Red Hat Advanced Cluster Management for Kubernetes 2.5

멀티 클러스터 엔진

다중 클러스터 엔진 Operator 사용 방법을 알아보려면 자세히 알아보십시오.

Last Updated: 2023-07-14

Red Hat Advanced Cluster Management for Kubernetes 2.5 멀티 클러스터 엔진

다중 클러스터 엔진 Operator 사용 방법을 알아보려면 자세히 알아보십시오.

법적 공지

초록

멀티 클러스터 엔진 사용 방법을 알아보려면 자세히 알아보십시오.

차례

1장. 요구 사항 및 권장 사항

Kubernetes Operator용 다중 클러스터 엔진을 설치하기 전에 다음 시스템 구성 요구 사항 및 설정을 검토하십시오.

- 지원되는 운영 체제 및 플랫폼
- 네트워크 구성

중요: 2.5 이전의 Kubernetes용 Red Hat Advanced Cluster Management가 없는 클러스터에 Kubernetes용 멀티 클러스터 엔진을 설치해야 합니다. Kubernetes용 멀티 클러스터 엔진은 동일한 관리 구성 요소를 제공하는 2.5 이전 버전에서 Red Hat Advanced Cluster Management for Kubernetes와 공존할 수 없습니다. 이전에 Red Hat Advanced Cluster Management를 설치하지 않은 클러스터에 Kubernetes용 멀티 클러스터 엔진을 설치하는 것이 좋습니다. 버전 2.5.0 이상에서 Kubernetes용 Red Hat Advanced Cluster Management를 사용하는 경우 Kubernetes용 다중 클러스터 엔진이 클러스터에 이미 설치되어 있습니다.

1.1. KUBERNETES 클러스터 및 관리 클러스터에 대해 멀티 클러스터 엔진에서 지원되는 운영 체제 및 플랫폼

지원되는 운영 체제는 다음 표를 참조하십시오.

플랫폼	Kubernetes 클러스터의 멀티 클러스터 엔진에서 지원	관리형 클러스터에서 지원
Red Hat OpenShift Container Platform 3.11.200 이상 3.11.x 릴리스	없음	제공됨
Red Hat OpenShift Container Platform 4.8.2 이상	제공됨	제공됨
Amazon Web Services의 Red Hat OpenShift Container Platform	제공됨	제공됨
Microsoft Azure의 Red Hat OpenShift Container Platform	제공됨	제공됨
Google Cloud Platform의 Red Hat OpenShift Container Platform	제공됨	제공됨
Red Hat OpenShift Kubernetes Engine	없음	제공됨
Google Kubernetes Engine (Google GKE) (Kubernetes 1.17, 이상)	없음	제공됨
Amazon Elastic Kubernetes Service(Amazon EKS)(Kubernetes 1.17.6 이상)	없음	제공됨

Microsoft Azure Kubernetes Service (Microsoft AKS) (Kubernetes 1.19.6 이상)	없음	제공됨

1.2. 네트워크 구성

다음 섹션에서 연결을 허용하도록 네트워크 설정을 구성합니다.

1.2.1. Kubernetes Operator 클러스터 네트워킹 요구 사항용 멀티 클러스터 엔진

Kubernetes 클러스터 네트워킹 요구 사항의 다중 클러스터 엔진의 경우 다음 표를 참조하십시오.

direction	연결	포트(지정된 경우)
outbound	클라우드 공급자의 API	
outbound	(선택 사항) 프로비저닝된 관리 클러스터의 Kubernetes API 서버	6443
아웃바운드 및 인바운드	관리 클러스터의 **WorkManager** 서비스 경로	443
인바운드	관리 클러스터에서 Kubernetes 클러스터에 대한 멀티 클러스터 엔진의 Kubernetes API 서버	6443
인바운드	GitHub의 post-commit 후크를 Kubernetes 클러스터의 다중 클러스터 엔진에 연결합니다. 이 설정은 특정 애플리케이션 관리 기능을 사용하는 경우에만 필요합니다.	6443

1.2.2. 관리형 클러스터 네트워킹 요구 사항

관리형 클러스터 네트워킹 요구 사항은 다음 표를 참조하십시오.

direction	연결	포트(지정된 경우)
아웃바운드 및 인바운드	Kubernetes 클러스터용 다중 클러스터 엔진의 Kubernetes API 서버	6443

1.2.3. 호스트된 컨트롤 플레인 네트워킹 요구 사항 (기술 프리뷰)

호스팅 컨트롤 플레인을 사용하는 경우 **HypershiftDeployment** 리소스는 다음 표에 나열된 끝점에 연결되어 있어야 합니다.

direction	연결	포트(지정된 경우)
outbound	OpenShift Container Platform 컨트롤 플레인 및 작업자 노드	
outbound	Amazon Web Services의 호스트 클러스터만의 경우: AWS API 및 S3 API에 대한 아웃 바운드 연결	
outbound	Microsoft Azure 클라우드 서비스의 호스팅 클러스터의 경우: Azure API에 대한 아웃 바운드 연결	
outbound	coreOS의 ISO 이미지와 OpenShift Container Platform Pod용 이미지 레지스트리를 저장하는 OpenShift Container Platform 이미지 리포지토리	

2장. 시작하기

2.1. 소개

Kubernetes Operator용 다중 클러스터 엔진을 설치하기 전에 요구 사항 및 권장 사항에서 시스템 구성 요구 사항 및 설정을 검토하십시오. 클러스터에 지원되는 OpenShift Container Platform 버전이 설치되어 실행 중이면 온라인에 연결된 동안 설치를 진행할 수 있습니다.

2.2. 클러스터 생성 및 관리

설치한 후 클러스터를 생성, 가져오기 및 관리할 준비가 된 것입니다. Kubernetes 클러스터용 멀티 클러스터 엔진에서 관리할 다른 Openshift Container Platform 클러스터를 생성할 수 있습니다.

1. 생성할 수 있는 관리형 클러스터 유형에 대한 자세한 내용은 클러스터 생성을 참조하십시오.
2. 수동으로 가져올 클러스터가 있는 경우 관리형 클러스터를 가져오는 방법을 알아보려면 클러스터 가져오기를 참조하십시오.
3. 더 이상 클러스터를 관리할 필요가 없는 경우 관리형 클러스터 분리를 볼 수 있습니다.

2.3. MANIFESTWORK 예

ManifestWork를 사용하여 워크로드 배포 시 프로세스와 예를 볼 수 있습니다.

3장. 온라인 연결 중 설치

Kubernetes Operator의 다중 클러스터 엔진은 Kubernetes 엔진의 다중 클러스터 엔진을 포함하는 구성 요소의 설치, 업그레이드 및 제거를 관리하는 Operator Lifecycle Manager를 사용하여 설치됩니다.

필수 액세스 권한: 클러스터 관리자

중요:

- 2.5 설치 이전의 Kubernetes용 Red Hat Advanced Cluster Management가 없는 클러스터에 Kubernetes용 멀티 클러스터 엔진을 설치해야 합니다. Kubernetes용 멀티 클러스터 엔진은 동일한 관리 구성 요소를 제공하는 2.5 이전 버전에서 Red Hat Advanced Cluster Management for Kubernetes와 공존할 수 없습니다. 이전에 Red Hat Advanced Cluster Management를 설치하지 않은 클러스터에 Kubernetes용 멀티 클러스터 엔진을 설치하는 것이 좋습니다. 버전 2.5.0 이상에서 Kubernetes용 Red Hat Advanced Cluster Management를 사용하는 경우 Kubernetes용 다중 클러스터 엔진이 클러스터에 이미 설치되어 있습니다.

- OpenShift Container Platform Dedicated 환경의 경우 **cluster-admin** 권한이 있어야 합니다. 기본적으로 **dedicated-admin** 역할에는 OpenShift Container Platform Dedicated 환경에서 네임스페이스를 생성하는 데 필요한 권한이 없습니다.

- 기본적으로 엔진 구성 요소는 추가 구성없이 OpenShift Container Platform 클러스터의 작업자 노드에 설치됩니다. OpenShift Container Platform OperatorHub 웹 콘솔 인터페이스를 사용하거나 OpenShift Container Platform CLI를 사용하여 엔진을 작업자 노드에 설치할 수 있습니다.

- 인프라 노드를 사용하여 OpenShift Container Platform 클러스터를 구성한 경우 추가 리소스 매개변수가 포함된 OpenShift Container Platform CLI를 사용하여 해당 인프라 노드에 엔진을 설치할 수 있습니다. 모든 엔진 구성 요소가 인프라 노드를 지원하는 것은 아니므로 인프라 노드에 Kubernetes용 다중 클러스터 엔진을 설치할 때 일부 작업자 노드가 여전히 필요합니다. 자세한 내용은 *인프라 노드에 Kubernetes 엔진의 다중 클러스터 엔진 설치* 섹션을 참조하십시오.

- OpenShift Container Platform 또는 Kubernetes용 다중 클러스터 엔진에서 생성하지 않은 Kubernetes 클러스터를 가져오려면 이미지 가져오기 보안을 구성해야 합니다. 이미지 풀 시크릿 및 기타 고급 구성을 구성하는 방법에 대한 자세한 내용은 이 문서의 고급 구성 섹션의 옵션을 참조하십시오.
  - 사전 요구 사항
  - OpenShift Container Platform 설치 확인
  - OperatorHub 웹 콘솔 인터페이스에서 설치
  - OpenShift Container Platform CLI에서 설치
  - 인프라 노드에 설치

3.1. 사전 요구 사항

Kubernetes용 멀티 클러스터 엔진을 설치하기 전에 다음 요구사항을 참조하십시오.

- RedHat OpenShift Container Platform 클러스터는 OpenShift Container Platform 콘솔에서 OperatorHub 카탈로그의 Kubernetes Operator의 다중 클러스터 엔진에 액세스할 수 있어야 합니다.

- catalog.redhat.com 에 액세스해야 합니다.

- OpenShift Container Platform 버전 4.8 이상은 사용자 환경에 배포해야 하며 OpenShift Container Platform CLI로 로그인해야 합니다. OpenShift Container Platform에 대한 다음 설치 설명서를 참조하십시오.
  - OpenShift Container Platform 버전 4.10
  - OpenShift Container Platform 버전 4.9
  - OpenShift Container Platform 버전 4.8
- **oc** 명령을 실행하려면 OpenShift Container Platform CLI(명령줄 인터페이스)를 구성해야 합니다. OpenShift Container Platform CLI 설치 및 구성에 대한 정보는 CLI 시작하기 를 참조하십시오.
- OpenShift Container Platform 권한을 통해 네임스페이스를 생성할 수 있어야 합니다.
- Operator의 종속성에 액세스하려면 인터넷 연결이 있어야 합니다.
- OpenShift Container Platform Dedicated 환경에 설치하려면 다음을 참조하십시오.
  - OpenShift Container Platform Dedicated 환경이 구성 및 실행되고 있어야 합니다.
  - 엔진을 설치하는 OpenShift Container Platform Dedicated 환경에 대한 **cluster-admin** 권한이 있어야 합니다.

3.2. OPENSHIFT CONTAINER PLATFORM 설치 확인

레지스트리 및 스토리지 서비스를 포함하여 지원되는 OpenShift Container Platform 버전이 설치되어 작동해야 합니다. OpenShift Container Platform 설치에 대한 자세한 내용은 OpenShift Container Platform 설명서를 참조하십시오.

1. Kubernetes 엔진 Operator의 다중 클러스터 엔진이 OpenShift Container Platform 클러스터에 설치되어 있지 않은지 확인합니다. Kubernetes Operator용 다중 클러스터 엔진에서는 각 OpenShift Container Platform 클러스터에 하나의 단일 설치만 허용합니다. 설치가 없는 경우 다음 단계를 계속합니다.
2. OpenShift Container Platform 클러스터가 올바르게 설정되었는지 확인하려면 다음 명령을 사용하여 OpenShift Container Platform 웹 콘솔에 액세스합니다.

   ```
   kubectl -n openshift-console get route
   ```

   다음 예제 출력을 참조하십시오.

   ```
   openshift-console console console-openshift-console.apps.new-coral.purple-chesterfield.com
   console   https   reencrypt/Redirect     None
   ```

3. 브라우저에서 URL을 열고 결과를 확인합니다. 콘솔 URL에 **console-openshift-console.router.default.svc.cluster.local** 이 표시되면 OpenShift Container Platform을 설치할 때 **openshift_master_default_subdomain** 의 값을 설정합니다. URL의 다음 예제를 참조하십시오. **https://console-openshift-console.apps.new-coral.purple-chesterfield.com**.

Kubernetes용 다중 클러스터 엔진을 설치할 수 있습니다.

3.3. OPERATORHUB 웹 콘솔 인터페이스에서 설치

모범 사례: OpenShift Container Platform 탐색의 *관리자* 보기에서 OpenShift Container Platform과 함께 제공되는 OperatorHub 웹 콘솔 인터페이스를 설치합니다.

