OpenShift Container Platform 4.5

설치

OpenShift Container Platform 클러스터 설치 및 구성

Last Updated: 2020-10-02

OpenShift Container Platform 4.5 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. 설치 문제 해결

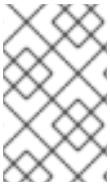
OpenShift Container Platform 설치 실패 문제를 해결하기 위해 부트 스트랩 및 컨트롤 플레인 또는 마스터 시스템에서 로그를 수집할 수 있습니다. 설치 프로그램에서 디버그 정보를 얻을 수도 있습니다.

전제 조건

- OpenShift Container Platform 클러스터를 설치하려고했으나 설치에 실패했습니다.

1.1. 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.

전제 조건

- 부트 스트랩 프로세스가 완료되기 전에 OpenShift Container Platform 설치에 실패합니다. 부트 스트랩 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 컨트롤 플레인 또는 마스터 노드의 정규화된 도메인 이름이 있어야합니다.

프로세스

1. 부트 스트랩 및 컨트롤 플레인 시스템에서 설치 로그를 가져오는데 필요한 명령을 생성합니다.
   - 설치 프로그램에서 제공하는 인프라를 사용한 경우 다음 명령을 실행합니다.

     ```
     $ ./openshift-install gather bootstrap --dir=<installation_directory> 1
     ```

     1 **installation_directory**는 **./openshift-install create cluster**를 실행할 때 지정한 디렉토리입니다. 이 디렉토리에는 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.

   - 프로비저닝한 인프라를 사용하는 경우 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_directory**의 경우 **./openshift-install create cluster**를 실행할 때 지정한 것과 동일한 디렉토리를 지정합니다. 이 디렉토리에는 설치 프로그램이 생성한 OpenShift Container Platform 정의 파일이 포함되어 있습니다.

2. **<bootstrap_address>** 는 클러스터 부트 스트랩 시스템의 정규화된 도메인 이름 또는 IP 주소입니다.

3 4 5 클러스터의 각 컨트롤 플레인 또는 마스터 시스템에 대해 **<master _ * _ address>**를 정규화된 도메인 이름 또는 IP 주소로 변경합니다.

참고

기본 클러스터에는 세 개의 컨트롤 플레인 시스템이 있습니다. 클러스터가 사용하는 수에 관계없이 표시된대로 모든 컨트롤 플레인 시스템을 나열합니다.

명령 출력은 다음 예제와 유사합니다.

```
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.

1.2. 호스트에 SSH 액세스를 통해 수동으로 로그 수집

must-gather 또는 자동화된 수집 방법이 작동하지 않는 경우 로그를 수동으로 수집합니다.

전제 조건

- 호스트에 대한 SSH 액세스 권한이 있어야합니다.

프로세스

1. 다음을 실행하여 **journalctl** 명령을 사용하여 부트 스트랩 호스트에서 **bootkube.service** 서비스 로그를 수집합니다.

```
$ journalctl -b -f -u bootkube.service
```

2. Podman 로그를 사용하여 부트 스트랩 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/*
```

1.3. 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.

전제 조건

- OpenShift Container Platform 설치가 완료되어야 합니다.
- API 서비스가 작동하고 있어여 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
```

1.4. 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_directory**의 경우 **./openshift-install create cluster**를 실행할 때 지정한 것과 동일한 디렉토리를 지정합니다.

- **--log-level = debug**를 사용하여 설치 프로그램을 다시 실행합니다.

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

1 **installation_directory**의 경우 **./openshift-install create cluster**를 실행할 때 지정한 것과 동일한 디렉토리를 지정합니다.

2장. FIPS 암호화 지원

버전 4.3부터는 FIPS 검증 / IUT (Implementation Under Test) 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 시스템에서는 작업자 시스템으로 사용하려는 시스템에 운영 체제를 설치할 때 FIPS 모드를 활성화해야합니다. 이러한 설정 방법을 사용하면 클러스터가 FIPS 준수 감사 요구 사항을 충족하는지 확인할 수 있습니다. 초기 시스템 부팅 전에 FIPS 검증/ IUT (Implementation Under Test) 암호화 패키지만 활성화됩니다.

클러스터가 사용하는 운영 체제를 처음 부팅하기 전에 FIPS를 활성화해야하므로 클러스터를 배포한 후 FIPS를 활성화할 수 없습니다.

2.1. OPENSHIFT CONTAINER PLATFORM에서 FIPS 검증

OpenShift Container Platform은 이를 사용하는 운영 체제 구성 요소에 대해 RHEL (Red Hat Enterprise Linux) 및 RHCOS에서 특정 FIPS 검증 / UT (Implementation Under Test) 모듈을 사용합니다. RHEL7 core crypto components의 내용을 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 Red Hat의 golang 컴파일러는 암호화 서명이 필요한 모든 OpenShift Container Platform 구성 요소에 대해 RHEL 및 RHCOS 암호화 라이브러리를 호출합니다. OpenShift Container Platform 버전 4.3의 초기 릴리스에서는 **ose-sdn** 패키지 만 FIPS 검증 / IUT (Implementation Under Test)가 아닌 기본 golang 암호화를 사용합니다. Red Hat은 다른 모든 패키지가 FIPS 검증 / IUT (Implementation Under Test) OpenSSL 모듈을 사용하는지 확인합니다.

표 2.1. OpenShift Container Platform 4.5의 FIPS 모드 속성 및 제한 사항

속성	제한
RHEL 7 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 및 RHCOS 바이너리 및 이미지에서 얻은 FIPS 검증 / IUT (Implementation Under Test) 암호화 모듈 및 알고리즘	
FIPS 호환 golang 컴파일러 사용	TLS FIPS 지원은 완전히 구현되어 있지 않지만 향후 OpenShift Container Platform 버전에서 완전히 지원될 예정입니다.

2.2. 클러스터가 사용되는 구성 요소에서 FIPS 지원

OpenShift Container Platform 클러스터 자체는 FIPS 검증 / IUT (Implementation Under Test) 모듈을 사용하지만 OpenShift Container Platform 클러스터를 지원하는 시스템이 암호화에 FIPS 검증 / IUT (Implementation Under Test) 모듈을 사용하고 있는지 확인하십시오.

2.2.1. etcd

etcd에 저장된 암호가 FIPS 검증 / IUT (Implementation Under Test) 암호화를 사용하도록하려면 FIPS 승인 암호화 알고리즘을 사용하여 etcd 데이터 저장소를 암호화합니다. 클러스터를 설치한 후 **aes cbc** 알고리즘을 사용하여 etcd 데이터를 암호화할 수 있습니다.

2.2.2. 스토리지

로컬 스토리지의 경우 RHEL 제공 디스크 암호화 또는 RHEL 제공 디스크 암호화를 사용하는 Container Native Storage를 사용합니다. RHEL 제공 디스크 암호화를 사용하는 볼륨에 모든 데이터를 저장하고 클러스터에 FIPS 모드를 활성화하면 미사용 데이터와 이동중인 데이터 또는 네트워크 데이터가 모두 FIPS 검증 / IUT (Implementation Under Test) 암호화를 통해 보호됩니다. 노드의 사용자 정의에 설명된 대로 각 노드의 루트 파일 시스템을 암호화하도록 클러스터를 설정할 수 있습니다.

2.2.3. 런타임

컨테이너가 FIPS 검증 / IUT (Implementation Under Test) 암호화 모듈을 사용하는 호스트에서 실행되고 있음을 확인하려면 CRI-O를 사용하여 런타임을 관리합니다. CRI-O는 컨테이너가 FIPS 모드에서 실행되고 있음을 알 수 있도록 컨테이너를 설정하는 FIPS-Mode를 지원합니다.

2.3. 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
- Bare metal
- Google Cloud Platform
- Red Hat OpenStack Platform (RHOSP)
- VMware vSphere

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 활성화해야합니다. RHEL 7 설명서에서Adding RHEL compute machines to an OpenShift Container Platform cluster 및 Enabling FIPS Mode를 참조하십시오.

3장. 설치 구성

3.1. 다른 플랫폼에서의 설치 방법

다른 플랫폼에서 다른 유형의 설치를 수행할 수 있습니다.

참고

다음 표에 표시된 대로 현재 모든 플랫폼에서 모든 설치 옵션을 사용할 수 있는 것은 아닙니다.

표 3.1. 설치 프로그램에서 제공하는 인프라 옵션

	AWS	Azure	GCP	OpenStack	RHV	Bare metal	vSphere	IBM Z
기본	X	X	X		X			
사용자 정의	X	X	X	X	X			
Network Operator	X	X	X					
프라이빗 클러스터	X	X	X					
기존 가상 사설망	X	X	X					

표 3.2. 사용자가 제공하는 인프라 옵션

	AWS	Azure	GCP	OpenStack	RHV	Bare metal	vSphere	IBM Z
사용자 정의	X	X	X	X		X	X	
Network Operator						X	X	
제한된 네트워크	X		X			X	X	

3.2. 노드의 사용자 정의

OpenShift Container Platform 노드를 직접 변경하는 것은 권장되지 않지만 필요에 따라 낮은 수준의 보안, 네트워킹 또는 성능 기능을 구현해야하는 경우가 있습니다. OpenShift Container Platform 노드를 직접 변경하려면 다음을 수행하십시오.

- **Openshift-install** 중 클러스터를 시작하기 위해 매니페스트 파일에 포함된 MachineConfig를 생성합니다.
- Machine Config Operator를 통해 OpenShift Container Platform 노드에 전달되는 MachineConfig를 생성합니다.

다음 섹션에서는 이러한 방식으로 노드에서 설정할 수 있는 기능에 대해 설명합니다.

3.2.1. 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.

- SELinux와 같은 기능을 비활성화하여 이 기능이 시스템에 영향을 미치지 않도록할 필요가 있는 경우
- 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.

프로세스

1. 클러스터에 대한 Kubernetes 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir=<installation_directory>
```

