 클러스터의 풀 시크릿이 있습니다.

프로세스

1. 필요한 설치 자산을 저장할 설치 디렉터리를 만듭니다.

```
$ mkdir <installation_directory>
```

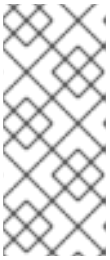
중요

디렉터리를 만들어야 합니다. 부트스트랩 **X.509** 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 **OpenShift Container Platform** 버전에서 설치 파일을 복사할 때는 주의하십시오.

2. 샘플 **install-config.yaml** 파일 템플릿을 사용자 지정하여 **<installation_directory>**에 저장합니다.

참고

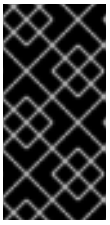
이 설정 파일의 이름을 **install-config.yaml**로 지정해야 합니다.



참고

일부 플랫폼 유형의 경우 대체로 **./openshift-install create install-config --dir <installation_directory>** 를 실행하여 **install-config.yaml** 파일을 생성할 수 있습니다. 프롬프트에서 클러스터 구성에 대한 세부 정보를 제공할 수 있습니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.



중요

install-config.yaml 파일은 설치 과정의 다음 단계에서 사용됩니다. 이 시점에서 이를 백업해야 합니다.

18.7.10.1. VMware vSphere용 샘플 install-config.yaml 파일

install-config.yaml 파일을 사용자 지정하여 **OpenShift Container Platform** 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.

```
apiVersion: v1
baseDomain: example.com 1
compute:
- hyperthreading: Enabled 2 3
  name: worker
  replicas: 0 4
controlPlane:
  hyperthreading: Enabled 5 6
  name: master
  replicas: 3 7
metadata:
  name: test 8
platform:
  vsphere:
    vcenter: your.vcenter.server 9
    username: username 10
    password: password 11
    datacenter: datacenter 12
    defaultDatastore: datastore 13
    folder: "/<datacenter_name>/vm/<folder_name>/<subfolder_name>" 14
fips: false 15
pullSecret: '{"auths": ...}' 16
sshKey: 'ssh-ed25519 AAAA...' 17
```

1

클러스터의 기본 도메인입니다. 모든 **DNS** 레코드는 이 기본 도메인의 하위 도메인이어야 하며 클러스터 이름을 포함해야 합니다.

2 5

controlPlane 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 현재 두 섹션이 모두 단일 시스템 풀을 정의하지만 향후 출시되는 **OpenShift Container Platform** 버전은 설치 과정에서 여러 컴퓨팅 풀 정의를 지원할 수 있습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

3 6

동시 멀티스레딩 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. 일부 클러스터 시스템에서 동시 멀티스레딩을 비활성화할 경우에는 해당 멀티스레딩을 모든 클러스터 시스템에서 비활성화해야 합니다.

중요

동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다. 동시 멀티스레딩을 비활성화하는 경우 시스템은 8개 이상의 **CPU**와 **32GB**의 **RAM**을 사용해야 합니다.

4

replicas 매개변수의 값을 **0**으로 설정해야 합니다. 이 매개변수는 클러스터를 생성하고 관리하는 작업자 수를 제어합니다. 이는 사용자 프로비저닝 인프라를 사용할 때 클러스터가 실행하지 않는 기능입니다. **OpenShift Container Platform** 설치를 완료하기 전에 클러스터에서 사용할 작업자 시스템을 수동으로 배포해야 합니다.

7

클러스터에 추가하는 컨트롤 플레인 시스템의 수입니다. 클러스터에서 이 값을 클러스터의 **etcd** 끝점 수로 사용하므로 이 값은 배포하는 컨트롤 플레인 시스템의 수와 일치해야 합니다.

8

DNS 레코드에 지정한 클러스터 이름입니다.

9

vCenter 서버의 정규화된 호스트 이름 또는 **IP** 주소입니다.

10

서버에 액세스하기 위한 사용자의 이름입니다. 이 사용자에게는 최소한 **vSphere**에서 정적 또는 동적 영구 볼륨 프로비저닝 에 필요한 역할과 권한이 있어야 합니다.

11

vSphere 사용자와 연결된 암호입니다.

12

vSphere 데이터 센터입니다.

13

사용할 기본 **vSphere** 데이터 저장소입니다.

14

선택사항: 설치 관리자 프로비저닝 인프라의 경우 설치 프로그램이 가상 머신을 생성하는 기존 폴더의 절대 경로(예: **/<datacenter_name>/vm/<folder_name>/<subfolder_name>**). 이 값을 제공하지 않으면 설치 프로그램이 인프라 **ID**로 이름이 지정된 데이터 센터 가상 머신 폴더에 최상위 폴더를 만듭니다. 클러스터에 인프라를 제공하는 경우에는 이 매개변수를 생략합니다.

15

FIPS 모드 활성화 또는 비활성화 여부입니다. 기본적으로 **FIPS** 모드는 비활성화됩니다. **FIPS** 모드가 활성화되면 **OpenShift Container Platform**이 실행되는 **RHCOS(Red Hat Enterprise Linux CoreOS)** 시스템에서 기본 **Kubernetes** 암호화 제품군은 우회하고 **RHCOS**와 함께 제공되는 암호화 모듈을 대신 사용합니다.

중요

FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 **OpenShift Container Platform** 배포에서만 지원됩니다.

16

OpenShift Cluster Manager 에서 얻은 풀 시크릿입니다. 이 풀 시크릿을 사용하면 **OpenShift Container Platform** 구성 요소에 대한 컨테이너 이미지를 제공하는 **Quay.io**를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

17

RHCOS(Red Hat Enterprise Linux CoreOS)의 **core** 사용자에 대한 기본 **SSH** 키의 공용 부분입니다.

18.7.10.2. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 **HTTP** 또는 **HTTPS** 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 **OpenShift Container Platform** 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.

- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 **API**에 대한 호출을 포함하여 모든 클러스터 발신(**Egress**) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.

참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr, networking.clusterNetwork[].cidr, networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 **Red Hat OpenStack Platform (RHOSP)**에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. 클러스터 외부에서 **HTTP** 연결을 구축하는 데 사용할 프록시 **URL**입니다. **URL** 스키마는 **http**여야 합니다.

2. 클러스터 외부에서 **HTTPS** 연결을 구축하는 데 사용할 프록시 **URL**입니다.

3.

4

이 값을 제공하면 설치 프로그램에서 추가 CA 인증서를 보유할 openshift -config 네임스페이스에 user-ca- bundle 이라는 구성 맵을 생성합니다. additionalTrustBundle 및 하나 이상의 프록시 설정을 제공하는 경우 프록시 오브젝트는 trustedCA 필드의 user-ca-bundle 구성 맵을 참조하도록 구성됩니다. 그러면 Cluster Network Operator에서 trustedCA 매개변수에 대해 지정된 콘텐츠를 RHCOS 신뢰 번들과 병합하는 trusted-ca-bundle 구성 맵을 생성합니다. 프록시의 ID 인증서를 RHCOS 트러스트 번들에 있는 기관에서 서명하지 않은 경우 additionalTrustBundle 필드가 있어야 합니다.

참고

설치 프로그램에서 프록시 adinessEndpoints 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 OpenShift Container Platform을 설치할 때 참조하십시오.

제공되는 install-config.yaml 파일의 프록시 설정을 사용하는 cluster라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 cluster Proxy 오브젝트는 계속 생성되지만 spec은 nil이 됩니다.

참고

cluster라는 Proxy 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

18.7.11. 고급 네트워크 구성 지정

클러스터 네트워크 제공자의 고급 네트워크 구성을 사용하여 클러스터를 기존 네트워크 환경에 통합할 수 있습니다. 클러스터를 설치하기 전에만 고급 네트워크 구성을 지정할 수 있습니다.

중요

설치 프로그램에서 생성한 OpenShift Container Platform 매니페스트 파일을 수정하여 네트워크 구성을 사용자 정의하는 것은 지원되지 않습니다. 다음 절차에서와 같이 생성한 매니페스트 파일을 적용할 수 있습니다.

사전 요구 사항

- install-config.yaml 파일을 생성하고 수정 작업을 완료했습니다.

절차

1. 설치 프로그램이 포함된 디렉터리로 변경하고 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 <installation_directory>는 클러스터의 install-config.yaml 파일이 포함된 디렉터리의 이름을 지정합니다.

2. <installation_directory>/ manifests/ 디렉토리에 cluster-network-03-config.yml이라는 stub 매니페스트 파일을 만듭니다.

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
```

3. 다음과 같이 cluster-network-03-config.yml 파일에서 클러스터의 고급 네트워크 구성을 지정합니다.

OpenShift SDN 네트워크 공급자에 대한 다른 VXLAN 포트 지정

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    openshiftSDNConfig:
      vxlanPort: 4800
```

OVN-Kubernetes 네트워크 공급자의 IPsec 활성화

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    ovnKubernetesConfig:
      ipsecConfig: {}
```

4. 선택사항: **manifests/cluster-network-03-config.yml** 파일을 백업합니다. 설치 프로그램은 **Ignition** 구성 파일을 생성할 때 **manifests/** 디렉터리를 사용합니다.

5. 컨트롤 플레인 시스템을 정의하는 **Kubernetes** 매니페스트 파일을 제거하고 **machineSets**를 컴퓨팅합니다.

```
$ rm -f openshift/99_openshift-cluster-api_master-machines-*.yaml
openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

이러한 리소스는 사용자가 직접 생성하고 관리하기 때문에 초기화할 필요가 없습니다.

- 시스템 **API**로 **MachineSet** 파일을 보존하여 컴퓨팅 시스템을 생성할 수 있지만 사용자 환경과 일치하도록 해당 참조를 업데이트해야 합니다.

18.7.12. CNO(Cluster Network Operator) 구성

클러스터 네트워크의 구성은 **CNO(Cluster Network Operator)** 구성의 일부로 지정되며 **cluster**라는 이름의 **CR**(사용자 정의 리소스) 오브젝트에 저장됩니다. **CR**은 **operator.openshift.io API** 그룹에서 **Network API**의 필드를 지정합니다.

CNO 구성은 **Network.config.openshift.io API** 그룹의 **Network API**에서 클러스터 설치 중에 다음 필드를 상속하며 이러한 필드는 변경할 수 없습니다.

clusterNetwork

Pod IP 주소가 할당되는 **IP** 주소 풀입니다.

serviceNetwork

서비스를 위한 IP 주소 풀입니다.

defaultNetwork.type

OpenShift SDN 또는 OVN-Kubernetes와 같은 클러스터 네트워크 공급자입니다.

cluster라는 CNO 오브젝트에서 defaultNetwork 오브젝트의 필드를 설정하여 클러스터의 클러스터 네트워크 공급자 구성을 지정할 수 있습니다.

18.7.12.1. CNO(Cluster Network Operator) 구성 오브젝트

CNO(Cluster Network Operator)의 필드는 다음 표에 설명되어 있습니다.

표 18.68. CNO(Cluster Network Operator) 구성 오브젝트

필드	유형	설명
metadata.name	**string**	CNO 개체 이름입니다. 이 이름은 항상 **cluster**입니다.

spec.clusterNetwork	**array**	Pod IP 주소가 할당되는 IP 주소 블록과 클러스터의 각 개별 노드에 할당된 서브넷 접두사 길이를 지정하는 목록입니다. 예를 들면 다음과 같습니다.
		`spec:`
		`  clusterNetwork:`
		`  - cidr: 10.128.0.0/19`
		`  hostPrefix: 23`
		`  - cidr: 10.128.32.0/19`
		`  hostPrefix: 23`
		매니페스트를 생성하기 전에 **install-config.yaml** 파일에서만 이 필드를 사용자 지정할 수 있습니다. 값은 매니페스트 파일에서 읽기 전용입니다.

필드	유형	설명
spec.serviceNetwork	**array**	서비스의 IP 주소 블록입니다. OpenShift SDN 및 OVN-Kubernetes CNI(Container Network Interface) 네트워크 공급자는 서비스 네트워크에 대한 단일 IP 주소 블록만 지원합니다. 예를 들면 다음과 같습니다. spec: serviceNetwork: - 172.30.0.0/14 매니페스트를 생성하기 전에 **install-config.yaml** 파일에서만 이 필드를 사용자 지정할 수 있습니다. 값은 매니페스트 파일에서 읽기 전용입니다.
spec.defaultNetwork	**object**	클러스터 네트워크의 CNI(Container Network Interface) 클러스터 네트워크 공급자를 구성합니다.
spec.kubeProxyConfig	**object**	이 개체의 필드는 kube-proxy 구성을 지정합니다. OVN-Kubernetes 클러스터 네트워크 공급자를 사용하는 경우 kube-proxy 구성이 적용되지 않습니다.

defaultNetwork 오브젝트 구성

defaultNetwork 오브젝트의 값은 다음 표에 정의되어 있습니다.

표 18.69. defaultNetwork 오브젝트

필드	유형	설명
type	**string**	**OpenShiftSDN** 또는 **OVNKubernetes** 중 하나이며, 클러스터 네트워크 공급자가 설치 중에 선택됩니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다. **참고** OpenShift Container Platform은 기본적으로 OpenShift SDN CNI(Container Network Interface) 클러스터 네트워크 공급자를 사용합니다.
openshiftSDNConfig	**object**	이 오브젝트는 OpenShift SDN 클러스터 네트워크 공급자에만 유효합니다.
ovnKubernetesConfig	**object**	이 오브젝트는 OVN-Kubernetes 클러스터 네트워크 공급자에만 유효합니다.

OpenShift SDN CNI 네트워크 공급자에 대한 구성

다음 표에서는 **OpenShift SDN Container Network Interface (CNI)** 클러스터 네트워크 공급자의 구성 필드를 설명합니다.

표 18.70. openshiftSDNConfig 오브젝트

필드	유형	설명
mode	**string**	OpenShift SDN의 네트워크 격리 모드를 구성합니다. 기본값은 **NetworkPolicy**입니다. **Multitenant** 및 **Subnet** 값은 OpenShift Container Platform 3.x 버전과의 역호환할 수 있지만 권장되지는 않습니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
mtu	**integer**	VXLAN 오버레이 네트워크의 최대 전송 단위(MTU)입니다. 이는 기본 네트워크 인터페이스의 MTU를 기준으로 자동 탐지됩니다. 일반적으로 감지된 MTU를 재정의할 필요는 없습니다. 자동 감지 값이 예상 밖인 경우 노드의 기본 네트워크 인터페이스의 MTU가 올바른지 확인합니다. 이 옵션을 사용하여 노드의 기본 네트워크 인터페이스의 MTU 값을 변경할 수 없습니다. 클러스터에 다른 노드에 대한 다른 MTU 값이 필요한 경우 이 값을 클러스터의 가장 낮은 MTU 값보다 **50** 미만으로 설정해야 합니다. 예를 들어 클러스터의 일부 노드에는 **9001**의 MTU가 있고 일부에는 **1500**의 MTU가 있는 경우 이 값을 **1450**으로 설정해야 합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
vxlanPort	**integer**	모든 VXLAN 패킷에 사용할 포트입니다. 기본값은 **4789**입니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다. 기존 노드가 다른 VXLAN 네트워크에 속하는 가상 환경에서 실행 중인 경우에는 기본값을 변경해야 할 수도 있습니다. 예를 들어 VMware NSX-T 위에서 OpenShift SDN 오버레이를 실행할 때 두 SDN이 동일한 기본 VXLAN 포트 번호를 사용하므로 VXLAN의 대체 포트를 선택해야 합니다. AWS(Amazon Web Services)에서는 포트 **9000**과 포트 **9999** 사이에서 VXLAN의 대체 포트를 선택할 수 있습니다.

OpenShift SDN 구성 예

```
defaultNetwork:
  type: OpenShiftSDN
  openshiftSDNConfig:
    mode: NetworkPolicy
    mtu: 1450
    vxlanPort: 4789
```

OVN-Kubernetes CNI 클러스터 네트워크 공급자에 대한 구성

다음 표에서는 OVN-Kubernetes CNI 클러스터 네트워크 공급자의 구성 필드를 설명합니다.

표 18.71. ovnKubernetesConfig object

필드	유형	설명
mtu	**integer**	Geneve(Generic Network Virtualization Encapsulation) 오버레이 네트워크의 MTU(최대 전송 단위)입니다. 이는 기본 네트워크 인터페이스의 MTU를 기준으로 자동 탐지됩니다. 일반적으로 감지된 MTU를 재정의할 필요는 없습니다. 자동 감지 값이 예상 밖인 경우 노드의 기본 네트워크 인터페이스의 MTU가 올바른지 확인합니다. 이 옵션을 사용하여 노드의 기본 네트워크 인터페이스의 MTU 값을 변경할 수 없습니다. 클러스터에 다른 노드에 대한 다른 MTU 값이 필요한 경우, 이 값을 클러스터의 가장 낮은 MTU 값보다 **100** 미만으로 설정해야 합니다. 예를 들어, 클러스터의 일부 노드에 **9001**의 MTU가 있고 일부에는 **1500**의 MTU가 있는 경우 이 값을**1400**으로 설정해야 합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
genevePort	**integer**	모든 Geneve 패킷에 사용할 포트입니다. 기본값은 **6081**입니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
ipsecConfig	**object**	IPsec 암호화를 활성화하려면 빈 오브젝트를 지정합니다. 클러스터를 설치한 후에는 이 값을 변경할 수 없습니다.
policyAuditConfig	**object**	네트워크 정책 감사 로깅을 사용자 정의할 구성 오브젝트를 지정합니다. 설정되지 않으면 기본값 감사 로그 설정이 사용됩니다.

표 18.72. policyAuditConfig object

필드	유형	설명
rateLimit	integer	노드당 1초마다 생성할 최대 메시지 수입니다. 기본값은 초당 **20** 개의 메시지입니다.
maxFileSize	integer	감사 로그의 최대 크기(바이트)입니다. 기본값은 **50000000** 또는 **50** MB입니다.

필드	유형	설명
대상	string	다음 추가 감사 로그 대상 중 하나입니다. **libc** 호스트에서 journald 프로세스의 libc **syslog()** 함수입니다. **udp:<host>:<port>** syslog 서버입니다. **<host>:<port>**를 syslog 서버의 호스트 및 포트로 바꿉니다. **unix:<file>** **<file>**로 지정된 Unix Domain Socket 파일입니다. **null** 감사 로그를 추가 대상으로 보내지 마십시오.
syslogFacility	string	RFC5424에 정의된 **kern**과 같은 syslog 기능입니다. 기본값은 **local0**입니다.

OVN-Kubernetes 구성 예

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
    ipsecConfig: {}
```

kubeProxyConfig 오브젝트 구성

kubeProxyConfig 오브젝트의 값은 다음 표에 정의되어 있습니다.

표 18.73. kubeProxyConfig object

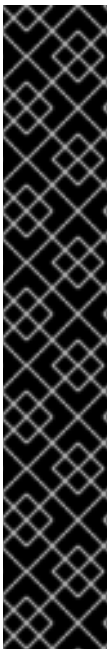
필드	유형	설명

필드	유형	설명
iptablesSyncPeriod	**string**	**iptables** 규칙의 새로 고침 간격입니다. 기본값은 **30s**입니다. 유효 접미사로 **s**, **m**, **h**가 있으며, 자세한 설명은 Go **time** 패키지 문서를 참조하십시오.  **참고** OpenShift Container Platform 4.3 이상에서는 성능이 개선되어 더 이상 **iptablesSyncPeriod** 매개변수를 조정할 필요가 없습니다.

proxyArguments.iptables-min-sync-period	**array**	**iptables** 규칙을 새로 고치기 전 최소 기간입니다. 이 필드를 통해 새로 고침 간격이 너무 짧지 않도록 조정할 수 있습니다. 유효 접미사로 **s**, **m**, **h**가 있으며, 자세한 설명은 Go **time** 패키지를 참조하십시오. 기본값은 다음과 같습니다.
		
		`kubeProxyConfig:`
		`  proxyArguments:`
		`  iptables-min-sync-period:`
		`  - 0s`

18.7.13. Ignition 구성 파일 생성

클러스터 시스템을 수동으로 시작해야 하므로 클러스터가 시스템을 생성하는 데 필요한 Ignition 구성 파일을 사용자가 생성해야 합니다.



중요

- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 node-bootstrapper 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *만료된 컨트롤 플레인 인증서에서 복구* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받습니다. 제한된 네트워크 설치의 경우, 해당 파일은 미러 호스트에 있습니다.

프로세스

- Ignition 구성 파일을 가져옵니다.

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 <installation_directory>는 설치 프로그램이 생성하는 파일을 저장할 디렉터리 이름을 지정합니다.

중요

install-config.yaml 파일을 생성한 경우 파일이 포함된 디렉터리를 지정하십시오. 그렇지 않으면 빈 디렉터리를 지정합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

다음 파일이 디렉터리에 생성됩니다.

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

18.7.14. 인프라 이름 추출

Ignition 구성 파일에는 AWS의 VMware Cloud에서 클러스터를 고유하게 식별하는 데 사용할 수 있는 고유한 클러스터 식별자가 포함되어 있습니다. 클러스터 ID를 가상 머신 폴더의 이름으로 사용하려면 해당 ID를 추출해야 합니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받으셨습니다.
- 클러스터에 대한 Ignition 구성 파일을 생성하셨습니다.
- jq CLI를 설치하셨습니다.

프로세스

- Ignition 구성 파일 메타데이터에서 인프라 이름을 추출하여 확인하려면 다음 명령을 실행하십시오.

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

출력 예

```
openshift-vw9j6 1
```

1 이 명령의 출력은 클러스터 이름과 임의의 문자열입니다.

18.7.15. RHCOS 설치 및 OpenShift Container Platform 부트스트랩 프로세스 시작

VMware vSphere의 사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치하려면 vSphere 호스트에 RHCOS(Red Hat Enterprise Linux Core OS)를 설치해야 합니다. RHCOS를 설치할 때 설치 중인 머신 유형에 대해 OpenShift Container Platform 설치 프로그램에서 생성한 Ignition 구성 파일을 제공해야 합니다. 적합한 네트워킹, DNS 및 로드 밸런싱 인프라를 구성한 경우 RHCOS 머신이 재부팅된 후 OpenShift Container Platform 부트스트랩 프로세스가 자동으로 시작됩니다.

사전 요구 사항

- 클러스터에 대한 Ignition 구성 파일이 있어야 합니다.
- 사용자 컴퓨터에서 액세스할 수 있고 생성한 시스템이 액세스할 수 있는 HTTP 서버에 액세스할 수 있습니다.
- vSphere 클러스터 를 생성했습니다.

프로세스

1. 설치 프로그램에서 생성된 부트스트랩 Ignition 구성 파일 (<installation_directory>/bootstrap.ign)을 HTTP 서버에 업로드합니다. 이 파일의 URL을 기록해 둡니다.
2. 부트스트랩 노드의 다음 보조 Ignition 구성 파일을 <installation_directory>/merge-bootstrap.ign으로 컴퓨터에 저장합니다.

```
{
  "ignition": {
    "config": {
      "merge": [
        {
          "source": "<bootstrap_ignition_config_url>", 1
          "verification": {}
        }
      ]
    },
    "timeouts": {},
    "version": "3.2.0"
  },
  "networkd": {},
  "passwd": {},
  "storage": {},
  "systemd": {}
}
```

1. 호스팅한 부트스트랩 Ignition 구성 파일의 URL을 지정합니다.

부트스트랩 머신에 대한 VM(가상 머신)을 생성할 때 이 Ignition 구성 파일을 사용합니다.

3. 설치 프로그램이 생성한 다음 **Ignition** 구성 파일을 찾습니다.

- **<installation_directory>/master.ign**
- **<installation_directory>/worker.ign**
- **<installation_directory>/merge-bootstrap.ign**

4. **Ignition** 구성 파일을 **Base64** 인코딩으로 변환합니다. 이 절차의 뒷부분에서는 이러한 파일을 **VM**의 추가 구성 매개변수 **guestinfo.ignition.config.data** 에 추가해야 합니다.

예를 들어 **Linux** 운영 체제를 사용하는 경우 **base64** 명령을 사용하여 파일을 인코딩할 수 있습니다.

```
$ base64 -w0 <installation_directory>/master.ign > <installation_directory>/master.64
```

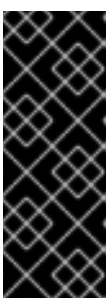
```
$ base64 -w0 <installation_directory>/worker.ign > <installation_directory>/worker.64
```

```
$ base64 -w0 <installation_directory>/merge-bootstrap.ign >
<installation_directory>/merge-bootstrap.64
```

중요

설치를 마친 후 클러스터에 컴퓨팅 시스템을 더 추가하려면 **Ignition** 구성 파일을 삭제하지 마십시오.

5. **RHCOS OVA** 이미지를 가져옵니다. **RHCOS** 이미지 미러 페이지에서 이미지를 사용할 수 있습니다.



중요

RHCOS 이미지는 **OpenShift Container Platform** 릴리스에 따라 변경되지 않을 수 있습니다. 설치하는 **OpenShift Container Platform** 버전과 같거나 그 이하의 버전 중 가장 최신 버전의 이미지를 다운로드해야 합니다. 지원되는 경우 **OpenShift Container Platform** 버전과 일치하는 이미지 버전을 사용합니다.

파일 이름에는 rhcos-vmware.<architecture>.ova 형식의 OpenShift Container Platform 버전 번호가 포함됩니다.

6. vSphere Client에서 VM을 저장할 데이터 센터 폴더를 생성합니다.

   a. VMs and Templates 보기를 클릭합니다.

   b. 데이터 센터 이름을 마우스 오른쪽 버튼으로 클릭합니다.

   c. New Folder → New VM and Template Folder를 클릭합니다.

   d. 표시되는 창에서 폴더 이름을 입력합니다. install-config.yaml 파일에서 기존 폴더를 지정하지 않은 경우 인프라 ID와 동일한 이름으로 폴더를 생성합니다. vCenter는 이 폴더 이름을 사용하여 Workspace 구성에 적절한 위치에 스토리지를 동적으로 프로비저닝합니다.

7. vSphere Client에서 OVA 이미지에 대한 템플릿을 생성한 다음 필요에 따라 템플릿을 복제합니다.

참고

다음 단계에서 템플릿을 생성한 다음 모든 클러스터 시스템에 대한 템플릿을 복제합니다. 그런 다음 VM을 프로비저닝할 때 복제된 머신 유형의 Ignition 구성 파일의 위치를 제공합니다.

   a. Hosts and Clusters 탭에서 클러스터 이름을 마우스 오른쪽 버튼으로 클릭하고 Deploy OVF Template을 선택합니다.

   b. Select an OVF 탭에서 다운로드한 RHCOS OVA 파일의 이름을 지정합니다.

   c. 이름 및 폴더 선택 탭에서 템플릿의 가상 시스템 이름 (예: Template-RHCOS)을 설정합니다. vSphere 클러스터의 이름을 클릭하고 이전 단계에서 생성한 폴더를 선택합니다.

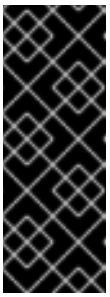
d. **Select a compute resource** 탭에서 **vSphere** 클러스터 이름을 클릭합니다.

e. **Select storage** 탭에서 **VM**의 스토리지 옵션을 구성합니다.

- 스토리지 기본 설정에 따라 **Thin Provision** 또는 **Thick Provision**을 선택합니다.
- **install-config.yaml** 파일에서 지정한 데이터 저장소를 선택합니다.

f. **Select network** 탭에서 사용 가능한 경우 클러스터에 대해 구성한 네트워크를 지정합니다.

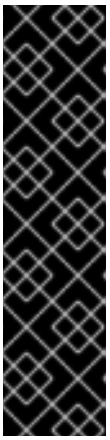
g. **OVF** 템플릿을 생성할 때 템플릿 사용자 지정 탭에 값을 지정하지 않거나 템플릿을 추가로 구성하지 마십시오.



중요

원래 **VM** 템플릿을 시작하지 마십시오. **VM** 템플릿이 꺼져 있어야 하며 새 **RHCOS** 머신에 대해 복제해야 합니다. **VM** 템플릿을 시작하면 **VM** 템플릿이 플랫폼의 **VM**으로 구성되므로 시스템 세트가 구성을 적용할 수 있는 템플릿으로 사용되지 않습니다.

8. 선택 사항: 필요한 경우 **VM** 템플릿에서 구성된 가상 하드웨어 버전을 업데이트합니다. 자세한 내용은 가상 머신을 **VMware** 설명서의 최신 하드웨어 버전으로 업그레이드를 참조하십시오.



중요

필요한 경우 **VM** 템플릿의 하드웨어 버전을 버전 **15**로 업데이트하는 것이 좋습니다. 이제 **vSphere**에서 실행 중인 클러스터 노드에 하드웨어 버전 **13**을 사용하는 것이 더 이상 사용되지 않습니다. 가져온 템플릿의 기본값이 하드웨어 버전 **13**인 경우 **VM** 템플릿을 하드웨어 버전 **15**로 업그레이드하기 전에 **ESXi** 호스트가 **6.7U3** 이상인지 확인해야 합니다. **vSphere** 버전이 **6.7U3** 미만인 경우 이 업그레이드 단계를 건너뛸 수 있습니다. 그러나 향후 **OpenShift Container Platform** 버전은 **6.7U3** 미만의 하드웨어 버전 **13** 및 **vSphere** 버전에 대한 지원을 제거하도록 예약되어 있습니다.

9. 템플릿이 배포된 후 클러스터에서 시스템의 가상 머신을 배포합니다.

a. 템플릿 이름을 마우스 오른쪽 버튼으로 클릭하고 **Clone → Clone to Virtual Machine**을 클릭합니다.

b. **Select a name and folder** 탭에서 가상 머신의 이름을 지정합니다. **control-plane-0** 또는 **compute-1**과 같은 시스템 유형을 이름에 포함할 수 있습니다.

c. **Select a name and folder** 탭에서 클러스터에 대해 생성한 폴더의 이름을 선택합니다.

d. **Select a compute resource** 탭에서 데이터 센터의 호스트 이름을 선택합니다.

e. 선택사항: **Select storage** 탭에서 스토리지 옵션을 사용자 지정합니다.

f. **Select clone options**에서 **Customize this virtual machine's hardware**를 선택합니다.

g. **Customize hardware** 탭에서 **VM Options → Advanced**를 클릭합니다.

- 선택 사항: **vSphere**에서 기본 **DHCP** 네트워킹을 재정의합니다. 고정 **IP** 네트워킹을 활성화하려면 다음을 수행합니다.

  i. 고정 **IP** 구성을 설정합니다.

```
$ export IPCFG="ip=<ip>::<gateway>:<netmask>:<hostname>:
<iface>:none nameserver=srv1 [nameserver=srv2 [nameserver=srv3 [...]]]"
```

명령 예

```
$ export IPCFG="ip=192.168.100.101::192.168.100.254:255.255.255.0:::none
nameserver=8.8.8.8"
```

ii.

guestinfo.afterburn.initrd.network-kargs 속성을 설정한 후 **vSphere**의 **OVA**에서 **VM**을 부팅합니다.

```
$ govc vm.change -vm "<vm_name>" -e
"guestinfo.afterburn.initrd.network-kargs=${IPCFG}"
```

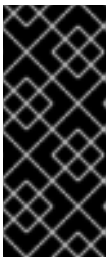
- 선택사항: 클러스터 성능 문제가 발생하는 경우 **Latency Sensitivity** 목록에서 **High**를 선택합니다. **VM**의 **CPU** 및 메모리 예약에 다음과 같은 값이 있는지 확인합니다.
  - 메모리 예약 값은 구성된 메모리 크기와 같아야 합니다.
  - **CPU** 예약 값은 측정된 물리적 **CPU** 속도를 곱한 최소 대기 시간이 짧은 가상 **CPU** 수여야 합니다.
- 구성 편집을 클릭하고 **Configuration Parameters** 창에서 사용 가능한 매개 변수 목록을 검색하여 스틸 클럭 회계 **(stealclock.enable)**를 검색합니다. 사용 가능한 경우 해당 값을 **TRUE** 로 설정합니다. 스틸링 클럭 계정을 사용하면 클러스터 문제 해결에 도움이 될 수 있습니다.
- 구성 추가 매개변수를 클릭합니다. 다음 매개변수 이름 및 값을 정의합니다.
  - **guestinfo.ignition.config.data:** 이 절차에서 이전에 생성한 **base-64** 인코딩 파일을 찾고 이 머신 유형에 대해 **base64**로 인코딩된 **Ignition** 구성 파일의 내용을 붙여넣습니다.
  - **guestinfo.ignition.config.data.encoding: base64**를 지정합니다.
  - **disk.EnableUUID: TRUE**를 지정합니다.
  - **stealclock.enable:**이 매개 변수가 정의되지 않은 경우 추가하고 **TRUE** 를 지정합니다.

h.

Customize hardware 탭의 **Virtual Hardware** 패널에서 지정된 값을 필요에 따라 수정합니다. **RAM, CPU** 및 디스크 스토리지의 양이 시스템 유형에 대한 최소 요구사항을 충족하는지 확인합니다.

i. 구성을 완료하고 **VM**의 전원을 켭니다.

10. 각 시스템에 대해 이전 단계에 따라 클러스터의 나머지 시스템을 생성합니다.



중요

이때 부트스트랩 및 컨트롤 플레인 시스템을 생성해야 합니다. 일부 **Pod**는 기본적으로 컴퓨팅 시스템에 배포되므로 클러스터를 설치하기 전에 컴퓨팅 시스템을 두 개 이상 생성합니다.

18.7.16. vSphere의 클러스터에 더 많은 컴퓨팅 머신 추가

VMware vSphere에서 사용자가 프로비저닝한 **OpenShift Container Platform** 클러스터에 더 많은 컴퓨팅 머신을 추가할 수 있습니다.

사전 요구 사항

- 컴퓨팅 머신의 **base64**로 인코딩된 **Ignition** 파일을 가져옵니다.
- 클러스터에 생성한 **vSphere** 템플릿에 액세스할 수 있습니다.

프로세스

1. 템플릿이 배포된 후 클러스터에서 시스템의 가상 머신을 배포합니다.

a. 템플릿 이름을 마우스 오른쪽 버튼으로 클릭하고 **Clone → Clone to Virtual Machine**을 클릭합니다.

b. **Select a name and folder** 탭에서 가상 머신의 이름을 지정합니다. **compute-1**과 같은 머신 유형을 이름에 포함할 수 있습니다.

c. **Select a name and folder** 탭에서 클러스터에 대해 생성한 폴더의 이름을 선택합니다.

d. **Select a compute resource** 탭에서 데이터 센터의 호스트 이름을 선택합니다.

e. 선택사항: **Select storage** 탭에서 스토리지 옵션을 사용자 지정합니다.

f. **Select clone options**에서 **Customize this virtual machine's hardware**를 선택합니다.

g. **Customize hardware** 탭에서 **VM Options → Advanced**를 클릭합니다.

- **Latency Sensitivity** 목록에서 **High**를 선택합니다.
- **Edit Configuration**을 클릭하고 **Configuration Parameters** 창에서 **Add Configuration Params**를 클릭합니다. 다음 매개변수 이름 및 값을 정의합니다.
  - **guestinfo.ignition.config.data:** 이 머신 유형에 대해 **base64**로 인코딩된 컴퓨팅 **Ignition** 구성 파일의 내용을 붙여넣습니다.
  - **guestinfo.ignition.config.data.encoding: base64**를 지정합니다.
  - **disk.EnableUUID: TRUE**를 지정합니다.

h. **Customize hardware** 탭의 **Virtual Hardware** 패널에서 지정된 값을 필요에 따라 수정합니다. **RAM, CPU** 및 디스크 스토리지의 양이 시스템 유형에 대한 최소 요구사항을 충족하는지 확인합니다. 또한 사용 가능한 네트워크가 여러 개인 경우 **Add network adapter**에서 올바른 네트워크를 선택해야 합니다.

i. 구성을 완료하고 **VM**의 전원을 켭니다.

2. 계속해서 클러스터에 추가 컴퓨팅 머신을 만듭니다.

18.7.17. 디스크 파티션 설정

대부분의 경우 데이터 파티션은 원래 다른 운영 체제를 설치하는 대신 RHCOS를 설치하여 생성됩니다. 이러한 경우 OpenShift Container Platform 설치 프로그램은 디스크 파티션을 설정할 수 있어야합니다.

그러나 OpenShift Container Platform 노드를 설치할 때 기본 파티션 설정을 덮어 쓰기하기 위해 개입이 필요한 두 가지 경우가 있습니다.

- 별도의 파티션 생성: 빈 디스크에 그린 필드 설치의 경우 파티션에 별도의 스토리지를 추가할 수 있습니다. 이는 /var 또는 /var의 하위 디렉터리 (예: /var/lib/etcd)중 하나를 별도의 파티션으로 만드는 경우에 공식적으로 지원됩니다.

중요

100GB 이상의 디스크 크기 및 특히 1TB보다 큰 디스크 크기의 경우 별도의 /var 파티션을 만듭니다. 자세한 내용은 " 별도의 /var 파티션 생성" 및 이 Red Hat 지식베이스 문서 를 참조하십시오.

중요

Kubernetes는 두 개의 파일 시스템 파티션만 지원합니다. 원래 구성에 둘 이상의 파티션을 추가하는 경우 Kubernetes는 모든 파티션을 모니터링할 수 없습니다.

- 기존 파티션 유지: 브라운 필드 설치에서 기존 노드에 OpenShift Container Platform을 재설치하고 이전 운영 체제에 설치된 데이터 파티션을 유지해야 하는 경우 기존 데이터 파티션을 유지할 수 있는 coreos-installer에 부팅 인수와 옵션이 모두 있습니다.

별도의 /var 파티션 만들기

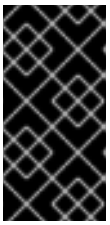
일반적으로 OpenShift Container Platform의 디스크 파티션 설정은 설치 프로그램에 맡겨야합니다. 그러나 확장하려는 파일 시스템의 일부에 별도의 파티션을 생성해야 하는 경우가 있습니다.

OpenShift 컨테이너 플랫폼은 /var 파티션 또는 /var의 하위 디렉터리 중 하나에 스토리지를 연결하는 단일 파티션의 추가를 지원합니다. 예를 들면 다음과 같습니다.

- /var/lib/containers: 시스템에 더 많은 이미지와 컨테이너가 추가됨에 따라 확장될 수 있는 컨테이너 관련 콘텐츠를 보관합니다.

- /var/lib/etcd: etcd 스토리지의 성능 최적화와 같은 목적으로 별도로 보관할 데이터를 보관합니다.

- /var: 감사 등의 목적에 맞게 별도로 분리하여 보관해야 하는 데이터를 보관합니다.



중요

100GB보다 큰 디스크 크기 및 특히 1TB보다 큰 디스크의 경우 별도의 /var 파티션을 만듭니다.

/var 디렉터리의 콘텐츠를 별도로 저장하면 필요에 따라 해당 영역에 대한 스토리지 확장을 보다 용이하게 하고 나중에 OpenShift Container Platform을 다시 설치하여 해당 데이터를 그대로 보존할 수 있습니다. 이 방법을 사용하면 모든 컨테이너를 다시 가져올 필요가 없으며 시스템을 업데이트할 때 대용량 로그 파일을 복사할 필요도 없습니다.

RHCOS(Red Hat Enterprise Linux CoreOS)를 새로 설치하기 전에 /var가 있어야 하므로 다음 절차에서는 OpenShift Container Platform 설치의 openshift-install 준비 단계 중에 삽입되는 머신 구성 매니페스트를 생성하여 별도의 /var 파티션을 설정합니다.

절차

1. OpenShift Container Platform 설치 파일을 저장할 디렉터리를 만듭니다.

```
$ mkdir $HOME/clusterconfig
```

2. openshift-install을 실행하여 manifest 및 openshift 하위 디렉터리에 파일 세트를 만듭니다. 프롬프트가 표시되면 시스템 질문에 대답합니다.

```
$ openshift-install create manifests --dir $HOME/clusterconfig
? SSH Public Key ...
$ ls $HOME/clusterconfig/openshift/
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

3.

추가 파티션을 구성하는 Butane 구성을 생성합니다. 예를 들어 파일 이름을 $HOME/clusterconfig/98-var-partition.bu 로 지정하고, 디스크 장치 이름을 worker 시스템의 스토리지 장치 이름으로 변경하고 스토리지 크기를 적절하게 설정합니다. 이 예에서는 /var 디렉토리를 별도의 파티션에 배치합니다.

```
variant: openshift
version: 4.9.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1

파티션을 설정해야하는 디스크 저장 장치 이름입니다.

2

데이터 파티션을 부트 디스크에 추가할 때 최소 25000 메비 바이트가 권장됩니다. 루트 파일 시스템은 지정된 오프셋까지 사용 가능한 모든 공간을 채우기 위해 자동으로 크기가 조정됩니다. 값이 지정되지 않거나 지정된 값이 권장 최소값보다 작으면 생성되는 루트 파일 시스템의 크기가 너무 작아지고 RHCOS를 나중에 다시 설치할 때 데이터 파티션의 첫 번째 부분을 덮어 쓸 수 있습니다.

3

데이터 파티션의 크기(MB)입니다.

4

컨테이너 스토리지에 사용되는 파일 시스템에 prjquota 마운트 옵션을 활성화해야 합니다.

참고

별도의 /var 파티션을 만들 때 다른 인스턴스 유형에 동일한 장치 이름이 없는 경우 작업자 노드에 다른 인스턴스 유형을 사용할 수 없습니다.

4. Butane 구성에서 매니페스트를 생성하여 clusterconfig/openshift 디렉터리에 저장합니다. 예를 들어 다음 명령을 실행합니다.

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o
$HOME/clusterconfig/openshift/98-var-partition.yaml
```

5. openshift-install을 다시 실행하여 manifest 및 openshift 하위 디렉터리의 파일 세트에서 Ignition 구성을 만듭니다.

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

이제 Ignition 구성 파일을 vSphere 설치 절차에 대한 입력으로 사용하여 RHCOS (Red Hat Enterprise Linux CoreOS) 시스템을 설치할 수 있습니다.

18.7.18. bootupd를 사용하여 부트로더 업데이트

bootupd를 사용하여 부트로더를 업데이트하려면 RHCOS 머신에 수동으로 bootupd를 설치하거나 systemd 유닛이 활성화된 머신 구성을 제공해야 합니다. grubby 또는 기타 부트로더 툴과는 달리 bootupd는 커널 인수 전달과 같은 커널 공간 구성을 관리하지 않습니다.

bootupd를 설치한 후 OpenShift Container Platform 클러스터에서 원격으로 관리할 수 있습니다.

참고

bootupd는 BootHole 취약점에 대한 보호와 같이 베어 메탈 또는 가상화 하이퍼바이저 설치에서만 사용하는 것이 좋습니다.

수동 설치 방법

bootctl 명령줄 툴을 사용하여 bootupd를 수동으로 설치할 수 있습니다.

1. 시스템 상태를 검사합니다.

```
# bootupctl status
```

출력 예

```
Component EFI
  Installed: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
  Update: At latest version
```

2. bootupd를 설치하지 않고 생성된 RHCOS 이미지에는 명시적 채택 단계가 필요합니다.

시스템 상태가 Adoptable인 경우 채택을 수행합니다.

```
# bootupctl adopt-and-update
```

출력 예

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

3. 업데이트를 사용할 수 있는 경우 다음 재부팅에 변경 사항이 적용되도록 업데이트를 적용합니다.

```
# bootupctl update
```

출력 예

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

시스템 구성 방법

bootupd를 활성화하는 또 다른 방법은 머신 구성을 제공하는 것입니다.

- 다음 예와 같이 **systemd** 단위가 활성화된 머신 구성 파일을 제공합니다.

출력 예

```
variant: rhcos
version: 1.1.0
systemd:
  units:
    - name: custom-bootupd-auto.service
      enabled: true
      contents: |
        [Unit]
        Description=Bootupd automatic update

        [Service]
        ExecStart=/usr/bin/bootupctl update
        RemainAfterExit=yes

        [Install]
        WantedBy=multi-user.target
```

18.7.19. 부트스트랩 프로세스가 완료될 때까지 대기 중

OpenShift Container Platform 부트스트랩 프로세스는 클러스터 노드가 먼저 디스크에 설치된 영구 **RHCOS** 환경으로 부팅된 후에 시작됩니다. **Ignition** 구성 파일을 통해 제공되는 구성 정보는 부트스트랩 프로세스를 초기화하고 머신에 **OpenShift Container Platform**을 설치하는 데 사용됩니다. 부트스트랩 프로세스가 완료될 때까지 기다려야 합니다.

사전 요구 사항

- 클러스터에 대한 **Ignition** 구성 파일이 생성되어 있습니다.

- 적합한 네트워크, DNS 및 로드 밸런싱 인프라가 구성되어 있습니다.
- 설치 프로그램을 받아서 클러스터의 Ignition 구성 파일을 생성했습니다.
- 클러스터 머신에 RHCOS를 설치하고 OpenShift Container Platform 설치 프로그램에서 생성된 Ignition 구성 파일을 제공했습니다.
- 사용자 시스템에서 직접 인터넷에 액세스하거나 HTTP 또는 HTTPS 프록시를 사용할 수 있습니다.

프로세스

1. 부트스트랩 프로세스를 모니터링합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2 다른 설치 세부 사항을 보려면 info 대신 warn, debug 또는 error를 지정합니다.

출력 예

```
INFO Waiting up to 30m0s for the Kubernetes API at
https://api.test.example.com:6443...
INFO API v1.22.1 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

이 명령은 Kubernetes API 서버가 컨트롤 플레인 시스템에서 부트스트랩되었다는 신호를 보낼 때 성공합니다.

2. 부트스트랩 프로세스가 완료된 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다.

중요

이 시점에 로드 밸런서에서 부트스트랩 시스템을 제거해야 합니다. 부트스트랩 머신 자체를 제거하거나 다시 포맷할 수도 있습니다.

18.7.20. CLI를 사용하여 클러스터에 로그인

클러스터 kubeconfig 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. kubeconfig 파일에는 CLI에서 올바른 클러스터 및 API 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 OpenShift Container Platform 설치 과정에서 생성됩니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포했습니다.
- oc CLI를 설치했습니다.

프로세스

1. kubeadmin 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 oc 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

18.7.21. 머신의 인증서 서명 요청 승인

클러스터에 시스템을 추가하면 추가한 시스템별로 보류 중인 인증서 서명 요청(CSR)이 두 개씩 생성됩니다. 이러한 CSR이 승인되었는지 확인해야 하며, 필요한 경우 이를 직접 승인해야 합니다. 클라이언트 요청을 먼저 승인한 다음 서버 요청을 승인해야 합니다.

사전 요구 사항

- 클러스터에 시스템을 추가했습니다.

프로세스

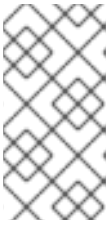
1. 클러스터가 시스템을 인식하는지 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0  Ready    master  63m  v1.22.1
master-1  Ready    master  63m  v1.22.1
master-2  Ready    master  64m  v1.22.1
```

출력에 생성된 모든 시스템이 나열됩니다.



참고

이전 출력에는 일부 CSR이 승인될 때까지 컴퓨팅 노드(작업자 노드라고도 함)가 포함되지 않을 수 있습니다.

2.

보류 중인 CSR을 검토하고 클러스터에 추가한 각 시스템에 대해 Pending 또는 Approved 상태의 클라이언트 및 서버 요청이 표시되는지 확인합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE    REQUESTOR                                                   CONDITION
csr-8b2br   15m    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

예에서는 두 시스템이 클러스터에 참여하고 있습니다. 목록에는 승인된 CSR이 더 많이 나타날 수도 있습니다.

3.

CSR이 승인되지 않은 경우, 추가된 시스템에 대한 모든 보류 중인 CSR이 Pending 상태로 전환된 후 클러스터 시스템의 CSR을 승인합니다.

참고

CSR은 교체 주기가 자동으로 만료되므로 클러스터에 시스템을 추가한 후 1 시간 이내에 CSR을 승인하십시오. 한 시간 내에 승인하지 않으면 인증서가 교체되고 각 노드에 대해 두 개 이상의 인증서가 표시됩니다. 이러한 인증서를 모두 승인해야 합니다. 클라이언트 CSR이 승인되면 Kubelet은 인증서에 대한 보조 CSR을 생성하므로 수동 승인이 필요합니다. 그러면 Kubelet에서 동일한 매개변수를 사용하여 새 인증서를 요청하는 경우 인증서 갱신 요청은 machine-approver에 의해 자동으로 승인됩니다.

참고

베어 메탈 및 기타 사용자 프로비저닝 인프라와 같이 머신 **API**를 사용하도록 활성화되지 않는 플랫폼에서 실행되는 클러스터의 경우 **CSR(Kubelet service Certificate Request)**을 자동으로 승인하는 방법을 구현해야 합니다. 요청이 승인되지 않으면 **API** 서버가 **kubelet**에 연결될 때 서비스 인증서가 필요하므로 **oc exec, oc rsh, oc logs** 명령을 성공적으로 수행할 수 없습니다. **Kubelet** 엔드 포인트에 연결하는 모든 작업을 수행하려면 이 인증서 승인이 필요합니다. 이 방법은 새 **CSR**을 감시하고 **CSR**이 **system:node** 또는 **system:admin** 그룹의 **node-bootstrapper** 서비스 계정에 의해 제출되었는지 확인하고 노드의 **ID**를 확인합니다.

- 개별적으로 승인하려면 유효한 **CSR** 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1

<csr_name>은 현재 **CSR** 목록에 있는 **CSR**의 이름입니다.

- 보류 중인 **CSR**을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

참고

일부 **Operator**는 일부 **CSR**이 승인될 때까지 사용할 수 없습니다.

4. 이제 클라이언트 요청이 승인되었으므로 클러스터에 추가한 각 머신의 서버 요청을 검토해야 합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
```

```
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 나머지 CSR이 승인되지 않고 Pending 상태인 경우 클러스터 머신의 CSR을 승인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 <csr_name>은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs oc adm certificate approve
```

6. 모든 클라이언트 및 서버 CSR이 승인된 후 머신은 Ready 상태가 됩니다. 다음 명령을 실행하여 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0  Ready    master  73m  v1.22.1
master-1  Ready    master  73m  v1.22.1
master-2  Ready    master  74m  v1.22.1
worker-0  Ready    worker  11m  v1.22.1
worker-1  Ready    worker  11m  v1.22.1
```

참고

머신이 **Ready** 상태로 전환하는 데 서버 **CSR**의 승인 후 몇 분이 걸릴 수 있습니다.

추가 정보

- **CSR**에 대한 자세한 내용은 인증서 서명 요청을 참조하십시오.

18.7.22. Operator의 초기 설정

컨트롤 플레인이 초기화된 후 일부 **Operator**를 즉시 구성하여 모두 사용 가능하도록 해야 합니다.

사전 요구 사항

- 컨트롤 플레인이 초기화되어 있습니다.