1. **Operator** > **OperatorHub** 를 선택하여 사용 가능한 Operator 목록에 액세스하고 *Kubernetes Operator용 다중 클러스터 엔진을* 선택합니다.
2. **설치를** 클릭합니다.
3. *Operator 설치 페이지에서 설치* 옵션을 선택합니다.
   - 네임스페이스:
     - Kubernetes 엔진의 다중 클러스터 엔진은 자체 네임스페이스 또는 프로젝트에 설치해야 합니다.
     - 기본적으로 OperatorHub 콘솔 설치 프로세스는 **다중 클러스터 엔진** 이라는 네임스페이스를 생성합니다. **모범 사례:** 사용 가능한 경우 **multicluster-engine** 네임스페이스를 계속 사용합니다.
     - 이미 **multicluster-engine** 이라는 네임스페이스가 있는 경우 다른 네임스페이스를 선택합니다.
   - Channel: 선택한 채널은 설치 중인 릴리스에 해당합니다. 채널을 선택하면 확인된 릴리스를 설치하고 해당 릴리스 내에서 향후 에라타 업데이트를 가져옵니다.
   - 승인 전략: 승인 전략은 구독한 채널에 업데이트를 적용하는 데 필요한 사람 상호 작용을 식별합니다.
     - 기본적으로 선택되어 있는 **자동** 을 선택하여 해당 릴리스 내의 모든 업데이트가 자동으로 적용되도록 합니다.
     - 업데이트를 사용할 수 있을 때 알림을 받으려면 **수동** 을 선택합니다. 언제 업데이트가 적용되었는지에 대한 우려가 있는 경우 이 방법이 모범 사례일 수 있습니다.

   참고: 다음 마이너 릴리스로 업그레이드하려면 *OperatorHub* 페이지로 돌아가 최신 릴리스의 새 채널을 선택해야 합니다.
4. **설치를** 선택하여 변경 사항을 적용하고 Operator를 생성합니다.
5. *MultiClusterEngine* 사용자 정의 리소스를 생성하려면 다음 프로세스를 참조하십시오.
   a. OpenShift Container Platform 콘솔 탐색에서 **설치된 Operator** > **Kubernetes용 다중 클러스터 엔진을** 선택합니다.
   b. **MultiCluster Engine** 탭을 선택합니다.
   c. **MultiClusterEngine 생성을** 선택합니다.
   d. YAML 파일에서 기본값을 업데이트합니다. 문서의 *MultiClusterEngine 고급 구성* 섹션의 옵션을 참조하십시오.
      - 다음 예제에서는 편집기에 복사할 수 있는 기본 템플릿을 보여줍니다.

```
apiVersion: multicluster.openshift.io/v1
kind: MultiClusterEngine
metadata:
  name: multiclusterengine
spec: {}
```

6. **생성을** 선택하여 사용자 정의 리소스를 초기화합니다. Kubernetes 엔진을 빌드하고 시작하는 데 다중 클러스터 엔진을 빌드하는 데 최대 10분이 걸릴 수 있습니다.
MultiClusterEngine 리소스가 생성되면 *MultiCluster Engine* 탭에서 리소스의 상태를 사용할 수 있습니다.

3.4. OPENSHIFT CONTAINER PLATFORM CLI에서 설치

1. **Operator 요구 사항이 포함된 Kubernetes 엔진 네임스페이스의 다중 클러스터 엔진을 생성합니다. 다음 명령을 실행합니다. 여기서 namespace 는 Kubernetes 엔진 네임스페이스의 다중 클러스터 엔진의 이름입니다. 네임스페이스 값은 OpenShift Container Platform 환경에서 *Project* 라고 할 수 있습니다.**

```
oc create namespace <namespace>
```

2. **프로젝트 네임스페이스를 생성한 네임스페이스로 전환합니다. namespace 를 1단계에서 생성한 Kubernetes 엔진의 다중 클러스터 엔진 이름으로 바꿉니다.**

```
oc project <namespace>
```

3. **YAML 파일을 생성하여 OperatorGroup 리소스를 구성합니다. 각 네임스페이스에는 Operator 그룹이 하나만 있을 수 있습니다. default 를 Operator 그룹의 이름으로 바꿉니다. namespace 를 프로젝트 네임스페이스의 이름으로 바꿉니다. 다음 예제를 참조하십시오.**

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: <default>
spec:
  targetNamespaces:
  - <namespace>
```

4. **다음 명령을 실행하여 OperatorGroup 리소스를 생성합니다. operator-group 을 생성한 Operator 그룹 YAML 파일의 이름으로 변경합니다.**

```
oc apply -f <path-to-file>/<operator-group>.yaml
```

5. **YAML 파일을 생성하여 OpenShift Container Platform 서브스크립션을 구성합니다. 파일은 다음 예와 유사해야 합니다.**

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
```

```
  name: multicluster-engine
spec:
  sourceNamespace: openshift-marketplace
  source: redhat-operators
  channel: stable-1.0
  installPlanApproval: Automatic
  name: multicluster-engine
```

참고: 인프라 노드에 **Kubernetes** 엔진의 다중 클러스터 엔진을 설치하려면 **Operator Lifecycle Manager** 서브스크립션 추가 구성 섹션을 참조하십시오.

6. 다음 명령을 실행하여 **OpenShift Container Platform** 서브스크립션을 생성합니다. 서브스크립션을 생성한 서브스크립션 파일의 이름으로 변경합니다.

```
oc apply -f <path-to-file>/<subscription>.yaml
```

7. **YAML** 파일을 생성하여 **MultiClusterEngine** 사용자 정의 리소스를 구성합니다. 기본 템플릿은 다음 예와 유사해야 합니다.

```
apiVersion: multicluster.openshift.io/v1
kind: MultiClusterEngine
metadata:
  name: multiclusterengine
spec: {}
```

참고: 인프라 노드에 **Kubernetes** 엔진에 대한 다중 클러스터 엔진을 설치하려면 **MultiClusterEngine** 사용자 정의 리소스 추가 구성 섹션을 참조하십시오.

8. 다음 명령을 실행하여 **MultiClusterEngine** 사용자 정의 리소스를 생성합니다. **custom-resource** 를 사용자 정의 리소스 파일의 이름으로 교체합니다.

```
oc apply -f <path-to-file>/<custom-resource>.yaml
```

이 단계가 다음 오류와 함께 실패하면 리소스가 계속 생성되고 적용되는 것입니다. 리소스가 생성되면 몇 분 후에 명령을 다시 실행합니다.

```
error: unable to recognize "./mce.yaml": no matches for kind "MultiClusterEngine" in version
"operator.multicluster-engine.io/v1"
```

9. 다음 명령을 실행하여 사용자 정의 리소스를 가져옵니다. 다음 명령을 실행한 후 **MultiClusterEngine** 사용자 정의 리소스 상태가 **status.phase** 필드에 **Available** 로 표시되는 데

최대 10분이 걸릴 수 있습니다.

```
oc get mce -o=jsonpath='{.items[0].status.phase}'
```

Kubernetes Operator용 다중 클러스터 엔진을 다시 설치하는 중이고 Pod가 시작되지 않는 경우 이 문제를 해결하기 위한 단계의 재설치 실패 문제 해결을 참조하십시오.

참고:

- ClusterRoleBinding 이 있는 ServiceAccount 는 Kubernetes의 다중 클러스터 엔진과 Kubernetes용 다중 클러스터 엔진을 설치하는 네임스페이스에 대한 액세스 권한이 있는 사용자 자격 증명에 대한 클러스터 관리자 권한을 자동으로 부여합니다.

3.5. 인프라 노드에 설치

OpenShift Container Platform 클러스터는 승인된 관리 구성 요소를 실행하기 위한 인프라 노드를 포함하도록 구성할 수 있습니다. 인프라 노드에서 구성 요소를 실행하면 해당 관리 구성 요소를 실행하는 노드에 OpenShift Container Platform 서브스크립션 할당량이 할당되지 않습니다.

OpenShift Container Platform 클러스터에 인프라 노드를 추가한 후 OpenShift Container Platform CLI 지침에서 설치를 수행하고 Operator Lifecycle Manager 서브스크립션 및 MultiClusterEngine 사용자 정의 리소스에 다음 구성을 추가합니다.

3.5.1. OpenShift Container Platform 클러스터에 인프라 노드 추가

OpenShift Container Platform 설명서의 인프라 머신 세트 생성에 설명된 절차를 따르십시오. 인프라 노드는 관리 이외의 워크로드가 실행되지 않도록 쿠버네티스 테인트 및 라벨 을 사용하여 구성됩니다.

Kubernetes용 다중 클러스터 엔진에서 제공하는 인프라 노드 활성화와 호환하려면 인프라 노드에 다음 테인트 및 라벨 이 적용되었는지 확인하십시오.

```
metadata:
  labels:
    node-role.kubernetes.io/infra: ""
spec:
  taints:
  - effect: NoSchedule
    key: node-role.kubernetes.io/infra
```

3.5.2. Operator Lifecycle Manager 서브스크립션 추가 구성

Operator Lifecycle Manager 서브스크립션을 적용하기 전에 다음 추가 구성을 추가합니다.

```
spec:
  config:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
    tolerations:
    - key: node-role.kubernetes.io/infra
      effect: NoSchedule
      operator: Exists
```

3.5.3. MultiClusterEngine 사용자 정의 리소스 추가 구성

MultiClusterEngine 사용자 정의 리소스를 적용하기 전에 다음 추가 구성을 추가합니다.

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

4장. 연결이 끊긴 네트워크에 설치

인터넷에 연결되지 않은 Red Hat OpenShift Container Platform 클러스터에 멀티 클러스터 엔진 Operator를 설치해야 할 수 있습니다. 연결이 끊긴 엔진에 설치하려면 연결된 설치와 동일한 몇 가지 단계가 필요합니다.

중요: 2.5 이전의 Kubernetes용 Red Hat Advanced Cluster Management가 없는 클러스터에 Kubernetes용 멀티 클러스터 엔진을 설치해야 합니다. Kubernetes용 멀티 클러스터 엔진은 동일한 관리 구성 요소를 제공하는 2.5 이전 버전에서 Red Hat Advanced Cluster Management for Kubernetes와 공존할 수 없습니다. 이전에 Red Hat Advanced Cluster Management를 설치하지 않은 클러스터에 Kubernetes용 멀티 클러스터 엔진을 설치하는 것이 좋습니다. 버전 2.5.0 이상에서 Kubernetes용 Red Hat Advanced Cluster Management를 사용하는 경우 Kubernetes용 다중 클러스터 엔진이 클러스터에 이미 설치되어 있습니다.

설치하는 동안 네트워크에서 직접 액세스하는 대신 설치 중에 액세스하려면 패키지 사본을 다운로드해야 합니다.

- 사전 요구 사항
- OpenShift Container Platform 설치 확인
- 인프라 노드에 엔진 설치 준비

4.1. 사전 요구 사항

The multicluster engine Operator를 설치하기 전에 다음 요구 사항을 충족해야 합니다.

- Red Hat OpenShift Container Platform 버전 4.8 이상은 사용자 환경에 배포해야 하며 CLI(명령줄 인터페이스)로 로그인해야 합니다.
- catalog.redhat.com 에 액세스해야 합니다.

  참고: 베어 메탈 클러스터를 관리하려면 OpenShift Container Platform 버전 4.8 이상이 있어야 합니다.

OpenShift Container Platform 버전 4.10,OpenShift Container Platform 버전 4.8 을 참조하십시오.

- Red Hat OpenShift Container Platform CLI는 버전 4.8 이상이어야 하며 oc 명령을 실행하도록 구성해야 합니다. Red Hat OpenShift CLI 설치 및 구성에 대한 정보는 CLI 시작하기 를 참조하십시오.

- Red Hat OpenShift Container Platform 권한을 통해 네임스페이스를 생성할 수 있어야 합니다.

- Operator의 종속성을 다운로드하려면 인터넷 연결이 있는 워크스테이션이 있어야 합니다.

4.2. OPENSHIFT CONTAINER PLATFORM 설치 확인

- 클러스터에서 설치 및 작동하는 레지스트리 및 스토리지 서비스를 포함하여 지원되는 OpenShift Container Platform 버전이 있어야 합니다. OpenShift Container Platform 버전 4.8 에 대한 자세한 내용은 OpenShift Container Platform 설명서를 참조하십시오.

- 연결 시 OpenShift Container Platform 클러스터가 올바르게 설정되었는지 확인할 수 있습니다. OpenShift Container Platform 웹 콘솔에 액세스합니다.

  kubectl -n openshift-console get route 명령을 실행하여 OpenShift Container Platform 웹 콘솔에 액세스합니다. 다음 예제 출력을 참조하십시오.

```
openshift-console  console  console-openshift-console.apps.new-coral.purple-chesterfield.com  console  https  reencrypt/Redirect  None
```

  이 예의 콘솔 URL은 https:// console-openshift-console.apps.new-coral.purple-chesterfield.com 입니다. 브라우저에서 URL을 열고 결과를 확인합니다.