2. 커널 매개 변수를 작업자 또는 마스터 노드에 추가할지 여부를 결정합니다.
3. **openshift** 디렉토리에서 파일 (예: **99-openshift-machineconfig-master-kargs.yaml**)을 작성하여 커널 설정을 추가할 MachineConfig 객체를 정의합니다. 다음 예제에서는 **loglevel=7** 커널 매개 변수를 마스터 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.

3.2.2. 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 Linux에서 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 Hat에서 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 액세스해야합니다. 해당 컨텐츠는 유효한 RHEL 서브스크립션을 통해 효과적으로 사용할 수 있습니다.

3.2.2.1. 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 가져온 후 등록하고 구독하십시오.

```
# subscription-manager register
Username: yourname
Password: ***************
# subscription-manager attach --auto
```

2. 소프트웨어 및 컨테이너를 빌드하는 데 필요한 소프트웨어를 설치하십시오.

```
# yum install podman make git -y
```

3. kmod-via-containers 리포지토리를 복제합니다.

```
$ mkdir kmods; cd kmods
$ git clone https://github.com/kmods-via-containers/kmods-via-containers
```

4. RHEL 8 빌드 호스트에 KVC 프레임 워크 인스턴스를 설치하여 모듈을 테스트합니다. kmods-via-container systemd 서비스가 추가되어 로드됩니다.

```
$ cd kmods-via-containers/
$ sudo make install
$ sudo systemctl daemon-reload
```

5. 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
```

6. 다음과 같이 설정 파일 **simple-kmod.conf**을 편집하고 Dockerfile의 이름을 **Dockerfile.rhel**로 변경합니다.

```
$ cd kvc-simple-kmod
$ cat simple-kmod.conf

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"
```

7. 커널 모듈의 **kmods-via-containers @ .service** 인스턴스 (이 예제에서는 **simple-kmod)**를 만들고 이를 활성화합니다.

```
$ sudo make install
$ sudo kmods-via-containers build simple-kmod $(uname -r)
```

8. systemd 서비스를 활성화하고 시작한 다음 상태를 확인하십시오.

```
$ sudo systemctl enable kmods-via-containers@simple-kmod.service
$ sudo systemctl start kmods-via-containers@simple-kmod.service
$ sudo systemctl status kmods-via-containers@simple-kmod.service
● 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...
```

9. 커널 모듈이 로드되었는지 확인하려면 **lsmod** 명령을 사용하여 모듈을 나열하십시오.

```
$ lsmod | grep simple_
simple_procfs_kmod     16384  0
simple_kmod            16384  0
```

10. simple-kmod 예제에는 작동 여부를 테스트하는 몇 가지 다른 방법이 있습니다. **dmesg**를 사용하여 커널 링 버퍼에 "Hello world"메시지를 찾으십시오.

```
$ dmesg | grep 'Hello world'
[ 6420.761332] Hello world from simple_kmod.
```

/proc에서 **simple-procfs-kmod**의 값을 확인합니다.

```
$ sudo cat /proc/simple-procfs-kmod
simple-procfs-kmod number = 0
```

spkut 명령을 실행하여 모듈에 대한 자세한 정보를 가져옵니다.

```
$ sudo spkut 44
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.

3.2.2.2. 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)** MachineConfig를 통해 컨텐츠를 작성하고 매니페스트 파일 세트와 함께 **openshift-install**에 제공할 수 있습니다.
- **Machine Config Operator를 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 컨텐츠를 가져올 수 있도록 각 노드를 설정할 수있는 몇 가지 방법이 있습니다.

- 각 노드에 RHEL 인타이틀먼트를 제공합니다.
- **/ etc / pki / entitlement** 디렉토리에서 기존 RHEL 호스트의 RHEL 인타이틀먼트를 취득하고 Ignition 설정을 빌드할 때 제공하는 다른 파일과 동일한 위치에 복사합니다.
- Dockerfile에서 커널 및 기타 패키지가 포함된 **yum** 저장소에 대한 포인터를 추가합니다. 여기에는 새로 설치된 커널과 일치해야하므로 새 커널 패키지가 포함되어 있어야합니다.

3.2.2.2.1. MachineConfig를 통한 커널 모듈 프로비저닝

MachineConfig로 커널 모듈 소프트웨어를 패키지하면 설치시 또는 Machine Config Operator를 통해 해당 소프트웨어를 작업자 또는 마스터 노드에 전달할 수 있습니다.

먼저 사용하려는 기본 Ignition 설정을 만듭니다 설치시 Ignition 설정에는 클러스터에서 **core** 사용자의 **authorized_keys** 파일에 추가할 ssh 공개 키가 포함됩니다. 나중에 MCO를 통해 MachineConfig를 추가하는 경우 ssh 공개키가 필요하지 않습니다. 두 가지 유형 모두 샘플 simple-kmod 서비스는 **kmds-via-containers@simple-kmod.service**가 필요한 systemd 장치 파일을 만듭니다.

참고

systemd 장치는 업스트림 버그에 대한 해결 방법이며 **kmods-via-containers@simple-kmod.service**가 부팅시 시작되도록합니다.

