OpenShift Container Platform 4.6

머신 관리

클러스터 머신 추가 및 유지 보수

Last Updated: 2023-06-23

OpenShift Container Platform 4.6 머신 관리

클러스터 머신 추가 및 유지 보수

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

법적 공지

초록

이 문서에서는 OpenShift Container Platform 클러스터를 구성하는 머신을 관리하는 방법을 설명합니다. 일부 작업은 OpenShift Container Platform 클러스터의 고급 자동 머신 관리 기능을 사용하며 일부 작업은 수동으로 완료해야 합니다. 이 문서에서 설명하는 모든 작업이 모든 설치 유형에서 사용 가능한 것은 아닙니다.

차례

1장. 머신 관리 개요 **5**
머신 세트로 수행할 수 있는 작업 5
자동 스케일러 5
사용자 프로비저닝 인프라 6
RHEL 컴퓨팅 머신에서 수행할 수 있는 작업 6

2장. 머신 세트 생성 **7**
2.1. AWS에서 머신 세트 생성 7
2.1.1. Machine API 개요 7
2.1.2. AWS에서 머신 세트 사용자 지정 리소스의 샘플 YAML 8
2.1.3. 머신 세트 만들기 9
2.1.4. 머신을 Spot 인스턴스로 배포하는 머신 세트 11
2.1.5. 머신 세트를 사용하여 Spot 인스턴스 생성 11
2.2. AZURE에서 머신 세트 만들기 12
2.2.1. Machine API 개요 12
2.2.2. Azure의 머신 세트사용자 지정 리소스에 대한 샘플 YAML 13
2.2.3. 머신 세트 만들기 15
2.2.4. 머신을 Spot 가상머신으로 배포하는 머신 세트 16
2.2.5. 머신 세트를 사용하여 Spot 가상 머신 생성 17
2.2.6. 머신 세트의 고객 관리 암호화 키 활성화 17
2.3. GCP에서 머신 세트 생성 18
2.3.1. Machine API 개요 18
2.3.2. GCP에서 머신 세트 사용자 정의 리소스의 샘플 YAML 19
2.3.3. 머신 세트 만들기 21
2.3.4. 머신을 선점할 수 있는 가상 머신 인스턴스로 배포하는 머신 세트 22
2.3.5. 머신 세트를 사용하여 선점 가능한 가상 머신 인스턴스 생성 23
2.4. OPENSTACK에서 머신 세트 생성 23
2.4.1. Machine API 개요 23
2.4.2. RHOSP에서 머신 세트 사용자 정의 리소스의 샘플 YAML 24
2.4.3. 머신 세트 만들기 26
2.5. RHV에 머신 세트 생성 27
2.5.1. Machine API 개요 28
2.5.2. RHV에서 머신 세트 사용자 정의 리소스의 샘플 YAML 29
2.5.3. 머신 세트 만들기 31
2.6. VSPHERE에서 머신 세트 생성 32
2.6.1. Machine API 개요 32
2.6.2. vSphere에서 머신 세트 사용자 정의 리소스의 샘플 YAML 33
2.6.3. 머신 세트 만들기 35

3장. 머신 세트 수동 스케일링 **38**
3.1. 사전 요구 사항 38
3.2. 머신 세트 수동 스케일링 38
3.3. 머신 세트 삭제 정책 39

4장. 머신 세트 수정 **40**
4.1. 머신 세트 수정 40
4.2. RHV의 다른 스토리지 도메인으로 노드 마이그레이션 41
4.2.1. RHV에서 다른 스토리지 도메인으로 컴퓨팅 노드 마이그레이션 41
4.2.2. 컨트롤 플레인 노드를 RHV의 다른 스토리지 도메인으로 마이그레이션 42

5장. 머신 삭제 **43**
5.1. 특정 머신 삭제 43

5.2. 추가 리소스 43

6장. OPENSHIFT CONTAINER PLATFORM 클러스터에 자동 스케일링 적용 44
6.1. 클러스터 자동 스케일러 정보 44
6.2. 머신 자동 스케일러 정보 45
6.3. 클러스터 자동 스케일러 구성 46
6.3.1. ClusterAutoscaler 리소스 정의 46
6.3.2. 클러스터 자동 스케일러 배포 47
6.4. 다음 단계 48
6.5. 머신 자동 스케일러 구성 48
6.5.1. MachineAutoscaler 리소스 정의 48
6.5.2. 머신 자동 스케일러 배포 49
6.6. 추가 리소스 49

7장. 인프라 머신 세트 생성 50
7.1. OPENSHIFT CONTAINER PLATFORM 인프라 구성 요소 50
7.2. 프로덕션 환경의 인프라 머신 세트 생성 50
7.2.1. 다른 클라우드의 머신 세트 생성 51
7.2.1.1. AWS에서 머신 세트 사용자 지정 리소스의 샘플 YAML 51
7.2.1.2. Azure의 머신 세트사용자 지정 리소스에 대한 샘플 YAML 52
7.2.1.3. GCP에서 머신 세트 사용자 정의 리소스의 샘플 YAML 55
7.2.1.4. RHOSP에서 머신 세트 사용자 정의 리소스의 샘플 YAML 56
7.2.1.5. vSphere에서 머신 세트 사용자 정의 리소스의 샘플 YAML 58
7.2.2. 머신 세트 만들기 60
7.2.3. 인프라 노드 생성 61
7.2.4. 인프라 머신의 머신 구성 풀 생성 62
7.3. 인프라 노드에 머신 세트 리소스 할당 66
7.3.1. 테인트 및 허용 오차를 사용하여 인프라 노드 워크로드 바인딩 66
7.4. 인프라 머신 세트로 리소스 이동 67
7.4.1. 라우터 이동 67
7.4.2. 기본 레지스트리 이동 69
7.4.3. 모니터링 솔루션 이동 70
7.4.4. 클러스터 로깅 리소스 이동 71

8장. OPENSHIFT CONTAINER PLATFORM 클러스터에 RHEL 컴퓨팅 머신 추가 76
8.1. 클러스터에 RHEL 컴퓨팅 노드 추가 정보 76
8.2. RHEL 컴퓨팅 노드의 시스템 요구 사항 76
8.2.1. 인증서 서명 요청 관리 77
8.3. 클라우드 이미지 준비 77
8.3.1. AWS에서 사용 가능한 최신 RHEL 이미지 나열 78
8.4. PLAYBOOK 실행을 위한 머신 준비 79
8.5. RHEL 컴퓨팅 노드 준비 80
8.6. AWS의 RHEL 인스턴스에 역할 권한 연결 81
8.7. RHEL 작업자 노드를 OWNED 또는 SHARED로 태그 지정 81
8.8. 클러스터에 RHEL 컴퓨팅 머신 추가 82
8.9. 시스템의 인증서 서명 요청 승인 83
8.10. ANSIBLE 호스트 파일의 필수 매개 변수 85
8.10.1. 선택 사항: 클러스터에서 RHCOS 컴퓨팅 머신 제거 86

9장. OPENSHIFT CONTAINER PLATFORM 클러스터에 RHEL 컴퓨팅 머신 추가 88
9.1. 클러스터에 RHEL 컴퓨팅 노드 추가 정보 88
9.2. RHEL 컴퓨팅 노드의 시스템 요구 사항 88
9.2.1. 인증서 서명 요청 관리 89
9.3. 클라우드 이미지 준비 89

9.3.1. AWS에서 사용 가능한 최신 RHEL 이미지 나열 90
9.4. RHEL 컴퓨팅 노드 준비 91
9.5. AWS의 RHEL 인스턴스에 역할 권한 연결 92
9.6. RHEL 작업자 노드를 OWNED 또는 SHARED로 태그 지정 92
9.7. 클러스터에 더 많은 RHEL 컴퓨팅 시스템 업데이트 추가 93
9.8. 시스템의 인증서 서명 요청 승인 94
9.9. ANSIBLE 호스트 파일의 필수 매개 변수 96

10장. 사용자 프로비저닝 인프라 98
10.1. 사용자 프로비저닝 인프라가 있는 클러스터에 컴퓨팅 머신 추가 98
10.1.1. Amazon Web Services에 컴퓨팅 머신 추가 98
10.1.2. Microsoft Azure에 컴퓨팅 머신 추가 98
10.1.3. Google Cloud Platform에 컴퓨팅 머신 추가 98
10.1.4. vSphere에 컴퓨팅 머신 추가 98
10.1.5. 베어 메탈에 컴퓨팅 머신 추가 98
10.2. CLOUDFORMATION 템플릿을 사용하여 AWS에 컴퓨팅 머신 추가 98
10.2.1. 사전 요구 사항 98
10.2.2. CloudFormation 템플릿을 사용하여 AWS 클러스터에 추가 98
10.2.3. 머신의 인증서 서명 요청 승인 99
10.3. VSPHERE에 컴퓨팅 머신 추가 102
10.3.1. 전제 조건 102
10.3.2. vSphere에서 RHCOS(Red Hat Enterprise Linux CoreOS) 머신 추가 생성 102
10.3.3. 시스템의 인증서 서명 요청 승인 103
10.4. 베어 메탈에 컴퓨팅 머신 추가 106
10.4.1. 전제 조건 106
10.4.2. RHCOS(Red Hat Enterprise Linux CoreOS) 시스템 생성 106
10.4.2.1. ISO 이미지를 사용하여 추가 RHCOS 머신 생성 107
10.4.2.2. PXE 또는 iPXE 부팅을 통해 RHCOS 머신 생성 107
10.4.3. 시스템의 인증서 서명 요청 승인 109

11장. 머신 상태 점검 배포 112
11.1. 머신 상태 점검 정보 112
11.1.1. 베어 메탈에서 MachineHealthCheck 112
11.1.2. 머신 상태 검사 배포 시 제한 사항 112
11.2. MACHINEHEALTHCHECK 리소스 샘플 113
11.2.1. 쇼트 서킷 (Short Circuit) 머신 상태 점검 및 수정 115
11.2.1.1. 절대 값을 사용하여 maxUnhealthy 설정 115
11.2.1.2. 백분율을 사용하여 maxUnhealthy 설정 115
11.3. MACHINEHEALTHCHECK 리소스 만들기 116

1장. 머신 관리 개요

머신 관리를 사용하여 AWS(Amazon Web Services), Azure, GCP(Google Cloud Platform), OpenStack, RHV(Red Hat Virtualization) 및 vSphere와 같은 기본 인프라에서 OpenShift Container Platform 클러스터를 유연하게 관리할 수 있습니다. 클러스터를 제어하고 특정 워크로드 정책에 따라 클러스터 확장 및 축소와 같은 자동 확장을 수행할 수 있습니다.

OpenShift Container Platform 클러스터는 로드가 증가하거나 감소하면 수평으로 확장 및 축소할 수 있습니다. 변화하는 워크로드에 맞게 조정되는 클러스터가 있어야 합니다.

머신 관리는 CRD(Custom Resource Definition)로 구현됩니다. CRD 오브젝트는 클러스터에서 새로운 고유한 오브젝트 **종류를** 정의하고 Kubernetes API 서버가 오브젝트의 전체 라이프사이클을 처리할 수 있도록 합니다.

Machine API Operator는 다음 리소스를 프로비저닝합니다.

- MachineSet
- Machine
- Cluster Autoscaler
- Machine Autoscaler
- 머신 상태 점검

머신 세트로 수행할 수 있는 작업

클러스터 관리자는 다음을 수행할 수 있습니다.

- 에 머신 세트를 생성합니다.
  - AWS
  - Azure
  - GCP
  - OpenStack
  - RHV
  - vSphere
- 머신 세트에서 머신을 추가하거나 제거하여 머신 세트를 수동으로 확장합니다 .
- MachineSet YAML 구성 파일을 통해 머신 세트를 수정합니다.
- 시스템을 삭제합니다.
- 인프라 머신 세트를 생성합니다.
- 머신 풀에서 손상된 머신을 자동으로 수정하도록 머신 상태 점검을 구성하고 배포합니다.

자동 스케일러

워크로드를 변경하는 유연성을 보장하기 위해 클러스터를 자동 스케일링합니다. OpenShift Container Platform 클러스터를 자동 확장하려면 먼저 클러스터 자동 스케일러를 배포한 다음 각 머신 세트에 대한 머신 자동 스케일러를 배포해야 합니다. 클러스터 자동 스케일러는 배포 요구 사항에 따라 클러스터 크기

를 늘리고 줄입니다. 머신 자동 스케일러는 OpenShift Container Platform 클러스터에 배포하는 머신 세트의 머신 수를 조정합니다.

사용자 프로비저닝 인프라

사용자 프로비저닝 인프라는 OpenShift Container Platform을 호스팅하는 컴퓨팅, 네트워크 및 스토리지 리소스와 같은 인프라를 배포할 수 있는 환경입니다. 사용자 프로비저닝 인프라의 클러스터에 컴퓨팅 머신을 설치 프로세스의 일부로 추가할 수 있습니다.

RHEL 컴퓨팅 머신에서 수행할 수 있는 작업

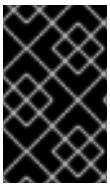
클러스터 관리자는 다음을 수행할 수 있습니다.

- 사용자 프로비저닝 인프라 클러스터 또는 설치 프로비저닝 인프라 클러스터에 RHEL (Red Hat Enterprise Linux) 컴퓨팅 머신 (작업자 머신이라고도 함)을 추가합니다.
- RHEL(Red Hat Enterprise Linux) 컴퓨팅 머신을 기존 클러스터에 추가합니다 .

2장. 머신 세트 생성

2.1. AWS에서 머신 세트 생성

AWS (Amazon Web Services)의 OpenShift Container Platform 클러스터에서 특정 목적을 수행하기 위해 다른 머신 세트를 생성할 수 있습니다. 예를 들어, 지원되는 워크로드를 새 머신으로 이동할 수 있도록 인프라 머신 세트 및 관련 머신을 작성할 수 있습니다.



중요

이 프로세스는 수동으로 프로비저닝된 시스템이 있는 클러스터에는 적용되지 않습니다. 머신 API가 작동하는 클러스터에서만 고급 머신 관리 및 스케일링 기능을 사용할 수 있습니다.

2.1.1. Machine API 개요

Machine API는 업스트림 Cluster API 프로젝트 및 사용자 정의 OpenShift Container Platform 리소스를 기반으로하는 주요 리소스의 조합입니다.

OpenShift Container Platform 4.6 클러스터의 경우 Machine API는 클러스터 설치가 완료된 후 모든 노드 호스트 프로비저닝 관리 작업을 수행합니다. 이 시스템으로 인해 OpenShift Container Platform 4.6은 퍼블릭 또는 프라이빗 클라우드 인프라에 더하여 탄력적이고 동적인 프로비저닝 방법을 제공합니다.

두 가지 주요 리소스는 다음과 같습니다.

Machine

노드의 호스트를 설명하는 기본 단위입니다. 머신에는 **providerSpec** 사양이 있으며 이는 다른 클라우드 플랫폼에 제공되는 컴퓨팅 노드 유형을 설명합니다. 예를 들어 AWS(Amazon Web Services)의 작업자 노드에 대한 머신 유형은 특정 머신 유형과 필요한 메타 데이터를 정의할 수 있습니다.

머신 세트

MachineSet 리소스는 머신 그룹입니다. 머신 세트는 머신에 연관되어 있고 복제본 세트는 pod에 연관되어 있습니다. 더 많은 머신이 필요하거나 규모를 줄여야 하는 경우 컴퓨터 요구 사항에 맞게 머신 세트의 **replicas** 필드를 변경합니다.

주의

컨트롤 플레인 시스템은 머신 세트에서 관리할 수 없습니다.

다음 사용자 지정 리소스는 클러스터에 더 많은 기능을 추가할 수 있습니다.

머신 자동 스케일러

MachineAutoscaler 리소스는 클라우드에서 머신을 자동으로 확장합니다. 지정된 머신 세트에서 노드의 최소 및 최대 스케일링 경계를 설정할 수 있으며 머신 자동 스케일러는 해당 노드 범위를 유지합니다. **MachineAutoscaler** 객체는 **ClusterAutoscaler** 객체를 설정한 후에 사용할 수 있습니다. **ClusterAutoscaler** 및 **MachineAutoscaler**리소스는 모두 **ClusterAutoscalerOperator** 오브젝트에서 사용 가능합니다.

Cluster autoscaler

이 리소스는 업스트림 클러스터 자동 스케일러 프로젝트를 기반으로 합니다. OpenShift Container Platform 구현에서는 머신 세트 API를 확장하여 Machine API와 통합됩니다. 코어, 노드, 메모리, GPU 등과 같은 리소스의 클러스터 전체에서 확장 제한을 설정할 수 있습니다. 중요도가 낮은 Pod에 대해 새 노드가 온라인 상태가 되지 않도록 클러스터가 Pod에 우선 순위를 설정할 수 있습니다. 노드를 확장할 수는 있지만 축소할 수 없도록 확장 정책을 설정할 수도 있습니다.

머신 상태 점검

MachineHealthCheck 리소스는 머신의 비정상적인 상태를 감지하여 삭제한 후 지원되는 플랫폼에서 새 머신을 생성합니다.

OpenShift Container Platform 버전 3.11에서는 클러스터가 머신 프로비저닝을 관리하지 않았기 때문에 다중 영역 아키텍처를 쉽게 롤아웃할 수 없었습니다. OpenShift Container Platform 버전 4.1부터 이러한 프로세스가 더 쉬워졌습니다. 각 머신 세트의 범위는 단일 영역에서 지정되므로 설치 프로그램은 사용자를 대신하여 가용성 영역 전체에 머신 세트를 보냅니다. 또한 계산이 동적이고 영역 장애가 발생하여 머신을 재조정해야하는 경우 처리할 수 있는 영역을 확보할 수 있습니다. Autoscaler는 클러스터의 수명 기간 동안 최적의 균형을 유지합니다.

2.1.2. AWS에서 머신 세트 사용자 지정 리소스의 샘플 YAML

이 샘플 YAML은 **us-east-1a** AWS (Amazon Web Services) 영역에서 실행되는 머신 세트를 정의하고 **node-role.kubernetes.io/<role>: ""** 로 레이블이 지정된 노드를 생성합니다.

이 샘플에서 **<infrastructure_id>**는 클러스터를 프로비저닝할 때 설정한 클러스터 ID를 기반으로 하는 인프라 ID 레이블이며 **<role>**은 추가할 노드 레이블입니다.

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
  name: <infrastructure_id>-<role>-<zone> 2
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 3
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<zone> 4
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
        machine.openshift.io/cluster-api-machine-role: <role> 6
        machine.openshift.io/cluster-api-machine-type: <role> 7
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<zone> 8
    spec:
      metadata:
        labels:
          node-role.kubernetes.io/<role>: "" 9
      providerSpec:
        value:
          ami:
            id: ami-046fe691f52a953f9 10
          apiVersion: awsproviderconfig.openshift.io/v1beta1
```

```
      blockDevices:
        - ebs:
            iops: 0
            volumeSize: 120
            volumeType: gp2
      credentialsSecret:
        name: aws-cloud-credentials
      deviceIndex: 0
      iamInstanceProfile:
        id: <infrastructure_id>-worker-profile 11
      instanceType: m4.large
      kind: AWSMachineProviderConfig
      placement:
        availabilityZone: us-east-1a
        region: us-east-1
      securityGroups:
        - filters:
            - name: tag:Name
              values:
                - <infrastructure_id>-worker-sg 12
      subnet:
        filters:
          - name: tag:Name
            values:
              - <infrastructure_id>-private-us-east-1a 13
      tags:
        - name: kubernetes.io/cluster/<infrastructure_id> 14
          value: owned
      userDataSecret:
        name: worker-user-data
```

1 3 5 11 12 13 14 클러스터를 프로비저닝할 때 설정한 클러스터 ID를 기반으로 하는 인프라 ID를 지정합니다. OpenShift CLI 패키지가 설치되어 있으면 다음 명령을 실행하여 인프라 ID를 얻을 수 있습니다.

```
$ oc get -o jsonpath='{.status.infrastructureName}{"\n"}' infrastructure cluster
```

2 4 8 인프라 ID, 노드 레이블 및 영역을 지정합니다.

6 7 9 추가할 노드 레이블을 지정합니다.

10 OpenShift Container Platform 노드의 AWS(Amazon Web Services) 영역에 유효한 RHCOS(Red Hat Enterprise Linux CoreOS) AMI를 지정합니다.

2.1.3. 머신 세트 만들기

설치 프로그램에 의해 생성되는 것 이외에도 고유한 머신 세트를 만들어 선택한 특정 워크로드의 머신 컴퓨팅 리소스를 동적으로 관리할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포합니다.
- OpenShift CLI(**oc**)를 설치합니다.

- **cluster-admin** 권한이 있는 사용자로 **oc**에 로그인합니다.

절차

1. 머신 세트 CR(사용자 지정 리소스) 샘플이 포함된 이름이 **<file_name>.yaml**인 새 YAML 파일을 만듭니다.
   <clusterID> 및 **<role>** 매개 변수 값을 설정해야 합니다.

   a. 특정 필드에 설정할 값이 확실하지 않은 경우 클러스터에서 기존 머신 세트를 확인할 수 있습니다.

   ```
   $ oc get machinesets -n openshift-machine-api
   ```

   출력 예

   ```
   NAME                                DESIRED   CURRENT   READY   AVAILABLE   AGE
   agl030519-vplxk-worker-us-east-1a   1         1         1       1           55m
   agl030519-vplxk-worker-us-east-1b   1         1         1       1           55m
   agl030519-vplxk-worker-us-east-1c   1         1         1       1           55m
   agl030519-vplxk-worker-us-east-1d   0         0                             55m
   agl030519-vplxk-worker-us-east-1e   0         0                             55m
   agl030519-vplxk-worker-us-east-1f   0         0                             55m
   ```

   b. 특정 머신 세트의 값을 확인합니다.

   ```
   $ oc get machineset <machineset_name> -n \
     openshift-machine-api -o yaml
   ```

   출력 예

   ```
   ...
   template:
     metadata:
       labels:
         machine.openshift.io/cluster-api-cluster: agl030519-vplxk 1
         machine.openshift.io/cluster-api-machine-role: worker 2
         machine.openshift.io/cluster-api-machine-type: worker
         machine.openshift.io/cluster-api-machineset: agl030519-vplxk-worker-us-east-1a
   ```

   1 클러스터 ID입니다.

   2 기본 노드 레이블입니다.

2. 새 **MachineSet** CR을 만듭니다.

   ```
   $ oc create -f <file_name>.yaml
   ```

3. 머신 세트 목록을 표시합니다.

   ```
   $ oc get machineset -n openshift-machine-api
   ```

   출력 예

```
NAME                           DESIRED   CURRENT   READY   AVAILABLE   AGE
agl030519-vplxk-infra-us-east-1a   1         1         1       1           11m
agl030519-vplxk-worker-us-east-1a  1         1         1       1           55m
agl030519-vplxk-worker-us-east-1b  1         1         1       1           55m
agl030519-vplxk-worker-us-east-1c  1         1         1       1           55m
agl030519-vplxk-worker-us-east-1d  0         0                             55m
agl030519-vplxk-worker-us-east-1e  0         0                             55m
agl030519-vplxk-worker-us-east-1f  0         0                             55m
```

새 머신 세트가 사용 가능한 경우**DESIRED** 및 **CURRENT** 값이 일치합니다. 머신 세트를 사용할 수 없는 경우 몇 분 후에 명령을 다시 실행합니다.

다음 단계

다른 가용 영역에 머신 세트가 필요한 경우 이 프로세스를 반복하여 추가 머신 세트를 생성합니다.

2.1.4. 머신을 Spot 인스턴스로 배포하는 머신 세트

AWS에서 실행되는 머신 세트를 생성하여 보장되지 않는 Spot 인스턴스로 머신을 배포하면 비용을 절감할 수 있습니다. Spot 인스턴스는 사용되지 않는 AWS EC2 용량을 사용하며 온 디맨드 인스턴스보다 저렴합니다. 일괄 처리 또는 상태 비저장, 수평적으로 확장 가능한 워크로드와 같이 인터럽트를 허용할 수 있는 워크로드에 Spot 인스턴스를 사용할 수 있습니다.

AWS EC2는 언제든지 Spot 인스턴스를 종료할 수 있습니다. AWS는 중단이 발생하면 사용자에게 2 분 동안 경고 메세지를 보냅니다. OpenShift Container Platform은 AWS가 종료에 대한 경고를 발행할 때 영향을 받는 인스턴스에서 워크로드를 제거하기 시작합니다.

다음과 같은 이유로 Spot 인스턴스를 사용할 때 중단될 수 있습니다.

- 인스턴스 가격이 최대 가격을 초과합니다.
- Spot 인스턴스에 대한 수요가 증가합니다.
- Spot 인스턴스의 공급이 감소합니다.

AWS가 인스턴스를 종료하면 Spot 인스턴스 노드에서 실행중인 종료 프로세스가 머신 리소스를 삭제합니다. 머신 세트 **replicas** 수량을 충족하기 위해 머신 세트는 Spot 인스턴스를 요청하는 머신을 생성합니다.

2.1.5. 머신 세트를 사용하여 Spot 인스턴스 생성

머신 세트 YAML 파일에 **spotMarketOptions**를 추가하여 AWS에서 Spot 인스턴스를 시작할 수 있습니다.

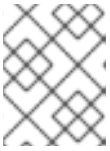
절차

- **providerSpec** 필드 아래에 다음 행을 추가합니다.