  콘솔 URL에 console-openshift-console.router.default.svc.cluster.local 이 표시되면 OpenShift Container Platform을 설치할 때 openshift_master_default_subdomain 의 값을 설정합니다.

4.3. 연결이 끊긴 환경에 설치

중요: 연결이 끊긴 환경에서 운영자를 설치하려면 필요한 이미지를 미러링 레지스트리에 다운로드해야 합니다. 다운로드하지 않으면 배포 중에 ImagePullBackOff 오류가 발생할 수 있습니다.

다음 단계에 따라 연결이 끊긴 환경에 Kubernetes Operator의 다중 클러스터 엔진을 설치합니다.

1. 미러 레지스트리를 생성합니다. 미러 레지스트리가 아직 없는 경우 Red Hat OpenShift Container Platform 설명서의 연결이 끊긴 설치 주제의 이미지 미러링 에 있는 절차를 완료하여 생성합니다.

   미러 레지스트리가 이미 있는 경우 기존 레지스트리를 구성하고 사용할 수 있습니다.

2. 참고: 베어 메탈의 경우 install-config.yaml 파일에서 연결이 끊긴 레지스트리의 인증서 정보를 제공해야 합니다. 연결이 끊긴 레지스트리에서 이미지에 액세스하려면 Kubernetes Operator의 다중 클러스터 엔진에서 레지스트리에 액세스할 수 있도록 인증서 정보를 제공해야 합니다.

   a. 레지스트리에서 인증서 정보를 복사합니다.

   b. 편집기에서 install-config.yaml 파일을 엽니다.

   c. additionalTrustBundle: | 에 대한 항목을 찾습니다.

   d. additionalTrustBundle 행 뒤에 인증서 정보를 추가합니다. 결과 내용은 다음 예와 유사해야 합니다.

   ```
   additionalTrustBundle: |
     -----BEGIN CERTIFICATE-----
     certificate_content
     -----END CERTIFICATE-----
   sshKey: >-
   ```

3. 중요: 다음 관리 정책이 필요한 경우 연결 해제된 이미지 레지스트리의 미러가 필요합니다.

   - Container Security Operator 정책: 이미지는 소스 registry.redhat.io/quay 에 있습니다.
   - Compliance Operator 정책: 이미지는 소스 registry.redhat.io/compliance에 있습니다.

- 게이트키퍼 운영자 정책: 이미지는 소스 registry.redhat.io/rhacm2에 있습니다.

  세 가지 Operator 모두의 미러 목록의 다음 예제를 참조하십시오.

```
- mirrors:
  - <your_registry>/rhacm2
  source: registry.redhat.io/rhacm2
- mirrors:
  - <your_registry>/quay
  source: registry.redhat.io/quay
- mirrors:
  - <your_registry>/compliance
  source: registry.redhat.io/compliance
```

4. install-config.yaml 파일을 저장합니다.

5. rhacm-policy.yaml 이라는 ImageContentSourcePolicy 가 포함된 YAML 파일을 생성합니다. 참고: 실행 중인 클러스터에서 이 값을 수정하면 모든 노드가 롤링 재시작됩니다.

```
apiVersion: operator.openshift.io/v1alpha1
kind: ImageContentSourcePolicy
metadata:
  name: mce-repo
spec:
  repositoryDigestMirrors:
  - mirrors:
    - mirror.registry.com:5000/multicluster-engine
    source: registry.redhat.io/multicluster-engine
```

6. 다음 명령을 입력하여 ImageContentSourcePolicy 파일을 적용합니다.

```
oc apply -f mce-policy.yaml
```

7. 연결이 끊긴 Operator Lifecycle Manager Red Hat Operator 및 커뮤니티 Operator를 활성화합니다.

   Kubernetes Operator용 다중 클러스터 엔진은 Operator Lifecycle Manager Red Hat Operator 카탈로그에 포함되어 있습니다.

8. Red Hat Operator 카탈로그에 대해 연결이 끊긴 Operator Lifecycle Manager를 구성합니다. Red Hat OpenShift Container Platform 설명서의 제한된 네트워크에서 Operator Lifecycle

Manager 사용 주제의 단계를 따르십시오.

9. 연결이 끊긴 Operator Lifecycle Manager에 이미지가 있으므로 Operator Lifecycle Manager 카탈로그에서 Kubernetes Operator의 다중 클러스터 엔진을 계속 설치합니다.

필요한 단계는 온라인에 연결된 동안 설치를 참조하십시오.

5장. 콘솔 개요

OpenShift Container Platform 콘솔 플러그인은 **OpenShift Container Platform 4.10** 웹 콘솔에서 사용할 수 있으며 통합할 수 있습니다. 이 기능을 사용하려면 콘솔 플러그인이 활성화된 상태를 유지해야 합니다. 멀티 클러스터 엔진 **Operator**는 인프라 및 인증 정보 탐색 항목에서 특정 콘솔 기능을 표시합니다. **Red Hat Advanced Cluster Management**를 설치하는 경우 더 많은 콘솔 기능이 표시됩니다.

참고: 플러그인이 활성화된 **OpenShift Container Platform 4.10**의 경우 드롭다운 메뉴에서 모든 클러스터를 선택하여 **OpenShift Container Platform** 콘솔 내에서 **Red Hat Advanced Cluster Management**에 액세스할 수 있습니다.

1. 플러그인을 비활성화하려면 **OpenShift Container Platform** 콘솔의 *관리자* 화면에 있어야 합니다.

2. 탐색 메뉴에서 **Administration** 을 찾아 **Cluster Settings** 을 클릭한 다음 ***Configuration*** 탭을 클릭합니다.

3. *구성 리소스* 목록에서 웹 콘솔에 대한 클러스터 전체 구성이 포함된 **operator.openshift.io API** 그룹이 있는 **Console** 리소스를 클릭합니다.

4. *콘솔 플러그인* 탭을 클릭합니다. **mce** 플러그인이 나열됩니다. 참고: **Red Hat Advanced Cluster Management**가 설치되어 있으면 **acm** 로도 나열됩니다.

5. 표에서 플러그인 상태를 수정합니다. 잠시 후에 콘솔을 새로 고침하라는 메시지가 표시됩니다.

6장. 고급 구성

Kubernetes Operator의 다중 클러스터 엔진은 필요한 모든 구성 요소를 배포하는 Operator를 사용하여 설치됩니다. Kubernetes Operator의 다중 클러스터 엔진은 MultiClusterEngine 사용자 정의 리소스에 다음 속성 중 하나 이상을 추가하여 설치 중 또는 설치 후 추가로 구성할 수 있습니다.

6.1. 사용자 정의 이미지 가져오기 보안

OpenShift Container Platform 또는 Kubernetes Operator용 다중 클러스터 엔진에서 생성하지 않은 Kubernetes 클러스터를 가져오려면 OpenShift Container Platform 풀 시크릿 정보가 포함된 시크릿을 생성하여 배포 레지스트리에서 권한이 있는 콘텐츠에 액세스합니다.

OpenShift Container Platform 클러스터의 보안 요구 사항은 OpenShift Container Platform 및 Kubernetes용 다중 클러스터 엔진에 의해 자동으로 해결되므로 관리할 다른 유형의 Kubernetes 클러스터를 가져오지 않는 경우 시크릿을 생성할 필요가 없습니다.

중요: 이러한 보안은 네임스페이스별이므로 엔진에 사용하는 네임스페이스에 있는지 확인합니다.

1. 가져오기 시크릿 다운로드를 선택하여 cloud.redhat.com/openshift/install/pull-secret 에서 OpenShift Container Platform 풀 시크릿 파일을 다운로드합니다. OpenShift Container Platform 풀 시크릿은 Red Hat 고객 포털 ID와 연결되어 있으며 모든 Kubernetes 공급자에서 동일합니다.

2. 다음 명령을 실행하여 보안을 생성합니다.

```
oc create secret generic <secret> -n <namespace> --from-file=.dockerconfigjson=<path-to-pull-secret> --type=kubernetes.io/dockerconfigjson
```

- 보안을 생성하려는 보안의 이름으로 교체합니다.
- 보안이 네임스페이스 와 일치하므로 네임스페이스를 프로젝트 네임스페이스로 바꿉니다.
- 다운로드한 OpenShift Container Platform 풀 시크릿의 경로로 path-to-pull-secret 을 교체합니다.

다음 예제에서는 사용자 정의 풀 시크릿을 사용하려는 경우 사용할 **spec.imagePullSecret** 템플릿을 표시합니다. **secret** 을 풀 시크릿의 이름으로 교체합니다.

```
apiVersion: multicluster.openshift.io/v1
kind: MultiClusterEngine
metadata:
  name: multiclusterengine
spec:
  imagePullSecret: <secret>
```

6.2. 대상 네임스페이스

MultiClusterEngine 사용자 정의 리소스에서 위치를 지정하여 지정된 네임스페이스에 피연산자를 설치할 수 있습니다. 이 네임스페이스는 **MultiClusterEngine** 사용자 정의 리소스의 적용 시 생성됩니다.

중요: 대상 네임스페이스가 지정되지 않은 경우 **Operator**는 **multicluster-engine** 네임스페이스에 설치하고 **MultiClusterEngine** 사용자 정의 리소스 사양에 설정합니다.

다음 예제에서는 대상 네임스페이스를 지정하는 데 사용할 수 있는 **spec.targetNamespace** 템플릿을 표시합니다. **target** 을 대상 네임스페이스의 이름으로 바꿉니다. 참고: 대상 네임스페이스는 기본 네임스페이스일 수 없습니다.

```
apiVersion: multicluster.openshift.io/v1
kind: MultiClusterEngine
metadata:
  name: multiclusterengine
spec:
  targetNamespace: <target>
```

6.3. AVAILABILITYCONFIG

Red Hat Advanced Cluster Management hub 클러스터에는 **High** 및 **Basic** 이라는 두 가지 기능이 있습니다. 기본적으로 **hub** 클러스터는 고가용성으로, 허브 클러스터 구성 요소에 **2** 의 **replicaCount** 를 제공합니다. 이를 통해 페일오버의 경우 지원이 향상되지만 기본 가용성보다 많은 리소스를 소비하여 구성 요소에 **1** 의 **replicaCount** 를 제공합니다.

다음 예제에서는 기본 가용성이 있는 **spec.availabilityConfig** 템플릿을 보여줍니다.

```
apiVersion: multicluster.openshift.io/v1
kind: MultiClusterEngine
metadata:
```

```
  name: multiclusterengine
spec:
  availabilityConfig: "Basic"
```

6.4. NODESELECTOR

클러스터의 특정 노드에 설치할 **MultiClusterEngine** 에서 노드 선택기 세트를 정의할 수 있습니다. 다음 예제에서는 **node-role.kubernetes.io/infra** 레이블이 있는 노드에 **Red Hat Advanced Cluster Management Pod**를 할당하는 **spec.nodeSelector** 를 보여줍니다.

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

6.5. 허용 오차

MultiClusterEngine 이 클러스터에 정의된 특정 테인트를 허용할 수 있도록 허용 오차 목록을 정의할 수 있습니다. 다음 예제는 **node-role.kubernetes.io/infra** 테인트와 일치하는 **spec.tolerations** 를 보여줍니다.

```
spec:
  tolerations:
  - key: node-role.kubernetes.io/infra
    effect: NoSchedule
    operator: Exists
```

이전 **infra-node** 허용 오차는 구성에서 허용 오차를 지정하지 않고 기본적으로 **Pod**에 설정됩니다. 구성에서 허용 오차를 사용자 정의하면 이 기본 동작이 대체됩니다.

6.6. MANAGEDSERVICEACCOUNT 애드온 (기술 프리뷰)

기본적으로 **Managed-ServiceAccount** 애드온은 비활성화되어 있습니다. 이 구성 요소를 활성화하면 관리형 클러스터에서 서비스 계정을 생성하거나 삭제할 수 있습니다. 이 애드온을 사용하여 설치하려면 **spec.overrides** 의 **MultiClusterEngine** 사양에 다음을 포함합니다.

```
apiVersion: multicluster.openshift.io/v1
kind: MultiClusterEngine
metadata:
  name: multiclusterengine
spec:
  overrides:
    components:
    - name: managedserviceaccount-preview
      enabled: true
```

명령줄에서 리소스를 편집하고 **managedserviceaccount-preview** 구성 요소를 **enabled: true** 로 설정하여 **Managed-ServiceAccount** 애드온을 활성화할 수 있습니다. 또는 다음 명령을 실행하고 **<multiclusterengine-name>**을 **MultiClusterEngine** 리소스의 이름으로 교체할 수 있습니다.

```
oc patch MultiClusterEngine <multiclusterengine-name> --type=json -p='[{"op": "add", "path": "/spec/overrides/components/-","value":{"name":"managedserviceaccount-preview","enabled":true}}]'
```

활성화할 **ManagedServiceAccount** 애드온 활성화를 참조하십시오.