1. RHEL 8 시스템을 가져온 후 등록합니다.

```
# subscription-manager register
Username: yourname
Password: ***************
# subscription-manager attach --auto
```

2. 소프트웨어를 빌드하는 데 필요한 소프트웨어를 설치합니다.

```
# yum install podman make git -y
```

3. systemd 장치 파일을 만들 Ignition 설정 파일을 만듭니다.

```
$ mkdir kmods; cd kmods
$ cat <<EOF > ./baseconfig.ign
{
  "ignition": { "version": "2.2.0" },
  "passwd": {
    "users": [
      {
        "name": "core",
        "groups": ["sudo"],
        "sshAuthorizedKeys": [
          "ssh-rsa AAAA"
        ]
      }
    ]
  },
  "systemd": {
    "units": [{
      "name": "require-kvc-simple-kmod.service",
      "enabled": true,
      "contents": "[Unit]\nRequires=kmods-via-containers@simple-
kmod.service\n[Service]\nType=oneshot\nExecStart=/usr/bin/true\n\n[Install]\nWantedBy=multi-
user.target"
    }]
  }
}
EOF
```

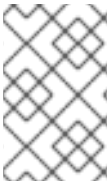
참고

openshift-install 중에 파일을 사용하려면 공개 SSH 키를 **baseconfig.ign** 파일에 추가해야합니다. MCO를 통해 MachineConfig를 생성하는 경우 공개 SSH 키가 필요하지 않습니다.

4. 다음과 같은 기본 MCO YAML 스니펫을 만듭니다.

```
$ cat <<EOF > mc-base.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 10-kvc-simple-kmod
spec:
  config:
EOF
```

+



참고

mc-base.yaml은 **worker** 노드에 커널 모듈을 배포하도록 설정되어 있습니다. 마스터 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.

1. **kmods-via-containers** 소프트웨어를 가져옵니다:

```
$ git clone https://github.com/kmods-via-containers/kmods-via-containers
$ git clone https://github.com/kmods-via-containers/kvc-simple-kmod
```

2. 모듈 소프트웨어를 가져옵니다. 이 예에서는 **kvc-simple-kmod**가 사용됩니다.

3. 이전에 복제된 리포지토리를 사용하여 fakeroot 디렉토리를 만들고 Ignition을 통해 전달할 파일을 이 디렉토리에 배치합니다.

```
$ FAKEROOT=$(mktemp -d)
$ cd kmods-via-containers
$ make install DESTDIR=${FAKEROOT}/usr/local CONFDIR=${FAKEROOT}/etc/
$ cd ../kvc-simple-kmod
$ make install DESTDIR=${FAKEROOT}/usr/local CONFDIR=${FAKEROOT}/etc/
```

4. **filetranspiler**라는 툴 및 종속 소프트웨어를 가져옵니다.

```
$ cd ..
$ sudo yum install -y python3
git clone https://github.com/ashcrow/filetranspiler.git
```

5. 최종 MachineConfig YAML (**mc.yaml**)을 생성하고 이를 전달하려는 파일과 함께 기본 Ignition 설정, 기본 MachineConfig 및 fakeroot 디렉토리를 포함합니다.

```
$ ./filetranspiler/filetranspile -i ./baseconfig.ign \
    -f ${FAKEROOT} --format=yaml --dereference-symlinks \
    | sed 's/^/     /' | (cat mc-base.yaml -) > 99-simple-kmod.yaml
```

6. 클러스터가 아직 작동하지 않은 경우 매니페스트 파일을 생성하고 해당 파일을 **openshift** 디렉토리에 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.

7. 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_
simple_procfs_kmod     16384  0
simple_kmod            16384  0
```

3.2.3. 설치시 디스크 암호화

OpenShift Container Platform을 설치할 때 모든 마스터 및 작업자 노드에서 디스크 암호화를 활성화할 수 있습니다. 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
- 루트 파일 시스템의 데이터만 암호화함 (/에서 / **dev** / **mapper** / **coreos-luks-root**)
- 사용자 개입없이 암호 제공 가능
- AES-256-CBC 암호화 사용
- 클러스터가 FIPS를 지원하도록 활성화해야 함

다음의 두 가지 암호화 모드가 지원됩니다.

- TPM v2: 기본 모드입니다. TPM v2는 암호를 보안 암호화 프로세서에 저장합니다. TPM v2 디스크 암호화를 구현하려면 아래 설명에 따라 Ignition 설정 파일을 만듭니다.
- Tang : Tang을 사용하여 클러스터를 암호화하려면 Tang 서버를 사용해야합니다. Clevis는 클라이언트 측에서 암호 해독을 구현합니다. Tang 암호화 모드는 베어 메탈 설치에만 지원됩니다.

클러스터 노드에 디스크 암호화를 사용하려면 다음 두 프로세스 중 하나를 수행하십시오.

3.2.3.1. TPM v2 디스크 암호화 활성화

다음 프로세스를 사용하여 OpenShift Container Platform 배포 중에 TPM v2 모드 디스크 암호화를 활성화합니다.

프로세스

1. 각 노드의 BIOS에서 TPM v2 암호화를 활성화해야하는지 확인하십시오. 이는 대부분의 Dell 시스템에서 필요합니다. 컴퓨터 설명서를 확인하십시오.

2. 클러스터에 대한 Kubernetes 매니페스트를 생성합니다.

```
$ ./openshift-install create manifests --dir=<installation_directory>
```

3. **openshift** 디렉토리에서 마스터 및/또는 작업자 파일을 작성하여 해당 노드의 디스크를 암호화하십시오. 이 두 파일의 예는 다음과 같습니다.

```
$ cat << EOF > ./99-openshift-worker-tpmv2-encryption.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: worker-tpm
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 2.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;base64,e30K
        filesystem: root
        mode: 420
        path: /etc/clevis.json
EOF
```

```
$ cat << EOF > ./99-openshift-master-tpmv2-encryption.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: master-tpm
  labels:
    machineconfiguration.openshift.io/role: master
spec:
  config:
    ignition:
      version: 2.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;base64,e30K
        filesystem: root
        mode: 420
        path: /etc/clevis.json
EOF
```

4. YAML 파일의 백업 사본을 만듭니다. 클러스터를 만들 때 파일이 삭제되므로 백업을 수행해야합니다.

5. 나머지 OpenShift Container Platform 배포를 계속합니다.

3.2.3.2. Tang 디스크 암호화 활성화

다음 프로세스를 사용하여 OpenShift Container Platform 배포 중에 Tang 모드 디스크 암호화를 활성화합니다.

프로세스

1. 암호화 설정을 구성하고 **openshift-install**을 실행하여 함께 작업할 클러스터 및 **oc**를 설치하기 위해 Red Hat Enterprise Linux 서버에 액세스합니다.

2. 기존 Tang 서버를 설정하거나 액세스합니다. 자세한 내용은 Network-bound disk encryption 에서 참조하십시오. Tang에 표시하는 방법에 대한 자세한 내용은 Securing Automated Decryption New Cryptography and Techniques에서 참조하십시오.

3. 클러스터에 대해 RHCOS 설치를 수행할 때 네트워크를 설정하기 위해 커널 인수를 추가합니다. 이 단계를 생략하면 두 번째 부팅이 실패합니다. 예를 들어 DHCP 네트워킹을 설정하려면 **ip=dhcp**를 지정하거나 커널 명령 줄에 매개 변수를 추가할 때 정적 네트워크를 설정합니다.

4. 지문을 생성합니다. clevis 패키지를 설치하십시오 (아직 설치되지 않은 경우). Tang 서버에서 지문을 생성합니다. **url** 값을 Tang 서버 URL로 바꾸십시오.