절차

1. 클러스터 구성 요소가 온라인 상태인지 확인합니다.

```
$ watch -n5 oc get clusteroperators
```

출력 예

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.9.0     True        False         False      19m
baremetal                                  4.9.0     True        False         False      37m
cloud-credential                           4.9.0     True        False         False      40m
cluster-autoscaler                         4.9.0     True        False         False      37m
config-operator                            4.9.0     True        False         False      38m
console                                    4.9.0     True        False         False      26m
csi-snapshot-controller                    4.9.0     True        False         False      37m
dns                                        4.9.0     True        False         False      37m
etcd                                       4.9.0     True        False         False      36m
image-registry                             4.9.0     True        False         False      31m
ingress                                    4.9.0     True        False         False      30m
insights                                   4.9.0     True        False         False      31m
kube-apiserver                             4.9.0     True        False         False      26m
kube-controller-manager                    4.9.0     True        False         False      36m
kube-scheduler                             4.9.0     True        False         False      36m
```

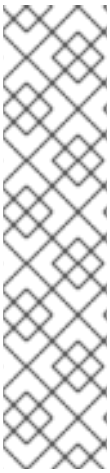
```
kube-storage-version-migrator              4.9.0     True        False         False      37m
machine-api                                4.9.0     True        False         False      29m
machine-approver                           4.9.0     True        False         False      37m
machine-config                             4.9.0     True        False         False      36m
marketplace                                4.9.0     True        False         False      37m
monitoring                                 4.9.0     True        False         False      29m
network                                    4.9.0     True        False         False      38m
node-tuning                                4.9.0     True        False         False      37m
openshift-apiserver                        4.9.0     True        False         False      32m
openshift-controller-manager               4.9.0     True        False         False      30m
openshift-samples                          4.9.0     True        False         False      32m
operator-lifecycle-manager                 4.9.0     True        False         False      37m
operator-lifecycle-manager-catalog         4.9.0     True        False         False      37m
operator-lifecycle-manager-packageserver   4.9.0     True        False         False      32m
service-ca                                 4.9.0     True        False         False      38m
storage                                    4.9.0     True        False         False      37m
```

2. 사용할 수 없는 **Operator**를 구성합니다.

18.7.22.1. 설치 중 제거된 이미지 레지스트리

공유 가능한 개체 스토리지를 제공하지 않는 플랫폼에서 **OpenShift Image Registry Operator**는 자체적으로 **Removed**로 부트스트랩합니다. 이를 통해 **openshift-installer**가 이러한 플랫폼 유형에서 설치를 완료할 수 있습니다.

설치 후 **managementState**를 **Removed**에서 **Managed**로 전환하도록 **Image Registry Operator** 구성을 편집해야 합니다.



참고

Prometheus 콘솔은 **ImageRegistryRemoved** 경고를 제공합니다. 다음은 예시 경고입니다.

"이미지 레지스트리가 제거되었습니다. **ImageStreamTags**를 참조하는 **ImageStreamTags, BuildConfigs** 및 **DeploymentConfigs**가 예상대로 작동하지 않을 수 있습니다. 스토리지를 구성하고 **configs.imageregistry.operator.openshift.io**를 편집하여 구성을 **Managed** 상태로 업데이트하십시오."

18.7.22.2. 이미지 레지스트리 스토리지 구성

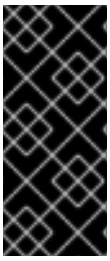
기본 스토리지를 제공하지 않는 플랫폼에서는 처음에 **Image Registry Operator**를 사용할 수 없습니다. 설치한 후에 스토리지를 사용하도록 레지스트리를 구성하여 **Registry Operator**를 사용 가능하도록 만들어야 합니다.

프로덕션 클러스터에 필요한 영구 볼륨을 구성하는 과정의 지침이 표시됩니다. 해당하는 경우, 프로덕션 환경 외 클러스터에서만 사용할 수 있는 저장 위치로서 빈 디렉터리를 구성하는 과정의 지침이 표시됩니다.

업그레이드 중에 **Recreate** 롤아웃 전략을 사용하여 이미지 레지스트리의 블록 스토리지 유형 사용을 허용하기 위한 추가 지침이 제공됩니다.

18.7.22.2.1. VMware vSphere용 블록 레지스트리 스토리지 구성

클러스터 관리자로서 업그레이드 중에 이미지 레지스트리가 **vSphere VMDK(Virtual Machine Disk)**와 같은 블록 스토리지 유형을 사용할 수 있도록 허용하기 위해 **Recreate** 롤아웃 전략을 사용할 수 있습니다.



중요

블록 스토리지 볼륨이 지원되지만 프로덕션 클러스터에서 이미지 레지스트리와 함께 사용하는 것은 권장되지 않습니다. 레지스트리가 블록 스토리지에 구성된 설치는 레지스트리가 둘 이상의 복제본을 가질 수 없기 때문에 가용성이 높지 않습니다.

절차

1. 이미지 레지스트리 스토리지를 블록 스토리지 유형으로 설정하려면 **Recreate** 롤아웃 전략을 사용하고 복제본 **1**개 만으로 실행되도록 레지스트리를 패치합니다.

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. 블록 스토리지 장치에 **PV**를 프로비저닝하고 해당 볼륨의 **PVC**를 생성합니다. 요청된 블록 볼륨은 **RWO(ReadWriteOnce)** 액세스 모드를 사용합니다.

   a. **VMware vSphere PersistentVolumeClaim** 개체를 정의하려면 다음 내용이 포함된 **pvc.yaml** 파일을 생성합니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
```

```
  name: image-registry-storage 1
  namespace: openshift-image-registry 2
spec:
  accessModes:
  - ReadWriteOnce 3
  resources:
    requests:
      storage: 100Gi 4
```

1 PersistentVolumeClaim 개체를 표시하는 고유한 이름입니다.

2 PersistentVolumeClaim 오브젝트의 네임스페이스로 openshift-image-registry입니다.

3 영구 볼륨 클레임의 액세스 모드입니다. ReadWriteOnce를 사용하면 단일 노드에서 읽기 및 쓰기 권한으로 볼륨을 마운트할 수 있습니다.

4 영구 볼륨 클레임의 크기입니다.

b. 파일에서 PersistentVolumeClaim 오브젝트를 만듭니다.

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 올바른 PVC를 참조하도록 레지스트리 설정을 편집합니다.

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

출력 예

```
storage:
  pvc:
    claim: 1
```

1 사용자 지정 PVC를 만들면 image-registry-storage PVC의 기본 자동 생성을 위해 claim 필드를 비워둘 수 있습니다.

올바른 PVC를 참조하도록 레지스트리 스토리지를 구성하는 방법은 vSphere 용 레지스트리 구성을 참조하십시오.

18.7.23. 사용자 프로비저닝 인프라에 설치 완료

Operator 구성을 완료한 후 제공하는 인프라에 클러스터 설치를 완료할 수 있습니다.

사전 요구 사항

- 컨트롤 플레인이 초기화되어 있습니다.
- 초기 Operator 구성을 완료해야 합니다.

절차

1. 다음 명령을 사용하여 모든 클러스터 구성 요소가 온라인 상태인지 확인합니다.

```
$ watch -n5 oc get clusteroperators
```

출력 예

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.9.0     True        False         False      19m
baremetal                                  4.9.0     True        False         False      37m
cloud-credential                           4.9.0     True        False         False      40m
cluster-autoscaler                         4.9.0     True        False         False      37m
config-operator                            4.9.0     True        False         False      38m
console                                    4.9.0     True        False         False      26m
csi-snapshot-controller                    4.9.0     True        False         False      37m
dns                                        4.9.0     True        False         False      37m
```

```
etcd                                       4.9.0     True        False         False      36m
image-registry                             4.9.0     True        False         False      31m
ingress                                    4.9.0     True        False         False      30m
insights                                   4.9.0     True        False         False      31m
kube-apiserver                             4.9.0     True        False         False      26m
kube-controller-manager                    4.9.0     True        False         False      36m
kube-scheduler                             4.9.0     True        False         False      36m
kube-storage-version-migrator              4.9.0     True        False         False      37m
machine-api                                4.9.0     True        False         False      29m
machine-approver                           4.9.0     True        False         False      37m
machine-config                             4.9.0     True        False         False      36m
marketplace                                4.9.0     True        False         False      37m
monitoring                                 4.9.0     True        False         False      29m
network                                    4.9.0     True        False         False      38m
node-tuning                                4.9.0     True        False         False      37m
openshift-apiserver                        4.9.0     True        False         False      32m
openshift-controller-manager               4.9.0     True        False         False      30m
openshift-samples                          4.9.0     True        False         False      32m
operator-lifecycle-manager                 4.9.0     True        False         False      37m
operator-lifecycle-manager-catalog         4.9.0     True        False         False      37m
operator-lifecycle-manager-packageserver   4.9.0     True        False         False      32m
service-ca                                 4.9.0     True        False         False      38m
storage                                    4.9.0     True        False         False      37m
```

또는 다음 명령은 모든 클러스터를 사용할 수 있을 때 알립니다. 또한 인증 정보를 검색하고 표시합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1

<installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

출력 예

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator가 Kubernetes API 서버에서 OpenShift Container Platform 클러스터 배포를 완료하면 명령이 성공합니다.

중요

- 설치 프로그램에서 생성하는 **Ignition** 구성 파일에 **24**시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 **24**시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 **kubelet** 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청**(CSR)**을 수동으로 승인해야 합니다. 자세한 내용은 ***Recovering from expired control plane certificates*** 문서를 참조하십시오.

- 클러스터를 설치한 후 **24**시간에서 **22**시간까지의 인증서가 교체되기 때문에 생성된 후 **12**시간 이내에 **Ignition** 구성 파일을 사용하는 것이 좋습니다. **12**시간 이내에 **Ignition** 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

2. **Kubernetes API** 서버가 **Pod**와 통신하고 있는지 확인합니다.

   a. 모든 **Pod** 목록을 보려면 다음 명령을 사용하십시오.

   ```
   $ oc get pods --all-namespaces
   ```

   출력 예

   ```
   NAMESPACE                  NAME                                        READY   STATUS
   RESTARTS   AGE
   openshift-apiserver-operator      openshift-apiserver-operator-85cb746d55-zqhs8
   1/1     Running     1          9m
   openshift-apiserver            apiserver-67b9g                          1/1     Running     0
   3m
   openshift-apiserver            apiserver-ljcmx                          1/1     Running     0
   1m
   openshift-apiserver            apiserver-z25h4                          1/1     Running     0
   2m
   openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8
   1/1     Running     0          5m
   ...
   ```

   b. 다음 명령을 사용하여 이전 명령의 출력에 나열된 **Pod**의 로그를 표시합니다.

   -

```
$ oc logs <pod_name> -n <namespace> 1
```

1

이전 명령의 출력에 표시된 대로 **Pod** 이름과 네임스페이스를 지정합니다.

Pod 로그가 표시되면 **Kubernetes API** 서버는 클러스터 시스템과 통신할 수 있습니다.

3. **FCP(Fibre Channel Protocol)**를 사용하는 설치에는 다중 경로를 활성화하기 위해 추가 단계가 필요합니다. 설치 중에 멀티패스를 활성화하지 마십시오.

자세한 내용은 *설치 후 머신 구성 작업* 설명서에서 **"RHCOS에서 커널 인수를 사용하여 멀티패스 활성화"**를 참조하십시오.

vSphere에 컴퓨팅 머신 추가에 따라 클러스터 설치가 완료된 후 추가 컴퓨팅 머신을 추가할 수 있습니다.

18.7.24. VMware vSphere 볼륨 백업

OpenShift Container Platform은 새 볼륨을 독립 영구 디스크로 프로비저닝하여 클러스터의 모든 노드에서 볼륨을 자유롭게 연결 및 분리합니다. 결과적으로 스냅샷을 사용하는 볼륨을 백업하거나 스냅샷에서 볼륨을 복원할 수 없습니다. 자세한 내용은 **스냅 샷 제한** 을 참조하십시오.

절차

영구 볼륨의 백업을 생성하려면 다음을 수행합니다.

1. 영구 볼륨을 사용중인 애플리케이션을 중지합니다.
2. 영구 볼륨을 복제합니다.
3. 애플리케이션을 다시 시작합니다.
4. 복제된 볼륨의 백업을 만듭니다.

5. 복제된 볼륨을 삭제합니다.

18.7.25. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

18.7.26. 다음 단계

- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.
- 레지스트리를 설정하고 레지스트리 스토리지를 구성합니다.
- 선택 사항: vSphere Problem Detector Operator에서 이벤트를 보고 클러스터에 권한 또는 스토리지 구성 문제가 있는지 확인합니다.

18.8. 사용자 프로비저닝 인프라가 있는 제한된 네트워크에서 VMC에 클러스터 설치

OpenShift Container Platform 4.9 버전에서는 AWS의 VMC(VMware Cloud)에 배포하여 제한된 네트워크에서 프로비저닝하는 VMware vSphere 인프라에 클러스터를 설치할 수 있습니다.

OpenShift Container Platform 배포를 위해 VMC 환경을 구성한 후 VMC 환경에 공동 배치된 bastion 관리 호스트에서 OpenShift Container Platform 설치 프로그램을 사용합니다. 설치 프로그램 및 컨트롤

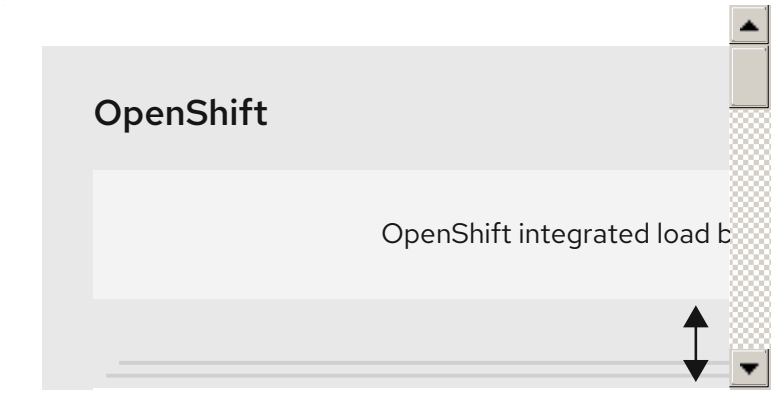
플레인은 OpenShift Container Platform 클러스터에 필요한 리소스 배포 및 관리 프로세스를 자동화합니다.

참고

OpenShift Container Platform은 단일 VMware vCenter에만 클러스터 배포를 지원합니다. 여러 vCenter에 머신/머신 세트가 있는 클러스터 배포는 지원되지 않습니다.

18.8.1. vSphere용 VMC 설정

AWS 호스팅된 vSphere 클러스터에 VMware Cloud (VMC)의 OpenShift Container Platform을 설치하여 하이브리드 클라우드 전반에 애플리케이션을 온프레미스 및 오프프레미스 모두 배포 및 관리할 수 있습니다.



VMware vSphere에 OpenShift Container Platform을 설치하기 전에 VMC 환경에서 여러 옵션을 구성해야 합니다. VMC 환경에 다음과 같은 사전 요구 사항이 있는지 확인합니다.

- 비독점 DHCP 사용, NSX-T 네트워크 세그먼트 및 서브넷을 생성합니다. 서브넷에서 다른 VM(가상 머신)을 호스팅할 수 있지만 OpenShift Container Platform 배포에서는 8개 이상의 IP 주소를 사용할 수 있어야 합니다.
- 다음 방화벽 규칙을 설정합니다.
  - 포트 443의 설치 호스트와 소프트웨어 정의 데이터 센터(SDDC) 관리 네트워크 간의 ANY:ANY 방화벽 규칙입니다. 이를 통해 배포 중에 RHCOS(Red Hat Enterprise Linux CoreOS) OVA를 업로드할 수 있습니다.
  - OpenShift Container Platform 컴퓨팅 네트워크와 인터넷 간의 HTTPS 방화벽 규칙입니다. 이 연결을 통해 OpenShift Container Platform은 노드, PVC(영구 볼륨 클레임) 및 기

타 리소스를 프로비저닝 및 관리하기 위해 **vCenter**과 통신할 수 있습니다.

- **OpenShift Container Platform**을 배포하려면 다음 정보가 있어야 합니다.
  - **OpenShift Container Platform** 클러스터에 **vmc-prod-1**로 로그인합니다.
  - **companyname.com**과 같은 기본 **DNS** 이름입니다.
  - 기본값을 사용하지 않는 경우 **Pod** 네트워크 **CIDR** 및 서비스 네트워크 **CIDR**을 식별해야 하며, 기본적으로 **10.128.0.0/14** 및 **172.30.0.0/16**으로 설정됩니다. 이러한 **CIDR**은 **pod-to-pod** 및 **pod-to-service** 통신에 사용되며 외부에서 액세스할 수 없지만 조직의 기존 서브넷과 중복되어서는 안 됩니다.
  - 다음 **vCenter** 정보를 참조하십시오:
    - **vCenter** 호스트 이름, 사용자 이름 및 암호
    - 데이터 센터 이름 (예: **SDDC-Datacenter**)
    - **Cluster-1**과 같은 클러스터 이름
    - 네트워크 이름
    - **WorkloadDatastore**와 같은 데이터 저장소 이름

참고

클러스터 설치가 완료된 후 **vSphere** 클러스터를 **VMC Compute-ResourcePool** 리소스 풀로 이동하는 것이 좋습니다.

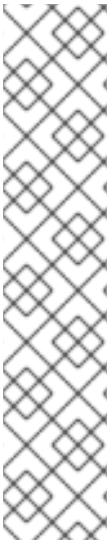
- **VMC**에 **bastion**으로 배포된 **Linux** 기반 호스트입니다.

- bastion 호스트는 RHEL(Red Hat Enterprise Linux) 또는 기타 Linux 기반 호스트일 수 있습니다. 인터넷에 연결되어 있어야 하고 OVA를 ESXi 호스트에 업로드하는 기능이 있어야 합니다.

- OpenShift CLI 툴을 bastion 호스트에 다운로드하여 설치합니다.

  - openshift-install 설치 프로그램

  - OpenShift CLI(oc) 툴



참고

Kubernetes 용 VMware NCP (NSX Conrainer Plugin)dOpen에는 사용할 수 없으며 NSX는 OpenShift SDN으로 사용되지 않습니다. 현재 VMC에서 사용할 수 있는 NSX 버전은 OpenShift Container Platform에서 인증된 NCP 버전과 호환되지 않습니다.

그러나 NSX DHCP 서비스는 전체 스택 자동화된 OpenShift Container Platform 배포와 vSphere와의 Machine API 통합을 통해 수동 또는 자동으로 프로비저닝된 노드를 사용하여 가상 머신 IP 관리에 사용됩니다. 또한 OpenShift Container Platform 클러스터와 bastion 호스트와 VMC vSphere 호스트 사이에서 액세스할 수 있도록 NSX 방화벽 규칙이 생성됩니다.

18.8.1.1. VMC Sizer 툴

AWS의 VMware Cloud는 AWS 베어메탈 인프라 상단에 구축됩니다. 이는 AWS 네이티브 서비스를 실행하는 것과 동일한 베어 메탈 인프라입니다. AWS 소프트웨어 정의 데이터 센터(SDDC)의 VMware 클라우드가 배포되면 이러한 물리적 서버 노드를 사용하고 단일 테넌트 방식으로 VMware ESXi 하이퍼바이저를 실행할 수 있습니다. 즉, VMC를 사용하는 모든 사용자가 물리적 인프라에 액세스할 수 없습니다. 가상 인프라를 호스팅하는데 필요한 물리적 호스트 수를 고려해야 합니다.

이를 확인하기 위해 VMware는 AWS Sizer에서 VMC를 제공합니다. 이 툴을 사용하면 VMC에서 호스팅할 리소스를 정의할 수 있습니다.

- 워크로드 유형

- 총 가상 머신 수

- 사양 정보는 다음과 같습니다:
  - 스토리지 요구사항
  - **vCPU**
  - **vRAM**
  - 오버커밋 비율

이러한 세부 정보를 사용하여 **VMware** 모범 사례를 기반으로 보고서를 생성하고 클러스터 구성과 필요한 호스트 수를 권장합니다.

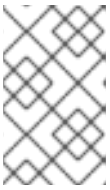
18.8.2. vSphere 사전 요구 사항

- **OpenShift Container Platform** 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.
- 미러 호스트에 레지스트리를 생성하고 사용 중인 **OpenShift Container Platform** 버전의 **imageContentSources** 데이터를 가져옵니다.

중요

미러 호스트에 설치 미디어가 있으므로 해당 컴퓨터를 사용하여 모든 설치 단계를 완료하십시오.

- 블록 레지스트리 스토리지가 프로비저닝되어 있습니다. 자세한 내용은 영구 저장 장치 이해를 참조하십시오.
- 방화벽을 사용하며 **Telemetry** 서비스를 사용할 예정인 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성해야 합니다.



참고

프록시를 구성하는 경우에도 해당 사이트 목록을 검토하십시오.

18.8.3. 네트워크가 제한된 환경에서의 설치 정보

OpenShift Container Platform 4.9에서는 소프트웨어 구성 요소를 받기 위한 인터넷 접속이 필요하지 않은 설치를 수행할 수 있습니다. 제한된 네트워크 설치는 클러스터를 설치하는 클라우드 플랫폼에 따라 설치 관리자 프로비저닝 인프라 또는 사용자 프로비저닝 인프라를 사용하여 완료할 수 있습니다.

클라우드 플랫폼에 제한된 네트워크 설치를 수행하는 방법을 선택해도 클라우드 **API**에 액세스는 가능해야 합니다. **Amazon Web Service**의 **Route 53 DNS** 및 **IAM** 서비스와 같은 일부 클라우드 기능에는 인터넷 액세스가 필요합니다. 사용 중인 네트워크에 따라 베어메탈 하드웨어 또는 **VMware vSphere**에 설치하기 위해 필요한 인터넷 액세스가 줄어들 수 있습니다.

제한된 네트워크 설치를 완료하려면 **OpenShift Container Platform** 레지스트리의 내용을 미러링하고 설치 미디어를 포함할 레지스트리를 생성해야 합니다. 인터넷과 폐쇄 네트워크에 모두 액세스하거나 제한 사항을 따르는 다른 방법을 통해 미러 호스트에 레지스트리를 생성할 수 있습니다.

중요

사용자 프로비저닝 설치의 구성이 복잡하므로 사용자 프로비저닝 인프라를 사용하여 제한된 네트워크 설치를 시도하기 전에 표준 사용자 프로비저닝 인프라 설치를 완료하는 것이 좋습니다. 이 테스트 설치를 완료하면 제한된 네트워크에 설치하는 동안 발생할 수 있는 문제를 보다 쉽게 파악 및 해결할 수 있습니다.

18.8.3.1. 추가 제한

제한된 네트워크의 클러스터에는 다음과 같은 추가 제한이 있습니다.

- **ClusterVersion** 상태에 사용 가능한 업데이트를 검색할 수 없음 오류가 포함되어 있습니다.
- 기본적으로 필요한 이미지 스트림 태그에 액세스할 수 없기 때문에 개발자 카탈로그의 내용을 사용할 수 없습니다.

18.8.4. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하기 위해 필요한 이미지를 받으려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- OpenShift Cluster Manager 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 **Telemetry** 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.
- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.

중요

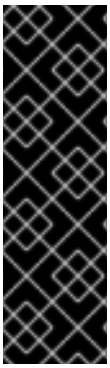
클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

18.8.5. VMware vSphere 인프라 요구사항

사용하는 구성 요소의 요구사항을 충족하는 **VMware vSphere** 버전 **6** 또는 **7** 인스턴스에 **OpenShift Container Platform** 클러스터를 설치해야 합니다.

표 **18.74. vSphere** 가상 환경의 버전 요구 사항

가상 환경 제품	필요한 버전
VM 하드웨어 버전	13 이상
vSphere ESXi hosts	6.5 이상
vCenter 호스트	6.5 이상



중요

이제 VMware vSphere 버전 6.7U2 또는 이전 버전에 클러스터를 설치하는 것이 더 이상 사용되지 않으며 가상 하드웨어 버전 13은 더 이상 사용되지 않습니다. 이러한 버전은 여전히 완전히 지원되지만 향후 OpenShift Container Platform 버전에서는 지원이 제거됩니다. 이제 OpenShift Container Platform의 vSphere 가상 머신의 하드웨어 버전 15가 기본값이 되었습니다. vSphere 노드의 하드웨어 버전을 업데이트하려면 " vSphere에서 실행되는 노드에서 하드웨어 업데이트" 문서를 참조하십시오.

표 18.75. VMware 구성 요소에 지원되는 최소 vSphere 버전

구성 요소	지원되는 최소 버전	설명
하이퍼바이저	HW 버전 13이 포함된 vSphere 6.5 이상	이 버전은 RHCOS(Red Hat Enterprise Linux CoreOS)가 지원하는 최소 버전입니다. Red Hat Enterprise Linux 8 지원 하이퍼바이저 목록을 참조하십시오.
In-tree 드라이버가 있는 스토리지	vSphere 6.5 이상	이 플러그인은 OpenShift Container Platform에 포함된 vSphere용 인트리 스토리지 드라이버를 사용하여 vSphere 스토리지를 생성합니다.

vSphere 버전 6.5 인스턴스를 사용하는 경우 OpenShift Container Platform을 설치하기 전에 6.7U3 또는 7.0로 업그레이드하는 것이 좋습니다.

중요

OpenShift Container Platform을 설치하기 전에 ESXi 호스트의 시간이 동기화되었는지 확인해야 합니다. VMware 문서에서 Edit Time Configuration for a Host를 참조하십시오.

추가 리소스

- vSphere 노드의 하드웨어 버전을 업데이트하려면 vSphere에서 실행 중인 노드에서 하드웨어 업데이트를 참조하십시오.

18.8.6. 사용자 프로비저닝 인프라를 포함한 클러스터의 시스템 요구사항

사용자 프로비저닝 인프라가 포함된 클러스터의 경우, 필요한 모든 시스템을 배포해야 합니다.

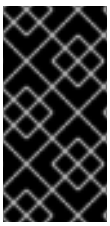
이 섹션에서는 사용자 프로비저닝 인프라에 OpenShift Container Platform을 배포해야 하는 요구 사항에 대해 설명합니다.

18.8.6.1. 클러스터 설치에 필요한 시스템

최소 OpenShift Container Platform 클러스터에 다음과 같은 호스트가 필요합니다.

표 18.76. 최소 필수 호스트

호스트	설명
임시 부트스트랩 시스템 한 개	컨트롤 플레인 시스템 세 개에 OpenShift Container Platform 클러스터를 배포하기 위한 부트스트랩 시스템이 클러스터에 필요합니다. 클러스터를 설치한 후 부트스트랩 시스템을 제거할 수 있습니다.
컨트롤 플레인 시스템 세 개	컨트롤 플레인 시스템은 컨트롤 플레인을 구성하는 Kubernetes 및 OpenShift Container Platform 서비스를 실행합니다.
두 개 이상의 컴퓨팅 시스템(작업자 시스템이라고도 함).	OpenShift Container Platform 사용자가 요청한 워크로드는 컴퓨팅 머신에서 실행됩니다.



중요

클러스터의 고가용성을 유지하려면 이러한 클러스터 시스템에 대해 별도의 물리적 호스트를 사용하십시오.

부트스트랩, 컨트롤 플레인 시스템은 운영 체제로 RHCOS (Red Hat Enterprise Linux CoreOS)를 사용해야 합니다. 그러나 컴퓨팅 머신은 RHCOS(Red Hat Enterprise Linux CoreOS), RHEL (Red Hat Enterprise Linux) 7.9 또는 RHEL 8.4 중에서 선택할 수 있습니다.

RHCOS는 RHEL 8(Red Hat Enterprise Linux)을 기반으로하며 모든 하드웨어 인증 및 요구사항을 모두 이어받습니다. Red Hat Enterprise Linux 기술 기능 및 제한을 참조하십시오.

18.8.6.2. 클러스터 설치를 위한 최소 리소스 요구 사항

각 클러스터 시스템이 다음과 같은 최소 요구사항을 충족해야 합니다.

표 18.77. 최소 리소스 요구사항

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
부트스트랩	RHCOS	4	16GB	100GB	300
컨트롤 플레인	RHCOS	4	16GB	100GB	300
컴퓨팅	RHCOS, RHEL 7.9 또는 RHEL 8.4 [3]	2	8GB	100GB	300

1. **SMT(동시 멀티스레딩) 또는 하이퍼 스레딩이 활성화되지 않은 경우 하나의 vCPU는 하나의 물리적 코어와 동일합니다. 활성화하면 다음과 같은 공식을 사용하여 해당 비율을 계산합니다. (코어 당 스레드 수 × 코어 수) × 소켓 수 = vCPU 수**

2. **OpenShift Container Platform 및 Kubernetes는 디스크 성능에 민감하며 특히 10ms p99 fsync 기간이 필요한 컨트롤 플레인 노드의 etcd에 더 빠른 스토리지가 권장됩니다. 많은 클라우드 플랫폼에서 스토리지 크기와 IOPS를 함께 확장되므로 충분한 성능을 얻으려면 스토리지 볼륨을 과도하게 할당해야 할 수 있습니다.**

3. **사용자가 프로비저닝한 모든 설치와 마찬가지로 클러스터에서 RHEL 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. RHEL 7 컴퓨팅 머신 사용은 더 이상 사용되지 않으며 향후 OpenShift Container Platform 4 릴리스에서 제거될 예정입니다.**

18.8.6.3. 인증서 서명 요청 관리

사용자가 프로비저닝하는 인프라를 사용하는 경우 자동 시스템 관리 기능으로 인해 클러스터의 액세스가 제한되므로 설치한 후 클러스터 인증서 서명 요청(CSR)을 승인하는 메커니즘을 제공해야 합니다. kube-controller-manager는 kubelet 클라이언트 CSR만 승인합니다. machine-approver는 올바른 시스템에서 발행한 요청인지 확인할 수 없기 때문에 kubelet 자격 증명을 사용하여 요청하는 서비스 인증서의 유효성을 보장할 수 없습니다. kubelet 서빙 인증서 요청의 유효성을 확인하고 요청을 승인하는 방법을 결정하여 구현해야 합니다.

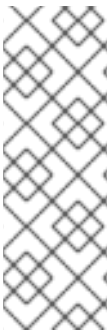
18.8.6.4. 사용자 프로비저닝 인프라에 대한 네트워킹 요구사항

모든 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템이 부팅 중에 Ignition 구성 파일을 가져오려면 initramfs에 네트워킹을 구성해야 합니다.

초기 부팅 과정에서 시스템에 필요한 부팅 옵션을 제공하여 DHCP 서버를 통해 설정하거나 정적으로 설정하는 IP 주소 구성이 필요합니다. 네트워크 연결이 설정된 후 시스템은 HTTP 또는 HTTPS 서버에서 Ignition 구성 파일을 다운로드합니다. 그런 다음 Ignition 구성 파일을 사용하여 각 머신의 정확한 상태를

설정합니다. Machine Config Operator는 설치 후 새 인증서 또는 키 적용과 같은 머신에 대한 추가 변경을 완료합니다.

클러스터 시스템의 장기적인 관리를 위해 DHCP 서버를 사용하는 것이 좋습니다. DHCP 서버가 클러스터 시스템에 영구 IP 주소, DNS 서버 정보 및 호스트 이름을 제공하도록 구성되었는지 확인합니다.



참고

사용자 프로비저닝 인프라에 DHCP 서비스를 사용할 수 없는 경우 RHCOS 설치 시 노드에 IP 네트워킹 구성과 DNS 서버의 주소를 대신 제공할 수 있습니다. ISO 이미지에서 설치하는 경우 부팅 인수로 전달할 수 있습니다. 고정 IP 프로비저닝 *및 고급 네트워킹 옵션에 대한 자세한 내용은 **RHCOS** 설치 및 **OpenShift Container Platform** 부트스트랩 프로세스 시작* 섹션을 참조하십시오.

Kubernetes API 서버가 클러스터 시스템의 노드 이름을 확인할 수 있어야 합니다. API 서버와 작업자 노드가 서로 다른 영역에 있는 경우, API 서버가 노드 이름을 확인할 수 있도록 기본 DNS 검색 영역을 설정할 수 있습니다. 노드 개체와 모든 DNS 요청에서 항상 정규화된 도메인 이름으로 호스트를 가리키는 것도 지원되는 방법입니다

18.8.6.4.1. DHCP를 통해 클러스터 노드의 호스트 이름 설정

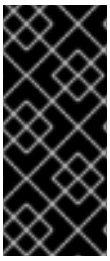
RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 호스트 이름은 NetworkManager를 통해 설정됩니다. 기본적으로 시스템은 DHCP를 통해 호스트 이름을 가져옵니다. DHCP에서 호스트 이름을 제공하지 않거나, 커널 인수를 통해 정적으로 설정하거나 다른 방법을 통해 설정하면 역방향 DNS 조회를 통해 얻을 수 있습니다. 역방향 DNS 조회는 노드에서 네트워크를 초기화한 후 수행되며 확인하는 데 시간이 걸릴 수 있습니다. 다른 시스템 서비스는 이 보다 먼저 시작하여 호스트 이름을 localhost 등으로 감지할 수 있습니다. DHCP를 사용하여 각 클러스터 노드의 호스트 이름을 제공하여 이 문제를 방지할 수 있습니다.

또한 DHCP를 통해 호스트 이름을 설정하면 DNS 분할 수평 구현 환경에서 수동으로 DNS 레코드 이름 구성 오류를 무시할 수 있습니다.

18.8.6.4.2. 네트워크 연결 요구사항

OpenShift Container Platform 클러스터 구성 요소가 통신할 수 있도록 시스템 간 네트워크 연결을 구성해야 합니다. 각 시스템에서 클러스터에 있는 다른 모든 시스템의 호스트 이름을 확인할 수 있어야 합니다.

이 섹션에서는 필요한 포트에 대해 자세히 설명합니다.



중요

연결된 OpenShift Container Platform 환경에서 모든 노드는 플랫폼 컨테이너의 이미지를 가져오고 Red Hat에 원격 측정 데이터를 제공하기 위해 인터넷에 액세스할 수 있어야 합니다.

표 18.78. 모든 시스템 간 통신에 사용되는 포트

프로토콜	포트	설명
ICMP	해당 없음	네트워크 연결성 테스트
TCP	**1936**	메트릭
	9000-9999	**9100-9101** 포트의 노드 내보내기 및 **9099** 포트의 Cluster Version Operator를 포함한 호스트 수준 서비스.
	10250-10259	Kubernetes에서 예약하는 기본 포트
	10256	openshift-sdn
UDP	**4789**	VXLAN 및 Geneve
	6081	VXLAN 및 Geneve
	9000-9999	**9100-9101** 포트의 노드 내보내기를 포함한 호스트 수준 서비스.
	500	IPsec IKE 패킷
	4500	IPsec NAT-T 패킷
TCP/UDP	**30000-32767**	Kubernetes 노드 포트
ESP	해당 없음	IPsec Encapsulating Security Payload (ESP)

표 18.79. 모든 시스템과 컨트롤 플레인 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**6443**	Kubernetes API

표 18.80. 컨트롤 플레인 머신 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**2379-2380**	etcd 서버 및 피어 포트

사용자 프로비저닝 인프라에 대한 NTP 구성

OpenShift Container Platform 클러스터는 기본적으로 공용 **NTP(Network Time Protocol)** 서버를 사용하도록 구성되어 있습니다. 로컬 엔터프라이즈 **NTP** 서버를 사용하거나 클러스터가 연결이 끊긴 네트워크에 배포되는 경우 특정 시간 서버를 사용하도록 클러스터를 구성할 수 있습니다. 자세한 내용은 ***chrony** 타임 서비스 설정* 문서를 참조하십시오.

DHCP 서버가 **NTP** 서버 정보를 제공하는 경우 **RHCOS(Red Hat Enterprise Linux CoreOS)** 시스템의 **chrony** 타임 서비스에서 정보를 읽고 **NTP** 서버와 클럭을 동기화할 수 있습니다.

18.8.6.5. 사용자 프로비저닝 DNS 요구사항

OpenShift Container Platform 배포의 경우 다음 구성 요소에 **DNS** 이름을 확인해야 합니다.

- **Kubernetes API**
- **OpenShift Container Platform** 애플리케이션 와일드카드
- 부트스트랩, 컨트롤 플레인 및 컴퓨팅 시스템

Kubernetes API, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템에 대한 역방향 **DNS** 확인이 필요합니다.

DNS A/AAAA 또는 **CNAME** 레코드는 이름 확인에 사용되며 **PTR** 레코드는 역방향 이름 확인에 사용됩니다. **RHCOS (Red Hat Enterprise Linux CoreOS)**는 **DHCP**에서 호스트 이름을 제공하지 않는 한 모든 노드의 호스트 이름을 설정할 때 역방향 레코드를 사용하기 때문에 역방향 레코드가 중요합니다. 또한 역방향 레코드는 **OpenShift Container Platform**이 작동하는 데 필요한 인증서 서명 요청 **(CSR)**을 생성하는 데 사용됩니다.

참고

DHCP 서버를 사용하여 각 클러스터 노드에 호스트 이름을 제공하는 것이 좋습니다. 자세한 내용은 *사용자 프로비저닝 인프라 섹션에 대한 DHCP 권장 사항* 섹션을 참조하십시오.

사용자가 프로비저닝한 OpenShift Container Platform 클러스터에 대해 다음 DNS 레코드가 필요하며 설치 전에 있어야 합니다. 각 레코드에서 <cluster_name>은 클러스터 이름이고 <base_domain>은 install-config.yaml 파일에서 지정하는 기반 도메인입니다. 전체 DNS 레코드는 <component>.<cluster_name>.<base_domain> 형식입니다.

표 18.81. 필수 DNS 레코드

구성 요소	레코드	설명
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API 로드 밸런서를 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.
	api-int.<cluster_name>.<base_domain>.	내부적으로 API 로드 밸런서를 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.  **중요** API 서버는 Kubernetes에 기록된 호스트 이름으로 작업자 노드를 확인할 수 있어야 합니다. API 서버가 노드 이름을 확인할 수 없는 경우 프록시된 API 호출이 실패할 수 있으며 pod에서 로그를 검색할 수 없습니다.
라우트	***.apps.<cluster_name>.<base_domain>.**	애플리케이션 인그레스 로드 밸런서를 참조하는 와일드카드 DNS A/AAA 또는 CNAME 레코드입니다. 애플리케이션 인그레스 로드 밸런서는 Ingress 컨트롤러 Pod를 실행하는 머신을 대상으로 합니다. Ingress 컨트롤러 Pod는 기본적으로 컴퓨팅 머신에서 실행됩니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다. 예를 들어 **console-openshift-console.apps.<cluster_name>.<base_domain>**은 OpenShift Container Platform 콘솔의 와일드카드 경로로 사용됩니다.
부트스트랩 시스템	**bootstrap.<cluster_name>.<base_domain>.**	부트스트랩 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.

구성 요소	레코드	설명
컨트롤 플레인 머신	**<master><n>. <cluster_name>. <base_domain>.**	컨트롤 플레인 노드의 각 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.
컴퓨팅 머신	**<worker><n>. <cluster_name>. <base_domain>.**	작업자 노드의 각 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.

참고

OpenShift Container Platform 4.4 이상에서는 DNS 구성에서 etcd 호스트 및 SRV 레코드를 지정할 필요가 없습니다.

작은 정보

dig 명령을 사용하여 이름과 역방향 이름을 확인할 수 있습니다. 자세한 검증 단계는 *사용자 프로비저닝 인프라의 DNS 확인* 섹션을 참조하십시오.

18.8.6.5.1. 사용자 프로비저닝 클러스터의 DNS 구성 예

이 섹션에서는 사용자 프로비저닝 인프라에 OpenShift Container Platform을 배포하기 위한 DNS 요구 사항을 충족하는 A 및 PTR 레코드 구성 샘플을 제공합니다. 샘플은 하나의 DNS 솔루션을 선택하기 위한 조언을 제공하기 위한 것이 아닙니다.

이 예제에서 클러스터 이름은 ocp4이고 기본 도메인은 example.com입니다.

사용자 프로비저닝 클러스터의 DNS A 레코드 구성 예

다음 BIND 영역 파일의 예제에서는 사용자가 프로비저닝한 클러스터의 이름 확인을 위한 샘플 A 레코드를 보여줍니다.

예 18.19. 샘플 DNS 영역 데이터베이스

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
```

```
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
master0.ocp4.example.com. IN A 192.168.1.97 5
master1.ocp4.example.com. IN A 192.168.1.98 6
master2.ocp4.example.com. IN A 192.168.1.99 7
;
worker0.ocp4.example.com. IN A 192.168.1.11 8
worker1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1

Kubernetes API의 이름 확인을 제공합니다. 레코드는 **API** 로드 밸런서의 **IP** 주소를 나타냅니다.

2

Kubernetes API의 이름 확인을 제공합니다. 레코드는 **API** 로드 밸런서의 **IP** 주소를 참조하며 내부 클러스터 통신에 사용됩니다.

3

와일드카드 경로의 이름 확인을 제공합니다. 레코드는 애플리케이션 인그레스 로드 밸런서의 **IP** 주소를 나타냅니다. 애플리케이션 인그레스 로드 밸런서는 **Ingress** 컨트롤러 **Pod**를 실행하는 머신을 대상으로 합니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

참고

이 예제에서는 **Kubernetes API** 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 **API** 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

4

부트스트랩 시스템의 이름 확인을 제공합니다.

5 6 7

컨트롤 플레인 시스템의 이름 확인을 제공합니다.

8 9

컴퓨팅 시스템의 이름 확인을 제공합니다.

사용자 프로비저닝 클러스터의 **DNS PTR** 레코드 구성 예

다음 예제 **BIND** 영역 파일은 사용자 프로비저닝 클러스터의 역방향 이름 확인을 위한 샘플 **PTR** 레코드를 보여줍니다.

예 18.20. 역방향 레코드의 샘플 DNS 영역 데이터베이스

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR master0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR master1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR master2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 7
```

```
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com. 8
;
;EOF
```

1

Kubernetes API의 역방향 DNS 확인을 제공합니다. PTR 레코드는 API 로드 밸런서의 레코드 이름을 참조합니다.

2

Kubernetes API의 역방향 DNS 확인을 제공합니다. PTR 레코드는 API 로드 밸런서의 레코드 이름을 참조하며 내부 클러스터 통신에 사용됩니다.

3

부트스트랩 시스템의 역방향 DNS 확인을 제공합니다.

4 5 6

컨트롤 플레인 시스템의 역방향 DNS 확인을 제공합니다.

7 8

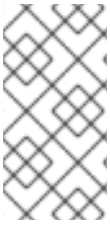
컴퓨팅 시스템의 역방향 DNS 확인을 제공합니다.

참고

OpenShift Container Platform 애플리케이션 와일드카드에는 PTR 레코드가 필요하지 않습니다.

18.8.6.6. 사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구사항

OpenShift Container Platform을 설치하기 전에 API 및 애플리케이션 인그레스 로드 밸런싱 인프라를 프로비저닝해야 합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 API 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.



참고

RHEL(Red Hat Enterprise Linux) 인스턴스를 사용하여 API 및 애플리케이션 인그레스 로드 밸런서를 배포하려면 RHEL 서브스크립션을 별도로 구매해야 합니다.

로드 밸런서 인프라는 다음 요구 사항을 충족해야 합니다.

1. **API 로드 밸런서: 플랫폼과 상호 작용하고 플랫폼을 구성할 수 있도록 사용자(인간과 시스템 모두)에게 공통 끝점을 제공합니다. 다음 조건을 설정합니다.**

   - **Layer 4 로드 밸런싱 전용입니다. 이를 Raw TCP, SSL Passthrough 또는 SSL Bridge 모드라고 합니다. SSL Bridge 모드를 사용하는 경우, API 경로에 대해 SNI(Server Name Indication, 서버 이름 표시)를 활성화해야 합니다.**

   - **스테이트리스 로드 밸런싱 알고리즘입니다. 옵션은 로드 밸런서 구현에 따라 달라집니다.**

   참고

   API 로드 밸런서가 제대로 작동하기 위해 세션 지속성이 필요하지 않습니다.

   로드 밸런서의 전면과 후면 모두에서 다음 포트를 구성하십시오.

표 18.82. API 로드 밸런서

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
6443	부트스트랩 및 컨트롤 플레인. 부트스트랩 시스템이 클러스터 컨트롤 플레인을 초기화한 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다. API 서버 상태 검사 프로브에 대한 **/readyz** 끝점을 구성해야 합니다.	X	X	Kubernetes API 서버
22623	부트스트랩 및 컨트롤 플레인. 부트스트랩 시스템이 클러스터 컨트롤 플레인을 초기화한 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다.	X		시스템 구성 서버

참고

API 서버가 /readyz 엔드포인트를 해제하는 시점부터 풀에서 API 서버 인스턴스가 제거되는 시점까지 시간이 30초를 넘지 않도록 로드 밸런서를 구성해야 합니다. /readyz가 오류를 반환하거나 정상 상태가 된 후 정해진 시간 안에 끝점이 제거 또는 추가되어야 합니다. 5초 또는 10초의 프로빙 주기(두 번의 성공적인 요청은 정상 상태, 세 번의 요청은 비정상 상태)는 충분한 테스트를 거친 값입니다.

2.

애플리케이션 인그레스 로드 밸런서: 클러스터 외부에서 유입되는 애플리케이션 트래픽에 대한 인그래스 포인트를 제공합니다. 다음 조건을 설정합니다.

- Layer 4 로드 밸런싱 전용입니다. 이를 Raw TCP, SSL Passthrough 또는 SSL Bridge 모드라고 합니다. SSL Bridge 모드를 사용하는 경우 인그레스 경로에 대해 SNI(Server Name Indication, 서버 이름 표시)를 활성화해야 합니다.

- 사용 가능한 옵션과 플랫폼에서 호스팅되는 애플리케이션 유형에 따라 연결 기반 또는 세션 기반 지속성이 권장됩니다.

작은 정보

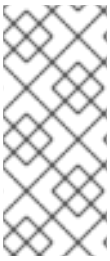
애플리케이션 인그레스 로드 밸런서에서 클라이언트의 실제 IP 주소를 확인할 수 있는 경우 소스 IP 기반 세션 지속성을 활성화하면 엔드 투 엔드 TLS 암호화를 사용하는 애플리케이션의 성능을 향상시킬 수 있습니다.

로드 밸런서의 전면과 후면 모두에서 다음 포트를 구성하십시오.

표 18.83. 애플리케이션 인그레스 로드 밸런서

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
443	기본적으로 인그레스 컨트롤러 pod, 컴퓨팅 또는 작업자를 실행하는 시스템입니다.	X	X	HTTPS 트래픽
80	기본적으로 인그레스 컨트롤러 pod, 컴퓨팅 또는 작업자를 실행하는 시스템입니다.	X	X	HTTP 트래픽

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
1936	기본적으로 Ingress 컨트롤러 Pod를 실행하는 작업자 노드입니다. 수신 상태 점검 프로브에 대해 **/healthz/ready** 끝점을 구성해야 합니다.	X	X	HTTP 트래픽



참고

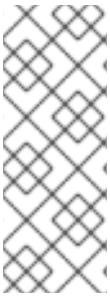
컴퓨팅 노드가 0인 3-노드 클러스터를 배포하는 경우 Ingress 컨트롤러 Pod는 컨트롤 플레인 노드에서 실행됩니다. 3-노드 클러스터 배포에서 HTTP 및 HTTPS 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다.

참고

인그레스 라우터에 대한 작업 구성이 OpenShift Container Platform 클러스터에 필요합니다. 컨트롤 플레인 초기화 후 인그레스 라우터를 설정해야 합니다.

18.8.6.6.1. 사용자 프로비저닝 클러스터의 로드 밸런서 구성 예

이 섹션에서는 사용자 프로비저닝 클러스터의 로드 밸런싱 요구 사항을 충족하는 API 및 애플리케이션 수신 로드 밸런서 구성 예를 제공합니다. 샘플은 HAProxy 로드 밸런서에 대한 **/etc/haproxy/haproxy.cfg** 구성입니다. 이 예제에서는 하나의 로드 밸런싱 솔루션을 선택하기 위한 조언을 제공하는 것을 목적으로 하지 않습니다.



참고

이 예제에서는 Kubernetes API 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 API 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

예 18.21. API 및 애플리케이션 인그레스 로드 밸런서 구성 샘플

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
```

```
    option                  redispatch
    retries                 3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s
    maxconn                 3000
  frontend stats
    bind *:1936
    mode            http
    log             global
    maxconn 10
    stats enable
    stats hide-version
    stats refresh 30s
    stats show-node
    stats show-desc Stats for ocp4 cluster (1)
    stats auth admin:ocp4
    stats uri /stats
  listen api-server-6443 (2)
    bind *:6443
    mode tcp
    server bootstrap bootstrap.ocp4.example.com:6443 check inter 1s backup (3)
    server master0 master0.ocp4.example.com:6443 check inter 1s
    server master1 master1.ocp4.example.com:6443 check inter 1s
    server master2 master2.ocp4.example.com:6443 check inter 1s
  listen machine-config-server-22623 (4)
    bind *:22623
    mode tcp
    server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup (5)
    server master0 master0.ocp4.example.com:22623 check inter 1s
    server master1 master1.ocp4.example.com:22623 check inter 1s
    server master2 master2.ocp4.example.com:22623 check inter 1s
  listen ingress-router-443 (6)
    bind *:443
    mode tcp
    balance source
    server worker0 worker0.ocp4.example.com:443 check inter 1s
    server worker1 worker1.ocp4.example.com:443 check inter 1s
  listen ingress-router-80 (7)
    bind *:80
    mode tcp
    balance source
    server worker0 worker0.ocp4.example.com:80 check inter 1s
    server worker1 worker1.ocp4.example.com:80 check inter 1s
```

(1)

이 예에서 클러스터 이름은 **ocp4** 입니다.

(2)

3 5

부트스트랩 항목은 **OpenShift Container Platform** 클러스터 설치 전에 있어야 하며 부트스트랩 프로세스가 완료된 후 제거해야 합니다.

4

포트 **22623**은 머신 구성 서버 트래픽을 처리하고 컨트롤 플레인 시스템을 가리킵니다.

6

포트 **443**은 **HTTPS** 트래픽을 처리하고 **Ingress** 컨트롤러 **Pod**를 실행하는 시스템을 가리킵니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

7

포트 **80**은 **HTTP** 트래픽을 처리하고 **Ingress** 컨트롤러 **Pod**를 실행하는 머신을 가리킵니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

참고

컴퓨팅 노드가 **0**인 **3-**노드 클러스터를 배포하는 경우 **Ingress** 컨트롤러 **Pod**는 컨트롤 플레인 노드에서 실행됩니다. **3-**노드 클러스터 배포에서 **HTTP** 및 **HTTPS** 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다.

작은 정보

HAProxy를 로드 밸런서로 사용하는 경우 **HAProxy** 노드에서 **netstat -nltupe**를 실행하여 **haproxy** 프로세스가 포트 **6443, 22623, 443** 및 **80**에서 수신 대기 중인지 확인할 수 있습니다.

참고

HAProxy를 로드 밸런서로 사용하고 **SELinux**가 **enforcing**으로 설정된 경우 **HAProxy** 서비스가 **setsebool -P haproxy_connect_any=1**을 실행하여 구성된 **TCP** 포트에 바인딩할 수 있는지 확인해야 합니다.

18.8.7. 사용자 프로비저닝 인프라 준비

사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치하기 전에 기본 인프라를 준비해야 합니다.

이 섹션에서는 OpenShift Container Platform 설치를 준비하기 위해 클러스터 인프라를 설정하는 데 필요한 높은 수준의 단계에 대해 자세히 설명합니다. 여기에는 클러스터 노드에 대한 IP 네트워킹 및 네트워크 연결 구성, 방화벽을 통해 필요한 포트 활성화, 필수 DNS 및 로드 밸런싱 인프라를 설정하는 작업이 포함됩니다.

준비 후 클러스터 인프라는 *사용자 프로비저닝 인프라가 있는 클러스터의 요구 사항* 섹션에 설명된 요구 사항을 충족해야 합니다.

사전 요구 사항

- OpenShift Container Platform 4.x 테스트된 통합 페이지를 검토했습니다.
- *사용자 프로비저닝 인프라가 있는 클러스터의 요구 사항* 섹션에 자세히 설명된 인프라 요구 사항을 검토했습니다.

프로세스

1. DHCP를 사용하여 클러스터 노드에 IP 네트워킹 구성을 제공하는 경우 DHCP 서비스를 구성합니다.
   a. 노드의 영구 IP 주소를 DHCP 서버 구성에 추가합니다. 구성에서 관련 네트워크 인터페이스의 MAC 주소를 각 노드의 의도한 IP 주소와 일치시킵니다.
   b. DHCP를 사용하여 클러스터 시스템의 IP 주소 지정을 구성하는 경우 시스템은 DHCP를 통해 DNS 서버 정보도 가져옵니다. DHCP 서버 구성을 통해 클러스터 노드에서 사용하는 영구 DNS 서버 주소를 정의합니다.

참고

DHCP 서비스를 사용하지 않는 경우 RHCOS 설치 시 IP 네트워킹 구성과 DNS 서버의 주소를 노드에 제공해야 합니다. ISO 이미지에서 설치하는 경우 부팅 인수로 전달할 수 있습니다. 고정 IP 프로비저닝 *및 고급 네트워킹 옵션에 대한 자세한 내용은 RHCOS 설치 및 OpenShift Container Platform 부트스트랩 프로세스 시작* 섹션을 참조하십시오.

c.

DHCP 서버 구성에 클러스터 노드의 호스트 이름을 정의합니다. 호스트 이름 고려 사항에 대한 자세한 내용은 ***DHCP****를 통해 클러스터 노드 호스트 이름 설정* 섹션을 참조하십시오.

참고

DHCP 서비스를 사용하지 않는 경우 클러스터 노드는 역방향 **DNS** 조회를 통해 호스트 이름을 가져옵니다.

2.

네트워크 인프라가 클러스터 구성 요소 간 필수 네트워크 연결을 제공하는지 확인합니다. 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 인프라 섹션의 네트워킹 요구 사항* 섹션을 참조하십시오.

3.

OpenShift Container Platform 클러스터 구성 요소가 통신하는 데 필요한 포트를 활성화하도록 방화벽을 구성합니다. 필요한 포트에 대한 자세한 내용은 *사용자 프로비저닝 인프라 섹션의 네트워킹 요구 사항* 섹션을 참조하십시오.

4.

클러스터에 필요한 **DNS** 인프라를 설정합니다.

a.

Kubernetes API, 애플리케이션 와일드카드, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템의 **DNS** 이름 확인을 구성합니다.

b.

Kubernetes API, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템에 대한 역방향 **DNS** 확인을 구성합니다.

OpenShift Container Platform DNS 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 **DNS** 요구 사항* 섹션을 참조하십시오.

5.

DNS 구성을 확인합니다.

a.

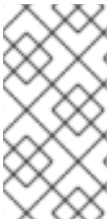
설치 노드에서 **Kubernetes API**의 레코드 이름, 와일드카드 경로 및 클러스터 노드에 대해 **DNS** 조회를 실행합니다. 응답의 **IP** 주소가 올바른 구성 요소에 해당하는지 확인합니다.

b.

설치 노드에서 로드 밸런서 및 클러스터 노드의 **IP** 주소에 대해 역방향 **DNS** 조회를 실행합니다. 응답의 레코드 이름이 올바른 구성 요소에 해당하는지 확인합니다.

자세한 DNS 검증 단계는 *사용자 프로비저닝 인프라에 대한 DNS 확인* 섹션을 참조하십시오.

6. 필요한 API 및 애플리케이션 수신 로드 밸런싱 인프라를 프로비저닝합니다. 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구 사항* 섹션을 참조하십시오.

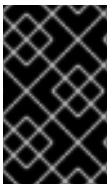


참고

일부 로드 밸런싱 솔루션에는 로드 밸런싱을 초기화하기 전에 클러스터 노드의 DNS 이름을 확인해야 합니다.

18.8.8. 사용자 프로비저닝 인프라에 대한 DNS 확인 검증

사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치하기 전에 DNS 구성을 확인할 수 있습니다.



중요

클러스터를 설치하기 전에 이 섹션에 설명된 검증 단계를 성공해야 합니다.

사전 요구 사항

- 사용자 프로비저닝 인프라에 필요한 DNS 레코드를 구성했습니다.

프로세스

1. 설치 노드에서 Kubernetes API의 레코드 이름, 와일드카드 경로 및 클러스터 노드에 대해 DNS 조회를 실행합니다. 응답에 포함된 IP 주소가 올바른 구성 요소에 해당하는지 확인합니다.

   a. Kubernetes API 레코드 이름을 조회합니다. 결과가 API 로드 밸런서의 IP 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
```

1 <nameserver_ip>를 네임서버의 IP 주소로, <cluster_name>을 클러스터 이름으로, <base_domain>을 기본 도메인 이름으로 바꿉니다.