6.7. HYPERSHIFT 애드온 (기술 프리뷰)

기본적으로 **Hypershift** 애드온은 비활성화되어 있습니다. 이 애드온을 사용하여 설치하려면 **spec.overrides** 의 **MultiClusterEngine** 값에 다음을 포함합니다.

```
apiVersion: multicluster.openshift.io/v1
kind: MultiClusterEngine
metadata:
  name: multiclusterengine
spec:
  overrides:
    components:
    - name: hypershift-preview
      enabled: true
```

명령줄에서 리소스를 편집하여 **MultiClusterEngine** 을 생성한 후 **Hypershift -preview** 구성 요소를 **enabled: true** 로 설정하여 **Hypershift** 추가 기능을 활성화할 수 있습니다. 또는 다음 명령을 실행하고 **<multiclusterengine-name>**을 **MultiClusterEngine** 리소스의 이름으로 교체할 수 있습니다.

```
oc patch MultiClusterEngine <multiclusterengine-name> --type=json -p='[{"op": "add", "path": "/spec/overrides/components/-","value":{"name":"hypershift-preview","enabled":true}}]'
```

7장. MANAGEDSERVICEACCOUNT 애드온 활성화 (기술 프리뷰)

Kubernetes Operator용 다중 클러스터 엔진을 설치하면 ManagedServiceAccount 애드온이 기본적으로 비활성화되어 있습니다. 이 구성 요소를 활성화하면 관리형 클러스터에서 서비스 계정을 생성하거나 삭제할 수 있습니다.

필수 액세스: 편집기

hub 클러스터의 < managed_cluster > 네임스페이스에 ManagedServiceAccount 사용자 지정 리소스가 생성되면 관리 클러스터에서 ServiceAccount 가 생성됩니다.

TokenRequest 는 관리 클러스터의 ServiceAccount 를 사용하여 관리 클러스터의 Kubernetes API 서버에 대해 수행됩니다. 그런 다음 토큰은 hub 클러스터의 < target_managed_cluster> 네임스페이스의 Secret 에 저장됩니다.

참고: 토큰이 만료되고 순환될 수 있습니다. 토큰 요청에 대한 자세한 내용은 TokenRequest 를 참조하십시오.

7.1. 사전 요구 사항

- Red Hat OpenShift Container Platform 버전 4.9 이상은 사용자 환경에 배포해야 하며 CLI(명령줄 인터페이스)로 로그인해야 합니다.
- Kubernetes Operator용 다중 클러스터 엔진이 설치되어 있어야 합니다.

7.2. ENABLING MANAGEDSERVICEACCOUNT

hub 클러스터 및 관리 클러스터에 Managed-ServiceAccount 애드온을 활성화하려면 다음 단계를 완료하십시오.

1. hub 클러스터에서 ManagedServiceAccount 애드온을 활성화합니다. 자세한 내용은 고급 구성을 참조하십시오.
2. ManagedServiceAccount 애드온을 배포하고 대상 관리 클러스터에 적용합니다. 다음 YAML 파일을 생성하고 target_managed_cluster 를 Managed-ServiceAccount 애드온을 적용

하는 관리 클러스터의 이름으로 교체합니다.

```
apiVersion: addon.open-cluster-management.io/v1alpha1
kind: ManagedClusterAddOn
metadata:
  name: managed-serviceaccount
  namespace: <target_managed_cluster>
spec:
  installNamespace: open-cluster-management-agent-addon
```

3. 다음 명령을 실행하여 파일을 적용합니다.

```
oc apply -f -
```

이제 관리 클러스터에 대해 **Managed-ServiceAccount** 플러그인을 활성화했습니다. **ManagedServiceAccount** 를 구성하려면 다음 단계를 참조하십시오.

4. 다음 **YAML** 소스를 사용하여 **ManagedServiceAccount** 사용자 정의 리소스를 생성합니다.

```
apiVersion: authentication.open-cluster-management.io/v1alpha1
kind: ManagedServiceAccount
metadata:
  name: <managed_serviceaccount_name>
  namespace: <target_managed_cluster>
spec:
  rotation: {}
```

- **managed_serviceaccount_name** 을 **ManagedServiceAccount** 의 이름으로 교체합니다.
- **target_managed_cluster** 를 **ManagedServiceAccount** 를 적용하는 관리 클러스터의 이름으로 교체합니다.

5. 확인하려면 **ManagedServiceAccount** 오브젝트 상태에서 **tokenSecretRef** 특성을 보고 시크릿 이름과 네임스페이스를 찾습니다. 계정 및 클러스터 이름을 사용하여 다음 명령을 실행합니다.

```
oc get managedserviceaccount <managed_serviceaccount_name> -n
<target_managed_cluster> -o yaml
```

6. 관리 클러스터에서 생성된 **ServiceAccount** 에 연결된 검색된 토큰이 포함된 보안을 확인합

니다. 다음 명령을 실행합니다.

```
oc get secret <managed_serviceaccount_name> -n <target_managed_cluster> -o yaml
```

8장. 클러스터 생성

Kubernetes용 멀티 클러스터 엔진은 내부 Hive 구성 요소를 사용하여 Red Hat OpenShift Container Platform 클러스터를 생성합니다. 클러스터 생성 방법을 알아보려면 다음 정보를 참조하십시오.

- 사전 요구 사항
- ClusterDeployment를 사용하여 클러스터 생성
- 클러스터 풀을 사용하여 클러스터 생성

8.1. 사전 요구 사항

클러스터를 생성하기 전에 clusterImageSets 리포지토리를 복제하고 hub 클러스터에 적용해야 합니다. 다음 단계를 참조하십시오.

1. 다음 명령을 실행하여 복제합니다.

```
git clone https://github.com/stolostron/acm-hive-openshift-releases.git
cd acm-hive-openshift-releases
git checkout origin/release-2.5
```

2. 다음 명령을 실행하여 hub 클러스터에 적용합니다.

```
find clusterImageSets/fast -type d -exec oc apply -f {} \; 2> /dev/null
```

클러스터를 생성할 때 Red Hat OpenShift Container Platform 릴리스 이미지를 선택합니다.

8.2. CLUSTERDEPLOYMENT를 사용하여 클러스터 생성

ClusterDeployment 는 Hive 사용자 지정 리소스입니다. 개별 클러스터를 생성하는 방법을 알아보려면 다음 설명서를 참조하십시오.

Hive 사용 설명서를 따라 ClusterDeployment 사용자 정의 리소스를 생성합니다.

8.3. CLUSTERPOOL을 사용하여 클러스터 생성

ClusterPool 은 여러 클러스터를 생성하는 데 사용되는 Hive 사용자 정의 리소스이기도 합니다. Hive ClusterPool API로 클러스터를 생성합니다.

클러스터 풀 문서에 따라 클러스터를 프로비저닝합니다.

9장. 클러스터 가져오기

Kubernetes Operator용 다중 클러스터 엔진을 설치한 후 관리할 클러스터를 가져올 준비가 되었습니다. Red Hat OpenShift Container Platform CLI를 사용하면 가져올 클러스터의 kubeconfig 파일을 사용하여 클러스터를 가져올 수 있습니다. 또는 가져올 클러스터에서 가져오기 명령을 수동으로 실행할 수도 있습니다. 두 절차를 모두 문서화합니다.

CLI에서 가져오려면 다음 절차를 참조하십시오.

- 사전 요구 사항
- 가져오기 준비
- 자동 가져오기 보안을 사용하여 가져오기
- 수동 명령으로 가져오기
- 관리형 클러스터 분리

9.1. 사전 요구 사항

- Linux (x86_64, s390x, ppc64le) 또는 macOS를 사용할 수 있습니다.
- Kubernetes Operator용 다중 클러스터 엔진을 설치하고 Kubernetes 클러스터에 MultiClusterEngine 사용자 정의 리소스를 설치했는지 확인합니다.
- 관리하고 인터넷 연결을 관리하려는 다른 별도의 클러스터가 필요합니다.
- oc 명령을 실행하려면 OpenShift Container Platform CLI 버전 4.8 이상이 필요합니다. Red Hat OpenShift CLI 설치 및 구성에 대한 정보는 OpenShift CLI 시작하기 를 참조하십시오.

  참고: OpenShift Container Platform 콘솔에서 CLI 툴용 설치 파일을 다운로드합니다.

- OpenShift Container Platform에서 생성되지 않은 클러스터를 가져오는 경우 multiclusterengine.spec.imagePullSecret 을 정의해야 합니다. 이 보안은 Kubernetes Operator의 다중 클러스터 엔진을 설치할 때 생성될 수 있습니다. 이 보안을 정의하는 방법에 대한 자세한 내용은 사용자 정의 이미지 가져오기 보안을 참조하십시오.

9.2. 가져오기 준비

1. *엔진 클러스터에* 로그인합니다. *엔진 클러스터는* Kubernetes Operator용 다중 클러스터 엔진과 사용자 정의 리소스가 포함된 클러스터입니다. 다음 명령을 실행합니다.

```
oc login
```

2. engine 클러스터에서 다음 명령을 실행하여 프로젝트를 생성합니다.

참고: CLUSTER_NAME 에 정의되어 있고 .yaml 파일의 클러스터 네임스페이스와 명령으로도 사용되는 클러스터 이름입니다.

```
oc new-project ${CLUSTER_NAME}
```

3. 다음 명령을 실행하여 네임스페이스를 생성합니다.

```
oc label namespace ${CLUSTER_NAME} cluster.open-cluster-
management.io/managedCluster=${CLUSTER_NAME}
```

4. 다음 YAML 샘플을 사용하여 ManagedCluster 예제를 편집합니다.

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  name: ${CLUSTER_NAME}
spec:
  hubAcceptsClient: true
```

5. 선택 사항: 이번 릴리스에서는 hub 클러스터를 로컬 클러스터라고 하는 관리 클러스터가 되도록 자동으로 가져올 *수 없습니다*. 관리 클러스터가 로컬 클러스터가 되도록 수동으로 활성화하려면 metadata.labels. local-cluster: "true" 를 추가합니다. 다음 예제 YAML을 참조하여 이름이 local-cluster 인지 확인합니다. local-cluster 가 이름이 아닌 경우 가져오기가 실패하거나 예기치 않은 결과를 생성합니다.

```
apiVersion: cluster.open-cluster-management.io/v1
```

```
kind: ManagedCluster
metadata:
  labels:
    local-cluster: "true"
    cloud: auto-detect
    vendor: auto-detect
  name: local-cluster
spec:
  hubAcceptsClient: true
```

6. 파일을 **managed-cluster.yaml** 로 저장합니다.

7. 다음 명령을 사용하여 **YAML** 파일을 적용합니다.

```
oc apply -f managed-cluster.yaml
```

9.3. 자동 가져오기 보안을 사용하여 가져오기

engine 클러스터에 계속 로그인하는 동안 다음 단계를 진행합니다.

1. 가져올 클러스터의 **kubeconfig** 파일 또는 가져오는 클러스터의 **kube API** 서버 및 토큰을 검색합니다. **kubeconfig** 파일 또는 **kube api** 서버 및 토큰을 찾을 위치를 알아보려면 **Kubernetes** 클러스터 설명서를 참조하십시오.

2. **kubeconfig** 또는 **server/token** 쌍을 사용하여 다음 템플릿과 유사한 콘텐츠가 포함된 **YAML** 파일을 생성합니다.

```
apiVersion: v1
kind: Secret
metadata:
  name: auto-import-secret
stringData:
  # the following value to specify the retry times when your cluster failed to import
  autoImportRetry: "5"
  # If you are using the kubeconfig file, add the following value for the kubeconfig file
  # that has the current context set to the cluster to import:
  kubeconfig: |- <kubeconfig_file>
  # If you are using the server/token pair, add the following two values:
  server: <cluster_api_url>
  token: <Token to access the cluster>
type: Opaque
```

3. 파일을 **auto-import-secret.yaml**로 저장

4. 가져올 클러스터의 **kubeconfig** 파일을 사용하여 **${CLUSTER_NAME}** 네임스페이스에 가져오기 보안을 생성합니다. **kubeconfig** 및 **CLUSTER_NAME** 경로와 함께 다음 명령을 실행합니다.

```
oc apply -f auto-import-secret.yaml
```

참고: 자동 가져오기 보안은 한 번 사용되며 가져오기 프로세스가 완료되면 삭제됩니다.

5. 가져온 클러스터에 대한 **status ED** 및 **AVAILABLE** 상태를 검증합니다. **Kubernetes** 클러스터의 다중 클러스터 엔진에서 다음 명령을 실행합니다.

```
oc get managedcluster ${CLUSTER_NAME}
```

가져올 별도의 클러스터에서 다음 단계를 진행합니다.

6. 가져올 클러스터에 로그인합니다. 다음 명령을 실행합니다.