```
$ sudo yum install clevis -y
$ echo nifty random wordwords \
    | clevis-encrypt-tang \
      '{"url":"https://tang.example.org"}'

The advertisement contains the following signing keys:

PLjNyRdGw03zlRoGjQYMahSZGu9

Do you wish to trust these keys? [ynYN] y
eyJhbmc3SlRyMXpPenc3ajhEQ01tZVJiTi1oM...
```

5. Base64로 인코딩된 파일을 만들고 값을 새로 생성된 Tang 서버 (**url**) URL 및 지문 (**thp**)으로 바꿉니다.

```
$ (cat <<EOM
{
 "url": "https://tang.example.com",
 "thp": "PLjNyRdGw03zlRoGjQYMahSZGu9"
}
EOM
) | base64 -w0

ewogInVybCI6ICJodHRwczovL3RhbmcuZXhhbXBsZS5jb20iLAogInRocCI6ICJaUk1leTFjR3cwN3psVExHYlhuUWFoUzBHdTAiCn0K
```

6. TPM2 예제의 "source"를 작업자 및/또는 마스터 노드의 다음 예제 중 하나 또는 둘 다에 대한 Base64 인코딩 파일로 바꿉니다.

```
$ cat << EOF > ./99-openshift-worker-tang-encryption.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: worker-tang
```

```
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 2.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;base64,e30K
          source:
data:text/plain;base64,ewogInVybCI6ICJodHRwczovL3RhbmcuZXhhbXBsZS5jb20iLAogInRoc
CI6ICJaUk1leTFjR3cwN3psVExxHYlhuUWFoUzBHdTAiCn0K
        filesystem: root
        mode: 420
        path: /etc/clevis.json
EOF
```

```
$ cat << EOF > ./99-openshift-master-encryption.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: master-tang
  labels:
    machineconfiguration.openshift.io/role: master
spec:
  config:
    ignition:
      version: 2.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;base64,e30K
          source:
data:text/plain;base64,ewogInVybCI6ICJodHRwczovL3RhbmcuZXhhbXBsZS5jb20iLAogInRoc
CI6ICJaUk1leTFjR3cwN3psVExxHYlhuUWFoUzBHdTAiCn0K
        filesystem: root
        mode: 420
        path: /etc/clevis.json
EOF
```

7. 나머지 OpenShift Container Platform 배포를 계속합니다.

3.2.4. chrony 타임 서비스 설정

chrony.conf 파일의 내용을 수정하고 해당 내용을 MachineConfig로 노드에 전달하여 chrony 타임 서비스 (chronyd)에서 사용하는 시간 서버 및 관련 구성을 설정할 수 있습니다.

프로세스

1. **chrony.conf** 파일의 내용을 작성하고 base64로 인코딩하십시오. 예를 들면 다음과 같습니다.

```
$ cat << EOF | base64
    server clock.redhat.com iburst
    driftfile /var/lib/chrony/drift
```

```
    makestep 1.0 3
    rtcsync
    logdir /var/log/chrony
    EOF

ICAgIHNlcnZlciBjbG9jay5yZWRoYXQuY29tIGlidXJzdAogICAgZHJpZnRmaWxlIC92YXIvbGli
L2Nocm9ueS9kcmlmdAogICAgbWFrZXN0ZXAgMS4wIDMKICAgIHJ0Y3N5bmMKICAgIGxvZ2Rpciav
dmFyL2xvZy9jaHJvbnkK
```

2. base64 문자열을 작성한 문자열로 바꾸어 MachineConfig 파일을 만듭니다. 이 예제에서는 파일을 **master** 노드에 추가합니다. 이를 **worker**로 변경하거나 **worker** 역할의 추가 MachineConfig을 만들 수 있습니다.

```
$ cat << EOF > ./99-masters-chrony-configuration.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: master
  name: masters-chrony-configuration
spec:
  config:
    ignition:
      config: {}
      security:
        tls: {}
      timeouts: {}
      version: 2.2.0
    networkd: {}
    passwd: {}
    storage:
      files:
      - contents:
          source: data:text/plain;charset=utf-
8;base64,c2VydmVyIGNsb2NrLnJlZGhhdC5jb20gaWJ1cnN0CmRyaWZ0ZmlsZSAvdmFyL2xp
Yi9jaHJvbnkvZHJpZnQKbWFrZXN0ZXAgMS4wIDMKcnRjc3luYwpsb2dkaXIgL3Zhci9sb2cvY2h
yb255Cg==
          verification: {}
        filesystem: root
        mode: 420
        path: /etc/chrony.conf
  osImageURL: ""
EOF
```

3. 설정 파일의 백업 사본을 만듭니다.

4. 클러스터가 아직 시작하지 않은 경우 매니페스트 파일을 생성하고 해당 파일을 **openshift** 디렉토리에 추가한 후 계속 클러스터를 작성합니다.

5. 클러스터가 이미 실행중인 경우 다음과 같이 파일을 적용합니다.

```
$ oc apply -f ./masters-chrony-configuration.yaml
```

3.2.5. 추가 리소스

FIPS 지원에 대한 자세한 내용은 FIPS 암호화 지원을 참조하십시오.

3.3. 제한된 네트워크가 제한된 환경에 설치하기 위해 미러 레지스트리 만들기

제한된 네트워크에서 프로비저닝된 인프라에 클러스터를 설치하기 전에 필요한 컨테이너 이미지를 해당 환경에 미러링해야합니다. 네트워크가 제한된 환경에서의 설치는 설치 프로그램에서 제공하는 인프라가 아닌 사용자가 제공하는 인프라에서만 지원됩니다. 이 프로세스를 무제한 네트워크에서 사용하여 클러스터가 외부 컨텐츠에 대해 조직의 제어 조건을 충족하는 컨테이너 이미지만 사용하도록 할 수 있습니다.

중요

필요한 컨테이너 이미지를 얻으려면 인터넷에 액세스해야합니다. 이 프로세스에서는 네트워크와 인터넷에 모두 액세스할 수 있는 미러 호스트에 미러 레지스트리를 배치합니다. 미러 호스트에 액세스할 수없는 경우 연결 해제된 프로세스를 사용하여 네트워크 경계를 이동할 수 있는 장치에 이미지를 복사합니다.

3.3.1. 미러 레지스트리 정보

OpenShift Container Platform 설치 및 후속 제품 업데이트에 필요한 이미지를 미러 레지스트리에 미러링할 수 있습니다. 이러한 작업은 동일한 프로세스를 사용합니다. 컨텐츠에 대한 설명이 포함된 릴리스 이미지와 이 컨텐츠가 참조하는 이미지가 모두 미러링됩니다. 또한 Operator 카탈로그 소스 이미지 및 참조하는 이미지는 사용하는 Operator마다 미러링해야합니다. 컨텐츠를 미러링한 후 미러 레지스트리에서 이 컨텐츠를 검색하도록 각 클러스터를 설정합니다.

미러 레지스트리에는 최신 컨테이너 이미지 API (**schema2**라고 함)를 지원하는 모든 컨테이너 레지스트리를 사용할 수 있습니다. 모든 주요 클라우드 공급자 레지스트리, Red Hat Quay, Artifactory, 오픈 소스 Docker 배포 레지스트리가 지원됩니다. 이러한 레지스트리 중 하나를 사용하면 OpenShift Container Platform이 연결되지 않은 환경에서 각 이미지의 무결성을 확인할 수 있습니다.

프로비저닝한 클러스터의 모든 시스템에서 미러 레지스트리에 액세스할 수 있어야합니다. 레지스트리에 연결할 수 없는 경우 설치, 업데이트 또는 워크로드 재배치와 같은 일반적인 작업이 실패할 수 있습니다. 따라서 고가용성 방식으로 미러 레지스트리를 실행해야하며 미러 레지스트리는 최소한 OpenShift Container Platform 클러스터의 프로덕션 환경의 가용성조건에 일치해야합니다.

미러 레지스트리를 OpenShift Container Platform 이미지로 설정하는 경우 다음 두 가지 시나리오를 수행할 수 있습니다. 호스트가 인터넷과 미러 레지스트리에 모두 액세스할 수 있지만 클러스터 노드에 액세스할 수 없는 경우 해당 머신의 컨텐츠를 직접 미러링할 수 있습니다. 이 프로세스를 *connected mirroring* (미러링 연결)이라고합니다. 그러한 호스트가 없는 경우 이미지를 파일 시스템에 미러링한 다음 해당 호스트 또는 이동식 미디어를 제한된 환경에 배치해야합니다. 이 프로세스를 *disconnected mirroring* (미러링 연결 해재)라고합니다.

3.3.2. 미러 호스트 준비

미러 단계를 수행하기 전에 호스트를 준비하여 컨텐츠를 검색하고 원격 위치로 푸시할 수 있도록 해야합니다.

3.3.2.1. 바이너리를 다운로드하여 CLI 설치

명령 줄 인터페이스에서 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 4.5의 모든 명령을 완료하는 데 해당 버전을 사용할 수 없습니다. 새 버전의 **oc**를 다운로드하여 설치해야 합니다.

3.3.2.1.1. Linux에서 CLI 설치

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

프로세스

1. Red Hat OpenShift Cluster Manager 사이트의 Infrastructure Provider 페이지로 이동합니다.
2. 인프라 공급 업체를 선택하고 설치 유형을 선택하십시오.
3. **Command-line interface** 섹션의 드롭다운 메뉴에서 **Linux** 를 선택하고 **Download command-line tools**를 클릭합니다.
4. 아카이브의 압축을 풉니다.

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

CLI를 설치한 후 **oc** 명령으로 통해 사용할 수 있습니다.