```
providerSpec:
  value:
    spotMarketOptions: {}
```

선택 옵션으로 **spotMarketOptions.maxPrice** 필드를 설정하여 Spot 인스턴스의 비용을 제한할 수 있습니다. 예를 들어 **maxPrice를 설정할 수 있습니다. '2.50'**.

maxPrice가 설정된 경우 이 값은 시간당 최대 Spot 가격으로 사용됩니다. 이 값이 설정되지 않은 경우 기본적으로 최대 가격은 온 디맨드 인스턴스 가격까지 청구됩니다.

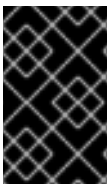


참고

기본적인 온 디맨드 가격을 **maxPrice** 값으로 사용하여 Spot 인스턴스의 최대 가격을 설정하지 않는 것이 좋습니다.

2.2. AZURE에서 머신 세트 만들기

Microsoft Azure의 OpenShift Container Platform 클러스터에서 특정 목적을 충족하기 위해 다른 머신 세트를 만들 수 있습니다. 예를 들어, 지원되는 워크로드를 새 머신으로 이동할 수 있도록 인프라 머신 세트 및 관련 머신을 작성할 수 있습니다.



중요

이 프로세스는 수동으로 프로비저닝된 시스템이 있는 클러스터에는 적용되지 않습니다. 머신 API가 작동하는 클러스터에서만 고급 머신 관리 및 스케일링 기능을 사용할 수 있습니다.

2.2.1. Machine API 개요

Machine API는 업스트림 Cluster API 프로젝트 및 사용자 정의 OpenShift Container Platform 리소스를 기반으로하는 주요 리소스의 조합입니다.

OpenShift Container Platform 4.6 클러스터의 경우 Machine API는 클러스터 설치가 완료된 후 모든 노드 호스트 프로비저닝 관리 작업을 수행합니다. 이 시스템으로 인해 OpenShift Container Platform 4.6은 퍼블릭 또는 프라이빗 클라우드 인프라에 더하여 탄력적이고 동적인 프로비저닝 방법을 제공합니다.

두 가지 주요 리소스는 다음과 같습니다.

Machine

노드의 호스트를 설명하는 기본 단위입니다. 머신에는 **providerSpec** 사양이 있으며 이는 다른 클라우드 플랫폼에 제공되는 컴퓨팅 노드 유형을 설명합니다. 예를 들어 AWS(Amazon Web Services)의 작업자 노드에 대한 머신 유형은 특정 머신 유형과 필요한 메타 데이터를 정의할 수 있습니다.

머신 세트

MachineSet 리소스는 머신 그룹입니다. 머신 세트는 머신에 연관되어 있고 복제본 세트는 pod에 연관되어 있습니다. 더 많은 머신이 필요하거나 규모를 줄여야 하는 경우 컴퓨터 요구 사항에 맞게 머신 세트의 **replicas** 필드를 변경합니다.

주의

컨트롤 플레인 시스템은 머신 세트에서 관리할 수 없습니다.

다음 사용자 지정 리소스는 클러스터에 더 많은 기능을 추가할 수 있습니다.

머신 자동 스케일러

MachineAutoscaler 리소스는 클라우드에서 머신을 자동으로 확장합니다. 지정된 머신 세트에서 노드의 최소 및 최대 스케일링 경계를 설정할 수 있으며 머신 자동 스케일러는 해당 노드 범위를 유지합니다. **MachineAutoscaler** 객체는 **ClusterAutoscaler** 객체를 설정한 후에 사용할 수 있습니다.

ClusterAutoscaler 및 **MachineAutoscaler**리소스는 모두 **ClusterAutoscalerOperator** 오브젝트에서 사용 가능합니다.

Cluster autoscaler

이 리소스는 업스트림 클러스터 자동 스케일러 프로젝트를 기반으로 합니다. OpenShift Container Platform 구현에서는 머신 세트 API를 확장하여 Machine API와 통합됩니다. 코어, 노드, 메모리, GPU 등과 같은 리소스의 클러스터 전체에서 확장 제한을 설정할 수 있습니다. 중요도가 낮은 Pod에 대해 새 노드가 온라인 상태가 되지 않도록 클러스터가 Pod에 우선 순위를 설정할 수 있습니다. 노드를 확장할 수는 있지만 축소할 수 없도록 확장 정책을 설정할 수도 있습니다.

머신 상태 점검

MachineHealthCheck 리소스는 머신의 비정상적인 상태를 감지하여 삭제한 후 지원되는 플랫폼에서 새 머신을 생성합니다.

OpenShift Container Platform 버전 3.11에서는 클러스터가 머신 프로비저닝을 관리하지 않았기 때문에 다중 영역 아키텍처를 쉽게 롤아웃할 수 없었습니다. OpenShift Container Platform 버전 4.1부터 이러한 프로세스가 더 쉬워졌습니다. 각 머신 세트의 범위는 단일 영역에서 지정되므로 설치 프로그램은 사용자를 대신하여 가용성 영역 전체에 머신 세트를 보냅니다. 또한 계산이 동적이고 영역 장애가 발생하여 머신을 재조정해야하는 경우 처리할 수 있는 영역을 확보할 수 있습니다. Autoscaler는 클러스터의 수명 기간 동안 최적의 균형을 유지합니다.

2.2.2. Azure의 머신 세트사용자 지정 리소스에 대한 샘플 YAML

이 샘플 YAML은 리전의 **1** Microsoft Azure 영역에서 실행되는 머신 세트를 정의하고 **node-role.kubernetes.io/<role>: ""** 로 레이블이 지정된 노드를 만듭니다.

이 샘플에서 **<infrastructure_id>**는 클러스터를 프로비저닝할 때 설정한 클러스터 ID를 기반으로 하는 인프라 ID 레이블이며 **<role>**은 추가할 노드 레이블입니다.

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
    machine.openshift.io/cluster-api-machine-role: <role> 2
    machine.openshift.io/cluster-api-machine-type: <role> 3
  name: <infrastructure_id>-<role>-<region> 4
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<region> 6
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 7
        machine.openshift.io/cluster-api-machine-role: <role> 8
        machine.openshift.io/cluster-api-machine-type: <role> 9
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<region> 10
    spec:
      metadata:
        creationTimestamp: null
```

```
      labels:
        node-role.kubernetes.io/<role>: "" (11)
    providerSpec:
      value:
        apiVersion: azureproviderconfig.openshift.io/v1beta1
        credentialsSecret:
          name: azure-cloud-credentials
          namespace: openshift-machine-api
        image:
          offer: ""
          publisher: ""
          resourceID: /resourceGroups/<infrastructure_id>-
rg/providers/Microsoft.Compute/images/<infrastructure_id> (12)
          sku: ""
          version: ""
        internalLoadBalancer: ""
        kind: AzureMachineProviderSpec
        location: <region> (13)
        managedIdentity: <infrastructure_id>-identity (14)
        metadata:
          creationTimestamp: null
        natRule: null
        networkResourceGroup: ""
        osDisk:
          diskSizeGB: 128
          managedDisk:
            storageAccountType: Premium_LRS
          osType: Linux
        publicIP: false
        publicLoadBalancer: ""
        resourceGroup: <infrastructure_id>-rg (15)
        sshPrivateKey: ""
        sshPublicKey: ""
        subnet: <infrastructure_id>-<role>-subnet (16) (17)
        userDataSecret:
          name: worker-user-data (18)
        vmSize: Standard_DS4_v2
        vnet: <infrastructure_id>-vnet (19)
        zone: "1" (20)
```

(1) (5) (7) (12) (14) (15) (16) (19) 클러스터를 프로비저닝할 때 설정한 클러스터 ID를 기반으로 하는 인프라 ID를 지정합니다. OpenShift CLI 패키지가 설치되어 있으면 다음 명령을 실행하여 인프라 ID를 얻을 수 있습니다.

```
$ oc get -o jsonpath='{.status.infrastructureName}{"\n"}' infrastructure cluster
```

다음 명령을 실행하여 서브넷을 가져올 수 있습니다.

```
$  oc -n openshift-machine-api \
    -o jsonpath='{.spec.template.spec.providerSpec.value.subnet}{"\n"}' \
    get machineset/<infrastructure_id>-worker-centralus1
```

다음 명령을 실행하여 vnet을 가져올 수 있습니다.

```
$ oc -n openshift-machine-api \
    -o jsonpath='{.spec.template.spec.providerSpec.value.vnet}{"\n"}' \
    get machineset/<infrastructure_id>-worker-centralus1
```

2 3 8 9 11 17 18 추가할 노드 레이블을 지정합니다.

4 6 10 인프라 ID, 노드 레이블, 리전을 지정합니다.

13 머신을 배치할 리전을 지정합니다.

20 머신을 배치할 리전 내 영역을 지정합니다. 해당 리전이 지정한 영역을 지원하는지 확인합니다.

2.2.3. 머신 세트 만들기

설치 프로그램에 의해 생성되는 것 이외에도 고유한 머신 세트를 만들어 선택한 특정 워크로드의 머신 컴퓨팅 리소스를 동적으로 관리할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포합니다.
- OpenShift CLI(**oc**)를 설치합니다.
- **cluster-admin** 권한이 있는 사용자로 **oc**에 로그인합니다.

절차

1. 머신 세트 CR(사용자 지정 리소스) 샘플이 포함된 이름이 **<file_name>.yaml**인 새 YAML 파일을 만듭니다.
   <clusterID> 및 **<role>** 매개 변수 값을 설정해야 합니다.

   a. 특정 필드에 설정할 값이 확실하지 않은 경우 클러스터에서 기존 머신 세트를 확인할 수 있습니다.

   ```
   $ oc get machinesets -n openshift-machine-api
   ```

   출력 예

   ```
   NAME                           DESIRED   CURRENT   READY   AVAILABLE   AGE
   agl030519-vplxk-worker-us-east-1a   1         1         1       1           55m
   agl030519-vplxk-worker-us-east-1b   1         1         1       1           55m
   agl030519-vplxk-worker-us-east-1c   1         1         1       1           55m
   agl030519-vplxk-worker-us-east-1d   0         0                             55m
   agl030519-vplxk-worker-us-east-1e   0         0                             55m
   agl030519-vplxk-worker-us-east-1f   0         0                             55m
   ```

   b. 특정 머신 세트의 값을 확인합니다.

   ```
   $ oc get machineset <machineset_name> -n \
       openshift-machine-api -o yaml
   ```

   출력 예

   ```
   ```

```
...
template:
   metadata:
     labels:
       machine.openshift.io/cluster-api-cluster: agl030519-vplxk 1
       machine.openshift.io/cluster-api-machine-role: worker 2
       machine.openshift.io/cluster-api-machine-type: worker
       machine.openshift.io/cluster-api-machineset: agl030519-vplxk-worker-us-east-1a
```

1 클러스터 ID입니다.

2 기본 노드 레이블입니다.

2. 새 **MachineSet** CR을 만듭니다.

```
$ oc create -f <file_name>.yaml
```

3. 머신 세트 목록을 표시합니다.

```
$ oc get machineset -n openshift-machine-api
```

출력 예

```
NAME                                DESIRED   CURRENT   READY   AVAILABLE   AGE
agl030519-vplxk-infra-us-east-1a    1         1         1       1           11m
agl030519-vplxk-worker-us-east-1a   1         1         1       1           55m
agl030519-vplxk-worker-us-east-1b   1         1         1       1           55m
agl030519-vplxk-worker-us-east-1c   1         1         1       1           55m
agl030519-vplxk-worker-us-east-1d   0         0                             55m
agl030519-vplxk-worker-us-east-1e   0         0                             55m
agl030519-vplxk-worker-us-east-1f   0         0                             55m
```

새 머신 세트가 사용 가능한 경우**DESIRED** 및 **CURRENT** 값이 일치합니다. 머신 세트를 사용할 수 없는 경우 몇 분 후에 명령을 다시 실행합니다.

2.2.4. 머신을 Spot 가상머신으로 배포하는 머신 세트

Azure에서 실행되는 머신 세트를 생성하여 머신을 보장되지 않는 Spot 가상 머신으로 배포하면 비용을 절감할 수 있습니다. Spot 가상 머신은 사용되지 않은 Azure 용량을 사용하며 표준 가상 머신보다 비용이 저렴합니다. 일괄 처리 또는 상태 비저장, 수평적으로 확장 가능한 워크로드와 같이 인터럽트를 허용할 수 있는 워크로드에 Spot 가상 머신을 사용할 수 있습니다.

Azure는 언제든지 Spot 가상 머신을 종료 할 수 있습니다. Azure는 중단이 발생하면 사용자에게 30 초 동안 경고 메세지를 보냅니다. OpenShift Container Platform은 Azure가 종료 경고를 발행할 때 영향을 받는 인스턴스에서 워크로드를 제거하기 시작합니다.

다음과 같은 이유로 Spot 가상 머신을 사용할 때 중단될 수 있습니다.

- 인스턴스 가격이 최대 가격을 초과합니다.
- Spot 가상 머신의 공급이 감소합니다.
- Azure는 용량을 복원해야합니다.

Azure가 인스턴스를 종료하면 Spot 가상 머신 노드에서 실행되는 종료 프로세스가 머신 리소스를 삭제합니다. 머신 세트 **replicas** 수량을 충족하기 위해 머신 세트는 Spot 가상 머신을 요청하는 머신을 생성합니다.

2.2.5. 머신 세트를 사용하여 Spot 가상 머신 생성

머신 세트 YAML 파일에 **spotVMOptions**를 추가하여 Azure에서 Spot 가상 머신을 시작할 수 있습니다.

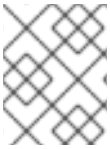
절차

- **providerSpec** 필드 아래에 다음 행을 추가합니다.

```
providerSpec:
  value:
    spotVMOptions: {}
```

선택 옵션으로 **spotVMOptions.maxPrice** 필드를 설정하여 Spot 가상 머신의 비용을 제한할 수 있습니다. 예를 들어 **maxPrice를 설정할 수 있습니다. '0.98765'**. **maxPrice**가 설정된 경우 이 값은 시간당 최대 Spot 가격으로 사용됩니다. 설정되지 않은 경우 최대 가격은 기본적으로 **-1**로 설정된 표준 가상 머신 가격까지 청구됩니다.

Azure는 Spot 가상 머신 가격을 표준 가격으로 제한합니다. 인스턴스가 기본 **maxPrice**로 설정된 경우 Azure는 가격 설정에 따라 인스턴스를 제거하지 않습니다. 그러나 용량 제한으로 인해 인스턴스를 제거할 수 있습니다.



참고

기본 표준 가상 머신 가격을 **maxPrice** 값으로 사용하고 Spot 가상 머신의 최대 가격을 설정하지 않는 것이 좋습니다.

2.2.6. 머신 세트의 고객 관리 암호화 키 활성화

Azure에 암호화 키를 제공하여 관리 대상 디스크의 데이터를 암호화할 수 있습니다. 시스템 API를 사용하여 고객 관리 키로 서버 측 암호화를 활성화할 수 있습니다.

고객 관리 키를 사용하려면 Azure Key Vault, 디스크 암호화 세트 및 암호화 키가 필요합니다. 디스크 암호화 세트는 CCO(Cloud Credential Operator)에 권한이 부여된 리소스 그룹 앞에 있어야 합니다. 그렇지 않은 경우 디스크 암호화 세트에 추가 reader 역할을 부여해야 합니다.

사전 요구 사항

- Azure Key Vault 인스턴스를 만듭니다.
- 디스크 암호화 세트의 인스턴스를 만듭니다.
- key vault에 디스크 암호화 세트 액세스 권한을 부여합니다 .

절차

- 머신 세트 YAML 파일의 **providerSpec** 필드에 설정된 디스크 암호화를 구성합니다. 예를 들면 다음과 같습니다.

```
...
providerSpec:
```

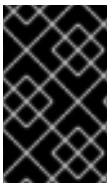
```
      value:
        ...
        osDisk:
          diskSizeGB: 128
          managedDisk:
            diskEncryptionSet:
              id:
/subscriptions/<subscription_id>/resourceGroups/<resource_group_name>/providers/Microsoft.
Compute/diskEncryptionSets/<disk_encryption_set_name>
            storageAccountType: Premium_LRS
...
```

추가 리소스

- Azure 문서에서 customer-managed 키에 대해 자세히 알아볼 수 있습니다.

2.3. GCP에서 머신 세트 생성

Google Cloud Platform (GCP)의 OpenShift Container Platform 클러스터에서 특정 목적을 충족하기 위해 다른 머신 세트를 생성할 수 있습니다. 예를 들어, 지원되는 워크로드를 새 머신으로 이동할 수 있도록 인프라 머신 세트 및 관련 머신을 작성할 수 있습니다.



중요

이 프로세스는 수동으로 프로비저닝된 시스템이 있는 클러스터에는 적용되지 않습니다. 머신 API가 작동하는 클러스터에서만 고급 머신 관리 및 스케일링 기능을 사용할 수 있습니다.

2.3.1. Machine API 개요

Machine API는 업스트림 Cluster API 프로젝트 및 사용자 정의 OpenShift Container Platform 리소스를 기반으로하는 주요 리소스의 조합입니다.

OpenShift Container Platform 4.6 클러스터의 경우 Machine API는 클러스터 설치가 완료된 후 모든 노드 호스트 프로비저닝 관리 작업을 수행합니다. 이 시스템으로 인해 OpenShift Container Platform 4.6은 퍼블릭 또는 프라이빗 클라우드 인프라에 더하여 탄력적이고 동적인 프로비저닝 방법을 제공합니다.

두 가지 주요 리소스는 다음과 같습니다.

Machine

노드의 호스트를 설명하는 기본 단위입니다. 머신에는 **providerSpec** 사양이 있으며 이는 다른 클라우드 플랫폼에 제공되는 컴퓨팅 노드 유형을 설명합니다. 예를 들어 AWS(Amazon Web Services)의 작업자 노드에 대한 머신 유형은 특정 머신 유형과 필요한 메타 데이터를 정의할 수 있습니다.

머신 세트

MachineSet 리소스는 머신 그룹입니다. 머신 세트는 머신에 연관되어 있고 복제본 세트는 pod에 연관되어 있습니다. 더 많은 머신이 필요하거나 규모를 줄여야 하는 경우 컴퓨터 요구 사항에 맞게 머신 세트의 **replicas** 필드를 변경합니다.

주의

컨트롤 플레인 시스템은 머신 세트에서 관리할 수 없습니다.

다음 사용자 지정 리소스는 클러스터에 더 많은 기능을 추가할 수 있습니다.

머신 자동 스케일러

MachineAutoscaler 리소스는 클라우드에서 머신을 자동으로 확장합니다. 지정된 머신 세트에서 노드의 최소 및 최대 스케일링 경계를 설정할 수 있으며 머신 자동 스케일러는 해당 노드 범위를 유지합니다. **MachineAutoscaler** 객체는 **ClusterAutoscaler** 객체를 설정한 후에 사용할 수 있습니다. **ClusterAutoscaler** 및 **MachineAutoscaler**리소스는 모두 **ClusterAutoscalerOperator** 오브젝트에서 사용 가능합니다.

Cluster autoscaler

이 리소스는 업스트림 클러스터 자동 스케일러 프로젝트를 기반으로 합니다. OpenShift Container Platform 구현에서는 머신 세트 API를 확장하여 Machine API와 통합됩니다. 코어, 노드, 메모리, GPU 등과 같은 리소스의 클러스터 전체에서 확장 제한을 설정할 수 있습니다. 중요도가 낮은 Pod에 대해 새 노드가 온라인 상태가 되지 않도록 클러스터가 Pod에 우선 순위를 설정할 수 있습니다. 노드를 확장할 수는 있지만 축소할 수 없도록 확장 정책을 설정할 수도 있습니다.

머신 상태 점검

MachineHealthCheck 리소스는 머신의 비정상적인 상태를 감지하여 삭제한 후 지원되는 플랫폼에서 새 머신을 생성합니다.

OpenShift Container Platform 버전 3.11에서는 클러스터가 머신 프로비저닝을 관리하지 않았기 때문에 다중 영역 아키텍처를 쉽게 롤아웃할 수 없었습니다. OpenShift Container Platform 버전 4.1부터 이러한 프로세스가 더 쉬워졌습니다. 각 머신 세트의 범위는 단일 영역에서 지정되므로 설치 프로그램은 사용자를 대신하여 가용성 영역 전체에 머신 세트를 보냅니다. 또한 계산이 동적이고 영역 장애가 발생하여 머신을 재조정해야하는 경우 처리할 수 있는 영역을 확보할 수 있습니다. Autoscaler는 클러스터의 수명 기간 동안 최적의 균형을 유지합니다.

2.3.2. GCP에서 머신 세트 사용자 정의 리소스의 샘플 YAML

이 샘플 YAML은 Google Cloud Platform (GCP)에서 실행되는 머신 세트를 정의하고 **node-role.kubernetes.io/<role>: ""**로 레이블이 지정된 노드를 만듭니다.

이 샘플에서 **<infrastructure_id>**는 클러스터를 프로비저닝할 때 설정한 클러스터 ID를 기반으로 하는 인프라 ID 레이블이며 **<role>**은 추가할 노드 레이블입니다.

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
  name: <infrastructure_id>-w-a 2
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
```

```
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 3
    machine.openshift.io/cluster-api-machineset: <infrastructure_id>-w-a 4
template:
  metadata:
    creationTimestamp: null
    labels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
      machine.openshift.io/cluster-api-machine-role: <role> 6
      machine.openshift.io/cluster-api-machine-type: <role> 7
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-w-a 8
  spec:
    metadata:
      labels:
        node-role.kubernetes.io/<role>: "" 9
    providerSpec:
      value:
        apiVersion: gcpprovider.openshift.io/v1beta1
        canIPForward: false
        credentialsSecret:
          name: gcp-cloud-credentials
        deletionProtection: false
        disks:
        - autoDelete: true
          boot: true
          image: <path_to_image> 10
          labels: null
          sizeGb: 128
          type: pd-ssd
        gcpMetadata: 11
        - key: <custom_metadata_key>
          value: <custom_metadata_value>
        kind: GCPMachineProviderSpec
        machineType: n1-standard-4
        metadata:
          creationTimestamp: null
        networkInterfaces:
        - network: <infrastructure_id>-network 12
          subnetwork: <infrastructure_id>-worker-subnet 13
        projectID: <project_name> 14
        region: us-central1
        serviceAccounts:
        - email: <infrastructure_id>-w@<project_name>.iam.gserviceaccount.com 15 16
          scopes:
          - https://www.googleapis.com/auth/cloud-platform
        tags:
        - <infrastructure_id>-worker 17
        userDataSecret:
          name: worker-user-data
        zone: us-central1-a
```

1 2 3 4 5 8 12 13 15 17 클러스터를 프로비저닝할 때 설정한 클러스터 ID를 기반으로 하는 인프라 ID를 지정합니다. OpenShift CLI 패키지가 설치되어 있으면 다음 명령을 실행하여 인프라 ID를 얻을 수 있습니다.

```
$ oc get -o jsonpath='{.status.infrastructureName}{"\n"}' infrastructure cluster
```

6 7 9 추가할 노드 레이블을 지정합니다.

10 현재 머신 세트에서 사용되는 이미지의 경로를 지정합니다. OpenShift CLI가 설치되어 있으면 다음 명령을 실행하여 이미지에 대한 경로를 얻을 수 있습니다.

```
$ oc -n openshift-machine-api \
    -o jsonpath='{.spec.template.spec.providerSpec.value.disks[0].image}{"\n"}' \
    get machineset/<infrastructure_id>-worker-a
```

11 선택 사항: **키:값** 쌍의 형식으로 사용자 정의 메타데이터를 지정합니다. 예를 들어 사용자 정의 메타데이터 설정에 대한 GCP 설명서를 참조하십시오.

14 16 클러스터에 사용하는 GCP 프로젝트의 이름을 지정합니다.

2.3.3. 머신 세트 만들기

설치 프로그램에 의해 생성되는 것 이외에도 고유한 머신 세트를 만들어 선택한 특정 워크로드의 머신 컴퓨팅 리소스를 동적으로 관리할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포합니다.
- OpenShift CLI(**oc**)를 설치합니다.
- **cluster-admin** 권한이 있는 사용자로 **oc**에 로그인합니다.

절차

1. 머신 세트 CR(사용자 지정 리소스) 샘플이 포함된 이름이 **<file_name>.yaml**인 새 YAML 파일을 만듭니다.
   <clusterID> 및 **<role>** 매개 변수 값을 설정해야 합니다.

   a. 특정 필드에 설정할 값이 확실하지 않은 경우 클러스터에서 기존 머신 세트를 확인할 수 있습니다.

   ```
   $ oc get machinesets -n openshift-machine-api
   ```

   출력 예

   ```
   NAME                                DESIRED   CURRENT   READY   AVAILABLE   AGE
   agl030519-vplxk-worker-us-east-1a   1         1         1       1           55m
   agl030519-vplxk-worker-us-east-1b   1         1         1       1           55m
   agl030519-vplxk-worker-us-east-1c   1         1         1       1           55m
   agl030519-vplxk-worker-us-east-1d   0         0                             55m
   agl030519-vplxk-worker-us-east-1e   0         0                             55m
   agl030519-vplxk-worker-us-east-1f   0         0                             55m
   ```

   b. 특정 머신 세트의 값을 확인합니다.

```
$ oc get machineset <machineset_name> -n \
    openshift-machine-api -o yaml
```

출력 예

```
...
template:
   metadata:
     labels:
       machine.openshift.io/cluster-api-cluster: agl030519-vplxk 1
       machine.openshift.io/cluster-api-machine-role: worker 2
       machine.openshift.io/cluster-api-machine-type: worker
       machine.openshift.io/cluster-api-machineset: agl030519-vplxk-worker-us-east-1a
```

1 클러스터 ID입니다.

2 기본 노드 레이블입니다.

2. 새 **MachineSet** CR을 만듭니다.

```
$ oc create -f <file_name>.yaml
```

3. 머신 세트 목록을 표시합니다.

```
$ oc get machineset -n openshift-machine-api
```

출력 예

```
NAME                              DESIRED   CURRENT   READY   AVAILABLE   AGE
agl030519-vplxk-infra-us-east-1a   1         1         1       1           11m
agl030519-vplxk-worker-us-east-1a  1         1         1       1           55m
agl030519-vplxk-worker-us-east-1b  1         1         1       1           55m
agl030519-vplxk-worker-us-east-1c  1         1         1       1           55m
agl030519-vplxk-worker-us-east-1d  0         0                             55m
agl030519-vplxk-worker-us-east-1e  0         0                             55m
agl030519-vplxk-worker-us-east-1f  0         0                             55m
```

새 머신 세트가 사용 가능한 경우**DESIRED** 및 **CURRENT** 값이 일치합니다. 머신 세트를 사용할 수 없는 경우 몇 분 후에 명령을 다시 실행합니다.

2.3.4. 머신을 선점할 수 있는 가상 머신 인스턴스로 배포하는 머신 세트

머신을 보장되지 않는 선점 가능한 가상 머신 인스턴스로 배포하는 GCP에서 실행되는 머신 세트를 만들어 비용을 절감할 수 있습니다. 선점 가능한 가상 머신 인스턴스는 과도한 Compute Engine 용량을 사용하며 일반 인스턴스보다 비용이 저렴합니다. 일괄 처리 또는 상태 비저장, 수평적으로 확장 가능한 워크로드와 같이 인터럽트를 허용할 수있는 워크로드에 선점 가능한 가상 머신 인스턴스를 사용할 수 있습니다.

GCP Compute Engine은 언제든지 선점 가능한 가상 머신 인스턴스를 종료할 수 있습니다. Compute Engine은 인터럽션이 30 초 후에 발생하는 것을 알리는 선점 알림을 사용자에게 보냅니다. OpenShift Container Platform은 Compute Engine이 선점 알림을 발행할 때 영향을 받는 인스턴스에서 워크로드를

제거하기 시작합니다. 인스턴스가 중지되지 않은 경우 ACPI G3 Mechanical Off 신호는 30 초 후에 운영 체제로 전송됩니다. 다음으로 선점 가능한 가상 머신 인스턴스가 Compute Engine에 의해 **TERMINATED** 상태로 전환됩니다.

다음과 같은 이유로 선점 가능한 가상 머신 인스턴스를 사용할 때 중단될 수 있습니다.

- 시스템 또는 유지 관리 이벤트가 있는 경우
- 선점 가능한 가상 머신 인스턴스의 공급이 감소하는 경우
- 인스턴스가 선점 가능한 가상 머신 인스턴스에 할당된 24 시간 후에 종료되는 경우

GCP가 인스턴스를 종료하면 선점 가능한 가상 머신 인스턴스 노드에서 실행되는 종료 프로세스가 머신 리소스를 삭제합니다. 머신 세트 **replicas** 수량을 충족하기 위해 머신 세트는 선점 가능한 가상 머신 인스턴스를 요청하는 머신을 생성합니다.

2.3.5. 머신 세트를 사용하여 선점 가능한 가상 머신 인스턴스 생성

머신 세트 YAML 파일에 **preemptible**을 추가하여 GCP에서 선점 가능한 가상 머신 인스턴스를 시작할 수 있습니다.

절차

- **providerSpec** 필드 아래에 다음 행을 추가합니다.

```
providerSpec:
  value:
    preemptible: true
```

preemptible 을 **true** 로 설정하면 인스턴스가 시작된 후 머신에 **interruptable-instance** 로 레이블이 지정됩니다.

2.4. OPENSTACK에서 머신 세트 생성

Red Hat OpenStack Platform (RHOSP)의 OpenShift Container Platform 클러스터에서 특정 목적을 제공하기 위해 다른 머신 세트를 생성할 수 있습니다. 예를 들어, 지원되는 워크로드를 새 머신으로 이동할 수 있도록 인프라 머신 세트 및 관련 머신을 작성할 수 있습니다.

중요

이 프로세스는 수동으로 프로비저닝된 시스템이 있는 클러스터에는 적용되지 않습니다. 머신 API가 작동하는 클러스터에서만 고급 머신 관리 및 스케일링 기능을 사용할 수 있습니다.

2.4.1. Machine API 개요

Machine API는 업스트림 Cluster API 프로젝트 및 사용자 정의 OpenShift Container Platform 리소스를 기반으로하는 주요 리소스의 조합입니다.

OpenShift Container Platform 4.6 클러스터의 경우 Machine API는 클러스터 설치가 완료된 후 모든 노드 호스트 프로비저닝 관리 작업을 수행합니다. 이 시스템으로 인해 OpenShift Container Platform 4.6은 퍼블릭 또는 프라이빗 클라우드 인프라에 더하여 탄력적이고 동적인 프로비저닝 방법을 제공합니다.

두 가지 주요 리소스는 다음과 같습니다.

Machine

노드의 호스트를 설명하는 기본 단위입니다. 머신에는 **providerSpec** 사양이 있으며 이는 다른 클라우드 플랫폼에 제공되는 컴퓨팅 노드 유형을 설명합니다. 예를 들어 AWS(Amazon Web Services)의 작업자 노드에 대한 머신 유형은 특정 머신 유형과 필요한 메타 데이터를 정의할 수 있습니다.

머신 세트

MachineSet 리소스는 머신 그룹입니다. 머신 세트는 머신에 연관되어 있고 복제본 세트는 pod에 연관되어 있습니다. 더 많은 머신이 필요하거나 규모를 줄여야 하는 경우 컴퓨터 요구 사항에 맞게 머신 세트의 **replicas** 필드를 변경합니다.

주의

컨트롤 플레인 시스템은 머신 세트에서 관리할 수 없습니다.

다음 사용자 지정 리소스는 클러스터에 더 많은 기능을 추가할 수 있습니다.

머신 자동 스케일러

MachineAutoscaler 리소스는 클라우드에서 머신을 자동으로 확장합니다. 지정된 머신 세트에서 노드의 최소 및 최대 스케일링 경계를 설정할 수 있으며 머신 자동 스케일러는 해당 노드 범위를 유지합니다. **MachineAutoscaler** 객체는 **ClusterAutoscaler** 객체를 설정한 후에 사용할 수 있습니다. **ClusterAutoscaler** 및 **MachineAutoscaler**리소스는 모두 **ClusterAutoscalerOperator** 오브젝트에서 사용 가능합니다.

Cluster autoscaler

이 리소스는 업스트림 클러스터 자동 스케일러 프로젝트를 기반으로 합니다. OpenShift Container Platform 구현에서는 머신 세트 API를 확장하여 Machine API와 통합됩니다. 코어, 노드, 메모리, GPU 등과 같은 리소스의 클러스터 전체에서 확장 제한을 설정할 수 있습니다. 중요도가 낮은 Pod에 대해 새 노드가 온라인 상태가 되지 않도록 클러스터가 Pod에 우선 순위를 설정할 수 있습니다. 노드를 확장할 수는 있지만 축소할 수 없도록 확장 정책을 설정할 수도 있습니다.

머신 상태 점검

MachineHealthCheck 리소스는 머신의 비정상적인 상태를 감지하여 삭제한 후 지원되는 플랫폼에서 새 머신을 생성합니다.

OpenShift Container Platform 버전 3.11에서는 클러스터가 머신 프로비저닝을 관리하지 않았기 때문에 다중 영역 아키텍처를 쉽게 롤아웃할 수 없었습니다. OpenShift Container Platform 버전 4.1부터 이러한 프로세스가 더 쉬워졌습니다. 각 머신 세트의 범위는 단일 영역에서 지정되므로 설치 프로그램은 사용자를 대신하여 가용성 영역 전체에 머신 세트를 보냅니다. 또한 계산이 동적이고 영역 장애가 발생하여 머신을 재조정해야하는 경우 처리할 수 있는 영역을 확보할 수 있습니다. Autoscaler는 클러스터의 수명 기간 동안 최적의 균형을 유지합니다.

2.4.2. RHOSP에서 머신 세트 사용자 정의 리소스의 샘플 YAML

이 샘플 YAML은 RHOSP(Red Hat OpenStack Platform)에서 실행되는 머신 세트를 정의하고 **node-role.kubernetes.io/<role>: ""** 로 레이블이 지정된 노드를 생성합니다.

이 샘플에서 **<infrastructure_id>**는 클러스터를 프로비저닝할 때 설정한 클러스터 ID를 기반으로 하는 인프라 ID 레이블이며 **<role>**은 추가할 노드 레이블입니다.