```
oc login
```

7. 가져올 클러스터에서 **Pod** 상태를 확인합니다. 다음 명령을 실행합니다.

```
oc get pod -n open-cluster-management-agent
```

8. 가져올 클러스터가 **AVAILABLE** 인 후 애드온이 설치됩니다. 클러스터에서 애드온의 포드 상태를 확인합니다. 다음 명령을 실행합니다.

```
oc get pod -n open-cluster-management-agent-addon
```

이제 클러스터를 가져옵니다.

9.4. 수동 명령을 사용하여 가져오기

중요: 가져오기 명령에는 가져온 각 클러스터에 복사되는 풀 시크릿 정보가 포함되어 있습니다. 가져온

클러스터에 액세스할 수 있는 모든 사용자는 풀 시크릿 정보도 볼 수 있습니다.

1. 엔진 클러스터의 가져오기 컨트롤러에서 생성한 **klusterlet-crd.yaml** 을 가져옵니다. 다음 명령을 실행합니다.

```
oc get secret ${CLUSTER_NAME}-import -n ${CLUSTER_NAME} -o jsonpath=
{.data.crds\\.yaml} | base64 --decode > klusterlet-crd.yaml
```

2. 엔진 클러스터의 가져오기 컨트롤러에서 생성한 **import.yaml** 을 가져옵니다. 다음 명령을 실행합니다.

```
oc get secret ${CLUSTER_NAME}-import -n ${CLUSTER_NAME} -o jsonpath=
{.data.import\\.yaml} | base64 --decode > import.yaml
```

가져올 별도의 클러스터에서 다음 단계를 진행합니다.

3. 가져올 클러스터에 로그인합니다.

```
oc login
```

4. 이전 단계에서 생성한 **klusterlet-crd.yaml** 을 적용합니다. 다음 명령을 실행합니다.

```
oc apply -f klusterlet-crd.yaml
```

5. 이전에 생성한 **import.yaml** 파일을 적용합니다. 다음 명령을 실행합니다.

```
oc apply -f import.yaml
```

6. 가져올 클러스터에서 **Pod** 상태를 확인합니다. 다음 명령을 실행합니다.

```
oc get pod -n open-cluster-management-agent
```

7. 가져올 클러스터에 대한 **status** 및 **AVAILABLE** 상태를 확인합니다. **engine** 클러스터에서 다음 명령을 실행합니다.

```
oc get managedcluster ${CLUSTER_NAME}
```

애드온은 가져오는 클러스터 후 설치됩니다. **AVAILABLE** 입니다.

8. 가져오고 있는 클러스터에서 애드온의 포드 상태를 확인합니다. 다음 명령을 실행합니다.

```
oc get pod -n open-cluster-management-agent-addon
```

이제 클러스터를 가져와서 엔진 클러스터에서 해당 클러스터를 관리할 수 있습니다.

9.5. 관리형 클러스터 분리

관리형 클러스터는 성공적으로 가져온 클러스터입니다. **engine** 클러스터에서 관리형 클러스터를 분리하려면 다음 명령을 실행합니다.

```
oc delete managedcluster ${CLUSTER_NAME}
```

이제 클러스터가 분리되었습니다.

10장. MANIFESTWORK를 사용하여 워크로드 배포

Kubernetes 클러스터의 다중 클러스터 엔진에서 관리형 클러스터에 워크로드를 배포할 수 있습니다. 예를 들면 다음과 같습니다. Kubernetes 클러스터의 다중 클러스터 엔진에서 관리 클러스터에 기본 배포를 생성하려면 ManifestWork 가 있는 다음 샘플을 참조하십시오.

1. Kubernetes 클러스터의 다중 클러스터 엔진에 로그인합니다.

```
oc login
```

2. YAML 파일을 생성하여 다음 예와 같이 ManifestWork 리소스를 구성합니다. CLUSTER_NAME 을 클러스터 가져오기 문서에서 가져온 관리형 클러스터 의 이름으로 교체합니다. 예제 YAML은 파일을 적용할 때 관리 클러스터 기본 네임스페이스에 배포합니다.

```
apiVersion: work.open-cluster-management.io/v1
kind: ManifestWork
metadata:
  name: hello-work
  namespace: ${CLUSTER_NAME}
  labels:
    app: hello
spec:
  workload:
    manifests:
    - apiVersion: apps/v1
      kind: Deployment
      metadata:
        name: hello
        namespace: default
      spec:
        selector:
          matchLabels:
            app: hello
        template:
          metadata:
            labels:
              app: hello
          spec:
            containers:
              - name: hello
                image: quay.io/asmacdo/busybox
                command: ['/bin/sh', '-c', 'echo "Hello, Kubernetes!" && sleep 300']
    - apiVersion: v1
      kind: Service
      metadata:
        labels:
          app: hello
        name: hello
        namespace: default
```

```
  spec:
    ports:
    - port: 8000
      protocol: TCP
      targetPort: 8000
    selector:
      app: hello
```

3. **YAML** 파일을 적용합니다. 다음 명령을 실행합니다.

```
oc apply -f manifestwork.yaml
```

4. 다음 명령을 실행하여 **Kubernetes** 클러스터의 다중 클러스터 엔진에서 **ManifestWork** 의 상태를 확인합니다.

```
oc get manifestwork -n ${CLUSTER_NAME} hello-work -o yaml
```

5. 관리 클러스터에 로그인하여 결과를 확인합니다. 다음 명령을 참조하십시오.

```
oc login
```

6. **Kubernetes** 클러스터용 다중 클러스터 엔진으로 생성한 배포를 확인합니다.

```
$ oc get deploy -n default
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
hello   1/1     1            1           37s
```

다음 명령을 사용하여 생성된 **Pod**를 볼 수도 있습니다.

```
$ oc get pod
NAME                     READY   STATUS    RESTARTS   AGE
hello-65f58985ff-4rm57   1/1     Running   0          42s
```

생성된 **Pod**의 로그를 보면 다음과 유사한 메시지가 표시됩니다.

```
$ oc logs hello-65f58985ff-4rm57
Hello, Kubernetes!
```

11장. API

클러스터 라이프사이클 관리를 위해 Kubernetes Operator용 멀티 클러스터 엔진의 API에 액세스할 수 있습니다. 사용자 필수 액세스: 역할이 할당된 작업만 수행할 수 있습니다. 자세한 내용은 다음 리소스 각각에 대한 API 설명서를 참조하십시오.

- 클러스터 API
- ClusterSets API (v1beta1)
- Clusterview API
- ClusterSetBindings API (v1beta1)
- MultiClusterEngine API
- placements API(v1alpha1)
- PlacementDecisions API (v1alpha1)
- 관리형 서비스 계정(기술 프리뷰)

11.1. 클러스터 API

11.1.1. 개요

이 문서는 Kubernetes용 다중 클러스터 엔진의 클러스터 리소스에 대한 것입니다. 클러스터 리소스에는 생성, 쿼리, 삭제 및 업데이트 등 네 가지 요청이 있습니다.

11.1.1.1. URI 스키마

BasePath : /kubernetes/apis
Schemes : HTTPS

11.1.1.2. 태그

- **cluster.open-cluster-management.io : 클러스터 생성 및 관리**

11.1.2. 경로

11.1.2.1. 모든 클러스터 쿼리

GET /cluster.open-cluster-management.io/v1/managedclusters

11.1.2.1.1. 설명

자세한 내용은 클러스터를 쿼리합니다.

11.1.2.1.2. 매개 변수

유형	이름	설명	스키마
header	**COOKIE** *필요*	권한 부여: 베어러 {ACCESS_TOKEN} ; ACCESS_TOKEN은 사용자 액세스 토큰입니다.	string

11.1.2.1.3. 응답

HTTP 코드	설명	스키마
200	성공	콘텐츠 없음
403	액세스 금지	콘텐츠 없음
404	리소스를 찾을 수 없음	콘텐츠 없음
500	내부 서비스 오류	콘텐츠 없음
503	서비스를 사용할 수 없음	콘텐츠 없음

11.1.2.1.4. Use

- **cluster/yaml**

11.1.2.1.5. 태그

- **cluster.open-cluster-management.io**

11.1.2.2. 클러스터 생성

POST /cluster.open-cluster-management.io/v1/managedclusters

11.1.2.2.1. 설명

클러스터 생성

11.1.2.2.2. 매개 변수

유형	이름	설명	스키마
header	**COOKIE** *필요*	권한 부여: 베어러 {ACCESS_TOKEN} ; ACCESS_TOKEN은 사용자 액세스 토큰입니다.	string
body	**본문** *필요*	생성할 클러스터를 설명하는 매개변수입니다.	Cluster

11.1.2.2.3. 응답

HTTP 코드	설명	스키마
200	성공	콘텐츠 없음
403	액세스 금지	콘텐츠 없음
404	리소스를 찾을 수 없음	콘텐츠 없음
500	내부 서비스 오류	콘텐츠 없음
503	서비스를 사용할 수 없음	콘텐츠 없음

11.1.2.2.4. Use

- **cluster/yaml**

11.1.2.2.5. 태그

- cluster.open-cluster-management.io

11.1.2.2.6. HTTP 요청의 예

11.1.2.2.6.1. 요청 본문

```
{
  "apiVersion" : "cluster.open-cluster-management.io/v1",
  "kind" : "ManagedCluster",
  "metadata" : {
    "labels" : {
      "vendor" : "OpenShift"
    },
    "name" : "cluster1"
  },
  "spec": {
    "hubAcceptsClient": true,
    "managedClusterClientConfigs": [
      {
        "caBundle": "test",
        "url": "https://test.com"
      }
    ]
  },
  "status" : { }
}
```

11.1.2.3. 단일 클러스터 쿼리

```
GET /cluster.open-cluster-management.io/v1/managedclusters/{cluster_name}
```

11.1.2.3.1. 설명

자세한 내용은 단일 클러스터를 쿼리합니다.

11.1.2.3.2. 매개 변수

유형	이름	설명	스키마
header	COOKIE *필요*	권한 부여: 베어러 {ACCESS_TOKEN} ; ACCESS_TOKEN은 사용자 액세스 토큰입니다.	string
경로	cluster_name *required*	쿼리할 클러스터의 이름입니다.	string

11.1.2.3.3. 응답

HTTP 코드	설명	스키마
200	성공	콘텐츠 없음
403	액세스 금지	콘텐츠 없음
404	리소스를 찾을 수 없음	콘텐츠 없음
500	내부 서비스 오류	콘텐츠 없음
503	서비스를 사용할 수 없음	콘텐츠 없음

11.1.2.3.4. 태그

- **cluster.open-cluster-management.io**

11.1.2.4. 클러스터 삭제

DELETE /cluster.open-cluster-management.io/v1/managedclusters/{cluster_name}

11.1.2.4.1. 설명

단일 클러스터 삭제

11.1.2.4.2. 매개 변수

유형	이름	설명	스키마
header	**COOKIE** *필요*	권한 부여: 베어러 {ACCESS_TOKEN} ; ACCESS_TOKEN은 사용자 액세스 토큰입니다.	string
경로	**cluster_name** *required*	삭제할 클러스터의 이름입니다.	string

11.1.2.4.3. 응답

HTTP 코드	설명	스키마
200	성공	콘텐츠 없음
403	액세스 금지	콘텐츠 없음

HTTP 코드	설명	스키마
404	리소스를 찾을 수 없음	콘텐츠 없음
500	내부 서비스 오류	콘텐츠 없음
503	서비스를 사용할 수 없음	콘텐츠 없음

11.1.2.4.4. 태그

- **cluster.open-cluster-management.io**

11.1.3. 정의

11.1.3.1. Cluster

이름	스키마
apiVersion *필요*	string
kind *필수*	string
메타데이터 *필요*	object
spec *필수*	spec

spec

이름	스키마
hubAcceptsClient *required*	bool
managedClusterClientConfigs *optional*	< managedClusterClientConfigs > array
leaseDurationSeconds *optional*	정수(int32)

managedClusterClientConfigs

이름	설명	스키마
URL *필요*		string
cabundle *선택 사항*	Pattern : **"^(?:[A-Za-z0-9+/]{4})*(?:[A-Za-z0-9+/]{2}==|[A-Za-z0-9+/]{3}=)?$"**	문자열(바이트)

11.2. CLUSTERSETS API (V1ALPHA1)

11.2.1. 개요

이 문서는 **Kubernetes**용 다중 클러스터 엔진의 **Clusterset** 리소스에 대한 것입니다. **Clusterset** 리소스에는 생성, 쿼리, 삭제 및 업데이트 등 네 가지 요청이 있습니다.

11.2.1.1. URI 스키마

BasePath **: /kubernetes/apis**
Schemes **: HTTPS**

11.2.1.2. 태그

- **cluster.open-cluster-management.io : Clustersets 생성 및 관리**

11.2.2. 경로

11.2.2.1. 모든 클러스터 세트 쿼리

GET /cluster.open-cluster-management.io/v1beta1/managedclustersets

11.2.2.1.1. 설명

자세한 내용은 클러스터 세트를 쿼리합니다.

11.2.2.1.2. 매개 변수

유형	이름	설명	스키마
header	**COOKIE** *필요*	권한 부여: 베어러 {ACCESS_TOKEN}; ACCESS_TOKEN은 사용자 액세스 토큰입니다.	string

11.2.2.1.3. 응답

HTTP 코드	설명	스키마
200	성공	콘텐츠 없음
403	액세스 금지	콘텐츠 없음
404	리소스를 찾을 수 없음	콘텐츠 없음
500	내부 서비스 오류	콘텐츠 없음
503	서비스를 사용할 수 없음	콘텐츠 없음

11.2.2.1.4. Use

- **clusterset/yaml**

11.2.2.1.5. 태그

- **cluster.open-cluster-management.io**

11.2.2.2. 클러스터 세트 생성

POST /cluster.open-cluster-management.io/v1beta1/managedclustersets

11.2.2.2.1. 설명

클러스터 세트를 생성합니다.

11.2.2.2.2. 매개 변수

유형	이름	설명	스키마

유형	이름	설명	스키마
header	**COOKIE** *필요*	권한 부여: 베어러 {ACCESS_TOKEN} ; ACCESS_TOKEN은 사용자 액세스 토큰입니다.	string
body	**본문** *필요*	생성할 clusterset을 설명하는 매개변수입니다.	Clusterset

11.2.2.2.3. 응답

HTTP 코드	설명	스키마
200	성공	콘텐츠 없음
403	액세스 금지	콘텐츠 없음
404	리소스를 찾을 수 없음	콘텐츠 없음
500	내부 서비스 오류	콘텐츠 없음
503	서비스를 사용할 수 없음	콘텐츠 없음

11.2.2.2.4. Use

- **clusterset/yaml**

11.2.2.2.5. 태그

- **cluster.open-cluster-management.io**

11.2.2.2.6. HTTP 요청의 예

11.2.2.2.6.1. 요청 본문