```
$ oc <command>
```

3.3.2.1.2. Windows에서 CLI 설치

다음 프로세스를 사용하여 Windows에 OpenShift CLI (**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat OpenShift Cluster Manager 사이트의 Infrastructure Provider 페이지로 이동합니다.
2. 인프라 공급 업체를 선택하고 설치 유형을 선택하십시오.
3. **Command-line interface** 섹션의 드롭다운 메뉴에서 **Windows**를 선택하고 **Download command-line tools**를 클릭합니다.
4. ZIP 프로그램으로 아카이브를 압축 해제합니다.
5. **oc** 바이너리를 **PATH**에 있는 디렉토리로 이동합니다.
PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

CLI를 설치한 후 **oc** 명령으로 통해 사용할 수 있습니다.

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

3.3.2.1.3. macOS에 CLI 설치

다음 프로세스에 따라 macOS에서 OpenShift CLI (**oc**) 바이너리를 설치할 수 있습니다.

프로세스

1. Red Hat OpenShift Cluster Manager 사이트의 Infrastructure Provider 페이지로 이동합니다.
2. 인프라 공급 업체를 선택하고 설치 유형을 선택하십시오.
3. **Command-line interface** 섹션의 드롭다운 메뉴에서 **MacOS**를 선택하고 **Download command-line tools**를 클릭합니다.
4. 아카이브의 압축을 해제합니다.
5. **oc** 바이너리 PATH의 디렉토리로 이동합니다.
   PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

   ```
   $ echo $PATH
   ```

CLI를 설치한 후 **oc** 명령으로 통해 사용할 수 있습니다.

```
$ oc <command>
```

3.3.3. 이미지를 미러링할 수 있는 인증 정보 설정

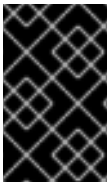
Red Hat에서 미러로 이미지를 미러링할 수 있는 컨테이너 이미지 레지스트리 인증 정보 파일을 생성합니다.

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 추가해야합니다.



중요

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.

전제 조건

- 네트워크가 제한된 환경에서 사용하도록 미러 레지스트리가 설정되어 있습니다.
- 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 Manager 사이트의 Pull Secret 페이지에서 **registry.redhat.io** pull secret을 다운로드하여 **.json** 파일에 저장하십시오.
2. 다음 명령을 사용하여 레지스트리에 로그인하십시오.

```
$ oc registry login --to ./pull-secret.json --registry "<registry_host_and_port>"
```

   레지스트리의 사용자 이름 및 비밀번호를 입력합니다.

3.3.4. OpenShift Container Platform 이미지 저장소 미러링

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

전제 조건

- 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 Manager 사이트의 Pull Secret 페이지에서 pull secret을 다운로드하여 미러 저장소의 인증 정보를 포함하도록 수정했습니다.

프로세스

미러 호스트에서 다음 단계를 완료하십시오.

1. OpenShift Container Platform 다운로드 페이지를 확인하여 설치할 OpenShift Container Platform 버전을 확인하고 Repository Tags 페이지에서 해당 태그를 지정합니다.
2. 필요한 환경 변수를 설정합니다.

```
$ export OCP_RELEASE=<release_version> 1
$ export LOCAL_REGISTRY='<local_registry_host_name>:<local_registry_host_port>' 2
$ export LOCAL_REPOSITORY='<local_repository_name>' 3
$ export PRODUCT_REPO='openshift-release-dev' 4
$ export LOCAL_SECRET_JSON='<path_to_pull_secret>' 5
```

```
$ export RELEASE_NAME="ocp-release" 6
$ export ARCHITECTURE=<server_architecture> 7
$ REMOVABLE_MEDIA_PATH=<path> 8
```

1. **<release_version>**에 대해 설치할 OpenShift Container Platform 버전에 해당하는 태그를 지정합니다 (예: **4.5.0**).
2. **<local_registry_host_name>**의 경우 미러 저장소의 레지스트리 도메인 이름을 지정하고 **<local_registry_host_port>**의 경우 컨텐츠를 제공하는데 사용되는 포트를 지정합니다.
3. **<local_repository_name>**의 경우 레지스트리에 작성할 저장소 이름 (예: **ocp4** / **openshift4**)을 지정합니다.
4. 미러링할 저장소입니다. 프로덕션 환경의 릴리스에 대해 **openshift-release-dev**를 지정해야합니다.
5. **<path_to_pull_secret>**에 대해 생성한 미러 레지스트리에 대한 풀 시크릿의 절대 경로 및 파일 이름을 지정합니다.
6. 릴리스 미러입니다. 프로덕션 환경의 릴리스에 대해 **ocp-release**를 지정해야합니다.
7. **server_architecture**에 대해 **x86_64**와 같은 서버 아키텍처를 지정합니다.
8. **<path>**에 대해 미러링된 이미지를 호스트할 디렉토리의 경로를 지정합니다.

3. 버전 이미지를 내부 컨테이너 레지스트리에 미러링합니다.
   - 미러 호스트가 인터넷에 액세스할 수 없는 경우 다음 작업을 수행하십시오.
     i. 이동식 미디어를 인터넷에 연결된 시스템에 연결합니다.
     ii. 미러링할 이미지 및 설정 매니페스트를 확인합니다.

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

     iii. 이전 명령의 출력에서 전체 **imageContentSources** 섹션을 기록합니다. 미러에 대한 정보는 미러링된 저장소에 고유하며 설치 중에 **imageContentSources** 섹션을 **install-config.yaml** 파일에 추가해야합니다.
     iv. 이동식 미디어의 디렉토리에 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}
```

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

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

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

  이 명령은 클러스터를 설치할 때 필요한 **imageContentSources** 데이터를 포함하여 릴리스 정보 요약을 출력합니다.

  ii. 이전 명령의 출력에서 **imageContentSources** 섹션 전체를 기록합니다. 미러에 대한 정보는 미러링된 저장소에 고유하며 설치 중에 **imageContentSources** 섹션을 **install-config.yaml** 파일에 추가해야합니다.