```
apiVersion: machine.openshift.io/v1beta1
```

```
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
    machine.openshift.io/cluster-api-machine-role: <role> 2
    machine.openshift.io/cluster-api-machine-type: <role> 3
  name: <infrastructure_id>-<role> 4
  namespace: openshift-machine-api
spec:
  replicas: <number_of_replicas>
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role> 6
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 7
        machine.openshift.io/cluster-api-machine-role: <role> 8
        machine.openshift.io/cluster-api-machine-type: <role> 9
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role> 10
    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: <optional_UUID_of_server_group> 11
          kind: OpenstackProviderSpec
          networks: 12
          - filter: {}
            subnets:
            - filter:
                name: <subnet_name>
                tags: openshiftClusterID=<infrastructure_id> 13
          primarySubnet: <rhosp_subnet_UUID> 14
          securityGroups:
          - filter: {}
            name: <infrastructure_id>-worker 15
          serverMetadata:
            Name: <infrastructure_id>-worker 16
            openshiftClusterID: <infrastructure_id> 17
          tags:
          - openshiftClusterID=<infrastructure_id> 18
          trunk: true
          userDataSecret:
            name: worker-user-data 19
          availabilityZone: <optional_openstack_availability_zone>
```

1 5 7 13 15 16 17 18 클러스터를 프로비저닝할 때 설정한 클러스터 ID를 기반으로 하는 인프라 ID를 지정합니다. OpenShift CLI 패키지가 설치되어 있으면 다음 명령을 실행하여 인

```
$ oc get -o jsonpath='{.status.infrastructureName}{"\n"}' infrastructure cluster
```

2 3 8 9 19 추가할 노드 레이블을 지정합니다.

4 6 10 인프라 ID 및 노드 레이블을 지정합니다.

11 MachineSet의 서버 그룹 정책을 설정하려면, 서버 그룹 생성에서 반환된 값을 입력합니다. 대부분의 배포에는 **anti-affinity** 또는 **soft-anti-affinity** 정책이 권장됩니다.

12 여러 네트워크에 배포해야 합니다. 여러 네트워크를 지정하려면 네트워크 배열에 다른 항목을 추가합니다. 또한 **primarySubnet** 값으로 사용되는 네트워크를 포함해야 합니다.

14 노드 엔드포인트를 게시할 RHOSP 서브넷을 지정합니다. 일반적으로 이 서브넷은 **install-config.yaml** 파일에서 **machineSubnet** 값으로 사용되는 서브넷과 동일합니다.

2.4.3. 머신 세트 만들기

설치 프로그램에 의해 생성되는 것 이외에도 고유한 머신 세트를 만들어 선택한 특정 워크로드의 머신 컴퓨팅 리소스를 동적으로 관리할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포합니다.
- OpenShift CLI(**oc**)를 설치합니다.
- **cluster-admin** 권한이 있는 사용자로 **oc**에 로그인합니다.

절차

1. 머신 세트 CR(사용자 지정 리소스) 샘플이 포함된 이름이 **<file_name>.yaml**인 새 YAML 파일을 만듭니다.
   <clusterID> 및 **<role>** 매개 변수 값을 설정해야 합니다.

   a. 특정 필드에 설정할 값이 확실하지 않은 경우 클러스터에서 기존 머신 세트를 확인할 수 있습니다.

   ```
   $ oc get machinesets -n openshift-machine-api
   ```

   출력 예

   ```
   NAME                                DESIRED   CURRENT   READY   AVAILABLE   AGE
   agl030519-vplxk-worker-us-east-1a   1         1         1       1           55m
   agl030519-vplxk-worker-us-east-1b   1         1         1       1           55m
   agl030519-vplxk-worker-us-east-1c   1         1         1       1           55m
   agl030519-vplxk-worker-us-east-1d   0         0                             55m
   agl030519-vplxk-worker-us-east-1e   0         0                             55m
   agl030519-vplxk-worker-us-east-1f   0         0                             55m
   ```

   b. 특정 머신 세트의 값을 확인합니다.

```
$ oc get machineset <machineset_name> -n \
    openshift-machine-api -o yaml
```

출력 예

```
...
template:
   metadata:
     labels:
       machine.openshift.io/cluster-api-cluster: agl030519-vplxk 1
       machine.openshift.io/cluster-api-machine-role: worker 2
       machine.openshift.io/cluster-api-machine-type: worker
       machine.openshift.io/cluster-api-machineset: agl030519-vplxk-worker-us-east-1a
```

1. 클러스터 ID입니다.
2. 기본 노드 레이블입니다.

2. 새 **MachineSet** CR을 만듭니다.

```
$ oc create -f <file_name>.yaml
```

3. 머신 세트 목록을 표시합니다.

```
$ oc get machineset -n openshift-machine-api
```

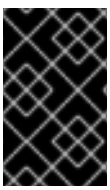
출력 예

```
NAME                                DESIRED   CURRENT   READY   AVAILABLE   AGE
agl030519-vplxk-infra-us-east-1a    1         1         1       1           11m
agl030519-vplxk-worker-us-east-1a   1         1         1       1           55m
agl030519-vplxk-worker-us-east-1b   1         1         1       1           55m
agl030519-vplxk-worker-us-east-1c   1         1         1       1           55m
agl030519-vplxk-worker-us-east-1d   0         0                             55m
agl030519-vplxk-worker-us-east-1e   0         0                             55m
agl030519-vplxk-worker-us-east-1f   0         0                             55m
```

새 머신 세트가 사용 가능한 경우**DESIRED** 및 **CURRENT** 값이 일치합니다. 머신 세트를 사용할 수 없는 경우 몇 분 후에 명령을 다시 실행합니다.

2.5. RHV에 머신 세트 생성

Red Hat Virtualization (RHV)의 OpenShift Container Platform 클러스터에서 특정 목적을 제공하기 위해 다른 머신 세트를 생성할 수 있습니다. 예를 들어, 지원되는 워크로드를 새 머신으로 이동할 수 있도록 인프라 머신 세트 및 관련 머신을 작성할 수 있습니다.



중요

이 프로세스는 수동으로 프로비저닝된 시스템이 있는 클러스터에는 적용되지 않습니다. 머신 API가 작동하는 클러스터에서만 고급 머신 관리 및 스케일링 기능을 사용할 수 있습니다.

2.5.1. Machine API 개요

Machine API는 업스트림 Cluster API 프로젝트 및 사용자 정의 OpenShift Container Platform 리소스를 기반으로하는 주요 리소스의 조합입니다.

OpenShift Container Platform 4.6 클러스터의 경우 Machine API는 클러스터 설치가 완료된 후 모든 노드 호스트 프로비저닝 관리 작업을 수행합니다. 이 시스템으로 인해 OpenShift Container Platform 4.6은 퍼블릭 또는 프라이빗 클라우드 인프라에 더하여 탄력적이고 동적인 프로비저닝 방법을 제공합니다.

두 가지 주요 리소스는 다음과 같습니다.

Machine

노드의 호스트를 설명하는 기본 단위입니다. 머신에는 **providerSpec** 사양이 있으며 이는 다른 클라우드 플랫폼에 제공되는 컴퓨팅 노드 유형을 설명합니다. 예를 들어 AWS(Amazon Web Services)의 작업자 노드에 대한 머신 유형은 특정 머신 유형과 필요한 메타 데이터를 정의할 수 있습니다.

머신 세트

MachineSet 리소스는 머신 그룹입니다. 머신 세트는 머신에 연관되어 있고 복제본 세트는 pod에 연관되어 있습니다. 더 많은 머신이 필요하거나 규모를 줄여야 하는 경우 컴퓨터 요구 사항에 맞게 머신 세트의 **replicas** 필드를 변경합니다.

주의

컨트롤 플레인 시스템은 머신 세트에서 관리할 수 없습니다.

다음 사용자 지정 리소스는 클러스터에 더 많은 기능을 추가할 수 있습니다.

머신 자동 스케일러

MachineAutoscaler 리소스는 클라우드에서 머신을 자동으로 확장합니다. 지정된 머신 세트에서 노드의 최소 및 최대 스케일링 경계를 설정할 수 있으며 머신 자동 스케일러는 해당 노드 범위를 유지합니다. **MachineAutoscaler** 객체는 **ClusterAutoscaler** 객체를 설정한 후에 사용할 수 있습니다. **ClusterAutoscaler** 및 **MachineAutoscaler**리소스는 모두 **ClusterAutoscalerOperator** 오브젝트에서 사용 가능합니다.

Cluster autoscaler

이 리소스는 업스트림 클러스터 자동 스케일러 프로젝트를 기반으로 합니다. OpenShift Container Platform 구현에서는 머신 세트 API를 확장하여 Machine API와 통합됩니다. 코어, 노드, 메모리, GPU 등과 같은 리소스의 클러스터 전체에서 확장 제한을 설정할 수 있습니다. 중요도가 낮은 Pod에 대해 새 노드가 온라인 상태가 되지 않도록 클러스터가 Pod에 우선 순위를 설정할 수 있습니다. 노드를 확장할 수는 있지만 축소할 수 없도록 확장 정책을 설정할 수도 있습니다.

머신 상태 점검

MachineHealthCheck 리소스는 머신의 비정상적인 상태를 감지하여 삭제한 후 지원되는 플랫폼에서 새 머신을 생성합니다.

OpenShift Container Platform 버전 3.11에서는 클러스터가 머신 프로비저닝을 관리하지 않았기 때문에 다중 영역 아키텍처를 쉽게 롤아웃할 수 없었습니다. OpenShift Container Platform 버전 4.1부터 이러한 프로세스가 더 쉬워졌습니다. 각 머신 세트의 범위는 단일 영역에서 지정되므로 설치 프로그램은 사용자를

대신하여 가용성 영역 전체에 머신 세트를 보냅니다. 또한 계산이 동적이고 영역 장애가 발생하여 머신을 재조정해야하는 경우 처리할 수 있는 영역을 확보할 수 있습니다. Autoscaler는 클러스터의 수명 기간 동안 최적의 균형을 유지합니다.

2.5.2. RHV에서 머신 세트 사용자 정의 리소스의 샘플 YAML

이 샘플 YAML은 RHV에서 실행되는 머신 세트를 정의하고 **node-role.kubernetes.io/<role>: ""**로 레이블이 지정된 노드를 만듭니다.

이 샘플에서 **<infrastructure_id>**는 클러스터를 프로비저닝할 때 설정한 클러스터 ID를 기반으로 하는 인프라 ID 레이블이며 **<role>**은 추가할 노드 레이블입니다.

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
    machine.openshift.io/cluster-api-machine-role: <role> 2
    machine.openshift.io/cluster-api-machine-type: <role> 3
  name: <infrastructure_id>-<role> 4
  namespace: openshift-machine-api
spec:
  replicas: <number_of_replicas> 5
  selector: 6
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 7
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role> 8
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 9
        machine.openshift.io/cluster-api-machine-role: <role> 10
        machine.openshift.io/cluster-api-machine-type: <role> 11
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role> 12
    spec:
      metadata:
        labels:
          node-role.kubernetes.io/<role>: "" 13
      providerSpec:
        value:
          apiVersion: ovirtproviderconfig.machine.openshift.io/v1beta1
          cluster_id: <ovirt_cluster_id> 14
          template_name: <ovirt_template_name> 15
          instance_type_id: <instance_type_id> 16
          cpu: 17
            sockets: <number_of_sockets> 18
            cores: <number_of_cores> 19
            threads: <number_of_threads> 20
          memory_mb: <memory_size> 21
          os_disk: 22
            size_gb: <disk_size> 23
          network_interfaces: 24
```

```
      vnic_profile_id:  <vnic_profile_id> 25
    credentialsSecret:
      name: ovirt-credentials 26
    kind: OvirtMachineProviderSpec
    type: <workload_type> 27
    userDataSecret:
      name: worker-user-data
```

1 7 9 클러스터를 프로비저닝할 때 설정한 클러스터 ID를 기반으로하는 인프라 ID를 지정합니다. OpenShift CLI (**oc**) 패키지가 설치되어 있으면 다음 명령을 실행하여 인프라 ID를 얻을 수 있습니다.

```
$ oc get -o jsonpath='{.status.infrastructureName}{"\n"}' infrastructure cluster
```

2 3 10 11 13 추가할 노드 레이블을 지정합니다.

4 8 12 인프라 ID 및 노드 레이블을 지정합니다. 이 두 문자열은 35자를 초과할 수 없습니다.

5 생성할 머신 수를 지정합니다.

6 머신의 선택기입니다.

14 이 VM 인스턴스가 속하는 RHV 클러스터의 UUID를 지정합니다.

15 머신을 생성하는 데 사용할 RHV VM 템플릿을 지정합니다.

16 선택 사항: VM 인스턴스 유형을 지정합니다. 이 매개변수를 포함하는 경우 이 매개변수는 모든 하드웨어 매개변수를 재정의하므로 CPU 및 메모리를 포함한 VM의 하드웨어 매개변수를 지정할 필요가 없습니다.

17 선택 사항: CPU 필드에는 소켓, 코어, 스레드를 포함한 CPU 구성이 포함됩니다.

18 선택 사항: VM의 소켓 수를 지정합니다.

19 선택 사항: 소켓당 코어 수를 지정합니다.

20 선택 사항: 코어당 스레드 수를 지정합니다.

21 선택 사항: VM의 메모리 크기를 MiB 단위로 지정합니다.

22 선택 사항: 노드의 루트 디스크.

23 선택 사항: 부팅 가능한 디스크의 크기를 GiB로 지정합니다.

24 선택 사항: VM의 네트워크 인터페이스 목록입니다. 이 매개변수를 포함하는 경우 OpenShift Container Platform은 템플릿에서 모든 네트워크 인터페이스를 삭제하고 새 네트워크 인터페이스를 생성합니다.

25 선택 사항: vNIC 프로필 ID를 지정합니다.

26 RHV 인증 정보를 보유한 시크릿 이름을 지정합니다.

27 선택 사항: 인스턴스가 최적화된 워크로드 유형을 지정합니다. 이 값은 **RHV VM** 매개변수에 영향을 미칩니다. 지원되는 값은 **desktop**, **server**, **high_performance**입니다.

참고

RHV는 VM을 생성할 때 템플릿을 사용하므로 선택적 매개변수에 대한 값을 지정하지 않으면 RHV는 템플릿에 지정된 해당 매개변수의 값을 사용합니다.

2.5.3. 머신 세트 만들기

설치 프로그램에 의해 생성되는 것 이외에도 고유한 머신 세트를 만들어 선택한 특정 워크로드의 머신 컴퓨팅 리소스를 동적으로 관리할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포합니다.
- OpenShift CLI(**oc**)를 설치합니다.
- **cluster-admin** 권한이 있는 사용자로 **oc**에 로그인합니다.

절차

1. 머신 세트 CR(사용자 지정 리소스) 샘플이 포함된 이름이 **<file_name>.yaml**인 새 YAML 파일을 만듭니다.
   <clusterID> 및 **<role>** 매개 변수 값을 설정해야 합니다.

   a. 특정 필드에 설정할 값이 확실하지 않은 경우 클러스터에서 기존 머신 세트를 확인할 수 있습니다.

   ```
   $ oc get machinesets -n openshift-machine-api
   ```

   출력 예

   ```
   NAME                              DESIRED   CURRENT   READY   AVAILABLE   AGE
   agl030519-vplxk-worker-us-east-1a   1         1         1       1           55m
   agl030519-vplxk-worker-us-east-1b   1         1         1       1           55m
   agl030519-vplxk-worker-us-east-1c   1         1         1       1           55m
   agl030519-vplxk-worker-us-east-1d   0         0                             55m
   agl030519-vplxk-worker-us-east-1e   0         0                             55m
   agl030519-vplxk-worker-us-east-1f   0         0                             55m
   ```

   b. 특정 머신 세트의 값을 확인합니다.

   ```
   $ oc get machineset <machineset_name> -n \
     openshift-machine-api -o yaml
   ```

   출력 예

   ```
   ...
   template:
     metadata:
       labels:
         machine.openshift.io/cluster-api-cluster: agl030519-vplxk (1)
   ```

```
      machine.openshift.io/cluster-api-machine-role: worker 2
      machine.openshift.io/cluster-api-machine-type: worker
      machine.openshift.io/cluster-api-machineset: agl030519-vplxk-worker-us-east-1a
```

1 클러스터 ID입니다.

2 기본 노드 레이블입니다.

2. 새 **MachineSet** CR을 만듭니다.

```
$ oc create -f <file_name>.yaml
```

3. 머신 세트 목록을 표시합니다.

```
$ oc get machineset -n openshift-machine-api
```

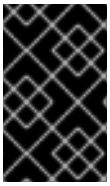
출력 예

```
NAME                              DESIRED   CURRENT   READY   AVAILABLE   AGE
agl030519-vplxk-infra-us-east-1a   1         1         1       1           11m
agl030519-vplxk-worker-us-east-1a  1         1         1       1           55m
agl030519-vplxk-worker-us-east-1b  1         1         1       1           55m
agl030519-vplxk-worker-us-east-1c  1         1         1       1           55m
agl030519-vplxk-worker-us-east-1d  0         0                             55m
agl030519-vplxk-worker-us-east-1e  0         0                             55m
agl030519-vplxk-worker-us-east-1f  0         0                             55m
```

새 머신 세트가 사용 가능한 경우**DESIRED** 및 **CURRENT** 값이 일치합니다. 머신 세트를 사용할 수 없는 경우 몇 분 후에 명령을 다시 실행합니다.

2.6. VSPHERE에서 머신 세트 생성

VMware vSphere의 OpenShift Container Platform 클러스터에서 특정 목적을 충족하기 위해 다른 머신 세트를 만들 수 있습니다. 예를 들어, 지원되는 워크로드를 새 머신으로 이동할 수 있도록 인프라 머신 세트 및 관련 머신을 작성할 수 있습니다.



중요

이 프로세스는 수동으로 프로비저닝된 시스템이 있는 클러스터에는 적용되지 않습니다. 머신 API가 작동하는 클러스터에서만 고급 머신 관리 및 스케일링 기능을 사용할 수 있습니다.

2.6.1. Machine API 개요

Machine API는 업스트림 Cluster API 프로젝트 및 사용자 정의 OpenShift Container Platform 리소스를 기반으로하는 주요 리소스의 조합입니다.

OpenShift Container Platform 4.6 클러스터의 경우 Machine API는 클러스터 설치가 완료된 후 모든 노드 호스트 프로비저닝 관리 작업을 수행합니다. 이 시스템으로 인해 OpenShift Container Platform 4.6은 퍼블릭 또는 프라이빗 클라우드 인프라에 더하여 탄력적이고 동적인 프로비저닝 방법을 제공합니다.

두 가지 주요 리소스는 다음과 같습니다.

Machine

노드의 호스트를 설명하는 기본 단위입니다. 머신에는 **providerSpec** 사양이 있으며 이는 다른 클라우드 플랫폼에 제공되는 컴퓨팅 노드 유형을 설명합니다. 예를 들어 AWS(Amazon Web Services)의 작업자 노드에 대한 머신 유형은 특정 머신 유형과 필요한 메타 데이터를 정의할 수 있습니다.

머신 세트

MachineSet 리소스는 머신 그룹입니다. 머신 세트는 머신에 연관되어 있고 복제본 세트는 pod에 연관되어 있습니다. 더 많은 머신이 필요하거나 규모를 줄여야 하는 경우 컴퓨터 요구 사항에 맞게 머신 세트의 **replicas** 필드를 변경합니다.

주의

컨트롤 플레인 시스템은 머신 세트에서 관리할 수 없습니다.

다음 사용자 지정 리소스는 클러스터에 더 많은 기능을 추가할 수 있습니다.

머신 자동 스케일러

MachineAutoscaler 리소스는 클라우드에서 머신을 자동으로 확장합니다. 지정된 머신 세트에서 노드의 최소 및 최대 스케일링 경계를 설정할 수 있으며 머신 자동 스케일러는 해당 노드 범위를 유지합니다. **MachineAutoscaler** 객체는 **ClusterAutoscaler** 객체를 설정한 후에 사용할 수 있습니다. **ClusterAutoscaler** 및 **MachineAutoscaler**리소스는 모두 **ClusterAutoscalerOperator** 오브젝트에서 사용 가능합니다.

Cluster autoscaler

이 리소스는 업스트림 클러스터 자동 스케일러 프로젝트를 기반으로 합니다. OpenShift Container Platform 구현에서는 머신 세트 API를 확장하여 Machine API와 통합됩니다. 코어, 노드, 메모리, GPU 등과 같은 리소스의 클러스터 전체에서 확장 제한을 설정할 수 있습니다. 중요도가 낮은 Pod에 대해 새 노드가 온라인 상태가 되지 않도록 클러스터가 Pod에 우선 순위를 설정할 수 있습니다. 노드를 확장할 수는 있지만 축소할 수 없도록 확장 정책을 설정할 수도 있습니다.

머신 상태 점검

MachineHealthCheck 리소스는 머신의 비정상적인 상태를 감지하여 삭제한 후 지원되는 플랫폼에서 새 머신을 생성합니다.

OpenShift Container Platform 버전 3.11에서는 클러스터가 머신 프로비저닝을 관리하지 않았기 때문에 다중 영역 아키텍처를 쉽게 롤아웃할 수 없었습니다. OpenShift Container Platform 버전 4.1부터 이러한 프로세스가 더 쉬워졌습니다. 각 머신 세트의 범위는 단일 영역에서 지정되므로 설치 프로그램은 사용자를 대신하여 가용성 영역 전체에 머신 세트를 보냅니다. 또한 계산이 동적이고 영역 장애가 발생하여 머신을 재조정해야하는 경우 처리할 수 있는 영역을 확보할 수 있습니다. Autoscaler는 클러스터의 수명 기간 동안 최적의 균형을 유지합니다.

2.6.2. vSphere에서 머신 세트 사용자 정의 리소스의 샘플 YAML

이 샘플 YAML은 VMware vSphere에서 실행되는 머신 세트를 정의하고 **node-role.kubernetes.io/<role>: ""**로 레이블이 지정된 노드를 만듭니다.

이 샘플에서 **<infrastructure_id>**는 클러스터를 프로비저닝할 때 설정한 클러스터 ID를 기반으로 하는 인프라 ID 레이블이며 **<role>**은 추가할 노드 레이블입니다.

```
apiVersion: machine.openshift.io/v1beta1
```

```
kind: MachineSet
metadata:
  creationTimestamp: null
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
  name: <infrastructure_id>-<role> 2
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 3
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role> 4
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
        machine.openshift.io/cluster-api-machine-role: <role> 6
        machine.openshift.io/cluster-api-machine-type: <role> 7
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role> 8
    spec:
      metadata:
        creationTimestamp: null
        labels:
          node-role.kubernetes.io/<role>: "" 9
      providerSpec:
        value:
          apiVersion: vsphereprovider.openshift.io/v1beta1
          credentialsSecret:
            name: vsphere-cloud-credentials
          diskGiB: 120
          kind: VSphereMachineProviderSpec
          memoryMiB: 8192
          metadata:
            creationTimestamp: null
          network:
            devices:
            - networkName: "<vm_network_name>" 10
          numCPUs: 4
          numCoresPerSocket: 1
          snapshot: ""
          template: <vm_template_name> 11
          userDataSecret:
            name: worker-user-data
          workspace:
            datacenter: <vcenter_datacenter_name> 12
            datastore: <vcenter_datastore_name> 13
            folder: <vcenter_vm_folder_path> 14
            resourcepool: <vsphere_resource_pool> 15
            server: <vcenter_server_ip> 16
```

1 3 5 클러스터를 프로비저닝할 때 설정한 클러스터 ID를 기반으로하는 인프라 ID를 지정합니다. OpenShift CLI (**oc**) 패키지가 설치되어 있으면 다음 명령을 실행하여 인프라 ID를 얻을 수 있습

```
$ oc get -o jsonpath='{.status.infrastructureName}{"\n"}' infrastructure cluster
```

2 4 8 인프라 ID 및 노드 레이블을 지정합니다.

6 7 9 추가할 노드 레이블을 지정합니다.

10 머신 세트를 배포할 vSphere VM 네트워크를 지정합니다.

11 사용할 템플릿의 vSphere VM 복제(예: **user-5ddjd-rhcos**)를 지정합니다.

중요

원래 VM 템플릿을 지정하지 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.

12 머신 세트를 배포할 vCenter Datacenter를 지정합니다.

13 머신 세트를 배포할 vCenter Datastore를 지정합니다.

14 vCenter의 vSphere VM 폴더에 경로(예: **/dc1/vm/user-inst-5ddjd**)를 지정합니다.

15 VM의 vSphere 리소스 풀을 지정합니다.

16 vCenter 서버 IP 또는 정규화된 도메인 이름을 지정합니다.

2.6.3. 머신 세트 만들기

설치 프로그램에 의해 생성되는 것 이외에도 고유한 머신 세트를 만들어 선택한 특정 워크로드의 머신 컴퓨팅 리소스를 동적으로 관리할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포합니다.
- OpenShift CLI(**oc**)를 설치합니다.
- **cluster-admin** 권한이 있는 사용자로 **oc**에 로그인합니다.
- 클러스터 API 이름을 기반으로 vCenter 인스턴스 내에 태그를 생성합니다. 이 태그는 머신 세트에서 OpenShift Container Platform 노드를 프로비저닝된 가상 머신 (VM)에 연결하는 데 사용됩니다. vCenter에서 태그 생성 방법에 대한 자세한 내용은 vSphere 태그 및 속성 에 대한 VMware 설명서를 참조하십시오.
- vCenter 인스턴스에 가상 머신을 배포하는데 필요한 권한이 있고 지정된 데이터 저장소에 필요한 액세스 권한이 있습니다.

절차

1. 머신 세트 CR(사용자 지정 리소스) 샘플이 포함된 이름이 **<file_name>.yaml**인 새 YAML 파일을 만듭니다.

<clusterID> 및 **<role>** 매개 변수 값을 설정해야 합니다.

a. 특정 필드에 설정할 값이 확실하지 않은 경우 클러스터에서 기존 머신 세트를 확인할 수 있습니다.

```
$ oc get machinesets -n openshift-machine-api
```

출력 예

```
NAME                                DESIRED   CURRENT   READY   AVAILABLE   AGE
agl030519-vplxk-worker-us-east-1a   1         1         1       1           55m
agl030519-vplxk-worker-us-east-1b   1         1         1       1           55m
agl030519-vplxk-worker-us-east-1c   1         1         1       1           55m
agl030519-vplxk-worker-us-east-1d   0         0                             55m
agl030519-vplxk-worker-us-east-1e   0         0                             55m
agl030519-vplxk-worker-us-east-1f   0         0                             55m
```

b. 특정 머신 세트의 값을 확인합니다.

```
$ oc get machineset <machineset_name> -n \
    openshift-machine-api -o yaml
```

출력 예

```
...
template:
   metadata:
     labels:
       machine.openshift.io/cluster-api-cluster: agl030519-vplxk 1
       machine.openshift.io/cluster-api-machine-role: worker 2
       machine.openshift.io/cluster-api-machine-type: worker
       machine.openshift.io/cluster-api-machineset: agl030519-vplxk-worker-us-east-1a
```

1 클러스터 ID입니다.

2 기본 노드 레이블입니다.

2. 새 **MachineSet** CR을 만듭니다.

```
$ oc create -f <file_name>.yaml
```

3. 머신 세트 목록을 표시합니다.

```
$ oc get machineset -n openshift-machine-api
```

출력 예

```
NAME                                DESIRED   CURRENT   READY   AVAILABLE   AGE
agl030519-vplxk-infra-us-east-1a    1         1         1       1           11m
agl030519-vplxk-worker-us-east-1a   1         1         1       1           55m
agl030519-vplxk-worker-us-east-1b   1         1         1       1           55m
agl030519-vplxk-worker-us-east-1c   1         1         1       1           55m
```

```
agl030519-vplxk-worker-us-east-1d   0         0                     55m
agl030519-vplxk-worker-us-east-1e   0         0                     55m
agl030519-vplxk-worker-us-east-1f   0         0                     55m
```

새 머신 세트가 사용 가능한 경우**DESIRED** 및 **CURRENT** 값이 일치합니다. 머신 세트를 사용할 수 없는 경우 몇 분 후에 명령을 다시 실행합니다.

3장. 머신 세트 수동 스케일링

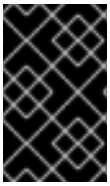
머신 세트에서 머신 인스턴스를 추가하거나 삭제할 수 있습니다.

참고

스케일링 이외의 머신 세트 요소를 수정해야하는 경우 머신 세트 변경을 참조하십시오.

3.1. 사전 요구 사항

- 클러스터 전체의 프록시를 활성화하여 설치 구성에서 **networking.machineNetwork[].cidr**에 포함되어 있지 않은 작업자를 확장하려면 연결 문제가 발생하지 않도록 프록시 개체의 **noProxy** 필드에 작업자를 추가해야 합니다.