```
{
  "apiVersion" : "cluster.open-cluster-management.io/v1beta1",
  "kind" : "ManagedClusterSet",
  "metadata" : {
    "name" : "clusterset1"
  },
  "spec": { },
  "status" : { }
}
```

11.2.2.3. 단일 클러스터 세트 쿼리

GET /cluster.open-cluster-management.io/v1beta1/managedclustersets/{clusterset_name}

11.2.2.3.1. 설명

자세한 내용은 단일 클러스터 세트를 쿼리합니다.

11.2.2.3.2. 매개 변수

유형	이름	설명	스키마
header	**COOKIE** *필요*	권한 부여: 베어러 {ACCESS_TOKEN} ; ACCESS_TOKEN은 사용자 액세스 토큰입니다.	string
경로	**clusterset_name** *required*	쿼리할 클러스터 세트의 이름입니다.	string

11.2.2.3.3. 응답

HTTP 코드	설명	스키마
200	성공	콘텐츠 없음
403	액세스 금지	콘텐츠 없음
404	리소스를 찾을 수 없음	콘텐츠 없음
500	내부 서비스 오류	콘텐츠 없음
503	서비스를 사용할 수 없음	콘텐츠 없음

11.2.2.3.4. 태그

- **cluster.open-cluster-management.io**

11.2.2.4. 클러스터 세트 삭제

DELETE /cluster.open-cluster-management.io/v1beta1/managedclustersets/{clusterset_name}

11.2.2.4.1. 설명

단일 클러스터 세트를 삭제합니다.

11.2.2.4.2. 매개 변수

유형	이름	설명	스키마
header	**COOKIE** *필요*	권한 부여: 베어러 {ACCESS_TOKEN} ; ACCESS_TOKEN은 사용자 액세스 토큰입니다.	string
경로	**clusterset_name** *required*	삭제할 클러스터 세트의 이름입니다.	string

11.2.2.4.3. 응답

HTTP 코드	설명	스키마
200	성공	콘텐츠 없음
403	액세스 금지	콘텐츠 없음
404	리소스를 찾을 수 없음	콘텐츠 없음
500	내부 서비스 오류	콘텐츠 없음
503	서비스를 사용할 수 없음	콘텐츠 없음

11.2.2.4.4. 태그

- **cluster.open-cluster-management.io**

11.2.3. 정의

11.2.3.1. Clusterset

이름	스키마
apiVersion *필요*	string
kind *필수*	string

이름	스키마
메타데이터 *필요*	object

11.3. CLUSTERVIEW API (V1ALPHA1)

11.3.1. 개요

이 문서는 **Kubernetes**용 다중 클러스터 엔진의 **clusterview** 리소스에 대한 것입니다. **clusterview** 리소스는 액세스할 수 있는 관리형 클러스터 및 관리형 클러스터 세트의 목록을 볼 수 있는 **CLI** 명령을 제공합니다. 사용 가능한 세 가지 요청은 **list, get, watch**입니다.

11.3.1.1. URI 스키마

BasePath **: /kubernetes/apis**
Schemes **: HTTPS**

11.3.1.2. 태그

- **clusterview.open-cluster-management.io: ID**에서 액세스할 수 있는 관리형 클러스터 목록을 확인합니다.

11.3.2. 경로

11.3.2.1. 관리 클러스터 가져오기

GET /managedclusters.clusterview.open-cluster-management.io

11.3.2.1.1. 설명

액세스할 수 있는 관리형 클러스터 목록을 확인합니다.

11.3.2.1.2. 매개 변수

유형	이름	설명	스키마
header	**COOKIE** *필요*	권한 부여: 베어러 {ACCESS_TOKEN} ; ACCESS_TOKEN은 사용자 액세스 토큰입니다.	string

11.3.2.1.3. 응답

HTTP 코드	설명	스키마
200	성공	콘텐츠 없음
403	액세스 금지	콘텐츠 없음
404	리소스를 찾을 수 없음	콘텐츠 없음
500	내부 서비스 오류	콘텐츠 없음
503	서비스를 사용할 수 없음	콘텐츠 없음

11.3.2.1.4. Use

- **managedcluster/yaml**

11.3.2.1.5. 태그

- **clusterview.open-cluster-management.io**

11.3.2.2. 관리형 클러스터 나열

LIST /managedclusters.clusterview.open-cluster-management.io

11.3.2.2.1. 설명

액세스할 수 있는 관리형 클러스터 목록을 확인합니다.

11.3.2.2.2. 매개 변수

유형	이름	설명	스키마
header	**COOKIE** *필요*	권한 부여: 베어러 {ACCESS_TOKEN} ; ACCESS_TOKEN은 사용자 액세스 토큰입니다.	string
body	**본문** *선택 사항*	관리 클러스터를 나열할 사용자 ID의 이름입니다.	string

11.3.2.2.3. 응답

HTTP 코드	설명	스키마
200	성공	콘텐츠 없음
403	액세스 금지	콘텐츠 없음
404	리소스를 찾을 수 없음	콘텐츠 없음
500	내부 서비스 오류	콘텐츠 없음
503	서비스를 사용할 수 없음	콘텐츠 없음

11.3.2.2.4. Use

- **managedcluster/yaml**

11.3.2.2.5. 태그

- **clusterview.open-cluster-management.io**

11.3.2.2.6. HTTP 요청의 예

11.3.2.2.6.1. 요청 본문

```
{
  "apiVersion" : "clusterview.open-cluster-management.io/v1alpha1",
  "kind" : "ClusterView",
  "metadata" : {
    "name" : "<user_ID>"
  },
  "spec": { },
  "status" : { }
}
```

11.3.2.3. 관리형 클러스터 세트 조사

```
WATCH /managedclusters.clusterview.open-cluster-management.io
```

11.3.2.3.1. 설명

액세스할 수 있는 관리형 클러스터를 확인합니다.

11.3.2.3.2. 매개 변수

유형	이름	설명	스키마
header	**COOKIE** *필요*	권한 부여: 베어러 {ACCESS_TOKEN}; ACCESS_TOKEN은 사용자 액세스 토큰입니다.	string
경로	**clusterview_name** *optional*	조사할 사용자 ID의 이름입니다.	string

11.3.2.3.3. 응답

HTTP 코드	설명	스키마
200	성공	콘텐츠 없음
403	액세스 금지	콘텐츠 없음
404	리소스를 찾을 수 없음	콘텐츠 없음
500	내부 서비스 오류	콘텐츠 없음
503	서비스를 사용할 수 없음	콘텐츠 없음

11.3.2.4. 관리형 클러스터 세트를 나열합니다.

GET /managedclustersets.clusterview.open-cluster-management.io

11.3.2.4.1. 설명

액세스할 수 있는 관리형 클러스터를 나열합니다.

11.3.2.4.2. 매개 변수

유형	이름	설명	스키마
header	**COOKIE** *필요*	권한 부여: 베어러 {ACCESS_TOKEN}; ACCESS_TOKEN은 사용자 액세스 토큰입니다.	string
경로	**clusterview_name** *optional*	조사할 사용자 ID의 이름입니다.	string

11.3.2.4.3. 응답

HTTP 코드	설명	스키마
200	성공	콘텐츠 없음
403	액세스 금지	콘텐츠 없음
404	리소스를 찾을 수 없음	콘텐츠 없음
500	내부 서비스 오류	콘텐츠 없음
503	서비스를 사용할 수 없음	콘텐츠 없음

11.3.2.5. 관리형 클러스터 세트를 나열합니다.

LIST /managedclustersets.clusterview.open-cluster-management.io

11.3.2.5.1. 설명

액세스할 수 있는 관리형 클러스터를 나열합니다.

11.3.2.5.2. 매개 변수

유형	이름	설명	스키마
header	**COOKIE** *필요*	권한 부여: 베어러 {ACCESS_TOKEN}; ACCESS_TOKEN은 사용자 액세스 토큰입니다.	string
경로	**clusterview_name** *optional*	조사할 사용자 ID의 이름입니다.	string

11.3.2.5.3. 응답

HTTP 코드	설명	스키마
200	성공	콘텐츠 없음
403	액세스 금지	콘텐츠 없음
404	리소스를 찾을 수 없음	콘텐츠 없음

HTTP 코드	설명	스키마
500	내부 서비스 오류	콘텐츠 없음
503	서비스를 사용할 수 없음	콘텐츠 없음

11.3.2.6. 관리형 클러스터 세트를 확인합니다.

WATCH /managedclustersets.clusterview.open-cluster-management.io

11.3.2.6.1. 설명

액세스할 수 있는 관리형 클러스터를 확인합니다.

11.3.2.6.2. 매개 변수

유형	이름	설명	스키마
header	**COOKIE** *필요*	권한 부여: 베어러 {ACCESS_TOKEN} ; ACCESS_TOKEN은 사용자 액세스 토큰입니다.	string
경로	**clusterview_name** *optional*	조사할 사용자 ID의 이름입니다.	string

11.3.2.6.3. 응답

HTTP 코드	설명	스키마
200	성공	콘텐츠 없음
403	액세스 금지	콘텐츠 없음
404	리소스를 찾을 수 없음	콘텐츠 없음
500	내부 서비스 오류	콘텐츠 없음
503	서비스를 사용할 수 없음	콘텐츠 없음

11.4. CLUSTERSETBINDINGS API (V1ALPHA1)

11.4.1. 개요

이 문서는 **Kubernetes**의 다중 클러스터 엔진의 **clustersetbinding** 리소스에 대한 것입니다. **Clustersetbinding** 리소스에는 생성, 쿼리, 삭제, 업데이트 등 네 가지 요청이 있습니다.

11.4.1.1. URI 스키마

BasePath **: /kubernetes/apis**
Schemes **: HTTPS**

11.4.1.2. 태그

- **cluster.open-cluster-management.io : clustersetbindings 생성 및 관리**

11.4.2. 경로

11.4.2.1. 모든 clustersetbindings 쿼리

> **GET /cluster.open-cluster-management.io/v1beta1/namespaces/{namespace}/managedclustersetbindings**

11.4.2.1.1. 설명

자세한 내용은 **clustersetbindings**를 쿼리합니다.

11.4.2.1.2. 매개 변수

유형	이름	설명	스키마
header	**COOKIE** *필요*	권한 부여: 베어러 {ACCESS_TOKEN} ; ACCESS_TOKEN은 사용자 액세스 토큰입니다.	string
경로	**네임스페이스** *필요*	사용하려는 네임스페이스(예: default)입니다.	string

11.4.2.1.3. 응답

HTTP 코드	설명	스키마
200	성공	콘텐츠 없음

HTTP 코드	설명	스키마
403	액세스 금지	콘텐츠 없음
404	리소스를 찾을 수 없음	콘텐츠 없음
500	내부 서비스 오류	콘텐츠 없음
503	서비스를 사용할 수 없음	콘텐츠 없음

11.4.2.1.4. Use

- **clustersetbinding/yaml**

11.4.2.1.5. 태그

- **cluster.open-cluster-management.io**

11.4.2.2. clustersetbinding 생성

POST /cluster.open-cluster-management.io/v1beta1/namespaces/{namespace}/managedclustersetbindings

11.4.2.2.1. 설명

clustersetbinding을 생성합니다.

11.4.2.2.2. 매개 변수

유형	이름	설명	스키마
header	**COOKIE** *필요*	권한 부여: 베어러 {ACCESS_TOKEN}; ACCESS_TOKEN은 사용자 액세스 토큰입니다.	string
경로	**네임스페이스** *필요*	사용하려는 네임스페이스(예: default)입니다.	string
body	**본문** *필요*	생성할 clustersetbinding을 설명하는 매개변수입니다.	clustersetbinding

11.4.2.2.3. 응답

HTTP 코드	설명	스키마
200	성공	콘텐츠 없음
403	액세스 금지	콘텐츠 없음
404	리소스를 찾을 수 없음	콘텐츠 없음
500	내부 서비스 오류	콘텐츠 없음
503	서비스를 사용할 수 없음	콘텐츠 없음

11.4.2.2.4. Use

- **clustersetbinding/yaml**

11.4.2.2.5. 태그

- **cluster.open-cluster-management.io**

11.4.2.2.6. HTTP 요청의 예

11.4.2.2.6.1. 요청 본문