4. 미러링된 컨텐츠를 기반으로 설치 프로그램을 만들려면 컨텐츠를 추출하여 릴리스 배포에 고정합니다.

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

중요

선택한 OpenShift Container Platform 버전에 올바른 이미지를 사용하려면 미러링된 컨텐츠에서 설치 프로그램을 배포해야합니다.

인터넷이 연결된 컴퓨터에서이 단계를 수행해야합니다.

3.3.5. 대체 레지스트리 또는 미러링된 레지스트리에서 Samples Operator 이미지 스트림 사용

Samples Operator에 의해 관리되는 OpenShift namespace에 있는 대부분의 이미지 스트림은 registry.redhat.io의 Red Hat 레지스트리에 있는 이미지를 참조합니다. 미러링은 이러한 이미지 스트림에 적용되지 않습니다.

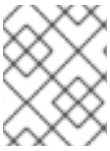
중요

jenkins, **jenkins-agent-maven** 및 **jenkins-agent-nodejs** 이미지스트림은 설치 페이로드에서 가져오고 Samples Operator로 관리되므로 해당 이미지 스트림에 대한 추가 미러링 절차가 필요하지 않습니다.

Sample Operator 설정 파일에서 **samplesRegistry** 필드를 registry.redhat.io로 설정하는 것은 Jenkins 이미지 및 이미지 스트림을 제외하고 registry.redhat.io로 전송되기 때문에 중복될 수 있습니다. 또한 Jenkins 이미지 스트림의 설치 페이로드도 손상됩니다.

Samples Operator는 Jenkins 이미지 스트림에 대해 다음 레지스트리를 사용하지 못하게 합니다.

- docker.io
- registry.redhat.io
- registry.access.redhat.com
- quay.io.



참고

설치 페이로드의 일부인 **cli**, **installer**, **must-gather** 및 **test** 이미지 스트림은 Samples Operator가 관리하지 않습니다. 이러한 내용은 이 프로세스에서 다루지 않습니다.

전제 조건

- **cluster-admin** 역할을 가진 사용자로 클러스터에 액세스합니다.
- 미러 레지스트리에 대한 풀 시크릿을 만듭니다.

프로세스

1. 미러링할 특정 이미지 스트림의 이미지에 액세스합니다. 예를 들면 다음과 같습니다.

```
$ oc get is <imagestream> -n openshift -o json | jq .spec.tags[].from.name | grep
registry.redhat.io
```

2. 제한된 네트워크 환경에서 필요한 모든 이미지 스트림과 관련된 registry.redhat.io의 이미지를 지정된 미러 중 하나로 미러링합니다.

```
$ oc image mirror registry.redhat.io/rhscl/ruby-25-rhel7:latest ${MIRROR_ADDR}/rhscl/ruby-
25-rhel7:latest
```

3. 클러스터의 이미지 설정 개체에서 미러에 필요한 신뢰할 수 있는 CA를 추가합니다.

```
$ oc create configmap registry-config --from-
file=${MIRROR_ADDR_HOSTNAME}..5000=$path/ca.crt -n openshift-config
$ oc patch image.config.openshift.io/cluster --patch '{"spec":{"additionalTrustedCA":
{"name":"registry-config"}}}' --type=merge
```

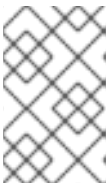
4. 미러 설정에 정의된 미러 위치의 **hostname** 부분을 포함하도록 Samples Operator 설정 개체에서 **samplesRegistry** 필드를 업데이트합니다.

```
$ oc get configs.samples.operator.openshift.io -n openshift-cluster-samples-operator
```

참고

현재 이미지 스트림 가져 오기 프로세스에서 미러 또는 검색 메커니즘이 사용되지 않기 때문에 이 작업이 필요합니다.

5. Samples Operator 설정 개체의 **skippedImagestreams** 필드에 미러링되지 않은 이미지 스트림을 추가합니다. 또는 샘플 이미지 스트림을 모두 지원할 필요가 없는 경우 Samples Operator 설정 개체에서 Samples Operator를 **Removed**로 설정합니다.



참고

건너 뛰지 않고 미러링되지 않은 이미지 스트림 또는 Samples Operator가 **Removed**로 변경되지 않는 경우 Samples Operator는 이미지 스트림 가져 오기를 실패한 후 2 시간 후에 **Degraded** 상태를 보고합니다.

OpenShift 네임 스페이스의 여러 템플릿은 이미지 스트림을 참조합니다. 따라서 **Removed**를 사용하여 이미지 스트림과 템플릿을 모두 제거하면 누락된 이미지 스트림으로 인해 기능이 제대로 작동하지 않을 경우 템플릿을 사용할 가능성이 없어집니다.

다음 단계

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

3.4. 사용 가능한 클러스터 사용자 정의

OpenShift Container Platform 클러스터를 배포한 후 대부분의 클러스터 설정 및 사용자 정의가 완료됩니다. 다양한 *설정 리소스*를 사용할 수 있습니다.

설정 리소스를 수정하여 이미지 레지스트리, 네트워킹 설정, 이미지 빌드 동작 및 아이덴티티 제공자와 같은 클러스터의 주요 기능을 설정합니다.

이러한 리소스를 사용하여 기능 제어를 설정하려면 **oc explain** 명령을 사용합니다. (예: **oc explain builds --api-version = config.openshift.io/v1)**

3.4.1. 클러스터 설정 리소스

모든 클러스터 설정 리소스는 전체적으로 범위가 지정되고 (네임 스페이스가 아님) **cluster**라는 이름을 지정할 수 있습니다.

리소스 이름	설명
apiserver.config.openshift.io	인증서 및 인증 기관과 같은 api-server 설정을 제공합니다.
authentication.config.openshift.io	클러스터의 아이덴티티 공급자 및 인증 설정을 제어합니다.

리소스 이름	설명
build.config.openshift.io	클러스터의 모든 빌드에 대한 기본 및 필수 구성되어 있는 설정을 제어합니다.
console.config.openshift.io	로그 아웃 동작을 포함하여 웹 콘솔 인터페이스의 동작을 설정합니다.
featuregate.config.openshift.io	기술 프리뷰 기능을 사용할 수 있도록 FeatureGates를 활성화합니다.
image.config.openshift.io	특정 이미지 레지스트리를 처리하는 방법 (allowed, disallowed, insecure, CA details)을 설정합니다.
ingress.config.openshift.io	라우팅의 기본 도메인과 같은 라우팅 관련 설정 상세 정보입니다.
oauth.config.openshift.io	내부 OAuth 서버 흐름에 대한 아이덴티티 제공자 및 다른 동작을 설정합니다.
project.config.openshift.io	프로젝트 템플릿을 포함하여 프로젝트를 만드는 방법을 설정합니다.
proxy.config.openshift.io	외부 네트워크 액세스를 필요로 하는 구성 요소에서 사용할 프록시를 정의합니다. 참고: 현재 모든 구성 요소가 이 값을 사용하는 것은 아닙니다.
scheduler.config.openshift.io	정책 및 기본 노드 선택기와 같은 스케줄러 동작을 설정합니다.