출력 예

```
api.ocp4.example.com.  0 IN A 192.168.1.5
```

b.

Kubernetes 내부 API 레코드 이름을 조회합니다. 결과가 API 로드 밸런서의 IP 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
```

출력 예

```
api-int.ocp4.example.com.  0 IN A 192.168.1.5
```

c.

예제 *.apps.<cluster_name>.<base_domain>을 테스트합니다. DNS 와일드카드를 조회합니다. 모든 애플리케이션 와일드카드 조회는 애플리케이션 인그레스 로드 밸런서의 IP 주소로 확인되어야 합니다.

```
$ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.
<base_domain>
```

출력 예

```
random.apps.ocp4.example.com.  0 IN A 192.168.1.5
```

참고

예제 출력에서는 **Kubernetes API** 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 **API** 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

random 항목을 다른 와일드카드 값으로 교체할 수 있습니다. 예를 들어 **OpenShift Container Platform** 콘솔의 경로를 쿼리할 수 있습니다.

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

출력 예

```
console-openshift-console.apps.ocp4.example.com. 0 IN A 192.168.1.5
```

d. 부트스트랩 **DNS** 레코드 이름에 대해 조회를 실행합니다. 결과가 부트스트랩 노드의 **IP** 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.
<base_domain>
```

출력 예

```
bootstrap.ocp4.example.com.  0 IN A 192.168.1.96
```

e. 이 방법을 사용하여 컨트롤 플레인 및 컴퓨팅 노드의 **DNS** 레코드 이름에 대해 조회를 수행합니다. 결과가 각 노드의 **IP** 주소에 해당하는지 확인합니다.

2. 설치 노드에서 로드 밸런서 및 클러스터 노드의 **IP** 주소에 대해 역방향 **DNS** 조회를 실행합니다. 응답에 포함된 레코드 이름이 올바른 구성 요소에 해당하는지 확인합니다.

a.

API 로드 밸런서의 IP 주소에 대해 역방향 조회를 수행합니다. 응답에 Kubernetes API 및 Kubernetes 내부 API의 레코드 이름이 포함되어 있는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

출력 예

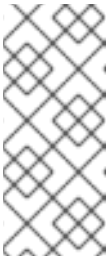
```
5.1.168.192.in-addr.arpa. 0 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 0 IN PTR api.ocp4.example.com. 2
```

1

Kubernetes 내부 API의 레코드 이름을 제공합니다.

2

Kubernetes API의 레코드 이름을 제공합니다.



참고

OpenShift Container Platform 애플리케이션 와일드카드에는 PTR 레코드가 필요하지 않습니다. 애플리케이션 인그레스 로드 밸런서의 IP 주소에 대한 역방향 DNS 확인에는 유효성 검사 단계가 필요하지 않습니다.

b.

부트스트랩 노드의 IP 주소에 대해 역방향 조회를 수행합니다. 결과가 부트스트랩 노드의 DNS 레코드 이름을 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

출력 예

```
96.1.168.192.in-addr.arpa. 0 IN PTR bootstrap.ocp4.example.com.
```

c.
이 방법을 사용하여 컨트롤 플레인 및 컴퓨팅 노드의 IP 주소에 대해 역방향 조회를 수행합니다. 결과가 각 노드의 DNS 레코드 이름과 일치하는지 확인합니다.

18.8.9. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

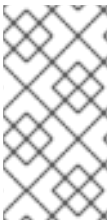
OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 core 사용자의 ~/.ssh/authorized_keys 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 core로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. ./openshift-install gather 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.

중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.



참고

AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

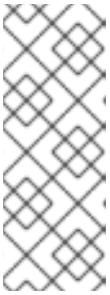
프로세스

1.
로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1

새 SSH 키의 경로 및 파일 이름(예: ~/.ssh/id_ed25519)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 '~/.ssh 디렉터리에 있는지 확인하십시오.



참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 ed25519 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 rsa 또는 ecdsa 알고리즘을 사용하는 키를 생성합니다.

2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 ~/.ssh/id_ed25519.pub 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 ./openshift-install gather 명령을 사용하려는 경우 필요합니다.

참고

일부 배포에서는 ~/.ssh/id_rsa 및 ~/.ssh/id_dsa와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. ssh-agent 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 **FIPS** 모드인 경우 **FIPS** 호환 알고리즘만 사용하여 **SSH** 키를 생성합니다. 키는 **RSA** 또는 **ECDSA**여야 합니다.

4. **ssh-agent**에 **SSH** 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 **SSH** 개인 키의 경로 및 파일 이름을 지정합니다(예: ~/.ssh/id_ed25519).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- **OpenShift Container Platform**을 설치할 때 **SSH** 공개 키를 설치 프로그램에 지정합니다. 프로비저닝하는 인프라에 클러스터를 설치하는 경우 설치 프로그램에 키를 제공해야 합니다.

18.8.10. 수동으로 설치 구성 파일 만들기

OpenShift Container Platform을 사용자가 프로비저닝한 설치의 경우 설치 구성 파일을 수동으로 생성합니다.

사전 요구 사항

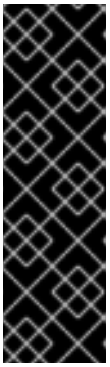
- 로컬 시스템에 설치 프로그램에 제공할 **SSH** 공개 키가 있습니다. 키는 디버깅 및 재해 복구를 위해 클러스터 노드에 대한 **SSH** 인증에 사용됩니다.

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿이 있습니다.
- 명령 출력에서 imageContentSources 섹션을 가져와서 리포지토리를 미러링합니다.
- 미러 레지스트리에 대한 인증서의 내용을 가져옵니다.

프로세스

1. 필요한 설치 자산을 저장할 설치 디렉터리를 만듭니다.

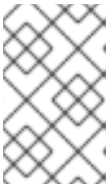
```
$ mkdir <installation_directory>
```



중요

디렉터리를 만들어야 합니다. 부트스트랩 X.509 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 OpenShift Container Platform 버전에서 설치 파일을 복사할 때는 주의하십시오.

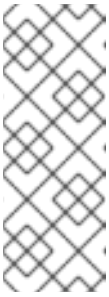
2. 샘플 install-config.yaml 파일 템플릿을 사용자 지정하여 <installation_directory>에 저장합니다.



참고

이 설정 파일의 이름을 install-config.yaml로 지정해야 합니다.

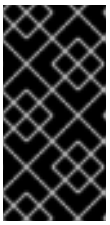
- docker.io와 같이 RHCOS가 기본적으로 신뢰하는 레지스트리를 사용하는 경우를 제외하고, additionalTrustBundle 섹션에 있는 미러 리포지토리에 대한 인증서 내용을 제공해야 합니다. 대부분의 경우 미러 인증서를 제공해야 합니다.
- 리포지토리를 미러링하려면 명령 출력의 imageContentSources 섹션을 삽입해야 합니다.



참고

일부 플랫폼 유형의 경우 대체로 **./openshift-install create install-config --dir <installation_directory>** 를 실행하여 **install-config.yaml** 파일을 생성할 수 있습니다. 프롬프트에서 클러스터 구성에 대한 세부 정보를 제공할 수 있습니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.



중요

install-config.yaml 파일은 설치 과정의 다음 단계에서 사용됩니다. 이 시점에서 이를 백업해야 합니다.

18.8.10.1. VMware vSphere용 샘플 install-config.yaml 파일

install-config.yaml 파일을 사용자 지정하여 **OpenShift Container Platform** 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.

```
apiVersion: v1
baseDomain: example.com 1
compute:
- hyperthreading: Enabled 2 3
  name: worker
  replicas: 0 4
controlPlane:
  hyperthreading: Enabled 5 6
  name: master
  replicas: 3 7
metadata:
  name: test 8
platform:
  vsphere:
    vcenter: your.vcenter.server 9
    username: username 10
    password: password 11
    datacenter: datacenter 12
    defaultDatastore: datastore 13
    folder: "/<datacenter_name>/vm/<folder_name>/<subfolder_name>" 14
fips: false 15
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email":
"you@example.com"}}}' 16
sshKey: 'ssh-ed25519 AAAA...' 17
additionalTrustBundle: | 18
```

```
  -----BEGIN CERTIFICATE-----
  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
imageContentSources: 19
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

1

클러스터의 기본 도메인입니다. 모든 **DNS** 레코드는 이 기본 도메인의 하위 도메인이어야 하며 클러스터 이름을 포함해야 합니다.

2 **5**

controlPlane 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 현재 두 섹션이 모두 단일 시스템 풀을 정의하지만 향후 출시되는 **OpenShift Container Platform** 버전은 설치 과정에서 여러 컴퓨팅 풀 정의를 지원할 수 있습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

3 **6**

동시 멀티스레딩 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 동시 멀티스레딩이 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. 일부 클러스터 시스템에서 동시 멀티스레딩을 비활성화할 경우에는 해당 멀티스레딩을 모든 클러스터 시스템에서 비활성화해야 합니다.

중요

동시 멀티스레딩을 비활성화하는 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다. 동시 멀티스레딩을 비활성화하는 경우 시스템은 **8**개 이상의 **CPU**와 **32GB**의 **RAM**을 사용해야 합니다.

4

replicas 매개변수의 값을 **0**으로 설정해야 합니다. 이 매개변수는 클러스터를 생성하고 관리하는 작업자 수를 제어합니다. 이는 사용자 프로비저닝 인프라를 사용할 때 클러스터가 실행하지 않는 기능입니다. **OpenShift Container Platform** 설치를 완료하기 전에 클러스터에서 사용할 작업자 시스템을 수동으로 배포해야 합니다.

7

클러스터에 추가하는 컨트롤 플레인 시스템의 수입니다. 클러스터에서 이 값을 클러스터의 **etcd** 끝점 수로 사용하므로 이 값은 배포하는 컨트롤 플레인 시스템의 수와 일치해야 합니다.

8

DNS 레코드에 지정한 클러스터 이름입니다.

9

vCenter 서버의 정규화된 호스트 이름 또는 **IP** 주소입니다.

10

서버에 액세스하기 위한 사용자의 이름입니다. 이 사용자에게는 최소한 **vSphere**에서 정적 또는 동적 영구 볼륨 프로비저닝 에 필요한 역할과 권한이 있어야 합니다.

11

vSphere 사용자와 연결된 암호입니다.

12

vSphere 데이터 센터입니다.

13

사용할 기본 **vSphere** 데이터 저장소입니다.

14

선택사항: 설치 관리자 프로비저닝 인프라의 경우 설치 프로그램이 가상 머신을 생성하는 기존 폴더의 절대 경로(예: /<datacenter_name>/vm/<folder_name>/<subfolder_name>). 이 값을 제공하지 않으면 설치 프로그램이 인프라 **ID**로 이름이 지정된 데이터 센터 가상 머신 폴더에 최상위 폴더를 만듭니다. 클러스터에 인프라를 제공하는 경우에는 이 매개변수를 생략합니다.

15

FIPS 모드 활성화 또는 비활성화 여부입니다. 기본적으로 **FIPS** 모드는 비활성화됩니다. **FIPS** 모드가 활성화되면 **OpenShift Container Platform**이 실행되는 **RHCOS(Red Hat Enterprise Linux CoreOS)** 시스템에서 기본 **Kubernetes** 암호화 제품군은 우회하고 **RHCOS**와 함께 제공되는 암호화 모듈을 대신 사용합니다.

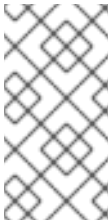
중요

FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 **x86_64** 아키텍처의 **OpenShift Container Platform** 배포에서만 지원됩니다.

16

17

RHCOS(Red Hat Enterprise Linux CoreOS)의 core 사용자에 대한 기본 SSH 키의 공용 부분입니다.



참고

설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 OpenShift Container Platform 클러스터의 경우 ssh-agent 프로세스가 사용하는 SSH 키를 지정합니다.

18

미러 레지스트리에 사용한 인증서 파일의 내용을 제공하십시오.

19

명령 출력에서 imageContentSources 섹션을 제공하여 리포지토리를 미러링하십시오.

18.8.10.2. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 HTTP 또는 HTTPS 프록시를 사용할 수 있습니다. install-config.yaml 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 OpenShift Container Platform 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 install-config.yaml 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 API에 대한 호출을 포함하여 모든 클러스터 발신(Egress) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 Proxy 오브젝트의 spec.noProxy 필드에 사이트를 추가했습니다.

참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, **networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 Red Hat OpenStack Platform (RHOSP)에 설치하는 경우 Proxy 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. 클러스터 외부에서 **HTTP** 연결을 구축하는 데 사용할 프록시 **URL**입니다. **URL** 스키마는 **http**여야 합니다.

2. 클러스터 외부에서 **HTTPS** 연결을 구축하는 데 사용할 프록시 **URL**입니다.

3. 대상 도메인 이름, **IP** 주소 또는 프록시에서 제외할 기타 네트워크 **CIDR**로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. ***를 사용하여 모든 대상에 대해 프록시를 바이패스합니다. **vCenter**의 **IP** 주소와 해당 시스템에 사용하는 **IP** 범위를 포함해야 합니다.

4.

참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 **OpenShift Container Platform**을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 **nil**이 됩니다.

참고

cluster라는 **Proxy** 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

18.8.11. Kubernetes 매니페스트 및 Ignition 설정 파일 생성

일부 클러스터 정의 파일을 수정하고 클러스터 시스템을 수동으로 시작해야 하므로 클러스터가 시스템을 구성하는 데 필요한 **Kubernetes** 매니페스트 및 **Ignition** 구성 파일을 사용자가 생성해야 합니다.

설치 구성 파일은 **Kubernetes** 매니페스트로 변환됩니다. 매니페스트는 나중에 클러스터 머신을 구성하는 데 사용되는 **Ignition** 구성 파일로 래핑됩니다.

중요

- **OpenShift Container Platform** 설치 프로그램에서 생성하는 **Ignition** 구성 파일에 **24**시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 **24**시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 **kubelet** 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청**(CSR)**을 수동으로 승인해야 합니다. 자세한 내용은 *만료된 컨트롤 플레인 인증서에서 복구* 문서를 참조하십시오.

- 클러스터를 설치한 후 **24**시간에서 **22**시간까지의 인증서가 교체되기 때문에 생성된 후 **12**시간 이내에 **Ignition** 구성 파일을 사용하는 것이 좋습니다. **12**시간 이내에 **Ignition** 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

사전 요구 사항

- **OpenShift Container Platform** 설치 프로그램을 가져오셨습니다. 제한된 네트워크 설치의 경우, 해당 파일은 미러 호스트에 있습니다.

- **install-config.yaml** 설치 구성 파일을 생성하셨습니다.

프로세스

1. **OpenShift Container Platform** 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터에 대한 **Kubernetes** 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>**는 사용자가 만든 **install-config.yaml** 파일이 포함된 설치 디렉터리를 지정합니다.

2. 컨트롤 플레인 시스템 및 컴퓨팅 머신 세트를 정의하는 **Kubernetes** 매니페스트 파일을 제거합니다.

```
$ rm -f openshift/99_openshift-cluster-api_master-machines-*.yaml
openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

이러한 리소스는 사용자가 직접 생성하고 관리하기 때문에 초기화할 필요가 없습니다.

- 시스템 API로 머신 세트 파일을 보존하여 컴퓨팅 시스템을 생성할 수 있지만 사용자 환경과 일치하도록 해당 참조를 업데이트해야 합니다.

3. <installation_directory>/manifests/cluster-scheduler-02-config.yml Kubernetes 매니페스트 파일의 mastersSchedulable 매개변수가 false로 설정되어 있는지 확인합니다. 이 설정으로 인해 컨트롤 플레인 머신에서 포드가 예약되지 않습니다.

   a. <installation_directory>/manifests/cluster-scheduler-02-config.yml 파일을 엽니다.

   b. mastersSchedulable 매개변수를 찾아서 값을 False로 설정되어 있는지 확인합니다.

   c. 파일을 저장하고 종료합니다.

4. Ignition 구성 파일을 생성하려면 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행합니다.

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 <installation_directory>는 동일한 설치 디렉터리를 지정합니다.

설치 디렉터리의 부트스트랩, 컨트롤 플레인 및 컴퓨팅 노드에 대한 Ignition 구성 파일이 생성됩니다. kubeadmin-password 및 kubeconfig 파일은 ./<installation_directory>/auth 디렉터리에 생성됩니다.

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

18.8.12. 인프라 이름 추출

Ignition 구성 파일에는 AWS의 VMware Cloud에서 클러스터를 고유하게 식별하는 데 사용할 수 있는 고유한 클러스터 식별가 포함되어 있습니다. 클러스터 ID를 가상 머신 폴더의 이름으로 사용하려면 해당 ID를 추출해야 합니다.

사전 요구 사항

- OpenShift Container Platform 설치 프로그램과 클러스터의 풀 시크릿을 받으셨습니다.
- 클러스터에 대한 Ignition 구성 파일을 생성하셨습니다.
- jq CLI를 설치하셨습니다.

프로세스

- Ignition 구성 파일 메타데이터에서 인프라 이름을 추출하여 확인하려면 다음 명령을 실행하십시오.

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

출력 예

```
openshift-vw9j6 1
```

1 이 명령의 출력은 클러스터 이름과 임의의 문자열입니다.

18.8.13. RHCOS 설치 및 OpenShift Container Platform 부트스트랩 프로세스 시작

VMware vSphere의 사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치하려면 vSphere 호스트에 RHCOS(Red Hat Enterprise Linux Core OS)를 설치해야 합니다. RHCOS를 설치할 때 설치 중인 머신 유형에 대해 OpenShift Container Platform 설치 프로그램에서 생성한 Ignition 구성 파일을 제공해야 합니다. 적합한 네트워킹, DNS 및 로드 밸런싱 인프라를 구성한 경우 RHCOS 머신이 재부팅된 후 OpenShift Container Platform 부트스트랩 프로세스가 자동으로 시작됩니다.

사전 요구 사항

- 클러스터에 대한 Ignition 구성 파일이 있어야 합니다.
- 사용자 컴퓨터에서 액세스할 수 있고 생성한 시스템이 액세스할 수 있는 HTTP 서버에 액세스할 수 있습니다.
- vSphere 클러스터 를 생성했습니다.

프로세스

1. 설치 프로그램에서 생성된 부트스트랩 Ignition 구성 파일 (<installation_directory>/bootstrap.ign)을 HTTP 서버에 업로드합니다. 이 파일의 URL을 기록해 둡니다.
2. 부트스트랩 노드의 다음 보조 Ignition 구성 파일을 <installation_directory>/merge-bootstrap.ign으로 컴퓨터에 저장합니다.

```
{
  "ignition": {
    "config": {
      "merge": [
        {
          "source": "<bootstrap_ignition_config_url>", 1
          "verification": {}
        }
      ]
    },
    "timeouts": {},
    "version": "3.2.0"
  },
  "networkd": {},
  "passwd": {},
  "storage": {},
  "systemd": {}
}
```

1

호스팅한 부트스트랩 Ignition 구성 파일의 URL을 지정합니다.

부트스트랩 머신에 대한 VM(가상 머신)을 생성할 때 이 Ignition 구성 파일을 사용합니다.

3. 설치 프로그램이 생성한 다음 Ignition 구성 파일을 찾습니다.

- <installation_directory>/master.ign
- <installation_directory>/worker.ign
- <installation_directory>/merge-bootstrap.ign

4. Ignition 구성 파일을 Base64 인코딩으로 변환합니다. 이 절차의 뒷부분에서는 이러한 파일을 VM의 추가 구성 매개변수 guestinfo.ignition.config.data 에 추가해야 합니다.

예를 들어 Linux 운영 체제를 사용하는 경우 base64 명령을 사용하여 파일을 인코딩할 수 있습니다.

```
$ base64 -w0 <installation_directory>/master.ign > <installation_directory>/master.64
```

```
$ base64 -w0 <installation_directory>/worker.ign > <installation_directory>/worker.64
```

```
$ base64 -w0 <installation_directory>/merge-bootstrap.ign >
<installation_directory>/merge-bootstrap.64
```

중요

설치를 마친 후 클러스터에 컴퓨팅 시스템을 더 추가하려면 Ignition 구성 파일을 삭제하지 마십시오.

5. RHCOS OVA 이미지를 가져옵니다. RHCOS 이미지 미러 페이지에서 이미지를 사용할 수 있습니다.

중요

RHCOS 이미지는 **OpenShift Container Platform** 릴리스에 따라 변경되지 않을 수 있습니다. 설치하는 **OpenShift Container Platform** 버전과 같거나 그 이하의 버전 중 가장 최신 버전의 이미지를 다운로드해야 합니다. 지원되는 경우 **OpenShift Container Platform** 버전과 일치하는 이미지 버전을 사용합니다.

파일 이름에는 **rhcos-vmware.<architecture>.ova** 형식의 **OpenShift Container Platform** 버전 번호가 포함됩니다.

6. **vSphere Client**에서 **VM**을 저장할 데이터 센터 폴더를 생성합니다.

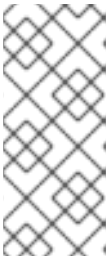
   a. **VMs and Templates** 보기를 클릭합니다.

   b. 데이터 센터 이름을 마우스 오른쪽 버튼으로 클릭합니다.

   c. **New Folder → New VM and Template Folder**를 클릭합니다.

   d. 표시되는 창에서 폴더 이름을 입력합니다. **install-config.yaml** 파일에서 기존 폴더를 지정하지 않은 경우 인프라 **ID**와 동일한 이름으로 폴더를 생성합니다. **vCenter**는 이 폴더 이름을 사용하여 **Workspace** 구성에 적절한 위치에 스토리지를 동적으로 프로비저닝합니다.

7. **vSphere Client**에서 **OVA** 이미지에 대한 템플릿을 생성한 다음 필요에 따라 템플릿을 복제합니다.



참고

다음 단계에서 템플릿을 생성한 다음 모든 클러스터 시스템에 대한 템플릿을 복제합니다. 그런 다음 **VM**을 프로비저닝할 때 복제된 머신 유형의 **Ignition** 구성 파일의 위치를 제공합니다.

   a. **Hosts and Clusters** 탭에서 클러스터 이름을 마우스 오른쪽 버튼으로 클릭하고 **Deploy OVF Template**을 선택합니다.

.

b. **Select an OVF** 탭에서 다운로드한 **RHCOS OVA** 파일의 이름을 지정합니다.

c. 이름 및 폴더 선택 탭에서 템플릿의 가상 시스템 이름 (예: **Template-RHCOS**)을 설정합니다. **vSphere** 클러스터의 이름을 클릭하고 이전 단계에서 생성한 폴더를 선택합니다.

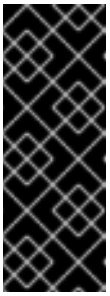
d. **Select a compute resource** 탭에서 **vSphere** 클러스터 이름을 클릭합니다.

e. **Select storage** 탭에서 **VM**의 스토리지 옵션을 구성합니다.

- 스토리지 기본 설정에 따라 **Thin Provision** 또는 **Thick Provision**을 선택합니다.
- **install-config.yaml** 파일에서 지정한 데이터 저장소를 선택합니다.

f. **Select network** 탭에서 사용 가능한 경우 클러스터에 대해 구성한 네트워크를 지정합니다.

g. **OVF** 템플릿을 생성할 때 템플릿 사용자 지정 탭에 값을 지정하지 않거나 템플릿을 추가로 구성하지 마십시오.



중요

원래 **VM** 템플릿을 시작하지 마십시오. **VM** 템플릿이 꺼져 있어야 하며 새 **RHCOS** 머신에 대해 복제해야 합니다. **VM** 템플릿을 시작하면 **VM** 템플릿이 플랫폼의 **VM**으로 구성되므로 시스템 세트가 구성을 적용할 수 있는 템플릿으로 사용되지 않습니다.

8. 선택 사항: 필요한 경우 **VM** 템플릿에서 구성된 가상 하드웨어 버전을 업데이트합니다. 자세한 내용은 가상 머신을 **VMware** 설명서의 최신 하드웨어 버전으로 업그레이드를 참조하십시오.

중요

필요한 경우 **VM** 템플릿의 하드웨어 버전을 버전 **15**로 업데이트하는 것이 좋습니다. 이제 **vSphere**에서 실행 중인 클러스터 노드에 하드웨어 버전 **13**을 사용하는 것이 더 이상 사용되지 않습니다. 가져온 템플릿의 기본값이 하드웨어 버전 **13**인 경우 **VM** 템플릿을 하드웨어 버전 **15**로 업그레이드하기 전에 **ESXi** 호스트가 **6.7U3** 이상인지 확인해야 합니다. **vSphere** 버전이 **6.7U3** 미만인 경우 이 업그레이드 단계를 건너뛸 수 있습니다. 그러나 향후 **OpenShift Container Platform** 버전은 **6.7U3** 미만의 하드웨어 버전 **13** 및 **vSphere** 버전에 대한 지원을 제거하도록 예약되어 있습니다.

9. 템플릿이 배포된 후 클러스터에서 시스템의 가상 머신을 배포합니다.

a. 템플릿 이름을 마우스 오른쪽 버튼으로 클릭하고 **Clone → Clone to Virtual Machine**을 클릭합니다.

b. **Select a name and folder** 탭에서 가상 머신의 이름을 지정합니다. **control-plane-0** 또는 **compute-1**과 같은 시스템 유형을 이름에 포함할 수 있습니다.

c. **Select a name and folder** 탭에서 클러스터에 대해 생성한 폴더의 이름을 선택합니다.

d. **Select a compute resource** 탭에서 데이터 센터의 호스트 이름을 선택합니다.

e. 선택사항: **Select storage** 탭에서 스토리지 옵션을 사용자 지정합니다.

f. **Select clone options**에서 **Customize this virtual machine's hardware**를 선택합니다.

g. **Customize hardware** 탭에서 **VM Options → Advanced**를 클릭합니다.

- 선택 사항: **vSphere**에서 기본 **DHCP** 네트워킹을 재정의합니다. 고정 **IP** 네트워킹을 활성화하려면 다음을 수행합니다.

  i. 고정 **IP** 구성을 설정합니다.

```
$ export IPCFG="ip=<ip>::<gateway>:<netmask>:<hostname>:
<iface>:none nameserver=srv1 [nameserver=srv2 [nameserver=srv3 [...]]]"
```

명령 예

```
$ export IPCFG="ip=192.168.100.101::192.168.100.254:255.255.255.0:::none
nameserver=8.8.8.8"
```

ii. guestinfo.afterburn.initrd.network-kargs 속성을 설정한 후 vSphere의 OVA에서 VM을 부팅합니다.

```
$ govc vm.change -vm "<vm_name>" -e
"guestinfo.afterburn.initrd.network-kargs=${IPCFG}"
```

- 선택사항: 클러스터 성능 문제가 발생하는 경우 Latency Sensitivity 목록에서 High를 선택합니다. VM의 CPU 및 메모리 예약에 다음과 같은 값이 있는지 확인합니다.
  - 메모리 예약 값은 구성된 메모리 크기와 같아야 합니다.
  - CPU 예약 값은 측정된 물리적 CPU 속도를 곱한 최소 대기 시간이 짧은 가상 CPU 수여야 합니다.
- 구성 편집을 클릭하고 Configuration Parameters 창에서 사용 가능한 매개 변수 목록을 검색하여 스틸 클럭 회계 (stealclock.enable)를 검색합니다. 사용 가능한 경우 해당 값을 TRUE 로 설정합니다. 스틸링 클럭 계정을 사용하면 클러스터 문제 해결에 도움이 될 수 있습니다.
- 구성 추가 매개변수를 클릭합니다. 다음 매개변수 이름 및 값을 정의합니다.
  - guestinfo.ignition.config.data: 이 절차에서 이전에 생성한 base-64 인코딩 파일을 찾고 이 머신 유형에 대해 base64로 인코딩된 Ignition 구성 파일의 내용을 붙여넣습니다.
  -

- guestinfo.ignition.config.data.encoding: base64를 지정합니다.
- disk.EnableUUID: TRUE를 지정합니다.
- stealclock.enable:이 매개 변수가 정의되지 않은 경우 추가하고 TRUE 를 지정합니다.

h. Customize hardware 탭의 Virtual Hardware 패널에서 지정된 값을 필요에 따라 수정합니다. RAM, CPU 및 디스크 스토리지의 양이 시스템 유형에 대한 최소 요구사항을 충족하는지 확인합니다.

i. 구성을 완료하고 VM의 전원을 켭니다.

10. 각 시스템에 대해 이전 단계에 따라 클러스터의 나머지 시스템을 생성합니다.

중요

이때 부트스트랩 및 컨트롤 플레인 시스템을 생성해야 합니다. 일부 Pod는 기본적으로 컴퓨팅 시스템에 배포되므로 클러스터를 설치하기 전에 컴퓨팅 시스템을 두 개 이상 생성합니다.

18.8.14. vSphere의 클러스터에 더 많은 컴퓨팅 머신 추가

VMware vSphere에서 사용자가 프로비저닝한 OpenShift Container Platform 클러스터에 더 많은 컴퓨팅 머신을 추가할 수 있습니다.

사전 요구 사항

- 컴퓨팅 머신의 base64로 인코딩된 Ignition 파일을 가져옵니다.
- 클러스터에 생성한 vSphere 템플릿에 액세스할 수 있습니다.

프로세스

1. 템플릿이 배포된 후 클러스터에서 시스템의 가상 머신을 배포합니다.

a. 템플릿 이름을 마우스 오른쪽 버튼으로 클릭하고 **Clone → Clone to Virtual Machine**을 클릭합니다.

b. **Select a name and folder** 탭에서 가상 머신의 이름을 지정합니다. **compute-1**과 같은 머신 유형을 이름에 포함할 수 있습니다.

c. **Select a name and folder** 탭에서 클러스터에 대해 생성한 폴더의 이름을 선택합니다.

d. **Select a compute resource** 탭에서 데이터 센터의 호스트 이름을 선택합니다.

e. 선택사항: **Select storage** 탭에서 스토리지 옵션을 사용자 지정합니다.

f. **Select clone options**에서 **Customize this virtual machine's hardware**를 선택합니다.

g. **Customize hardware** 탭에서 **VM Options → Advanced**를 클릭합니다.

- **Latency Sensitivity** 목록에서 **High**를 선택합니다.
- **Edit Configuration**을 클릭하고 **Configuration Parameters** 창에서 **Add Configuration Params**를 클릭합니다. 다음 매개변수 이름 및 값을 정의합니다.
  - **guestinfo.ignition.config.data:** 이 머신 유형에 대해 **base64**로 인코딩된 컴퓨팅 **Ignition** 구성 파일의 내용을 붙여넣습니다.
  - **guestinfo.ignition.config.data.encoding: base64**를 지정합니다.
  - **disk.EnableUUID: TRUE**를 지정합니다.

h.

Customize hardware 탭의 **Virtual Hardware** 패널에서 지정된 값을 필요에 따라 수정합니다. **RAM, CPU** 및 디스크 스토리지의 양이 시스템 유형에 대한 최소 요구사항을 충족하는지 확인합니다. 또한 사용 가능한 네트워크가 여러 개인 경우 **Add network adapter**에서 올바른 네트워크를 선택해야 합니다.

i. 구성을 완료하고 **VM**의 전원을 켭니다.

2. 계속해서 클러스터에 추가 컴퓨팅 머신을 만듭니다.

18.8.15. 디스크 파티션 설정

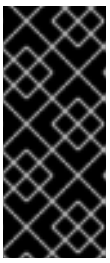
대부분의 경우 데이터 파티션은 원래 다른 운영 체제를 설치하는 대신 **RHCOS**를 설치하여 생성됩니다. 이러한 경우 **OpenShift Container Platform** 설치 프로그램은 디스크 파티션을 설정할 수 있어야합니다.

그러나 **OpenShift Container Platform** 노드를 설치할 때 기본 파티션 설정을 덮어 쓰기하기 위해 개입이 필요한 두 가지 경우가 있습니다.

- 별도의 파티션 생성: 빈 디스크에 그린 필드 설치의 경우 파티션에 별도의 스토리지를 추가할 수 있습니다. 이는 **/var** 또는 **/var**의 하위 디렉터리 (예: **/var/lib/etcd**)중 하나를 별도의 파티션으로 만드는 경우에 공식적으로 지원됩니다.

중요

100GB 이상의 디스크 크기 및 특히 **1TB**보다 큰 디스크 크기의 경우 별도의 **/var** 파티션을 만듭니다. 자세한 내용은 " 별도의 **/var** 파티션 생성" 및 이 Red Hat 지식베이스 문서 를 참조하십시오.



중요

Kubernetes는 두 개의 파일 시스템 파티션만 지원합니다. 원래 구성에 둘 이상의 파티션을 추가하는 경우 **Kubernetes**는 모든 파티션을 모니터링할 수 없습니다.

- 기존 파티션 유지: 브라운 필드 설치에서 기존 노드에 **OpenShift Container Platform**을 재설치하고 이전 운영 체제에 설치된 데이터 파티션을 유지해야 하는 경우 기존 데이터 파티션을 유지할 수 있는 **coreos-installer**에 부팅 인수와 옵션이 모두 있습니다.

별도의 /var 파티션 만들기

일반적으로 OpenShift Container Platform의 디스크 파티션 설정은 설치 프로그램에 맡겨야합니다. 그러나 확장하려는 파일 시스템의 일부에 별도의 파티션을 생성해야 하는 경우가 있습니다.

OpenShift 컨테이너 플랫폼은 /var 파티션 또는 /var의 하위 디렉터리 중 하나에 스토리지를 연결하는 단일 파티션의 추가를 지원합니다. 예를 들면 다음과 같습니다.

- /var/lib/containers: 시스템에 더 많은 이미지와 컨테이너가 추가됨에 따라 확장될 수 있는 컨테이너 관련 콘텐츠를 보관합니다.
- /var/lib/etcd: etcd 스토리지의 성능 최적화와 같은 목적으로 별도로 보관할 데이터를 보관합니다.
- /var: 감사 등의 목적에 맞게 별도로 분리하여 보관해야 하는 데이터를 보관합니다.

중요

100GB보다 큰 디스크 크기 및 특히 1TB보다 큰 디스크의 경우 별도의 /var 파티션을 만듭니다.

/var 디렉터리의 콘텐츠를 별도로 저장하면 필요에 따라 해당 영역에 대한 스토리지 확장을 보다 용이하게 하고 나중에 OpenShift Container Platform을 다시 설치하여 해당 데이터를 그대로 보존할 수 있습니다. 이 방법을 사용하면 모든 컨테이너를 다시 가져올 필요가 없으며 시스템을 업데이트할 때 대용량 로그 파일을 복사할 필요도 없습니다.

RHCOS(Red Hat Enterprise Linux CoreOS)를 새로 설치하기 전에 /var가 있어야 하므로 다음 절차에서는 OpenShift Container Platform 설치의 openshift-install 준비 단계 중에 삽입되는 머신 구성 매니페스트를 생성하여 별도의 /var 파티션을 설정합니다.

절차

1. OpenShift Container Platform 설치 파일을 저장할 디렉터리를 만듭니다.

```
$ mkdir $HOME/clusterconfig
```

2. openshift-install을 실행하여 manifest 및 openshift 하위 디렉터리에 파일 세트를 만듭니

다. 프롬프트가 표시되면 시스템 질문에 대답합니다.

```
$ openshift-install create manifests --dir $HOME/clusterconfig
? SSH Public Key ...
$ ls $HOME/clusterconfig/openshift/
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

3. 추가 파티션을 구성하는 **Butane** 구성을 생성합니다. 예를 들어 파일 이름을 **$HOME/clusterconfig/98-var-partition.bu** 로 지정하고, 디스크 장치 이름을 **worker** 시스템의 스토리지 장치 이름으로 변경하고 스토리지 크기를 적절하게 설정합니다. 이 예에서는 **/var** 디렉토리를 별도의 파티션에 배치합니다.

```
variant: openshift
version: 4.9.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1 파티션을 설정해야하는 디스크 저장 장치 이름입니다.

2 데이터 파티션을 부트 디스크에 추가할 때 최소 **25000** 메비 바이트가 권장됩니다. 루트 파일 시스템은 지정된 오프셋까지 사용 가능한 모든 공간을 채우기 위해 자동으로 크기가 조정됩니다. 값이 지정되지 않거나 지정된 값이 권장 최소값보다 작으면 생성되는 루트 파일 시스템의 크기가 너무 작아지고 **RHCOS**를 나중에 다시 설치할 때 데이터 파티션의 첫 번째 부분을 덮어 쓸 수 있습니다.

3

데이터 파티션의 크기(MB)입니다.

4

컨테이너 스토리지에 사용되는 파일 시스템에 **prjquota** 마운트 옵션을 활성화해야 합니다.

참고

별도의 **/var** 파티션을 만들 때 다른 인스턴스 유형에 동일한 장치 이름이 없는 경우 작업자 노드에 다른 인스턴스 유형을 사용할 수 없습니다.

4. Butane 구성에서 매니페스트를 생성하여 **clusterconfig/openshift** 디렉터리에 저장합니다. 예를 들어 다음 명령을 실행합니다.

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o
$HOME/clusterconfig/openshift/98-var-partition.yaml
```

5. **openshift-install**을 다시 실행하여 **manifest** 및 **openshift** 하위 디렉터리의 파일 세트에서 **Ignition** 구성을 만듭니다.

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

이제 **Ignition** 구성 파일을 **vSphere** 설치 절차에 대한 입력으로 사용하여 **RHCOS (Red Hat Enterprise Linux CoreOS)** 시스템을 설치할 수 있습니다.

18.8.16. bootupd를 사용하여 부트로더 업데이트

bootupd를 사용하여 부트로더를 업데이트하려면 **RHCOS** 머신에 수동으로 **bootupd**를 설치하거나 **systemd** 유닛이 활성화된 머신 구성을 제공해야 합니다. **grubby** 또는 기타 부트로더 툴과는 달리 **bootupd**는 커널 인수 전달과 같은 커널 공간 구성을 관리하지 않습니다.

bootupd를 설치한 후 **OpenShift Container Platform** 클러스터에서 원격으로 관리할 수 있습니다.

참고

bootupd는 **BootHole** 취약점에 대한 보호와 같이 베어 메탈 또는 가상화 하이퍼바이저 설치에서만 사용하는 것이 좋습니다.

수동 설치 방법

bootctl 명령줄 툴을 사용하여 **bootupd**를 수동으로 설치할 수 있습니다.

1. 시스템 상태를 검사합니다.

```
# bootupctl status
```

출력 예

```
Component EFI
  Installed: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
  Update: At latest version
```

2. **bootupd**를 설치하지 않고 생성된 **RHCOS** 이미지에는 명시적 채택 단계가 필요합니다.

시스템 상태가 **Adoptable**인 경우 채택을 수행합니다.

```
# bootupctl adopt-and-update
```

출력 예

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

3. 업데이트를 사용할 수 있는 경우 다음 재부팅에 변경 사항이 적용되도록 업데이트를 적용합니다.

```
# bootupctl update
```

출력 예

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

시스템 구성 방법

bootupd를 활성화하는 또 다른 방법은 머신 구성을 제공하는 것입니다.

- 다음 예와 같이 systemd 단위가 활성화된 머신 구성 파일을 제공합니다.

출력 예

```
variant: rhcos
version: 1.1.0
systemd:
  units:
    - name: custom-bootupd-auto.service
      enabled: true
      contents: |
        [Unit]
        Description=Bootupd automatic update

        [Service]
        ExecStart=/usr/bin/bootupctl update
        RemainAfterExit=yes

        [Install]
        WantedBy=multi-user.target
```

18.8.17. 부트스트랩 프로세스가 완료될 때까지 대기 중

OpenShift Container Platform 부트스트랩 프로세스는 클러스터 노드가 먼저 디스크에 설치된 영구

RHCOS 환경으로 부팅된 후에 시작됩니다. Ignition 구성 파일을 통해 제공되는 구성 정보는 부트스트랩 프로세스를 초기화하고 머신에 OpenShift Container Platform을 설치하는 데 사용됩니다. 부트스트랩 프로세스가 완료될 때까지 기다려야 합니다.

사전 요구 사항

- 클러스터에 대한 Ignition 구성 파일이 생성되어 있습니다.
- 적합한 네트워크, DNS 및 로드 밸런싱 인프라가 구성되어 있습니다.
- 설치 프로그램을 받아서 클러스터의 Ignition 구성 파일을 생성했습니다.
- 클러스터 머신에 RHCOS를 설치하고 OpenShift Container Platform 설치 프로그램에서 생성된 Ignition 구성 파일을 제공했습니다.

프로세스

1. 부트스트랩 프로세스를 모니터링합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2 다른 설치 세부 사항을 보려면 info 대신 warn, debug 또는 error를 지정합니다.

출력 예

```
INFO Waiting up to 30m0s for the Kubernetes API at
https://api.test.example.com:6443...
INFO API v1.22.1 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

이 명령은 **Kubernetes API** 서버가 컨트롤 플레인 시스템에서 부트스트랩되었다는 신호를 보낼 때 성공합니다.

2. 부트스트랩 프로세스가 완료된 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다.

중요

이 시점에 로드 밸런서에서 부트스트랩 시스템을 제거해야 합니다. 부트스트랩 머신 자체를 제거하거나 다시 포맷할 수도 있습니다.

18.8.18. CLI를 사용하여 클러스터에 로그인

클러스터 **kubeconfig** 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. **kubeconfig** 파일에는 **CLI**에서 올바른 클러스터 및 **API** 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 **OpenShift Container Platform** 설치 과정에서 생성됩니다.

사전 요구 사항

- **OpenShift Container Platform** 클러스터를 배포했습니다.
- **oc CLI**를 설치했습니다.

프로세스

1. **kubeadmin** 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>**는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 **oc** 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

18.8.19. 머신의 인증서 서명 요청 승인

클러스터에 시스템을 추가하면 추가한 시스템별로 보류 중인 인증서 서명 요청(CSR)이 두 개씩 생성됩니다. 이러한 CSR이 승인되었는지 확인해야 하며, 필요한 경우 이를 직접 승인해야 합니다. 클라이언트 요청을 먼저 승인한 다음 서버 요청을 승인해야 합니다.

사전 요구 사항

- 클러스터에 시스템을 추가했습니다.

프로세스

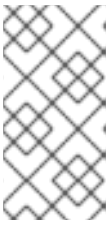
1. 클러스터가 시스템을 인식하는지 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0  Ready    master  63m  v1.22.1
master-1  Ready    master  63m  v1.22.1
master-2  Ready    master  64m  v1.22.1
```

출력에 생성된 모든 시스템이 나열됩니다.



참고

이전 출력에는 일부 **CSR**이 승인될 때까지 컴퓨팅 노드(작업자 노드라고도 함)가 포함되지 않을 수 있습니다.

2.

보류 중인 **CSR**을 검토하고 클러스터에 추가한 각 시스템에 대해 **Pending** 또는 **Approved** 상태의 클라이언트 및 서버 요청이 표시되는지 확인합니다.

```
$ oc get csr
```

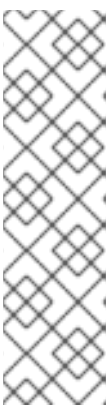
출력 예

```
NAME        AGE    REQUESTOR                                                   CONDITION
csr-8b2br   15m    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

예에서는 두 시스템이 클러스터에 참여하고 있습니다. 목록에는 승인된 **CSR**이 더 많이 나타날 수도 있습니다.

3.

CSR이 승인되지 않은 경우, 추가된 시스템에 대한 모든 보류 중인 **CSR**이 **Pending** 상태로 전환된 후 클러스터 시스템의 **CSR**을 승인합니다.



참고

CSR은 교체 주기가 자동으로 만료되므로 클러스터에 시스템을 추가한 후 **1** 시간 이내에 **CSR**을 승인하십시오. 한 시간 내에 승인하지 않으면 인증서가 교체되고 각 노드에 대해 두 개 이상의 인증서가 표시됩니다. 이러한 인증서를 모두 승인해야 합니다. 클라이언트 **CSR**이 승인되면 **Kubelet**은 인증서에 대한 보조 **CSR**을 생성하므로 수동 승인이 필요합니다. 그러면 **Kubelet**에서 동일한 매개변수를 사용하여 새 인증서를 요청하는 경우 인증서 갱신 요청은 **machine-approver**에 의해 자동으로 승인됩니다.

참고

베어 메탈 및 기타 사용자 프로비저닝 인프라와 같이 머신 **API**를 사용하도록 활성화되지 않는 플랫폼에서 실행되는 클러스터의 경우 **CSR(Kubelet service Certificate Request)**을 자동으로 승인하는 방법을 구현해야 합니다. 요청이 승인되지 않으면 **API** 서버가 **kubelet**에 연결될 때 서비스 인증서가 필요하므로 **oc exec, oc rsh, oc logs** 명령을 성공적으로 수행할 수 없습니다. **Kubelet** 엔드 포인트에 연결하는 모든 작업을 수행하려면 이 인증서 승인이 필요합니다. 이 방법은 새 **CSR**을 감시하고 **CSR**이 **system:node** 또는 **system:admin** 그룹의 **node-bootstrapper** 서비스 계정에 의해 제출되었는지 확인하고 노드의 **ID**를 확인합니다.

- 개별적으로 승인하려면 유효한 **CSR** 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1

<csr_name>은 현재 **CSR** 목록에 있는 **CSR**의 이름입니다.

- 보류 중인 **CSR**을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

참고

일부 **Operator**는 일부 **CSR**이 승인될 때까지 사용할 수 없습니다.

4. 이제 클라이언트 요청이 승인되었으므로 클러스터에 추가한 각 머신의 서버 요청을 검토해야 합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
```

```
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 나머지 CSR이 승인되지 않고 Pending 상태인 경우 클러스터 머신의 CSR을 승인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 <csr_name>은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs oc adm certificate approve
```

6. 모든 클라이언트 및 서버 CSR이 승인된 후 머신은 Ready 상태가 됩니다. 다음 명령을 실행하여 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0  Ready    master  73m  v1.22.1
master-1  Ready    master  73m  v1.22.1
master-2  Ready    master  74m  v1.22.1
worker-0  Ready    worker  11m  v1.22.1
worker-1  Ready    worker  11m  v1.22.1
```

참고

머신이 **Ready** 상태로 전환하는 데 서버 **CSR**의 승인 후 몇 분이 걸릴 수 있습니다.

추가 정보

- **CSR**에 대한 자세한 내용은 인증서 서명 요청을 참조하십시오.

18.8.20. Operator의 초기 설정

컨트롤 플레인이 초기화된 후 일부 **Operator**를 즉시 구성하여 모두 사용 가능하도록 해야 합니다.

사전 요구 사항

- 컨트롤 플레인이 초기화되어 있습니다.