```
{
  "apiVersion" : "cluster.open-cluster-management.io/v1",
  "kind" : "ManagedClusterSetBinding",
  "metadata" : {
    "name" : "clusterset1",
    "namespace" : "ns1"
  },
 "spec": {
    "clusterSet": "clusterset1"
  },
  "status" : { }
}
```

11.4.2.3. 단일 clustersetbinding 쿼리

```
GET /cluster.open-cluster-management.io/v1beta1/namespaces/{namespace}/managedclustersetbindings/{clustersetbinding_name}
```

11.4.2.3.1. 설명

자세한 내용은 단일 clustersetbinding을 쿼리합니다.

11.4.2.3.2. 매개 변수

유형	이름	설명	스키마
header	**COOKIE** *필요*	권한 부여: 베어러 {ACCESS_TOKEN}; ACCESS_TOKEN은 사용자 액세스 토큰입니다.	string
경로	**네임스페이스** *필요*	사용하려는 네임스페이스(예: default)입니다.	string
경로	**clustersetbinding_name** *required*	쿼리할 clustersetbinding의 이름입니다.	string

11.4.2.3.3. 응답

HTTP 코드	설명	스키마
200	성공	콘텐츠 없음
403	액세스 금지	콘텐츠 없음
404	리소스를 찾을 수 없음	콘텐츠 없음
500	내부 서비스 오류	콘텐츠 없음
503	서비스를 사용할 수 없음	콘텐츠 없음

11.4.2.3.4. 태그

- **cluster.open-cluster-management.io**

11.4.2.4. clustersetbinding 삭제

```
DELETE /cluster.open-cluster-management.io/v1beta1/managedclustersetbindings/{clustersetbinding_name}
```

11.4.2.4.1. 설명

단일 clustersetbinding을 삭제합니다.

11.4.2.4.2. 매개 변수

유형	이름	설명	스키마
header	**COOKIE** *필요*	권한 부여: 베어러 {ACCESS_TOKEN}; ACCESS_TOKEN은 사용자 액세스 토큰입니다.	string
경로	**네임스페이스** *필요*	사용하려는 네임스페이스(예: default)입니다.	string
경로	**clustersetbinding_name** *required*	삭제할 clustersetbinding의 이름입니다.	string

11.4.2.4.3. 응답

HTTP 코드	설명	스키마
200	성공	콘텐츠 없음
403	액세스 금지	콘텐츠 없음
404	리소스를 찾을 수 없음	콘텐츠 없음
500	내부 서비스 오류	콘텐츠 없음
503	서비스를 사용할 수 없음	콘텐츠 없음

11.4.2.4.4. 태그

- **cluster.open-cluster-management.io**

11.4.3. 정의

11.4.3.1. clustersetbinding

이름	스키마
apiVersion *필요*	string
kind *필수*	string

이름	스키마
메타데이터 *필요*	object
spec *필수*	spec

spec

이름	스키마
clusterSet *required*	string

11.5. API

11.5.1. 개요

이 문서는 **Kubernetes**용 다중 클러스터 엔진의 **MultiClusterEngine** 리소스에 대한 것입니다. **MultiClusterEngine** 리소스에는 생성, 쿼리, 삭제, 업데이트 등 네 가지 요청이 있습니다.

11.5.1.1. URI 스키마

BasePath **: /kubernetes/apis**
Schemes **: HTTPS**

11.5.1.2. 태그

- **Multiclusterengines.multicluster.openshift.io : MultiClusterEngines 생성 및 관리**

11.5.2. 경로

11.5.2.1. Create a MultiClusterEngine

POST /apis/multicluster.openshift.io/v1alpha1/multiclusterengines

11.5.2.1.1. 설명

MultiClusterEngine을 생성합니다.

11.5.2.1.2. 매개 변수

유형	이름	설명	스키마
header	**COOKIE** *필요*	권한 부여: 베어러 {ACCESS_TOKEN} ; ACCESS_TOKEN은 사용자 액세스 토큰입니다.	string
body	**본문** *필요*	생성할 MultiClusterEngine을 설명하는 매개변수입니다.	MultiClusterEngine

11.5.2.1.3. 응답

HTTP 코드	설명	스키마
200	성공	콘텐츠 없음
403	액세스 금지	콘텐츠 없음
404	리소스를 찾을 수 없음	콘텐츠 없음
500	내부 서비스 오류	콘텐츠 없음
503	서비스를 사용할 수 없음	콘텐츠 없음

11.5.2.1.4. Use

- **MultiClusterEngines/yaml**

11.5.2.1.5. 태그

- **multiclusterengines.multicluster.openshift.io**

11.5.2.1.5.1. 요청 본문

```
{
  "apiVersion": "apiextensions.k8s.io/v1",
  "kind": "CustomResourceDefinition",
  "metadata": {
    "annotations": {
      "controller-gen.kubebuilder.io/version": "v0.4.1"
    },
```

```
    "creationTimestamp": null,
    "name": "multiclusterengines.multicluster.openshift.io"
  },
  "spec": {
    "group": "multicluster.openshift.io",
    "names": {
      "kind": "MultiClusterEngine",
      "listKind": "MultiClusterEngineList",
      "plural": "multiclusterengines",
      "shortNames": [
        "mce"
      ],
      "singular": "multiclusterengine"
    },
    "scope": "Cluster",
    "versions": [
      {
        "additionalPrinterColumns": [
          {
            "description": "The overall state of the MultiClusterEngine",
            "jsonPath": ".status.phase",
            "name": "Status",
            "type": "string"
          },
          {
            "jsonPath": ".metadata.creationTimestamp",
            "name": "Age",
            "type": "date"
          }
        ],
        "name": "v1alpha1",
        "schema": {
          "openAPIV3Schema": {
            "description": "MultiClusterEngine is the Schema for the multiclusterengines\nAPI",
            "properties": {
              "apiVersion": {
                "description": "APIVersion defines the versioned schema of this representation\nof an object. Servers should convert recognized schemas to the latest\ninternal value, and may reject unrecognized values. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources",
                "type": "string"
              },
              "kind": {
                "description": "Kind is a string value representing the REST resource this\nobject represents. Servers may infer this from the endpoint the client\nsubmits requests to. Cannot be updated. In CamelCase. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds",
                "type": "string"
              },
              "metadata": {
                "type": "object"
              },
              "spec": {
                "description": "MultiClusterEngineSpec defines the desired state of MultiClusterEngine",
                "properties": {
```

```
            "imagePullSecret": {
              "description": "Override pull secret for accessing MultiClusterEngine\noperand
and endpoint images",
              "type": "string"
            },
            "nodeSelector": {
              "additionalProperties": {
                "type": "string"
              },
              "description": "Set the nodeselectors",
              "type": "object"
            },
            "targetNamespace": {
              "description": "Location where MCE resources will be placed",
              "type": "string"
            },
            "tolerations": {
              "description": "Tolerations causes all components to tolerate any taints.",
              "items": {
                "description": "The pod this Toleration is attached to tolerates any\ntaint that
matches the triple <key,value,effect> using the matching\noperator <operator>.",
                "properties": {
                  "effect": {
                    "description": "Effect indicates the taint effect to match. Empty\nmeans match
all taint effects. When specified, allowed values\nare NoSchedule, PreferNoSchedule and
NoExecute.",
                    "type": "string"
                  },
                  "key": {
                    "description": "Key is the taint key that the toleration applies\nto. Empty
means match all taint keys. If the key is empty,\noperator must be Exists; this combination
means to match all\nvalues and all keys.",
                    "type": "string"
                  },
                  "operator": {
                    "description": "Operator represents a key's relationship to the\nvalue. Valid
operators are Exists and Equal. Defaults to Equal.\nExists is equivalent to wildcard for value,
so that a pod\ncan tolerate all taints of a particular category.",
                    "type": "string"
                  },
                  "tolerationSeconds": {
                    "description": "TolerationSeconds represents the period of time\nthe toleration
(which must be of effect NoExecute, otherwise\nthis field is ignored) tolerates the taint. By
default, it\nis not set, which means tolerate the taint forever (do not\nevict). Zero and negative
values will be treated as 0 (evict\nimmediately) by the system.",
                    "format": "int64",
                    "type": "integer"
                  },
                  "value": {
                    "description": "Value is the taint value the toleration matches\nto. If the
operator is Exists, the value should be empty,\notherwise just a regular string.",
                    "type": "string"
                  }
                },
                "type": "object"
              },
```

```
              "type": "array"
             }
            },
            "type": "object"
           },
           "status": {
            "description": "MultiClusterEngineStatus defines the observed state of
MultiClusterEngine",
            "properties": {
             "components": {
              "items": {
               "description": "ComponentCondition contains condition information for\ntracked
components",
               "properties": {
                "kind": {
                 "description": "The resource kind this condition represents",
                 "type": "string"
                },
                "lastTransitionTime": {
                 "description": "LastTransitionTime is the last time the condition\nchanged
from one status to another.",
                 "format": "date-time",
                 "type": "string"
                },
                "message": {
                 "description": "Message is a human-readable message indicating\ndetails
about the last status change.",
                 "type": "string"
                },
                "name": {
                 "description": "The component name",
                 "type": "string"
                },
                "reason": {
                 "description": "Reason is a (brief) reason for the condition's\nlast status
change.",
                 "type": "string"
                },
                "status": {
                 "description": "Status is the status of the condition. One of True,\nFalse,
Unknown.",
                 "type": "string"
                },
                "type": {
                 "description": "Type is the type of the cluster condition.",
                 "type": "string"
                }
               },
               "type": "object"
              },
              "type": "array"
             },
             "conditions": {
              "items": {
               "properties": {
                "lastTransitionTime": {
```

```
                    "description": "LastTransitionTime is the last time the condition\nchanged
from one status to another.",
                    "format": "date-time",
                    "type": "string"
                   },
                   "lastUpdateTime": {
                    "description": "The last time this condition was updated.",
                    "format": "date-time",
                    "type": "string"
                   },
                   "message": {
                    "description": "Message is a human-readable message indicating\ndetails
about the last status change.",
                    "type": "string"
                   },
                   "reason": {
                    "description": "Reason is a (brief) reason for the condition's\nlast status
change.",
                    "type": "string"
                   },
                   "status": {
                    "description": "Status is the status of the condition. One of True,\nFalse,
Unknown.",
                    "type": "string"
                   },
                   "type": {
                    "description": "Type is the type of the cluster condition.",
                    "type": "string"
                   }
                  },
                  "type": "object"
                 },
                 "type": "array"
                },
                "phase": {
                 "description": "Latest observed overall state",
                 "type": "string"
                }
               },
               "type": "object"
              }
             },
             "type": "object"
            }
           },
           "served": true,
           "storage": true,
           "subresources": {
            "status": {}
           }
          }
         ]
        },
        "status": {
         "acceptedNames": {
          "kind": "",
```

```
      "plural": ""
    },
    "conditions": [],
    "storedVersions": []
  }
}
```

11.5.2.2. 모든 MultiClusterEngines 쿼리