중요

이 프로세스는 수동으로 프로비저닝된 시스템이 있는 클러스터에는 적용되지 않습니다. 머신 API가 작동하는 클러스터에서만 고급 머신 관리 및 스케일링 기능을 사용할 수 있습니다.

3.2. 머신 세트 수동 스케일링

머신 세트에서 머신 인스턴스를 추가하거나 제거하려면 머신 세트를 수동으로 스케일링할 수 있습니다.

이는 완전히 자동화된 설치 프로그램에 의해 프로비저닝된 인프라 설치와 관련이 있습니다. 사용자 지정된 사용자 프로비저닝 인프라 설치에는 머신 세트가 없습니다.

사전 요구 사항

- OpenShift Container Platform 클러스터 및 **oc** 명령행을 설치합니다.
- **cluster-admin** 권한이 있는 사용자로 **oc**에 로그인합니다.

절차

1. 클러스터에 있는 머신 세트를 확인합니다.

```
$ oc get machinesets -n openshift-machine-api
```

   머신 세트는 **<clusterid>-worker-<aws-region-az>** 형식으로 나열됩니다.

2. 클러스터에 있는 머신을 확인합니다.

```
$ oc get machine -n openshift-machine-api
```

3. 삭제하려는 머신에 주석을 설정합니다.

```
$ oc annotate machine/<machine_name> -n openshift-machine-api
machine.openshift.io/cluster-api-delete-machine="true"
```

4. 삭제하려는 노드를 비우고 제외합니다.

```
$ oc adm cordon <node_name>
$ oc adm drain <node_name>
```

5. 머신 세트를 스케일링합니다.

```
$ oc scale --replicas=2 machineset <machineset> -n openshift-machine-api
```

또는 다음을 수행합니다.

```
$ oc edit machineset <machineset> -n openshift-machine-api
```

머신 세트를 확장 또는 축소할 수 있습니다. 새 머신을 사용할 수 있을 때 까지 몇 분 정도 소요됩니다.

검증

- 원하는 머신 삭제를 확인합니다.

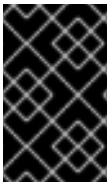
```
$ oc get machines
```

3.3. 머신 세트 삭제 정책

Random, **Newest** 및 **Oldest**의 세 가지 삭제 옵션이 지원됩니다. 기본값은 **Random**입니다. 따라서 머신 세트를 축소할 때 임의의 머신이 선택되어 삭제됩니다. 특정 머신 세트를 변경하고 유스 케이스에 따라 삭제 정책을 설정할 수 있습니다.

```
spec:
  deletePolicy: <delete_policy>
  replicas: <desired_replica_count>
```

삭제 정책에 관계없이 관련 머신에 **machine.openshift.io/cluster-api-delete-machine=true** 주석을 추가하여 특정 머신의 삭제 우선 순위를 지정할 수도 있습니다.



중요

기본적으로 OpenShift Container Platform 라우터 Pod는 작업자에게 배포됩니다. 라우터는 웹 콘솔을 포함한 일부 클러스터 리소스에 액세스해야 하므로 먼저 라우터 Pod를 재배치하지 않는 한 작업자 머신 세트를 **0**으로 스케일링하지 마십시오.

참고

사용자 정의 머신 세트는 특정 노드에서 서비스가 실행되고 작업자 머신 세트가 축소될 때 컨트롤러에서 해당 서비스를 무시해야 하는 유스 케이스에 사용할 수 있습니다. 이로 인해 서비스 중단을 피할 수 있습니다.

4장. 머신 세트 수정

레이블 추가, 인스턴스 유형 변경 또는 블록 스토리지 변경과 같은 머신 세트를 수정할 수 있습니다.

RHV(Red Hat Virtualization)에서는 머신 세트를 변경하여 다른 스토리지 도메인에 새 노드를 프로비저닝할 수도 있습니다.

참고

다른 변경없이 머신 세트를 확장해야하는 경우 머신 세트 수동 스케일링 을 참조하십시오.

4.1. 머신 세트 수정

머신 세트를 변경하려면 **MachineSet** YAML을 편집합니다. 다음으로 각 머신을 삭제하거나 복제본 수가 **0**이 되도록 머신 세트를 축소하여 머신 세트와 연관된 모든 머신을 제거합니다. 복제본을 필요한 수로 다시 조정합니다. 머신 세트를 변경해도 기존 머신에는 영향을 미치지 않습니다.

다른 변경을 수행하지 않고 머신 세트를 스케일링해야 하는 경우 머신을 삭제할 필요가 없습니다.

참고

기본적으로 OpenShift Container Platform 라우터 Pod는 작업자에게 배포됩니다. 라우터는 웹 콘솔을 포함한 일부 클러스터 리소스에 액세스해야 하므로 먼저 라우터 Pod를 재배치하지 않는 한 작업자 머신 세트를 **0**으로 스케일링하지 마십시오.

전제 조건

- OpenShift Container Platform 클러스터 및 **oc** 명령행을 설치합니다.
- **cluster-admin** 권한이 있는 사용자로 **oc**에 로그인합니다.

프로세스

1. 머신 세트를 편집합니다.

```
$ oc edit machineset <machineset> -n openshift-machine-api
```

2. 머신 세트를 **0**으로 축소합니다.

```
$ oc scale --replicas=0 machineset <machineset> -n openshift-machine-api
```

   또는 다음을 수행합니다.

```
$ oc edit machineset <machineset> -n openshift-machine-api
```

   머신이 제거될 때까지 기다립니다.

3. 필요에 따라 머신 세트를 확장합니다.

```
$ oc scale --replicas=2 machineset <machineset> -n openshift-machine-api
```

   또는 다음을 수행합니다.

```
$ oc edit machineset <machineset> -n openshift-machine-api
```

머신이 시작될 때까지 기다립니다. 새 머신에는 머신 세트에 대한 변경사항이 포함되어 있습니다.

4.2. RHV의 다른 스토리지 도메인으로 노드 마이그레이션

OpenShift Container Platform 컨트롤 플레인 및 컴퓨팅 노드를 RHV(Red Hat Virtualization) 클러스터에서 다른 스토리지 도메인으로 마이그레이션할 수 있습니다.

4.2.1. RHV에서 다른 스토리지 도메인으로 컴퓨팅 노드 마이그레이션

사전 요구 사항

- 관리자에게 로그인되어 있습니다.
- 대상 스토리지 도메인의 이름이 있습니다.

절차

1. 가상 머신 템플릿을 식별합니다.

```
$ oc get -o jsonpath='{.items[0].spec.template.spec.providerSpec.value.template_name}{"\n"}'
machineset -A
```

2. 식별한 템플릿을 기반으로 Manager에서 새 가상 시스템을 생성합니다. 다른 모든 설정은 변경하지 않고 그대로 둡니다. 자세한 내용은 Red Hat Virtualization 가상 머신 *관리 가이드의 템플릿 기반의* 가상 머신 생성을 참조하십시오.

   작은 정보

   새 가상 시스템을 시작할 필요가 없습니다.

3. 새 가상 시스템에서 새 템플릿을 생성합니다. **Target** (대상)에서 대상 스토리지 도메인을 지정합니다. 자세한 내용은 Red Hat Virtualization *가상 머신 관리 가이드에서 템플릿 생성을 참조하십시오.*

4. 새 템플릿으로 OpenShift Container Platform 클러스터에 새 머신 세트를 추가합니다.

   a. 현재 머신 세트의 세부 정보를 가져옵니다.

   ```
   $ oc get machineset -o yaml
   ```

   b. 다음 세부 정보를 사용하여 머신 세트를 생성합니다. 자세한 내용은 *머신 세트 생성을* 참조하십시오.
   template **_name 필드에 새 가상 시스템 템플릿**이름을 입력합니다. Manager(관리자)의 **New template(새 템플릿)** 대화 상자에서 사용한 것과 동일한 템플릿 이름을 사용합니다.

   c. 이전 시스템 집합과 새 시스템 집합의 이름을 확인합니다. 후속 단계에서 참조해야 합니다.

5. 워크로드를 마이그레이션합니다.

   a. 새 시스템 집합을 확장합니다. 머신 세트를 수동으로 스케일링하는 방법에 대한 자세한 내용은 머신 세트 수동 *스케일링을 참조하십시오.*

OpenShift Container Platform은 이전 머신이 제거되면 Pod를 사용 가능한 작업자로 이동합니다.

b. 이전 시스템 집합을 축소합니다.

6. 이전 머신 세트를 제거합니다.

```
$ oc delete machineset <machineset-name>
```

추가 리소스

- 머신 세트 생성.
- 머신 세트 수동 스케일링
- 스케줄러를 사용하여 Pod 배치 제어

4.2.2. 컨트롤 플레인 노드를 RHV의 다른 스토리지 도메인으로 마이그레이션

OpenShift Container Platform은 컨트롤 플레인 노드를 관리하지 않으므로 컴퓨팅 노드보다 더 쉽게 마이그레이션할 수 있습니다. RHV(Red Hat Virtualization)의 다른 가상 머신과 마찬가지로 마이그레이션할 수 있습니다.

각 노드에 대해 이 절차를 별도로 수행합니다.

사전 요구 사항

- 관리자에게 로그인되어 있습니다.
- 컨트롤 플레인 노드를 식별했습니다. 관리자의 **마스터** 로 레이블이 지정됩니다.

절차

1. **master** 라는 가상 시스템을 선택합니다.
2. 가상 시스템을 종료합니다.
3. **디스크** 탭을 클릭합니다.
4. 가상 시스템의 디스크를 클릭합니다.
5. 추가 **작업**  을 클릭하고 **Move(이동)**를 선택합니다.
6. 대상 스토리지 도메인을 선택하고 마이그레이션 프로세스가 완료될 때까지 기다립니다.
7. 가상 머신을 시작합니다.
8. OpenShift Container Platform 클러스터가 안정적인지 확인합니다.

```
$ oc get nodes
```

출력에 **Ready** 상태가 있는 노드가 표시되어야 합니다.

9. 각 컨트롤 플레인 노드에 대해 이 절차를 반복합니다.

5장. 머신 삭제

특정 머신을 삭제할 수 있습니다.

5.1. 특정 머신 삭제

특정 머신을 삭제할 수 있습니다.

참고

컨트롤 플레인 머신을 삭제할 수 없습니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 설치합니다.
- OpenShift CLI(**oc**)를 설치합니다.
- **cluster-admin** 권한이 있는 사용자로 **oc**에 로그인합니다.

절차

1. 클러스터에있는 머신을 확인하고 삭제할 머신을 식별합니다.

```
$ oc get machine -n openshift-machine-api
```

   명령 출력에는 **<clusterid>-worker-<cloud_region>** 형식의 머신 목록이 포함되어 있습니다.

2. 머신을 삭제합니다.

```
$ oc delete machine <machine> -n openshift-machine-api
```

중요

기본적으로 머신 컨트롤러는 성공할 때까지 머신이 지원하는 노드를 드레이닝하려고 합니다. 일부 경우, Pod의 중단 예산을 잘못 구성하는 등의 경우와 같이 노드 드레이닝 작업으로 인해 머신이 삭제되지 않을 수 있습니다. 특정 머신에서 "machine.openshift.io/exclude-node-draining"에 주석을 사용하여 노드 드레이닝 프로세스를 건너 뛸 수 있습니다. 삭제 중인 머신이 머신 세트에 속하는 경우 지정된 복제본 수를 충족하기 위해 새 머신이 즉시 생성됩니다.

5.2. 추가 리소스

- 비정상적인 etcd 멤버 교체.

6장. OPENSHIFT CONTAINER PLATFORM 클러스터에 자동 스케일링 적용

OpenShift Container Platform 클러스터에 자동 스케일링을 적용하려면클러스터 자동 스케일러를 배포한 다음 클러스터의 각 머신 유형에 대해 머신 자동 스케일러를 배포해야 합니다.

중요

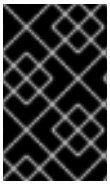
머신 API가 작동하는 클러스터에서만 클러스터 자동 스케일러를 구성할 수 있습니다.

6.1. 클러스터 자동 스케일러 정보

클러스터 자동 스케일러는 현재 배포 요구 사항에 따라 OpenShift Container Platform 클러스터의 크기를 조정합니다. 이는 Kubernetes 형식의 선언적 인수를 사용하여 특정 클라우드 공급자의 개체에 의존하지 않는 인프라 관리를 제공합니다. 클러스터 자동 스케일러에는 클러스터 범위가 있으며 특정 네임 스페이스와 연결되어 있지 않습니다.

리소스가 부족하여 현재 작업자 노드에서 Pod를 예약할 수 없거나 배포 요구 사항을 충족하는 데 다른 노드가 필요한 경우 클러스터 자동 스케일러는 클러스터 크기를 늘립니다. 클러스터 자동 스케일러는 사용자가 지정한 제한을 초과하여 클러스터 리소스를 늘리지 않습니다.

클러스터 자동 스케일러는 컨트롤 플레인 노드를 관리하지 않더라도 클러스터의 모든 노드에서 총 메모리, CPU 및 GPU를 계산합니다. 이러한 값은 단일 시스템 지향이 아닙니다. 전체 클러스터에 있는 모든 리소스를 집계한 것입니다. 예를 들어 최대 메모리 리소스 제한을 설정하는 경우 클러스터 자동 스케일러에는 현재 메모리 사용량을 계산할 때 클러스터의 모든 노드가 포함됩니다. 그런 다음 해당 계산을 사용하여 클러스터 자동 스케일러에 더 많은 작업자 리소스를 추가할 수 있는 용량이 있는지 확인합니다.



중요

작성한 **ClusterAutoscaler** 리솟스 정의의 **maxNodesTotal** 값이 클러스터에서 예상되는 총 머신 수를 대응하기에 충분한 크기의 값인지 확인합니다. 이 값에는 컨트롤 플레인 머신 수 및 확장 가능한 컴퓨팅 머신 수가 포함되어야 합니다.

10초마다 클러스터 자동 스케일러는 클러스터에서 불필요한 노드를 확인하고 제거합니다. 클러스터 자동 스케일러는 다음 조건이 적용되는 경우 제거할 노드를 고려합니다.

- 노드에서 실행 중인 모든 Pod의 CPU 및 메모리 요청 합계는 노드에서 할당된 리소스의 50% 미만입니다.
- 클러스터 자동 스케일러는 노드에서 실행 중인 모든 포드를 다른 노드로 이동할 수 있습니다.
- 클러스터 자동 확장기에는 축소 비활성화 주석이 없습니다.

노드에 다음 유형의 pod가 있는 경우 클러스터 자동 스케일러는 해당 노드를 제거하지 않습니다.

- 제한적인 PDB (Pod Disruption Budgets)가 있는 pod
- 기본적으로 노드에서 실행되지 않는 Kube 시스템 pod
- PDB가 없거나 제한적인 PDB가있는 Kube 시스템 pod
- deployment, replica set 또는 stateful set와 같은 컨트롤러 객체가 지원하지 않는 pod
- 로컬 스토리지가 있는 pod

- 리소스 부족, 호환되지 않는 노드 선택기 또는 어피니티(affinity), 안티-어피니티(anti-affinity) 일치 등으로 인해 다른 위치로 이동할 수 없는 pod
- **"cluster-autoscaler.kubernetes.io/safe-to-evict": "true"** 주석이없는 경우**"cluster-autoscaler.kubernetes.io/safe-to-evict": "false"** 주석이 있는 pod

예를 들어 최대 CPU 제한을 64코어로 설정하고 각각 8개의 코어만 있는 머신을 생성하도록 클러스터 자동 스케일러를 구성합니다. 클러스터가 30개 코어로 시작하는 경우 클러스터 자동 스케일러는 총 62개의 코어가 32개의 노드를 더 추가할 수 있습니다.

클러스터 자동 스케일러를 구성하면 추가 사용 제한이 적용됩니다.

- 자동 스케일링된 노드 그룹에 있는 노드를 직접 변경하지 마십시오. 동일한 노드 그룹 내의 모든 노드는 동일한 용량 및 레이블을 가지며 동일한 시스템 pod를 실행합니다.
- pod 요청을 지정합니다.
- pod가 너무 빨리 삭제되지 않도록 해야 하는 경우 적절한 PDB를 구성합니다.
- 클라우드 제공자 할당량이 구성하는 최대 노드 풀을 지원할 수 있는 충분한 크기인지를 확인합니다.
- 추가 노드 그룹 Autoscaler, 특히 클라우드 제공자가 제공하는 Autoscaler를 실행하지 마십시오.

HPA (Horizond Pod Autoscaler) 및 클러스터 자동 스케일러는 다른 방식으로 클러스터 리소스를 변경합니다. HPA는 현재 CPU 로드를 기준으로 배포 또는 복제 세트의 복제 수를 변경합니다. 로드가 증가하면 HPA는 클러스터에 사용 가능한 리소스 양에 관계없이 새 복제본을 만듭니다. 리소스가 충분하지 않은 경우 클러스터 자동 스케일러는 리소스를 추가하고 HPA가 생성한 pod를 실행할 수 있도록 합니다. 로드가 감소하면 HPA는 일부 복제를 중지합니다. 이 동작으로 일부 노드가 충분히 활용되지 않거나 완전히 비어 있을 경우 클러스터 자동 스케일러가 불필요한 노드를 삭제합니다.

클러스터 자동 스케일러는 pod 우선 순위를 고려합니다. Pod 우선 순위 및 선점 기능을 사용하면 클러스터에 충분한 리소스가 없는 경우 우선 순위에 따라 pod를 예약할 수 있지만 클러스터 자동 스케일러는 클러스터에 모든 pod를 실행하는 데 필요한 리소스가 있는지 확인합니다. 두 기능을 충족하기 위해 클러스터 자동 스케일러에는 우선 순위 컷오프 기능이 포함되어 있습니다. 이 컷오프 기능을 사용하여 "best-effort" pod를 예약하면 클러스터 자동 스케일러가 리소스를 늘리지 않고 사용 가능한 예비 리소스가 있을 때만 실행됩니다.

컷오프 값보다 우선 순위가 낮은 pod는 클러스터가 확장되지 않거나 클러스터가 축소되지 않도록합니다. pod를 실행하기 위해 추가된 새 노드가 없으며 이러한 pod를 실행하는 노드는 리소스를 확보하기 위해 삭제될 수 있습니다.

6.2. 머신 자동 스케일러 정보

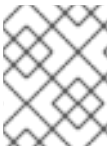
머신 자동 스케일러는 OpenShift Container Platform 클러스터에 배포하는 머신 세트의 Machine 수를 조정합니다. 기본 **worker** 머신 세트와 사용자가 만든 다른 머신 세트를 모두 확장할 수 있습니다. 머신 자동 스케일러는 클러스터에 더 많은 배포를 지원하기에 충분한 리소스가 없으면 Machine을 추가합니다. 최소 또는 최대 인스턴스 수와 같은 **MachineAutoscaler** 리소스의 값에 대한 모든 변경 사항은 대상이 되는 머신 세트에 즉시 적용됩니다.

중요

머신을 확장하려면 클러스터 자동 스케일러의 머신 자동 스케일러를 배포해야합니다. 클러스터 자동 스케일러는 머신 자동 스케일러가 설정한 머신 세트의 주석을 사용하여 확장할 수있는 리소스를 결정합니다. 머신 자동 스케일러도 정의하지 않고 클러스터 자동 스케일러를 정의하면 클러스터 자동 스케일러는 클러스터를 확장하지 않습니다.

6.3. 클러스터 자동 스케일러 구성

먼저 클러스터 자동 스케일러를 배포하여 OpenShift Container Platform 클러스터에서 리소스의 자동 스케일링을 관리합니다.



참고

클러스터 자동 스케일러의 범위는 전체 클러스터에 설정되므로 클러스터에 대해 하나의 클러스터 자동 스케일러만 만들 수 있습니다.

6.3.1. ClusterAutoscaler 리소스 정의

이 **ClusterAutoscaler** 리소스 정의는 클러스터 자동 스케일러의 매개 변수 및 샘플 값을 표시합니다.

```
apiVersion: "autoscaling.openshift.io/v1"
kind: "ClusterAutoscaler"
metadata:
  name: "default"
spec:
  podPriorityThreshold: -10 (1)
  resourceLimits:
    maxNodesTotal: 24 (2)
    cores:
      min: 8 (3)
      max: 128 (4)
    memory:
      min: 4 (5)
      max: 256 (6)
    gpus:
      - type: nvidia.com/gpu (7)
        min: 0 (8)
        max: 16 (9)
      - type: amd.com/gpu
        min: 0
        max: 4
  scaleDown: (10)
    enabled: true (11)
    delayAfterAdd: 10m (12)
    delayAfterDelete: 5m (13)
    delayAfterFailure: 30s (14)
    unneededTime: 5m (15)
```

(1) 클러스터 자동 스케일러가 추가 노드를 배포하도록 하려면 pod가 초과해야하는 우선 순위를 지정합니다. 32 비트 정수 값을 입력합니다. **podPriorityThreshold** 값은 각 pod에 할당한 **PriorityClass**의 값과 비교됩니다.

(2) 배포할 최대 노드 수를 지정합니다. 이 값은 Autoscaler가 제어하는 머신뿐 만 아니라 클러스터에 배치 된 총 머신 수입니다. 이 값이 모든 컨트롤 플레인 및 컴퓨팅 머신과 **MachineAutoscaler** 리소스에 지정한 총 복제본 수에 대응할 수 있을 만큼 충분한 크기의 값인지 확인합니다.

(3) 클러스터에 배포할 최소 코어 수를 지정합니다.

(4) 클러스터에 배포할 최대 코어 수를 지정합니다.

5 클러스터에서 최소 메모리 크기를 GiB 단위로 지정합니다.

6 클러스터에서 최대 메모리 크기를 GiB단위로 지정합니다.

7 선택 옵션으로 배포할 GPU 노드 유형을 지정합니다. **nvidia.com/gpu** 및 **amd.com/gpu** 만 유효한 유형입니다.

8 클러스터에 배포할 최소 GPU 수를 지정합니다.

9 클러스터에 배포할 최대 GPU 수를 지정합니다.

10 이 섹션에서는 **ns**, **us**, **ms**, **s**, **m** 및 **h**를 포함하여 유효한 ParseDuration 간격을 사용하여 각 작업에 대해 대기하는 기간을 지정할 수 있습니다.

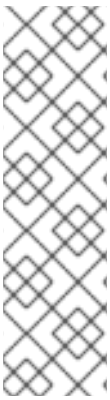
11 클러스터 자동 스케일러가 불필요한 노드를 제거할 수 있는지 여부를 지정합니다.

12 선택 사항으로 노드가 최근에 *추가된* 후 노드를 삭제하기 전까지 대기할 기간을 지정합니다. 값을 지정하지 않으면 기본값으로 **10m**이 사용됩니다.

13 최근에 노드가 *삭제 된* 후 노드를 삭제하기 전에 대기할 기간을 지정하십시오. 값을 지정하지 않으면 기본값으로 **10s**가 사용됩니다.

14 스케일 다운 실패 후 노드를 삭제하기 전에 대기할 기간을 지정합니다. 값을 지정하지 않으면 기본값으로 **3m**가 사용됩니다.

15 불필요한 노드가 삭제되기 전 까지 걸기는 기간을 지정합니다. 값을 지정하지 않으면 기본값으로 **10m**이 사용됩니다.



참고

스케일링 작업을 수행할 때 클러스터 자동 스케일러는 클러스터에서 배포할 최소 및 최대 코어 수 또는 메모리 양과 같은 **ClusterAutoscaler** 리소스 정의에 설정된 범위 내에 유지됩니다. 그러나 클러스터 자동 스케일러는 해당 범위 내에 있는 클러스터의 현재 값을 수정하지 않습니다.

최소 및 최대 CPU, 메모리 및 GPU 값은 클러스터의 모든 노드에서 해당 리소스를 계산하여 결정합니다(클러스터 자동 스케일러가 노드를 관리하지 않는 경우에도). 예를 들어 클러스터 자동 스케일러가 컨트롤 플레인 노드를 관리하지 않더라도 컨트롤 플레인 노드는 클러스터의 총 메모리에서 고려됩니다.

6.3.2. 클러스터 자동 스케일러 배포

클러스터 자동 스케일러를 배포하려면 **ClusterAutoscaler** 리소스의 인스턴스를 만듭니다.

절차

1. 사용자 정의된 리소스 정의가 포함된 **ClusterAutoscaler**리소스의 YAML 파일을 만듭니다.
2. 클러스터에서 리소스를 생성합니다.

```
$ oc create -f <filename>.yaml 1
```

1 **<filename>**은 사용자 정의 리소스 파일의 이름입니다.

6.4. 다음 단계

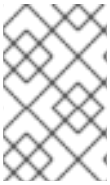
- 클러스터 자동 스케일러를 구성한 후 하나 이상의 머신 자동 스케일러를 구성해야 합니다.

6.5. 머신 자동 스케일러 구성

ClusterAutoscaler를 배포한 후 클러스터를 확장하는 데 사용되는 머신 세트를 참조하는 **MachineAutoscaler** 리소스를 배포합니다.

중요

ClusterAutoscaler 리소스를 배포 한 후 하나 이상의 **MachineAutoscaler** 리소스를 배포해야합니다.



참고

각 머신 세트에 대해 별도의 리소스를 구성해야합니다. 머신 세트는 리전마다 다르므로 여러 지역에서 머신 스케일링을 활성화할지 여부를 고려합니다. 스케일링하는 머신 세트에는 하나 이상의 머신이 있어야합니다.

6.5.1. MachineAutoscaler 리소스 정의

이 **MachineAutoscaler** 리소스 정의는 머신 자동 스케일러의 매개 변수 및 샘플 값을 표시합니다.

```
apiVersion: "autoscaling.openshift.io/v1beta1"
kind: "MachineAutoscaler"
metadata:
  name: "worker-us-east-1a" (1)
  namespace: "openshift-machine-api"
spec:
  minReplicas: 1 (2)
  maxReplicas: 12 (3)
  scaleTargetRef: (4)
    apiVersion: machine.openshift.io/v1beta1
    kind: MachineSet (5)
    name: worker-us-east-1a (6)
```

(1) 머신 자동 스케일러 이름을 지정합니다. 이 머신 자동 스케일러가 스케일링하는 머신 세트 보다 쉽게 식별할 수 있도록 스케일링할 머신 세트의 이름을 지정하거나 포함합니다. 머신 세트 이름의 형식은 **<clusterid>-<machineset>-<region>**입니다.

(2) 클러스터 자동 스케일러가 클러스터 스케일링을 시작한 후 지정된 영역에 남아 있어야하는 지정된 유형의 최소 머신 수를 지정하십시오. AWS, GCP 또는 Azure에서 실행중인 경우 이 값을 **0**으로 설정할 수 있습니다. 다른 공급 업체의 경우 이 값을 **0**으로 설정하지 마십시오.

특수 워크로드에 사용되는 비용이 많이 드는 하드웨어 또는 대규모 머신으로 머신 세트를 확장하는 등의 사용 사례에 이 값을 **0**으로 설정하여 비용을 절감할 수 있습니다. 머신을 사용하지 않는 경우 클러스터 자동 스케일러가 머신 세트를 0으로 축소합니다.

중요

설치 관리자 프로비저닝 인프라의 OpenShift Container Platform 설치 프로세스 중에 생성된 세 개의 컴퓨팅 머신 세트의 **spec.minReplicas** 값을 **0** 으로 설정하지 마십시오.

3. 클러스터 자동 스케일러가 클러스터 스케일링을 초기화한 후 지정된 영역에 배포할 수 있는 지정된 유형의 최대 머신 수를 지정합니다. **ClusterAutoscaler** 리소스 정의에서 **maxNodesTotal** 값이 머신 자동 스케일러가 머신 수를 배포할 수 있는 충분한 크기의 값임을 확인합니다.
4. 이 섹션에서는 스케일링할 기존 머신 세트를 설명하는 값을 지정합니다.
5. **kind** 매개 변수 값은 항상 **MachineSet**입니다.
6. **metadata.name** 매개 변수 값에 표시된 것처럼 **name** 값은 기존 머신 세트의 이름과 일치해야합니다.

6.5.2. 머신 자동 스케일러 배포

머신 자동 스케일러를 배포하려면 **MachineAutoscaler** 리소스의 인스턴스를 만듭니다.

절차

1. 사용자 정의된 리소스 정의가 포함된 **MachineAutoscaler** 리소스의 YAML 파일을 생성합니다.
2. 클러스터에서 리소스를 생성합니다.

```
$ oc create -f <filename>.yaml 1
```

1. **<filename>**은 사용자 정의 리소스 파일의 이름입니다.

6.6. 추가 리소스

- pod의 우선 순위에 대한 자세한 내용은 OpenShift Container Platform의 pod 스케줄링 결정에 pod 우선 순위 포함을 참조하십시오.

7장. 인프라 머신 세트 생성

중요

이 프로세스는 수동으로 프로비저닝된 시스템이 있는 클러스터에는 적용되지 않습니다. 머신 API가 작동하는 클러스터에서만 고급 머신 관리 및 스케일링 기능을 사용할 수 있습니다.

인프라 머신 세트를 사용하여 기본 라우터, 통합 컨테이너 이미지 레지스트리 및 클러스터 지표 및 모니터링을 위한 구성 요소와 같은 인프라 구성 요소만 호스팅하는 머신을 생성할 수 있습니다. 이러한 인프라 시스템은 환경을 실행하는 데 필요한 총 서브스크립션 수에 포함되지 않습니다.