절차

1. 클러스터 구성 요소가 온라인 상태인지 확인합니다.

```
$ watch -n5 oc get clusteroperators
```

출력 예

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.9.0     True        False         False      19m
baremetal                                  4.9.0     True        False         False      37m
cloud-credential                           4.9.0     True        False         False      40m
cluster-autoscaler                         4.9.0     True        False         False      37m
config-operator                            4.9.0     True        False         False      38m
console                                    4.9.0     True        False         False      26m
csi-snapshot-controller                    4.9.0     True        False         False      37m
dns                                        4.9.0     True        False         False      37m
etcd                                       4.9.0     True        False         False      36m
image-registry                             4.9.0     True        False         False      31m
ingress                                    4.9.0     True        False         False      30m
insights                                   4.9.0     True        False         False      31m
kube-apiserver                             4.9.0     True        False         False      26m
kube-controller-manager                    4.9.0     True        False         False      36m
kube-scheduler                             4.9.0     True        False         False      36m
```

```
kube-storage-version-migrator              4.9.0     True        False         False      37m
machine-api                                4.9.0     True        False         False      29m
machine-approver                           4.9.0     True        False         False      37m
machine-config                             4.9.0     True        False         False      36m
marketplace                                4.9.0     True        False         False      37m
monitoring                                 4.9.0     True        False         False      29m
network                                    4.9.0     True        False         False      38m
node-tuning                                4.9.0     True        False         False      37m
openshift-apiserver                        4.9.0     True        False         False      32m
openshift-controller-manager               4.9.0     True        False         False      30m
openshift-samples                          4.9.0     True        False         False      32m
operator-lifecycle-manager                 4.9.0     True        False         False      37m
operator-lifecycle-manager-catalog         4.9.0     True        False         False      37m
operator-lifecycle-manager-packageserver   4.9.0     True        False         False      32m
service-ca                                 4.9.0     True        False         False      38m
storage                                    4.9.0     True        False         False      37m
```

2. 사용할 수 없는 **Operator**를 구성합니다.

18.8.20.1. 기본 OperatorHub 소스 비활성화

Red Hat 및 커뮤니티 프로젝트에서 제공하는 콘텐츠를 소싱하는 **Operator** 카탈로그는 **OpenShift Container Platform**을 설치하는 동안 기본적으로 **OperatorHub**용으로 구성됩니다. 제한된 네트워크 환경에서는 클러스터 관리자로서 기본 카탈로그를 비활성화해야 합니다.

절차

- **OperatorHub** 오브젝트에 **disableAllDefaultSources: true**를 추가하여 기본 카탈로그의 소스를 비활성화합니다.

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

작은 정보

또는 웹 콘솔을 사용하여 카탈로그 소스를 관리할 수 있습니다. 관리 → 클러스터 설정 → 구성 → **OperatorHub** 페이지에서 개별 소스를 생성, 삭제, 비활성화 및 활성화할 수 있는 소스 탭을 클릭합니다.

18.8.20.2. 이미지 레지스트리 스토리지 구성

기본 스토리지를 제공하지 않는 플랫폼에서는 처음에 **Image Registry Operator**를 사용할 수 없습니

다. 설치한 후에 스토리지를 사용하도록 레지스트리를 구성하여 **Registry Operator**를 사용 가능하도록 만들어야 합니다.

프로덕션 클러스터에 필요한 영구 볼륨을 구성하는 과정의 지침이 표시됩니다. 해당하는 경우, 프로덕션 환경 외 클러스터에서만 사용할 수 있는 저장 위치로서 빈 디렉터리를 구성하는 과정의 지침이 표시됩니다.

업그레이드 중에 **Recreate** 롤아웃 전략을 사용하여 이미지 레지스트리의 블록 스토리지 유형 사용을 허용하기 위한 추가 지침이 제공됩니다.

18.8.20.2.1. VMware vSphere용 레지스트리 스토리지 구성

클러스터 관리자는 설치한 후 스토리지를 사용하도록 레지스트리를 구성해야 합니다.

사전 요구 사항

- 클러스터 관리자 권한이 있어야 합니다.
- **VMware vSphere**에 클러스터가 있어야 합니다.
- **Red Hat OpenShift Container Storage**와 같이 클러스터용 영구 스토리지 프로비저닝.

중요

OpenShift Container Platform은 복제본이 하나만 있는 경우 이미지 레지스트리 스토리지에 대한 **ReadWriteOnce** 액세스를 지원합니다. **ReadWriteOnce** 액세스에서는 레지스트리가 **Recreate** 롤아웃 전략을 사용해야 합니다. 두 개 이상의 복제본으로 고 가용성을 지원하는 이미지 레지스트리를 배포하려면 **ReadWriteMany** 액세스가 필요합니다.

- **"100Gi"** 용량이 필요합니다.

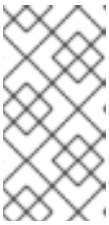
중요

테스트 결과, **RHEL**의 **NFS** 서버를 핵심 서비스용 스토리지 백엔드로 사용하는 데 문제가 있는 것으로 나타납니다. 여기에는 **OpenShift Container Registry and Quay**, 스토리지 모니터링을 위한 **Prometheus**, 로깅 스토리지를 위한 **Elasticsearch**가 포함됩니다. 따라서 **RHEL NFS**를 사용하여 핵심 서비스에서 사용하는 **PV**를 백업하는 것은 권장되지 않습니다.

마켓플레이스의 다른 **NFS** 구현에는 이러한 문제가 나타나지 않을 수 있습니다. 이러한 **OpenShift Container Platform** 핵심 구성 요소에 대해 완료된 테스트에 대한 자세한 내용은 개별 **NFS** 구현 공급업체에 문의하십시오.

절차

1. 스토리지를 사용하도록 레지스트리를 구성하기 위해 **configs.imageregistry/cluster** 리소스에서 **spec.storage.pvc**를 변경합니다.



참고

공유 스토리지를 사용할 때 보안 설정을 확인하여 외부에서의 액세스를 방지합니다.

2. 레지스트리 **pod**가 없는지 확인합니다.

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

출력 예

```
No resourses found in openshift-image-registry namespace
```

참고

출력에 레지스트리 **Pod**가 있는 경우 이 절차를 계속할 필요가 없습니다.

3. 레지스트리 구성을 확인합니다.

```
$ oc edit configs.imageregistry.operator.openshift.io
```

출력 예

```
storage:
  pvc:
    claim: 1
```

1 image-registry-storage 영구 볼륨 클레임 (PVC)을 자동으로 생성할 수 있도록 claim 필드를 비워 둡니다. PVC는 기본 스토리지 클래스를 기반으로 생성됩니다. 그러나 기본 스토리지 클래스에서 RADOS 블록 장치(RBD)와 같은 ReadWriteOnce(ReadWriteOnce) 볼륨을 제공할 수 있으므로 복제본이 두 개 이상될 때 문제가 발생할 수 있습니다.

4. clusteroperator 상태를 확인합니다.

```
$ oc get clusteroperator image-registry
```

출력 예

```
NAME             VERSION                             AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                                 True        False         False     6h50m
```

18.8.20.2.2. 프로덕션 환경 외 클러스터에서 이미지 레지스트리의 스토리지 구성

이미지 레지스트리 Operator에 대한 스토리지를 구성해야 합니다. 프로덕션 환경 외 클러스터의 경우, 이미지 레지스트리를 빈 디렉터리로 설정할 수 있습니다. 이렇게 하는 경우 레지스트리를 다시 시작하면 모든 이미지가 손실됩니다.

프로세스

- 이미지 레지스트리 스토리지를 빈 디렉터리로 설정하려면 다음을 수행하십시오.

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch
'{"spec":{"storage":{"emptyDir":{}}}}'
```

주의

프로덕션 환경 외 클러스터에 대해서만 이 옵션을 구성하십시오.

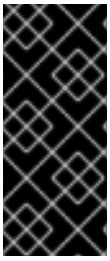
Image Registry Operator가 구성 요소를 초기화하기 전에 이 명령을 실행하면 **oc patch** 명령이 실패하며 다음 오류가 발생합니다.

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster"
not found
```

몇 분 후에 명령을 다시 실행하십시오.

18.8.20.2.3. VMware vSphere용 블록 레지스트리 스토리지 구성

클러스터 관리자로서 업그레이드 중에 이미지 레지스트리가 **vSphere VMDK(Virtual Machine Disk)**와 같은 블록 스토리지 유형을 사용할 수 있도록 허용하기 위해 **Recreate** 롤아웃 전략을 사용할 수 있습니다.



중요

블록 스토리지 볼륨이 지원되지만 프로덕션 클러스터에서 이미지 레지스트리와 함께 사용하는 것은 권장되지 않습니다. 레지스트리가 블록 스토리지에 구성된 설치는 레지스트리가 둘 이상의 복제본을 가질 수 없기 때문에 가용성이 높지 않습니다.

프로세스

1. 이미지 레지스트리 스토리지를 블록 스토리지 유형으로 설정하려면 **Recreate** 롤아웃 전략을 사용하고 복제본 **1**개 만으로 실행되도록 레지스트리를 패치합니다.

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. 블록 스토리지 장치에 PV를 프로비저닝하고 해당 볼륨의 PVC를 생성합니다. 요청된 블록 볼륨은 RWO(ReadWriteOnce) 액세스 모드를 사용합니다.

   a. VMware vSphere PersistentVolumeClaim 개체를 정의하려면 다음 내용이 포함된 pvc.yaml 파일을 생성합니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage 1
  namespace: openshift-image-registry 2
spec:
  accessModes:
  - ReadWriteOnce 3
  resources:
    requests:
      storage: 100Gi 4
```

   1 PersistentVolumeClaim 개체를 표시하는 고유한 이름입니다.

   2 PersistentVolumeClaim 오브젝트의 네임스페이스로 openshift-image-registry입니다.

   3 영구 볼륨 클레임의 액세스 모드입니다. ReadWriteOnce를 사용하면 단일 노드에서 읽기 및 쓰기 권한으로 볼륨을 마운트할 수 있습니다.

   4 영구 볼륨 클레임의 크기입니다.

   b. 파일에서 PersistentVolumeClaim 오브젝트를 만듭니다.

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 올바른 PVC를 참조하도록 레지스트리 설정을 편집합니다.

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

출력 예

```
storage:
  pvc:
    claim: 1
```

1 사용자 지정 PVC를 만들면 image-registry-storage PVC의 기본 자동 생성을 위해 claim 필드를 비워둘 수 있습니다.

올바른 PVC를 참조하도록 레지스트리 스토리지를 구성하는 방법은 VMware vSphere 용 레지스트리 스토리지 구성을 참조하십시오.

18.8.21. 사용자 프로비저닝 인프라에 설치 완료

Operator 구성을 완료한 후 제공하는 인프라에 클러스터 설치를 완료할 수 있습니다.

사전 요구 사항

- 컨트롤 플레인이 초기화되어 있습니다.
- 초기 Operator 구성을 완료해야 합니다.

프로세스

1. 다음 명령을 사용하여 모든 클러스터 구성 요소가 온라인 상태인지 확인합니다.

```
$ watch -n5 oc get clusteroperators
```

출력 예

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.9.0     True        False         False      19m
baremetal                                  4.9.0     True        False         False      37m
cloud-credential                           4.9.0     True        False         False      40m
cluster-autoscaler                         4.9.0     True        False         False      37m
config-operator                            4.9.0     True        False         False      38m
console                                    4.9.0     True        False         False      26m
csi-snapshot-controller                    4.9.0     True        False         False      37m
dns                                        4.9.0     True        False         False      37m
etcd                                       4.9.0     True        False         False      36m
image-registry                             4.9.0     True        False         False      31m
ingress                                    4.9.0     True        False         False      30m
insights                                   4.9.0     True        False         False      31m
kube-apiserver                             4.9.0     True        False         False      26m
kube-controller-manager                    4.9.0     True        False         False      36m
kube-scheduler                             4.9.0     True        False         False      36m
kube-storage-version-migrator              4.9.0     True        False         False      37m
machine-api                                4.9.0     True        False         False      29m
machine-approver                           4.9.0     True        False         False      37m
machine-config                             4.9.0     True        False         False      36m
marketplace                                4.9.0     True        False         False      37m
monitoring                                 4.9.0     True        False         False      29m
network                                    4.9.0     True        False         False      38m
node-tuning                                4.9.0     True        False         False      37m
openshift-apiserver                        4.9.0     True        False         False      32m
openshift-controller-manager               4.9.0     True        False         False      30m
openshift-samples                          4.9.0     True        False         False      32m
operator-lifecycle-manager                 4.9.0     True        False         False      37m
operator-lifecycle-manager-catalog         4.9.0     True        False         False      37m
operator-lifecycle-manager-packageserver   4.9.0     True        False         False      32m
service-ca                                 4.9.0     True        False         False      38m
storage                                    4.9.0     True        False         False      37m
```

또는 다음 명령은 모든 클러스터를 사용할 수 있을 때 알립니다. 또한 인증 정보를 검색하고 표시합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

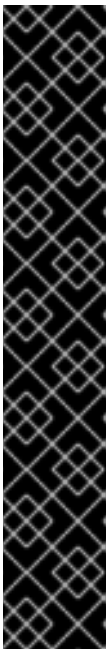
1

<installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

출력 예

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator가 Kubernetes API 서버에서 OpenShift Container Platform 클러스터 배포를 완료하면 명령이 성공합니다.



중요

- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 node-bootstrapper 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

2. Kubernetes API 서버가 Pod와 통신하고 있는지 확인합니다.

   a. 모든 Pod 목록을 보려면 다음 명령을 사용하십시오.

```
$ oc get pods --all-namespaces
```

출력 예

```
NAMESPACE                  NAME                                        READY   STATUS
RESTARTS   AGE
openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8
1/1    Running    1        9m
openshift-apiserver             apiserver-67b9g                         1/1     Running    0
```

```
3m
openshift-apiserver            apiserver-ljcmx                         1/1     Running     0
1m
openshift-apiserver            apiserver-z25h4                         1/1     Running     0
2m
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8
1/1     Running     0          5m
...
```

b. 다음 명령을 사용하여 이전 명령의 출력에 나열된 **Pod**의 로그를 표시합니다.

```
$ oc logs <pod_name> -n <namespace> 1
```

1 이전 명령의 출력에 표시된 대로 **Pod** 이름과 네임스페이스를 지정합니다.

Pod 로그가 표시되면 **Kubernetes API** 서버는 클러스터 시스템과 통신할 수 있습니다.

3. **FCP(Fibre Channel Protocol)**를 사용하는 설치에는 다중 경로를 활성화하기 위해 추가 단계가 필요합니다. 설치 중에 멀티패스를 활성화하지 마십시오.

자세한 내용은 *설치 후 머신 구성 작업* 설명서에서 **"RHCOS에서 커널 인수를 사용하여 멀티패스 활성화"**를 참조하십시오.

4. 클러스터 등록 페이지에서 클러스터를 등록합니다.

vSphere에 컴퓨팅 머신 추가에 따라 클러스터 설치가 완료된 후 추가 컴퓨팅 머신을 추가할 수 있습니다.

18.8.22. VMware vSphere 볼륨 백업

OpenShift Container Platform은 새 볼륨을 독립 영구 디스크로 프로비저닝하여 클러스터의 모든 노드에서 볼륨을 자유롭게 연결 및 분리합니다. 결과적으로 스냅샷을 사용하는 볼륨을 백업하거나 스냅샷에서 볼륨을 복원할 수 없습니다. 자세한 내용은 스냅 샷 제한 을 참조하십시오.

프로세스

영구 볼륨의 백업을 생성하려면 다음을 수행합니다.

1. 영구 볼륨을 사용중인 애플리케이션을 중지합니다.
2. 영구 볼륨을 복제합니다.
3. 애플리케이션을 다시 시작합니다.
4. 복제된 볼륨의 백업을 만듭니다.
5. 복제된 볼륨을 삭제합니다.

18.8.23. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

18.8.24. 다음 단계

- 클러스터를 사용자 지정합니다.
- Cluster Samples Operator 및 must-gather 툴의 이미지 스트림을 구성합니다.

- 제한된 네트워크에서 Operator Lifecycle Manager (OLM) 사용 방법에 대해 살펴봅니다.
- 클러스터를 설치하는 데 사용한 미러 레지스트리에 신뢰할 수 있는 CA가 있는 경우 추가 신뢰 저장소를 구성하여 클러스터에 추가합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.
- 선택 사항: vSphere Problem Detector Operator에서 이벤트를 보고 클러스터에 권한 또는 스토리지 구성 문제가 있는지 확인합니다.

18.9. VMC에 클러스터 설치 제거

설치 관리자 프로비저닝 인프라를 사용하여 AWS의 VMware Cloud(VMC)에 배포한 VMware vSphere 인프라에 설치된 클러스터를 제거할 수 있습니다.

18.9.1. 설치 관리자가 프로비저닝한 인프라를 사용하는 클러스터 제거

클라우드에서 설치 관리자 프로비저닝 인프라를 사용하는 클러스터를 제거할 수 있습니다.

참고

설치 제거 후 특히 UPI(User Provisioned Infrastructure) 클러스터에서 제거되지 않은 리소스에 대해 클라우드 공급자를 확인합니다. 설치 관리자가 생성하지 않았거나 설치 프로그램이 액세스할 수 없는 리소스가 있을 수 있습니다.

사전 요구 사항

- 클러스터를 배포하는 데 사용한 설치 프로그램의 사본을 준비합니다.
- 클러스터를 만들 때 설치 프로그램에 의해 생성된 파일을 준비합니다.

프로세스

1. 클러스터를 설치하는 데 사용한 컴퓨터에 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행합니다.

```
$ ./openshift-install destroy cluster \
--dir <installation_directory> --log-level info 1 2
```

1

<installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2

다른 세부 사항을 보려면 **info** 대신 **warn**, **debug** 또는 **error**를 지정합니다.

참고

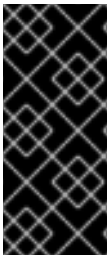
클러스터의 클러스터 정의 파일이 포함되어 있는 디렉터리를 지정해야 합니다. 설치 프로그램이 클러스터를 삭제하려면 이 디렉터리에 있는 **metadata.json** 파일이 필요합니다.

2. 선택사항: **<installation_directory>** 디렉터리와 **OpenShift Container Platform** 설치 프로그램을 삭제합니다.

19장. 모든 플랫폼에 설치

19.1. 모든 플랫폼에 클러스터 설치

OpenShift Container Platform 버전 **4.9**에서는 가상화 및 클라우드 환경을 포함하여 사용자가 프로비저닝하는 모든 인프라에 클러스터를 설치할 수 있습니다.

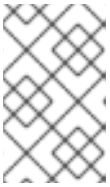


중요

가상화 또는 클라우드 환경에서 **OpenShift Container Platform** 클러스터를 설치하기 전에 테스트되지 않은 플랫폼에 **OpenShift Container Platform** 배포 지침의 정보를 검토하십시오.

19.1.1. 사전 요구 사항

- **OpenShift Container Platform** 설치 및 업데이트 프로세스에 대한 세부 사항을 검토합니다.
- 클러스터 설치 방법 선택 및 사용자를 위한 준비에 대한 문서를 읽습니다.
- 방화벽을 사용하는 경우 클러스터가 액세스해야 하는 사이트를 허용하도록 방화벽을 구성해야 합니다.



참고

프록시를 구성하는 경우에도 해당 사이트 목록을 검토하십시오.

19.1.2. OpenShift Container Platform 용 인터넷 액세스

OpenShift Container Platform 4.9에서 클러스터를 설치하려면 인터넷 액세스가 필요합니다.

다음의 경우 인터넷 액세스가 필요합니다.

- **OpenShift Cluster Manager** 에 액세스하여 설치 프로그램을 다운로드하고 서브스크립션 관리를 수행합니다. 클러스터가 인터넷에 액세스할 수 있고 **Telemetry** 서비스를 비활성화하지 않은 경우, 클러스터에 자동으로 권한이 부여됩니다.

- Quay.io에 액세스. 클러스터를 설치하는 데 필요한 패키지를 받을 수 있습니다.
- 클러스터 업데이트를 수행하는 데 필요한 패키지를 받을 수 있습니다.

중요

클러스터가 직접 인터넷에 액세스할 수 없는 경우, 프로비저닝하는 일부 유형의 인프라에서 제한된 네트워크 설치를 수행할 수 있습니다. 이 프로세스 동안 필요한 콘텐츠를 다운로드하고 이를 사용하여 설치 패키지로 미러 레지스트리를 채웁니다. 설치 유형에 따라서는 클러스터를 설치하는 환경에 인터넷 액세스가 필요하지 않을 수도 있습니다. 클러스터를 업데이트하기 전에 미러 레지스트리의 내용을 업데이트합니다.

19.1.3. 사용자 프로비저닝 인프라를 포함한 클러스터의 시스템 요구사항

사용자 프로비저닝 인프라가 포함된 클러스터의 경우, 필요한 모든 시스템을 배포해야 합니다.

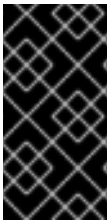
이 섹션에서는 사용자 프로비저닝 인프라에 **OpenShift Container Platform**을 배포해야 하는 요구 사항에 대해 설명합니다.

19.1.3.1. 클러스터 설치에 필요한 시스템

최소 **OpenShift Container Platform** 클러스터에 다음과 같은 호스트가 필요합니다.

표 19.1. 최소 필수 호스트

호스트	설명
임시 부트스트랩 시스템 한 개	컨트롤 플레인 시스템 세 개에 OpenShift Container Platform 클러스터를 배포하기 위한 부트스트랩 시스템이 클러스터에 필요합니다. 클러스터를 설치한 후 부트스트랩 시스템을 제거할 수 있습니다.
컨트롤 플레인 시스템 세 개	컨트롤 플레인 시스템은 컨트롤 플레인을 구성하는 Kubernetes 및 OpenShift Container Platform 서비스를 실행합니다.
두 개 이상의 컴퓨팅 시스템(작업자 시스템이라고도 함).	OpenShift Container Platform 사용자가 요청한 워크로드는 컴퓨팅 머신에서 실행됩니다.



중요

클러스터의 고가용성을 유지하려면 이러한 클러스터 시스템에 대해 별도의 물리적 호스트를 사용하십시오.

부트스트랩, 컨트롤 플레인 시스템은 운영 체제로 **RHCOS (Red Hat Enterprise Linux CoreOS)**를 사용해야 합니다. 그러나 컴퓨팅 머신은 **RHCOS(Red Hat Enterprise Linux CoreOS), RHEL (Red Hat Enterprise Linux) 7.9** 또는 **RHEL 8.4** 중에서 선택할 수 있습니다.

RHCOS는 **RHEL 8(Red Hat Enterprise Linux)**을 기반으로하며 모든 하드웨어 인증 및 요구사항을 모두 이어받습니다. **Red Hat Enterprise Linux 기술 기능 및 제한**을 참조하십시오.

19.1.3.2. 클러스터 설치를 위한 최소 리소스 요구 사항

각 클러스터 시스템이 다음과 같은 최소 요구사항을 충족해야 합니다.

표 **19.2.** 최소 리소스 요구사항

시스템	운영 체제	vCPU [1]	가상 RAM	스토리지	IOPS [2]
부트스트랩	RHCOS	4	16GB	100GB	300
컨트롤 플레인	RHCOS	4	16GB	100GB	300
컴퓨팅	RHCOS, RHEL 7.9 또는 RHEL 8.4 [3]	2	8GB	100GB	300

1. **SMT**(동시 멀티스레딩) 또는 하이퍼 스레딩이 활성화되지 않은 경우 하나의 **vCPU**는 하나의 물리적 코어와 동일합니다. 활성화하면 다음과 같은 공식을 사용하여 해당 비율을 계산합니다. **(**코어 당 스레드 수 **×** 코어 수**) ×** 소켓 수 **= vCPU** 수

2. **OpenShift Container Platform** 및 **Kubernetes**는 디스크 성능에 민감하며 특히 **10ms p99 fsync** 기간이 필요한 컨트롤 플레인 노드의 **etcd**에 더 빠른 스토리지가 권장됩니다. 많은 클라우드 플랫폼에서 스토리지 크기와 **IOPS**를 함께 확장되므로 충분한 성능을 얻으려면 스토리지 볼륨을 과도하게 할당해야 할 수 있습니다.

3. 사용자가 프로비저닝한 모든 설치와 마찬가지로 클러스터에서 **RHEL** 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. **RHEL 7** 컴퓨팅 머신 사용

은 더 이상 사용되지 않으며 향후 **OpenShift Container Platform 4** 릴리스에서 제거될 예정입니다.

19.1.3.3. 인증서 서명 요청 관리

사용자가 프로비저닝하는 인프라를 사용하는 경우 자동 시스템 관리 기능으로 인해 클러스터의 액세스가 제한되므로 설치한 후 클러스터 인증서 서명 요청(**CSR**)을 승인하는 메커니즘을 제공해야 합니다. **kube-controller-manager**는 **kubelet** 클라이언트 **CSR**만 승인합니다. **machine-approver**는 올바른 시스템에서 발행한 요청인지 확인할 수 없기 때문에 **kubelet** 자격 증명을 사용하여 요청하는 서비스 인증서의 유효성을 보장할 수 없습니다. **kubelet** 서빙 인증서 요청의 유효성을 확인하고 요청을 승인하는 방법을 결정하여 구현해야 합니다.

19.1.3.4. 사용자 프로비저닝 인프라에 대한 네트워킹 요구사항

모든 **RHCOS(Red Hat Enterprise Linux CoreOS)** 시스템이 부팅 중에 **Ignition** 구성 파일을 가져오려면 **initramfs**에 네트워킹을 구성해야 합니다.

초기 부팅 과정에서 시스템에 필요한 부팅 옵션을 제공하여 **DHCP** 서버를 통해 설정하거나 정적으로 설정하는 **IP** 주소 구성이 필요합니다. 네트워크 연결이 설정된 후 시스템은 **HTTP** 또는 **HTTPS** 서버에서 **Ignition** 구성 파일을 다운로드합니다. 그런 다음 **Ignition** 구성 파일을 사용하여 각 머신의 정확한 상태를 설정합니다. **Machine Config Operator**는 설치 후 새 인증서 또는 키 적용과 같은 머신에 대한 추가 변경을 완료합니다.

클러스터 시스템의 장기적인 관리를 위해 **DHCP** 서버를 사용하는 것이 좋습니다. **DHCP** 서버가 클러스터 시스템에 영구 **IP** 주소, **DNS** 서버 정보 및 호스트 이름을 제공하도록 구성되었는지 확인합니다.

참고

사용자 프로비저닝 인프라에 **DHCP** 서비스를 사용할 수 없는 경우 **RHCOS** 설치 시 노드에 **IP** 네트워킹 구성과 **DNS** 서버의 주소를 대신 제공할 수 있습니다. **ISO** 이미지에서 설치하는 경우 부팅 인수로 전달할 수 있습니다. 고정 **IP** 프로비저닝 *및 고급 네트워킹 옵션에 대한 자세한 내용은 **RHCOS** 설치 및 **OpenShift Container Platform** 부트스트랩 프로세스 시작* 섹션을 참조하십시오.

Kubernetes API 서버가 클러스터 시스템의 노드 이름을 확인할 수 있어야 합니다. **API** 서버와 작업자 노드가 서로 다른 영역에 있는 경우, **API** 서버가 노드 이름을 확인할 수 있도록 기본 **DNS** 검색 영역을 설정할 수 있습니다. 노드 개체와 모든 **DNS** 요청에서 항상 정규화된 도메인 이름으로 호스트를 가리키는 것도 지원되는 방법입니다

19.1.3.4.1. DHCP를 통해 클러스터 노드의 호스트 이름 설정

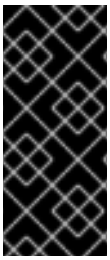
RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 호스트 이름은 NetworkManager를 통해 설정됩니다. 기본적으로 시스템은 DHCP를 통해 호스트 이름을 가져옵니다. DHCP에서 호스트 이름을 제공하지 않거나, 커널 인수를 통해 정적으로 설정하거나 다른 방법을 통해 설정하면 역방향 DNS 조회를 통해 얻을 수 있습니다. 역방향 DNS 조회는 노드에서 네트워크를 초기화한 후 수행되며 확인하는 데 시간이 걸릴 수 있습니다. 다른 시스템 서비스는 이 보다 먼저 시작하여 호스트 이름을 localhost 등으로 감지할 수 있습니다. DHCP를 사용하여 각 클러스터 노드의 호스트 이름을 제공하여 이 문제를 방지할 수 있습니다.

또한 DHCP를 통해 호스트 이름을 설정하면 DNS 분할 수평 구현 환경에서 수동으로 DNS 레코드 이름 구성 오류를 무시할 수 있습니다.

19.1.3.4.2. 네트워크 연결 요구사항

OpenShift Container Platform 클러스터 구성 요소가 통신할 수 있도록 시스템 간 네트워크 연결을 구성해야 합니다. 각 시스템에서 클러스터에 있는 다른 모든 시스템의 호스트 이름을 확인할 수 있어야 합니다.

이 섹션에서는 필요한 포트에 대해 자세히 설명합니다.



중요

연결된 OpenShift Container Platform 환경에서 모든 노드는 플랫폼 컨테이너의 이미지를 가져오고 Red Hat에 원격 측정 데이터를 제공하기 위해 인터넷에 액세스할 수 있어야 합니다.

표 19.3. 모든 시스템 간 통신에 사용되는 포트

프로토콜	포트	설명
ICMP	해당 없음	네트워크 연결성 테스트
TCP	**1936**	메트릭
	9000-**9999**	**9100**-**9101** 포트의 노드 내보내기 및 **9099** 포트의 Cluster Version Operator를 포함한 호스트 수준 서비스.
	10250-**10259**	Kubernetes에서 예약하는 기본 포트
	10256	openshift-sdn
UDP	**4789**	VXLAN 및 Geneve

프로토콜	포트	설명
	6081	VXLAN 및 Geneve
	9000-9999	**9100-9101** 포트의 노드 내보내기를 포함한 호스트 수준 서비스.
	500	IPsec IKE 패킷
	4500	IPsec NAT-T 패킷
TCP/UDP	**30000-32767**	Kubernetes 노드 포트
ESP	해당 없음	IPsec Encapsulating Security Payload (ESP)

표 19.4. 모든 시스템과 컨트롤 플레인 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**6443**	Kubernetes API

표 19.5. 컨트롤 플레인 머신 간 통신에 사용되는 포트

프로토콜	포트	설명
TCP	**2379-2380**	etcd 서버 및 피어 포트

사용자 프로비저닝 인프라에 대한 NTP 구성

OpenShift Container Platform 클러스터는 기본적으로 공용 **NTP(Network Time Protocol)** 서버를 사용하도록 구성되어 있습니다. 로컬 엔터프라이즈 **NTP** 서버를 사용하거나 클러스터가 연결이 끊긴 네트워크에 배포되는 경우 특정 시간 서버를 사용하도록 클러스터를 구성할 수 있습니다. 자세한 내용은 ***chrony* 타임 서비스 설정** 문서를 참조하십시오.

DHCP 서버가 **NTP** 서버 정보를 제공하는 경우 **RHCOS(Red Hat Enterprise Linux CoreOS)** 시스템의 **chrony** 타임 서비스에서 정보를 읽고 **NTP** 서버와 클럭을 동기화할 수 있습니다.

추가 리소스

- **chrony 타임 서비스 설정**

19.1.3.5. 사용자 프로비저닝 DNS 요구사항

OpenShift Container Platform 배포의 경우 다음 구성 요소에 **DNS** 이름을 확인해야 합니다.

- **Kubernetes API**
- **OpenShift Container Platform** 애플리케이션 와일드카드
- 부트스트랩, 컨트롤 플레인 및 컴퓨팅 시스템

Kubernetes API, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템에 대한 역방향 **DNS** 확인이 필요합니다.

DNS A/AAAA 또는 **CNAME** 레코드는 이름 확인에 사용되며 **PTR** 레코드는 역방향 이름 확인에 사용됩니다. **RHCOS (Red Hat Enterprise Linux CoreOS)**는 **DHCP**에서 호스트 이름을 제공하지 않는 한 모든 노드의 호스트 이름을 설정할 때 역방향 레코드를 사용하기 때문에 역방향 레코드가 중요합니다. 또한 역방향 레코드는 **OpenShift Container Platform**이 작동하는 데 필요한 인증서 서명 요청 **(CSR)**을 생성하는 데 사용됩니다.

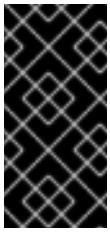
참고

DHCP 서버를 사용하여 각 클러스터 노드에 호스트 이름을 제공하는 것이 좋습니다. 자세한 내용은 *사용자 프로비저닝 인프라 섹션에 대한 DHCP 권장 사항* 섹션을 참조하십시오.

사용자가 프로비저닝한 **OpenShift Container Platform** 클러스터에 대해 다음 **DNS** 레코드가 필요하며 설치 전에 있어야 합니다. 각 레코드에서 **<cluster_name>**은 클러스터 이름이고 **<base_domain>**은 **install-config.yaml** 파일에서 지정하는 기반 도메인입니다. 전체 **DNS** 레코드는 **<component>.<cluster_name>.<base_domain>** 형식입니다.

표 **19.6.** 필수 **DNS** 레코드

구성 요소	레코드	설명
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API 로드 밸런서를 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.

구성 요소	레코드	설명
	api-int.<cluster_name>.<base_domain>.	내부적으로 API 로드 밸런서를 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다.  **중요** API 서버는 Kubernetes에 기록된 호스트 이름으로 작업자 노드를 확인할 수 있어야 합니다. API 서버가 노드 이름을 확인할 수 없는 경우 프록시된 API 호출이 실패할 수 있으며 pod에서 로그를 검색할 수 없습니다.
라우트	***.apps.<cluster_name>.<base_domain>.**	애플리케이션 인그레스 로드 밸런서를 참조하는 와일드카드 DNS A/AAA 또는 CNAME 레코드입니다. 애플리케이션 인그레스 로드 밸런서는 Ingress 컨트롤러 Pod를 실행하는 머신을 대상으로 합니다. Ingress 컨트롤러 Pod는 기본적으로 컴퓨팅 머신에서 실행됩니다. 이 레코드는 클러스터 외부의 클라이언트와 클러스터 내의 모든 노드에서 확인할 수 있어야 합니다. 예를 들어 **console-openshift-console.apps.<cluster_name>.<base_domain>**은 OpenShift Container Platform 콘솔의 와일드카드 경로로 사용됩니다.
부트스트랩 시스템	**bootstrap.<cluster_name>.<base_domain>.**	부트스트랩 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.
컨트롤 플레인 머신	**<master><n>.<cluster_name>.<base_domain>.**	컨트롤 플레인 노드의 각 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.
컴퓨팅 머신	**<worker><n>.<cluster_name>.<base_domain>.**	작업자 노드의 각 머신을 식별하는 DNS A/AAAA 또는 CNAME 레코드와 DNS PTR 레코드입니다. 이 레코드는 클러스터 내의 노드에서 확인할 수 있어야 합니다.

참고

OpenShift Container Platform 4.4 이상에서는 DNS 구성에서 etcd 호스트 및 SRV 레코드를 지정할 필요가 없습니다.

작은 정보

dig 명령을 사용하여 이름과 역방향 이름을 확인할 수 있습니다. 자세한 검증 단계는 *사용자 프로비저닝 인프라의 DNS 확인* 섹션을 참조하십시오.

19.1.3.5.1. 사용자 프로비저닝 클러스터의 DNS 구성 예

이 섹션에서는 사용자 프로비저닝 인프라에 OpenShift Container Platform을 배포하기 위한 DNS 요구 사항을 충족하는 A 및 PTR 레코드 구성 샘플을 제공합니다. 샘플은 하나의 DNS 솔루션을 선택하기 위한 조언을 제공하기 위한 것이 아닙니다.

이 예제에서 클러스터 이름은 ocp4이고 기본 도메인은 example.com입니다.

사용자 프로비저닝 클러스터의 DNS A 레코드 구성 예

다음 BIND 영역 파일의 예제에서는 사용자가 프로비저닝한 클러스터의 이름 확인을 위한 샘플 A 레코드를 보여줍니다.

예 19.1. 샘플 DNS 영역 데이터베이스

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
master0.ocp4.example.com. IN A 192.168.1.97 5
master1.ocp4.example.com. IN A 192.168.1.98 6
master2.ocp4.example.com. IN A 192.168.1.99 7
```

```
;
worker0.ocp4.example.com. IN A 192.168.1.11 8
worker1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1

Kubernetes API의 이름 확인을 제공합니다. 레코드는 **API** 로드 밸런서의 **IP** 주소를 나타냅니다.

2

Kubernetes API의 이름 확인을 제공합니다. 레코드는 **API** 로드 밸런서의 **IP** 주소를 참조하며 내부 클러스터 통신에 사용됩니다.

3

와일드카드 경로의 이름 확인을 제공합니다. 레코드는 애플리케이션 인그레스 로드 밸런서의 **IP** 주소를 나타냅니다. 애플리케이션 인그레스 로드 밸런서는 **Ingress** 컨트롤러 **Pod**를 실행하는 머신을 대상으로 합니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

참고

이 예제에서는 **Kubernetes API** 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 **API** 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

4

부트스트랩 시스템의 이름 확인을 제공합니다.

5 6 7

컨트롤 플레인 시스템의 이름 확인을 제공합니다.

8 9

컴퓨팅 시스템의 이름 확인을 제공합니다.

사용자 프로비저닝 클러스터의 **DNS PTR** 레코드 구성 예

다음 예제 BIND 영역 파일은 사용자 프로비저닝 클러스터의 역방향 이름 확인을 위한 샘플 PTR 레코드를 보여줍니다.

예 19.2. 역방향 레코드의 샘플 DNS 영역 데이터베이스

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR master0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR master1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR master2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com. 8
;
;EOF
```

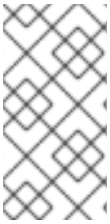
1 Kubernetes API의 역방향 DNS 확인을 제공합니다. PTR 레코드는 API 로드 밸런서의 레코드 이름을 참조합니다.

2 Kubernetes API의 역방향 DNS 확인을 제공합니다. PTR 레코드는 API 로드 밸런서의 레코드 이름을 참조하며 내부 클러스터 통신에 사용됩니다.

3 부트스트랩 시스템의 역방향 DNS 확인을 제공합니다.

4 5 6 컨트롤 플레인 시스템의 역방향 DNS 확인을 제공합니다.

7 8 컴퓨팅 시스템의 역방향 DNS 확인을 제공합니다.

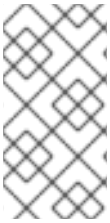


참고

OpenShift Container Platform 애플리케이션 와일드카드에는 **PTR** 레코드가 필요하지 않습니다.

19.1.3.6. 사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구사항

OpenShift Container Platform을 설치하기 전에 **API** 및 애플리케이션 인그레스 로드 밸런싱 인프라를 프로비저닝해야 합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 **API** 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.



참고

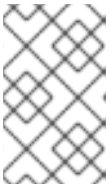
RHEL(Red Hat Enterprise Linux) 인스턴스를 사용하여 **API** 및 애플리케이션 인그레스 로드 밸런서를 배포하려면 **RHEL** 서브스크립션을 별도로 구매해야 합니다.

로드 밸런서 인프라는 다음 요구 사항을 충족해야 합니다.

1. **API** 로드 밸런서: 플랫폼과 상호 작용하고 플랫폼을 구성할 수 있도록 사용자(인간과 시스템 모두)에게 공통 끝점을 제공합니다. 다음 조건을 설정합니다.

   - **Layer 4** 로드 밸런싱 전용입니다. 이를 **Raw TCP, SSL Passthrough** 또는 **SSL Bridge** 모드라고 합니다. **SSL Bridge** 모드를 사용하는 경우, **API** 경로에 대해 **SNI(Server Name Indication,** 서버 이름 표시)를 활성화해야 합니다.

   - 스테이트리스 로드 밸런싱 알고리즘입니다. 옵션은 로드 밸런서 구현에 따라 달라집니다.

     참고

     API 로드 밸런서가 제대로 작동하기 위해 세션 지속성이 필요하지 않습니다.

     로드 밸런서의 전면과 후면 모두에서 다음 포트를 구성하십시오.

표 19.7. API 로드 밸런서

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
6443	부트스트랩 및 컨트롤 플레인. 부트스트랩 시스템이 클러스터 컨트롤 플레인을 초기화한 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다. API 서버 상태 검사 프로브에 대한 **/readyz** 끝점을 구성해야 합니다.	X	X	Kubernetes API 서버
22623	부트스트랩 및 컨트롤 플레인. 부트스트랩 시스템이 클러스터 컨트롤 플레인을 초기화한 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다.	X		시스템 구성 서버

참고

API 서버가 /readyz 엔드포인트를 해제하는 시점부터 풀에서 API 서버 인스턴스가 제거되는 시점까지 시간이 30초를 넘지 않도록 로드 밸런서를 구성해야 합니다. /readyz가 오류를 반환하거나 정상 상태가 된 후 정해진 시간 안에 끝점이 제거 또는 추가되어야 합니다. 5초 또는 10초의 프로빙 주기(두 번의 성공적인 요청은 정상 상태, 세 번의 요청은 비정상 상태)는 충분한 테스트를 거친 값입니다.

2. **애플리케이션 인그레스 로드 밸런서: 클러스터 외부에서 유입되는 애플리케이션 트래픽에 대한 인그래스 포인트를 제공합니다. 다음 조건을 설정합니다.**

- **Layer 4 로드 밸런싱 전용입니다. 이를 Raw TCP, SSL Passthrough 또는 SSL Bridge 모드라고 합니다. SSL Bridge 모드를 사용하는 경우 인그레스 경로에 대해 SNI(Server Name Indication, 서버 이름 표시)를 활성화해야 합니다.**

- **사용 가능한 옵션과 플랫폼에서 호스팅되는 애플리케이션 유형에 따라 연결 기반 또는 세션 기반 지속성이 권장됩니다.**

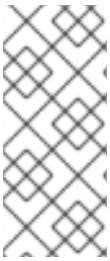
작은 정보

애플리케이션 인그레스 로드 밸런서에서 클라이언트의 실제 IP 주소를 확인할 수 있는 경우 소스 IP 기반 세션 지속성을 활성화하면 엔드 투 엔드 TLS 암호화를 사용하는 애플리케이션의 성능을 향상시킬 수 있습니다.

로드 밸런서의 전면과 후면 모두에서 다음 포트를 구성하십시오.

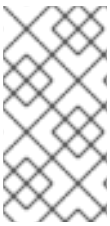
표 19.8. 애플리케이션 인그레스 로드 밸런서

포트	백엔드 시스템(풀 멤버)	내부	외부	설명
443	기본적으로 인그레스 컨트롤러 pod, 컴퓨팅 또는 작업자를 실행하는 시스템입니다.	X	X	HTTPS 트래픽
80	기본적으로 인그레스 컨트롤러 pod, 컴퓨팅 또는 작업자를 실행하는 시스템입니다.	X	X	HTTP 트래픽
1936	기본적으로 Ingress 컨트롤러 Pod를 실행하는 작업자 노드입니다. 수신 상태 점검 프로브에 대해 **/healthz/ready** 끝점을 구성해야 합니다.	X	X	HTTP 트래픽



참고

컴퓨팅 노드가 0인 3-노드 클러스터를 배포하는 경우 Ingress 컨트롤러 Pod는 컨트롤 플레인 노드에서 실행됩니다. 3-노드 클러스터 배포에서 HTTP 및 HTTPS 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다.

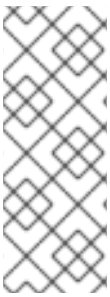


참고

인그레스 라우터에 대한 작업 구성이 OpenShift Container Platform 클러스터에 필요합니다. 컨트롤 플레인 초기화 후 인그레스 라우터를 설정해야 합니다.

19.1.3.6.1. 사용자 프로비저닝 클러스터의 로드 밸런서 구성 예

이 섹션에서는 사용자 프로비저닝 클러스터의 로드 밸런싱 요구 사항을 충족하는 API 및 애플리케이션 수신 로드 밸런서 구성 예를 제공합니다. 샘플은 HAProxy 로드 밸런서에 대한 /etc/haproxy/haproxy.cfg 구성입니다. 이 예제에서는 하나의 로드 밸런싱 솔루션을 선택하기 위한 조언을 제공하는 것을 목적으로 하지 않습니다.



참고

이 예제에서는 Kubernetes API 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 API 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

예 19.3. API 및 애플리케이션 인그레스 로드 밸런서 구성 샘플

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
frontend stats
  bind *:1936
  mode            http
  log             global
  maxconn 10
  stats enable
  stats hide-version
  stats refresh 30s
  stats show-node
  stats show-desc Stats for ocp4 cluster (1)
  stats auth admin:ocp4
  stats uri /stats
listen api-server-6443 (2)
  bind *:6443
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:6443 check inter 1s backup (3)
  server master0 master0.ocp4.example.com:6443 check inter 1s
  server master1 master1.ocp4.example.com:6443 check inter 1s
  server master2 master2.ocp4.example.com:6443 check inter 1s
listen machine-config-server-22623 (4)
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup (5)
  server master0 master0.ocp4.example.com:22623 check inter 1s
  server master1 master1.ocp4.example.com:22623 check inter 1s
  server master2 master2.ocp4.example.com:22623 check inter 1s
listen ingress-router-443 (6)
  bind *:443
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:443 check inter 1s
  server worker1 worker1.ocp4.example.com:443 check inter 1s
listen ingress-router-80 (7)
  bind *:80
  mode tcp
```

```
balance source
server worker0 worker0.ocp4.example.com:80 check inter 1s
server worker1 worker1.ocp4.example.com:80 check inter 1s
```

1

이 예에서 클러스터 이름은 **ocp4** 입니다.

2

포트 **6443**은 **Kubernetes API** 트래픽을 처리하고 컨트롤 플레인 시스템을 가리킵니다.

3 5

부트스트랩 항목은 **OpenShift Container Platform** 클러스터 설치 전에 있어야 하며 부트스트랩 프로세스가 완료된 후 제거해야 합니다.

4

포트 **22623**은 머신 구성 서버 트래픽을 처리하고 컨트롤 플레인 시스템을 가리킵니다.

6

포트 **443**은 **HTTPS** 트래픽을 처리하고 **Ingress** 컨트롤러 **Pod**를 실행하는 시스템을 가리킵니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

7

포트 **80**은 **HTTP** 트래픽을 처리하고 **Ingress** 컨트롤러 **Pod**를 실행하는 머신을 가리킵니다. **Ingress** 컨트롤러 **Pod**는 기본적으로 컴퓨팅 머신에서 실행됩니다.

참고

컴퓨팅 노드가 **0**인 **3-**노드 클러스터를 배포하는 경우 **Ingress** 컨트롤러 **Pod**는 컨트롤 플레인 노드에서 실행됩니다. **3-**노드 클러스터 배포에서 **HTTP** 및 **HTTPS** 트래픽을 컨트롤 플레인 노드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다.

작은 정보

HAProxy를 로드 밸런서로 사용하는 경우 **HAProxy** 노드에서 **netstat -nltupe**를 실행하여 **haproxy** 프로세스가 포트 **6443, 22623, 443** 및 **80**에서 수신 대기 중인지 확인할 수 있습니다.

참고

HAProxy를 로드 밸런서로 사용하고 SELinux가 enforcing으로 설정된 경우 HAProxy 서비스가 setsebool -P haproxy_connect_any=1을 실행하여 구성된 TCP 포트에 바인딩할 수 있는지 확인해야 합니다.

19.1.4. 사용자 프로비저닝 인프라 준비

사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치하기 전에 기본 인프라를 준비해야 합니다.

이 섹션에서는 OpenShift Container Platform 설치를 준비하기 위해 클러스터 인프라를 설정하는 데 필요한 높은 수준의 단계에 대해 자세히 설명합니다. 여기에는 클러스터 노드에 대한 IP 네트워킹 및 네트워크 연결 구성, 방화벽을 통해 필요한 포트 활성화, 필수 DNS 및 로드 밸런싱 인프라를 설정하는 작업이 포함됩니다.

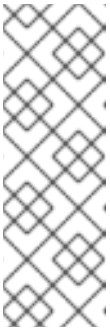
준비 후 클러스터 인프라는 *사용자 프로비저닝 인프라가 있는 클러스터의 요구 사항* 섹션에 설명된 요구 사항을 충족해야 합니다.

사전 요구 사항

- OpenShift Container Platform 4.x 테스트된 통합 페이지를 검토했습니다.
- *사용자 프로비저닝 인프라가 있는 클러스터의 요구 사항* 섹션에 자세히 설명된 인프라 요구 사항을 검토했습니다.

프로세스

1. DHCP를 사용하여 클러스터 노드에 IP 네트워킹 구성을 제공하는 경우 DHCP 서비스를 구성합니다.
   a. 노드의 영구 IP 주소를 DHCP 서버 구성에 추가합니다. 구성에서 관련 네트워크 인터페이스의 MAC 주소를 각 노드의 의도한 IP 주소와 일치시킵니다.
   b. DHCP를 사용하여 클러스터 시스템의 IP 주소 지정을 구성하는 경우 시스템은 DHCP를 통해 DNS 서버 정보도 가져옵니다. DHCP 서버 구성을 통해 클러스터 노드에서 사용하는 영구 DNS 서버 주소를 정의합니다.

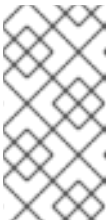


참고

DHCP 서비스를 사용하지 않는 경우 **RHCOS** 설치 시 **IP** 네트워킹 구성과 **DNS** 서버의 주소를 노드에 제공해야 합니다. **ISO** 이미지에서 설치하는 경우 부팅 인수로 전달할 수 있습니다. 고정 **IP** 프로비저닝 *및 고급 네트워킹 옵션에 대한 자세한 내용은 **RHCOS** 설치 및 **OpenShift Container Platform** 부트스트랩 프로세스 시작* 섹션을 참조하십시오.

c.

DHCP 서버 구성에 클러스터 노드의 호스트 이름을 정의합니다. 호스트 이름 고려 사항에 대한 자세한 내용은 ***DHCP**를 통해 클러스터 노드 호스트 이름 설정* 섹션을 참조하십시오.



참고

DHCP 서비스를 사용하지 않는 경우 클러스터 노드는 역방향 **DNS** 조회를 통해 호스트 이름을 가져옵니다.

2.

네트워크 인프라가 클러스터 구성 요소 간 필수 네트워크 연결을 제공하는지 확인합니다. 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 인프라 섹션의 네트워킹 요구 사항* 섹션을 참조하십시오.

3.

OpenShift Container Platform 클러스터 구성 요소가 통신하는 데 필요한 포트를 활성화하도록 방화벽을 구성합니다. 필요한 포트에 대한 자세한 내용은 *사용자 프로비저닝 인프라 섹션의 네트워킹 요구 사항* 섹션을 참조하십시오.

4.

클러스터에 필요한 **DNS** 인프라를 설정합니다.

a.

Kubernetes API, 애플리케이션 와일드카드, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템의 **DNS** 이름 확인을 구성합니다.

b.

Kubernetes API, 부트스트랩 시스템, 컨트롤 플레인 시스템 및 컴퓨팅 시스템에 대한 역방향 **DNS** 확인을 구성합니다.

OpenShift Container Platform DNS 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 **DNS** 요구 사항* 섹션을 참조하십시오.

5.

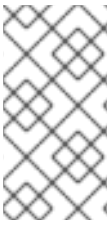
DNS 구성을 확인합니다.

a.
설치 노드에서 **Kubernetes API**의 레코드 이름, 와일드카드 경로 및 클러스터 노드에 대해 **DNS** 조회를 실행합니다. 응답의 **IP** 주소가 올바른 구성 요소에 해당하는지 확인합니다.

b.
설치 노드에서 로드 밸런서 및 클러스터 노드의 **IP** 주소에 대해 역방향 **DNS** 조회를 실행합니다. 응답의 레코드 이름이 올바른 구성 요소에 해당하는지 확인합니다.

자세한 **DNS** 검증 단계는 *사용자 프로비저닝 인프라에 대한 DNS 확인* 섹션을 참조하십시오.

6.
필요한 **API** 및 애플리케이션 수신 로드 밸런싱 인프라를 프로비저닝합니다. 요구 사항에 대한 자세한 내용은 *사용자 프로비저닝 인프라에 대한 로드 밸런싱 요구 사항* 섹션을 참조하십시오.

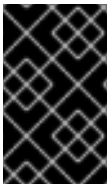


참고

일부 로드 밸런싱 솔루션에는 로드 밸런싱을 초기화하기 전에 클러스터 노드의 **DNS** 이름을 확인해야 합니다.

19.1.5. 사용자 프로비저닝 인프라에 대한 DNS 확인 검증

사용자 프로비저닝 인프라에 **OpenShift Container Platform**을 설치하기 전에 **DNS** 구성을 확인할 수 있습니다.



중요

클러스터를 설치하기 전에 이 섹션에 설명된 검증 단계를 성공해야 합니다.

사전 요구 사항

- 사용자 프로비저닝 인프라에 필요한 **DNS** 레코드를 구성했습니다.

프로세스

1.
설치 노드에서 **Kubernetes API**의 레코드 이름, 와일드카드 경로 및 클러스터 노드에 대해 **DNS** 조회를 실행합니다. 응답에 포함된 **IP** 주소가 올바른 구성 요소에 해당하는지 확인합니다.

a.
Kubernetes API 레코드 이름을 조회합니다. 결과가 **API** 로드 밸런서의 **IP** 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
```

1

<nameserver_ip>를 네임서버의 IP 주소로, <cluster_name>을 클러스터 이름으로, <base_domain>을 기본 도메인 이름으로 바꿉니다.