3.4.2. Operator 설정 자원

이러한 설정 리소스는 **cluster**라는 클러스터 범위의 인스턴스로 특정 Operator가 소유한 특정 구성 요소의 동작을 제어합니다.

리소스 이름	설명
console.operator.openshift.io	브랜딩 사용자 정의와 같은 콘솔 모양을 제어합니다
config.imageregistry.operator.openshift.io	공용 라우팅, 로그 수준, 프록시 설정, 리소스 제약 조건, 복제본 수, 스토리지 유형과 같은 내부 이미지 레지스트리 설정을 구성합니다.
config.samples.operator.openshift.io	Samples Operator를 설정하여 클러스터에 설치된 이미지 스트림 및 템플릿 샘플을 제어합니다.

3.4.3. 추가 설정 리소스

이러한 설정 리소스는 특정 구성 요소의 단일 인스턴스를 나타냅니다. 경우에 따라 리소스의 여러 인스턴스를 작성하고 여러 인스턴스를 요청할 수 있습니다. 다른 경우 Operator는 특정 네임 스페이스에서 특정 리소스 인스턴스 이름 만 사용할 수 있습니다. 추가 리소스 인스턴스를 생성하는 방법과 시기에 대한 자세한 내용은 구성 요소 별 설명서를 참조하십시오.

리소스 이름	인스턴스 이름	네임 스페이스	설명
alertmanager.monitoring.coreos.com	main	openshift-monitoring	alertmanager 배포 매개 변수를 제어합니다.
ingresscontroller.operator.openshift.io	default	openshift-ingress-operator	도메인, 복제본 수, 인증서 및 컨트롤러 배치와 같은 Ingress Operator 동작을 설정합니다.

3.4.4. 정보 리소스

이러한 리소스를 사용하여 클러스터에 대한 정보를 검색합니다. 이러한 리소스를 직접 편집하지 마십시오.

리소스 이름	인스턴스 이름	설명
clusterversion.config.openshift.io	version	OpenShift Container Platform 4.5에서는 프로덕션 클러스터에 대한 ClusterVersion 리소스를 사용자 정의할 수 없습니다. 대신 클러스터 업데이트 프로세스를 실행합니다.
dns.config.openshift.io	cluster	클러스터의 DNS 설정을 변경할 수 없습니다. DNS Operator 상태를 표시할 수 있습니다.
infrastructure.config.openshift.io	cluster	클러스터가 클라우드 공급자와 상호 작용을 가능하게 하는 설정 세부 정보입니다.
network.config.openshift.io	cluster	설치 후 클러스터 네트워크를 변경할 수 없습니다. 네트워크를 사용자 정의하려면설치시 네트워크를 정의하는 프로세스를 실행합니다.

3.5. 방화벽 설정

방화벽을 사용하는 경우 OpenShift Container Platform이 작동하는데 필요한 사이트에 액세스할 수 있도록 방화벽을 설정해야합니다. 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.

3.5.1. OpenShift Container Platform의 방화벽 설정

OpenShift Container Platform을 설치하기 전에 OpenShift Container Platform에 필요한 사이트에 대한 액세스 권한을 부여하도록 방화벽을 설정해야합니다.

프로세스

1. 다음 레지스트리 URL을 허용 목록에 추가합니다.

URL	함수
registry.redhat.io	코어 컨테이너 이미지를 제공합니다.
quay.io	코어 컨테이너 이미지를 제공합니다.
sso.redhat.com	**https://cloud.redhat.com/openshift** 사이트에서 **sso.redhat.com**의 인증을 사용합니다.

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	함수
cert-api.access.redhat.com	Telemetry 필수
api.access.redhat.com	Telemetry 필수
infogw.api.openshift.com	Telemetry 필수
https://cloud.redhat.com/api/ingress	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 부여해야합니다.

클라우드	URL	함수
AWS	***.amazonaws.com**	AWS 서비스 및 리소스에 액세스하는데 필요합니다. 사용하는 리전에서 허용되는 특정 엔드 포인트를 확인하려면 AWS 설명서에서 AWS 서비스 엔드 포인트를 참조하십시오.
GCP	***.googleapis.com**	GCP 서비스 및 리소스에 액세스하는 데 필요합니다. API에서 허용하는 엔드 포인트를 확인하려면 GCP 설명서에서 Cloud Endpoints를 참조하십시오.
	accounts.google.com	GCP 계정에 액세스하는 데 필요합니다.

클라우드	URL	함수
Azure	**management.azure.com**	Azure 서비스 및 리소스에 액세스하는데 필요합니다. API에서 허용할 끝점을 확인하려면 Azure 설명서에서 Azure REST API Referenc를 참조하십시오.

5. 다음 URL을 허용 목록에 추가하십시오.

URL	함수
mirror.openshift.com	미러링된 설치 컨텐츠 및 이미지에 액세스하는 데 필요합니다. Cluster Version Operator에는 단일 기능 소스만 필요하지만 이 사이트는 릴리스 이미지 서명의 소스이기도 합니다.
storage.googleapis.com/openshift-release	릴리스 이미지 서명 소스입니다 (Cluster Version Operator에는 단일 기능 소스만 필요)
***.apps.<cluster_name>.<base_domain>**	설치 중에 ingress 와일드 카드를 설정하지 않으면 기본 클러스터 라우트에 액세스하는데 필요합니다.
quay-registry.s3.amazonaws.com	AWS에서 Quay 이미지 컨텐츠에 액세스하는데 필요합니다.
api.openshift.com	클러스터에 사용 가능한 업데이트가 있는지 확인하는 데 필요합니다.
art-rhcos-ci.s3.amazonaws.com	RHCOS (Red Hat Enterprise Linux CoreOS) 이미지를 다운로드하는 데 필요합니다.
api.openshift.com	클러스터 토큰에 필요합니다.
cloud.redhat.com/openshift	클러스터 토큰에 필요합니다.

3.6. 프라이빗 클러스터 설정

OpenShift Container Platform 버전 4.5 클러스터를 설치한 후 일부 핵심 구성 요소를 프라이빗으로 설정할 수 있습니다.

중요

이 변경은 클라우드 공급자에게 프로비저닝한 인프라를 사용하는 클러스터에 대해서만 설정할 수 있습니다.

3.6.1. 프라이빗 클러스터 정보

기본적으로 OpenShift Container Platform은 공개적으로 액세스 가능한 DNS 및 엔드 포인트를 사용하여 프로비저닝됩니다. 클러스터를 배포 한 후 DNS, Ingress 컨트롤러 및 API 서버를 프라이빗으로 설정할 수 있습니다.

DNS
설치 프로그램이 프로비저닝한 인프라에 OpenShift Container Platform을 설치하는 경우 설치 프로그램은 기존 퍼블릭 영역에 레코드를 만들고 가능한 경우 클러스터 자체 DNS 확인을 위한 프라이빗 영역을 만듭니다. 퍼블릭 영역과 프라이빗 영역 모두에서 설치 프로그램 또는 클러스터는 API 서버에 대한 *** .apps**, Ingress, **api**의 DNS 항목을 만듭니다.

퍼블릭 영역과 프라이빗 영역의 *** .apps** 레코드는 동일하므로 퍼블릭 영역을 삭제하면 프라이빗 영역이 클러스터에 대한 모든 DNS 확인을 완벽하게 제공합니다.

Ingress 컨트롤러
기본 Ingress 개체는 퍼블릭으로 생성되기 때문에 로드 밸런서는 인터넷에 연결되어 퍼블릭 서브넷에서 사용됩니다. 기본 Ingress 컨트롤러를 내부 컨트롤러로 교체할 수 있습니다.