7.1. OPENSHIFT CONTAINER PLATFORM 인프라 구성 요소

다음 인프라 워크로드에서는 OpenShift Container Platform 작업자 서브스크립션이 발생하지 않습니다.

- 마스터에서 실행되는 Kubernetes 및 OpenShift Container Platform 컨트롤 플레인 서비스
- 기본 라우터
- 통합된 컨테이너 이미지 레지스트리
- HAProxy 기반 Ingress 컨트롤러
- 사용자 정의 프로젝트를 모니터링하기위한 구성 요소를 포함한 클러스터 메트릭 수집 또는 모니터링 서비스
- 클러스터 집계 로깅
- 서비스 브로커
- Red Hat Quay
- Red Hat OpenShift Container Storage
- Red Hat Advanced Cluster Manager
- Red Hat Advanced Cluster Security for Kubernetes
- Red Hat OpenShift GitOps
- Red Hat OpenShift Pipelines

다른 컨테이너, Pod 또는 구성 요소를 실행하는 모든 노드는 서브스크립션을 적용해야 하는 작업자 노드입니다.

추가 리소스

- 인프라 노드 및 인프라 노드에서 실행할 수 있는 구성 요소에 대한 자세한 내용은 엔터프라이즈 Kubernetes 문서의 OpenShift 크기 조정 및 서브스크립션 가이드의 "Red Hat OpenShift 컨트롤 플레인 및 인프라 노드" 섹션을 참조하십시오.

7.2. 프로덕션 환경의 인프라 머신 세트 생성

프로덕션 배포에서는 인프라 구성 요소를 유지하기 위해 3개 이상의 머신 세트를 배포하는 것이 좋습니다. OpenShift Logging 및 Red Hat OpenShift Service Mesh는 모두 Elasticsearch를 배포합니다. 이 경우 서로 다른 노드에 3개의 인스턴스를 설치해야 합니다. 이러한 각 노드는 고가용성을 위해 다양한 가용 영역에 배포할 수 있습니다. 이와 같은 구성에는 가용성 영역마다 하나씩 세 개의 다른 시스템 집합이 필요합니다. 여러 가용성 영역이 없는 글로벌 Azure 리전에서는 가용성 세트를 사용하여 고가용성을 보장할 수 있습니다.

7.2.1. 다른 클라우드의 머신 세트 생성

클라우드에 샘플 머신 세트를 사용합니다.

7.2.1.1. AWS에서 머신 세트 사용자 지정 리소스의 샘플 YAML

이 샘플 YAML은 **us-east-1a** AWS(Amazon Web Services) 영역에서 실행되는 머신 세트를 정의하고 **node-role.kubernetes.io/infra: ""** 로 레이블이 지정된 노드를 생성합니다.

이 샘플에서 **<infrastructure_id>**는 클러스터를 프로비저닝할 때 설정한 클러스터 ID를 기반으로 하는 인프라 ID 레이블이며 **<infra>**는 추가할 노드 레이블입니다.

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
  name: <infrastructure_id>-infra-<zone> 2
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 3
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra-<zone> 4
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
        machine.openshift.io/cluster-api-machine-role: <infra> 6
        machine.openshift.io/cluster-api-machine-type: <infra> 7
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra-<zone> 8
    spec:
      metadata:
        labels:
          node-role.kubernetes.io/infra: "" 9
      taints: 10
        - key: node-role.kubernetes.io/infra
          effect: NoSchedule
      providerSpec:
        value:
          ami:
            id: ami-046fe691f52a953f9 11
          apiVersion: awsproviderconfig.openshift.io/v1beta1
          blockDevices:
            - ebs:
                iops: 0
```

```
          volumeSize: 120
          volumeType: gp2
      credentialsSecret:
        name: aws-cloud-credentials
      deviceIndex: 0
      iamInstanceProfile:
        id: <infrastructure_id>-worker-profile (12)
      instanceType: m4.large
      kind: AWSMachineProviderConfig
      placement:
        availabilityZone: us-east-1a
        region: us-east-1
      securityGroups:
        - filters:
            - name: tag:Name
              values:
                - <infrastructure_id>-worker-sg (13)
      subnet:
        filters:
          - name: tag:Name
            values:
              - <infrastructure_id>-private-us-east-1a (14)
      tags:
        - name: kubernetes.io/cluster/<infrastructure_id> (15)
          value: owned
      userDataSecret:
        name: worker-user-data
```

(1) (3) (5) (12) (13) (14) (15) 클러스터를 프로비저닝할 때 설정한 클러스터 ID를 기반으로 하는 인프라 ID를 지정합니다. OpenShift CLI 패키지가 설치되어 있으면 다음 명령을 실행하여 인프라 ID를 얻을 수 있습니다.

```
$ oc get -o jsonpath='{.status.infrastructureName}{"\n"}' infrastructure cluster
```

(2) (4) (8) 인프라 ID, **<infra>** 노드 레이블 및 영역을 지정합니다.

(6) (7) (9) **<infra>** 노드 레이블을 지정합니다.

(10) 사용자 워크로드가 인프라 노드에서 예약되지 않도록 테인트를 지정합니다.

(11) OpenShift Container Platform 노드의 AWS(Amazon Web Services) 영역에 유효한 RHCOS(Red Hat Enterprise Linux CoreOS) AMI를 지정합니다.

```
$ oc -n openshift-machine-api \
    -o jsonpath='{.spec.template.spec.providerSpec.value.ami.id}{"\n"}' \
    get machineset/<infrastructure_id>-worker-<zone>
```

AWS에서 실행되는 머신 세트는 보장되지 않는 Spot 인스턴스를 지원합니다. AWS의 온 디맨드 인스턴스에 비해 저렴한 가격으로 Spot 인스턴스를 사용하여 비용을 절약할 수 있습니다. **spotMarketOptions**를 **MachineSet** YAML 파일에 추가하여 Spot 인스턴스를 구성 합니다.

7.2.1.2. Azure의 머신 세트사용자 지정 리소스에 대한 샘플 YAML

이 샘플 YAML은 리전의 **1** Microsoft Azure 영역에서 실행되는 머신 세트를 정의하고 **node-role.kubernetes.io/infra: ""** 로 레이블이 지정된 노드를 만듭니다.

이 샘플에서 **<infrastructure_id>**는 클러스터를 프로비저닝할 때 설정한 클러스터 ID를 기반으로 하는 인프라 ID 레이블이며 **<infra>**는 추가할 노드 레이블입니다.

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
    machine.openshift.io/cluster-api-machine-role: <infra> 2
    machine.openshift.io/cluster-api-machine-type: <infra> 3
  name: <infrastructure_id>-infra-<region> 4
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra-<region> 6
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 7
        machine.openshift.io/cluster-api-machine-role: <infra> 8
        machine.openshift.io/cluster-api-machine-type: <infra> 9
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra-<region> 10
    spec:
      metadata:
        creationTimestamp: null
        labels:
          node-role.kubernetes.io/infra: "" 11
      taints: 12
      - key: node-role.kubernetes.io/infra
        effect: NoSchedule
      providerSpec:
        value:
          apiVersion: azureproviderconfig.openshift.io/v1beta1
          credentialsSecret:
            name: azure-cloud-credentials
            namespace: openshift-machine-api
          image:
            offer: ""
            publisher: ""
            resourceID: /resourceGroups/<infrastructure_id>-
rg/providers/Microsoft.Compute/images/<infrastructure_id> 13
            sku: ""
            version: ""
          internalLoadBalancer: ""
          kind: AzureMachineProviderSpec
          location: <region> 14
          managedIdentity: <infrastructure_id>-identity 15
```

```
        metadata:
          creationTimestamp: null
        natRule: null
        networkResourceGroup: ""
        osDisk:
          diskSizeGB: 128
          managedDisk:
            storageAccountType: Premium_LRS
          osType: Linux
        publicIP: false
        publicLoadBalancer: ""
        resourceGroup: <infrastructure_id>-rg (16)
        sshPrivateKey: ""
        sshPublicKey: ""
        subnet: <infrastructure_id>-<role>-subnet (17) (18)
        userDataSecret:
          name: worker-user-data (19)
        vmSize: Standard_DS4_v2
        vnet: <infrastructure_id>-vnet (20)
        zone: "1" (21)
```

(1) (5) (7) (13) (15) (16) (17) (20) 클러스터를 프로비저닝할 때 설정한 클러스터 ID를 기반으로 하는 인프라 ID를 지정합니다. OpenShift CLI 패키지가 설치되어 있으면 다음 명령을 실행하여 인프라 ID를 얻을 수 있습니다.

```
$ oc get -o jsonpath='{.status.infrastructureName}{"\n"}' infrastructure cluster
```

다음 명령을 실행하여 서브넷을 가져올 수 있습니다.

```
$ oc -n openshift-machine-api \
    -o jsonpath='{.spec.template.spec.providerSpec.value.subnet}{"\n"}' \
    get machineset/<infrastructure_id>-worker-centralus1
```

다음 명령을 실행하여 vnet을 가져올 수 있습니다.

```
$ oc -n openshift-machine-api \
    -o jsonpath='{.spec.template.spec.providerSpec.value.vnet}{"\n"}' \
    get machineset/<infrastructure_id>-worker-centralus1
```

(2) (3) (8) (9) (11) (18) (19) **<infra>** 노드 레이블을 지정합니다.

(4) (6) (10) 인프라 ID, **<infra>** 노드 레이블 및 리전을 지정합니다.

(12) 사용자 워크로드가 인프라 노드에서 예약되지 않도록 테인트를 지정합니다.

(14) 머신을 배치할 리전을 지정합니다.

(21) 머신을 배치할 리전 내 영역을 지정합니다. 해당 리전이 지정한 영역을 지원하는지 확인합니다.

Azure에서 실행되는 머신 세트는 보장되지 않는 Spot 가상 머신을 지원합니다. Azure의 표준 가상 머신에 비해 더 낮은 가격으로 Spot 가상 머신을 사용하여 비용을 절감할 수 있습니다. **spotVMOptions**를 **MachineSet** YAML 파일에 추가하여 Spot 가상 머신을 구성 할 수 있습니다.

7.2.1.3. GCP에서 머신 세트 사용자 정의 리소스의 샘플 YAML

이 샘플 YAML은 GCP(Google Cloud Platform)에서 실행되는 머신 세트를 정의하고 **node-role.kubernetes.io/infra: ""**로 레이블이 지정된 노드를 생성합니다.

이 샘플에서 **<infrastructure_id>**는 클러스터를 프로비저닝할 때 설정한 클러스터 ID를 기반으로 하는 인프라 ID 레이블이며 **<infra>**는 추가할 노드 레이블입니다.

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
  name: <infrastructure_id>-w-a 2
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 3
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-w-a 4
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
        machine.openshift.io/cluster-api-machine-role: <infra> 6
        machine.openshift.io/cluster-api-machine-type: <infra> 7
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-w-a 8
    spec:
      metadata:
        labels:
          node-role.kubernetes.io/infra: "" 9
      taints: 10
      - key: node-role.kubernetes.io/infra
        effect: NoSchedule
      providerSpec:
        value:
          apiVersion: gcpprovider.openshift.io/v1beta1
          canIPForward: false
          credentialsSecret:
            name: gcp-cloud-credentials
          deletionProtection: false
          disks:
          - autoDelete: true
            boot: true
            image: <path_to_image> 11
            labels: null
            sizeGb: 128
            type: pd-ssd
          gcpMetadata: 12
          - key: <custom_metadata_key>
            value: <custom_metadata_value>
          kind: GCPMachineProviderSpec
```

```
        machineType: n1-standard-4
        metadata:
          creationTimestamp: null
        networkInterfaces:
        - network: <infrastructure_id>-network (13)
          subnetwork: <infrastructure_id>-worker-subnet (14)
        projectID: <project_name> (15)
        region: us-central1
        serviceAccounts:
        - email: <infrastructure_id>-w@<project_name>.iam.gserviceaccount.com (16) (17)
          scopes:
          - https://www.googleapis.com/auth/cloud-platform
        tags:
        - <infrastructure_id>-worker (18)
        userDataSecret:
          name: worker-user-data
        zone: us-central1-a
```

(1) (2) (3) (4) (5) (8) (13) (14) (16) (18) 클러스터를 프로비저닝할 때 설정한 클러스터 ID를 기반으로 하는 인프라 ID를 지정합니다. OpenShift CLI 패키지가 설치되어 있으면 다음 명령을 실행하여 인프라 ID를 얻을 수 있습니다.

```
$ oc get -o jsonpath='{.status.infrastructureName}{"\n"}' infrastructure cluster
```

(6) (7) (9) **<infra>** 노드 레이블을 지정합니다.

(10) 사용자 워크로드가 인프라 노드에서 예약되지 않도록 테인트를 지정합니다.

(11) 현재 머신 세트에서 사용되는 이미지의 경로를 지정합니다. OpenShift CLI가 설치되어 있으면 다음 명령을 실행하여 이미지에 대한 경로를 얻을 수 있습니다.

```
$ oc -n openshift-machine-api \
    -o jsonpath='{.spec.template.spec.providerSpec.value.disks[0].image}{"\n"}' \
    get machineset/<infrastructure_id>-worker-a
```

(12) 선택 사항: **키:값** 쌍의 형식으로 사용자 정의 메타데이터를 지정합니다. 예를 들어 사용자 정의 메타데이터 설정에 대한 GCP 설명서를 참조하십시오.

(15) (17) 클러스터에 사용하는 GCP 프로젝트의 이름을 지정합니다.

GCP에서 실행되는 머신 세트는 보장되지 않은 선점 가능한 가상 머신 인스턴스를 지원합니다. GCP의 일반 인스턴스에 비해 더 낮은 가격으로 선점 가능한 가상 머신 인스턴스를 사용하여 비용을 절감할 수 있습니다. **preemptible**을 **MachineSet** YAML 파일에 추가하고 선점할 수 있는 가상 머신 인스턴스를 구성 할 수 있습니다.

7.2.1.4. RHOSP에서 머신 세트 사용자 정의 리소스의 샘플 YAML

이 샘플 YAML은 RHOSP(Red Hat OpenStack Platform)에서 실행되는 머신 세트를 정의하고 **node-role.kubernetes.io/infra: ""** 로 레이블이 지정된 노드를 생성합니다.

이 샘플에서 **<infrastructure_id>**는 클러스터를 프로비저닝할 때 설정한 클러스터 ID를 기반으로 하는 인프라 ID 레이블이며 **<infra>**는 추가할 노드 레이블입니다.

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
    machine.openshift.io/cluster-api-machine-role: <infra> 2
    machine.openshift.io/cluster-api-machine-type: <infra> 3
  name: <infrastructure_id>-infra 4
  namespace: openshift-machine-api
spec:
  replicas: <number_of_replicas>
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra 6
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 7
        machine.openshift.io/cluster-api-machine-role: <infra> 8
        machine.openshift.io/cluster-api-machine-type: <infra> 9
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra 10
    spec:
      metadata:
        creationTimestamp: null
        labels:
          node-role.kubernetes.io/infra: ""
      taints: 11
      - key: node-role.kubernetes.io/infra
        effect: NoSchedule
      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: <optional_UUID_of_server_group> 12
          kind: OpenstackProviderSpec
          networks: 13
          - filter: {}
            subnets:
            - filter:
                name: <subnet_name>
                tags: openshiftClusterID=<infrastructure_id> 14
          primarySubnet: <rhosp_subnet_UUID> 15
          securityGroups:
          - filter: {}
            name: <infrastructure_id>-worker 16
          serverMetadata:
            Name: <infrastructure_id>-worker 17
```

```
    openshiftClusterID: <infrastructure_id> 18
  tags:
  - openshiftClusterID=<infrastructure_id> 19
  trunk: true
  userDataSecret:
    name: worker-user-data 20
  availabilityZone: <optional_openstack_availability_zone>
```

1 5 7 14 16 17 18 19 클러스터를 프로비저닝할 때 설정한 클러스터 ID를 기반으로 하는 인프라 ID를 지정합니다. OpenShift CLI 패키지가 설치되어 있으면 다음 명령을 실행하여 인프라 ID를 얻을 수 있습니다.

```
$ oc get -o jsonpath='{.status.infrastructureName}{"\n"}' infrastructure cluster
```

2 3 8 9 20 **<infra>** 노드 레이블을 지정합니다.

4 6 10 인프라 ID 및 **<infra>** 노드 레이블을 지정합니다.

11 사용자 워크로드가 인프라 노드에서 예약되지 않도록 테인트를 지정합니다.

12 MachineSet의 서버 그룹 정책을 설정하려면, 서버 그룹 생성에서 반환된 값을 입력합니다. 대부분의 배포에는 **anti-affinity** 또는 **soft-anti-affinity** 정책이 권장됩니다.

13 여러 네트워크에 배포해야 합니다. 여러 네트워크에 배포하는 경우 이 목록에는 **primarySubnet** 값으로 사용되는 네트워크를 포함해야 합니다.

15 노드 엔드포인트를 게시할 RHOSP 서브넷을 지정합니다. 일반적으로 이 서브넷은 **install-config.yaml** 파일에서 **machineSubnet** 값으로 사용되는 서브넷과 동일합니다.

7.2.1.5. vSphere에서 머신 세트 사용자 정의 리소스의 샘플 YAML

이 샘플 YAML은 VMware vSphere에서 실행되는 머신 세트를 정의하고 **node-role.kubernetes.io/infra: ""** 로 레이블이 지정된 노드를 만듭니다.

이 샘플에서 **<infrastructure_id>**는 클러스터를 프로비저닝할 때 설정한 클러스터 ID를 기반으로 하는 인프라 ID 레이블이며 **<infra>**는 추가할 노드 레이블입니다.

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  creationTimestamp: null
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
  name: <infrastructure_id>-infra 2
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 3
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra 4
  template:
    metadata:
      creationTimestamp: null
```

```
    labels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
      machine.openshift.io/cluster-api-machine-role: <infra> 6
      machine.openshift.io/cluster-api-machine-type: <infra> 7
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra 8
  spec:
    metadata:
      creationTimestamp: null
      labels:
        node-role.kubernetes.io/infra: "" 9
    taints: 10
    - key: node-role.kubernetes.io/infra
      effect: NoSchedule
    providerSpec:
      value:
        apiVersion: vsphereprovider.openshift.io/v1beta1
        credentialsSecret:
          name: vsphere-cloud-credentials
        diskGiB: 120
        kind: VSphereMachineProviderSpec
        memoryMiB: 8192
        metadata:
          creationTimestamp: null
        network:
          devices:
          - networkName: "<vm_network_name>" 11
        numCPUs: 4
        numCoresPerSocket: 1
        snapshot: ""
        template: <vm_template_name> 12
        userDataSecret:
          name: worker-user-data
        workspace:
          datacenter: <vcenter_datacenter_name> 13
          datastore: <vcenter_datastore_name> 14
          folder: <vcenter_vm_folder_path> 15
          resourcepool: <vsphere_resource_pool> 16
          server: <vcenter_server_ip> 17
```

1 3 5 클러스터를 프로비저닝할 때 설정한 클러스터 ID를 기반으로 하는 인프라 ID를 지정합니다. OpenShift CLI (**oc**) 패키지가 설치되어 있으면 다음 명령을 실행하여 인프라 ID를 얻을 수 있습니다.

```
$ oc get -o jsonpath='{.status.infrastructureName}{"\n"}' infrastructure cluster
```

2 4 8 인프라 ID 및 **<infra>** 노드 레이블을 지정합니다.

6 7 9 **<infra>** 노드 레이블을 지정합니다.

10 사용자 워크로드가 인프라 노드에서 예약되지 않도록 테인트를 지정합니다.

11 머신 세트를 배포할 vSphere VM 네트워크를 지정합니다.

12 사용할 vSphere VM 템플릿 (예: **user-5ddjd-rhcos**)을 지정합니다.

13 머신 세트를 배포할 vCenter Datacenter를 지정합니다.

14 머신 세트를 배포할 vCenter Datastore를 지정합니다.

15 vCenter의 vSphere VM 폴더에 경로(예: **/dc1/vm/user-inst-5ddjd**)를 지정합니다.

16 VM의 vSphere 리소스 풀을 지정합니다.

17 vCenter 서버 IP 또는 정규화된 도메인 이름을 지정합니다.

7.2.2. 머신 세트 만들기

설치 프로그램에 의해 생성되는 것 이외에도 고유한 머신 세트를 만들어 선택한 특정 워크로드의 머신 컴퓨팅 리소스를 동적으로 관리할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 클러스터를 배포합니다.
- OpenShift CLI(**oc**)를 설치합니다.
- **cluster-admin** 권한이 있는 사용자로 **oc**에 로그인합니다.

절차

1. 머신 세트 CR(사용자 지정 리소스) 샘플이 포함된 이름이 **<file_name>.yaml**인 새 YAML 파일을 만듭니다.
   <clusterID> 및 **<role>** 매개 변수 값을 설정해야 합니다.

   a. 특정 필드에 설정할 값이 확실하지 않은 경우 클러스터에서 기존 머신 세트를 확인할 수 있습니다.

   ```
   $ oc get machinesets -n openshift-machine-api
   ```

   출력 예

   ```
   NAME                                DESIRED   CURRENT   READY   AVAILABLE   AGE
   agl030519-vplxk-worker-us-east-1a   1         1         1       1           55m
   agl030519-vplxk-worker-us-east-1b   1         1         1       1           55m
   agl030519-vplxk-worker-us-east-1c   1         1         1       1           55m
   agl030519-vplxk-worker-us-east-1d   0         0                             55m
   agl030519-vplxk-worker-us-east-1e   0         0                             55m
   agl030519-vplxk-worker-us-east-1f   0         0                             55m
   ```

   b. 특정 머신 세트의 값을 확인합니다.

   ```
   $ oc get machineset <machineset_name> -n \
     openshift-machine-api -o yaml
   ```

   출력 예

   ```
   ...
   template:
   ```

```
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: agl030519-vplxk (1)
    machine.openshift.io/cluster-api-machine-role: worker (2)
    machine.openshift.io/cluster-api-machine-type: worker
    machine.openshift.io/cluster-api-machineset: agl030519-vplxk-worker-us-east-1a
```

(1) 클러스터 ID입니다.

(2) 기본 노드 레이블입니다.

2. 새 **MachineSet** CR을 만듭니다.

```
$ oc create -f <file_name>.yaml
```

3. 머신 세트 목록을 표시합니다.

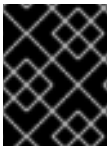
```
$ oc get machineset -n openshift-machine-api
```

출력 예

```
NAME                                 DESIRED   CURRENT   READY   AVAILABLE   AGE
agl030519-vplxk-infra-us-east-1a     1         1         1       1           11m
agl030519-vplxk-worker-us-east-1a    1         1         1       1           55m
agl030519-vplxk-worker-us-east-1b    1         1         1       1           55m
agl030519-vplxk-worker-us-east-1c    1         1         1       1           55m
agl030519-vplxk-worker-us-east-1d    0         0                             55m
agl030519-vplxk-worker-us-east-1e    0         0                             55m
agl030519-vplxk-worker-us-east-1f    0         0                             55m
```

새 머신 세트가 사용 가능한 경우**DESIRED** 및 **CURRENT** 값이 일치합니다. 머신 세트를 사용할 수 없는 경우 몇 분 후에 명령을 다시 실행합니다.

7.2.3. 인프라 노드 생성



중요

설치 관리자 프로비저닝 인프라 환경 또는 컨트롤 플레인 노드(마스터 노드라고도 함)가 머신 API에서 관리하는 클러스터의 인프라 머신 세트 생성을 참조하십시오.

클러스터의 요구 사항은 **infra** 노드라고도 불리는 인프라를 프로비저닝해야 합니다. 설치 프로그램은 컨트롤 플레인 및 작업자 노드에 대한 프로비저닝만 제공합니다. 작업자 노드는 레이블을 통해 인프라 노드 또는 애플리케이션 (app, **app** 이라고도 함)으로 지정할 수 있습니다.

절차

1. 애플리케이션 노드 역할을 수행할 작업자 노드에 레이블을 추가합니다.

```
$ oc label node <node-name> node-role.kubernetes.io/app=""
```

2. 인프라 노드 역할을 수행할 작업자 노드에 레이블을 추가합니다.

-

```
$ oc label node <node-name> node-role.kubernetes.io/infra=""
```

3. 해당 노드에 **infra** 역할 및 **app** 역할이 있는지 확인합니다.

```
$ oc get nodes
```

4. 기본 클러스터 수준 노드 선택기를 생성합니다. 기본 노드 선택기는 모든 네임스페이스에서 생성된 Pod에 적용됩니다. 이렇게 하면 Pod의 기존 노드 선택기와 교차점이 생성되어 Pod의 선택기가 추가로 제한됩니다.

중요

기본 노드 선택기 키가 Pod 라벨 키와 충돌하는 경우 기본 노드 선택기가 적용되지 않습니다.

그러나 Pod를 예약할 수 없게 만들 수 있는 기본 노드 선택기를 설정하지 마십시오. 예를 들어 Pod의 라벨이 **node-role.kubernetes.io/master=""**와 같은 다른 노드 역할로 설정된 경우 기본 노드 선택기를 **node-role.kubernetes.io/infra=""**와 같은 특정 노드 역할로 설정하면 Pod를 예약할 수 없게 될 수 있습니다. 따라서 기본 노드 선택기를 특정 노드 역할로 설정할 때 주의해야 합니다.

또는 프로젝트 노드 선택기를 사용하여 클러스터 수준 노드 선택기 키 충돌을 방지할 수 있습니다.

a. **Scheduler** 오브젝트를 편집합니다.

```
$ oc edit scheduler cluster
```

b. 적절한 노드 선택기를 사용하여 **defaultNodeSelector** 필드를 추가합니다.

```
apiVersion: config.openshift.io/v1
kind: Scheduler
metadata:
  name: cluster
...
spec:
  defaultNodeSelector: topology.kubernetes.io/region=us-east-1 1
...
```

1 이 예제 노드 선택기는 기본적으로 **us-east-1** 리전의 노드에 Pod를 배포합니다.

c. 파일을 저장하여 변경 사항을 적용합니다.

인프라 리소스를 새로 레이블이 지정된 인프라 노드로 이동할 수 있습니다.

추가 리소스

- 인프라 머신 세트로 리소스 이동

7.2.4. 인프라 머신의 머신 구성 풀 생성

전용 구성을 위한 인프라 머신이 필요한 경우 인프라 풀을 생성해야 합니다.

절차

1. 특정 레이블이 있는 인프라 노드로 할당하려는 노드에 레이블을 추가합니다.

```
$ oc label node <node_name> <label>
```

```
$ oc label node ci-ln-n8mqwr2-f76d1-xscn2-worker-c-6fmtx node-role.kubernetes.io/infra=
```

2. 작업자 역할과 사용자 지정 역할을 모두 포함하는 머신 구성 풀을 머신 구성 선택기로 생성합니다.

```
$ cat infra.mcp.yaml
```

출력 예

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfigPool
metadata:
  name: infra
spec:
  machineConfigSelector:
    matchExpressions:
      - {key: machineconfiguration.openshift.io/role, operator: In, values: [worker,infra]} (1)
  nodeSelector:
    matchLabels:
      node-role.kubernetes.io/infra: "" (2)
```

(1) 작업자 역할 및 사용자 지정 역할을 추가합니다.

(2) 노드에 추가한 레이블을 **nodeSelector**로 추가합니다.

참고

사용자 지정 머신 구성 풀은 작업자 풀의 머신 구성을 상속합니다. 사용자 지정 풀은 작업자 풀을 대상으로 하는 머신 구성을 사용하지만 사용자 지정 풀을 대상으로 하는 변경 사항만 배포할 수 있는 기능을 추가합니다. 사용자 지정 풀은 작업자 풀에서 리소스를 상속하므로 작업자 풀을 변경하면 사용자 지정 풀에도 영향을 줍니다.

3. YAML 파일이 있으면 머신 구성 풀을 생성할 수 있습니다.

```
$ oc create -f infra.mcp.yaml
```

4. 머신 구성을 확인하고 인프라 구성이 성공적으로 렌더링되었는지 확인합니다.

```
$ oc get machineconfig
```

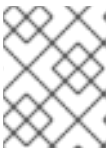
출력 예

```
NAME                                               GENERATEDBYCONTROLLER
IGNITIONVERSION   CREATED
```

```
00-master                                  365c1cfd14de5b0e3b85e0fc815b0060f36ab955
2.2.0             31d
00-worker                                  365c1cfd14de5b0e3b85e0fc815b0060f36ab955
2.2.0             31d
01-master-container-runtime
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   2.2.0             31d
01-master-kubelet                          365c1cfd14de5b0e3b85e0fc815b0060f36ab955
2.2.0             31d
01-worker-container-runtime
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   2.2.0             31d
01-worker-kubelet                          365c1cfd14de5b0e3b85e0fc815b0060f36ab955
2.2.0             31d
99-master-1ae2a1e0-a115-11e9-8f14-005056899d54-registries
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   2.2.0             31d
99-master-ssh                                                2.2.0             31d
99-worker-1ae64748-a115-11e9-8f14-005056899d54-registries
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   2.2.0             31d
99-worker-ssh                                                2.2.0             31d
rendered-infra-4e48906dca84ee702959c71a53ee80e7
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   2.2.0             19s
rendered-master-072d4b2da7f88162636902b074e9e28e
5b6fb8349a29735e48446d435962dec4547d3090   2.2.0             31d
rendered-master-3e88ec72aed3886dec061df60d16d1af
02c07496ba0417b3e12b78fb32baf6293d314f79   2.2.0             31d
rendered-master-419bee7de96134963a15fdf9dd473b25
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   2.2.0             17d
rendered-master-53f5c91c7661708adce18739cc0f40fb
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   2.2.0             13d
rendered-master-a6a357ec18e5bce7f5ac426fc7c5ffcd
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   2.2.0             7d3h
rendered-master-dc7f874ec77fc4b969674204332da037
5b6fb8349a29735e48446d435962dec4547d3090   2.2.0             31d
rendered-worker-1a75960c52ad18ff5dfa6674eb7e533d
5b6fb8349a29735e48446d435962dec4547d3090   2.2.0             31d
rendered-worker-2640531be11ba43c61d72e82dc634ce6
5b6fb8349a29735e48446d435962dec4547d3090   2.2.0             31d
rendered-worker-4e48906dca84ee702959c71a53ee80e7
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   2.2.0             7d3h
rendered-worker-4f110718fe88e5f349987854a1147755
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   2.2.0             17d
rendered-worker-afc758e194d6188677eb837842d3b379
02c07496ba0417b3e12b78fb32baf6293d314f79   2.2.0             31d
rendered-worker-daa08cc1e8f5fcdeba24de60cd955cc3
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   2.2.0             13d
```

rendered-infra-* 접두사가 있는 새 머신 구성이 표시되어야 합니다.