출력 예

```
api.ocp4.example.com.  0 IN A 192.168.1.5
```

b.

Kubernetes 내부 API 레코드 이름을 조회합니다. 결과가 API 로드 밸런서의 IP 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
```

출력 예

```
api-int.ocp4.example.com.  0 IN A 192.168.1.5
```

c.

예제 *.apps.<cluster_name>.<base_domain>을 테스트합니다. DNS 와일드카드를 조회합니다. 모든 애플리케이션 와일드카드 조회는 애플리케이션 인그레스 로드 밸런서의 IP 주소로 확인되어야 합니다.

```
$ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.
<base_domain>
```

출력 예

```
random.apps.ocp4.example.com.  0 IN A 192.168.1.5
```

참고

예제 출력에서는 Kubernetes API 및 애플리케이션 인그레스 트래픽에 동일한 로드 밸런서를 사용합니다. 프로덕션 시나리오에서는 각각에 대해 개별적으로 로드 밸런서 인프라를 확장할 수 있도록 API 및 애플리케이션 인그레스 로드 밸런서를 별도로 배포할 수 있습니다.

random 항목을 다른 와일드카드 값으로 교체할 수 있습니다. 예를 들어 OpenShift Container Platform 콘솔의 경로를 쿼리할 수 있습니다.

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

출력 예

```
console-openshift-console.apps.ocp4.example.com. 0 IN A 192.168.1.5
```

d.

부트스트랩 DNS 레코드 이름에 대해 조회를 실행합니다. 결과가 부트스트랩 노드의 IP 주소를 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.
<base_domain>
```

출력 예

```
bootstrap.ocp4.example.com.  0 IN A 192.168.1.96
```

e.

이 방법을 사용하여 컨트롤 플레인 및 컴퓨팅 노드의 DNS 레코드 이름에 대해 조회를 수행합니다. 결과가 각 노드의 IP 주소에 해당하는지 확인합니다.

2. 설치 노드에서 로드 밸런서 및 클러스터 노드의 IP 주소에 대해 역방향 DNS 조회를 실행합니다. 응답에 포함된 레코드 이름이 올바른 구성 요소에 해당하는지 확인합니다.

a. API 로드 밸런서의 IP 주소에 대해 역방향 조회를 수행합니다. 응답에 Kubernetes API 및 Kubernetes 내부 API의 레코드 이름이 포함되어 있는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

출력 예

```
5.1.168.192.in-addr.arpa. 0 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 0 IN PTR api.ocp4.example.com. 2
```

1 Kubernetes 내부 API의 레코드 이름을 제공합니다.

2 Kubernetes API의 레코드 이름을 제공합니다.

참고

OpenShift Container Platform 애플리케이션 와일드카드에는 PTR 레코드가 필요하지 않습니다. 애플리케이션 인그레스 로드 밸런서의 IP 주소에 대한 역방향 DNS 확인에는 유효성 검사 단계가 필요하지 않습니다.

b. 부트스트랩 노드의 IP 주소에 대해 역방향 조회를 수행합니다. 결과가 부트스트랩 노드의 DNS 레코드 이름을 가리키는지 확인합니다.

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

출력 예

```
96.1.168.192.in-addr.arpa. 0 IN PTR bootstrap.ocp4.example.com.
```

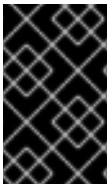
c. 이 방법을 사용하여 컨트롤 플레인 및 컴퓨팅 노드의 IP 주소에 대해 역방향 조회를 수행합니다. 결과가 각 노드의 DNS 레코드 이름과 일치하는지 확인합니다.

19.1.6. 클러스터 노드 SSH 액세스를 위한 키 쌍 생성

OpenShift Container Platform을 설치하는 동안 SSH 공개 키를 설치 프로그램에 지정할 수 있습니다. 키는 Ignition 구성 파일을 통해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드에 전달되며 노드에 대한 SSH 액세스를 인증하는 데 사용됩니다. 키는 각 노드에서 core 사용자의 ~/.ssh/authorized_keys 목록에 추가되어 암호 없는 인증을 활성화합니다.

키가 노드에 전달되면 키 쌍을 사용하여 사용자 core로 RHCOS 노드에 SSH로 SSH 연결을 수행할 수 있습니다 . SSH를 통해 노드에 액세스하려면 로컬 사용자의 SSH에서 개인 키 ID를 관리해야 합니다.

설치 디버깅 또는 재해 복구를 수행하기 위해 클러스터 노드에 SSH를 실행하려면 설치 프로세스 중에 SSH 공용 키를 지정해야 합니다. ./openshift-install gather 명령에도 SSH 공개 키가 클러스터 노드에 있어야 합니다.



중요

재해 복구 및 디버깅이 필요한 프로덕션 환경에서는이 단계를 생략하지 마십시오.

참고

AWS 키 쌍과 같이 플랫폼 고유의 방식으로 구성된 키가 아닌 로컬 키를 사용해야 합니다.

프로세스

1. 로컬 시스템에 클러스터 노드의 인증에 사용할 기존 SSH 키 쌍이 없는 경우 새로 생성합니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1

새 SSH 키의 경로 및 파일 이름(예: ~/.ssh/id_ed25519)을 지정합니다. 기존 키 쌍이 있는 경우 공개 키가 '~/.ssh 디렉터리에 있는지 확인하십시오.

참고

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치하려면 ed25519 알고리즘을 사용하는 키를 생성하지 마십시오. 대신 rsa 또는 ecdsa 알고리즘을 사용하는 키를 생성합니다.

2. 공개 SSH 키를 확인합니다.

```
$ cat <path>/<file_name>.pub
```

예를 들어 다음을 실행하여 ~/.ssh/id_ed25519.pub 공개 키를 확인합니다.

```
$ cat ~/.ssh/id_ed25519.pub
```

3. 아직 추가되지 않은 경우 로컬 사용자의 SSH 에이전트에 SSH 개인 키 ID를 추가합니다. 키의 SSH 에이전트 관리는 클러스터 노드에 암호 없는 SSH 인증을 수행하거나 ./openshift-install gather 명령을 사용하려는 경우 필요합니다.

참고

일부 배포에서는 ~/.ssh/id_rsa 및 ~/.ssh/id_dsa와 같은 기본 SSH 개인 키 ID가 자동으로 관리됩니다.

a. ssh-agent 프로세스가 로컬 사용자에 대해 실행되지 않은 경우 백그라운드 작업으로 시작합니다.

```
$ eval "$(ssh-agent -s)"
```

출력 예

```
Agent pid 31874
```

참고

클러스터가 FIPS 모드인 경우 FIPS 호환 알고리즘만 사용하여 SSH 키를 생성합니다. 키는 RSA 또는 ECDSA여야 합니다.

4. ssh-agent에 SSH 개인 키를 추가합니다.

```
$ ssh-add <path>/<file_name> 1
```

1 SSH 개인 키의 경로 및 파일 이름을 지정합니다(예: ~/.ssh/id_ed25519).

출력 예

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

다음 단계

- OpenShift Container Platform을 설치할 때 SSH 공개 키를 설치 프로그램에 지정합니다. 프로비저닝하는 인프라에 클러스터를 설치하는 경우 설치 프로그램에 키를 제공해야 합니다.

19.1.7. 설치 프로그램 받기

OpenShift Container Platform을 설치하기 전에 로컬 컴퓨터에 설치 파일을 다운로드합니다.

사전 요구 사항

- 500MB의 로컬 디스크 공간이 있는 Linux 또는 macOS를 실행하는 컴퓨터가 있습니다.

프로세스

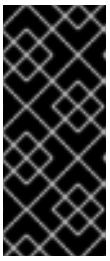
1. OpenShift Cluster Manager 사이트의 인프라 공급자 페이지에 액세스합니다. Red Hat 계정이 있으면 사용자 자격 증명으로 로그인합니다. 계정이 없으면 계정을 만드십시오.

2. 인프라 공급자를 선택합니다.

3. 설치 유형 페이지로 이동한 다음, 운영 체제에 맞는 설치 프로그램을 다운로드하여 설치 구성 파일을 저장할 디렉터리에 파일을 저장합니다.

중요

설치 프로그램은 클러스터를 설치하는 데 사용하는 컴퓨터에 여러 파일을 만듭니다. 클러스터 설치를 마친 후 설치 프로그램과 설치 프로그램으로 생성되는 파일을 보관해야 합니다. 클러스터를 삭제하려면 두 파일이 모두 필요합니다.



중요

클러스터 설치에 실패하거나 설치 프로그램으로 만든 파일을 삭제해도 클러스터는 제거되지 않습니다. 클러스터를 제거하려면 해당 클라우드 공급자에 적용되는 OpenShift Container Platform 설치 제거 절차를 완료해야 합니다.

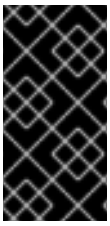
4. 설치 프로그램 파일의 압축을 풉니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar -xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager에서 설치 풀 시크릿 을 다운로드합니다. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

19.1.8. 바이너리를 다운로드하여 OpenShift CLI 설치

명령줄 인터페이스를 사용하여 OpenShift Container Platform과 상호 작용하기 위해 OpenShift CLI(oc)를 설치할 수 있습니다. Linux, Windows 또는 macOS에 oc를 설치할 수 있습니다.



중요

이전 버전의 oc를 설치한 경우 OpenShift Container Platform 4.9의 모든 명령을 완료하는 데 사용할 수 없습니다. 새 버전의 oc를 다운로드하여 설치합니다.

Linux에서 OpenShift CLI 설치

다음 절차를 사용하여 Linux에서 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.
2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. OpenShift v4.9 Linux Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvf <file>
```

5. oc 바이너리를 PATH에 있는 디렉터리에 배치합니다.

   PATH를 확인하려면 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
$ oc <command>
```

Windows에서 OpenSfhit CLI 설치

다음 절차에 따라 Windows에 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.
2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. OpenShift v4.9 Windows Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.
4. ZIP 프로그램으로 아카이브의 압축을 풉니다.
5. oc 바이너리를 PATH에 있는 디렉터리로 이동합니다.

   PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

OpenShift CLI를 설치한 후 oc 명령을 사용할 수 있습니다.

```
C:\> oc <command>
```

macOS에 OpenShift CLI 설치

다음 절차에 따라 macOS에서 OpenShift CLI(oc) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지로 이동합니다.
2. 버전 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. OpenShift v4.9 MacOSX Client 항목 옆에 있는 지금 다운로드를 클릭하고 파일을 저장합니다.

4. 아카이브의 압축을 해제하고 압축을 풉니다.

5. **oc** 바이너리 **PATH**의 디렉터리로 이동합니다.

   PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

```
$ echo $PATH
```

OpenShift CLI를 설치한 후 **oc** 명령을 사용할 수 있습니다.

```
$ oc <command>
```

19.1.9. 수동으로 설치 구성 파일 만들기

OpenShift Container Platform을 사용자가 프로비저닝한 설치의 경우 설치 구성 파일을 수동으로 생성합니다.

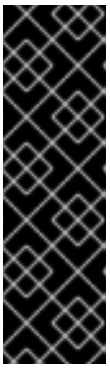
사전 요구 사항

- 로컬 시스템에 설치 프로그램에 제공할 **SSH** 공개 키가 있습니다. 키는 디버깅 및 재해 복구를 위해 클러스터 노드에 대한 **SSH** 인증에 사용됩니다.
- **OpenShift Container Platform** 설치 프로그램과 클러스터의 풀 시크릿이 있습니다.

프로세스

1. 필요한 설치 자산을 저장할 설치 디렉터리를 만듭니다.

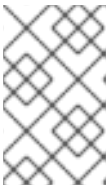
```
$ mkdir <installation_directory>
```



중요

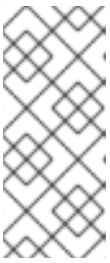
디렉터리를 만들어야 합니다. 부트스트랩 **X.509** 인증서와 같은 일부 설치 자산은 단기간에 만료되므로 설치 디렉터리를 재사용해서는 안 됩니다. 다른 클러스터 설치의 개별 파일을 재사용하려면 해당 파일을 사용자 디렉터리에 복사하면 됩니다. 그러나 설치 자산의 파일 이름은 릴리스간에 변경될 수 있습니다. 따라서 이전 **OpenShift Container Platform** 버전에서 설치 파일을 복사할 때는 주의하십시오.

2. 샘플 **install-config.yaml** 파일 템플릿을 사용자 지정하여 **<installation_directory>**에 저장합니다.



참고

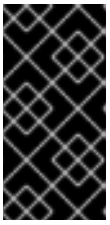
이 설정 파일의 이름을 **install-config.yaml**로 지정해야 합니다.



참고

일부 플랫폼 유형의 경우 대체로 **./openshift-install create install-config --dir <installation_directory>** 를 실행하여 **install-config.yaml** 파일을 생성할 수 있습니다. 프롬프트에서 클러스터 구성에 대한 세부 정보를 제공할 수 있습니다.

3. 여러 클러스터를 설치하는 데 사용할 수 있도록 **install-config.yaml** 파일을 백업합니다.



중요

install-config.yaml 파일은 설치 과정의 다음 단계에서 사용됩니다. 이 시점에서 이를 백업해야 합니다.

19.1.9.1. 기타 플랫폼의 샘플 install-config.yaml 파일

install-config.yaml 파일을 사용자 지정하여 **OpenShift Container Platform** 클러스터 플랫폼에 대한 자세한 정보를 지정하거나 필수 매개변수 값을 수정할 수 있습니다.

```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
```

```
  replicas: 0 4
controlPlane: 5
  hyperthreading: Enabled 6
  name: master
  replicas: 3 7
metadata:
  name: test 8
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 9
    hostPrefix: 23 10
  networkType: OpenShiftSDN
  serviceNetwork: 11
  - 172.30.0.0/16
platform:
  none: {} 12
fips: false 13
pullSecret: '{"auths": ...}' 14
sshKey: 'ssh-ed25519 AAAA...' 15
```

1

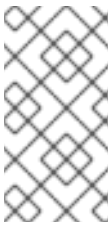
클러스터의 기본 도메인입니다. 모든 **DNS** 레코드는 이 기본 도메인의 하위 도메인이어야 하며 클러스터 이름을 포함해야 합니다.

2 5

controlPlane 섹션은 단일 매핑이지만 **compute** 섹션은 일련의 매핑입니다. 서로 다른 데이터 구조의 요구사항을 충족하도록 **compute** 섹션의 첫 번째 줄은 하이픈(**-**)으로 시작해야 하며 **controlPlane** 섹션의 첫 번째 줄은 하이픈으로 시작할 수 없습니다. 하나의 컨트롤 플레인 풀만 사용됩니다.

3 6

동시 멀티스레딩(**SMT**) 또는 **hyperthreading** 활성화/비활성화 여부를 지정합니다. 시스템 코어의 성능을 높이기 위해 기본적으로 **SMT**가 활성화됩니다. 매개변수 값을 **Disabled**로 설정하여 비활성화할 수 있습니다. **SMT**를 비활성화하는 경우 모든 클러스터 머신에서 이를 비활성화해야 합니다. 여기에는 컨트롤 플레인과 컴퓨팅 머신이 모두 포함됩니다.



참고

SMT(동시 멀티 스레딩)는 기본적으로 활성화되어 있습니다. **BIOS** 설정에서 **SMT**를 활성화하지 않으면 **hyperthreading** 매개변수가 적용되지 않습니다.

중요

BIOS에서든 **install-config.yaml** 파일에서든 **hyperthreading**을 비활성화한 경우 용량 계획에서 시스템 성능이 크게 저하될 수 있는 문제를 고려해야 합니다.

4

사용자 프로비저닝 인프라에 **OpenShift Container Platform**을 설치할 때 이 값을 **0**으로 설정해야 합니다. 설치 프로그램에서 제공하는 설치에서 매개 변수는 클러스터가 생성 및 관리하는 컴퓨팅 머신 수를 제어합니다. 사용자 프로비저닝 설치에서는 클러스터 설치를 완료하기 전에 컴퓨팅 시스템을 수동으로 배포해야 합니다.

참고

3-노드 클러스터를 설치하는 경우 **RHCOS(Red Hat Enterprise Linux CoreOS)** 시스템을 설치할 때 컴퓨팅 머신을 배포하지 마십시오.

7

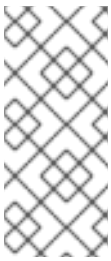
클러스터에 추가하는 컨트롤 플레인 시스템의 수입니다. 클러스터에서 이 값을 클러스터의 **etcd** 끝점 수로 사용하므로 이 값은 배포하는 컨트롤 플레인 시스템의 수와 일치해야 합니다.

8

DNS 레코드에 지정한 클러스터 이름입니다.

9

Pod IP 주소가 할당되는 **IP** 주소 블록입니다. 이 블록은 기존 물리적 네트워크와 중복되지 않아야합니다. 이러한 **IP** 주소는 **Pod** 네트워크에 사용됩니다. 외부 네트워크에서 **Pod**에 액세스해야 하는 경우, 트래픽을 관리하도록 로드 밸런서와 라우터를 설정해야 합니다.



참고

클래스 **E CIDR** 범위는 향후 사용을 위해 예약되어 있습니다. 클래스 **E CIDR** 범위를 사용하려면 네트워킹 환경에서 클래스 **E CIDR** 범위 내에서 **IP** 주소를 수락하는지 확인해야 합니다.

10

개별 노드 각각에 할당할 서브넷 접두사 길이입니다. 예를 들어 **hostPrefix**를 **23**으로 설정하면 지정된 **cidr** 이외 **/23** 서브넷이 각 노드에 할당되어 **510(2^(32 - 23) - 2) Pod IP** 주소가 허용됩니다. 외부 네트워크에서 노드에 액세스해야 하는 경우 트래픽을 관리하도록 로드 밸런서와 라우터를 구성합니다.

11

서비스 IP 주소에 사용할 IP 주소 풀입니다. IP 주소 풀은 하나만 입력할 수 있습니다. 이 블록은 기존 물리적 네트워크와 중복되지 않아야합니다. 외부 네트워크에서 서비스에 액세스해야 하는 경우, 트래픽을 관리하도록 로드 밸런서와 라우터를 구성합니다.

12

플랫폼을 none으로 설정해야 합니다. 플랫폼에 대한 추가 플랫폼 구성 변수는 지정할 수 없습니다.

중요

플랫폼 유형으로 설치된 클러스터는 Machine API로 컴퓨팅 머신 관리와 같은 일부 기능을 사용할 수 없습니다. 이 제한은 클러스터에 연결된 컴퓨팅 시스템이 일반적으로 기능을 지원하는 플랫폼에 설치된 경우에도 적용됩니다. 이 매개변수는 설치 후 변경할 수 없습니다.

13

FIPS 모드 활성화 또는 비활성화 여부입니다. 기본적으로 FIPS 모드는 비활성화됩니다. FIPS 모드가 활성화되면 OpenShift Container Platform이 실행되는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 기본 Kubernetes 암호화 제품군은 우회하고 RHCOS와 함께 제공되는 암호화 모듈을 대신 사용합니다.

중요

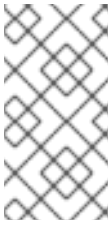
FIPS 검증 / 진행중인 모듈 암호화 라이브러리 사용은 x86_64 아키텍처의 OpenShift Container Platform 배포에서만 지원됩니다.

14

Red Hat OpenShift Cluster Manager의 풀 시크릿. 이 풀 시크릿을 사용하면 OpenShift Container Platform 구성 요소에 대한 컨테이너 이미지를 제공하는 Quay.io를 포함하여 인증 기관에서 제공하는 서비스로 인증할 수 있습니다.

15

RHCOS(Red Hat Enterprise Linux CoreOS)의 core 사용자에 대한 SSH 공용 키입니다.



참고

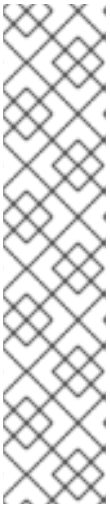
설치 디버깅 또는 재해 복구를 수행하려는 프로덕션 **OpenShift Container Platform** 클러스터의 경우 **ssh-agent** 프로세스가 사용하는 **SSH** 키를 지정합니다.

19.1.9.2. 설치 중 클러스터 단위 프록시 구성

프로덕션 환경에서는 인터넷에 대한 직접 액세스를 거부하고 대신 **HTTP** 또는 **HTTPS** 프록시를 사용할 수 있습니다. **install-config.yaml** 파일에서 프록시 설정을 구성하여 프록시가 사용되도록 새 **OpenShift Container Platform** 클러스터를 구성할 수 있습니다.

사전 요구 사항

- 기존 **install-config.yaml** 파일이 있습니다.
- 클러스터에서 액세스해야 하는 사이트를 검토하고 프록시를 바이패스해야 하는지 확인했습니다. 기본적으로 호스팅 클라우드 공급자 **API**에 대한 호출을 포함하여 모든 클러스터 발신(**Egress**) 트래픽이 프록시됩니다. 필요한 경우 프록시를 바이패스하기 위해 **Proxy** 오브젝트의 **spec.noProxy** 필드에 사이트를 추가했습니다.



참고

Proxy 오브젝트의 **status.noProxy** 필드는 설치 구성에 있는 **networking.machineNetwork[].cidr, networking.clusterNetwork[].cidr, networking.serviceNetwork[]** 필드의 값으로 채워집니다.

Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure 및 **Red Hat OpenStack Platform (RHOSP)**에 설치하는 경우 **Proxy** 오브젝트 **status.noProxy** 필드도 인스턴스 메타데이터 끝점(**169.254.169.254**)로 채워집니다.

절차

1. **install-config.yaml** 파일을 편집하고 프록시 설정을 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
```

```
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1

클러스터 외부에서 **HTTP** 연결을 구축하는 데 사용할 프록시 **URL**입니다. **URL** 스키마는 **http**여야 합니다.

2

클러스터 외부에서 **HTTPS** 연결을 구축하는 데 사용할 프록시 **URL**입니다.

3

대상 도메인 이름, **IP** 주소 또는 프록시에서 제외할 기타 네트워크 **CIDR**로 이루어진 쉼표로 구분된 목록입니다. 하위 도메인과 일치하려면 도메인 앞에 **.**을 입력합니다. 예를 들어, **.y.com**은 **x.y.com**과 일치하지만 **y.com**은 일치하지 않습니다. ***를 사용하여 모든 대상에 대해 프록시를 바이패스합니다.

4

이 값을 제공하면 설치 프로그램에서 추가 **CA** 인증서를 보유할 **openshift -config** 네임스페이스에 **user-ca- bundle** 이라는 구성 맵을 생성합니다. **additionalTrustBundle** 및 하나 이상의 프록시 설정을 제공하는 경우 프록시 오브젝트는 **trustedCA** 필드의 **user-ca-bundle** 구성 맵을 참조하도록 구성됩니다. 그러면 **Cluster Network Operator**에서 **trustedCA** 매개변수에 대해 지정된 콘텐츠를 **RHCOS** 신뢰 번들과 병합하는 **trusted-ca-bundle** 구성 맵을 생성합니다. 프록시의 **ID** 인증서를 **RHCOS** 트러스트 번들에 있는 기관에서 서명하지 않은 경우 **additionalTrustBundle** 필드가 있어야 합니다.

참고

설치 프로그램에서 프록시 **adinessEndpoints** 필드를 지원하지 않습니다.

2. 파일을 저장해 놓고 **OpenShift Container Platform**을 설치할 때 참조하십시오.

제공되는 **install-config.yaml** 파일의 프록시 설정을 사용하는 **cluster**라는 이름의 클러스터 전체 프록시가 설치 프로그램에 의해 생성됩니다. 프록시 설정을 제공하지 않아도 **cluster Proxy** 오브젝트는 계속 생성되지만 **spec**은 **nil**이 됩니다.

참고

cluster라는 Proxy 오브젝트만 지원되며 추가 프록시는 생성할 수 없습니다.

19.1.9.3. 3개의 노드 클러스터 구성

필요한 경우 세 개의 컨트롤 플레인 시스템으로 구성된 베어 메탈 클러스터에 제로 컴퓨팅 머신을 배포할 수 있습니다. 이를 통해 클러스터 관리자와 개발자들이 테스트, 개발, 프로덕션에 사용할 수 있는 소형화되고 리소스 효율이 높은 클러스터를 제공합니다.

3-노드 OpenShift Container Platform 환경에서 세 개의 컨트롤 플레인 머신을 예약할 수 있습니다. 즉, 애플리케이션 워크로드가 해당 플랫폼에서 실행되도록 예약됩니다.

사전 요구 사항

- 기존 install-config.yaml 파일이 있습니다.

프로세스

- install-config.yaml 파일에서 다음 compute 스탠자에 표시된 대로 컴퓨팅 복제본 수가 0으로 설정되어 있는지 확인합니다.

```
compute:
- name: worker
  platform: {}
  replicas: 0
```

참고

배포 중인 컴퓨팅 머신 수에 관계없이 사용자 프로비저닝 인프라에 OpenShift Container Platform을 설치할 때 컴퓨팅 머신의 replicas 매개변수 값을 0으로 설정해야 합니다. 설치 프로그램에서 제공하는 설치에서 매개 변수는 클러스터가 생성 및 관리하는 컴퓨팅 머신 수를 제어합니다. 이 설정은 컴퓨팅 시스템이 수동으로 배포되는 사용자 프로비저닝 설치에는 적용되지 않습니다.

3-노드 클러스터 설치의 경우 다음 단계를 따르십시오.

- 컴퓨팅 노드가 0인 3-노드 클러스터를 배포하는 경우 Ingress 컨트롤러 Pod는 컨트롤 플레인 노드에서 실행됩니다. 3-노드 클러스터 배포에서 HTTP 및 HTTPS 트래픽을 컨트롤 플레인 노

드로 라우팅하도록 애플리케이션 인그레스 로드 밸런서를 구성해야 합니다. 자세한 내용은 *사용자 프로비저닝 인프라 섹션에 대한 로드 밸런싱 요구 사항* 섹션을 참조하십시오.

- 다음 절차에서 **Kubernetes** 매니페스트 파일을 생성할 때 **<installation_directory>/manifests/cluster-scheduler-02-config.yml** 파일의 **mastersSchedulable** 매개변수가 **true**로 설정되어 있는지 확인합니다. 이렇게 하면 애플리케이션 워크로드를 컨트롤 플레인 노드에서 실행할 수 있습니다.

- **RHCOS(Red Hat Enterprise Linux CoreOS)** 시스템을 생성할 때 컴퓨팅 노드를 배포하지 마십시오.

19.1.10. Kubernetes 매니페스트 및 Ignition 설정 파일 생성

일부 클러스터 정의 파일을 수정하고 클러스터 시스템을 수동으로 시작해야 하므로 클러스터가 시스템을 구성하는 데 필요한 **Kubernetes** 매니페스트 및 **Ignition** 구성 파일을 사용자가 생성해야 합니다.

설치 구성 파일은 **Kubernetes** 매니페스트로 변환됩니다. 매니페스트는 나중에 클러스터 머신을 구성하는 데 사용되는 **Ignition** 구성 파일로 래핑됩니다.

중요

- **OpenShift Container Platform** 설치 프로그램에서 생성하는 **Ignition** 구성 파일에 **24**시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 **24**시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 **kubelet** 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청**(CSR)**을 수동으로 승인해야 합니다. 자세한 내용은 ***Recovering from expired control plane certificates*** 문서를 참조하십시오.

- 클러스터를 설치한 후 **24**시간에서 **22**시간까지의 인증서가 교체되기 때문에 생성된 후 **12**시간 이내에 **Ignition** 구성 파일을 사용하는 것이 좋습니다. **12**시간 이내에 **Ignition** 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

사전 요구 사항

- **OpenShift Container Platform** 설치 프로그램을 가져오셨습니다.

- install-config.yaml 설치 구성 파일을 생성하셨습니다.

프로세스

1. OpenShift Container Platform 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터에 대한 Kubernetes 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 <installation_directory>는 사용자가 만든 install-config.yaml 파일이 포함된 설치 디렉터리를 지정합니다.

주의

3 노드 클러스터를 실행 중이면 다음 단계를 건너 뛰어 컨트롤 플레인 노드 일정을 계획할 수 있도록 하십시오.

중요

예약할 수 없는 기본에서 컨트롤 플레인 노드를 구성하면 추가 서브스크립션이 필요합니다. 이는 컨트롤 플레인 노드가 작업자 노드가 되기 때문입니다.

2. <installation_directory>/manifests/cluster-scheduler-02-config.yml Kubernetes 매니페스트 파일의 mastersSchedulable 매개변수가 false로 설정되어 있는지 확인합니다. 이 설정으로 인해 컨트롤 플레인 머신에서 포드가 예약되지 않습니다.

a. <installation_directory>/manifests/cluster-scheduler-02-config.yml 파일을 엽니다.

b. mastersSchedulable 매개변수를 찾아서 값을 False로 설정되어 있는지 확인합니다.

c.

파일을 저장하고 종료합니다.

3.

Ignition 구성 파일을 생성하려면 설치 프로그램이 포함된 디렉터리에서 다음 명령을 실행합니다.

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1

<installation_directory>는 동일한 설치 디렉터리를 지정합니다.

설치 디렉터리의 부트스트랩, 컨트롤 플레인 및 컴퓨팅 노드에 대한 **Ignition** 구성 파일이 생성됩니다. **kubeadmin-password** 및 **kubeconfig** 파일은 **./<installation_directory>/auth** 디렉터리에 생성됩니다.

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

19.1.11. RHCOS 설치 및 OpenShift Container Platform 부트스트랩 프로세스 시작

프로비저닝하는 베어메탈 인프라에 **OpenShift Container Platform**을 설치하려면 머신에 **RHCOS(Red Hat Enterprise Linux CoreOS)**를 설치해야 합니다. **RHCOS**를 설치할 때 설치 중인 머신 유형에 대해 **OpenShift Container Platform** 설치 프로그램에서 생성한 **Ignition** 구성 파일을 제공해야 합니다. 적합한 네트워킹, **DNS** 및 로드 밸런싱 인프라를 구성한 경우 **RHCOS** 머신이 재부팅된 후 **OpenShift Container Platform** 부트스트랩 프로세스가 자동으로 시작됩니다.

단계에 따라 **ISO** 이미지 또는 네트워크 **PXE** 부팅을 사용하여 시스템에 **RHCOS**를 설치합니다.

참고

이 설치 문서에 포함된 컴퓨팅 노드 배포 단계는 **RHCOS**에 따라 다릅니다. **RHEL** 기반 컴퓨팅 머신을 사용하기로 선택한 경우 시스템 업데이트 수행, 패치 적용 및 기타 필요한 모든 작업 실행을 포함한 모든 운영 체제의 라이프 사이클 관리 및 유지 관리에 대한 책임이 있습니다. **RHEL 7** 컴퓨팅 머신 사용은 더 이상 사용되지 않으며 향후 **OpenShift Container Platform 4** 릴리스에서 제거될 예정입니다.

다음 방법을 사용하여 ISO 및 PXE 설치 중에 RHCOS를 구성할 수 있습니다.

- 커널 인수: 커널 인수를 사용하여 설치 관련 정보를 제공할 수 있습니다. 예를 들어 HTTP 서버에 업로드한 RHCOS 설치 파일의 위치와 설치 중인 노드 유형에 대한 Ignition 구성 파일의 위치를 지정할 수 있습니다. PXE 설치의 경우 APPEND 매개 변수를 사용하여 라이브 설치 프로그램의 커널에 인수를 전달할 수 있습니다. ISO 설치의 경우는 라이브 설치 부팅 프로세스를 중단하고 커널 매개 변수를 추가할 수 있습니다. 두 경우 모두 특정 coreos.inst.* 인수를 사용하여 라이브 설치 프로그램을 지시 할 수 있을 뿐 만 아니라 표준 커널 서비스를 활성화/비활성화하기 위해 표준 설치 부팅 인수를 사용할 수 있습니다.

- Ignition 구성: OpenShift Container Platform Ignition 구성 파일(*.ign)은 설치 중인 노드 유형에 따라 다릅니다. RHCOS 설치 중에 부트스트랩, 컨트롤 플레인 또는 컴퓨팅 노드 Ignition 구성 파일의 위치를 전달하여 첫 번째 부팅 시 적용됩니다. 특별한 경우에는 라이브 시스템으로 전달할 별도의 제한된 Ignition 설정을 만들 수 있습니다. 이 Ignition 설정은 설치 완료 후 프로비저닝 시스템에 설치가 성공적으로 완료되었는지를 보고하는 것과 같은 일련의 작업을 수행할 수 있습니다. 이러한 특수 Ignition 구성은 설치된 시스템의 처음 부팅 시 적용되는 coreos-installer에 의해 소비됩니다. 라이브 ISO에 표준 컨트롤 플레인 및 컴퓨팅 노드 Ignition 구성을 직접 제공하지 마십시오.

- coreos-installer : 처음 부팅하기 전에 다양한 방법으로 영구 시스템을 준비 할 수 있도록 쉘 프롬프트에서 라이브 ISO 설치 프로그램을 시작할 수 있습니다. coreos-installer 명령을 실행하여 추가하는 다양한 아티팩트를 식별하고 디스크 파티션을 사용하여 네트워크를 설정할 수 있습니다. 경우에 따라 라이브 시스템에서 기능을 구성하고 설치된 시스템에 복사할 수도 있습니다.

ISO 또는 PXE 설치 사용 여부는 상황에 따라 달라집니다. PXE 설치에는 사용 가능한 DHCP 서비스와 추가 준비가 필요하지만 설치 프로세스를 보다 자동화할 수 있습니다. ISO 설치는 주로 수동적인 프로세스에서 여러 시스템을 설정하는 경우 불편할 수 있습니다.

참고

OpenShift Container Platform 4.6부터 RHCOS ISO 및 기타 설치 아티팩트는 4K 섹터 디스크에 설치를 지원합니다.

19.1.11.1. ISO 이미지를 사용하여 RHCOS 설치

ISO 이미지를 사용하여 시스템에 RHCOS을 설치할 수 있습니다.

사전 요구 사항

- 클러스터에 대한 Ignition 구성 파일이 생성되어 있습니다.
- 적합한 네트워크, DNS 및 로드 밸런싱 인프라가 구성되어 있습니다.
- 사용자 컴퓨터에서 액세스할 수 있고 사용자가 생성한 시스템에서 액세스할 수 있는 HTTP 서버에 대한 액세스 권한을 확보합니다.
- 네트워킹 및 디스크 파티션과 같은 기능을 구성하는 다양한 방법에 대해서는 *고급 RHCOS 설치 구성*섹션을 살펴보십시오.

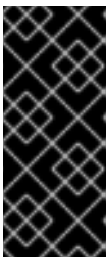
프로세스

1. 각 Ignition 구성 파일에 대해 SHA512 다이제스트를 가져옵니다. 예를 들어 Linux를 실행하는 시스템에서 다음을 사용하여 bootstrap.ign Ignition 구성 파일의 SHA512 다이제스트를 가져올 수 있습니다.

```
$ sha512sum <installation_directory>/bootstrap.ign
```

   다이제스트는 클러스터 노드에서 Ignition 구성 파일의 신뢰성을 검증하기 위해 이후 단계에서 coreos-installer에 제공됩니다.

2. 설치 프로그램에서 생성한 부트스트랩, 컨트롤 플레인, 컴퓨팅 노드 Ignition 구성 파일을 HTTP 서버에 업로드합니다. 해당 파일의 URL을 기록해 둡니다.



   중요

   HTTP 서버에 저장하기 전에 Ignition 구성에서 구성 설정을 추가하거나 변경할 수 있습니다. 설치를 마친 후 클러스터에 컴퓨팅 시스템을 더 추가하려면 Ignition 구성 파일을 삭제하지 마십시오.

3. 설치 호스트에서 Ignition 구성 파일을 URL에서 사용할 수 있는지 확인합니다. 다음 예에서는 부트스트랩 노드에 대한 Ignition 구성 파일을 가져옵니다.

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

   출력 예

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-
rsa...
```

명령에서 **bootstrap.ign**을 **master.ign** 또는 **worker.ign**으로 교체하여 컨트롤 플레인 및 컴퓨팅 노드의 **Ignition** 구성 파일도 사용할 수 있는지 확인합니다.

4. **RHCOS** 이미지 미러 페이지에서 운영 체제 인스턴스 설치 방법에 필요한 **RHCOS** 이미지를 얻을 수 있지만 올바른 버전의 **RHCOS** 이미지를 얻는 것이 좋습니다. **openshift-install** 명령 출력에서 사용할 수 있습니다.

```
$ openshift-install coreos print-stream-json | grep '\.iso[^.]'
```

출력 예

```
"location": "<url>/art/storage/releases/rhcos-4.9-aarch64/<release>/aarch64/rhcos-
<release>-live.aarch64.iso",
"location": "<url>/art/storage/releases/rhcos-4.9-ppc64le/<release>/ppc64le/rhcos-
<release>-live.ppc64le.iso",
"location": "<url>/art/storage/releases/rhcos-4.9-s390x/<release>/s390x/rhcos-
<release>-live.s390x.iso",
"location": "<url>/art/storage/releases/rhcos-4.9/<release>/x86_64/rhcos-<release>-
live.x86_64.iso",
```

중요

RHCOS 이미지는 **OpenShift Container Platform** 릴리스에 따라 변경되지 않을 수 있습니다. 설치하는 **OpenShift Container Platform** 버전과 같거나 그 이하의 버전 중 가장 최신 버전의 이미지를 다운로드해야 합니다. 사용 가능한 경우 **OpenShift Container Platform** 버전과 일치하는 이미지 버전을 사용합니다. 이 프로세스에는 **ISO** 이미지만 사용하십시오. 이 설치 유형에서는 **RHCOS qcow2** 이미지가 지원되지 않습니다.

ISO 파일 이름은 다음 예와 유사합니다.

```
rhcos-<version>-live.<architecture>.iso
```

5. ISO를 사용하여 RHCOS 설치를 시작합니다. 다음 설치 옵션 중 하나를 사용합니다.

- ISO 이미지를 디스크에 굽고 직접 부팅합니다.
- LOM(Lightweight-out Management) 인터페이스를 사용하여 ISO 리디렉션을 사용합니다.

6. 옵션을 지정하거나 라이브 부팅 시퀀스를 중단하지 않고 RHCOS ISO 이미지를 부팅합니다. 설치 프로그램이 RHCOS 라이브 환경에서 쉘 프롬프트로 부팅될 때까지 기다립니다.

참고

커널 인수를 추가하기 위해 RHCOS 설치 부팅 프로세스를 중단할 수 있습니다. 하지만 이 ISO 프로세스에서는 커널 인수를 추가하지 않고 다음 단계에 설명된 대로 coreos-installer 명령을 사용해야 합니다.

7. coreos-installer 명령을 실행하고 설치 요구 사항을 충족하는 옵션을 지정합니다. 최소한 노드 유형에 대한 Ignition 구성 파일과 설치할 장치를 가리키는 URL을 지정해야 합니다.

```
$ sudo coreos-installer install --ignition-url=http://<HTTP_server>/<node_type>.ign
<device> --ignition-hash=sha512-<digest> 1 2
```

1 1

core 사용자에게 설치를 수행하는 데 필요한 root 권한이 없으므로 sudo를 사용하여 coreos-installer 명령을 실행해야 합니다.

2

클러스터 노드에서 Ignition 구성 파일을 HTTP URL을 통해 가져오려면 --ignition-hash 옵션이 필요합니다. <digest>는 이전 단계에서 얻은 Ignition 구성 파일 SHA512 다이제스트입니다.

참고

TLS를 사용하는 HTTPS 서버를 통해 Ignition 구성 파일을 제공하려는 경우 coreos-installer를 실행하기 전에 내부 인증 기관(CA)을 시스템 신뢰 저장소에 추가할 수 있습니다.

다음 예제에서는 /dev/sda 장치에 부트스트랩 노드 설치를 초기화합니다. 부트스트랩 노드의 Ignition 구성 파일은 IP 주소 192.168.1.2가 있는 HTTP 웹 서버에서 가져옵니다.

```
$ sudo coreos-installer install --ignition-
url=http://192.168.1.2:80/installation_directory/bootstrap.ign /dev/sda --ignition-
hash=sha512-
a5a2d43879223273c9b60af66b44202a1d1248fc01cf156c46d4a79f552b6bad47bc8cc78dd
f0116e80c59d2ea9e32ba53bc807afbca581aa059311def2c3e3b
```

8. 머신 콘솔에서 RHCOS 설치 진행률을 모니터링합니다.

중요

OpenShift Container Platform 설치를 시작하기 전에 각 노드에서 성공적으로 설치되었는지 확인합니다. 설치 프로세스를 관찰하면 발생할 수 있는 RHCOS 설치 문제의 원인을 파악하는 데 도움이 될 수 있습니다.

9. RHCOS를 설치한 후 시스템을 재부팅해야 합니다. 시스템이 재부팅되는 동안 지정한 Ignition 구성 파일이 적용됩니다.

10. 계속해서 클러스터에 대한 나머지 시스템을 모두 생성합니다.

중요

이때 부트스트랩 및 컨트롤 플레인 시스템을 생성해야 합니다. 컨트롤 플레인 시스템이 예약 가능하지 않은 경우 OpenShift Container Platform을 설치하기 전에 두 개 이상의 컴퓨팅 시스템도 생성합니다.

적합한 네트워킹, DNS 및 로드 밸런싱 인프라가 있는 경우 RHCOS 노드가 재부팅된 후 OpenShift Container Platform 부트스트랩 프로세스가 자동으로 시작됩니다.

참고

RHCOS 노드에는 core 사용자의 기본 암호가 포함되지어 있지 않습니다. install_config.yaml 파일에 지정한 공개 키에 대한 액세스 권한이 있는 사용자로 ssh core@<node>.<cluster_name>.<base_domain>을 실행하여 노드에 액세스할 수 있습니다. RHCOS를 실행하는 OpenShift Container Platform 4 클러스터 노드는 변경할 수 없으며 Operator를 통해 클러스터 변경 사항을 적용합니다. SSH를 사용하여 클러스터 노드에 액세스하는 것은 권장되지 않습니다. 그러나 설치 문제를 조사할 때 OpenShift Container Platform API를 사용할 수 없거나 kubelet이 대상 노드에서 제대로 작동하지 않는 경우 디버깅 또는 재해 복구에 SSH 액세스가 필요할 수 있습니다.

19.1.11.2. PXE 또는 iPXE 부팅을 사용하여 RHCOS 설치

PXE 또는 iPXE 부팅을 사용하여 시스템에 RHCOS를 설치할 수 있습니다.

사전 요구 사항

- 클러스터에 대한 Ignition 구성 파일이 생성되어 있습니다.
- 적합한 네트워크, DNS 및 로드 밸런싱 인프라가 구성되어 있습니다.
- 적합한 PXE 또는 iPXE 인프라가 구성되어 있습니다.
- 사용자 컴퓨터에서 액세스할 수 있고 사용자가 생성한 시스템에서 액세스할 수 있는 HTTP 서버에 대한 액세스 권한을 확보합니다.
- 네트워킹 및 디스크 파티션과 같은 기능을 구성하는 다양한 방법에 대해서는 *고급 RHCOS 설치 구성*섹션을 살펴보십시오.

프로세스

1. 설치 프로그램에서 생성한 부트스트랩, 컨트롤 플레인, 컴퓨팅 노드 Ignition 구성 파일을 HTTP 서버에 업로드합니다. 해당 파일의 URL을 기록해 둡니다.

중요

HTTP 서버에 저장하기 전에 Ignition 구성에서 구성 설정을 추가하거나 변경할 수 있습니다. 설치를 마친 후 클러스터에 컴퓨팅 시스템을 더 추가하려면 Ignition 구성 파일을 삭제하지 마십시오.

2. 설치 호스트에서 Ignition 구성 파일을 URL에서 사용할 수 있는지 확인합니다. 다음 예에서는 부트스트랩 노드에 대한 Ignition 구성 파일을 가져옵니다.

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

출력 예

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-
rsa...
```

명령에서 bootstrap.ign을 master.ign 또는 worker.ign으로 교체하여 컨트롤 플레인 및 컴퓨팅 노드의 Ignition 구성 파일도 사용할 수 있는지 확인합니다.

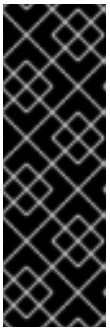
3. RHCOS 커널,initramfs 및 rootfs 파일을 RHCOS 이미지 미러 페이지에서 원하는 운영 체제 인스턴스를 설치하는 데 필요한 경우, RHCOS 파일의 올바른 버전을 가져오는 권장 방법은 openshift-install 명령 출력에서 얻을 수 있습니다.

```
$ openshift-install coreos print-stream-json | grep -Eo '"https.*(kernel-
|initramfs.|rootfs.)\w+(\.img)?"'
```

출력 예

```
"<url>/art/storage/releases/rhcos-4.9-aarch64/<release>/aarch64/rhcos-<release>-live-
kernel-aarch64"
"<url>/art/storage/releases/rhcos-4.9-aarch64/<release>/aarch64/rhcos-<release>-live-
initramfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.9-aarch64/<release>/aarch64/rhcos-<release>-live-
rootfs.aarch64.img"
```

```
"<url>/art/storage/releases/rhcos-4.9-ppc64le/49.84.202110081256-0/ppc64le/rhcos-
<release>-live-kernel-ppc64le"
"<url>/art/storage/releases/rhcos-4.9-ppc64le/<release>/ppc64le/rhcos-<release>-live-
initramfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.9-ppc64le/<release>/ppc64le/rhcos-<release>-live-
rootfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.9-s390x/<release>/s390x/rhcos-<release>-live-
kernel-s390x"
"<url>/art/storage/releases/rhcos-4.9-s390x/<release>/s390x/rhcos-<release>-live-
initramfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.9-s390x/<release>/s390x/rhcos-<release>-live-
rootfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.9/<release>/x86_64/rhcos-<release>-live-kernel-
x86_64"
"<url>/art/storage/releases/rhcos-4.9/<release>/x86_64/rhcos-<release>-live-
initramfs.x86_64.img"
"<url>/art/storage/releases/rhcos-4.9/<release>/x86_64/rhcos-<release>-live-
rootfs.x86_64.img"
```



중요

OpenShift Container Platform의 모든 릴리스에서 **RHCOS** 아티팩트가 변경되지 않을 수도 있습니다. 설치하는 **OpenShift Container Platform** 버전과 같거나 그 이하의 버전 중 가장 최신 버전의 이미지를 다운로드해야 합니다. 이 프로시저에는 아래 설명된 적절한 **kernel, initramfs** 및 **rootfs** 아티팩트만 사용하십시오. 이 설치 유형에서는 **RHCOS QCOW2** 이미지가 지원되지 않습니다.

OpenShift Container Platform 버전 번호가 파일 이름에 포함됩니다. 다음 예와 유사합니다.

- **kernel: rhcos-<version>-live-kernel-<architecture>**
- **initramfs: rhcos-<version>-live-initramfs.<architecture>.img**
- **rootfs: rhcos-<version>-live-rootfs.<architecture>.img**

4. **rootfs, kernel** 및 **initramfs** 파일을 **HTTP** 서버에 업로드합니다.

중요

설치를 마친 후 클러스터에 컴퓨팅 시스템을 더 추가하려면 **Ignition** 구성 파일을 삭제하지 마십시오.

5. **RHCOS**가 설치된 후 시스템이 로컬 디스크에서 부팅되도록 네트워크 부팅 인프라를 구성하십시오.

6. **RHCOS** 이미지에 대한 **PXE** 또는 **iPXE** 설치를 구성하고 설치를 시작하십시오.

사용 환경에 대한 다음 예시 메뉴 항목 중 하나를 수정하고, 이미지 및 **Ignition** 파일에 적절히 접근할 수 있는지 확인하십시오.

- **PXE**의 경우:

```
DEFAULT pxeboot
TIMEOUT 20
PROMPT 0
LABEL pxeboot
    KERNEL http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> 1
    APPEND initrd=http://<HTTP_server>/rhcos-<version>-live-initramfs.
<architecture>.img coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-
rootfs.<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign 2 3
```

1 1 **HTTP** 서버에 업로드한 라이브 **kernel** 파일의 위치를 지정합니다. **URL**은 **HTTP**, **TFTP** 또는 **FTP**여야 합니다. **HTTPS**와 **NFS**는 지원되지 않습니다.

2 **NIC**를 여러 개 사용하는 경우 **ip** 옵션에 단일 인터페이스를 지정합니다. 예를 들어, **eno1**라는 **NIC**에서 **DHCP**를 사용하려면 **ip=eno1:dhcp**를 설정하십시오.

3 **HTTP** 서버에 업로드한 **RHCOS** 파일의 위치를 지정합니다. **initrd** 매개변수 값은 **initramfs** 파일의 위치, **coreos.live.rootfs_url** 매개변수 값은 **rootfs** 파일의 위치, **coreos.inst.ignition_url** 매개변수 값은 부트스트랩 **Ignition** 구성 파일의 위치입니다. **APPEND** 줄에 커널 인수를 더 추가하여 네트워킹 또는 기타 부팅 옵션도 구성할 수 있습니다.

참고

이 구성은 그래픽 콘솔이 있는 시스템에서 직렬 콘솔 액세스를 활성화하지 않습니다. 다른 콘솔을 구성하려면 **APPEND** 행에 하나 이상의 **console=** 인수를 추가합니다. 예를 들어 **console=tty0 console=ttyS0**을 추가하여 첫 번째 **PC** 직렬 포트를 기본 콘솔로 설정하고 그래픽 콘솔을 보조 콘솔로 설정합니다. 자세한 내용은 **Red Hat Enterprise Linux**에서 직렬 터미널 및/또는 콘솔 설정 방법을 참조하십시오.

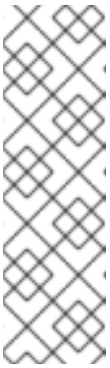
- **iPXE**의 경우 :

```
kernel http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> initrd=main
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign (1) (2)
initrd --name main http://<HTTP_server>/rhcos-<version>-live-initramfs.
<architecture>.img (3)
boot
```

(1) **HTTP** 서버에 업로드한 **RHCOS** 파일의 위치를 지정합니다. **kernel** 매개변수 값은 **kernel** 파일의 위치이고 **initrd=main** 인수는 **UEFI** 시스템에서 부팅하는 데 필요하며 **coreos.live.rootfs_url** 매개 변수 값은 **rootfs** 파일의 위치이며, **coreos.inst.ignition_url** 매개 변수 값은 부트스트랩 **Ignition** 설정 파일의 위치입니다.

(2) **NIC**를 여러 개 사용하는 경우 **ip** 옵션에 단일 인터페이스를 지정합니다. 예를 들어, **eno1**라는 **NIC**에서 **DHCP**를 사용하려면 **ip=eno1:dhcp**를 설정하십시오.

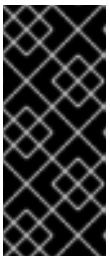
(3) **HTTP** 서버에 업로드한 **initramfs** 파일의 위치를 지정합니다.



참고

이 구성은 그래픽 콘솔이 있는 시스템에서 직렬 콘솔 액세스를 활성화하지 않습니다. 다른 콘솔을 구성하려면 **kernel** 행에 하나 이상의 **console=** 인수를 추가합니다. 예를 들어 **console=tty0 console=ttyS0**을 추가하여 첫 번째 **PC** 직렬 포트를 기본 콘솔로 설정하고 그래픽 콘솔을 보조 콘솔로 설정합니다. 자세한 내용은 **Red Hat Enterprise Linux**에서 직렬 터미널 및/또는 콘솔 설정 방법을 참조하십시오.