API 서버
기본적으로 설치 프로그램은 API 서버가 내부 트래픽 및 외부 트래픽 모두에 사용할 적절한 네트워크로드 밸런서를 만듭니다.

AWS (Amazon Web Services)에서 별도의 퍼블릭 및 프라이빗 로드 밸런서가 생성됩니다. 클러스터에서 사용하기 위해 내부 포트에서 추가 포트를 사용할 수 있다는 점을 제외하고 로드 밸런서는 동일합니다. 설치 프로그램이 API 서버 요구 사항에 따라 로드 밸런서를 자동으로 생성하거나 제거하더라도 클러스터는 이를 관리하거나 유지하지 않습니다. API 서버에 대한 클러스터의 액세스 권한을 유지하는 한 로드 밸런서를 수동으로 변경하거나 이동할 수 있습니다. 퍼블릭 로드 밸런서의 경우 포트 6443이 열려있고 상태 확인은 HTTPS의 / **readyz** 경로에 대해 설정되어 있습니다.

Google Cloud Platform에서 내부 및 외부 API 트래픽을 모두 관리하기 위해 단일 로드 밸런서가 생성되므로 로드 밸런서를 변경할 필요가 없습니다.

Microsoft Azure에서는 퍼블릭 및 프라이빗 로드 밸런서가 모두 생성됩니다. 그러나 현재 구현에 한계가 있기 때문에 프라이빗 클러스터에서 두 로드 밸런서를 유지합니다.

3.6.2. DNS를 프라이빗으로 설정

클러스터를 배포한 후 프라이빗 영역만 사용하도록 DNS를 변경할 수 있습니다.

프로세스

1. 클러스터의 DNS 사용자 정의 리소스를 확인합니다.

```
$ oc get dnses.config.openshift.io/cluster -o yaml
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: "2019-10-25T18:27:09Z"
  generation: 2
  name: cluster
  resourceVersion: "37966"
  selfLink: /apis/config.openshift.io/v1/dnses/cluster
  uid: 0e714746-f755-11f9-9cb1-02ff55d8f976
spec:
  baseDomain: <base_domain>
  privateZone:
    tags:
```

```
      Name: <infrastructureID>-int
      kubernetes.io/cluster/<infrastructureID>: owned
  publicZone:
    id: Z2XXXXXXXXXXA4
status: {}
```

spec 섹션에는 프라이빗 영역과 퍼블릭 영역이 모두 포함되어 있습니다.

2. DNS 사용자 지정 리소스를 패치하여 퍼블릭 영역을 제거합니다.

```
$ oc patch dnses.config.openshift.io/cluster --type=merge --patch='{"spec": {"publicZone":
null}}'
dns.config.openshift.io/cluster patched
```

Ingress 컨트롤러는 Ingress 개체를 만들 때 DNS 정의를 참조하기 때문에 Ingress 개체를 만들거나 수정할 때 프라이빗 레코드만 생성됩니다.

중요

퍼블릭 영역을 제거해도 기존 Ingress 개체의 DNS 레코드는 변경되지 않습니다.

3. 선택 사항: 클러스터의 DNS 사용자 정의 리소스를 확인하고 퍼블릭 영역이 제거되었는지 확인하십시오.

```
$ oc get dnses.config.openshift.io/cluster -o yaml
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: "2019-10-25T18:27:09Z"
  generation: 2
  name: cluster
  resourceVersion: "37966"
  selfLink: /apis/config.openshift.io/v1/dnses/cluster
  uid: 0e714746-f755-11f9-9cb1-02ff55d8f976
spec:
  baseDomain: <base_domain>
  privateZone:
    tags:
      Name: <infrastructureID>-int
      kubernetes.io/cluster/<infrastructureID>-wfpg4: owned
status: {}
```

3.6.3. Ingress 컨트롤러를 프라이빗으로 설정

클러스터를 배포한 후 프라이빗 영역만 사용하도록 Ingress 컨트롤러를 변경할 수 있습니다.

프로세스

1. 내부 엔드 포인트만 사용하도록 기본 Ingress 컨트롤러를 변경합니다.

```
$ oc replace --force --wait --filename - <<EOF
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
```

```
  namespace: openshift-ingress-operator
  name: default
spec:
  endpointPublishingStrategy:
    type: LoadBalancerService
    loadBalancer:
      scope: Internal
EOF
ingresscontroller.operator.openshift.io "default" deleted
ingresscontroller.operator.openshift.io/default replaced
```

퍼블릭 DNS 항목이 제거되고 프라이빗 영역 항목이 업데이트됩니다.

3.6.4. API 서버를 프라이빗으로 제한

AWS (Amazon Web Services) 또는 Microsoft Azure에 클러스터를 배포한 후 프라이빗 영역만 사용하도록 API 서버를 재구성할 수 있습니다.

전제 조건

- **oc**로 알려진 OpenShift 명령 인터페이스 (CLI)를 설치합니다.
- **admin** 권한이 있는 사용자로 웹 콘솔에 액세스합니다.

프로세스

1. AWS 또는 Azure의 웹 포털 또는 콘솔에서 다음 작업을 수행합니다.
   a. 적절한 로드 밸런서 구성 요소를 찾아 삭제합니다.
      - AWS의 경우 외부 로드 밸런서를 삭제합니다. 프라이빗 영역의 API DNS 항목은 동일한 설정을 사용하는 내부 로드 밸런서를 가리키므로 내부 로드 밸런서를 변경할 필요가 없습니다.
      - Azure의 경우 로드 밸런서의 **api-internal** 규칙을 삭제합니다.
   b. 퍼블릭 영역의 **api.$clustername.$yourdomain** DNS 항목을 삭제합니다.
2. 터미널에서 클러스터 시스템을 나열합니다.

```
$ oc get machine -n openshift-machine-api
NAME                          STATE    TYPE       REGION     ZONE         AGE
lk4pj-master-0                running  m4.xlarge  us-east-1  us-east-1a   17m
lk4pj-master-1                running  m4.xlarge  us-east-1  us-east-1b   17m
lk4pj-master-2                running  m4.xlarge  us-east-1  us-east-1a   17m
lk4pj-worker-us-east-1a-5fzfj running  m4.xlarge  us-east-1  us-east-1a   15m
lk4pj-worker-us-east-1a-vbghs running  m4.xlarge  us-east-1  us-east-1a   15m
lk4pj-worker-us-east-1b-zgpzg running  m4.xlarge  us-east-1  us-east-1b   15m
```

다음 단계에서 이름에 **master**가 포함된 컨트롤 플레인 시스템을 변경합니다.

3. 각 컨트롤 플레인 시스템에서 외부 로드 밸런서를 제거합니다.
   a. **master** 시스템 개체를 편집하여 외부 로드 밸런서에 대한 참조를 제거합니다.

```
$ oc edit machines -n openshift-machine-api <master_name> 1
```

-

1 변경할 컨트롤 플레인 또는 마스터 시스템의 이름을 지정합니다.

b. 다음 예에 표시된 외부 로드 밸런서를 설명하는 행을 제거하고 개체 사양을 저장한 후 종료합니다.

```
...
spec:
  providerSpec:
    value:
    ...
      loadBalancers:
      - name: lk4pj-ext 1
        type: network 2
      - name: lk4pj-int
        type: network
```

1 2 이 줄을 삭제합니다.

c. 이름에 **master**가 포함된 각 시스템에 대해 이 프로세스를 반복합니다.