5. 선택 사항: 사용자 지정 풀에 변경 사항을 배포하려면 **infra** 와 같은 사용자 지정 풀 이름을 레이블로 사용하는 머신 구성을 생성합니다. 필수 사항은 아니며 지침 용도로만 표시됩니다. 이렇게 하면 인프라 노드에 고유한 사용자 지정 구성을 적용할 수 있습니다.



참고

새 머신 구성 풀을 생성한 후 MCO는 해당 풀에 대해 새로 렌더링된 구성과 해당 풀의 관련 노드를 다시 부팅하여 새 구성을 적용합니다.

a. 머신 구성을 생성합니다.

```
$ cat infra.mc.yaml
```

출력 예

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: 51-infra
  labels:
    machineconfiguration.openshift.io/role: infra (1)
spec:
  config:
    ignition:
      version: 3.1.0
    storage:
      files:
      - path: /etc/infratest
        mode: 0644
        contents:
          source: data:,infra
```

(1) 노드에 추가한 레이블을 **nodeSelector**로 추가합니다.

b. 머신 구성을 인프라 레이블 노드에 적용합니다.

```
$ oc create -f infra.mc.yaml
```

6. 새 머신 구성 풀을 사용할 수 있는지 확인합니다.

```
$ oc get mcp
```

출력 예

```
NAME     CONFIG                                             UPDATED   UPDATING   DEGRADED
MACHINECOUNT   READYMACHINECOUNT   UPDATEDMACHINECOUNT
DEGRADEDMACHINECOUNT   AGE
infra    rendered-infra-60e35c2e99f42d976e084fa94da4d0fc    True      False      False      1
1              1                   0                    4m20s
master   rendered-master-9360fdb895d4c131c7c4bebbae099c90   True      False      False
3              3                   3                    0                    91m
worker   rendered-worker-60e35c2e99f42d976e084fa94da4d0fc   True      False      False
2              2                   2                    0                    91m
```

이 예에서는 작업자 노드가 인프라 노드로 변경되었습니다.

추가 리소스

- 사용자 정의 풀에서 인프라 머신 그룹화에 대한 자세한 내용은 머신 구성 풀을 사용한 노드 구성 관리를 참조하십시오.

7.3. 인프라 노드에 머신 세트 리소스 할당

인프라 머신 세트를 생성 한 후 **worker** 및 **infra** 역할이 새 인프라 노드에 적용됩니다. **infra** 역할이 적용된 노드는 **worker** 역할이 적용된 경우에도 환경을 실행하는 데 필요한 총 서브스크립션 수에 포함되지 않습니다.

그러나 인프라 노드가 작업자로 할당되면 사용자 워크로드가 실수로 인프라 노드에 할당될 수 있습니다. 이를 방지하려면 제어하려는 pod에 대한 허용 오차를 적용하고 인프라 노드에 테인트를 적용할 수 있습니다.

7.3.1. 테인트 및 허용 오차를 사용하여 인프라 노드 워크로드 바인딩

infra 및 **worker** 역할이 할당된 인프라 노드가 있는 경우 사용자 워크로드가 할당되지 않도록 노드를 구성해야 합니다.

중요

인프라 노드에 대해 생성된 이중 **infra,worker** 레이블을 유지하고 테인트 및 허용 오차를 사용하여 사용자 워크로드가 예약된 노드를 관리하는 것이 좋습니다. 노드에서 **worker** 레이블을 제거하는 경우 이를 관리할 사용자 지정 풀을 생성해야 합니다. **master** 또는 **worker** 이외의 레이블이 있는 노드는 사용자 지정 풀없이 MCO에서 인식되지 않습니다. **worker** 레이블을 유지 관리하면 사용자 정의 레이블을 선택하는 사용자 정의 풀이 없는 경우 기본 작업자 머신 구성 풀에서 노드를 관리할 수 있습니다. **infra** 레이블은 총 서브스크립션 수에 포함되지 않는 클러스터와 통신합니다.

사전 요구 사항

- OpenShift Container Platform 클러스터에서 추가 **MachineSet** 개체를 구성합니다.

절차

1. 인프라 노드에 테인트를 추가하여 사용자 워크로드를 예약하지 않도록합니다.

   a. 노드에 테인트가 있는지 확인합니다.

   ```
   $ oc describe nodes <node_name>
   ```

   샘플 출력

   ```
   oc describe node ci-ln-iyhx092-f76d1-nvdfm-worker-b-wln2l
   Name:               ci-ln-iyhx092-f76d1-nvdfm-worker-b-wln2l
   Roles:              worker
    ...
   Taints:             node-role.kubernetes.io/infra:NoSchedule
    ...
   ```

   이 예에서는 노드에 테인트가 있음을 보여줍니다. 다음 단계에서 Pod에 허용 오차를 추가할 수 있습니다.

   b. 사용자 워크로드를 예약하지 않도록 테인트를 구성하지 않은 경우 다음을 수행합니다.

   ```
   $ oc adm taint nodes <node_name> <key>:<effect>
   ```

   예를 들면 다음과 같습니다.

```
$ oc adm taint nodes node1 node-role.kubernetes.io/infra:NoSchedule
```

이 예에서는 키**node-role.kubernetes.io/infra** 및 taint 효과 **NoSchedule**이 있는 **node1**에 taint를 배치합니다. **NoSchedule** 효과가 있는 노드는 taint를 허용하는 pod만 예약하지만 기존 pod는 노드에서 예약된 상태를 유지할 수 있습니다.

참고

descheduler를 사용하면 노드 taint를 위반하는 pod가 클러스터에서 제거될 수 있습니다.

2. 라우터, 레지스트리 및 모니터링 워크로드와 같이 인프라 노드에서 예약하려는 pod 구성에 대한 허용 오차를 추가합니다. 다음 코드를 **Pod** 개체 사양에 추가합니다.

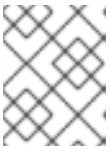
```
tolerations:
 - effect: NoSchedule (1)
   key: node-role.kubernetes.io/infra (2)
   operator: Exists (3)
```

(1) 노드에 추가한 효과를 지정합니다.

(2) 노드에 추가한 키를 지정합니다.

(3) 노드에 **elasticsearch** 키가 있는 taint를 요구하도록 **Exists** Operator를 지정합니다.

이 허용 오차는 **oc adm taint** 명령으로 생성된 taint와 일치합니다. 이 허용 오차가 있는 pod를 인프라 노드에 예약할 수 있습니다.



참고

OLM을 통해 설치된 Operator의 pod를 인프라 노드로 이동할 수는 없습니다. Operator pod를 이동하는 기능은 각 Operator의 구성에 따라 다릅니다.

3. 스케줄러를 사용하여 pod를 인프라 노드에 예약합니다. 자세한 내용은 *노드에서 pod 배치 제어*에 대한 설명서를 참조하십시오.

추가 리소스

- 노드에 Pod 예약에 대한 일반 정보는 스케줄러를 사용하여 Pod 배치 제어 에서 참조하십시오.
- pod를 인프라 노드로 예약하는 방법에 대한 지침은 인프라 머신 세트로 리소스 이동 을 참조하십시오.

7.4. 인프라 머신 세트로 리소스 이동

일부 인프라 리소스는 기본적으로 클러스터에 배포됩니다. 이를 생성한 인프라 머신 세트로 이동할 수 있습니다.

7.4.1. 라우터 이동

라우터 Pod를 다른 머신 세트에 배포할 수 있습니다. 기본적으로 Pod는 작업자 노드에 배포됩니다.

전제 조건

- OpenShift Container Platform 클러스터에서 추가 머신 세트를 구성합니다.

프로세스

1. 라우터 Operator의 **IngressController** 사용자 정의 리소스를 표시합니다.

```
$ oc get ingresscontroller default -n openshift-ingress-operator -o yaml
```

명령 출력은 다음 예제와 유사합니다.

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  creationTimestamp: 2019-04-18T12:35:39Z
  finalizers:
  - ingresscontroller.operator.openshift.io/finalizer-ingresscontroller
  generation: 1
  name: default
  namespace: openshift-ingress-operator
  resourceVersion: "11341"
  selfLink: /apis/operator.openshift.io/v1/namespaces/openshift-ingress-
operator/ingresscontrollers/default
  uid: 79509e05-61d6-11e9-bc55-02ce4781844a
spec: {}
status:
  availableReplicas: 2
  conditions:
  - lastTransitionTime: 2019-04-18T12:36:15Z
    status: "True"
    type: Available
  domain: apps.<cluster>.example.com
  endpointPublishingStrategy:
    type: LoadBalancerService
  selector: ingresscontroller.operator.openshift.io/deployment-ingresscontroller=default
```

2. **ingresscontroller** 리소스를 편집하고 **infra** 레이블을 사용하도록 **nodeSelector**를 변경합니다.

```
$ oc edit ingresscontroller default -n openshift-ingress-operator
```

다음과 같이 **infra** 레이블을 참조하는 **nodeSelector** 부분을 **spec** 섹션에 추가합니다.

```
spec:
  nodePlacement:
    nodeSelector:
      matchLabels:
        node-role.kubernetes.io/infra: ""
```

3. 라우터 pod가 **infra** 노드에서 실행되고 있는지 확인합니다.

   a. 라우터 pod 목록을 표시하고 실행중인 pod의 노드 이름을 기록해 둡니다.

```
$ oc get pod -n openshift-ingress -o wide
```

출력 예

```
NAME                         READY   STATUS        RESTARTS   AGE     IP           NODE
NOMINATED NODE   READINESS GATES
router-default-86798b4b5d-bdlvd   1/1     Running       0          28s     10.130.2.4   ip-10-
0-217-226.ec2.internal   <none>           <none>
router-default-955d875f4-255g8    0/1     Terminating   0          19h     10.129.2.4   ip-10-
0-148-172.ec2.internal   <none>           <none>
```

이 예에서 실행중인 pod는 **ip-10-0-217-226.ec2.internal** 노드에 있습니다.

b. 실행중인 pod의 노드 상태를 표시합니다.

```
$ oc get node <node_name> 1
```

1 pod 목록에서 얻은 **<node_name>**을 지정합니다.

출력 예

```
NAME                         STATUS  ROLES        AGE   VERSION
ip-10-0-217-226.ec2.internal  Ready   infra,worker  17h   v1.19.0
```

역할 목록에 **infra**가 포함되어 있으므로 pod가 올바른 노드에서 실행됩니다.

7.4.2. 기본 레지스트리 이동

Pod를 다른 노드에 배포하도록 레지스트리 Operator를 구성합니다.

전제 조건

- OpenShift Container Platform 클러스터에서 추가 머신 세트를 구성합니다.

프로세스

1. **config/instance** 개체를 표시합니다.

```
$ oc get configs.imageregistry.operator.openshift.io/cluster -o yaml
```

출력 예

```
apiVersion: imageregistry.operator.openshift.io/v1
kind: Config
metadata:
  creationTimestamp: 2019-02-05T13:52:05Z
  finalizers:
  - imageregistry.operator.openshift.io/finalizer
  generation: 1
  name: cluster
  resourceVersion: "56174"
  selfLink: /apis/imageregistry.operator.openshift.io/v1/configs/cluster
  uid: 36fd3724-294d-11e9-a524-12ffeee2931b
spec:
```

```
  httpSecret: d9a012ccd117b1e6616ceccb2c3bb66a5fed1b5e481623
  logging: 2
  managementState: Managed
  proxy: {}
  replicas: 1
  requests:
    read: {}
    write: {}
  storage:
    s3:
      bucket: image-registry-us-east-1-c92e88cad85b48ec8b312344dff03c82-392c
      region: us-east-1
status:
...
```

2. **config/instance** 개체를 편집합니다.

```
$ oc edit configs.imageregistry.operator.openshift.io/cluster
```

3. 다음 YAML과 동일하게 오브젝트의 **spec** 섹션을 수정합니다.

```
spec:
  affinity:
    podAntiAffinity:
      preferredDuringSchedulingIgnoredDuringExecution:
      - podAffinityTerm:
          namespaces:
          - openshift-image-registry
          topologyKey: kubernetes.io/hostname
        weight: 100
  logLevel: Normal
  managementState: Managed
  nodeSelector:
    node-role.kubernetes.io/infra: ""
```

4. 레지스트리 pod가 인프라 노드로 이동되었는지 검증합니다.

   a. 다음 명령을 실행하여 레지스트리 pod가 있는 노드를 식별합니다.

   ```
   $ oc get pods -o wide -n openshift-image-registry
   ```

   b. 노드에 지정된 레이블이 있는지 확인합니다.

   ```
   $ oc describe node <node_name>
   ```

   명령 출력을 확인하고 **node-role.kubernetes.io/infra**가 **LABELS** 목록에 있는지 확인합니다.

7.4.3. 모니터링 솔루션 이동

기본적으로 Prometheus, Grafana 및 AlertManager가 포함된 Prometheus Cluster Monitoring 스택은 클러스터 모니터링을 제공하기 위해 배포됩니다. 이는 Cluster Monitoring Operator가 관리합니다. 이러한 구성 요소를 다른 머신으로 이동하려면 사용자 정의 구성 맵을 생성하고 적용해야 합니다.

프로세스

1. 다음 **ConfigMap** 정의를 **cluster-monitoring-configmap.yaml** 파일로 저장합니다.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |+
    alertmanagerMain:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    prometheusK8s:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    prometheusOperator:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    grafana:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    k8sPrometheusAdapter:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    kubeStateMetrics:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    telemeterClient:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    openshiftStateMetrics:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    thanosQuerier:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
```

   이 구성 맵을 실행하면 모니터링 스택의 구성 요소가 인프라 노드에 재배포됩니다.

2. 새 구성 맵을 적용합니다.

```
$ oc create -f cluster-monitoring-configmap.yaml
```

3. 모니터링 pod가 새 머신으로 이동하는 것을 확인합니다.

```
$ watch 'oc get pod -n openshift-monitoring -o wide'
```

4. 구성 요소가 **infra** 노드로 이동하지 않은 경우 이 구성 요소가 있는 pod를 제거합니다.

```
$ oc delete pod -n openshift-monitoring <pod>
```

   삭제된 pod의 구성 요소가 **infra** 노드에 다시 생성됩니다.

7.4.4. 클러스터 로깅 리소스 이동

클러스터 로깅 구성 요소, Elasticsearch, Kibana 및 Curator의 Pod를 다른 노드에 배포하도록 Cluster Logging Operator를 구성할 수 있습니다. 설치된 위치에서 Cluster Logging Operator Pod를 이동할 수 없습니다.

예를 들어 높은 CPU, 메모리 및 디스크 요구 사항으로 인해 Elasticsearch Pod를 다른 노드로 옮길 수 있습니다.

사전 요구 사항

- 클러스터 로깅 및 Elasticsearch가 설치되어 있어야 합니다. 이러한 기능은 기본적으로 설치되지 않습니다.

프로세스

1. **openshift-logging** 프로젝트에서 **ClusterLogging** 사용자 정의 리소스(CR)를 편집합니다.

```
$ oc edit ClusterLogging instance
```

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging

...

spec:
  collection:
    logs:
      fluentd:
        resources: null
      type: fluentd
  curation:
    curator:
      nodeSelector: 1
        node-role.kubernetes.io/infra: ''
      resources: null
      schedule: 30 3 * * *
    type: curator
  logStore:
    elasticsearch:
      nodeCount: 3
      nodeSelector: 2
        node-role.kubernetes.io/infra: ''
      redundancyPolicy: SingleRedundancy
      resources:
        limits:
          cpu: 500m
          memory: 16Gi
        requests:
          cpu: 500m
          memory: 16Gi
      storage: {}
    type: elasticsearch
  managementState: Managed
  visualization:
    kibana:
      nodeSelector: 3
```

```
        node-role.kubernetes.io/infra: ''
      proxy:
        resources: null
      replicas: 1
      resources: null
    type: kibana

...
```

1 2 3 적절한 값이 설정된 **nodeSelector** 매개변수를 이동하려는 구성 요소에 추가합니다. 표시된 형식으로 **nodeSelector**를 사용하거나 노드에 지정된 값에 따라 **<key>: <value>** 쌍을 사용할 수 있습니다.

검증

oc get pod -o wide 명령을 사용하여 구성 요소가 이동했는지 확인할 수 있습니다.

예를 들면 다음과 같습니다.

- **ip-10-0-147-79.us-east-2.compute.internal** 노드에서 Kibana pod를 이동하려고 경우 다음을 실행합니다.

```
$ oc get pod kibana-5b8bdf44f9-ccpq9 -o wide
```

출력 예

```
NAME                     READY   STATUS    RESTARTS   AGE   IP            NODE
NOMINATED NODE   READINESS GATES
kibana-5b8bdf44f9-ccpq9  2/2     Running   0          27s   10.129.2.18   ip-10-0-147-79.us-
east-2.compute.internal   <none>           <none>
```

- Kibana Pod를 전용 인프라 노드인 **ip-10-0-139-48.us-east-2.compute.internal** 노드로 이동하려는 경우 다음을 실행합니다.

```
$ oc get nodes
```

출력 예

```
NAME                                         STATUS   ROLES        AGE   VERSION
ip-10-0-133-216.us-east-2.compute.internal   Ready    master       60m   v1.19.0
ip-10-0-139-146.us-east-2.compute.internal   Ready    master       60m   v1.19.0
ip-10-0-139-192.us-east-2.compute.internal   Ready    worker       51m   v1.19.0
ip-10-0-139-241.us-east-2.compute.internal   Ready    worker       51m   v1.19.0
ip-10-0-147-79.us-east-2.compute.internal    Ready    worker       51m   v1.19.0
ip-10-0-152-241.us-east-2.compute.internal   Ready    master       60m   v1.19.0
ip-10-0-139-48.us-east-2.compute.internal    Ready    infra        51m   v1.19.0
```

노드에는 **node-role.kubernetes.io/infra : ''** 레이블이 있음에 유의합니다.

```
$ oc get node ip-10-0-139-48.us-east-2.compute.internal -o yaml
```

출력 예

```
kind: Node
apiVersion: v1
metadata:
  name: ip-10-0-139-48.us-east-2.compute.internal
  selfLink: /api/v1/nodes/ip-10-0-139-48.us-east-2.compute.internal
  uid: 62038aa9-661f-41d7-ba93-b5f1b6ef8751
  resourceVersion: '39083'
  creationTimestamp: '2020-04-13T19:07:55Z'
  labels:
    node-role.kubernetes.io/infra: ''
...
```

- Kibana pod를 이동하려면 **ClusterLogging** CR을 편집하여 노드 선택기를 추가합니다.

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging

...

spec:

...

  visualization:
    kibana:
      nodeSelector: 1
        node-role.kubernetes.io/infra: ''
      proxy:
        resources: null
      replicas: 1
      resources: null
    type: kibana
```

1 노드 사양의 레이블과 일치하는 노드 선택기를 추가합니다.

- CR을 저장하면 현재 Kibana pod가 종료되고 새 pod가 배포됩니다.

```
$ oc get pods
```

출력 예

```
NAME                                            READY   STATUS        RESTARTS   AGE
cluster-logging-operator-84d98649c4-zb9g7       1/1     Running       0          29m
elasticsearch-cdm-hwv01pf7-1-56588f554f-kpmlg   2/2     Running       0          28m
elasticsearch-cdm-hwv01pf7-2-84c877d75d-75wqj   2/2     Running       0          28m
elasticsearch-cdm-hwv01pf7-3-f5d95b87b-4nx78    2/2     Running       0          28m
fluentd-42dzz                                   1/1     Running       0          28m
fluentd-d74rq                                   1/1     Running       0          28m
fluentd-m5vr9                                   1/1     Running       0          28m
fluentd-nkxl7                                   1/1     Running       0          28m
fluentd-pdvqb                                   1/1     Running       0          28m
fluentd-tflh6                                   1/1     Running       0          28m
kibana-5b8bdf44f9-ccpq9                         2/2     Terminating   0          4m11s
kibana-7d85dcffc8-bfpfp                         2/2     Running       0          33s
```

- 새 pod는 **ip-10-0-139-48.us-east-2.compute.internal** 노드에 있습니다.

```
$ oc get pod kibana-7d85dcffc8-bfpfp -o wide
```

출력 예

```
NAME                    READY   STATUS       RESTARTS   AGE   IP          NODE
NOMINATED NODE   READINESS GATES
kibana-7d85dcffc8-bfpfp   2/2     Running      0          43s   10.131.0.22   ip-10-0-139-48.us-
east-2.compute.internal   <none>           <none>
```

- 잠시 후 원래 Kibana pod가 제거됩니다.

```
$ oc get pods
```

출력 예

```
NAME                                READY   STATUS    RESTARTS   AGE
cluster-logging-operator-84d98649c4-zb9g7       1/1     Running   0          30m
elasticsearch-cdm-hwv01pf7-1-56588f554f-kpmlg   2/2     Running   0          29m
elasticsearch-cdm-hwv01pf7-2-84c877d75d-75wqj   2/2     Running   0          29m
elasticsearch-cdm-hwv01pf7-3-f5d95b87b-4nx78    2/2     Running   0          29m
fluentd-42dzz                                   1/1     Running   0          29m
fluentd-d74rq                                   1/1     Running   0          29m
fluentd-m5vr9                                   1/1     Running   0          29m
fluentd-nkxl7                                   1/1     Running   0          29m
fluentd-pdvqb                                   1/1     Running   0          29m
fluentd-tflh6                                   1/1     Running   0          29m
kibana-7d85dcffc8-bfpfp                         2/2     Running   0          62s
```

추가 리소스

- OpenShift Container Platform 구성 요소 이동에 대한 일반적인 방법은 모니터링 설명서를 참조하십시오.

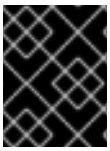
8장. OPENSHIFT CONTAINER PLATFORM 클러스터에 RHEL 컴퓨팅 머신 추가

OpenShift Container Platform에서 RHEL (Red Hat Enterprise Linux) 컴퓨팅 또는 작업자, 머신을 사용자 프로비저닝 인프라 클러스터 또는 설치 프로비저닝 인프라 클러스터에 추가할 수 있습니다. 컴퓨팅 머신에서만 RHEL을 운영 체제로 사용할 수 있습니다.

8.1. 클러스터에 RHEL 컴퓨팅 노드 추가 정보

OpenShift Container Platform 4.6에서는 사용자 프로비저닝 인프라 설치를 사용하는 경우 클러스터에서 Red Hat Enterprise Linux (RHEL) 머신을 컴퓨팅 머신 (작업자 머신이라고도 함)으로 사용하는 옵션이 있습니다. 클러스터의 컨트롤 플레인 또는 마스터 시스템에 RHCOS (Red Hat Enterprise Linux CoreOS) 머신을 사용해야합니다.

사용자 프로비저닝 인프라를 사용하는 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.



중요

클러스터의 시스템에서 OpenShift Container Platform을 제거하려면 운영 체제를 제거해야하므로 클러스터에 추가한 모든 RHEL 머신에 전용 하드웨어를 사용해야합니다.

중요

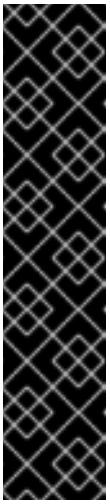
OpenShift Container Platform 클러스터에 추가한 모든 RHEL 머신에서 스왑 메모리가 비활성화됩니다. 이 머신에서 스왑 메모리를 활성화할 수 없습니다.

컨트롤 플레인을 초기화한 후 RHEL 컴퓨팅 머신을 클러스터에 추가해야합니다.

8.2. RHEL 컴퓨팅 노드의 시스템 요구 사항

OpenShift Container Platform 환경에서 RHEL (Red Hat Enterprise Linux) 컴퓨팅 또는 작업자 머신 호스트는 다음과 같은 최소 하드웨어 사양 및 시스템 수준 요구 사항을 충족해야합니다.

- Red Hat 계정에 유효한 OpenShift Container Platform 서브스크립션이 있어야합니다. 서브스크립션이 없는 경우 영업 담당자에게 자세한 내용을 문의하십시오.
- 프로덕션 환경에서 예상 워크로드를 지원할 수 있는 컴퓨팅 머신을 제공해야합니다. 클러스터 관리자는 예상 워크로드를 계산하고 오버 헤드에 약 10 %를 추가해야합니다. 프로덕션 환경의 경우 노드 호스트 장애가 최대 용량에 영향을 미치지 않도록 충분한 리소스를 할당해야 합니다.
- 각 시스템은 다음 하드웨어 요구 사항을 충족해야합니다.
  - 물리적 또는 가상 시스템 또는 퍼블릭 또는 프라이빗 IaaS에서 실행되는 인스턴스.
  - 기본 OS: RHEL 7.9 ("최소"설치 옵션).



중요

OpenShift Container Platform 클러스터에 RHEL 7 컴퓨팅 머신을 추가하는 것은 더 이상 사용되지 않습니다. 더 이상 사용되지 않는 기능은 여전히 OpenShift Container Platform에 포함되어 있으며 계속 지원됩니다. 그러나 이 기능은 향후 릴리스에서 제거될 예정이므로 새로운 배포에는 사용하지 않는 것이 좋습니다.

또한 이 릴리스에서는 지원되지 않으므로 컴퓨팅 머신을 RHEL 8로 업그레이드해서는 안 됩니다.

OpenShift Container Platform에서 더 이상 사용되지 않거나 삭제된 주요 기능의 최신 목록은 OpenShift Container Platform 릴리스 노트에서 *더 이상 사용되지 않고 삭제된 기능* 섹션을 참조하십시오.

- FIPS 모드에서 OpenShift Container Platform을 배포하는 경우 부팅하기 전에 RHEL 시스템에서 FIPS를 활성화해야합니다. RHEL 7 설명서에서 FIPS 모드 활성화 를 참조하십시오.

중요

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

- NetworkManager 1.0 이상
- vCPU 1개
- 최소 8GB RAM
- **/var/**를 포함하는 파일 시스템의 최소 15GB 하드 디스크 공간
- **/usr/local/bin/**을 포함하는 파일 시스템의 최소 1GB 하드 디스크 공간
- 시스템의 임시 디렉토리를 포함하는 파일 시스템의 최소 1GB의 하드 디스크 공간 시스템의 임시 디렉토리는 표준 Python 라이브러리의 tempfile 모듈에 정의된 규칙에 따라 결정됩니다.
  - 각 시스템은 시스템 제공 업체의 추가 요구 사항을 충족해야합니다. 예를 들어 VMware vSphere에 클러스터를 설치하는 경우 스토리지 지침에 따라 디스크를 구성하고 **disk.enableUUID=true** 속성을 설정해야합니다.
  - 각 시스템은 DNS 확인 가능한 호스트 이름을 사용하여 클러스터의 API 끝점에 액세스할 수 있어야 합니다. 모든 네트워크 보안 액세스 제어는 클러스터의 API 서비스 엔드 포인트에 대한 시스템 액세스를 허용해야합니다.

8.2.1. 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.

8.3. 클라우드 이미지 준비

AWS에서 다양한 이미지 형식을 직접 사용할 수 없기 때문에 Amazon 머신 이미지(AMI)가 필요합니다. Red Hat에서 제공하는 AMI를 사용하거나 자체 이미지를 수동으로 가져올 수 있습니다. EC2 인스턴스를 프로비저닝하기 전에 AMI가 있어야 합니다. 컴퓨팅 시스템에 필요한 올바른 RHEL 버전이 선택하려면유효한 AMI ID가 필요합니다.

8.3.1. AWS에서 사용 가능한 최신 RHEL 이미지 나열

AMI ID는 AWS의 기본 부팅 이미지에 해당합니다. AMI는 EC2 인스턴스를 프로비저닝하기 전에 존재해야 하므로 구성 전에 AMI ID를 알아야 합니다. AWS CLI(Command Line Interface)는 사용 가능한 RHEL(Red Hat Enterprise Linux) 이미지 ID를 나열하는 데 사용됩니다.

사전 요구 사항

- AWS CLI를 설치했습니다.

절차

- 이 명령을 사용하여 RHEL 7.9 Amazon 머신 이미지(AMI)를 나열합니다.