7. 머신 콘솔에서 RHCOS 설치 진행률을 모니터링합니다.



중요

OpenShift Container Platform 설치를 시작하기 전에 각 노드에서 성공적으로 설치되었는지 확인합니다. 설치 프로세스를 관찰하면 발생할 수 있는 RHCOS 설치 문제의 원인을 파악하는 데 도움이 될 수 있습니다.

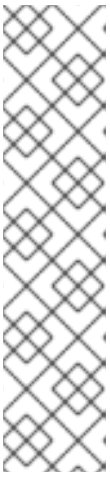
8. RHCOS가 설치되면 시스템을 재부팅합니다. 시스템이 재부팅되는 동안 지정한 Ignition 구성 파일이 적용됩니다.

9. 클러스터용 시스템 생성을 계속합니다.

중요

이때 부트스트랩 및 컨트롤 플레인 시스템을 생성해야 합니다. 컨트롤 플레인 시스템이 예약 가능하지 않은 경우 클러스터를 설치하기 전에 두 개 이상의 컴퓨팅 시스템도 생성합니다.

적합한 네트워킹, DNS 및 로드 밸런싱 인프라가 있는 경우 RHCOS 노드가 재부팅된 후 OpenShift Container Platform 부트스트랩 프로세스가 자동으로 시작됩니다.



참고

RHCOS 노드에는 core 사용자의 기본 암호가 포함되지어 있지 않습니다. install_config.yaml 파일에 지정한 공개 키에 대한 액세스 권한이 있는 사용자로 ssh core@<node>.<cluster_name>.<base_domain>을 실행하여 노드에 액세스할 수 있습니다. RHCOS를 실행하는 OpenShift Container Platform 4 클러스터 노드는 변경할 수 없으며 Operator를 통해 클러스터 변경 사항을 적용합니다. SSH를 사용하여 클러스터 노드에 액세스하는 것은 권장되지 않습니다. 그러나 설치 문제를 조사할 때 OpenShift Container Platform API를 사용할 수 없거나 kubelet이 대상 노드에서 제대로 작동하지 않는 경우 디버깅 또는 재해 복구에 SSH 액세스가 필요할 수 있습니다.

19.1.11.3. 고급 RHCOS 설치 구성 옵션

OpenShift Container Platform용 RHCOS (Red Hat Enterprise Linux CoreOS) 노드를 수동으로 프로비저닝하는 주요 이점은 기본 OpenShift Container Platform 설치 방법을 통해 사용할 수 없는 구성을

수행할 수 있는 것입니다. 이 섹션에서는 다음과 같은 방법을 사용하여 수행할 수있는 몇 가지 구성에 대해 설명합니다.

- 라이브 설치 프로그램에 커널 인수 전달
- 라이브 시스템에서 수동으로 **coreos-installer** 실행
- ISO에 Ignition 구성 포함

이 섹션에 설명되어 있는 수동 **Red Hat Enterprise Linux CoreOS(RHCOS)** 설치에 대한 고급 구성 항목은 디스크 파티션 설정, 네트워킹 및 다양한 방식의 **Ignition** 구성 사용과 관련되어 있습니다.

19.1.11.3.1. PXE 및 ISO 설치를 위한 고급 네트워크 옵션 사용

OpenShift Container Platform 노드의 네트워크는 기본적으로 **DHCP**를 사용하여 필요한 모든 구성 설정을 수집합니다. 고정 **IP** 주소를 설정하거나 본딩과 같은 특정 설정을 구성하려면 다음 중 하나의 방법으로 수행할 수 있습니다.

- 라이브 설치 프로그램을 시작할 때 특수 커널 매개 변수를 전달합니다.
- 머신 구성을 사용하여 네트워크 파일을 설치된 시스템에 복사합니다.
- 라이브 설치 프로그램 쉘 프롬프트에서 네트워크를 구성한 다음 설치된 시스템에 복사하여 설치한 시스템을 처음 시작할 때 사용하도록합니다.

PXE 또는 **iPXE** 설치를 구성하려면 다음 옵션 중 하나를 사용합니다.

- "고급 **RHCOS** 설치 참조" 표를 참조하십시오.
- 머신 구성을 사용하여 네트워크 파일을 설치된 시스템에 복사합니다.

다음 프로세스에 따라 **ISO** 설치를 구성합니다.

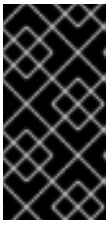
절차

1. ISO 설치 프로그램을 시작합니다.

2. 라이브 시스템 쉘 프롬프트에서 사용 가능한 RHEL 도구 (예: nmcli 또는 nmtui)를 사용하여 라이브 시스템의 네트워킹를 구성합니다.

3. coreos-installer 명령을 실행하여 시스템을 설치하고 --copy-network 옵션을 추가하여 네트워크 구성을 복사합니다. 예를 들면 다음과 같습니다.

```
$ sudo coreos-installer install --copy-network \
    --ignition-url=http://host/worker.ign /dev/sda
```



중요

copy-network 옵션은 /etc/NetworkManager/system-connections에 있는 네트워킹 구성만 복사합니다. 특히 시스템 호스트 이름을 복사하지 않습니다.

4. 설치된 시스템으로 재부팅하십시오.

추가 리소스

- nmcli 및 nmtui 툴에 대한 자세한 내용은 RHEL 8 설명서에서 nmcli로 시작하기 및 nmtui로 시작하기를 참조하십시오.

19.1.11.3.2. 디스크 파티션 설정

디스크 파티션은 RHCOS(Red Hat Enterprise Linux CoreOS) 설치 중에 OpenShift Container Platform 클러스터 노드에 생성됩니다. 특정 아키텍처의 각 RHCOS 노드는 기본 파티션 구성을 재정의하지 않는 한 동일한 파티션 레이아웃을 사용합니다. RHCOS 설치 중에 대상 장치에서 사용 가능한 나머지 공간을 사용하도록 루트 파일 시스템의 크기가 증가합니다.

OpenShift Container Platform 클러스터 노드에 RHCOS를 설치할 때 다음과 같은 두 가지 경우 기본 파티셔닝을 재정의할 수 있습니다.

- 별도의 파티션 생성: 빈 디스크에 그린 필드 설치의 경우 파티션에 별도의 스토리지를 추가할 수 있습니다. 이는 /var 또는 /var의 하위 디렉터리 (예: /var/lib/etcd)중 하나를 별도의 파티션

으로 만드는 경우에 공식적으로 지원되지만 둘 다 지원되지는 않습니다.

중요

100GB 이상의 디스크 크기 및 특히 **1TB**보다 큰 디스크 크기의 경우 별도의 **/var** 파티션을 만듭니다. 자세한 내용은 " 별도의 **/var** 파티션 생성" 및 이 Red Hat 지식베이스 문서 를 참조하십시오.

중요

Kubernetes는 두 개의 파일 시스템 파티션만 지원합니다. 원래 구성에 둘 이상의 파티션을 추가하는 경우 **Kubernetes**는 모든 파티션을 모니터링할 수 없습니다.

- 기존 파티션 유지: 브라운 필드 설치에서 기존 노드에 **OpenShift Container Platform**을 재설치하고 이전 운영 체제에 설치된 데이터 파티션을 유지해야 하는 경우 기존 데이터 파티션을 유지할 수 있는 **coreos-installer**에 부팅 인수와 옵션이 모두 있습니다.

주의

사용자 지정 파티션을 사용하면 **OpenShift Container Platform**에서 해당 파티션을 모니터링하지 않거나 경고를 받을 수 있습니다. 기본 파티션을 재정의하는 경우 **OpenShift Container Platform**이 호스트 파일 시스템을 모니터링하는 방법에 대한 자세한 내용은 **OpenShift File System Monitoring(제거 조건) 이해**를 참조하십시오.

19.1.11.3.2.1. 별도의 /var 파티션 만들기

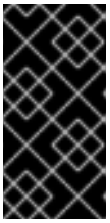
일반적으로 **RHCOS** 설치 중에 생성된 기본 디스크 파티션을 사용해야 합니다. 그러나 확장하려는 디렉토리에 별도의 파티션을 생성해야 하는 경우가 있습니다.

OpenShift 컨테이너 플랫폼은 **/var** 디렉토리 또는 **/var**의 하위 디렉터리 중 하나에 스토리지를 연결하는 단일 파티션의 추가를 지원합니다. 예를 들면 다음과 같습니다.

- **/var/lib/containers:** 시스템에 더 많은 이미지와 컨테이너가 추가됨에 따라 확장될 수 있는 컨테이너 관련 콘텐츠를 보관합니다.

- /var/lib/etcd: etcd 스토리지의 성능 최적화와 같은 목적으로 별도로 보관할 데이터를 보관합니다.

- /var: 감사 등의 목적에 맞게 별도로 분리하여 보관해야 하는 데이터를 보관합니다.



중요

100GB보다 큰 디스크 크기 및 특히 1TB보다 큰 디스크의 경우 별도의 /var 파티션을 만듭니다.

/var 디렉터리의 콘텐츠를 별도로 저장하면 필요에 따라 해당 영역에 대한 스토리지 확장을 보다 용이하게 하고 나중에 OpenShift Container Platform을 다시 설치하여 해당 데이터를 그대로 보존할 수 있습니다. 이 방법을 사용하면 모든 컨테이너를 다시 가져올 필요가 없으며 시스템을 업데이트할 때 대용량 로그 파일을 복사할 필요도 없습니다.

/var 디렉토리 또는 /var의 하위 디렉토리에 대해 별도의 파티션을 사용하면 분할된 디렉토리의 데이터 증가로 루트 파일 시스템이 채워지는 것을 방지할 수 있습니다.

다음 절차에서는 설치 준비 단계에서 노드 유형의 Ignition 구성 파일에 래핑되는 머신 구성 매니페스트를 추가하여 별도의 /var 파티션을 설정합니다.

절차

1. 설치 호스트에서 OpenShift Container Platform 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터에 대한 Kubernetes 매니페스트를 생성합니다.

```
$ openshift-install create manifests --dir <installation_directory>
```

2. 추가 파티션을 구성하는 Butane 구성을 생성합니다. 예를 들어 파일 이름을 $HOME/clusterconfig/98-var-partition.bu 로 지정하고, 디스크 장치 이름을 worker 시스템의 스토리지 장치 이름으로 변경하고 스토리지 크기를 적절하게 설정합니다. 이 예에서는 /var 디렉토리를 별도의 파티션에 배치합니다.

```
variant: openshift
version: 4.9.0
metadata:
  labels:
```

```
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1

파티션을 설정해야하는 디스크 저장 장치 이름입니다.

2

데이터 파티션을 부트 디스크에 추가할 때 최소 오프셋 값 **25000** 메비 바이트가 권장됩니다. 루트 파일 시스템은 지정된 오프셋까지 사용 가능한 모든 공간을 채우기 위해 자동으로 크기가 조정됩니다. 오프셋 값이 지정되지 않거나 지정된 값이 권장 최소값보다 작으면 생성되는 루트 파일 시스템의 크기가 너무 작아지고 **RHCOS**를 나중에 다시 설치할 때 데이터 파티션의 첫 번째 부분을 덮어 쓸 수 있습니다.

3

데이터 파티션의 크기(**MB**)입니다.

4

컨테이너 스토리지에 사용되는 파일 시스템에 **prjquota** 마운트 옵션을 활성화해야 합니다.

참고

별도의 **/var** 파티션을 만들 때 다른 인스턴스 유형에 동일한 장치 이름이 없는 경우 컴퓨팅 노드에 다른 인스턴스 유형을 사용할 수 없습니다.

3. **Butane** 구성에서 매니페스트를 생성하여 **clusterconfig/openshift** 디렉터리에 저장합니다. 예를 들어 다음 명령을 실행합니다.

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o
$HOME/clusterconfig/openshift/98-var-partition.yaml
```

4. Ignition 구성 파일을 만듭니다.

```
$ openshift-install create ignition-configs --dir <installation_directory> 1
```

1 <installation_directory>는 동일한 설치 디렉터리를 지정합니다.

설치 디렉터리의 부트스트랩, 컨트롤 플레인 및 컴퓨팅 노드에 대한 Ignition 구성 파일이 생성됩니다.

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

<installation_directory>/manifest 및 <installation_directory>/openshift 디렉터리의 파일은 98-var-partition 사용자 정의 MachineConfig 오브젝트가 포함된 파일을 포함하여 Ignition 구성 파일로 래핑됩니다.

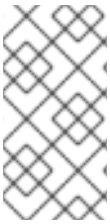
다음 단계

- RHCOS 설치 중에 Ignition 구성 파일을 참조하여 사용자 정의 디스크 파티션을 적용할 수 있습니다.

19.1.11.3.2.2. 기존 파티션 유지

ISO 설치의 경우 설치 프로그램이 하나 이상의 기존 파티션을 유지하도록하는 옵션을 coreos-installer 명령에 추가할 수 있습니다. PXE 설치의 경우 coreos.inst.* 옵션을 APPEND 매개 변수에 추가하여 파티션을 유지할 수 있습니다.

저장된 파티션은 기존 OpenShift Container Platform 시스템의 데이터 파티션이 될 수 있습니다. 파티션 레이블 또는 번호 중 하나로 보관하려는 디스크 파티션을 확인할 수 있습니다.



참고

기존 파티션을 저장하고 해당 파티션이 **RHCOS**를 위한 충분한 공간을 남겨 두지 않으면 저장된 파티션이 손상되지는 않지만 설치에 실패합니다.

ISO 설치 중 기존 파티션 유지

이 예에서는 파티션 레이블이 **data (data*)**로 시작하는 모든 파티션을 유지합니다.

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign \
      --save-partlabel 'data*' /dev/sda
```

다음 예는 디스크의 여섯 번째 **(6)** 파티션을 유지하는 방식으로 **coreos-installer**를 실행하는 방법을 보여줍니다.

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign \
      --save-partindex 6 /dev/sda
```

이 예에서는 파티션 **5** 이상을 유지합니다.

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign
      --save-partindex 5- /dev/sda
```

파티션 저장이 사용된 이전 예에서 **coreos-installer**는 파티션을 즉시 다시 만듭니다.

PXE 설치 중 기존 파티션 유지

이 **APPEND** 옵션은 파티션 레이블이 **'data'('data *')**로 시작하는 모든 파티션을 유지합니다.

```
coreos.inst.save_partlabel=data*
```

이 **APPEND** 옵션은 파티션 **5** 이상을 유지합니다.

```
coreos.inst.save_partindex=5-
```

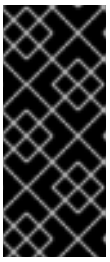
이 **APPEND** 옵션은 파티션 **6**을 유지합니다.

```
coreos.inst.save_partindex=6
```

19.1.11.3.3. Ignition 설정 확인

RHCOS 베어 메탈 설치를 수행할 때 제공할 수 있는 두 가지 유형의 Ignition 구성이 있으며 각 구성을 제공하는 이유도 각각 다릅니다.

- Permanent install Ignition config: 모든 수동 RHCOS 설치는 설치를 수행하기 위해 openshift-installer가 생성한 Ignition 구성 파일 (예: bootstrap.ign, master.ign 및 worker.ign) 중 하나를 전달해야 합니다.



중요

이러한 Ignition 구성 파일을 직접 수정하지 않는 것이 좋습니다. 이전 섹션의 예에 설명된 대로 Ignition 구성 파일로 래핑된 매니페스트 파일을 업데이트할 수 있습니다.

PXE 설치의 경우 coreos.inst.ignition_url= 옵션을 사용하여 APPEND 행에서 Ignition 구성을 전달합니다. ISO 설치의 경우 쉘 프롬프트에서 ISO를 시작한 후 coreos-installer 명령 줄에서 --ignition-url= 옵션을 사용하여 Ignition 구성을 식별합니다. 두 경우 모두 HTTP 및 HTTPS 프로토콜만 지원됩니다.

- Live install Ignition config :이 유형은 수동으로 작성되어 Red Hat에서 지원되지 않으므로 가능하면 사용하지 않도록 해야 합니다. 이 방법을 사용하면 Ignition 구성이 라이브 설치 미디어로 전달되고 부팅시 즉시 실행되며 RHCOS 시스템이 디스크에 설치되기 전후에 설치 작업을 수행합니다. 이 방법은 시스템 구성을 사용하여 실행할 수 없는 고급 파티션 설정과 같이 한 번만 수행하고 나중에 다시 적용할 필요가 없는 작업의 실행에만 사용해야 합니다.

PXE 또는 ISO 부팅의 경우 Ignition 설정을 만들고 ignition.config.url= 옵션에 APPEND를 실행하여 Ignition 설정 위치를 확인할 수 있습니다. 또한 ignition.firstboot ignition.platform.id = metal도 추가해야 합니다. 추가하지 않으면 ignition.config.url 옵션이 무시됩니다.

19.1.11.3.3.1. RHCOS ISO에 실시간 설치 Ignition 구성 포함

RHCOS ISO 이미지에 직접 라이브 설치 Ignition 구성을 포함할 수 있습니다. ISO 이미지를 부팅하면 내장된 구성이 자동으로 적용됩니다.

절차

1. 다음 이미지 미러 페이지에서 coreos-installer 바이너리를 다운로드합니다. https://mirror.openshift.com/pub/openshift-v4/clients/coreos-installer/latest/.

2. RHCOS ISO 이미지와 Ignition 구성 파일을 검색하고 이를 액세스 가능한 디렉터리 (예: /mnt)에 복사합니다.

```
# cp rhcos-<version>-live.x86_64.iso bootstrap.ign /mnt/
# chmod 644 /mnt/rhcos-<version>-live.x86_64.iso
```

3. 다음 명령을 실행하여 Ignition 구성을 ISO에 포함합니다.

```
# ./coreos-installer iso ignition embed -i /mnt/bootstrap.ign \
    /mnt/rhcos-<version>-live.x86_64.iso
```

이제 해당 ISO를 사용하여 지정된 라이브 설치 Ignition 구성을 사용하여 RHCOS를 설치할 수 있습니다.

중요

coreos-installer iso ignition embed를 사용하여 bootstrap.ign, master.ign 및 worker.ign 과 같이 openshift-installer에 의해 생성된 파일을 포함하는 것은 지원되지 않으며 권장되지 않습니다.

4. 포함된 Ignition 구성 내용을 표시하고 이를 파일로 보내려면 다음을 실행합니다.

```
# ./coreos-installer iso ignition show /mnt/rhcos-<version>-live.x86_64.iso >
mybootstrap.ign
```

```
# diff -s bootstrap.ign mybootstrap.ign
```

출력 예

```
Files bootstrap.ign and mybootstrap.ign are identical
```

5. Ignition 구성을 제거하고 다시 사용할 수 있도록 ISO를 초기 상태로 복원하려면 다음을 실행합니다.

```
# ./coreos-installer iso ignition remove /mnt/rhcos-<version>-live.x86_64.iso
```

이제 다른 Ignition 구성을 ISO에 포함하거나 초기 상태의 ISO를 사용할 수 있습니다.

19.1.11.3.4. 고급 RHCOS 설치 참조

여기서는 RHCOS(Red Hat Enterprise Linux CoreOS) 수동 설치 프로세스를 수정하는 데 사용할 수 있는 네트워킹 구성 및 기타 고급 옵션에 대해 설명합니다. 다음 표에서는 RHCOS 라이브 설치 프로그램 및 coreos-installer 명령과 함께 사용할 수있는 커널 인수 및 명령 줄 옵션에 대해 설명합니다.

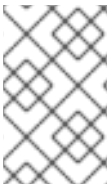
19.1.11.3.4.1. ISO 설치를 위한 네트워킹 및 본딩 옵션

ISO 이미지에서 RHCOS를 설치하는 경우, 해당 이미지를 부팅할 때 수동으로 커널 인수를 추가하여 노드의 네트워킹을 구성할 수 있습니다. 네트워킹 인수를 지정하지 않으면 RHCOS에서 Ignition 구성 파일을 가져오는 데 네트워킹이 필요함을 감지하면 initramfs에서 DHCP가 활성화됩니다.

중요

네트워킹 인수를 수동으로 추가할 때 initramfs에서 네트워크를 가져오려면 rd.neednet=1 커널 인수도 추가해야 합니다.

다음 표는 ISO 설치를 위해 RHCOS(Red Hat Enterprise Linux CoreOS) 노드의 네트워킹 및 본딩 구성 예를 보여줍니다. 예제에서는 ip=, nameserver=, bond= 커널 인수를 사용하는 방법을 설명합니다.

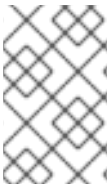


참고

커널 인수를 추가할 때 순서가 중요합니다: ip=, nameserver= 및 bond= 입니다.

이는 시스템 부팅 중에 dracut 툴로 전달되는 네트워킹 옵션입니다. dracut에서 지원하는 네트워킹 옵션에 대한 자세한 내용은 dracut.cmdline 메뉴얼 페이지를 참조하십시오.

다음 정보는 ISO 설치를 위해 RHCOS 노드에서 네트워킹을 구성하는 예제를 제공합니다. 예제에서는 ip= 및 nameserver= 커널 인수를 사용하는 방법을 설명합니다.



참고

커널 인수를 추가할 때 순서가 중요합니다(ip = 및 nameserver=).

이는 시스템 부팅 중에 dracut 툴로 전달되는 네트워킹 옵션입니다. dracut에서 지원하는 네트워킹 옵션에 대한 자세한 내용은 dracut.cmdline 메뉴얼 페이지를 참조하십시오.

다음 예제는 ISO 설치를 위한 네트워킹 옵션입니다.

DHCP 또는 고정 IP 주소 구성

IP 주소를 구성하려면 DHCP(ip=dhcp)를 사용하거나 개별 고정 IP 주소(ip=<host_ip>)를 설정합니다. 정적 IP를 설정하는 경우 각 노드에서 DNS 서버 IP 주소 (nameserver=<dns_ip>)를 확인합니다. 다음 예제에서는 다음을 설정합니다.

- 노드의 IP 주소는 10.10.10.2로 설정됩니다.
- 게이트웨이 주소는 10.10.10.254로 설정됩니다.
- 넷마스크는 255.255.255.0으로
- core0.example.com에 대한 호스트 이름
- 4.4.4.41의 DNS 서버 주소
- auto-configuration 값을 none으로 설정합니다. IP 네트워킹이 정적으로 구성되면 자동 구성이 필요하지 않습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
nameserver=4.4.4.41
```

참고

DHCP를 사용하여 RHCOS 시스템의 IP 주소 지정을 구성하는 경우 시스템은 DHCP를 통해 DNS 서버 정보도 가져옵니다. DHCP 기반 배포의 경우 DHCP 서버 구성을 통해 RHCOS 노드에서 사용할 DNS 서버 주소를 정의할 수 있습니다.

고정 호스트 이름 없이 IP 주소 구성

정적 호스트 이름을 할당하지 않고 IP 주소를 구성할 수 있습니다. 사용자가 정적 호스트 이름을 설정하지 않으면 역방향 DNS 조회에 의해 자동으로 선택됩니다. 정적 호스트 이름이 없는 IP 주소를 구성하려면 다음 예제를 참조합니다.

- 노드의 IP 주소는 10.10.10.2로 설정됩니다.
- 게이트웨이 주소는 10.10.10.254로 설정됩니다.
- 넷마스크는 255.255.255.0으로
- 4.4.4.41의 DNS 서버 주소
- auto-configuration 값을 none으로 설정합니다. IP 네트워킹이 정적으로 구성되면 자동 구성이 필요하지 않습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0::enp1s0:none
nameserver=4.4.4.41
```

여러 네트워크 인터페이스 지정

여러 ip= 항목을 설정하여 여러 네트워크 인터페이스를 지정할 수 있습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

기본 게이트웨이 및 경로 구성

선택 사항: rd.route= 값을 설정하여 추가 네트워크에 대한 경로를 구성할 수 있습니다.

참고

하나 이상의 네트워크를 구성할 때 하나의 기본 게이트웨이가 필요합니다. 추가 네트워크 게이트웨이가 기본 네트워크 게이트웨이와 다른 경우 기본 게이트웨이가 기본 네트워크 게이트웨이어야 합니다.

- 다음 명령을 실행하여 기본 게이트웨이를 구성합니다.

```
ip=::10.10.10.254::::
```

- 다음 명령을 입력하여 추가 네트워크의 경로를 구성합니다.

```
rd.route=20.20.20.0/24:20.20.20.254:enp2s0
```

단일 인터페이스에서 DHCP 비활성화

두 개 이상의 네트워크 인터페이스가 있고 하나의 인터페이스만 사용되는 경우와 같이 단일 인터페이스에서 DHCP를 비활성화할 수 있습니다. 이 예에서 enp1s0 인터페이스에는 정적 네트워킹 구성이 있으며 enp2s0 용으로 DHCP가 사용되지 않습니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=::::core0.example.com:enp2s0:none
```

DHCP 및 고정 IP 구성 결합

다음과 같이 시스템의 DHCP 및 고정 IP 구성을 여러 네트워크 인터페이스와 결합할 수 있습니다.

```
ip=enp1s0:dhcp
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

개별 인터페이스에서 VLAN 구성

선택 사항: vlan= 매개변수를 사용하여 개별 인터페이스에서 VLAN을 구성할 수 있습니다.

- 네트워크 인터페이스에서 VLAN을 구성하고 고정 IP 주소를 사용하려면 다음 명령을 실행합니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0.100:none
vlan=enp2s0.100:enp2s0
```

- 네트워크 인터페이스에서 VLAN을 구성하고 DHCP를 사용하려면 다음 명령을 실행합니다.

```
ip=enp2s0.100:dhcp
vlan=enp2s0.100:enp2s0
```

여러 DNS 서버 제공

각 서버에 nameserver= 항목을 추가하여 여러 DNS 서버를 제공할 수 있습니다. 예를 들면 다음과 같습니다.

```
nameserver=1.1.1.1
nameserver=8.8.8.8
```

단일 인터페이스에 여러 네트워크 인터페이스 본딩

선택 사항: bond= 옵션을 사용하여 여러 네트워크 인터페이스를 단일 인터페이스에 결합할 수 있습니다. 다음 예제를 참조하십시오.

- 결합된 인터페이스를 구성하는 구문: bond = name [: network_interfaces] [: options]

  *name*은 결합하는 기기 이름(bond0)이고 *network_interfaces*는 쉼표로 구분되는 물리적(이더넷) 인터페이스 목록(em1, em2)이며, *options*은 쉼표로 구분되는 결합 옵션 목록입니다. 사용 가능한 옵션을 보려면 modinfo bonding을 입력하십시오.

- bond=를 사용하여 결합된 인터페이스를 생성할 때 IP 주소가 할당되는 방법과 결합된 인터페이스에 대한 기타 정보를 지정해야 합니다.

- DHCP를 사용하도록 결합된 인터페이스를 구성하려면 bond의 IP 주소를 dhcp로 설정하십시오. 예를 들면 다음과 같습니다.

  ```
  bond=bond0:em1,em2:mode=active-backup
  ip=bond0:dhcp
  ```

- 고정 IP 주소를 사용하도록 결합된 인터페이스를 구성하려면 원하는 특정 IP 주소 및 관련 정보를 입력합니다. 예를 들면 다음과 같습니다.

  ```
  bond=bond0:em1,em2:mode=active-backup
  ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
  ```

단일 인터페이스에 여러 네트워크 인터페이스 본딩

선택 사항: vlan= 매개변수를 사용하고 DHCP를 사용하도록 결합된 인터페이스에서 VLAN을 구성할 수 있습니다. 예를 들면 다음과 같습니다.

```
ip=bond0.100:dhcp
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

다음 예제를 사용하여 VLAN을 사용하여 결합된 인터페이스를 구성하고 고정 IP 주소를 사용합니다.

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0.100:none
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

네트워크 팀 구성 사용

선택 사항: team= 매개변수를 사용하여 네트워크 팀을 결합의 대안으로 사용할 수 있습니다.

- 팀 인터페이스를 구성하는 구문은 team=name[:network_interfaces]입니다.

  name 은 팀 장치 이름(team0)이고 *network_interfaces* 는 쉼표로 구분된 실제(이더넷) 인터페이스 목록(em1, em2)을 나타냅니다.

RHCOS가 향후 RHEL 버전으로 전환되면 티밍이 더 이상 사용되지 않을 예정입니다. 자세한 내용은 이 Red Hat 지식베이스 문서를 참조하십시오.

다음 예제를 사용하여 네트워크 팀을 구성합니다.

```
team=team0:em1,em2
ip=team0:dhcp
```

19.1.11.3.4.2. ISO 설치를 위한 coreos-installer 옵션

ISO 이미지에서 RHCOS 라이브 환경으로 부팅한 후 명령 프롬프트에서 coreos-installer install <options> <device>를 실행하여 RHCOS를 설치할 수 있습니다.

다음 표는 coreos-installer 명령으로 전달할 수 있는 하위 명령, 옵션 및 인수를 보여줍니다.

표 19.9. coreos-installer 하위 명령, 명령줄 옵션 및 인수

coreos-installer 설치 하위 명령	
하위 명령	*설명*

$ coreos-installer install <options> <device>	ISO 이미지에 Ignition 구성를 삽입합니다.
coreos-installer 설치 하위 명령 옵션	
옵션	*설명*
-u, --image-url <url>	이미지 URL을 수동으로 지정합니다.
-f, --image-file <path>	로컬 이미지 파일을 수동으로 지정합니다. 디버깅에 사용됩니다.
-i, --ignition-file <path>	파일의 Ignition 구성을 삽입합니다.
-I, --ignition-url <URL>	URL의 Ignition 구성을 삽입합니다.
--ignition-hash <digest>	Ignition 구성의 **type-value**를 요약합니다.
-p, --platform <name>	설치된 시스템의 Ignition 플랫폼 ID를 재정의합니다.
--append-karg <arg>…	설치된 시스템에 기본 커널 인수를 추가합니다.
--delete-karg <arg>…	설치된 시스템에서 기본 커널 인수를 삭제합니다.
-n, --copy-network	설치 환경의 네트워크 구성을 복사합니다. **중요** **copy-network** 옵션은 **/etc/NetworkManager/system-connections**에 있는 네트워킹 구성만 복사합니다. 특히 시스템 호스트 이름을 복사하지 않습니다.
--network-dir <path>	**-n**과 함께 사용됩니다. 기본값은 **/etc/NetworkManager/system-connections/**입니다.
--save-partlabel <lx>..	이 레이블 glob로 파티션을 저장합니다.
--save-partindex <id>…	이 번호 또는 범위로 파티션을 저장합니다.
--insecure	서명 확인을 건너뜁니다.
--insecure-ignition	HTTPS 또는 해시 없는 Ignition URL을 허용합니다.
--architecture <name>	대상 CPU 아키텍처입니다. 기본값은 **x86_64**입니다.

--preserve-on-error	오류 발생한 파티션 테이블을 지우지 않습니다.
-h, --help	도움말 정보를 출력합니다.
coreos-install 설치 하위 명령 인수	
인수	*설명*
<device>	대상 장치입니다.
coreos-installer ISO Ignition 하위 명령	
하위 명령	*설명*
$ coreos-installer iso ignition embed <options> --ignition-file <file_path> <ISO_image>	ISO 이미지에 Ignition 구성를 삽입합니다.
coreos-installer iso ignition show <options> <ISO_image>	ISO 이미지에 삽입된 Ignition 구성를 표시합니다.
coreos-installer iso ignition remove <options> <ISO_image>	ISO 이미지에서 삽입된 Ignition 구성를 제거합니다.
coreos-installer ISO Ignition 하위 명령 옵션	
옵션	*설명*
-f, --force	기존 Ignition 구성를 덮어씁니다.
-i, --ignition-file <path>	사용할 Ignition 구성입니다. 기본값은 **stdin**입니다.
-o, --output <path>	새 출력 파일에 ISO를 씁니다.
-h, --help	도움말 정보를 출력합니다.
coreos-installer PXE Ignition 하위 명령	
하위 명령	*설명*
이러한 모든 옵션이 모든 하위 명령에서 허용되지는 않습니다.	
coreos-installer pxe ignition wrap <options>	Ignition 구성을 이미지로 래핑합니다.
coreos-installer pxe ignition unwrap <options> <image_name>	이미지에 래핑된 Ignition 구성를 표시합니다.

coreos-installer PXE Ignition 하위 명령 옵션	
옵션	*설명*
이러한 모든 옵션이 모든 하위 명령에서 허용되지는 않습니다.	
-i, --ignition-file <path>	사용할 Ignition 구성입니다. 기본값은 **stdin**입니다.
-o, --output <path>	새 출력 파일에 ISO를 씁니다.
-h, --help	도움말 정보를 출력합니다.

19.1.11.3.4.3. ISO 또는 PXE 설치를 위한 coreos.inst 부팅 옵션

coreos.inst 부팅 인수를 RHCOS 라이브 설치 프로그램에 전달하여 부팅 시 coreos-installer 옵션을 자동으로 호출할 수 있습니다. 이러한 매개 변수는 표준 부팅 인수 외에 제공됩니다.

- **ISO 설치의 경우 부트 로더 메뉴에서 자동 부팅을 중단하여 coreos.inst 옵션을 추가할 수 있습니다. RHEL CoreOS (Live) 메뉴 옵션이 강조 표시된 상태에서 TAB을 눌러 자동 부팅을 중단할 수 있습니다.**

- **PXE 또는 iPXE 설치의 경우 RHCOS 라이브 설치 프로그램을 부팅하기 전에 coreos.inst 옵션을 APPEND 줄에 추가해야 합니다.**

다음 표는 ISO 및 PXE 설치를 위한 RHCOS 라이브 설치 관리자 coreos.inst 부팅 옵션을 보여줍니다.

표 19.10. coreos.inst 부팅 옵션

인수	설명
coreos.inst.install_dev	필수 항목입니다. 설치할 시스템의 블록 장치입니다. **sda**가 허용되더라도 전체 경로 (예:**/dev/sda**)를 사용하는 것이 좋습니다.
coreos.inst.ignition_url	선택사항: 설치된 시스템에 삽입할 Ignition 구성의 URL입니다. URL을 지정하지 않으면 Ignition 구성이 포함되지 않습니다. HTTP 및 HTTPS 프로토콜만 지원됩니다.
coreos.inst.save_partlabel	선택사항: 설치 중에 보존 할 파티션의 쉼표로 구분된 레이블입니다. Glob 스타일 와일드카드가 허용됩니다. 지정된 파티션이 존재할 필요는 없습니다.

인수	설명
coreos.inst.save_partindex	선택사항: 설치 도중 보존할 파티션 인덱스들입니다(쉼표로 구분됨). **m-n** 범위가 허용되며 **m** 또는 **n**은 생략할 수 있습니다. 지정된 파티션이 존재할 필요는 없습니다.
coreos.inst.insecure	선택사항: **coreos.inst.image_url**로 지정된 OS 이미지의 서명되지 않은 상태를 허용합니다.
coreos.inst.image_url	선택사항: 지정된 RHCOS 이미지를 다운로드하여 설치합니다. ● 이 인수는 프로덕션 환경에서 사용할 수 없으며, 디버깅 용도로만 사용됩니다. ● 이 인수를 사용하면 라이브 미디어와 일치하지 않는 RHCOS 버전을 설치할 수 있지만, 설치하려는 버전과 일치하는 미디어를 사용하는 것이 좋습니다. ● **coreos.inst.image_url**을 사용하는 경우 **coreos.inst.insecure**도 사용해야 합니다. 베어메탈 미디어가 OpenShift Container Platform용으로 GPG 서명되지 않았기 때문입니다. ● HTTP 및 HTTPS 프로토콜만 지원됩니다.
coreos.inst.skip_reboot	선택사항: 설치 후 시스템을 재부팅하지 않습니다. 설치가 완료되면 설치 과정에서 발생되는 상황을 검사할 수 있는 프롬프트가 표시됩니다. 이 인수는 프로덕션 환경에서 사용할 수 없으며, 디버깅 용도로만 사용됩니다.
coreos.inst.platform_id	선택사항: RHCOS 이미지가 설치되고 있는 플랫폼의 Ignition 플랫폼 ID입니다. 기본값은 **metal**입니다. 이 옵션에 따라 VMware와 같은 클라우드 공급자의 Ignition 구성를 요청할지 여부가 결정됩니다. 예: **coreos.inst.platform_id=vmware**.
ignition.config.url	선택사항: 라이브 부팅을 위한 Ignition 구성의 URL입니다. 예를 들어 **coreos-installer**가 호출되는 방식을 사용자 지정하거나 설치 전과 후에 코드를 실행하는 데 사용할 수 있습니다. 이 URL은 설치된 시스템의 Ignition 구성인 **coreos.inst.ignition_url**과 다릅니다.

19.1.11.4. bootupd를 사용하여 부트로더 업데이트

bootupd를 사용하여 부트로더를 업데이트하려면 RHCOS 머신에 수동으로 bootupd를 설치하거나 systemd 유닛이 활성화된 머신 구성을 제공해야 합니다. grubby 또는 기타 부트로더 툴과는 달리 bootupd는 커널 인수 전달과 같은 커널 공간 구성을 관리하지 않습니다.

bootupd를 설치한 후 **OpenShift Container Platform** 클러스터에서 원격으로 관리할 수 있습니다.

참고

bootupd는 **BootHole** 취약점에 대한 보호와 같이 베어 메탈 또는 가상화 하이퍼바이저 설치에서만 사용하는 것이 좋습니다.

수동 설치 방법

bootctl 명령줄 툴을 사용하여 **bootupd**를 수동으로 설치할 수 있습니다.

1. 시스템 상태를 검사합니다.

```
# bootupctl status
```

출력 예

```
Component EFI
  Installed: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
  Update: At latest version
```

2. **bootupd**를 설치하지 않고 생성된 **RHCOS** 이미지에는 명시적 채택 단계가 필요합니다.

   시스템 상태가 **Adoptable**인 경우 채택을 수행합니다.

```
# bootupctl adopt-and-update
```

출력 예

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

3. 업데이트를 사용할 수 있는 경우 다음 재부팅에 변경 사항이 적용되도록 업데이트를 적용합니다.

```
# bootupctl update
```

출력 예

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

시스템 구성 방법

bootupd를 활성화하는 또 다른 방법은 머신 구성을 제공하는 것입니다.

- 다음 예와 같이 systemd 단위가 활성화된 머신 구성 파일을 제공합니다.

출력 예

```
variant: rhcos
version: 1.1.0
systemd:
  units:
    - name: custom-bootupd-auto.service
      enabled: true
      contents: |
        [Unit]
        Description=Bootupd automatic update

        [Service]
        ExecStart=/usr/bin/bootupctl update
        RemainAfterExit=yes

        [Install]
        WantedBy=multi-user.target
```

19.1.12. 부트스트랩 프로세스가 완료될 때까지 대기 중

OpenShift Container Platform 부트스트랩 프로세스는 클러스터 노드가 먼저 디스크에 설치된 영구 RHCOS 환경으로 부팅된 후에 시작됩니다. Ignition 구성 파일을 통해 제공되는 구성 정보는 부트스트랩 프로세스를 초기화하고 머신에 OpenShift Container Platform을 설치하는 데 사용됩니다. 부트스트랩 프로세스가 완료될 때까지 기다려야 합니다.

사전 요구 사항

- 클러스터에 대한 Ignition 구성 파일이 생성되어 있습니다.
- 적합한 네트워크, DNS 및 로드 밸런싱 인프라가 구성되어 있습니다.
- 설치 프로그램을 받아서 클러스터의 Ignition 구성 파일을 생성했습니다.
- 클러스터 머신에 RHCOS를 설치하고 OpenShift Container Platform 설치 프로그램에서 생성된 Ignition 구성 파일을 제공했습니다.
- 사용자 시스템에서 직접 인터넷에 액세스하거나 HTTP 또는 HTTPS 프록시를 사용할 수 있습니다.

절차

1. 부트스트랩 프로세스를 모니터링합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
  --log-level=info 2
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2 다른 설치 세부 사항을 보려면 info 대신 warn, debug 또는 error를 지정합니다.

출력 예

```
INFO Waiting up to 30m0s for the Kubernetes API at
https://api.test.example.com:6443...
INFO API v1.22.1 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

이 명령은 **Kubernetes API** 서버가 컨트롤 플레인 시스템에서 부트스트랩되었다는 신호를 보낼 때 성공합니다.

2. 부트스트랩 프로세스가 완료된 후 로드 밸런서에서 부트스트랩 시스템을 제거합니다.

중요

이 시점에 로드 밸런서에서 부트스트랩 시스템을 제거해야 합니다. 부트스트랩 머신 자체를 제거하거나 다시 포맷할 수도 있습니다.

19.1.13. CLI를 사용하여 클러스터에 로그인

클러스터 **kubeconfig** 파일을 내보내서 기본 시스템 사용자로 클러스터에 로그인할 수 있습니다. **kubeconfig** 파일에는 **CLI**에서 올바른 클러스터 및 **API** 서버에 클라이언트를 연결하는 데 사용하는 클러스터에 대한 정보가 포함되어 있습니다. 이 파일은 클러스터별로 고유하며 **OpenShift Container Platform** 설치 과정에서 생성됩니다.

사전 요구 사항

- **OpenShift Container Platform** 클러스터를 배포했습니다.
- **oc CLI**를 설치했습니다.

프로세스

1. **kubeadmin** 인증 정보를 내보냅니다.

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 <installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

2. 내보낸 구성을 사용하여 oc 명령을 성공적으로 실행할 수 있는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:admin
```

19.1.14. 머신의 인증서 서명 요청 승인

클러스터에 시스템을 추가하면 추가한 시스템별로 보류 중인 인증서 서명 요청(CSR)이 두 개씩 생성됩니다. 이러한 CSR이 승인되었는지 확인해야 하며, 필요한 경우 이를 직접 승인해야 합니다. 클라이언트 요청을 먼저 승인한 다음 서버 요청을 승인해야 합니다.

사전 요구 사항

- 클러스터에 시스템을 추가했습니다.

프로세스

1. 클러스터가 시스템을 인식하는지 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0 Ready    master  63m  v1.22.1
```

```
master-1  Ready    master  63m  v1.22.1
master-2  Ready    master  64m  v1.22.1
```

출력에 생성된 모든 시스템이 나열됩니다.

참고

이전 출력에는 일부 CSR이 승인될 때까지 컴퓨팅 노드(작업자 노드라고도 함)가 포함되지 않을 수 있습니다.

2. 보류 중인 CSR을 검토하고 클러스터에 추가한 각 시스템에 대해 Pending 또는 Approved 상태의 클라이언트 및 서버 요청이 표시되는지 확인합니다.

```
$ oc get csr
```

출력 예

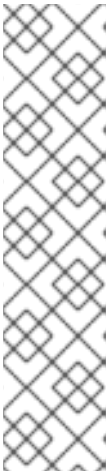
```
NAME        AGE    REQUESTOR                                                   CONDITION
csr-8b2br   15m    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

예에서는 두 시스템이 클러스터에 참여하고 있습니다. 목록에는 승인된 CSR이 더 많이 나타날 수도 있습니다.

3. CSR이 승인되지 않은 경우, 추가된 시스템에 대한 모든 보류 중인 CSR이 Pending 상태로 전환된 후 클러스터 시스템의 CSR을 승인합니다.

참고

CSR은 교체 주기가 자동으로 만료되므로 클러스터에 시스템을 추가한 후 1 시간 이내에 CSR을 승인하십시오. 한 시간 내에 승인하지 않으면 인증서가 교체되고 각 노드에 대해 두 개 이상의 인증서가 표시됩니다. 이러한 인증서를 모두 승인해야 합니다. 클라이언트 CSR이 승인되면 Kubelet은 인증서에 대한 보조 CSR을 생성하므로 수동 승인이 필요합니다. 그러면 Kubelet에서 동일한 매개변수를 사용하여 새 인증서를 요청하는 경우 인증서 갱신 요청은 machine-approver에 의해 자동으로 승인됩니다.



참고

베어 메탈 및 기타 사용자 프로비저닝 인프라와 같이 머신 API를 사용하도록 활성화되지 않는 플랫폼에서 실행되는 클러스터의 경우 CSR(Kubelet service Certificate Request)을 자동으로 승인하는 방법을 구현해야 합니다. 요청이 승인되지 않으면 API 서버가 kubelet에 연결될 때 서비스 인증서가 필요하므로 oc exec, oc rsh, oc logs 명령을 성공적으로 수행할 수 없습니다. Kubelet 엔드 포인트에 연결하는 모든 작업을 수행하려면 이 인증서 승인이 필요합니다. 이 방법은 새 CSR을 감시하고 CSR이 system:node 또는 system:admin 그룹의 node-bootstrapper 서비스 계정에 의해 제출되었는지 확인하고 노드의 ID를 확인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 <csr_name>은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

참고

일부 Operator는 일부 CSR이 승인될 때까지 사용할 수 없습니다.

4. 이제 클라이언트 요청이 승인되었으므로 클러스터에 추가한 각 머신의 서버 요청을 검토해야 합니다.

```
$ oc get csr
```

출력 예

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal        Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal        Pending
...
```

5. 나머지 CSR이 승인되지 않고 Pending 상태인 경우 클러스터 머신의 CSR을 승인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

1 <csr_name>은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs oc adm certificate approve
```

6. 모든 클라이언트 및 서버 CSR이 승인된 후 머신은 Ready 상태가 됩니다. 다음 명령을 실행하여 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0  Ready    master  73m  v1.22.1
master-1  Ready    master  73m  v1.22.1
master-2  Ready    master  74m  v1.22.1
worker-0  Ready    worker  11m  v1.22.1
worker-1  Ready    worker  11m  v1.22.1
```

참고

머신이 **Ready** 상태로 전환하는 데 서버 **CSR**의 승인 후 몇 분이 걸릴 수 있습니다.

추가 정보

- **CSR**에 대한 자세한 내용은 인증서 서명 요청을 참조하십시오.

19.1.15. Operator의 초기 설정

컨트롤 플레인이 초기화된 후 일부 **Operator**를 즉시 구성하여 모두 사용 가능하도록 해야 합니다.

사전 요구 사항

- 컨트롤 플레인이 초기화되어 있습니다.

프로세스

1. 클러스터 구성 요소가 온라인 상태인지 확인합니다.

```
$ watch -n5 oc get clusteroperators
```

출력 예

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.9.0     True        False         False     19m
baremetal                                  4.9.0     True        False         False     37m
cloud-credential                           4.9.0     True        False         False     40m
```

```
cluster-autoscaler                         4.9.0     True        False         False      37m
config-operator                            4.9.0     True        False         False      38m
console                                    4.9.0     True        False         False      26m
csi-snapshot-controller                    4.9.0     True        False         False      37m
dns                                        4.9.0     True        False         False      37m
etcd                                       4.9.0     True        False         False      36m
image-registry                             4.9.0     True        False         False      31m
ingress                                    4.9.0     True        False         False      30m
insights                                   4.9.0     True        False         False      31m
kube-apiserver                             4.9.0     True        False         False      26m
kube-controller-manager                    4.9.0     True        False         False      36m
kube-scheduler                             4.9.0     True        False         False      36m
kube-storage-version-migrator              4.9.0     True        False         False      37m
machine-api                                4.9.0     True        False         False      29m
machine-approver                           4.9.0     True        False         False      37m
machine-config                             4.9.0     True        False         False      36m
marketplace                                4.9.0     True        False         False      37m
monitoring                                 4.9.0     True        False         False      29m
network                                    4.9.0     True        False         False      38m
node-tuning                                4.9.0     True        False         False      37m
openshift-apiserver                        4.9.0     True        False         False      32m
openshift-controller-manager               4.9.0     True        False         False      30m
openshift-samples                          4.9.0     True        False         False      32m
operator-lifecycle-manager                 4.9.0     True        False         False      37m
operator-lifecycle-manager-catalog         4.9.0     True        False         False      37m
operator-lifecycle-manager-packageserver   4.9.0     True        False         False      32m
service-ca                                 4.9.0     True        False         False      38m
storage                                    4.9.0     True        False         False      37m
```

2. 사용할 수 없는 **Operator**를 구성합니다.

19.1.15.1. 기본 OperatorHub 소스 비활성화

Red Hat 및 커뮤니티 프로젝트에서 제공하는 콘텐츠를 소싱하는 **Operator** 카탈로그는 **OpenShift Container Platform**을 설치하는 동안 기본적으로 **OperatorHub**용으로 구성됩니다. 제한된 네트워크 환경에서는 클러스터 관리자로서 기본 카탈로그를 비활성화해야 합니다.

프로세스

- **OperatorHub** 오브젝트에 **disableAllDefaultSources: true**를 추가하여 기본 카탈로그의 소스를 비활성화합니다.

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

작은 정보

또는 웹 콘솔을 사용하여 카탈로그 소스를 관리할 수 있습니다. 관리 → 클러스터 설정 → 구성 → OperatorHub 페이지에서 개별 소스를 생성, 삭제, 비활성화 및 활성화할 수 있는 소스 탭을 클릭합니다.

19.1.15.2. 설치 중 제거된 이미지 레지스트리

공유 가능한 개체 스토리지를 제공하지 않는 플랫폼에서 OpenShift Image Registry Operator는 자체적으로 Removed로 부트스트랩합니다. 이를 통해 openshift-installer가 이러한 플랫폼 유형에서 설치를 완료할 수 있습니다.

설치 후 managementState를 Removed에서 Managed로 전환하도록 Image Registry Operator 구성을 편집해야 합니다.

참고

Prometheus 콘솔은 ImageRegistryRemoved 경고를 제공합니다. 다음은 예시 경고입니다.

"이미지 레지스트리가 제거되었습니다. ImageStreamTags를 참조하는 ImageStreamTags, BuildConfigs 및 DeploymentConfigs가 예상대로 작동하지 않을 수 있습니다. 스토리지를 구성하고 configs.imageregistry.operator.openshift.io를 편집하여 구성을 Managed 상태로 업데이트하십시오."

19.1.15.3. 이미지 레지스트리 스토리지 구성

기본 스토리지를 제공하지 않는 플랫폼에서는 처음에 Image Registry Operator를 사용할 수 없습니다. 설치한 후에 스토리지를 사용하도록 레지스트리를 구성하여 Registry Operator를 사용 가능하도록 만들어야 합니다.

프로덕션 클러스터에 필요한 영구 볼륨을 구성하는 과정의 지침이 표시됩니다. 해당하는 경우, 프로덕션 환경 외 클러스터에서만 사용할 수 있는 저장 위치로서 빈 디렉터리를 구성하는 과정의 지침이 표시됩니다.

업그레이드 중에 Recreate 롤아웃 전략을 사용하여 이미지 레지스트리의 블록 스토리지 유형 사용을 허용하기 위한 추가 지침이 제공됩니다.

19.1.15.3.1. 베어메탈 및 기타 수동 설치를 위한 레지스트리 스토리지 구성

클러스터 관리자는 설치한 후 스토리지를 사용하도록 레지스트리를 구성해야 합니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- 베어 메탈과 같이 수동으로 프로비저닝된 **RHCOS(Red Hat Enterprise Linux CoreOS)** 노드를 사용하는 클러스터가 있어야 합니다.
- **Red Hat OpenShift Container Storage**와 같이 클러스터용 영구 스토리지 프로비저닝이 있습니다.

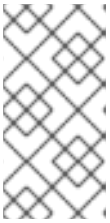
중요

OpenShift Container Platform은 복제본이 하나만 있는 경우 이미지 레지스트리 스토리지에 대한 **ReadWriteOnce** 액세스를 지원합니다. **ReadWriteOnce** 액세스에서는 레지스트리가 **Recreate** 롤아웃 전략을 사용해야 합니다. 두 개 이상의 복제본으로 고 가용성을 지원하는 이미지 레지스트리를 배포하려면 **ReadWriteMany** 액세스가 필요합니다.

- **"100Gi"** 용량이 필요합니다.

절차

1. 스토리지를 사용하도록 레지스트리를 구성하기 위해 **configs.imageregistry/cluster** 리소스에서 **spec.storage.pvc**를 변경합니다.



참고

공유 스토리지를 사용할 때 보안 설정을 확인하여 외부에서의 액세스를 방지합니다.

2. 레지스트리 **pod**가 없는지 확인합니다.

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

출력 예

```
No resourses found in openshift-image-registry namespace
```

참고

출력에 레지스트리 Pod가 있는 경우 이 절차를 계속할 필요가 없습니다.

3. 레지스트리 구성을 확인합니다.

```
$ oc edit configs.imageregistry.operator.openshift.io
```

출력 예

```
storage:
  pvc:
    claim:
```

image-registry-storage PVC의 자동 생성을 허용하도록 claim 필드를 비워 둡니다.

4. clusteroperator 상태를 확인합니다.

```
$ oc get clusteroperator image-registry
```

출력 예

```
NAME             VERSION       AVAILABLE   PROGRESSING   DEGRADED   SINCE
MESSAGE
image-registry   4.9           True        False         False      6h50m
```

5. 이미지를 빌드 및 푸시할 수 있도록 레지스트리의 관리가 설정되어 있는지 확인하십시오.

- 다음을 실행합니다.

```
$ oc edit configs.imageregistry/cluster
```

다음으로 라인을 변경하십시오.

```
managementState: Removed
```

다음으로 변경

```
managementState: Managed
```

19.1.15.3.2. 프로덕션 환경 외 클러스터에서 이미지 레지스트리의 스토리지 구성

이미지 레지스트리 **Operator**에 대한 스토리지를 구성해야 합니다. 프로덕션 환경 외 클러스터의 경우, 이미지 레지스트리를 빈 디렉터리로 설정할 수 있습니다. 이렇게 하는 경우 레지스트리를 다시 시작하면 모든 이미지가 손실됩니다.

절차

- 이미지 레지스트리 스토리지를 빈 디렉터리로 설정하려면 다음을 수행하십시오.

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch
'{"spec":{"storage":{"emptyDir":{}}}}'
```

주의

프로덕션 환경 외 클러스터에 대해서만 이 옵션을 구성하십시오.

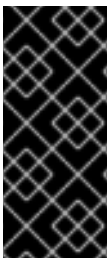
Image Registry Operator가 구성 요소를 초기화하기 전에 이 명령을 실행하면 oc patch 명령이 실패하며 다음 오류가 발생합니다.

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster"
not found
```

몇 분 후에 명령을 다시 실행하십시오.

19.1.15.3.3. 블록 레지스트리 스토리지 구성

클러스터 관리자로서 업그레이드 중에 이미지 레지스트리가 블록 스토리지 유형을 사용할 수 있도록 허용하기 위해 Recreate 롤아웃 전략을 사용할 수 있습니다.



중요

블록 스토리지 볼륨이 지원되지만 프로덕션 클러스터에서 이미지 레지스트리와 함께 사용하는 것은 권장되지 않습니다. 레지스트리가 블록 스토리지에 구성된 설치는 레지스트리가 둘 이상의 복제본을 가질 수 없기 때문에 가용성이 높지 않습니다.

절차

1. 이미지 레지스트리 스토리지를 블록 스토리지 유형으로 설정하려면 레지스트리가 Recreate 롤아웃 전략을 사용하고 하나의 (1) 복제본에서만 실행되도록 레지스트리를 패치합니다.

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. 블록 스토리지 장치에 PV를 프로비저닝하고 해당 볼륨의 PVC를 생성합니다. 요청된 블록 볼륨은 RWO(ReadWriteOnce) 액세스 모드를 사용합니다.

3. 올바른 PVC를 참조하도록 레지스트리 구성을 편집합니다.

19.1.16. 사용자 프로비저닝 인프라에 설치 완료

Operator 구성을 완료한 후 제공하는 인프라에 클러스터 설치를 완료할 수 있습니다.

사전 요구 사항

- 컨트롤 플레인이 초기화되어 있습니다.
- 초기 Operator 구성을 완료해야 합니다.

프로세스

1. 다음 명령을 사용하여 모든 클러스터 구성 요소가 온라인 상태인지 확인합니다.

```
$ watch -n5 oc get clusteroperators
```

출력 예

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.9.0     True        False         False      19m
baremetal                                  4.9.0     True        False         False      37m
cloud-credential                           4.9.0     True        False         False      40m
cluster-autoscaler                         4.9.0     True        False         False      37m
config-operator                            4.9.0     True        False         False      38m
console                                    4.9.0     True        False         False      26m
csi-snapshot-controller                    4.9.0     True        False         False      37m
dns                                        4.9.0     True        False         False      37m
etcd                                       4.9.0     True        False         False      36m
image-registry                             4.9.0     True        False         False      31m
ingress                                    4.9.0     True        False         False      30m
insights                                   4.9.0     True        False         False      31m
kube-apiserver                             4.9.0     True        False         False      26m
kube-controller-manager                    4.9.0     True        False         False      36m
kube-scheduler                             4.9.0     True        False         False      36m
kube-storage-version-migrator              4.9.0     True        False         False      37m
machine-api                                4.9.0     True        False         False      29m
machine-approver                           4.9.0     True        False         False      37m
machine-config                             4.9.0     True        False         False      36m
marketplace                                4.9.0     True        False         False      37m
monitoring                                 4.9.0     True        False         False      29m
network                                    4.9.0     True        False         False      38m
node-tuning                                4.9.0     True        False         False      37m
openshift-apiserver                        4.9.0     True        False         False      32m
openshift-controller-manager               4.9.0     True        False         False      30m
openshift-samples                          4.9.0     True        False         False      32m
operator-lifecycle-manager                 4.9.0     True        False         False      37m
operator-lifecycle-manager-catalog         4.9.0     True        False         False      37m
operator-lifecycle-manager-packageserver   4.9.0     True        False         False      32m
service-ca                                 4.9.0     True        False         False      38m
storage                                    4.9.0     True        False         False      37m
```

또는 다음 명령은 모든 클러스터를 사용할 수 있을 때 알립니다. 또한 인증 정보를 검색하고 표시합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1

<installation_directory>는 설치 파일을 저장한 디렉터리의 경로를 지정합니다.

출력 예

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator가 Kubernetes API 서버에서 OpenShift Container Platform 클러스터 배포를 완료하면 명령이 성공합니다.

중요

- 설치 프로그램에서 생성하는 Ignition 구성 파일에 24시간 후에 만료되는 인증서가 포함되어 있습니다. 이 인증서는 그 후에 갱신됩니다. 인증서를 갱신하기 전에 클러스터가 종료되고 24시간이 지난 후에 클러스터가 다시 시작되면 클러스터는 만료된 인증서를 자동으로 복구합니다. 예외적으로 kubelet 인증서를 복구하려면 대기 중인 node-bootstrapper 인증서 서명 요청(CSR)을 수동으로 승인해야 합니다. 자세한 내용은 *Recovering from expired control plane certificates* 문서를 참조하십시오.
- 클러스터를 설치한 후 24시간에서 22시간까지의 인증서가 교체되기 때문에 생성된 후 12시간 이내에 Ignition 구성 파일을 사용하는 것이 좋습니다. 12시간 이내에 Ignition 구성 파일을 사용하면 설치 중에 인증서 업데이트가 실행되는 경우 설치 실패를 방지할 수 있습니다.

2. Kubernetes API 서버가 Pod와 통신하고 있는지 확인합니다.

a.

모든 **Pod** 목록을 보려면 다음 명령을 사용하십시오.

```
$ oc get pods --all-namespaces
```

출력 예

```
NAMESPACE                      NAME                                        READY   STATUS
RESTARTS   AGE
openshift-apiserver-operator   openshift-apiserver-operator-85cb746d55-zqhs8
1/1     Running   1          9m
openshift-apiserver            apiserver-67b9g                             1/1     Running   0
3m
openshift-apiserver            apiserver-ljcmx                             1/1     Running   0
1m
openshift-apiserver            apiserver-z25h4                             1/1     Running   0
2m
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8
1/1     Running   0          5m
...
```

b.

다음 명령을 사용하여 이전 명령의 출력에 나열된 **Pod**의 로그를 표시합니다.

```
$ oc logs <pod_name> -n <namespace>
```

1

이전 명령의 출력에 표시된 대로 **Pod** 이름과 네임스페이스를 지정합니다.

Pod 로그가 표시되면 **Kubernetes API** 서버는 클러스터 시스템과 통신할 수 있습니다.

3.

FCP(Fibre Channel Protocol)를 사용하는 설치에는 다중 경로를 활성화하기 위해 추가 단계가 필요합니다. 설치 중에 멀티패스를 활성화하지 마십시오.

자세한 내용은 *설치 후 머신 구성 작업* 설명서에서 **"RHCOS**에서 커널 인수를 사용하여 멀티패스 활성화**"**를 참조하십시오.

19.1.17. OpenShift Container Platform의 Telemetry 액세스

OpenShift Container Platform 4.9에서는 클러스터 상태 및 업데이트 진행에 대한 메트릭을 제공하기 위해 기본적으로 실행되는 Telemetry 서비스에 인터넷 액세스가 필요합니다. 클러스터가 인터넷에 연결되어 있으면 Telemetry가 자동으로 실행되고 OpenShift Cluster Manager 에 클러스터가 자동으로 등록됩니다.

OpenShift Cluster Manager 인벤토리가 올바르거나 OpenShift Cluster Manager를 사용하여 자동으로 또는 OpenShift Cluster Manager를 사용하여 수동으로 유지 관리되는지 확인한 후 subscription watch를 사용하여 계정 또는 다중 클러스터 수준에서 OpenShift Container Platform 서브스크립션을 추적합니다.

추가 리소스

- Telemetry 서비스에 대한 자세한 내용은 원격 상태 모니터링 정보를 참조하십시오.

19.1.18. 다음 단계

- 클러스터를 사용자 지정합니다.
- 필요한 경우 원격 상태 보고 옵트아웃을 수행할 수 있습니다.
- 레지스트리를 설정하고 레지스트리 스토리지를 구성합니다.

20장. 설치 구성

20.1. 노드의 사용자 정의

OpenShift Container Platform 노드를 직접 변경하는 것은 권장되지 않지만 필요에 따라 낮은 수준의 보안, 중복, 네트워킹 또는 성능 기능을 구현해야 하는 경우가 있습니다. OpenShift Container Platform 노드를 직접 변경하려면 다음을 수행하십시오.

- Openshift-install 중 클러스터를 시작하기 위해 매니페스트 파일에 포함된 머신 구성을 생성합니다.
- Machine Config Operator를 통해 OpenShift Container Platform 노드에 전달되는 머신 구성을 생성합니다.
- 베어 메탈 노드를 설치할 때 coreos-installer에 전달되는 Ignition 구성을 생성합니다.

다음 섹션에서는 이러한 방식으로 노드에서 설정할 수 있는 기능에 대해 설명합니다.

20.1.1. Butane을 사용하여 머신 구성 생성

머신 구성은 사용자 및 파일 시스템 생성, 네트워크 설정, systemd 장치 설치 등에 대해 시스템에 지시하여 컨트롤 플레인 및 작업자 시스템을 구성하는 데 사용됩니다.

머신 구성 수정은 어려울 수 있기 때문에 Butane configs를 사용하여 머신 구성을 생성하여 노드 구성을 훨씬 쉽게 구성할 수 있습니다.

20.1.1.1. Butane 정보

butane은 OpenShift Container Platform이 머신 구성 작성에 편리한 단기 구문을 제공하고 머신 구성에 대한 추가 검증을 수행하는 데 사용하는 명령줄 유틸리티입니다. Butane이 허용하는 Butane 구성 파일의 형식은 OpenShift Butane config 사양에 정의되어 있습니다.

20.1.1.2. Butane 설치

Butane 툴(butane)을 설치하여 명령줄 인터페이스에서 OpenShift Container Platform 머신 구성을 생성할 수 있습니다. 해당 바이너리 파일을 다운로드하여 Linux, Windows 또는 macOS에 butane을 설

치할 수 있습니다.

작은 정보

Butane 릴리스는 이전 릴리스 및 Fedora CoreOS Config Transpiler(FCCT)와 이전 버전과 호환됩니다.

프로세스

1. https://mirror.openshift.com/pub/openshift-v4/clients/butane/ Butane 이미지 다운로드 페이지로 이동합니다.

2. butane 바이너리를 가져옵니다.

   a. Butane의 최신 버전의 경우 최신 butane 이미지를 현재 디렉터리에 저장합니다.

   ```
   $ curl https://mirror.openshift.com/pub/openshift-v4/clients/butane/latest/butane --output butane
   ```

   b. 선택 사항: butane을 설치할 특정 아키텍처 유형(예: arch64 또는 ppc64le)에 적절한 URL을 지정합니다. 예를 들면 다음과 같습니다.

   ```
   $ curl https://mirror.openshift.com/pub/openshift-v4/clients/butane/latest/butane-aarch64 --output butane
   ```

3. 다운로드한 바이너리 파일을 실행 가능하게 합니다.

   ```
   $ chmod +x butane
   ```

4. butane 바이너리 파일을 PATH의 디렉터리로 이동합니다.

   PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

   ```
   $ echo $PATH
   ```

검증 절차

- 이제 butane 명령을 실행하여 Butane 툴을 사용할 수 있습니다.

```
$ butane <butane_file>
```

20.1.1.3. Butane을 사용하여 MachineConfig 오브젝트 생성

설치 시 또는 Machine Config Operator를 통해 작업자 또는 컨트롤 플레인 노드를 구성할 수 있도록 Butane을 사용하여 MachineConfig 오브젝트를 생성할 수 있습니다.

사전 요구 사항

- butane 유틸리티가 설치되어 있습니다.

프로세스

1. Butane 구성 파일을 생성합니다. 다음 예제에서는 시스템 콘솔을 구성하여 커널 디버그 메시지를 표시하고 chrony 시간 서비스에 대한 사용자 지정 설정을 지정하는 99-worker-custom.bu라는 파일을 생성합니다.

```
variant: openshift
version: 4.9.0
metadata:
  name: 99-worker-custom
  labels:
    machineconfiguration.openshift.io/role: worker
openshift:
  kernel_arguments:
    - loglevel=7
storage:
  files:
    - path: /etc/chrony.conf
      mode: 0644
      overwrite: true
      contents:
        inline: |
          pool 0.rhel.pool.ntp.org iburst
          driftfile /var/lib/chrony/drift
          makestep 1.0 3
          rtcsync
          logdir /var/log/chrony
```

참고

99-worker-custom.bu 파일은 작업자 노드에 대한 머신 구성을 생성하도록 설정되어 있습니다. 컨트롤 플레인 노드에 배포하려면 역할을 **worker**에서 **master**로 변경하십시오. 이 두 가지 작업을 수행하려면 여러유형의 배포에 서로 다른 파일 이름을 사용하여 전체 프로세스를 반복할 수 있습니다.

2. 이전 단계에서 생성한 파일을 **Butane**으로 지정하여 **MachineConfig** 오브젝트를 생성합니다.

```
$ butane 99-worker-custom.bu -o ./99-worker-custom.yaml
```

머신 구성을 완료하기 위해 **MachineConfig** 오브젝트 **YAML** 파일이 생성됩니다.

3. 향후 **MachineConfig** 오브젝트를 업데이트해야 하는 경우 **Butane** 구성을 저장합니다.

4. 클러스터가 아직 작동하지 않은 경우 매니페스트 파일을 생성하고 **MachineConfig** 오브젝트 **YAML** 파일을 **openshift** 디렉토리에 추가합니다. 클러스터가 이미 실행중인 경우 다음과 같이 파일을 적용합니다.

```
$ oc create -f 99-worker-custom.yaml
```

추가 리소스

- 노드에 커널 모듈 추가
- 설치하는 동안 디스크 암호화 및 미러링

20.1.2. day-1 커널 매개 변수 추가

day-2 활동으로 커널 매개 변수를 수정하는 것이 바람직하지만 초기 클러스터 설치 중에 모든 마스터 또는 작업자 노드에 커널 매개 변수를 추가할 수 있습니다. 시스템이 처음으로 부팅되기 전에 적용되도록 클러스터 설치 중에 커널 매개 변수를 추가해야하는 이유는 다음과 같습니다.

- **SELinux**와 같은 기능을 비활성화하여 이 기능이 시스템에 영향을 미치지 않도록할 필요가 있는 경우

주의

RHCOS에서 SELinux 비활성화는 지원되지 않습니다.

- 시스템을 시작하기 전에 몇 가지 낮은 수준의 네트워크 설정을 수행해야하는 경우

마스터 노드 또는 작업자 노드에 커널 매개 변수를 추가하기 위해 MachineConfig 객체를 생성하고 해당 객체를 클러스터 설정 중에 Ignition에서 사용하는 매니페스트 파일 세트에 삽입할 수 있습니다.

부팅시 RHEL 8 커널에 전달할 수 있는 매개 변수 목록은 Kernel.org 커널 매개 변수를 참조하십시오. 초기 OpenShift Container Platform 설치를 완료하기 위해 필요한 경우 이 절차를 사용하여 커널 매개 변수를 추가하는 것이 좋습니다.

절차

1. 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터에 대한 Kubernetes 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory>
```

2. 커널 매개 변수를 작업자 또는 컨트롤 플레인 노드에 추가할지 여부를 결정합니다.

3. openshift 디렉터리에서 파일 (예: 99-openshift-machineconfig-master-kargs.yaml)을 작성하여 커널 설정을 추가할 MachineConfig 객체를 정의합니다. 다음 예제에서는 loglevel=7 커널 매개 변수를 컨트롤 플레인 노드에 추가합니다.

```
$ cat << EOF > 99-openshift-machineconfig-master-kargs.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: master
  name: 99-openshift-machineconfig-master-kargs
spec:
  kernelArguments:
    - loglevel=7
EOF
```

-

커널 매개 변수를 작업자 노드에 추가하는 경우 **master**를 **worker**로 변경할 수 있습니다. 마스터 및 작업자 노드 모두에 추가할 별도의 **YAML** 파일을 생성합니다.

이제 계속해서 클러스터를 만들 수 있습니다.

20.1.3. 노드에 커널 모듈 추가

대부분의 일반적인 하드웨어의 경우 컴퓨터가 시작되면 **Linux** 커널에 이러한 하드웨어를 사용하는 데 필요한 장치 드라이버 모듈이 포함됩니다. 그러나 일부 하드웨어의 경우 해당 모듈은 **Linux**에서 제공되지 않습니다. 따라서 각 호스트 컴퓨터에 대해 이러한 모듈을 제공하는 방법을 찾아야합니다. 이 단계에서는 **OpenShift Container Platform** 클러스터 노드에 대해 이를 수행하는 방법을 설명합니다.

이 프로세스에 따라 커널 모듈을 처음 배포할 때 현재 커널에서 모듈을 사용할 수 있게 됩니다. 새 커널이 설치되면 **kmods-via-containers** 소프트웨어가 다시 빌드되고 모듈이 배포되어 새 커널과 호환되는 버전의 모듈을 사용할 수 있습니다.

이 기능이 각 노드에서 모듈을 최신 상태로 유지하는 방법은 다음과 같습니다.

- 새 커널이 설치되었는지 감지하기 위해 부팅시 시작되는 각 노드에 **systemd** 서비스를 추가합니다.
- 새로운 커널이 감지되면 서비스는 모듈을 다시 빌드하여 커널에 설치합니다.

이 단계에 필요한 소프트웨어에 대한 자세한 내용은 **kmods-via-containers github** 사이트를 참조하십시오.

다음의 몇 가지 중요 사항에 유의하십시오.

- 이 단계는 기술 프리뷰입니다.
- 소프트웨어 툴과 샘플은 공식 **RPM** 형식으로 제공되지 않으며 현재 이 절차에 명시된 비공식 **github.com** 사이트에서만 구할 수 있습니다.

- 이 절차를 통해 추가할 수 있는 타사 커널 모듈은 **Red Hat**에서 지원하지 않습니다.

- 이 절차에서는 커널 모듈을 빌드하는 데 필요한 소프트웨어가 **RHEL 8** 컨테이너에 배포됩니다. 노드가 새 커널을 가져 오면 각 노드에서 모듈이 자동으로 다시 빌드됩니다. 따라서 각 노드는 모듈을 다시 빌드하는 데 필요한 커널 및 관련 패키지가 포함된 **yum** 저장소에 액세스해야 합니다. 해당 콘텐츠는 유효한 **RHEL** 서브스크립션을 통해 효과적으로 사용할 수 있습니다.

20.1.3.1. 커널 모듈 컨테이너 빌드 및 테스트

커널 모듈을 **OpenShift Container Platform** 클러스터에 배포하기 전에 별도의 **RHEL** 시스템에서 프로세스를 테스트할 수 있습니다. 커널 모듈의 소스 코드, **KVC** 프레임 워크 및 **kmod-via-containers** 소프트웨어를 수집합니다. 다음으로 모듈을 빌드하고 테스트합니다. **RHEL 8** 시스템에서 이를 수행하려면 다음 프로세스를 따르십시오.

프로세스

1. **RHEL 8** 시스템을 등록합니다.

```
# subscription-manager register
```

2. **RHEL 8** 시스템에 서브스크립션을 연결합니다.

```
# subscription-manager attach --auto
```

3. 소프트웨어 및 컨테이너를 빌드하는 데 필요한 소프트웨어를 설치합니다.

```
# yum install podman make git -y
```

4. **kmod-via-containers** 저장소를 복제합니다.

   a. 저장소의 폴더를 만듭니다.

```
$ mkdir kmods; cd kmods
```

   b. 저장소를 복제합니다.

```
$ git clone https://github.com/kmods-via-containers/kmods-via-containers
```

5. RHEL 8 빌드 호스트에 KVC 프레임 워크 인스턴스를 설치하여 모듈을 테스트합니다. kmods-via-container systemd 서비스가 추가되어 로드됩니다.

   a. kmod-via-containers 디렉터리로 변경합니다.

   ```
   $ cd kmods-via-containers/
   ```

   b. KVC 프레임워크 인스턴스를 설치합니다.

   ```
   $ sudo make install
   ```

   c. systemd 관리자 설정을 다시로드합니다.

   ```
   $ sudo systemctl daemon-reload
   ```

6. 커널 모듈의 소스 코드를 가져옵니다. 소스 코드는 제어할 수 없지만 다른 사람이 제공하는 타사 모듈을 빌드하는 데 사용될 수 있습니다. 다음과 같이 시스템에 복제할 수있는 kvc-simple-kmod 예제에 표시된 내용과 유사한 내용이 필요합니다.

   ```
   $ cd .. ; git clone https://github.com/kmods-via-containers/kvc-simple-kmod
   ```

7. 설정 파일 simple-kmod.conf을 편집하고 Dockerfile의 이름을 Dockerfile.rhel로 변경합니다.

   a. kvc-simple-kmod 디렉터리로 변경합니다.

   ```
   $ cd kvc-simple-kmod
   ```

   b. Dockerfile의 이름을 바꿉니다.

   ```
   $ cat simple-kmod.conf
   ```

   Dockerfile 예

```
KMOD_CONTAINER_BUILD_CONTEXT="https://github.com/kmods-via-containers/kvc-simple-kmod.git"
KMOD_CONTAINER_BUILD_FILE=Dockerfile.rhel
KMOD_SOFTWARE_VERSION=dd1a7d4
KMOD_NAMES="simple-kmod simple-procfs-kmod"
```

8. 커널 모듈의 **kmods-via-containers @.service** 인스턴스 (이 예제에서는 **simple-kmod**)를 만듭니다.

```
$ sudo make install
```

9. **kmods-via-containers @.service** 인스턴스를 활성화합니다.

```
$ sudo kmods-via-containers build simple-kmod $(uname -r)
```

10. **systemd** 서비스를 활성화하고 시작합니다.

```
$ sudo systemctl enable kmods-via-containers@simple-kmod.service --now
```

a. 서비스 상태를 확인합니다.

```
$ sudo systemctl status kmods-via-containers@simple-kmod.service
```

출력 예

```
● kmods-via-containers@simple-kmod.service - Kmods Via Containers - simple-kmod
  Loaded: loaded (/etc/systemd/system/kmods-via-containers@.service;
      enabled; vendor preset: disabled)
  Active: active (exited) since Sun 2020-01-12 23:49:49 EST; 5s ago...
```

11. 커널 모듈이 로드되었는지 확인하려면 **lsmod** 명령을 사용하여 모듈을 나열하십시오.

```
$ lsmod | grep simple_
```

출력 예

```
simple_procfs_kmod     16384  0
simple_kmod            16384  0
```

12. 선택사항. 다른 방법을 사용하여 **simple-kmod** 예제가 작동하는지 확인합니다.

- **dmesg**를 사용하여 커널 링 버퍼에 **"Hello world"**메시지를 찾으십시오.

```
$ dmesg | grep 'Hello world'
```

출력 예

```
[ 6420.761332] Hello world from simple_kmod.
```

- **/proc**에서 **simple-procfs-kmod**의 값을 확인합니다.

```
$ sudo cat /proc/simple-procfs-kmod
```

출력 예

```
simple-procfs-kmod number = 0
```

- **spkut** 명령을 실행하여 모듈에 대한 자세한 정보를 가져옵니다.

```
$ sudo spkut 44
```

출력 예

```
KVC: wrapper simple-kmod for 4.18.0-147.3.1.el8_1.x86_64
Running userspace wrapper using the kernel module container...
+ podman run -i --rm --privileged
  simple-kmod-dd1a7d4:4.18.0-147.3.1.el8_1.x86_64 spkut 44
simple-procfs-kmod number = 0
simple-procfs-kmod number = 44
```

시스템이 부팅될 때 이 서비스는 새 커널이 실행 중인지를 확인합니다. 새 커널이 있으면 서비스는 새 버전의 커널 모듈을 빌드한 다음 로드합니다. 모듈이 이미 구축 된 경우 이를 로드합니다.

20.1.3.2. OpenShift Container Platform에 커널 모듈 프로비저닝

OpenShift Container Platform 클러스터를 처음 부팅할 때 커널 모듈을 활성화할 필요가 있는지 여부에 따라 다음 두 가지 방법 중 하나로 커널 모듈을 배포하도록 설정할 수 있습니다.

- 클러스터 설치시 커널 모듈 프로비저닝 **(day-1): MachineConfig** 개체를 통해 콘텐츠를 작성하고 매니페스트 파일 세트와 함께 **openshift-install**에 제공할 수 있습니다.

- **Machine Config Operator**를 통해 커널 모듈 프로비저닝 **(day-2):** 커널 모듈을 추가하기 위해 클러스터가 가동 될 때까지 대기할 경우 **MCO (Machine Config Operator)**를 통해 커널 모듈 소프트웨어를 배포할 수 있습니다.

두 경우 모두 새 커널이 감지되면 각 노드에서 커널 소프트웨어 패키지 및 관련 소프트웨어 패키지를 가져올 수 있어야 합니다. 해당 콘텐츠를 가져올 수 있도록 각 노드를 설정할 수있는 몇 가지 방법이 있습니다.

- 각 노드에 **RHEL** 인타이틀먼트를 제공합니다.

- **/ etc / pki / entitlement** 디렉터리에서 기존 **RHEL** 호스트의 **RHEL** 인타이틀먼트를 취득하고 **Ignition** 설정을 빌드할 때 제공하는 다른 파일과 동일한 위치에 복사합니다.

- **Dockerfile**에서 커널 및 기타 패키지가 포함된 **yum** 저장소에 대한 포인터를 추가합니다. 여기에는 새로 설치된 커널과 일치해야하므로 새 커널 패키지가 포함되어 있어야합니다.

20.1.3.2.1. MachineConfig 개체를 통한 커널 모듈 프로비저닝

MachineConfig 개체로 커널 모듈 소프트웨어를 패키지하면 설치시 또는 **Machine Config Operator**를 통해 해당 소프트웨어를 작업자 또는 컨트롤 플레인 노드에 전달할 수 있습니다.

프로세스

1. **RHEL 8** 시스템을 등록합니다.

```
# subscription-manager register
```

2. **RHEL 8** 시스템에 서브스크립션을 연결합니다.

```
# subscription-manager attach --auto
```

3. 소프트웨어를 빌드하는 데 필요한 소프트웨어를 설치합니다.

```
# yum install podman make git -y
```

4. 커널 모듈 및 툴을 호스팅할 디렉터리를 생성합니다.

```
$ mkdir kmods; cd kmods
```

5. **kmods-via-containers** 소프트웨어를 가져옵니다:

   a. **kmods-via-containers** 저장소를 복제합니다.

   ```
   $ git clone https://github.com/kmods-via-containers/kmods-via-containers
   ```

   b. **kvc-simple-kmod** 저장소를 복제합니다.

   ```
   $ git clone https://github.com/kmods-via-containers/kvc-simple-kmod
   ```

6. 모듈 소프트웨어를 가져옵니다. 이 예에서는 **kvc-simple-kmod**가 사용됩니다.

7. 이전에 복제된 리포지토리를 사용하여 **fakeroot** 디렉터리를 만들고 **Ignition**을 통해 전달할 파일을 이 디렉터리에 배치합니다.

   a. 디렉터리를 만듭니다.

   ```
   $ FAKEROOT=$(mktemp -d)
   ```

   b. **kmod-via-containers** 디렉터리로 변경합니다.

   ```
   $ cd kmods-via-containers
   ```

   c. **KVC** 프레임워크 인스턴스를 설치합니다.

   ```
   $ make install DESTDIR=${FAKEROOT}/usr/local CONFDIR=${FAKEROOT}/etc/
   ```

   d. **kvc-simple-kmod** 디렉터리로 변경합니다.

   ```
   $ cd ../kvc-simple-kmod
   ```

   e. 인스턴스를 생성합니다.

   ```
   $ make install DESTDIR=${FAKEROOT}/usr/local CONFDIR=${FAKEROOT}/etc/
   ```

8. 다음 명령을 실행하여 **fakeroot** 디렉토리를 복제하고 모든 심볼릭 링크를 대상 복사본으로 교체합니다.

   ```
   $ cd .. && rm -rf kmod-tree && cp -Lpr ${FAKEROOT} kmod-tree
   ```

9. 커널 모듈 트리를 포함하는 **Butane** 구성 파일 **99-simple-kmod.bu**를 생성하고 **systemd** 서비스를 활성화합니다.

참고

Butane에 대한 자세한 내용은 “Butane 을 사용하여 머신 구성 생성”을 참조하십시오.

```
variant: openshift
version: 4.9.0
metadata:
  name: 99-simple-kmod
  labels:
    machineconfiguration.openshift.io/role: worker 1
storage:
  trees:
    - local: kmod-tree
systemd:
  units:
    - name: kmods-via-containers@simple-kmod.service
      enabled: true
```

1

컨트롤 플레인 노드에 배포하려면 **worker**를 **master**로 변경합니다. 컨트롤 플레인 및 작업자 노드에 모두 배포하려면 각 노드 유형에 대해 이러한 지침의 나머지 부분을 한 번씩 수행합니다.

10. Butane을 사용하여 전달할 파일과 구성이 포함된 머신 구성 YAML 파일 **99-simple-kmod.yaml**을 생성합니다.

```
$ butane 99-simple-kmod.bu --files-dir . -o 99-simple-kmod.yaml
```

11. 클러스터가 아직 작동하지 않은 경우 매니페스트 파일을 생성하고 해당 파일을 **openshift** 디렉터리에 추가합니다. 클러스터가 이미 실행중인 경우 다음과 같이 파일을 적용합니다.

```
$ oc create -f 99-simple-kmod.yaml
```

노드는 **kmods-via-containers@simple-kmod.service** 서비스를 시작하고 커널 모듈이 로드됩니다.

12. 커널 모듈이 로드되었는지 확인하려면 **oc debug node / <openshift-node>**를 사용 후 **chroot / host**를 사용하여 노드에 로그인할 수 있습니다. 모듈을 나열하려면 **lsmod** 명령을 사용합니다.

```
$ lsmod | grep simple_
```

출력 예

```
simple_procfs_kmod     16384  0
simple_kmod            16384  0
```

20.1.4. 설치하는 동안 디스크 암호화 및 미러링

OpenShift Container Platform을 설치하는 동안 클러스터 노드에서 부팅 디스크 암호화 및 미러링을 활성화할 수 있습니다.

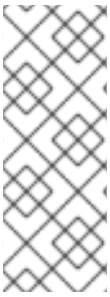
20.1.4.1. 디스크 암호화 정보

설치 시 컨트롤 플레인 및 컴퓨팅 노드에서 부팅 디스크에 대한 암호화를 활성화할 수 있습니다. OpenShift Container Platform은 Trusted Platform Module (TPM) v2 및 Tang 암호화 모드를 지원합니다.

- TPM v2: 기본 모드입니다. TPM v2는 서버에 포함된 보안 암호화 프로세서에 암호를 저장합니다. 이 모드를 사용하여 디스크가 서버에서 제거되면 클러스터 노드의 부팅 디스크 데이터가 암호 해독되지 않도록 할 수 있습니다.
- Tang: Tang 및 Clevis는 네트워크 바인딩 디스크 암호화(NBDE)를 활성화하는 서버 및 클라이언트 구성 요소입니다. 클러스터 노드의 부팅 디스크 데이터를 하나 이상의 Tang 서버에 바인딩할 수 있습니다. 이는 노드가 Tang 서버에 액세스할 수 있는 보안 네트워크에 있지 않는 한 데이터의 암호가 해독되지 않습니다. Clevis는 클라이언트 측에서 암호 해독을 구현하는 데 사용되는 자동화된 암호 해독 프레임워크입니다.

중요

Tang 암호화 모드를 사용하여 디스크를 암호화하는 것은 사용자 프로비저닝 인프라의 베어 메탈 및 vSphere 설치에만 지원됩니다.



참고

이전 버전의 **RHCOS(Red Hat Enterprise Linux CoreOS)**에서 디스크 암호화는 **Ignition** 구성에 **/etc/clevis.json**을 지정하여 구성되었습니다. 해당 파일은 **OpenShift Container Platform 4.7** 이상으로 생성된 클러스터에서 지원되지 않으며 다음 절차를 사용하여 디스크 암호화를 구성해야 합니다.

TPM v2 또는 **Tang** 암호화 모드가 활성화되면 **RHCOS** 부팅 디스크는 **LUKS2** 형식을 사용하여 암호화됩니다.

이는 다음과 같은 기능을 제공합니다:

- 설치 프로그램 프로비저닝 인프라 및 사용자 프로비저닝 인프라 배포에서 사용 가능
- **RHCOS (Red Hat Enterprise Linux CoreOS)** 시스템에서만 지원됨
- 매니페스트 설치 단계에서 디스크 암호화를 설정하여 첫 번째 부팅부터 디스크에 기록된 모든 데이터를 암호화함
- 사용자 개입없이 암호 제공 가능
- **FIPS** 모드가 활성화된 경우 **AES-256-XTS** 암호화 또는 **AES-256-CBC** 사용

20.1.4.1.1. 암호화 임계값 구성

OpenShift Container Platform에서는 둘 이상의 **Tang** 서버에 대한 요구 사항을 지정할 수 있습니다. 또한 **TPM** 보안 암호화 프로세서가 있고 **Tang** 서버에 보안 네트워크를 통해 액세스할 수 있는 경우에만 부팅 디스크 데이터를 해독할 수 있도록 **TPM v2** 및 **Tang** 암호화 모드를 동시에 구성할 수 있습니다.

Butane 구성에서 **threshold** 속성을 사용하여 암호 해독을 수행하려면 충족해야 하는 최소 **TPM v2** 및 **Tang** 암호화 조건을 정의할 수 있습니다. 선언된 조건의 조합을 통해 명시된 값에 도달하면 임계값이 충족됩니다. 예를 들어 다음 구성의 **threshold 2**는 두 개의 **Tang** 서버에 액세스하거나 **TPM** 보안 암호화 프로세서 및 **Tang** 서버 중 하나에 액세스하여 연결할 수 있습니다.

디스크 암호화를 위한 **Butane** 구성의 예

```
variant: openshift
version: 4.9.0
metadata:
  name: worker-storage
  labels:
    machineconfiguration.openshift.io/role: worker
boot_device:
  layout: x86_64
  luks:
    tpm2: true (1)
    tang: (2)
      - url: http://tang1.example.com:7500
        thumbprint: jwGN5tRFK-kF6pIX89ssF3khxxX
      - url: http://tang2.example.com:7500
        thumbprint: VCJsvZFjBSIHSldw78rOrq7h2ZF
    threshold: 2 (3)
openshift:
  fips: true
```

(1)

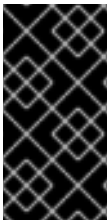
Trusted Platform Module (TPM)을 사용하여 **root** 파일 시스템을 암호화하려면 이 필드를 포함합니다.

(2)

하나 이상의 **Tang** 서버를 사용하려면 이 섹션을 포함합니다.

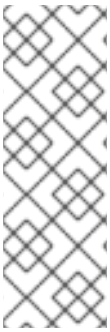
(3)

암호 해독을 수행하려면 충족해야 하는 최소 **TPM v2** 및 **Tang** 암호화 조건 수를 지정합니다.



중요

기본 **threshold**는 1입니다. 구성에 여러 암호화 조건을 포함하지만 임계값을 지정하지 않을 경우 조건이 하나라도 충족되면 암호 해독이 발생할 수 있습니다.



참고

암호 해독에 **TPM v2** 및 **Tang**이 모두 필요한 경우 **threshold** 속성의 값은 명시된 **Tang** 서버의 총 수에 **1**을 더한 것과 같아야 합니다. **threshold**가 낮으면 암호화 모드 중 하나를 사용하여 임계값에 도달할 수 있습니다. 예를 들어 **tpm2**가 **true**로 설정되어 있고 두 개의 **Tang** 서버를 지정하면 **TPM** 보안 암호화 프로세서를 사용할 수 없는 경우에도 두 개의 **Tang** 서버에 액세스하여 임계값 **2**를 충족할 수 있습니다.

20.1.4.2. 디스크 미러링 정보

컨트롤 플레인 및 작업자 노드에 **OpenShift Container Platform**을 설치하는 동안 부팅 디스크를 두 개 이상의 중복 스토리지 장치로 미러링할 수 있습니다. 하나의 장치를 사용할 수 있는 한 노드가 스토리지 장치 오류 후에도 계속 작동합니다.

미러링은 실패한 디스크 교체를 지원하지 않습니다. 미러를 성능이 저하되지 않은 초기 상태로 복원하려면 노드를 다시 프로비저닝합니다.

참고

사용자 프로비저닝 인프라 배포의 경우 **RHCOS** 시스템에서만 미러링을 사용할 수 있습니다. 미러링 지원은 **BIOS** 또는 **UEFI** 및 **ppc64le** 노드에서 부팅되는 **x86_64** 노드에서 사용할 수 있습니다.

20.1.4.3. 디스크 암호화 및 미러링 구성

OpenShift Container Platform을 설치하는 동안 암호화 및 미러링을 활성화하고 구성할 수 있습니다.

사전 요구 사항

- 설치 노드에 **OpenShift Container Platform** 설치 프로그램이 다운로드되어 있습니다.
- 설치 노드에 **Butane**이 설치되어 있습니다.

참고

butane은 OpenShift Container Platform이 머신 구성 작성에 편리한 단기 구문을 제공하고 머신 구성에 대한 추가 검증을 수행하는 데 사용하는 명령줄 유틸리티입니다. 자세한 내용은 *Butane을 사용하여 머신 구성 생성* 섹션을 참조하십시오.

- Tang 교환 키의 지문을 생성하는 데 사용할 수 있는 RHEL (Red Hat Enterprise Linux) 8 시스템에 액세스할 수 있습니다.

프로세스

1. TPM v2를 사용하여 클러스터를 암호화하려면 각 노드의 BIOS에서 TPM v2 암호화를 사용하도록 설정해야 하는지 확인합니다. 이는 대부분의 Dell 시스템에서 필요합니다. 컴퓨터 설명서를 확인하십시오.

2. Tang을 사용하여 클러스터를 암호화하려면 다음 준비 단계를 수행하십시오.

   a. Tang 서버를 설정하거나 기존 서버에 액세스합니다. 자세한 내용은 Network-bound disk encryption에서 참조하십시오.

   b. 아직 설치되지 않은 경우 RHEL 8 시스템에 clevis 패키지를 설치합니다.

   ```
   $ sudo yum install clevis
   ```

   c. RHEL 8 시스템에서 다음 명령을 실행하여 교환 키의 지문을 생성합니다. http://tang.example.com:7500을 Tang 서버의 URL로 바꿉니다.

   ```
   $ clevis-encrypt-tang '{"url":"http://tang.example.com:7500"}' < /dev/null > /dev/null
   (1)
   ```

   (1) 이 예에서 tangd.socket은 Tang 서버의 포트 7500에서 수신 대기 중입니다.

참고

clevis-encrypt-tang 명령은 이 단계에서 교환 키의 지문을 생성하는 데만 사용됩니다. 이때 데이터가 암호화를 위해 명령에 전달되지 않으므로 **/dev/null** 은 일반 텍스트 대신 입력으로 제공됩니다. 이 프로세스에 필요하지 않으므로 암호화된 출력은 **/dev/null**에도 전송됩니다.

출력 예

```
The advertisement contains the following signing keys:

PLjNyRdGw03zlRoGjQYMahSZGu9 1
```

1

교환 키의 지문입니다.

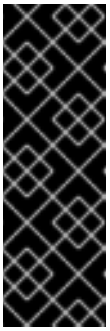
Do you wish to trust these keys? [ynYN] 프롬프트가 표시되면 **Y**를 입력합니다.

참고

RHEL 8에서는 **SHA-1** 해시 알고리즘을 사용하여 지문을 생성하는 **Clevis** 버전 **15**를 제공합니다. 다른 일부 배포에서는 지문에 **SHA-256** 해시 알고리즘을 사용하는 **Clevis** 버전 **17** 이상을 제공합니다. **OpenShift Container Platform** 클러스터 노드에 **RHCOS(Red Hat Enterprise Linux CoreOS)**를 설치할 때 **Clevis** 바인딩 문제를 방지하려면 **SHA-1**을 사용하여 지문을 생성하는 **Clevis** 버전을 사용해야 합니다.

d.

노드가 고정 **IP** 주소로 구성된 경우 **RHCOS** 노드를 설치할 때 **coreos-installer --append-karg** 옵션을 사용하여 설치된 시스템의 **IP** 주소를 설정합니다. 네트워크에 필요한 **ip=** 및 기타 인수를 추가합니다.



중요

정적 IP를 구성하는 일부 방법은 첫 번째 부팅 후 initramfs에 영향을 미치지 않으며 Tang 암호화에서는 작동하지 않습니다. 여기에는 coreos-installer --copy-network 옵션과 설치 중에 라이브 ISO 또는 PXE 이미지의 커널 명령줄에 ip= 인수가 포함됩니다. 잘못된 고정 IP 구성으로 노드의 두 번째 부팅이 실패합니다.

3. 설치 노드에서 설치 프로그램이 포함된 디렉터리로 변경하고 클러스터에 대한 Kubernetes 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 <installation_directory>를 설치 파일을 저장하려는 디렉토리의 경로로 바꿉니다.

4. 디스크 암호화, 미러링 또는 둘 다를 구성하는 Butane 구성을 생성합니다. 예를 들어 컴퓨팅 노드의 스토리지를 구성하려면 $HOME/clusterconfig/worker-storage.bu 파일을 생성합니다.

부팅 장치의 butane config 예

```
variant: openshift
version: 4.9.0
metadata:
  name: worker-storage 1
  labels:
    machineconfiguration.openshift.io/role: worker 2
boot_device:
  layout: x86_64 3
  luks: 4
    tpm2: true 5
    tang: 6
      - url: http://tang.example.com:7500 7
        thumbprint: PLjNyRdGw03zlRoGjQYMahSZGu9 8
    threshold: 1 9
  mirror: 10
    devices: 11
      - /dev/sda
      - /dev/sdb
openshift:
  fips: true 12
```

1 2

컨트롤 플레인 구성의 경우 두 위치 모두에서 **worker**를 **master**로 바꿉니다.

3

ppc64le 노드에서 이 필드를 **ppc64le**로 설정합니다. 다른 모든 노드에서 이 필드를 생략할 수 있습니다.

4

root 파일 시스템을 암호화하려면 이 섹션을 포함합니다. 자세한 내용은 *디스크 암호화 정보* 섹션을 참조하십시오.

5

Trusted Platform Module (TPM)을 사용하여 **root** 파일 시스템을 암호화하려면 이 필드를 포함합니다.

6

하나 이상의 **Tang** 서버를 사용하려면 이 섹션을 포함합니다.

7

Tang 서버의 **URL**을 지정합니다. 이 예에서 **tangd.socket**은 **Tang** 서버의 포트 **7500**에서 수신 대기 중입니다.

8

이전 단계에서 생성된 교환 키 지문을 지정합니다.

9

암호 해독을 수행하려면 충족해야 하는 최소 **TPM v2** 및 **Tang** 암호화 조건 수를 지정합니다. 기본값은 **1** 입니다. 보다 자세한 내용은 *암호화 임계값 구성* 섹션을 참조하십시오.

10

부팅 디스크를 미러링하려면 이 섹션을 포함합니다. 자세한 내용은 *디스크 미러링 정보*를 참조하십시오.

11

12

클러스터에서 **FIPS** 모드를 활성화하려면 이 지시문을 포함합니다.

중요

디스크 암호화와 미러링을 모두 사용하도록 노드를 구성하는 경우 두 기능을 동일한 **Butane config**에서 구성해야 합니다. 또한 **FIPS** 모드가 활성화된 노드에서 디스크 암호화를 구성하는 경우 별도의 매니페스트에서 **FIPS** 모드도 활성화된 경우에도 동일한 **Butane** 구성에 **fips** 지시문을 포함해야 합니다.

5. 해당 **Butane** 구성에서 컨트롤 플레인 또는 컴퓨팅 노드 매니페스트를 생성하여 **<installation_directory>/openshift** 디렉터리에 저장합니다. 예를 들어 컴퓨팅 노드에 대한 매니페스트를 생성하려면 다음 명령을 실행합니다.

```
$ butane $HOME/clusterconfig/worker-storage.bu -o
<installation_directory>/openshift/99-worker-storage.yaml
```

디스크 암호화 또는 미러링이 필요한 각 노드 유형에 대해 이 단계를 반복합니다.

6. 나중에 매니페스트를 업데이트해야 하는 경우 **Butane** 구성을 저장합니다.

7. 나머지 **OpenShift Container Platform** 설치를 계속합니다.

작은 정보

디스크 암호화 또는 미러링과 관련된 오류 메시지가 설치 중에 **RHCOS** 노드에서 콘솔 로그를 모니터링할 수 있습니다.

중요

추가 데이터 파티션을 구성하면 암호화가 명시적으로 요청되지 않는 한 암호화되지 않습니다.

검증

OpenShift Container Platform을 설치한 후 클러스터 노드에서 부팅 디스크 암호화 또는 미러링이 활성화되어 있는지 확인할 수 있습니다.

1. 설치 호스트에서 디버그 **Pod**를 사용하여 클러스터 노드에 액세스합니다.

   a. 노드의 디버그 **pod**를 시작합니다. 다음 예제에서는 **compute-1** 노드의 디버그 **Pod**를 시작합니다.

   ```
   $ oc debug node/compute-1
   ```

   b. 디버그 쉘 내에서 **/host**를 **root** 디렉터리로 설정합니다. 디버그 **Pod**는 **Pod** 내의 **/host**에 노드의 **root** 파일 시스템을 마운트합니다. **root** 디렉토리를 **/host**로 변경하면 노드의 실행 경로에 포함된 바이너리를 실행할 수 있습니다.

   ```
   # chroot /host
   ```

   참고

   Red Hat Enterprise Linux CoreOS (RHCOS)를 실행하는 **OpenShift Container Platform** 클러스터 노드는 변경할 수 없으며 **Operator**를 통해 클러스터 변경 사항을 적용합니다. **SSH**를 사용하여 클러스터 노드에 액세스하는 것은 권장되지 않습니다. 그러나 **OpenShift Container Platform API**를 사용할 수 없거나 **kubelet**이 대상 노드에서 제대로 작동하지 않는 경우 **oc** 작업이 영향을 받습니다. 이러한 상황에서 대신 **ssh core @ <node>. <cluster_name>.<base_domain>**을 사용하여 노드에 액세스할 수 있습니다.

2. 부팅 디스크 암호화를 구성한 경우 활성화되어 있는지 확인합니다.

   a. 디버그 쉘에서 노드의 **root** 매핑 상태를 검토합니다.

   ```
   # cryptsetup status root
   ```

   출력 예

   ```
   /dev/mapper/root is active and is in use.
     type:    LUKS2 1
     cipher:  aes-xts-plain64 2
   ```

```
keysize: 512 bits
key location: keyring
device:  /dev/sda4 3
sector size:  512
offset:  32768 sectors
size:    15683456 sectors
mode:    read/write
```

1

암호화 형식입니다. **TPM v2** 또는 **Tang** 암호화 모드가 활성화되면 **RHCOS** 부팅 디스크는 **LUKS2** 형식을 사용하여 암호화됩니다.

2

LUKS2 볼륨을 암호화하는 데 사용되는 암호화 알고리즘입니다. **FIPS** 모드가 활성화된 경우 **aes-cbc-essiv:sha256** 암호가 사용됩니다.

3

암호화된 **LUKS2** 볼륨을 포함하는 장치입니다. 미러링이 활성화된 경우 값은 소프트웨어 미러 장치(예: **/dev/md126**)를 나타냅니다.

b.

암호화된 장치에 바인딩된 **Clevis** 플러그인을 나열합니다.

```
# clevis luks list -d /dev/sda4 1
```

1

이전 단계의 출력에서 **device** 필드에 나열된 장치를 지정합니다.

출력 예

```
1: sss '{"t":1,"pins":{"tang":[{"url":"http://tang.example.com:7500"}]}}' 1
```

1

3. 미러링을 구성한 경우 활성화되어 있는지 확인합니다.

a. 디버그 쉘에서 노드의 소프트웨어 **RAID** 장치를 나열합니다.

```
# cat /proc/mdstat
```

출력 예

```
Personalities : [raid1]
md126 : active raid1 sdb3[1] sda3[0] 1
  393152 blocks super 1.0 [2/2] [UU]

md127 : active raid1 sda4[0] sdb4[1] 2
  51869632 blocks super 1.2 [2/2] [UU]

unused devices: <none>
```

1 이 예에서 **/dev/md126** 소프트웨어 **RAID** 미러 장치는 클러스터 노드에서 **/dev/sda3** 및 **/dev/sdb3** 디스크 장치를 사용합니다.

2 이 예에서 **/dev/md127** 소프트웨어 **RAID** 미러 장치는 클러스터 노드에서 **/dev/sda4** 및 **/dev/sdb4** 디스크 장치를 사용합니다.

b. 이전 명령의 출력에 나열된 각 소프트웨어 **RAID** 장치의 세부 정보를 검토합니다. 다음 예제에서는 **/dev/md126** 장치의 세부 정보를 나열합니다.

```
# mdadm --detail /dev/md126
```

출력 예

```
/dev/md126:
           Version : 1.0
     Creation Time : Wed Jul  7 11:07:36 2021
        Raid Level : raid1 (1)
        Array Size : 393152 (383.94 MiB 402.59 MB)
     Used Dev Size : 393152 (383.94 MiB 402.59 MB)
      Raid Devices : 2
     Total Devices : 2
       Persistence : Superblock is persistent

       Update Time : Wed Jul  7 11:18:24 2021
             State : clean (2)
    Active Devices : 2 (3)
   Working Devices : 2 (4)
    Failed Devices : 0 (5)
     Spare Devices : 0

Consistency Policy : resync

              Name : any:md-boot (6)
              UUID : ccfa3801:c520e0b5:2bee2755:69043055
            Events : 19

    Number   Major   Minor   RaidDevice State
       0     252       3        0      active sync   /dev/sda3 (7)
       1     252      19        1      active sync   /dev/sdb3 (8)
```

(1)

장치의 **RAID** 수준을 지정합니다. **RAID1** 은 **RAID 1** 디스크 미러링을 나타냅니다.

(2)

RAID 장치의 상태를 지정합니다.

(3) (4)

활성 상태이고 작동 중인 기본 디스크 장치의 수를 나타냅니다.

(5)

실패 상태에 있는 기본 디스크 장치의 수를 나타냅니다.

(6)

7 8 소프트웨어 RAID 장치에서 사용하는 기본 디스크 장치에 대한 정보를 제공합니다.

c. 소프트웨어 RAID 장치에 마운트된 파일 시스템을 나열합니다.