```
$ aws ec2 describe-images --owners 309956199498 \ 1
--query 'sort_by(Images, &CreationDate)[*].[CreationDate,Name,ImageId]' \ 2
--filters "Name=name,Values=RHEL-7.9*" \ 3
--region us-east-1 \ 4
--output table 5
```

1. **--owners** 명령 옵션은 계정 ID **309956199498**에 기반한 Red Hat 이미지를 보여줍니다.

   중요

   Red Hat에서 제공하는 이미지의 AMI ID를 표시하려면 이 계정 ID가 필요합니다.

2. **--query** 명령 옵션은 **'sort_by(Images, &CreationDate)[*].[CreationDate,Name,ImageId]** 매개변수로 이미지가 정렬되는 방법을 설정합니다. 이 경우 이미지는 생성 날짜에 따라 정렬되며 테이블은 생성 날짜, 이미지 이름 및 AMI ID를 표시하도록 구성됩니다.
3. **--filter** 명령 옵션은 표시되는 RHEL 버전을 설정합니다. 이 예에서 필터는" **Name=name,Values=RHEL-7.9*"**로 설정되므로 RHEL 7.9 AMI가 표시됩니다.
4. **--region** 명령 옵션은 AMI가 저장된 리전을 설정합니다.
5. **--output** 명령 옵션은 결과가 표시되는 방법을 설정합니다.

참고

AWS용 RHEL 컴퓨팅 머신을 생성할 때 AMI가 RHEL 7.9인지 확인합니다.

출력 예

```
----------------------------------------------------------------------------------------------
|                                       DescribeImages                                       |
+--------------------------+-------------------------------------------------+----------------+
```

```
| 2020-05-13T09:50:36.000Z | RHEL-7.9_HVM_BETA-20200422-x86_64-0-Hourly2-GP2 | ami-
038714142142a6a64 |
| 2020-09-18T07:51:03.000Z | RHEL-7.9_HVM_GA-20200917-x86_64-0-Hourly2-GP2 | ami-
005b7876121b7244d |
| 2021-02-09T09:46:19.000Z | RHEL-7.9_HVM-20210208-x86_64-0-Hourly2-GP2 | ami-
030e754805234517e |
+--------------------------+-----------------------------------------------------+----------------------+
```

추가 리소스

- RHEL 이미지를 AWS로 수동으로 가져올 수도 있습니다.

8.4. PLAYBOOK 실행을 위한 머신 준비

RHEL (Red Hat Enterprise Linux)을 운영 체제로 사용하는 컴퓨팅 머신을 OpenShift Container Platform 4.6 클러스터에 추가하려면 RHEL 7 머신을 준비하여 새 노드를 클러스터에 추가하는 Ansible 플레이북을 실행해야 합니다. 이 머신은 클러스터의 일부가 아니지만 클러스터에 액세스할 수 있어야합니다.

전제 조건

- Playbook을 실행하는 머신에 OpenShift CLI (**oc**)를 설치합니다.
- **cluster-admin** 권한이 있는 사용자로 로그인합니다.

프로세스

1. 클러스터의 **kubeconfig** 파일 및 클러스터 설치에 사용된 설치 프로그램이 머신에 있는지 확인합니다. 이를 수행하는 한 가지 방법으로 클러스터 설치에 사용된 머신과 동일한 머신을 사용하는 것입니다.
2. 컴퓨팅 머신으로 사용하려는 모든 RHEL 호스트에 액세스하도록 머신을 구성합니다. SSH 프록시 또는 VPN을 사용하는 Bastion를 포함하여 회사에서 허용하는 모든 방법을 사용할 수 있습니다.
3. Playbook을 실행하는 머신에서 모든 RHEL 호스트에 대한 SSH 액세스 권한이있는 사용자를 구성하십시오.

중요

SSH 키 기반 인증을 사용하는 경우 SSH 에이전트를 사용하여 키를 관리해야합니다.

4. 아직 등록하지 않은 경우 RHSM으로 머신을 등록하고 **OpenShift** 서브스크립션이 있는 풀을 머신에 연결합니다.
   a. RHSM으로 머신를 등록합니다.

   ```
   # subscription-manager register --username=<user_name> --password=<password>
   ```

   b. RHSM에서 최신 서브스크립션 데이터를 가져옵니다.

   ```
   # subscription-manager refresh
   ```

   c. 사용 가능한 서브스크립션을 나열하십시오.

```
# subscription-manager list --available --matches '*OpenShift*'
```

d. 이전 명령의 출력에서 OpenShift Container Platform 서브스크립션의 풀 ID를 찾아서 이를 연결합니다.

```
# subscription-manager attach --pool=<pool_id>
```

5. OpenShift Container Platform 4.6에 필요한 리포지토리를 활성화합니다.

```
# subscription-manager repos \
    --enable="rhel-7-server-rpms" \
    --enable="rhel-7-server-extras-rpms" \
    --enable="rhel-7-server-ansible-2.9-rpms" \
    --enable="rhel-7-server-ose-4.6-rpms"
```

6. **openshift-ansible**을 포함한 필수 패키지를 설치합니다.

```
# yum install openshift-ansible openshift-clients jq
```

openshift-ansible 패키지는 설치 프로그램 유틸리티를 제공하고 Ansible, Playbook 및 관련 구성 파일과 같이 RHEL 컴퓨팅 노드를 클러스터에 추가하는데 필요한 다른 패키지를 가져옵니다. **openshift-clients**는 **oc** CLI를 제공하고 **jq** 패키지는 명령 행에서 JSON 출력 표시 방법을 개선할 수 있습니다.

8.5. RHEL 컴퓨팅 노드 준비

RHEL (Red Hat Enterprise Linux) 시스템을 OpenShift Container Platform 클러스터에 추가하기 전에 각 호스트를 RHSM (Red Hat Subscription Manager)에 등록하고 활성 OpenShift Container Platform 서브스크립션을 연결하고 필요한 저장소를 활성화해야합니다.

1. 각 호스트에서 RHSM으로 동륵합니다.

```
# subscription-manager register --username=<user_name> --password=<password>
```

2. RHSM에서 최신 서브스크립션 데이터를 가져옵니다.

```
# subscription-manager refresh
```

3. 사용 가능한 서브스크립션을 나열하십시오.

```
# subscription-manager list --available --matches '*OpenShift*'
```

4. 이전 명령의 출력에서 OpenShift Container Platform 서브스크립션의 풀 ID를 찾아서 이를 연결합니다.

```
# subscription-manager attach --pool=<pool_id>
```

5. 모든 yum 저장소를 비활성화합니다.

a. 활성화된 모든 RHSM 저장소를 비활성화합니다.

```
# subscription-manager repos --disable="*"
```

b. 나머지 yum 저장소를 나열하고 **repo id** 아래에 해당 이름을 적어 둡니다.

```
# yum repolist
```

c. **yum-config-manager**를 사용하여 나머지 yum 리포지토리를 비활성화합니다.

```
# yum-config-manager --disable <repo_id>
```

또는 모든 리포지토리를 비활성화합니다.

```
# yum-config-manager --disable \*
```

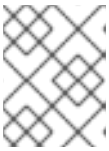
사용 가능한 리포지토리가 많으면 몇 분의 시간이 소요될 수 있습니다.

6. OpenShift Container Platform 4.6에 필요한 리포지토리를 활성화합니다.

```
# subscription-manager repos \
    --enable="rhel-7-server-rpms" \
    --enable="rhel-7-fast-datapath-rpms" \
    --enable="rhel-7-server-extras-rpms" \
    --enable="rhel-7-server-optional-rpms" \
    --enable="rhel-7-server-ose-4.6-rpms"
```

7. 호스트에서 firewalld를 중지하고 비활성화합니다.

```
# systemctl disable --now firewalld.service
```



참고

나중에 firewalld를 활성화할 수 없습니다. 활성화하면 작업자의 OpenShift Container Platform 로그에 액세스할 수 없습니다.

8.6. AWS의 RHEL 인스턴스에 역할 권한 연결

브라우저에서 Amazon IAM 콘솔을 사용하여 필요한 역할을 선택하고 작업자 노드에 할당할 수 있습니다.

절차

1. AWS IAM 콘솔에서 원하는 IAM 역할을 생성합니다.
2. 원하는 작업자 노드에 IAM 역할을 연결합니다. 다음 권한이 필요합니다.
   - **sts:AssumeRole**
   - **ec2:DescribeInstances**
   - **ec2:DescribeRegions**

8.7. RHEL 작업자 노드를 OWNED 또는 SHARED로 태그 지정

클러스터는 **kubernetes.io/cluster/<clusterid>,Value=(owned|shared)** 태그 값을 사용하여 AWS 클러스터와 관련된 리소스의 수명을 결정합니다.

- 클러스터 제거의 일부로 리소스를 제거해야 하는 경우 **owned** 태그 값을 추가해야 합니다.
- 클러스터가 제거된 후에도 리소스가 계속 존재하는 경우 **shared** 태그 값을 추가해야 합니다. 이 태그는 클러스터가 이 리소스를 사용하지만 리소스에 대해 별도의 소유자가 있음을 나타냅니다.

절차

- RHEL 컴퓨팅 머신의 경우 RHEL 작업자 인스턴스는**kubernetes.io/cluster/<clusterid>=owned** 또는 **kubernetes.io/cluster/<cluster-id>=shared**로 태그 지정해야 합니다.

참고

kubernetes.io/cluster/<name>,Value=<clusterid> 태그로 기존 보안 그룹을 태그하지 마십시오. 그렇지 않으면 ELB(Elastic Load Balancing)에서 로드 밸런서를 생성할 수 없습니다.

8.8. 클러스터에 RHEL 컴퓨팅 머신 추가

Red Hat Enterprise Linux를 운영 체제로 사용하는 컴퓨팅 머신을 OpenShift Container Platform 4.6 클러스터에 추가할 수 있습니다.

전제 조건

- Playbook을 실행하는 머신에 필요한 패키지를 설치하고 필요한 구성이 수행되어 있습니다.
- RHEL 호스트 설치가 준비되어 있습니다.

프로세스

Playbook을 실행할 준비가 되어 있는 머신에서 다음 단계를 수행합니다.

1. 컴퓨팅 머신 호스트 및 필수 변수를 정의하는 **/<path>/inventory/hosts**라는 Ansible 인벤토리 파일을 만듭니다.

```
[all:vars]
ansible_user=root 1
#ansible_become=True 2

openshift_kubeconfig_path="~/.kube/config" 3

[new_workers] 4
mycluster-rhel7-0.example.com
mycluster-rhel7-1.example.com
```

1. 원격 컴퓨팅 머신에서 Ansible 태스크를 실행하는 사용자 이름을 지정합니다.
2. **ansible_user**의 **root**를 지정하지 않으면 **ansible_become**을 **True**로 설정하고 사용자 sudo 권한을 지정해야합니다.
3. 클러스터 **kubeconfig** 파일의 경로와 파일 이름을 지정합니다.
4. 클러스터에 추가할 각 RHEL 머신을 나열합니다. 각 호스트에 대해 정규화된 도메인 이름을 지정해야합니다. 이 이름은 클러스터가 시스템에 액세스하는 데 사용하는 호스트 이름이므로 올바른 공용 또는 개인 이름을 설정하여 시스템에 액세스합니다.

2. Ansible Playbook 디렉토리로 이동합니다.

```
$ cd /usr/share/ansible/openshift-ansible
```

3. Playbook을 실행합니다.

```
$ ansible-playbook -i /<path>/inventory/hosts playbooks/scaleup.yml 1
```

1 **<path>**에 대해 생성한 Ansible 인벤토리 파일의 경로를 지정합니다.

8.9. 시스템의 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

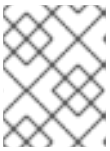
1. 클러스터가 시스템을 인식하는지 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0 Ready    master  63m  v1.19.0
master-1 Ready    master  63m  v1.19.0
master-2 Ready    master  64m  v1.19.0
```

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  v1.20.0
master-1  Ready     master  73m  v1.20.0
master-2  Ready     master  74m  v1.20.0
worker-0  Ready     worker  11m  v1.20.0
worker-1  Ready     worker  11m  v1.20.0
```

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.

8.10. ANSIBLE 호스트 파일의 필수 매개 변수

RHEL (Red Hat Enterprise Linux) 컴퓨팅 머신을 클러스터에 추가하기 전에 Ansible 호스트 파일에서 다음 매개 변수를 정의해야합니다.

매개 변수	설명	값
ansible_user	암호없이 SSH 기반 인증을 허용하는 SSH 사용자입니다. SSH 키 기반 인증을 사용하는 경우 SSH 에이전트를 사용하여 키를 관리해야합니다.	시스템의 사용자 이름입니다. 기본값은 **root**입니다.
ansible_become	**ansible_user** 값이 root가 아닌 경우 **ansible_become**을 **True**로 설정하고 **ansible_user**로 지정하는 사용자는 암호없이 sudo 액세스를 구성해야합니다.	**True**. 값이 **True** 가 아닌 경우 이 매개 변수를 지정하거나 정의하지 마십시오.
openshift_kubeconfig_path	클러스터의 **kubeconfig** 파일이 포함된 로컬 디렉토리의 경로와 파일 이름을 지정합니다.	구성 파일의 경로 및 이름

8.10.1. 선택 사항: 클러스터에서 RHCOS 컴퓨팅 머신 제거

RHEL (Red Hat Enterprise Linux) 컴퓨팅 머신을 클러스터에 추가 한 후 선택 옵션으로 RHCOS (Red Hat Enterprise Linux CoreOS) 컴퓨팅 머신을 제거하여 리소스를 확보할 수 있습니다.

전제 조건

- RHEL 컴퓨팅 머신이 클러스터에 추가되어 있습니다.

프로세스

1. 머신 목록을 표시하고 RHCOS 컴퓨팅 머신의 노드 이름을 기록합니다.

```
$ oc get nodes -o wide
```

2. 각 RHCOS 컴퓨팅 머신의 노드를 제거합니다.
   a. **oc adm cordon** 명령을 실행하여 노드를 스케줄 예약 해제로 표시합니다.

```
$ oc adm cordon <node_name> 1
```

   1 RHCOS 컴퓨팅 머신의 노드 이름을 지정합니다.

   b. 노드에서 모든 pod를 드레인합니다.

```
$ oc adm drain <node_name> --force --delete-local-data --ignore-daemonsets 1
```

   1 분리 한 RHCOS 컴퓨팅 머신의 노드 이름을 지정합니다.

   c. 노드를 제거합니다.

```
$ oc delete nodes <node_name> 1
```

1 드레인한 RHCOS 컴퓨팅 머신의 노드 이름을 지정합니다.

3. 컴퓨팅 머신 목록을 확인하고 RHEL 노드만 남아 있는지 확인합니다.

```
$ oc get nodes -o wide
```

4. 클러스터 컴퓨팅 머신의 로드 밸런서에서 RHCOS 머신을 제거합니다. 가상 머신을 삭제하거나 RHCOS 컴퓨팅 머신의 실제 하드웨어를 다시 이미지화 할 수 있습니다.

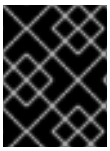
9장. OPENSHIFT CONTAINER PLATFORM 클러스터에 RHEL 컴퓨팅 머신 추가

OpenShift Container Platform 클러스터에 작업자 시스템이라고도하는 RHEL (Red Hat Enterprise Linux) 컴퓨팅 시스템이 이미 포함되어있는 경우 RHEL 컴퓨팅 시스템을 더 추가할 수 있습니다.

9.1. 클러스터에 RHEL 컴퓨팅 노드 추가 정보

OpenShift Container Platform 4.6에서는 사용자 프로비저닝 인프라 설치를 사용하는 경우 클러스터에서 Red Hat Enterprise Linux (RHEL) 머신을 컴퓨팅 머신 (작업자 머신이라고도 함)으로 사용하는 옵션이 있습니다. 클러스터의 컨트롤 플레인 또는 마스터 시스템에 RHCOS (Red Hat Enterprise Linux CoreOS) 머신을 사용해야합니다.

사용자 프로비저닝 인프라를 사용하는 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.



중요

클러스터의 시스템에서 OpenShift Container Platform을 제거하려면 운영 체제를 제거해야하므로 클러스터에 추가한 모든 RHEL 머신에 전용 하드웨어를 사용해야합니다.

중요

OpenShift Container Platform 클러스터에 추가한 모든 RHEL 머신에서 스왑 메모리가 비활성화됩니다. 이 머신에서 스왑 메모리를 활성화할 수 없습니다.

컨트롤 플레인을 초기화한 후 RHEL 컴퓨팅 머신을 클러스터에 추가해야합니다.

9.2. RHEL 컴퓨팅 노드의 시스템 요구 사항

OpenShift Container Platform 환경에서 RHEL (Red Hat Enterprise Linux) 컴퓨팅 또는 작업자 머신 호스트는 다음과 같은 최소 하드웨어 사양 및 시스템 수준 요구 사항을 충족해야합니다.

- Red Hat 계정에 유효한 OpenShift Container Platform 서브스크립션이 있어야합니다. 서브스크립션이 없는 경우 영업 담당자에게 자세한 내용을 문의하십시오.
- 프로덕션 환경에서 예상 워크로드를 지원할 수 있는 컴퓨팅 머신을 제공해야합니다. 클러스터 관리자는 예상 워크로드를 계산하고 오버 헤드에 약 10 %를 추가해야합니다. 프로덕션 환경의 경우 노드 호스트 장애가 최대 용량에 영향을 미치지 않도록 충분한 리소스를 할당해야 합니다.
- 각 시스템은 다음 하드웨어 요구 사항을 충족해야합니다.
  - 물리적 또는 가상 시스템 또는 퍼블릭 또는 프라이빗 IaaS에서 실행되는 인스턴스.
  - 기본 OS: RHEL 7.9 ("최소"설치 옵션).

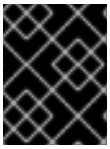
중요

OpenShift Container Platform 클러스터에 RHEL 7 컴퓨팅 머신을 추가하는 것은 더 이상 사용되지 않습니다. 더 이상 사용되지 않는 기능은 여전히 OpenShift Container Platform에 포함되어 있으며 계속 지원됩니다. 그러나 이 기능은 향후 릴리스에서 제거될 예정이므로 새로운 배포에는 사용하지 않는 것이 좋습니다.

또한 이 릴리스에서는 지원되지 않으므로 컴퓨팅 머신을 RHEL 8로 업그레이드해서는 안 됩니다.

OpenShift Container Platform에서 더 이상 사용되지 않거나 삭제된 주요 기능의 최신 목록은 OpenShift Container Platform 릴리스 노트에서 *더 이상 사용되지 않고 삭제된 기능* 섹션을 참조하십시오.

- FIPS 모드에서 OpenShift Container Platform을 배포하는 경우 부팅하기 전에 RHEL 시스템에서 FIPS를 활성화해야합니다. RHEL 7 설명서에서 FIPS 모드 활성화 를 참조하십시오.



중요

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

- NetworkManager 1.0 이상
- vCPU 1개
- 최소 8GB RAM
- **/var/**를 포함하는 파일 시스템의 최소 15GB 하드 디스크 공간
- **/usr/local/bin/**을 포함하는 파일 시스템의 최소 1GB 하드 디스크 공간
- 시스템의 임시 디렉토리를 포함하는 파일 시스템의 최소 1GB의 하드 디스크 공간 시스템의 임시 디렉토리는 표준 Python 라이브러리의 tempfile 모듈에 정의된 규칙에 따라 결정됩니다.
    - 각 시스템은 시스템 제공 업체의 추가 요구 사항을 충족해야합니다. 예를 들어 VMware vSphere에 클러스터를 설치하는 경우 스토리지 지침에 따라 디스크를 구성하고 **disk.enableUUID=true** 속성을 설정해야합니다.
    - 각 시스템은 DNS 확인 가능한 호스트 이름을 사용하여 클러스터의 API 끝점에 액세스할 수 있어야 합니다. 모든 네트워크 보안 액세스 제어는 클러스터의 API 서비스 엔드 포인트에 대한 시스템 액세스를 허용해야합니다.

9.2.1. 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.

9.3. 클라우드 이미지 준비

AWS에서 다양한 이미지 형식을 직접 사용할 수 없기 때문에 Amazon 머신 이미지(AMI)가 필요합니다. Red Hat에서 제공하는 AMI를 사용하거나 자체 이미지를 수동으로 가져올 수 있습니다. EC2 인스턴스를 프로비저닝하기 전에 AMI가 있어야 합니다. 컴퓨팅 시스템에 필요한 올바른 RHEL 버전이 선택되도록 AMI ID를 나열해야 합니다.

9.3.1. AWS에서 사용 가능한 최신 RHEL 이미지 나열

AMI ID는 AWS의 기본 부팅 이미지에 해당합니다. AMI는 EC2 인스턴스를 프로비저닝하기 전에 존재해야 하므로 구성 전에 AMI ID를 알아야 합니다. AWS CLI(Command Line Interface)는 사용 가능한 RHEL(Red Hat Enterprise Linux) 이미지 ID를 나열하는 데 사용됩니다.

사전 요구 사항

- AWS CLI를 설치했습니다.

절차

- 이 명령을 사용하여 RHEL 7.9 Amazon 머신 이미지(AMI)를 나열합니다.

```
$ aws ec2 describe-images --owners 309956199498 \ 1
--query 'sort_by(Images, &CreationDate)[*].[CreationDate,Name,ImageId]' \ 2
--filters "Name=name,Values=RHEL-7.9*" \ 3
--region us-east-1 \ 4
--output table 5
```

1. **--owners** 명령 옵션은 계정 ID **309956199498**에 기반한 Red Hat 이미지를 보여줍니다.

   중요

   Red Hat에서 제공하는 이미지의 AMI ID를 표시하려면 이 계정 ID가 필요합니다.

2. **--query** 명령 옵션은 **'sort_by(Images, &CreationDate)[*].[CreationDate,Name,ImageId]** 매개변수로 이미지가 정렬되는 방법을 설정합니다. 이 경우 이미지는 생성 날짜에 따라 정렬되며 테이블은 생성 날짜, 이미지 이름 및 AMI ID를 표시하도록 구성됩니다.
3. **--filter** 명령 옵션은 표시되는 RHEL 버전을 설정합니다. 이 예에서 필터는" **Name=name,Values=RHEL-7.9*"**로 설정되므로 RHEL 7.9 AMI가 표시됩니다.
4. **--region** 명령 옵션은 AMI가 저장된 리전을 설정합니다.
5. **--output** 명령 옵션은 결과가 표시되는 방법을 설정합니다.

참고

AWS용 RHEL 컴퓨팅 머신을 생성할 때 AMI가 RHEL 7.9인지 확인합니다.

출력 예

```
----------------------------------------------------------------------------------------------
|                                       DescribeImages                                       |
+--------------------------+-------------------------------------------------+-----------------+
```

```
| 2020-05-13T09:50:36.000Z | RHEL-7.9_HVM_BETA-20200422-x86_64-0-Hourly2-GP2 | ami-
038714142142a6a64 |
| 2020-09-18T07:51:03.000Z | RHEL-7.9_HVM_GA-20200917-x86_64-0-Hourly2-GP2 | ami-
005b7876121b7244d |
| 2021-02-09T09:46:19.000Z | RHEL-7.9_HVM-20210208-x86_64-0-Hourly2-GP2 | ami-
030e754805234517e |
+--------------------------+-----------------------------------------------------+----------------------+
```

추가 리소스

- RHEL 이미지를 AWS로 수동으로 가져올 수도 있습니다.

9.4. RHEL 컴퓨팅 노드 준비

RHEL (Red Hat Enterprise Linux) 시스템을 OpenShift Container Platform 클러스터에 추가하기 전에 각 호스트를 RHSM (Red Hat Subscription Manager)에 등록하고 활성 OpenShift Container Platform 서브스크립션을 연결하고 필요한 저장소를 활성화해야합니다.

1. 각 호스트에서 RHSM으로 동록합니다.

```
# subscription-manager register --username=<user_name> --password=<password>
```

2. RHSM에서 최신 서브스크립션 데이터를 가져옵니다.

```
# subscription-manager refresh
```

3. 사용 가능한 서브스크립션을 나열하십시오.

```
# subscription-manager list --available --matches '*OpenShift*'
```

4. 이전 명령의 출력에서 OpenShift Container Platform 서브스크립션의 풀 ID를 찾아서 이를 연결합니다.

```
# subscription-manager attach --pool=<pool_id>
```

5. 모든 yum 저장소를 비활성화합니다.
   a. 활성화된 모든 RHSM 저장소를 비활성화합니다.

```
# subscription-manager repos --disable="*"
```

   b. 나머지 yum 저장소를 나열하고 **repo id** 아래에 해당 이름을 적어 둡니다.

```
# yum repolist
```

   c. **yum-config-manager**를 사용하여 나머지 yum 리포지토리를 비활성화합니다.

```
# yum-config-manager --disable <repo_id>
```

   또는 모든 리포지토리를 비활성화합니다.

```
# yum-config-manager --disable \*
```

사용 가능한 리포지토리가 많으면 몇 분의 시간이 소요될 수 있습니다.

6. OpenShift Container Platform 4.6에 필요한 리포지토리를 활성화합니다.

```
# subscription-manager repos \
    --enable="rhel-7-server-rpms" \
    --enable="rhel-7-fast-datapath-rpms" \
    --enable="rhel-7-server-extras-rpms" \
    --enable="rhel-7-server-optional-rpms" \
    --enable="rhel-7-server-ose-4.6-rpms"
```

7. 호스트에서 firewalld를 중지하고 비활성화합니다.

```
# systemctl disable --now firewalld.service
```

참고

나중에 firewalld를 활성화할 수 없습니다. 활성화하면 작업자의 OpenShift Container Platform 로그에 액세스할 수 없습니다.

9.5. AWS의 RHEL 인스턴스에 역할 권한 연결

브라우저에서 Amazon IAM 콘솔을 사용하여 필요한 역할을 선택하고 작업자 노드에 할당할 수 있습니다.

절차

1. AWS IAM 콘솔에서 원하는 IAM 역할을 생성합니다.
2. 원하는 작업자 노드에 IAM 역할을 연결합니다. 다음 권한이 필요합니다.
   - **sts:AssumeRole**
   - **ec2:DescribeInstances**
   - **ec2:DescribeRegions**

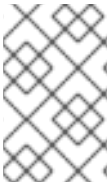
9.6. RHEL 작업자 노드를 OWNED 또는 SHARED로 태그 지정

클러스터는 **kubernetes.io/cluster/<clusterid>,Value=(owned|shared)** 태그 값을 사용하여 AWS 클러스터와 관련된 리소스의 수명을 결정합니다.

- 클러스터 제거의 일부로 리소스를 제거해야 하는 경우 **owned** 태그 값을 추가해야 합니다.
- 클러스터가 제거된 후에도 리소스가 계속 존재하는 경우 **shared** 태그 값을 추가해야 합니다. 이 태그는 클러스터가 이 리소스를 사용하지만 리소스에 대해 별도의 소유자가 있음을 나타냅니다.

절차

- RHEL 컴퓨팅 머신의 경우 RHEL 작업자 인스턴스는**kubernetes.io/cluster/<clusterid>=owned** 또는 **kubernetes.io/cluster/<cluster-id>=shared**로 태그 지정해야 합니다.



참고

kubernetes.io/cluster/<name>,Value=<clusterid> 태그로 기존 보안 그룹을 태그하지 마십시오. 그렇지 않으면 ELB(Elastic Load Balancing)에서 로드 밸런서를 생성할 수 없습니다.

9.7. 클러스터에 더 많은 RHEL 컴퓨팅 시스템 업데이트 추가

RHEL(Red Hat Enterprise Linux)을 운영 체제로 사용하는 컴퓨팅 머신을 OpenShift Container Platform 4.6 클러스터에 더 추가할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform 클러스터에는 이미 RHEL 컴퓨팅 노드가 포함되어 있습니다.
- 첫 번째 RHEL 컴퓨팅 머신을 클러스터에 추가하는 데 사용한 **hosts** 파일은 Playbook을 실행하는 머신에 있습니다.
- Playbook을 실행하는 컴퓨터는 모든 RHEL 호스트에 액세스할 수 있어야합니다. SSH 프록시 또는 VPN을 사용하는 Bastion를 포함하여 회사에서 허용하는 모든 방법을 사용할 수 있습니다.
- 클러스터의 **kubeconfig** 파일 및 클러스터를 설치에 사용된 설치 프로그램은 Playbook을 실행하는 머신에 있습니다.
- 설치를 위해 RHEL 호스트를 준비해야합니다.
- Playbook을 실행하는 머신에서 모든 RHEL 호스트에 대한 SSH 액세스 권한이있는 사용자를 구성하십시오.
- SSH 키 기반 인증을 사용하는 경우 SSH 에이전트를 사용하여 키를 관리해야합니다.
- Playbook을 실행하는 머신에 OpenShift CLI (**oc**)를 설치합니다.

프로세스

1. 컴퓨팅 시스템 호스트 및 필수 변수를 정의하는 **/<path>/inventory/hosts**에서 Ansible 인벤토리 파일을 엽니 다.
2. 파일의 **[new_workers]** 섹션 이름을 **[workers]**로 변경합니다.
3. **[new_workers]** 섹션을 파일에 추가하고 각각의 새 호스트의 정규화된 도메인 이름을 정의합니다. 파일은 다음 예제와 유사합니다.

```
[all:vars]
ansible_user=root
#ansible_become=True

openshift_kubeconfig_path="~/.kube/config"

[workers]
mycluster-rhel7-0.example.com
mycluster-rhel7-1.example.com

[new_workers]
mycluster-rhel7-2.example.com
mycluster-rhel7-3.example.com
```

-

이 예에서 **mycluster-rhel7-0.example.com** 및 **mycluster-rhel7-1.example.com** 시스템은 클러스터에 있으며 **mycluster-rhel7-2.example.com** 및 **mycluster-rhel7-3.example.com** 머신을 추가합니다.

4. Ansible Playbook 디렉토리로 이동합니다.

```
$ cd /usr/share/ansible/openshift-ansible
```

5. 스케일 업 Playbook을 실행합니다.

```
$ ansible-playbook -i /<path>/inventory/hosts playbooks/scaleup.yml 1
```

1 **<path>**에 대해 생성한 Ansible 인벤토리 파일의 경로를 지정합니다.

9.8. 시스템의 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

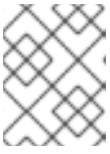
1. 클러스터가 시스템을 인식하는지 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0 Ready    master  63m  v1.19.0
master-1 Ready    master  63m  v1.19.0
master-2 Ready    master  64m  v1.19.0
```

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.

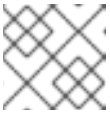
- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ 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  v1.20.0
master-1  Ready    master  73m  v1.20.0
master-2  Ready    master  74m  v1.20.0
worker-0  Ready    worker  11m  v1.20.0
worker-1  Ready    worker  11m  v1.20.0
```

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.

9.9. ANSIBLE 호스트 파일의 필수 매개 변수

RHEL (Red Hat Enterprise Linux) 컴퓨팅 머신을 클러스터에 추가하기 전에 Ansible 호스트 파일에서 다음 매개 변수를 정의해야합니다.

매개 변수	설명	값
ansible_user	암호없이 SSH 기반 인증을 허용하는 SSH 사용자입니다. SSH 키 기반 인증을 사용하는 경우 SSH 에이전트를 사용하여 키를 관리해야합니다.	시스템의 사용자 이름입니다. 기본값은 **root**입니다.
ansible_become	**ansible_user** 값이 root가 아닌 경우 **ansible_become**을 **True**로 설정하고 **ansible_user**로 지정하는 사용자는 암호없이 sudo 액세스를 구성해야합니다.	**True**. 값이 **True** 가 아닌 경우 이 매개 변수를 지정하거나 정의하지 마십시오.
openshift_kubeconfig_path	클러스터의 **kubeconfig** 파일이 포함된 로컬 디렉토리의 경로와 파일 이름을 지정합니다.	구성 파일의 경로 및 이름

10장. 사용자 프로비저닝 인프라

10.1. 사용자 프로비저닝 인프라가 있는 클러스터에 컴퓨팅 머신 추가

설치 프로세스의 일부 또는 설치 후 사용자 프로비저닝 인프라의 클러스터에 컴퓨팅 머신을 추가할 수 있습니다. 설치 후 프로세스에는 설치 중에 사용된 것과 동일한 구성 파일과 매개 변수 중 일부가 필요합니다.

10.1.1. Amazon Web Services에 컴퓨팅 머신 추가

AWS(Amazon Web Services)의 OpenShift Container Platform 클러스터에 컴퓨팅 머신을 추가하려면 CloudFormation 템플릿을 사용하여 AWS에 컴퓨팅 머신 추가 를 참조하십시오.

10.1.2. Microsoft Azure에 컴퓨팅 머신 추가

Microsoft Azure의 OpenShift Container Platform 클러스터에 컴퓨팅 머신을 추가하려면 Azure에서 추가 작업자 머신 생성을 참조하십시오.

10.1.3. Google Cloud Platform에 컴퓨팅 머신 추가

GCP(Google Cloud Platform)의 OpenShift Container Platform 클러스터에 컴퓨팅 머신을 추가하려면 GCP에서 추가 작업자 머신 생성 을 참조하십시오.

10.1.4. vSphere에 컴퓨팅 머신 추가

vSphere의 OpenShift Container Platform 클러스터에 컴퓨팅 머신을 추가하려면 vSphere에 컴퓨팅 머신 추가를 참조하십시오.

10.1.5. 베어 메탈에 컴퓨팅 머신 추가

베어 메탈의 OpenShift Container Platform 클러스터에 컴퓨팅 머신을 추가하려면 베어 메탈에 컴퓨팅 머신 추가를 참조하십시오.

10.2. CLOUDFORMATION 템플릿을 사용하여 AWS에 컴퓨팅 머신 추가

샘플 CloudFormation 템플릿을 사용하여 생성한 AWS(Amazon Web Services)의 OpenShift Container Platform 클러스터에 더 많은 컴퓨팅 머신을 추가할 수 있습니다.

10.2.1. 사전 요구 사항

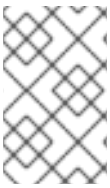
- 제공된 AWS CloudFormation 템플릿을 사용하여 AWS에 클러스터가 설치되어 있어야 합니다.
- 클러스터 설치 중에 컴퓨팅 시스템을 생성하는 데 사용한 JSON 파일 및 CloudFormation 템플릿이 있습니다. 이러한 파일이 없는 경우 설치 절차에 따라 파일을 재생성해야 합니다.

10.2.2. CloudFormation 템플릿을 사용하여 AWS 클러스터에 추가

샘플 CloudFormation 템플릿을 사용하여 생성한 AWS(Amazon Web Services)의 OpenShift Container Platform 클러스터에 더 많은 컴퓨팅 머신을 추가할 수 있습니다.

중요

CloudFormation 템플릿은 하나의 컴퓨팅 시스템을 나타내는 스택을 생성합니다. 각 컴퓨팅 시스템의 스택을 생성해야 합니다.



참고

컴퓨팅 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

- CloudFormation 템플릿을 사용하여 OpenShift Container Platform 클러스터를 설치하고 클러스터 설치 중에 컴퓨팅 시스템을 생성하는 데 사용한 JSON 파일 및 CloudFormation 템플릿에 액세스할 수 있습니다.
- AWS CLI를 설치했습니다.

절차

1. 다른 컴퓨팅 스택을 생성합니다.
   a. 템플릿을 시작합니다.

```
$ 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-workers**). 클러스터를 제거하는 경우 이 스택의 이름을 제공해야 합니다.

   2 **<template>**은 저장한 CloudFormation 템플릿 YAML 파일의 상대 경로 및 이름입니다.

   3 **<parameters>**는 CloudFormation 매개변수 JSON 파일의 상대 경로 및 이름입니다.

   b. 템플릿 구성 요소가 있는지 확인합니다.

```
$ aws cloudformation describe-stacks --stack-name <name>
```

2. 클러스터에 충분한 컴퓨팅 시스템을 생성할 때까지 계속해서 컴퓨팅 스택을 생성합니다.

10.2.3. 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  v1.19.0
master-1 Ready    master  63m  v1.19.0
master-2 Ready    master  64m  v1.19.0
```

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.

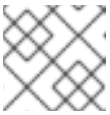
- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ 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.

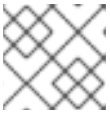
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

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  v1.20.0
master-1 Ready    master  73m  v1.20.0
master-2 Ready    master  74m  v1.20.0
worker-0 Ready    worker  11m  v1.20.0
worker-1 Ready    worker  11m  v1.20.0
```



참고

머신이 **Ready** 상태로 전환하는 데 서버 CSR의 승인 후 몇 분이 걸릴 수 있습니다.

추가 정보

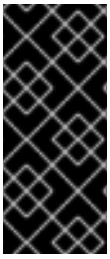
- CSR에 대한 자세한 내용은 인증서 서명 요청을 참조하십시오.

10.3. VSPHERE에 컴퓨팅 머신 추가

VMware vSphere의 OpenShift Container Platform 클러스터에 더 많은 컴퓨팅 머신을 추가할 수 있습니다.

10.3.1. 전제 조건

- vSphere에 클러스터가 설치되어 있어야 합니다.
- 클러스터를 만드는 데 사용한 설치 미디어 및 Red Hat Enterprise Linux CoreOS (RHCOS) 이미지가 있습니다. 이러한 파일이 없는 경우 설치 절차에 따라 파일을 가져와야 합니다.



중요

클러스터를 생성하는 데 사용되는 RHCOS(Red Hat Enterprise Linux CoreOS) 이미지에 액세스할 수 없는 경우 최신 버전의 RHCOS(Red Hat Enterprise Linux CoreOS) 이미지를 사용하여 OpenShift Container Platform 클러스터에 더 많은 컴퓨팅 머신을 추가할 수 있습니다. 자세한 내용은 OpenShift 4.6+로 업그레이드한 후 UPI 클러스터에 새 노드 추가 실패를 참조하십시오.

10.3.2. vSphere에서 RHCOS(Red Hat Enterprise Linux CoreOS) 머신 추가 생성

VMware vSphere에서 사용자 프로비저닝 인프라를 사용하는 클러스터에 대해 추가로 컴퓨팅 머신을 생성할 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. 선택 사항: **Select storage(스토리지 선택)** 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 대한 base64로 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 VM의 전원을 켭니다.
2. 계속해서 클러스터에 추가 컴퓨팅 머신을 만듭니다.

10.3.3. 시스템의 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  v1.19.0
master-1 Ready    master  63m  v1.19.0
master-2 Ready    master  64m  v1.19.0
```

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

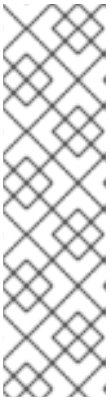
```
NAME        AGE    REQUESTOR                                                    CONDITION
csr-8b2br   15m    system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
csr-8vnps   15m    system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
...
```

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.

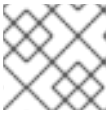
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

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  v1.20.0
master-1 Ready    master  73m  v1.20.0
master-2 Ready    master  74m  v1.20.0
worker-0 Ready    worker  11m  v1.20.0
worker-1 Ready    worker  11m  v1.20.0
```



참고

머신이 **Ready** 상태로 전환하는 데 서버 CSR의 승인 후 몇 분이 걸릴 수 있습니다.

추가 정보

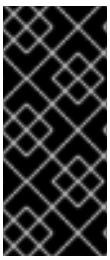
- CSR에 대한 자세한 내용은 인증서 서명 요청을 참조하십시오.

10.4. 베어 메탈에 컴퓨팅 머신 추가

베어 메탈의 OpenShift Container Platform 클러스터에 더 많은 컴퓨팅 머신을 추가할 수 있습니다.

10.4.1. 전제 조건

- 베어 메탈에 클러스터가 설치되어 있어야 합니다.
- 클러스터를 만드는 데 사용한 설치 미디어 및 Red Hat Enterprise Linux CoreOS (RHCOS) 이미지가 있습니다. 이러한 파일이 없는 경우 설치 절차에 따라 파일을 가져와야 합니다.



중요

클러스터를 생성하는 데 사용되는 RHCOS(Red Hat Enterprise Linux CoreOS) 이미지에 액세스할 수 없는 경우 최신 버전의 RHCOS(Red Hat Enterprise Linux CoreOS) 이미지를 사용하여 OpenShift Container Platform 클러스터에 더 많은 컴퓨팅 머신을 추가할 수 있습니다. 자세한 내용은 OpenShift 4.6+로 업그레이드한 후 UPI 클러스터에 새 노드 추가 실패를 참조하십시오.

10.4.2. RHCOS(Red Hat Enterprise Linux CoreOS) 시스템 생성

베어메탈 인프라에 설치된 클러스터에 컴퓨팅 머신을 추가하기 전에 사용할 RHCOS 머신을 생성해야 합니다. ISO 이미지 또는 네트워크 PXE 부팅을 사용하여 시스템을 생성합니다.

참고

클러스터에 새 노드를 모두 배포하기 위해 클러스터를 설치하는 데 사용한 것과 동일한 ISO 이미지를 사용해야 합니다. 동일한 Ignition 구성 파일을 사용하는 것이 좋습니다. 워크로드를 실행하기 전에 노드가 처음 부팅 시 자동으로 업그레이드됩니다. 업그레이드 전후에 노드를 추가할 수 있습니다.

10.4.2.1. ISO 이미지를 사용하여 추가 RHCOS 머신 생성

ISO 이미지를 사용하여 머신을 생성함으로써 베어 메탈 클러스터에 대해 더 많은 Red Hat Enterprise Linux CoreOS (RHCOS) 컴퓨팅 머신을 생성할 수 있습니다.

사전 요구 사항

- 클러스터의 컴퓨팅 머신에 대한 Ignition 구성 파일의 URL을 가져옵니다. 설치 중에 이 파일은 HTTP 서버에 업로드되어 있어야 합니다.

절차

1. ISO 파일을 사용하여 추가 컴퓨팅 머신에 RHCOS를 설치합니다. 클러스터를 설치하기 전에 머신을 만들 때 사용한 것과 동일한 방법을 사용합니다.
   - ISO 이미지를 디스크에 굽고 직접 부팅합니다.
   - LOM 인터페이스에서 ISO 리디렉션을 사용합니다.
2. 인스턴스 시작 후 **TAB** 또는 **E** 키를 눌러 커널 명령 줄을 편집합니다.
3. 커널 명령 줄에 매개 변수를 추가합니다.

```
coreos.inst.install_dev=sda 1
coreos.inst.ignition_url=http://example.com/worker.ign 2
```

1. 설치할 시스템의 블록 장치를 지정합니다.
2. 컴퓨팅 Ignition 구성 파일의 URL을 지정합니다. HTTP 및 HTTPS 프로토콜만 지원됩니다.

4. **Enter**를 눌러 설치를 완료합니다. RHCOS가 설치되면 시스템을 재부팅합니다. 시스템이 재부팅되는 동안 지정된 Ignition 구성 파일을 적용합니다.
5. 계속해서 클러스터에 추가 컴퓨팅 머신을 만듭니다.

10.4.2.2. PXE 또는 iPXE 부팅을 통해 RHCOS 머신 생성

PXE 또는 iPXE 부팅을 사용하여 베어 메탈 클러스터에 대해 추가 Red Hat Enterprise Linux CoreOS (RHCOS) 컴퓨팅 머신을 생성할 수 있습니다.

사전 요구 사항

- 클러스터의 컴퓨팅 머신에 대한 Ignition 구성 파일의 URL을 가져옵니다. 설치 중에 이 파일은 HTTP 서버에 업로드되어 있어야 합니다.
- 클러스터 설치 중에 HTTP 서버에 업로드 한 RHCOS ISO 이미지, 압축된 메탈 BIOS, **kernel** 및 **initramfs** 파일의 URL을 가져옵니다.

- 설치 중에 OpenShift Container Platform 클러스터에 대한 머신을 생성하는 데 사용한 PXE 부팅 인프라에 액세스할 수 있습니다. RHCOS가 설치된 후 로컬 디스크에서 머신을 부팅해야합니다.
- UEFI를 사용하는 경우 OpenShift Container Platform 설치 중에 수정 한 **grub.conf** 파일에 액세스할 수 있습니다.

프로세스

1. RHCOS 이미지의 PXE 또는 iPXE가 올바르게 설치되었는지 확인합니다.
   - 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.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/worker.ign
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
<architecture>.img (2)
```

(1) HTTP 서버에 업로드한 라이브 **kernel** 파일의 위치를 지정합니다.

(2) HTTP 서버에 업로드한 RHCOS 파일의 위치를 지정합니다. **initrd** 매개변수 값은 **initramfs** 파일의 위치이고 **coreos.inst.ignition_url** 매개변수 값은 작업자 Ignition 설정 파일의 위치이며 **coreos.live.rootfs_url** 매개 변수 값은 라이브 **rootfs** 파일의 위치입니다. **coreos.inst.ignition_url** 및 **coreos.live.rootfs_url** 매개변수는 HTTP 및 HTTPS만 지원합니다.

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.

   - iPXE의 경우 :

```
kernel http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> initrd=main
coreos.inst.install_dev=/dev/sda coreos.inst.ignition_url=http://<HTTP_server>/worker.ign
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.<architecture>.img
(1)
initrd --name main http://<HTTP_server>/rhcos-<version>-live-initramfs.<architecture>.img
(2)
```

(1) HTTP 서버에 업로드한 RHCOS 파일의 위치를 지정합니다. **kernel** 매개변수 값은 **kernel** 파일의 위치이고 **initrd=main** 매개변수는 UEFI 시스템에서 부팅하는 데 필요하며 **coreos.inst.ignition_url** 매개 변수 값은 작업자 Ignition 설정 파일의 위치이며, **coreos.live.rootfs_url** 매개 변수 값은 라이브 **rootfs** 파일의 위치입니다. **coreos.inst.ignition_url** 및 **coreos.live.rootfs_url** 매개변수는 HTTP 및 HTTPS만 지원합니다.

(2) HTTP 서버에 업로드한 **initramfs** 파일의 위치를 지정합니다.

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.

1. PXE 또는 iPXE 인프라를 사용하여 클러스터에 필요한 컴퓨팅 머신을 만듭니다.

10.4.3. 시스템의 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

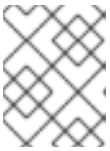
1. 클러스터가 시스템을 인식하는지 확인합니다.

```
$ oc get nodes
```

출력 예

```
NAME     STATUS   ROLES   AGE  VERSION
master-0 Ready    master  63m  v1.19.0
master-1 Ready    master  63m  v1.19.0
master-2 Ready    master  64m  v1.19.0
```

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.

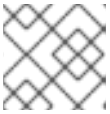
     ```
     $ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
     {{end}}{{end}}' | xargs oc adm certificate approve
     ```

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  v1.20.0
   master-1  Ready     master  73m  v1.20.0
   master-2  Ready     master  74m  v1.20.0
   worker-0  Ready     worker  11m  v1.20.0
   worker-1  Ready     worker  11m  v1.20.0
   ```



참고

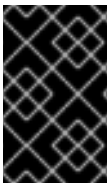
머신이 **Ready** 상태로 전환하는 데 서버 CSR의 승인 후 몇 분이 걸릴 수 있습니다.

추가 정보

- CSR에 대한 자세한 내용은 인증서 서명 요청을 참조하십시오.

11장. 머신 상태 점검 배포

머신 풀에서 손상된 머신을 자동으로 복구하도록 머신 상태 점검을 구성하고 배포할 수 있습니다.



중요

이 프로세스는 수동으로 프로비저닝된 시스템이 있는 클러스터에는 적용되지 않습니다. 머신 API가 작동하는 클러스터에서만 고급 머신 관리 및 스케일링 기능을 사용할 수 있습니다.

11.1. 머신 상태 점검 정보

MachineHealthCheck 리소스를 사용하여 클러스터의 시스템이 비정상적으로 간주되는 조건을 정의할 수 있습니다. 조건과 일치하는 머신이 자동으로 수정됩니다.

시스템 상태를 모니터링하려면 **NotReady** 상태를 15 분 동안 유지하거나 노드 문제 탐지기(node-problem-detector)에 영구적인 조건을 표시하는 등 검사할 조건과 모니터링할 머신 세트의 레이블을 설정이 포함된 **MachineHealthCheck** 사용자 지정 리소스 (CR)를 생성합니다.

MachineHealthCheck CR을 모니터링하는 컨트롤러에서 사용자가 정의한 상태를 점검합니다. 머신이 상태 확인에 실패하면 머신이 자동으로 삭제되고 대체할 새로운 머신이 만들어집니다. 머신이 삭제되면 **machine deleted** 이벤트가 표시됩니다.

참고

마스터 역할이 있는 머신의 경우 머신 상태 확인을 통해 상태가 좋지 않은 노드 수를 보고하지만 머신은 삭제되지 않습니다. 예를 들면 다음과 같습니다.

출력 예

```
$ oc get machinehealthcheck example -n openshift-machine-api
```

```
NAME      MAXUNHEALTHY   EXPECTEDMACHINES   CURRENTHEALTHY
example   40%            3                  1
```

머신 삭제로 인한 영향을 제한하기 위해 컨트롤러는 한 번에 하나의 노드만 드레인하고 삭제합니다. 대상 머신 풀에서 허용된 **maxUnhealthy** 임계값 보다 많은 비정상적인 머신이 있는 경우 컨트롤러가 머신 삭제를 중지하고 사용자가 수동으로 개입해야 합니다.

검사를 중지하려면 사용자 정의 리소스를 제거합니다.

11.1.1. 베어 메탈에서 MachineHealthCheck

베어 메탈 클러스터에서 머신 삭제를 사용하면 베어 메탈 호스트의 재프로비저닝이 트리거됩니다. 일반적으로 베어 메탈 재프로비저닝은 시간이 오래 걸리는 프로세스로, 이 과정에서 클러스터에 컴퓨팅 리소스가 누락되고 애플리케이션이 중단될 수 있습니다. 기본 수정 프로세스를 머신 삭제에서 호스트 전원 사이클로 변경하려면 **machine.openshift.io/remediation-strategy: external-baremetal** 주석을 MachineHealthCheck 리소스에 추가합니다.

주석을 설정하면 BMC 인증 정보를 사용하여 비정상 머신이 전원을 껐다가 켭니다.

11.1.2. 머신 상태 검사 배포 시 제한 사항

머신 상태 점검을 배포하기 전에 고려해야 할 제한 사항은 다음과 같습니다.

- 머신 세트가 소유한 머신만 머신 상태 검사를 통해 업데이트를 적용합니다.
- 컨트롤 플레인 시스템은 현재 지원되지 않으며 비정상적인 경우 업데이트 적용되지 않습니다.
- 머신의 노드가 클러스터에서 제거되면 머신 상태 점검에서 이 머신을 비정상적으로 간주하고 즉시 업데이트를 적용합니다.
- **nodeStartupTimeout** 후 시스템의 해당 노드가 클러스터에 참여하지 않으면 업데이트가 적용됩니다.
- **Machine** 리소스 단계가 **Failed**하면 즉시 머신에 업데이트를 적용합니다.

추가 리소스

- **MachineHealthCheck** CR에서 정의할 수 있는 노드 상태에 대한 자세한 내용은 클러스터의 모든 노드 나열을 참조하십시오.
- 쇼트 서킷에 대한 자세한 내용은 쇼트 서킷 (Short Circuit) 머신 상태 점검 및 수정 을 참조하십시오.

11.2. MACHINEHEALTHCHECK 리소스 샘플

MachineHealthCheck 리소스는 다음 YAML 파일 중 하나와 유사합니다.

베어 메탈 용 MachineHealthCheck

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineHealthCheck
metadata:
  name: example 1
  namespace: openshift-machine-api
  annotations:
    machine.openshift.io/remediation-strategy: external-baremetal 2
spec:
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-machine-role: <role> 3
      machine.openshift.io/cluster-api-machine-type: <role> 4
      machine.openshift.io/cluster-api-machineset: <cluster_name>-<label>-<zone> 5
  unhealthyConditions:
  - type:    "Ready"
    timeout: "300s" 6
    status: "False"
  - type:    "Ready"
    timeout: "300s" 7
    status: "Unknown"
  maxUnhealthy: "40%" 8
  nodeStartupTimeout: "10m" 9
```

1. 배포할 머신 상태 점검의 이름을 지정합니다.
2. 베어 메탈 클러스터의 경우 전원 사이클 수정을 활성화하려면 **annotations** 섹션에 **machine.openshift.io/remediation-strategy: external-baremetal** 주석을 포함해야 합니다. 이 주

machine.openshift.io/remediation-strategy: external-baremetal 주석을 포함해야 합니다. 이 업데이트 적용 전략으로 비정상 호스트가 클러스터에서 제거되지 않고 재부팅됩니다.

3 4 확인할 머신 풀의 레이블을 지정합니다.

5 추적할 머신 세트를 **<cluster_name>-<label>-<zone>** 형식으로 지정합니다. 예를 들어 **prod-node-us-east-1a**입니다.

6 7 노드 상태에 대한 시간 제한을 지정합니다. 시간 제한 기간 중 상태가 일치되면 머신이 수정됩니다. 시간 제한이 길어지면 비정상 머신의 워크로드에 대한 다운타임이 길어질 수 있습니다.

8 대상 풀에서 동시에 복구할 수 있는 시스템 수를 지정합니다. 이는 백분율 또는 정수로 설정할 수 있습니다. 비정상 머신의 수가 **maxUnhealthy**에서의 설정 제한을 초과하면 복구가 수행되지 않습니다.

9 머신 상태가 비정상으로 확인되기 전에 노드가 클러스터에 참여할 때까지 기다려야 하는 시간 초과 기간을 지정합니다.

참고

matchLabels는 예제일 뿐입니다. 특정 요구에 따라 머신 그룹을 매핑해야 합니다.

다른 모든 설치 유형에 대한 MachineHealthCheck

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineHealthCheck
metadata:
  name: example 1
  namespace: openshift-machine-api
spec:
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-machine-role: <role> 2
      machine.openshift.io/cluster-api-machine-type: <role> 3
      machine.openshift.io/cluster-api-machineset: <cluster_name>-<label>-<zone> 4
  unhealthyConditions:
  - type:    "Ready"
    timeout: "300s" 5
    status: "False"
  - type:    "Ready"
    timeout: "300s" 6
    status: "Unknown"
  maxUnhealthy: "40%" 7
  nodeStartupTimeout: "10m" 8
```

1 배포할 머신 상태 점검의 이름을 지정합니다.

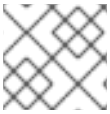
2 3 확인할 머신 풀의 레이블을 지정합니다.

4 추적할 머신 세트를 **<cluster_name>-<label>-<zone>** 형식으로 지정합니다. 예를 들어 **prod-node-us-east-1a**입니다.

5 6 노드 상태에 대한 시간 제한을 지정합니다. 시간 제한 기간 중 상태가 일치되면 머신이 수정됩니다. 시간 제한이 길어지면 비정상 머신의 워크로드에 대한 다운타임이 길어질 수 있습니다.

7 대상 풀에서 동시에 복구할 수 있는 시스템 수를 지정합니다. 이는 백분율 또는 정수로 설정할 수 있습니다. 비정상 머신의 수가 **maxUnhealthy**에서의 설정 제한을 초과하면 복구가 수행되지 않습니다.

8 머신 상태가 비정상으로 확인되기 전에 노드가 클러스터에 참여할 때까지 기다려야 하는 시간 초과 기간을 지정합니다.



참고

matchLabels는 예제일 뿐입니다. 특정 요구에 따라 머신 그룹을 매핑해야 합니다.

11.2.1. 쇼트 서킷 (Short Circuit) 머신 상태 점검 및 수정

쇼트 서킷 (Short Circuit)은 클러스터가 정상일 경우에만 머신 상태 점검을 통해 머신을 조정합니다. 쇼트 서킷은 **MachineHealthCheck** 리소스의 **maxUnhealthy** 필드를 통해 구성됩니다.

사용자가 시스템을 조정하기 전에 **maxUnhealthy** 필드 값을 정의하는 경우 **MachineHealthCheck**는 비정상적으로 결정된 대상 풀 내의 **maxUnhealthy** 값과 비교합니다. 비정상 머신의 수가 **maxUnhealthy** 제한을 초과하면 수정을 위한 업데이트가 수행되지 않습니다.

중요

maxUnhealthy가 설정되지 않은 경우 기본값은 **100%**로 설정되고 클러스터 상태와 관계없이 머신이 수정됩니다.

적절한 **maxUnhealthy** 값은 배포하는 클러스터의 규모와 **MachineHealthCheck에서** 다루는 시스템 수에 따라 달라집니다. 예를 들어 **maxUnhealthy** 값을 사용하여 여러 가용 영역에서 여러 머신 세트를 처리할 수 있으므로 전체 영역을 손실하면 **maxUnhealthy** 설정이 클러스터 내에서 추가 수정을 방지 할 수 있습니다.

maxUnhealthy 필드는 정수 또는 백분율로 설정할 수 있습니다. **maxUnhealthy** 값에 따라 다양한 수정을 적용할 수 있습니다.

11.2.1.1. 절대 값을 사용하여 maxUnhealthy 설정

maxUnhealthy가 **2**로 설정된 경우

- 2개 이상의 노드가 비정상인 경우 수정을 위한 업데이트가 수행됩니다.
- 3개 이상의 노드가 비정상이면 수정을 위한 업데이트가 수행되지 않습니다

이러한 값은 머신 상태 점검에서 확인할 수 있는 머신 수와 관련이 없습니다.

11.2.1.2. 백분율을 사용하여 maxUnhealthy 설정

maxUnhealthy가 **40%**로 설정되어 있고 25 대의 시스템이 확인되고 있는 경우 다음을 수행하십시오.

- 10개 이상의 노드가 비정상인 경우 수정을 위한 업데이트가 수행됩니다.
- 11개 이상의 노드가 비정상인 경우 수정을 위한 업데이트가 수행되지 않습니다.

maxUnhealthy가 **40%**로 설정되어 있고 6 대의 시스템이 확인되고 있는 경우 다음을 수행하십시오.

- 2개 이상의 노드가 비정상인 경우 수정을 위한 업데이트가 수행됩니다.

- 3개 이상의 노드가 비정상이면 수정을 위한 업데이트가 수행되지 않습니다

참고

maxUnhealthy 머신의 백분율이 정수가 아닌 경우 허용되는 머신 수가 반올림됩니다.

11.3. MACHINEHEALTHCHECK 리소스 만들기

클러스터의 모든 **MachineSets**에 대해 **MachineHealthCheck** 리소스를 생성할 수 있습니다. 컨트롤 플레인 시스템을 대상으로 하는 **MachineHealthCheck** 리소스를 생성해서는 안 됩니다.

사전 요구 사항

- **oc** 명령행 인터페이스를 설치합니다.

프로세스

1. 머신 상태 점검 정의가 포함된 **healthcheck.yml** 파일을 생성합니다.
2. **healthcheck.yml** 파일을 클러스터에 적용합니다.

```
$ oc apply -f healthcheck.yml
```