```
# mount | grep /dev/md
```

출력 예

```
/dev/md127 on / type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /etc type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /usr type xfs
(ro,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /sysroot type xfs
(ro,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /var type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /var/lib/containers/storage/overlay type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /var/lib/kubelet/pods/e5054ed5-f882-4d14-b599-
99c050d4e0c0/volume-subpaths/etc/tuned/1 type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /var/lib/kubelet/pods/e5054ed5-f882-4d14-b599-
99c050d4e0c0/volume-subpaths/etc/tuned/2 type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /var/lib/kubelet/pods/e5054ed5-f882-4d14-b599-
99c050d4e0c0/volume-subpaths/etc/tuned/3 type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /var/lib/kubelet/pods/e5054ed5-f882-4d14-b599-
99c050d4e0c0/volume-subpaths/etc/tuned/4 type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /var/lib/kubelet/pods/e5054ed5-f882-4d14-b599-
99c050d4e0c0/volume-subpaths/etc/tuned/5 type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md126 on /boot type ext4 (rw,relatime,seclabel)
```

예제 출력에서 /boot 파일 시스템은 dev/md126 소프트웨어 RAID 장치에 마운트되고 root 파일 시스템은 /dev/md127에 마운트됩니다.

4. 각 OpenShift Container Platform 노드 유형에 대한 확인 단계를 반복합니다.

추가 리소스

- TPM v2 및 Tang 암호화 모드에 대한 자세한 내용은 정책 기반 암호 해독을 사용하여 암호화된 볼륨의 자동화된 잠금 해제 구성을 참조하십시오.

20.1.4.4. RAID 지원 데이터 볼륨 구성

소프트웨어 RAID 파티셔닝을 활성화하여 외부 데이터 볼륨을 제공할 수 있습니다. OpenShift Container Platform은 데이터 보호 및 내결함성을 위해 RAID 0, RAID 1, RAID 4, RAID 5, RAID 6 및 RAID 10을 지원합니다. 자세한 내용은 "디스크 미러링 정보"를 참조하십시오.

사전 요구 사항

- 설치 노드에 OpenShift Container Platform 설치 프로그램이 다운로드되어 있습니다.
- 설치 노드에 Butane을 설치했습니다.

참고

butane은 OpenShift Container Platform이 머신 구성 작성에 편리한 단기 구문을 제공하고 머신 구성에 대한 추가 검증을 수행하는 데 사용하는 명령줄 유틸리티입니다. 자세한 내용은 *Butane을 사용하여 머신 구성 생성* 섹션을 참조하십시오.

절차

1. 소프트웨어 RAID를 사용하여 데이터 볼륨을 구성하는 Butane 구성을 만듭니다.
   - 미러링된 부팅 디스크에 사용되는 것과 동일한 디스크에 RAID 1을 사용하여 데이터 볼륨을 구성하려면 $HOME/clusterconfig/raid1-storage.bu 파일을 생성합니다. 예를 들면 다음과 같습니다.

미러링된 부팅 디스크의 RAID 1

```
variant: openshift
version: 4.9.0
metadata:
  name: raid1-storage
  labels:
    machineconfiguration.openshift.io/role: worker
boot_device:
  mirror:
    devices:
      - /dev/sda
      - /dev/sdb
storage:
  disks:
    - device: /dev/sda
      partitions:
        - label: root-1
          size_mib: 25000 (1)
        - label: var-1
    - device: /dev/sdb
      partitions:
        - label: root-2
          size_mib: 25000 (2)
        - label: var-2
  raid:
    - name: md-var
      level: raid1
      devices:
        - /dev/disk/by-partlabel/var-1
        - /dev/disk/by-partlabel/var-2
  filesystems:
    - device: /dev/md/md-var
      path: /var
      format: xfs
      wipe_filesystem: true
      with_mount_unit: true
```

(1) (2)

데이터 파티션을 부트 디스크에 추가할 때 최소 **25000** 메비 바이트가 권장됩니다. 값이 지정되지 않거나 지정된 값이 권장 최소값보다 작으면 생성되는 루트 파일 시스템의 크기가 너무 작아지고 **RHCOS**를 나중에 다시 설치할 때 데이터 파티션의 첫 번째 부분을 덮어 쓸 수 있습니다.

- 보조 디스크에서 **RAID 1**을 사용하여 데이터 볼륨을 구성하려면 **$HOME/clusterconfig/raid1-alt-storage.bu** 파일을 생성합니다. 예를 들면 다음과 같습니다.

보조 디스크에서 RAID 1

```
variant: openshift
version: 4.9.0
metadata:
  name: raid1-alt-storage
  labels:
    machineconfiguration.openshift.io/role: worker
storage:
  disks:
    - device: /dev/sdc
      wipe_table: true
      partitions:
        - label: data-1
    - device: /dev/sdd
      wipe_table: true
      partitions:
        - label: data-2
  raid:
    - name: md-var-lib-containers
      level: raid1
      devices:
        - /dev/disk/by-partlabel/data-1
        - /dev/disk/by-partlabel/data-2
  filesystems:
    - device: /dev/md/md-var-lib-containers
      path: /var/lib/containers
      format: xfs
      wipe_filesystem: true
      with_mount_unit: true
```

2. 이전 단계에서 생성한 Butane config에서 RAID 매니페스트를 생성하여 < installation_directory>/openshift 디렉터리에 저장합니다. 예를 들어 컴퓨팅 노드에 대한 매니페스트를 생성하려면 다음 명령을 실행합니다.

```
$ butane $HOME/clusterconfig/<butane_config>.bu -o
<installation_directory>/openshift/<manifest_name>.yaml 1
```

1 &lt ;butane_config > 및 < manifest_name >을 이전 단계의 파일 이름으로 바꿉니다. 예를 들어 보조 디스크의 경우 raid1-alt-storage.bu 및 raid1-alt-storage.yaml입니다.

3. 나중에 매니페스트를 업데이트해야 하는 경우 **Butane config**를 저장합니다.

4. 나머지 **OpenShift Container Platform** 설치를 계속합니다.

20.1.5. chrony 타임 서비스 설정

chrony.conf 파일의 내용을 수정하고 해당 내용을 머신 구성으로 노드에 전달하여 **chrony** 타임 서비스 **(chronyd)**에서 사용하는 시간 서버 및 관련 구성을 설정할 수 있습니다.

절차

1. **chrony.conf** 파일의 내용을 포함하여 **Butane config**를 만듭니다. 예를 들어 작업자 노드에 **chrony**를 구성하려면 **99-worker-chrony.bu** 파일을 만듭니다.

참고

Butane에 대한 자세한 내용은 **"Butane** 을 사용하여 머신 구성 생성**"**을 참조하십시오.

```
variant: openshift
version: 4.9.0
metadata:
  name: 99-worker-chrony (1)
  labels:
    machineconfiguration.openshift.io/role: worker (2)
storage:
  files:
  - path: /etc/chrony.conf
    mode: 0644 (3)
    overwrite: true
    contents:
      inline: |
        pool 0.rhel.pool.ntp.org iburst (4)
        driftfile /var/lib/chrony/drift
        makestep 1.0 3
        rtcsync
        logdir /var/log/chrony
```

(1) (2)

컨트롤 플레인 노드에서 두 위치에 있는 **master**를 **worker**로 대체합니다.

(3)

4

2. Butane을 사용하여 노드에 전달할 구성이 포함된 MachineConfig 파일 99-worker-chrony.yaml을 생성합니다.

```
$ butane 99-worker-chrony.bu -o 99-worker-chrony.yaml
```

3. 다음 두 가지 방법 중 하나로 설정을 적용하십시오.

- 클러스터가 아직 실행되지 않은 경우 매니페스트 파일을 생성한 후 <installation_directory>/openshift 디렉터리에 MachineConfig 개체 파일을 추가한 다음 클러스터를 계속 작성합니다.
- 클러스터가 이미 실행중인 경우 다음과 같은 파일을 적용합니다.

```
$ oc apply -f ./99-worker-chrony.yaml
```

20.1.6. 추가 리소스

- Butane에 대한 자세한 내용은 Butane 을 사용하여 머신 구성 생성을 참조하십시오.
- FIPS 지원에 대한 자세한 내용은 FIPS 암호화 지원을 참조하십시오.

20.2. 방화벽 설정

방화벽을 사용하는 경우 OpenShift Container Platform이 작동하는데 필요한 사이트에 액세스할 수 있도록 방화벽을 설정해야 합니다. 일부 사이트에 대한 액세스 권한을 부여하고 Red Hat Insights, Telemetry 서비스, 클러스터를 호스팅하는 클라우드 및 특정 빌드 전략을 사용하는 경우 추가 액세스 권한을 부여해야 합니다.

20.2.1. OpenShift Container Platform의 방화벽 설정

OpenShift Container Platform을 설치하기 전에 OpenShift Container Platform에 필요한 사이트에 대한 액세스 권한을 부여하도록 방화벽을 설정해야 합니다.

컨트롤러 노드에서만 실행되는 서비스에 대한 특별한 설정 고려 사항은 작업자 노드와 비교되지 않습니다.

참고

환경에 OpenShift Container Platform 클러스터 앞에 전용 로드 밸런서가 있는 경우 방화벽과 로드 밸런서 간의 허용 목록을 확인하여 클러스터에 불필요한 네트워크 제한을 방지합니다.

절차

1. 다음 레지스트리 URL을 허용 목록에 추가합니다.

URL	포트	함수
registry.redhat.io	443, 80	코어 컨테이너 이미지를 제공합니다.
access.redhat.com	443, 80	코어 컨테이너 이미지를 제공합니다.
quay.io	443, 80	코어 컨테이너 이미지를 제공합니다.
cdn.quay.io	443, 80	코어 컨테이너 이미지를 제공합니다.
cdn01.quay.io	443, 80	코어 컨테이너 이미지를 제공합니다.
cdn02.quay.io	443, 80	코어 컨테이너 이미지를 제공합니다.
cdn03.quay.io	443, 80	코어 컨테이너 이미지를 제공합니다.
sso.redhat.com	443, 80	**https://console.redhat.com/openshift** 사이트에서 **sso.redhat.com**의 인증을 사용합니다.

허용 목록에 **cdn0[1-3].quay.io** 대신 와일드카드 ***.quay.io** 를 사용할 수 있습니다. 허용 목록에 **quay.io**와 같은 사이트를 추가할 때 ***.quay.io**와 같은 와일드카드 항목을 거부 목록에 추가하지 마십시오. 대부분의 경우 이미지 레지스트리는 **CDN(Content deliver network)**을 사용하여 이미지를 제공합니다. 방화벽 블록에 액세스하는 경우 초기 다운로드 요청이 **cdn01.quay.io** 와 같은 호스트 이름으로 리디렉션될 때 이미지 다운로드가 거부됩니다.

2. 빌드에 필요한 언어 또는 프레임 워크에 대한 리소스를 제공하는 사이트를 허용 목록에 추가합니다.

3. **Telemetry를 비활성화하지 않은 경우 Red Hat Insights에 액세스하려면 다음 URL에 대한 액세스 권한을 부여해야합니다**

URL	포트	함수
cert-api.access.redhat.com	443, 80	Telemetry 필수
api.access.redhat.com	443, 80	Telemetry 필수
infogw.api.openshift.com	443, 80	Telemetry 필수
console.redhat.com/api/ingress, **cloud.redhat.com/api/ingress**	443, 80	Telemetry 및 **insights-operator** 필수

4. **AWS (Amazon Web Services), Microsoft Azure 또는 Google Cloud Platform (GCP)을 사용하여 클러스터를 호스팅하는 경우 클라우드 공급자 API 및 DNS를 제공하는 URL에 대한 액세스 권한을 부여해야 합니다.**

클라우드	URL	포트	함수
AWS	***.amazonaws.com** 또는 AWS API에 와일드카드를 사용하지 않도록 선택하는 경우 다음 URL을 허용해야 합니다.	443, 80	AWS 서비스 및 리소스에 액세스하는데 필요합니다. 사용하는 리전에서 허용되는 특정 엔드 포인트를 확인하려면 AWS 설명서에서 AWS 서비스 엔드 포인트를 참조하십시오.
	ec2.amazonaws.com	443	AWS 환경에서 클러스터를 설치하고 관리하는 데 사용됩니다.
	events.amazonaws.com	443	AWS 환경에서 클러스터를 설치하고 관리하는 데 사용됩니다.
	iam.amazonaws.com	443	AWS 환경에서 클러스터를 설치하고 관리하는 데 사용됩니다.
	route53.amazonaws.com	443	AWS 환경에서 클러스터를 설치하고 관리하는 데 사용됩니다.
	s3.amazonaws.com	443	AWS 환경에서 클러스터를 설치하고 관리하는 데 사용됩니다.
	s3.<aws_region>.amazonaws.com	443	AWS 환경에서 클러스터를 설치하고 관리하는 데 사용됩니다.

클라우드	URL	포트	함수
	s3.dualstack.<aws_region>.amazonaws.com	443	AWS 환경에서 클러스터를 설치하고 관리하는 데 사용됩니다.
	sts.amazonaws.com	443	AWS 환경에서 클러스터를 설치하고 관리하는 데 사용됩니다.
	sts.<aws_region>.amazonaws.com	443	AWS 환경에서 클러스터를 설치하고 관리하는 데 사용됩니다.
	tagging.us-east-1.amazonaws.com	443	AWS 환경에서 클러스터를 설치하고 관리하는 데 사용됩니다. 이 끝점은 클러스터가 배포된 지역에 관계없이 항상 **us-east-1** 입니다.
	ec2.<aws_region>.amazonaws.com	443	AWS 환경에서 클러스터를 설치하고 관리하는 데 사용됩니다.
	elasticloadbalancing.<aws_region>.amazonaws.com	443	AWS 환경에서 클러스터를 설치하고 관리하는 데 사용됩니다.
	servicequotas.<aws_region>.amazonaws.com	443, 80	필수 항목입니다. 서비스 배포에 대한 할당량을 확인하는 데 사용됩니다.
	tagging.<aws_region>.amazonaws.com	443, 80	태그 형태로 AWS 리소스에 대한 메타데이터를 할당할 수 있습니다.
GCP	***.googleapis.com**	443, 80	GCP 서비스 및 리소스에 액세스하는 데 필요합니다. API에서 허용하는 엔드포인트를 확인하려면 GCP 설명서에서 Cloud Endpoints를 참조하십시오.
	accounts.google.com	443, 80	GCP 계정에 액세스하는 데 필요합니다.
Azure	**management.azure.com**	443, 80	Azure 서비스 및 리소스에 액세스하는 데 필요합니다. API에 허용되는 끝점을 확인하려면 Azure 문서의 Azure REST API 참조를 참조하십시오.
	***.blob.core.windows.net**	443, 80	Ignition 파일을 다운로드하는 데 필요합니다.
	login.microsoftonline.com	443, 80	Azure 서비스 및 리소스에 액세스하는 데 필요합니다. API에 허용되는 끝점을 확인하려면 Azure 문서의 Azure REST API 참조를 참조하십시오.

5. 다음 **URL**을 허용 목록에 추가하십시오.

URL	포트	함수
mirror.openshift.com	443, 80	미러링된 설치 콘텐츠 및 이미지에 액세스하는 데 필요합니다. Cluster Version Operator에는 단일 기능 소스만 필요하지만 이 사이트는 릴리스 이미지 서명의 소스이기도합니다.
storage.googleapis.com/openshift-release	443, 80	릴리스 이미지 서명 소스입니다 (Cluster Version Operator에는 단일 기능 소스만 필요)
***.apps.<cluster_name>.<base_domain>**	443, 80	설치 중에 ingress 와일드카드를 설정하지 않으면 기본 클러스터 라우트에 액세스하는데 필요합니다.
quayio-production-s3.s3.amazonaws.com	443, 80	AWS에서 Quay 이미지 콘텐츠에 액세스하는데 필요합니다.
api.openshift.com	443, 80	클러스터 토큰과 클러스터에 업데이트를 사용할 수 있는지 확인하는 데 필요합니다.
rhcos-redirector.apps.art.xq1c.p1.openshiftapps.com, **rhcos.mirror.openshift.com**	443, 80	RHCOS (Red Hat Enterprise Linux CoreOS) 이미지를 다운로드하는 데 필요합니다.
console.redhat.com/openshift	443, 80	클러스터 토큰에 필요합니다.
registry.access.redhat.com	443, 80	**odo** CLI에 필요합니다.
sso.redhat.com	443, 80	**https://console.redhat.com/openshift** 사이트에서 **sso.redhat.com**의 인증을 사용합니다.

Operator는 상태 확인을 수행하기 위해 경로 액세스가 필요합니다. 특히 인증 및 웹 콘솔 **Operator**는 두 경로에 연결하여 경로가 작동하는지 확인합니다. 클러스터 관리자이고 * **.apps.**

<cluster_name>.<base_domain>을 허용하지 않으려면 다음 경로를 허용하십시오.

- **oauth-openshift.apps.<cluster_name>.<base_domain>**
- **console-openshift-console.apps. <cluster_name>.<base_domain> 또는 필드가 비어 있지 않은 경우 consoles.operator/cluster 객체의 spec.route.hostname 필드에 지정된 호스트 이름입니다.**

6. **선택적 타사 콘텐츠에 대해 다음 URL을 허용 목록에 추가합니다.**

URL	포트	함수
registry.connect.redhat.com	443, 80	모든 타사 이미지 및 인증된 운영자에 필요합니다.
rhc4tp-prod-z8cxf-image-registry-us-east-1-evenkyleffocxqvofrk.s3.dualstack.us-east-1.amazonaws.com	443, 80	**registry.connect.redhat.com**에서 호스팅되는 컨테이너 이미지에 대한 액세스 제공
oso-rhc4tp-docker-registry.s3-us-west-2.amazonaws.com	443, 80	Sonatype Nexus, F5 빅 IP 운영자에 필요합니다.

7. **기본 NTP(Red Hat Network Time Protocol) 서버를 사용하는 경우 다음 URL을 허용합니다.**

- **1.rhel.pool.ntp.org**
- **2.rhel.pool.ntp.org**
- **3.rhel.pool.ntp.org**

참고

기본 **Red Hat NTP** 서버를 사용하지 않는 경우 플랫폼의 **NTP** 서버를 확인하고 방화벽에서 허용합니다.

21장. 설치 검증

이 문서의 절차에 따라 설치 후 **OpenShift Container Platform** 클러스터의 상태를 검증할 수 있습니다.

21.1. 설치 로그 검토

OpenShift Container Platform 설치 로그에서 설치 요약을 검토할 수 있습니다. 설치에 성공하면 클러스터에 액세스하는 데 필요한 정보가 로그에 포함됩니다.

사전 요구 사항

- 설치 호스트에 대한 액세스 권한이 있어야 합니다.

프로세스

- 설치 호스트의 설치 디렉터리에서 **.openshift_install.log** 로그 파일을 검토합니다.

```
$ cat <install_dir>/.openshift_install.log
```

출력 예

다음 예에 설명된 대로 설치에 성공하면 클러스터 인증 정보가 로그 끝에 포함됩니다.

```
...
time="2020-12-03T09:50:47Z" level=info msg="Install complete!"
time="2020-12-03T09:50:47Z" level=info msg="To access the cluster as the
system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'"
time="2020-12-03T09:50:47Z" level=info msg="Access the OpenShift web-console
here: https://console-openshift-console.apps.mycluster.example.com"
time="2020-12-03T09:50:47Z" level=info msg="Login to the console with user:
\"kubeadmin\", and password: \"6zYIx-ckbW3-4d2Ne-IWvDF\""
time="2020-12-03T09:50:47Z" level=debug msg="Time elapsed per stage:"
time="2020-12-03T09:50:47Z" level=debug msg="    Infrastructure: 6m45s"
time="2020-12-03T09:50:47Z" level=debug msg="Bootstrap Complete: 11m30s"
time="2020-12-03T09:50:47Z" level=debug msg=" Bootstrap Destroy: 1m5s"
time="2020-12-03T09:50:47Z" level=debug msg=" Cluster Operators: 17m31s"
time="2020-12-03T09:50:47Z" level=info msg="Time elapsed: 37m26s"
```

21.2. 이미지 풀 소스 보기

무제한 네트워크 연결이 있는 클러스터의 경우 **crictl images**와 같은 노드에서 명령을 사용하여 가져온

이미지의 소스를 볼 수 있습니다.

그러나 연결이 끊긴 설치의 경우 가져온 이미지 소스를 보려면 **CRI-O** 로그를 검토하여 다음 절차에 표시된 대로 로그 항목에 **Trying to access**를 찾아야 합니다. **crictl images** 명령과 같은 이미지 가져오기 소스를 보는 다른 방법은 미러링된 위치에서 이미지를 가져오는 경우에도 미러링되지 않은 이미지 이름을 표시합니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

프로세스

- 마스터 또는 작업자 노드의 **CRI-O** 로그를 확인합니다.

```
$  oc adm node-logs <node_name> -u crio
```

출력 예

Trying to access 로그 항목은 이미지를 가져오는 위치를 나타냅니다.

```
...
Mar 17 02:52:50 ip-10-0-138-140.ec2.internal crio[1366]: time="2021-08-05
10:33:21.594930907Z" level=info msg="Pulling image: quay.io/openshift-release-
dev/ocp-release:4.9.0-ppc64le" id=abcd713b-d0e1-4844-ac1c-474c5b60c07c
name=/runtime.v1alpha2.ImageService/PullImage
Mar 17 02:52:50 ip-10-0-138-140.ec2.internal crio[1484]: time="2021-03-17
02:52:50.194341109Z" level=info msg="Trying to access \"li0317gcp1.mirror-
registry.qe.gcp.devcluster.openshift.com:5000/ocp/release@sha256:1926eae7cacb9c0
0f142ec98b00628970e974284b6ddaf9a6a086cb9af7a6c31\""
Mar 17 02:52:50 ip-10-0-138-140.ec2.internal crio[1484]: time="2021-03-17
02:52:50.226788351Z" level=info msg="Trying to access \"li0317gcp1.mirror-
registry.qe.gcp.devcluster.openshift.com:5000/ocp/release@sha256:1926eae7cacb9c0
0f142ec98b00628970e974284b6ddaf9a6a086cb9af7a6c31\""
...
```

위 예제와 같이 로그에 이미지 가져오기 소스가 두 번 표시될 수 있습니다.

ImageContentSourcePolicy 오브젝트가 여러 미러를 나열하는 경우 **OpenShift Container Platform**은 구성에 나열된 순서대로 이미지를 가져오려고 시도합니다. 예를 들면 다음과 같습니다.

```
Trying to access \"li0317gcp1.mirror-
registry.qe.gcp.devcluster.openshift.com:5000/ocp/release@sha256:1926eae7cacb9c00f142ec
98b00628970e974284b6ddaf9a6a086cb9af7a6c31\"
Trying to access \"li0317gcp2.mirror-
registry.qe.gcp.devcluster.openshift.com:5000/ocp/release@sha256:1926eae7cacb9c00f142ec
98b00628970e974284b6ddaf9a6a086cb9af7a6c31\"
```

21.3. 클러스터 버전, 상태 및 업데이트 세부 정보 가져오기

oc get clusterversion 명령을 실행하여 클러스터 버전 및 상태를 검토할 수 있습니다. 설치가 여전히 진행 중임을 표시하는 경우 자세한 내용은 **Operator**의 상태를 검토할 수 있습니다.

현재 업데이트 채널을 나열하고 사용 가능한 클러스터 업데이트를 검토할 수도 있습니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- **OpenShift CLI(oc)**가 설치되어 있습니다.

프로세스

1. 클러스터 버전 및 전체 상태를 가져옵니다.

```
$ oc get clusterversion
```

출력 예

```
NAME      VERSION   AVAILABLE   PROGRESSING   SINCE   STATUS
version   4.6.4     True        False         6m25s   Cluster version is 4.6.4
```

예제 출력은 클러스터가 성공적으로 설치되었음을 나타냅니다.

2. 클러스터 상태가 설치가 여전히 진행 중임을 나타내는 경우 **Operator** 상태를 확인하여 더 자

세한 진행 정보를 얻을 수 있습니다.

```
$ oc get clusteroperators.config.openshift.io
```

3. 클러스터 사양, 업데이트 가용성 및 업데이트 기록에 대한 자세한 요약을 확인합니다.

```
$ oc describe clusterversion
```

4. 현재 업데이트 채널을 나열합니다.

```
$ oc get clusterversion -o jsonpath='{.items[0].spec}{"\n"}'
```

출력 예

```
{"channel":"stable-4.6","clusterID":"245539c1-72a3-41aa-9cec-72ed8cf25c5c"}
```

5. 사용 가능한 클러스터 업데이트를 검토합니다.

```
$ oc adm upgrade
```

출력 예

```
Cluster version is 4.6.4

Updates:

VERSION IMAGE
4.6.6   quay.io/openshift-release-dev/ocp-release@sha256:c7e8f18e8116356701bd23ae3a23fb9892dd5ea66c8300662ef30563d7104f39
```

추가 리소스

- 설치가 계속 진행 중인 경우 Operator 상태를 쿼리하는 방법에 대한 자세한 내용은 설치 후 Operator 상태 쿼리를 참조하십시오.
- Operator 문제 조사에 대한 정보는 Operator 문제 해결을 참조하십시오.
- 클러스터 업데이트에 대한 자세한 내용은 클러스터 업데이트를 참조하십시오.
- 업그레이드 릴리스 채널에 대한 개요는 OpenShift Container Platform 업그레이드 채널 및 릴리스를 참조하십시오.

21.4. CLI를 사용하여 클러스터 노드의 상태 쿼리

설치 후 클러스터 노드의 상태를 확인할 수 있습니다.

사전 요구 사항

- cluster-admin 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift CLI(oc)가 설치되어 있습니다.

프로세스

1. 클러스터 노드의 상태를 나열합니다. 출력에 모든 예상 컨트롤 플레인 및 컴퓨팅 노드가 나열되고 각 노드에 Ready 상태가 있는지 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME                         STATUS   ROLES    AGE   VERSION
compute-1.example.com        Ready    worker   33m   v1.22.1
control-plane-1.example.com  Ready    master   41m   v1.22.1
control-plane-2.example.com  Ready    master   45m   v1.22.1
compute-2.example.com        Ready    worker   38m   v1.22.1
compute-3.example.com        Ready    worker   33m   v1.22.1
control-plane-3.example.com  Ready    master   41m   v1.22.1
```

2. 각 클러스터 노드의 CPU 및 메모리 리소스 가용성을 검토합니다.

```
$ oc adm top nodes
```

출력 예

```
NAME                         CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
compute-1.example.com        128m         8%     1132Mi          16%
control-plane-1.example.com  801m         22%    3471Mi          23%
control-plane-2.example.com  1718m        49%    6085Mi          40%
compute-2.example.com        935m         62%    5178Mi          75%
compute-3.example.com        111m         7%     1131Mi          16%
control-plane-3.example.com  942m         26%    4100Mi          27%
```

추가 리소스

- 노드 상태 검토 및 노드 문제 조사에 대한 자세한 내용은 노드 상태 확인을 참조하십시오.

21.5. OPENSHIFT CONTAINER PLATFORM 웹 콘솔에서 클러스터 상태 검토

OpenShift Container Platform 웹 콘솔의 개요 페이지에서 다음 정보를 검토할 수 있습니다.

- 클러스터의 일반 상태
- 컨트롤 플레인, 클러스터 Operator 및 스토리지의 상태
- CPU, 메모리, 파일 시스템, 네트워크 전송, Pod 가용성
- 클러스터의 API 주소, 클러스터 ID 및 공급자의 이름

- 클러스터 버전 정보
- 현재 업데이트 채널 및 사용 가능한 업데이트를 포함하여 클러스터 업데이트 상태
- 노드, 포드, 스토리지 클래스 및 PVC(영구 볼륨 클레임) 정보를 설명하는 클러스터 인벤토리
- 진행 중인 클러스터 활동 및 최근 이벤트 목록

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

프로세스

- 관리자 관점에서 홈 → 개요로 이동합니다.

21.6. RED HAT OPENSHIFT CLUSTER MANAGER에서 클러스터 상태 검토

OpenShift Container Platform 웹 콘솔에서 **OpenShift Cluster Manager**의 클러스터 상태에 대한 자세한 정보를 확인할 수 있습니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

프로세스

1. 관리자 관점에서 홈 → 개요 → 세부 정보 → 클러스터 **ID** → **OpenShift Cluster Manager** 로 이동하여 **OpenShift Cluster Manager** 웹 콘솔에서 클러스터의 개요 탭을 엽니다.
2. OpenShift Cluster Manager 의 개요 탭에서 클러스터에 대한 다음 정보를 검토합니다.
   - **vCPU** 및 메모리 가용성 및 리소스 사용

- 클러스터 ID, 상태, 유형, 리전 및 공급자 이름
- 노드 유형별 노드 수
- 클러스터 버전 세부 정보, 클러스터 생성 날짜, 클러스터 소유자의 이름
- 클러스터의 라이프사이클 지원 상태
- SLA(서비스 수준 계약) 상태, 서브스크립션 단위 유형, 클러스터의 프로덕션 상태, 서브스크립션을 포함한 서브스크립션 정보

작은 정보

클러스터의 기록을 보려면 클러스터 기록 탭을 클릭합니다.

3. 모니터링 페이지로 이동하여 다음 정보를 검토합니다.
   - 감지된 모든 문제 목록
   - 실행 중인 경고 목록
   - 클러스터 Operator 상태 및 버전
   - 클러스터의 리소스 사용량
4. 선택 사항: 개요 메뉴로 이동하여 Red Hat Insights에서 수집하는 클러스터에 대한 정보를 볼 수 있습니다. 이 메뉴에서는 다음 정보를 볼 수 있습니다.
   - 위험 수준별로 분류된, 클러스터가 노출될 수 있는 잠재적인 문제

- 카테고리별 상태 검사 상태

추가 리소스

- 클러스터의 잠재적인 문제 검토에 대한 자세한 내용은 Insights를 사용하여 클러스터 문제 식별을 참조하십시오.

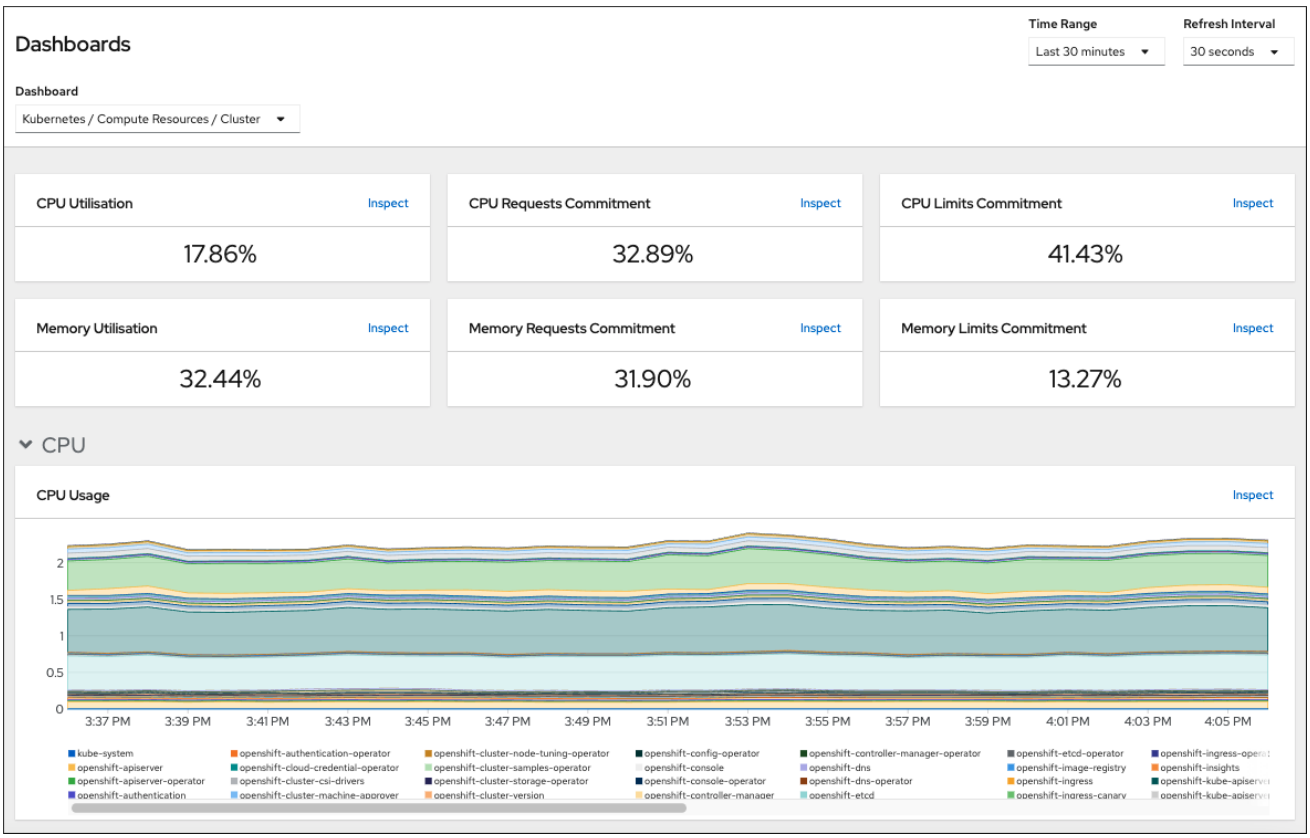
21.7. 클러스터 리소스 가용성 및 사용 여부 확인

OpenShift Container Platform은 클러스터 구성 요소의 상태를 이해하는 데 도움이 되는 포괄적인 모니터링 대시보드 세트를 제공합니다.

관리자 관점에서 다음을 포함하여 핵심 OpenShift Container Platform 구성 요소에 대한 대시보드에 액세스할 수 있습니다.

- etcd
- Kubernetes 컴퓨팅 리소스
- Kubernetes 네트워크 리소스
- Prometheus
- 클러스터 및 노드 성능과 관련된 대시보드

그림 21.1. 컴퓨팅 리소스 대시보드 예



사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

프로세스

1. **OpenShift Container Platform** 웹 콘솔의 관리자 관점에서 **Observe → Dashboards** 로 이동합니다.

2. 대시보드 목록에서 대시보드를 선택합니다. **etcd** 대시보드와 같은 일부 대시보드를 선택하면 추가 하위 메뉴가 생성됩니다.

3. 선택 사항: 시간 범위 목록에서 그래프의 시간 범위를 선택합니다.

   - 미리 정의된 기간을 선택합니다.

   - 시간 범위 목록에서 사용자 지정 시간 범위를 선택하여 사용자 지정 시간 범위를 설정합니다.

a. 시작 및 종료 날짜 및 시간을 입력하거나 선택합니다.

b. 저장을 클릭하여 사용자 지정 시간 범위를 저장합니다.

4. 선택 사항: 새로 고침 간격을 선택합니다.

5. 대시보드 내에서 각각의 그래프로 마우스를 이동하여 특정 항목에 대한 세부 정보를 표시합니다.

추가 리소스

- OpenShift Container Platform 모니터링 스택에 대한 자세한 내용은 모니터링 개요 를 참조하십시오.

21.8. 실행 중인 경고 나열

경고는 OpenShift Container Platform 클러스터에서 정의된 조건 집합이 적용되는 경우 알림을 제공합니다. OpenShift Container Platform 웹 콘솔에서 경고 UI를 사용하여 클러스터에서 실행되는 경고를 확인할 수 있습니다.

사전 요구 사항

- cluster-admin 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

프로세스

1. 관리자 관점에서 관찰 → 경고 → 경고 페이지로 이동합니다.

2. 심각도, 상태 및 소스를 포함하여 실행 중인 경고를 검토합니다.

3. 경고 세부 정보 페이지에서 자세한 정보를 보려면 경고를 선택합니다.

추가 리소스

- OpenShift Container Platform의 경고에 대한 자세한 내용은 알림 관리를 참조하십시오.

21.9. 다음 단계

- 클러스터를 설치할 때 문제가 발생하는 경우 설치 문제 해결을 참조하십시오.

- OpenShift Container Platform을 설치한 후 클러스터를 추가로 확장하고 사용자 정의할 수 있습니다.

22장. 설치 문제 해결

OpenShift Container Platform 설치 실패 문제를 해결하기 위해 부트스트랩 및 컨트롤 플레인 시스템에서 로그를 수집할 수 있습니다. 설치 프로그램에서 디버그 정보를 얻을 수도 있습니다. 로그 및 디버그 정보를 사용하여 문제를 해결할 수 없는 경우 구성 요소별 문제 해결에 대한 설치 문제 해결을 참조하십시오.

참고

OpenShift Container Platform 설치에 실패하고 디버그 출력 또는 로그에 네트워크 시간 초과 또는 기타 연결 오류가 포함된 경우 방화벽 구성에 대한 지침을 검토하십시오. 방화벽 및 로드 밸런서에서 로그를 수집하면 네트워크 관련 오류를 진단하는 데 도움이 될 수 있습니다.

22.1. 사전 요구 사항

- OpenShift Container Platform 클러스터를 설치하려고했으나 설치에 실패했습니다.

22.2. 실패한 설치에서 로그 수집

설치 프로그램에 SSH 키를 지정한 경우 실패한 설치에 대한 데이터를 수집할 수 있습니다.

참고

실패한 설치에 대한 로그를 수집하는데 사용되는 명령은 실행중인 클러스터에서 로그를 수집할 때 사용되는 명령과 다릅니다. 실행중인 클러스터에서 로그를 수집해야하는 경우 oc adm must-gather 명령을 사용하십시오.

사전 요구 사항

- 부트스트랩 프로세스가 완료되기 전에 OpenShift Container Platform 설치에 실패합니다. 부트스트랩 노드가 실행 중이며 SSH를 통해 액세스할 수 있습니다.
- ssh-agent 프로세스는 컴퓨터에서 활성화되어 있으며 ssh-agent 프로세스 및 설치 프로그램에 동일한 SSH 키를 제공하고 있습니다.
- 프로비저닝하는 인프라에 클러스터를 설치하려는 경우 부트스트랩 및 컨트롤 플레인 노드의 정규화된 도메인 이름이 있어야 합니다.

프로세스

1. 부트스트랩 및 컨트롤 플레인 시스템에서 설치 로그를 가져오는데 필요한 명령을 생성합니다.

- 설치 관리자 프로비저닝 인프라를 사용한 경우 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

```
$ ./openshift-install gather bootstrap --dir <installation_directory> 1
```

1 installation_directory는 ./openshift-install create cluster를 실행할 때 지정한 디렉터리입니다. 이 디렉터리에는 설치 프로그램이 생성한 OpenShift Container Platform 정의 파일이 포함되어 있습니다.

설치 프로그램이 프로비저닝한 인프라의 경우 설치 프로그램은 클러스터에 대한 정보를 저장하므로 호스트 이름 또는 IP 주소를 지정할 필요가 없습니다.

- 자체적으로 프로비저닝한 인프라를 사용한 경우 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

```
$ ./openshift-install gather bootstrap --dir <installation_directory> \ 1
  --bootstrap <bootstrap_address> \ 2
  --master <master_1_address> \ 3
  --master <master_2_address> \ 4
  --master <master_3_address>" 5
```

1 installation_directory의 경우 ./openshift-install create cluster를 실행할 때 지정한 것과 동일한 디렉터리를 지정합니다. 이 디렉터리에는 설치 프로그램이 생성한 OpenShift Container Platform 정의 파일이 포함되어 있습니다.

2 <bootstrap_address>는 클러스터 부트스트랩 시스템의 정규화된 도메인 이름 또는 IP 주소입니다.

3 4 5

참고

기본 클러스터에는 세 개의 컨트롤 플레인 시스템이 있습니다. 클러스터가 사용하는 수에 관계없이 표시된대로 모든 컨트롤 플레인 시스템을 나열합니다.

출력 예

```
INFO Pulling debug logs from the bootstrap machine
INFO Bootstrap gather logs captured here "<installation_directory>/log-bundle-<timestamp>.tar.gz"
```

설치 실패에 대한 Red Hat 지원 케이스를 만들 경우 압축된 로그를 케이스에 포함해야 합니다.

22.3. 호스트에 SSH 액세스를 통해 수동으로 로그 수집

must-gather 또는 자동화된 수집 방법이 작동하지 않는 경우 로그를 수동으로 수집합니다.

중요

기본적으로 OpenShift Container Platform 노드에 대한 SSH 액세스는 RHOSP(Red Hat OpenStack Platform) 기반 설치에서 비활성화됩니다.

사전 요구 사항

- 호스트에 대한 SSH 액세스 권한이 있어야합니다.

프로세스

1. 다음을 실행하여 journalctl 명령을 사용하여 부트스트랩 호스트에서 bootkube.service 서비스 로그를 수집합니다.

```
$ journalctl -b -f -u bootkube.service
```

2. **podman** 로그를 사용하여 부트스트랩 호스트의 컨테이너 로그를 수집합니다. 이는 호스트에서 모든 컨테이너 로그를 가져오기 위해 루프로 표시됩니다.

```
$ for pod in $(sudo podman ps -a -q); do sudo podman logs $pod; done
```

3. 또는 다음을 실행하여 **tail** 명령을 사용하여 호스트 컨테이너 로그를 수집합니다.

```
# tail -f /var/lib/containers/storage/overlay-containers/*/userdata/ctr.log
```

4. 다음과 같이 **journalctl** 명령을 사용하여 마스터 및 작업자 호스트에서 **kubelet.service** 및 **crio.service** 서비스 로그를 수집합니다.

```
$ journalctl -b -f -u kubelet.service -u crio.service
```

5. 다음과 같이 **tail** 명령을 사용하여 마스터 및 작업자 호스트 컨테이너 로그를 수집합니다.

```
$ sudo tail -f /var/log/containers/*
```

22.4. 호스트에 SSH 액세스없이 수동으로 로그 수집

must-gather 또는 자동화된 수집 방법이 작동하지 않는 경우 로그를 수동으로 수집합니다.

노드에 대한 **SSH** 액세스 권한이 없는 경우 시스템 저널에 액세스하여 호스트에서 발생하는 상황을 조사할 수 있습니다.

사전 요구 사항

- **OpenShift Container Platform** 설치가 완료되어야 합니다.
- **API** 서비스가 작동하고 있어야 합니다.
- 시스템 관리자 권한이 있어야 합니다.

프로세스

1. 다음을 실행하여 /var/log 아래의 journald 유닛 로그에 액세스합니다.

```
$ oc adm node-logs --role=master -u kubelet
```

2. 다음을 실행하여 /var/log 아래의 호스트 파일 경로에 액세스합니다.

```
$ oc adm node-logs --role=master --path=openshift-apiserver
```

22.5. 설치 프로그램에서 디버그 정보 검색

다음 조치 중 하나를 사용하여 설치 프로그램에서 디버그 정보를 얻을 수 있습니다.

- 숨겨진 .openshift_install.log 파일에서 이전 설치의 디버그 정보를 확인합니다. 예를 들면 다음과 같습니다.

```
$ cat ~/<installation_directory>/.openshift_install.log 1
```

1 installation_directory의 경우 ./openshift-install create cluster를 실행할 때 지정한 것과 동일한 디렉터리를 지정합니다.

- 설치 프로그램이 포함된 디렉터리로 변경하고 --log-level=debug로 다시 실행합니다.

```
$ ./openshift-install create cluster --dir <installation_directory> --log-level debug 1
```

1 installation_directory의 경우 ./openshift-install create cluster를 실행할 때 지정한 것과 동일한 디렉터리를 지정합니다.

22.6. OPENSHIFT CONTAINER PLATFORM 클러스터 다시 설치

실패한 OpenShift Container Platform 설치의 문제를 디버그하고 해결할 수 없는 경우 새로운 OpenShift Container Platform 클러스터를 설치하는 것이 좋습니다. 설치 프로세스를 다시 시작하기 전에 철저한 정리를 완료해야 합니다. 사용자 프로비저닝 인프라(UPI) 설치의 경우 클러스터를 수동으로 삭

제하고 연결된 모든 리소스를 삭제해야 합니다. 다음 절차에서는 설치 관리자 프로비저닝 인프라(IPI) 설치를 위한 절차입니다.

프로세스

1. 설치 디렉터리에서 숨겨진 설치 프로그램 상태 파일을 포함하여 클러스터를 삭제하고 클러스터와 관련된 모든 리소스를 제거합니다.

```
$ ./openshift-install destroy cluster --dir <installation_directory> 1
```

1 installation_directory는 ./openshift-install create cluster를 실행할 때 지정한 디렉터리입니다. 이 디렉터리에는 설치 프로그램이 생성한 OpenShift Container Platform 정의 파일이 포함되어 있습니다.

2. 클러스터를 다시 설치하기 전에 설치 디렉터리를 삭제합니다.

```
$ rm -rf <installation_directory>
```

3. 새로운 OpenShift Container Platform 클러스터를 설치하려면 절차를 따르십시오.

추가 리소스

- OpenShift Container Platform 클러스터 설치

23장. FIPS 암호화 지원

x86_64 아키텍처에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 라이브러리를 사용하는 OpenShift Container Platform 클러스터를 설치할 수 있습니다.

클러스터의 RHCOS (Red Hat Enterprise Linux CoreOS) 시스템의 경우 이 변경 사항은 사용자가 클러스터의 배치시 변경할 수 있는 클러스터 옵션을 제어하는 install-config.yaml 파일의 옵션 상태를 기반으로 시스템을 배치할 때 적용됩니다. Red Hat Enterprise Linux (RHEL) 시스템에서는 작업자 시스템으로 사용하려는 시스템에 운영 체제를 설치할 때 FIPS 모드를 활성화해야 합니다. 이러한 설정 방법을 사용하면 클러스터가 FIPS 준수 감사 요구 사항을 충족하는지 확인할 수 있습니다. 초기 시스템 부팅 전에 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 패키지만 활성화됩니다.

클러스터가 사용하는 운영 체제를 처음 부팅하기 전에 FIPS를 활성화해야하므로 클러스터를 배포한 후 FIPS를 활성화할 수 없습니다.

23.1. OPENSHIFT CONTAINER PLATFORM에서 FIPS 검증

OpenShift Container Platform은 이를 사용하는 운영 체제 구성 요소에 대해 RHEL 및 RHCOS에서 특정 FIPS 검증 / 진행중인 모듈 (Modules in Process) 모듈을 사용합니다. RHEL8 core crypto components 를 참조하십시오. 예를 들어 사용자가 OpenShift Container Platform 클러스터 및 컨테이너에 SSH를 실행하면 해당 연결이 올바르게 암호화됩니다.

OpenShift Container Platform 구성 요소는 Go로 작성된 Red Hat의 golang 컴파일러를 사용하여 빌드됩니다. 클러스터의 FIPS 모드를 활성화하면 암호화 서명이 필요한 모든 OpenShift Container Platform 구성 요소에 대해 RHEL 및 RHCOS 암호화 라이브러리를 호출합니다.

표 23.1. OpenShift Container Platform 4.9의 FIPS 모드 속성 및 제한 사항

속성	제한
RHEL 7, RHEL 8 및 RHCOS 운영 체제에서 FIPS 지원	FIPS 구현은 해시 함수를 계산하고 해당 해시 기반 키를 검증하는 단일 함수를 제공하지 않습니다. 이 제한은 향후 OpenShift Container Platform 릴리스에서 지속적으로 평가 및 개선될 예정입니다.
CRI-O 런타임에서 FIPS 지원	
OpenShift Container Platform 서비스에서 FIPS 지원	
RHEL 7, RHEL 8 및 RHCOS 바이너리 및 이미지에서 얻은 FIPS 검증 / 진행중인 모듈 (Modules in Process) 암호화 모듈 및 알고리즘	
FIPS 호환 golang 컴파일러 사용	TLS FIPS 지원은 완전히 구현되어 있지 않지만 향후 OpenShift Container Platform 버전에서 완전히 지원될 예정입니다.

속성	제한
여러 아키텍처에서 FIPS 지원	FIPS는 현재 **x86_64** 아키텍처를 사용하는 OpenShift Container Platform 배포에서만 지원됩니다.

23.2. 클러스터가 사용되는 구성 요소에서 FIPS 지원

OpenShift Container Platform 클러스터 자체는 **FIPS** 검증 / 진행중인 모듈 **(Modules in Process)** 모듈을 사용하지만 **OpenShift Container Platform** 클러스터를 지원하는 시스템이 암호화에 **FIPS** 검증 / 진행중인 모듈 **(Modules in Process)** 모듈을 사용하고 있는지 확인하십시오.

23.2.1. etcd

etcd에 저장된 비밀 정보가 **FIPS** 검증 / 진행중인 모듈 **(Modules in Process)** 암호화를 사용하는지 확인하려면 **FIPS** 모드에서 노드를 부팅합니다. **FIPS** 모드에서 클러스터를 설치한 후 **FIPS** 승인 **aes cbc** 암호화 알고리즘을 사용하여 **etcd** 데이터를 암호화할 수 있습니다.

23.2.2. 스토리지

로컬 스토리지의 경우 **RHEL** 제공 디스크 암호화 또는 **RHEL** 제공 디스크 암호화를 사용하는 **Container Native Storage**를 사용합니다. **RHEL** 제공 디스크 암호화를 사용하는 볼륨에 모든 데이터를 저장하고 클러스터에 **FIPS** 모드를 활성화하면 미사용 데이터와 이동중인 데이터 또는 네트워크 데이터가 모두 **FIPS** 검증 / 진행중인 모듈 **(Modules in Process)** 암호화를 통해 보호됩니다. 노드의 사용자 정의에 설명된 대로 각 노드의 **root** 파일 시스템을 암호화하도록 클러스터를 설정할 수 있습니다.

23.2.3. 런타임

컨테이너가 **FIPS** 검증 / 진행중인 모듈 **(Modules in Process)** 암호화 모듈을 사용하는 호스트에서 실행되고 있음을 확인하려면 **CRI-O**를 사용하여 런타임을 관리합니다. **CRI-O**는 컨테이너가 **FIPS** 모드에서 실행되고 있음을 알 수 있도록 컨테이너를 설정하는 **FIPS-Mode**를 지원합니다.

23.3. FIPS 모드에서 클러스터 설치

FIPS 모드에서 클러스터를 설치하려면 해당 인프라에 사용자 정의된 클러스터를 설치하는 방법에 대한 프로세스를 따르십시오. 클러스터를 배포하기 전에 **install-config.yaml** 파일에 **fips: true**가 설정되어 있는지 확인하십시오.

- Amazon Web Services
- Microsoft Azure
- 베어 메탈
- Google Cloud Platform
- Red Hat OpenStack Platform (RHOSP)
- VMware vSphere

참고

Azure File 스토리지를 사용하는 경우 FIPS 모드를 활성화할 수 없습니다.

etcd 데이터 저장소에 AES CBC 암호화를 적용하려면 클러스터를 설치한 후 etcd 데이터 암호화 프로세스를 수행하십시오.

RHEL 노드를 클러스터에 추가하는 경우 초기 부팅 전에 머신에서 FIPS 모드를 활성화해야 합니다. RHEL 7 설명서에서 RHEL 컴퓨팅 머신 추가 및 OpenShift Container Platform 클러스터에 FIPS 모드 활성화 를 참조하거나 RHEL 8 설명서에서 FIPS 모드 활성화 를 참조하십시오.