```
GET /apis/multicluster.openshift.io/v1alpha1/multiclusterengines
```

11.5.2.2.1. 설명

자세한 내용은 다중 클러스터 엔진을 쿼리합니다.

11.5.2.2.2. 매개 변수

유형	이름	설명	스키마
header	COOKIE *필요*	권한 부여: 베어러 {ACCESS_TOKEN} ; ACCESS_TOKEN은 사용자 액세스 토큰입니다.	string

11.5.2.2.3. 응답

HTTP 코드	설명	스키마
200	성공	콘텐츠 없음
403	액세스 금지	콘텐츠 없음
404	리소스를 찾을 수 없음	콘텐츠 없음
500	내부 서비스 오류	콘텐츠 없음
503	서비스를 사용할 수 없음	콘텐츠 없음

11.5.2.2.4. Use

- operator/yaml

11.5.2.2.5. 태그

- **multiclusterengines.multicluster.openshift.io**

11.5.2.3. MultiClusterEngine Operator 삭제

DELETE /apis/multicluster.openshift.io/v1alpha1/multiclusterengines/{name}

11.5.2.3.1. 매개 변수

유형	이름	설명	스키마
header	**COOKIE** *필요*	권한 부여: 베어러 {ACCESS_TOKEN} ; ACCESS_TOKEN은 사용자 액세스 토큰입니다.	string
경로	**이름** *필요*	삭제할 다중 클러스터 엔진의 이름입니다.	string

11.5.2.3.2. 응답

HTTP 코드	설명	스키마
200	성공	콘텐츠 없음
403	액세스 금지	콘텐츠 없음
404	리소스를 찾을 수 없음	콘텐츠 없음
500	내부 서비스 오류	콘텐츠 없음
503	서비스를 사용할 수 없음	콘텐츠 없음

11.5.2.3.3. 태그

- **multiclusterengines.multicluster.openshift.io**

11.5.3. 정의

11.5.3.1. MultiClusterEngine

이름	설명	스키마

이름	설명	스키마
apiVersion *필요*	버전이 지정된 MultiClusterEngines 스키마입니다.	string
kind *필수*	REST 리소스를 나타내는 문자열 값입니다.	string
메타데이터 *필요*	리소스를 정의하는 규칙을 설명합니다.	object
spec *필수*	MultiClusterEngineSpec은 원하는 MultiClusterEngine 상태를 정의합니다.	*사양 목록보기*

11.5.3.2. 사양 목록

이름	설명	스키마
nodeSelector *optional*	노드 선택기를 설정합니다.	map[string]string
imagePullSecret *optional*	MultiClusterEngine 피연산자 및 끝점 이미지에 액세스하기 위해 가져오기 보안을 재정의합니다.	string
허용 오차 *선택 사항*	허용 오차는 모든 구성 요소가 테인트를 허용하도록 합니다.	[]corev1.Toleration
targetNamespace *optional*	MCE 리소스가 배치될 위치입니다.	string

11.6. PLACEMENTS API(V1ALPHA1)

11.6.1. 개요

이 문서는 Kubernetes용 다중 클러스터 엔진의 배치 리소스에 대한 것입니다. 배치 리소스에는 생성, 쿼리, 삭제 및 업데이트 등 네 가지 요청이 있습니다.

11.6.1.1. URI 스키마

***BasePath* : /kubernetes/apis**
***Schemes* : HTTPS**

11.6.1.2. 태그

- **cluster.open-cluster-management.io : 배치 생성 및 관리**

11.6.2. 경로

11.6.2.1. 모든 배치 쿼리

GET /cluster.open-cluster-management.io/v1alpha1/namespaces/{namespace}/placements

11.6.2.1.1. 설명

자세한 내용은 배치를 쿼리합니다.

11.6.2.1.2. 매개 변수

유형	이름	설명	스키마
header	**COOKIE** *필요*	권한 부여: 베어러 {ACCESS_TOKEN} ; ACCESS_TOKEN은 사용자 액세스 토큰입니다.	string

11.6.2.1.3. 응답

HTTP 코드	설명	스키마
200	성공	콘텐츠 없음
403	액세스 금지	콘텐츠 없음
404	리소스를 찾을 수 없음	콘텐츠 없음
500	내부 서비스 오류	콘텐츠 없음
503	서비스를 사용할 수 없음	콘텐츠 없음

11.6.2.1.4. Use

- **placement/yaml**

11.6.2.1.5. 태그

- **cluster.open-cluster-management.io**

11.6.2.2. 배치 생성

POST /cluster.open-cluster-management.io/v1alpha1/namespaces/{namespace}/placements

11.6.2.2.1. 설명

배치 생성.

11.6.2.2.2. 매개 변수

유형	이름	설명	스키마
header	**COOKIE** *필요*	권한 부여: 베어러 {ACCESS_TOKEN}; ACCESS_TOKEN은 사용자 액세스 토큰입니다.	string
body	**본문** *필요*	생성할 배치를 설명하는 매개변수입니다.	placement

11.6.2.2.3. 응답

HTTP 코드	설명	스키마
200	성공	콘텐츠 없음
403	액세스 금지	콘텐츠 없음
404	리소스를 찾을 수 없음	콘텐츠 없음
500	내부 서비스 오류	콘텐츠 없음
503	서비스를 사용할 수 없음	콘텐츠 없음

11.6.2.2.4. Use

- **placement/yaml**

11.6.2.2.5. 태그

- cluster.open-cluster-management.io

11.6.2.2.6. HTTP 요청의 예

11.6.2.2.6.1. 요청 본문

```
{
  "apiVersion" : "cluster.open-cluster-management.io/v1alpha1",
  "kind" : "Placement",
  "metadata" : {
    "name" : "placement1",
    "namespace": "ns1"
  },
  "spec": {
    "predicates": [
      {
        "requiredClusterSelector": {
          "labelSelector": {
            "matchLabels": {
              "vendor": "OpenShift"
            }
          }
        }
      }
    ]
  },
  "status" : { }
}
```

11.6.2.3. 단일 배치 쿼리

```
GET /cluster.open-cluster-
management.io/v1alpha1/namespaces/{namespace}/placements/{placement_name}
```

11.6.2.3.1. 설명

자세한 내용은 단일 배치를 쿼리합니다.

11.6.2.3.2. 매개 변수

유형	이름	설명	스키마
header	COOKIE *필요*	권한 부여: 베어러 {ACCESS_TOKEN} ; ACCESS_TOKEN은 사용자 액세스 토큰입니다.	string
경로	placement_name *required*	쿼리할 배치의 이름입니다.	string

11.6.2.3.3. 응답

HTTP 코드	설명	스키마
200	성공	콘텐츠 없음
403	액세스 금지	콘텐츠 없음
404	리소스를 찾을 수 없음	콘텐츠 없음
500	내부 서비스 오류	콘텐츠 없음
503	서비스를 사용할 수 없음	콘텐츠 없음

11.6.2.3.4. 태그

- **cluster.open-cluster-management.io**

11.6.2.4. 배치 삭제

DELETE /cluster.open-cluster-management.io/v1alpha1/namespaces/{namespace}/placements/{placement_name}

11.6.2.4.1. 설명

단일 배치 삭제.

11.6.2.4.2. 매개 변수

유형	이름	설명	스키마
header	**COOKIE** *필요*	권한 부여: 베어러 {ACCESS_TOKEN} ; ACCESS_TOKEN은 사용자 액세스 토큰입니다.	string
경로	**placement_name** *required*	삭제할 배치의 이름입니다.	string

11.6.2.4.3. 응답

HTTP 코드	설명	스키마
200	성공	콘텐츠 없음
403	액세스 금지	콘텐츠 없음
404	리소스를 찾을 수 없음	콘텐츠 없음
500	내부 서비스 오류	콘텐츠 없음
503	서비스를 사용할 수 없음	콘텐츠 없음

11.6.2.4.4. 태그

- **cluster.open-cluster-management.io**

11.6.3. 정의

11.6.3.1. placement

이름	설명	스키마
apiVersion *필요*	배치의 버전이 지정된 스키마입니다.	string
kind *필수*	REST 리소스를 나타내는 문자열 값입니다.	string
메타데이터 *필요*	배치의 메타 데이터입니다.	object
spec *필수*	배치 사양입니다.	spec

spec

이름	설명	스키마

이름	설명	스키마
ClusterSets *선택 사항*	ManagedClusters가 선택된 ManagedClusterSets의 하위 집합입니다. 비어 있는 경우 배치 네임스페이스에 바인딩된 ManagedClusterSets에서 ManagedClusters가 선택됩니다. 그렇지 않으면 ManagedClusters가 이 하위 집합의 교집합에서 선택되고 ManagedClusterSets는 placement 네임스페이스에 바인딩됩니다.	문자열 배열
numberOfClusters *optional*	선택할 ManagedCluster의 수입니다.	정수(int32)
서술자 *선택 사항*	ManagedClusters를 선택할 클러스터 서술자의 하위 집합입니다. 조건부 논리는 *OR* 입니다.	clusterPredicate 배열

clusterPredicate

이름	설명	스키마
requiredClusterSelector *optional*	레이블 및 클러스터 클레임을 사용하여 ManagedClusters를 선택하는 클러스터 선택기입니다.	clusterSelector

clusterSelector

이름	설명	스키마
labelSelector *optional*	라벨별 ManagedClusters의 선택기입니다.	object
claimSelector *optional*	클레임별 ManagedClusters의 선택기입니다.	clusterClaimSelector

clusterClaimSelector

이름	설명	스키마
matchExpressions *optional*	클러스터 클레임 선택기 요구 사항의 하위 집합입니다. 조건부 논리는 *AND* 입니다.	< object > array

11.7. PLACEMENTDECISIONS API (V1ALPHA1)

11.7.1. 개요

이 문서는 **Kubernetes**용 다중 클러스터 엔진의 **PlacementDecision** 리소스를 위한 것입니다. **PlacementDecision** 리소스에는 네 가지 요청(생성, 쿼리, 삭제 및 업데이트)이 있습니다.

11.7.1.1. URI 스키마

BasePath **: /kubernetes/apis**
Schemes **: HTTPS**

11.7.1.2. 태그

- **cluster.open-cluster-management.io : PlacementDecisions 생성 및 관리**

11.7.2. 경로

11.7.2.1. 모든 PlacementDecisions 쿼리

GET /cluster.open-cluster-management.io/v1alpha1/namespaces/{namespace}/placementdecisions

11.7.2.1.1. 설명

자세한 내용은 **PlacementDecisions**를 쿼리합니다.

11.7.2.1.2. 매개 변수

유형	이름	설명	스키마
header	COOKIE *필요*	권한 부여: 베어러 {ACCESS_TOKEN} ; ACCESS_TOKEN은 사용자 액세스 토큰입니다.	string

11.7.2.1.3. 응답

HTTP 코드	설명	스키마
200	성공	콘텐츠 없음
403	액세스 금지	콘텐츠 없음
404	리소스를 찾을 수 없음	콘텐츠 없음
500	내부 서비스 오류	콘텐츠 없음
503	서비스를 사용할 수 없음	콘텐츠 없음

11.7.2.1.4. Use

- **placementdecision/yaml**

11.7.2.1.5. 태그

- **cluster.open-cluster-management.io**

11.7.2.2. PlacementDecision 생성

> **POST /cluster.open-cluster-management.io/v1alpha1/namespaces/{namespace}/placementdecisions**

11.7.2.2.1. 설명

PlacementDecision 생성.

11.7.2.2.2. 매개 변수

유형	이름	설명	스키마
header	**COOKIE** *필요*	권한 부여: 베어러 {ACCESS_TOKEN}; ACCESS_TOKEN은 사용자 액세스 토큰입니다.	string
body	**본문** *필요*	생성할 PlacementDecision을 설명하는 매개변수입니다.	PlacementDecision

11.7.2.2.3. 응답

HTTP 코드	설명	스키마
200	성공	콘텐츠 없음
403	액세스 금지	콘텐츠 없음
404	리소스를 찾을 수 없음	콘텐츠 없음
500	내부 서비스 오류	콘텐츠 없음
503	서비스를 사용할 수 없음	콘텐츠 없음

11.7.2.2.4. Use

- **placementdecision/yaml**

11.7.2.2.5. 태그

- **cluster.open-cluster-management.io**

11.7.2.2.6. HTTP 요청의 예

11.7.2.2.6.1. 요청 본문

```
{
  "apiVersion" : "cluster.open-cluster-management.io/v1alpha1",
  "kind" : "PlacementDecision",
  "metadata" : {
    "labels" : {
      "cluster.open-cluster-management.io/placement" : "placement1"
    },
    "name" : "placement1-decision1",
    "namespace": "ns1"
  },
  "status" : { }
}
```

11.7.2.3. 단일 PlacementDecision 쿼리

```
GET /cluster.open-cluster-management.io/v1alpha1/namespaces/{namespace}/placementdecisions/{placementdecision_name}
```

11.7.2.3.1. 설명

자세한 내용은 단일 PlacementDecision을 쿼리합니다.

11.7.2.3.2. 매개 변수

유형	이름	설명	스키마
header	**COOKIE** *필요*	권한 부여: 베어러 {ACCESS_TOKEN} ; ACCESS_TOKEN은 사용자 액세스 토큰입니다.	string
경로	**placementdecision_name** *required*	쿼리할 PlacementDecision의 이름입니다.	string

11.7.2.3.3. 응답

HTTP 코드	설명	스키마
200	성공	콘텐츠 없음
403	액세스 금지	콘텐츠 없음
404	리소스를 찾을 수 없음	콘텐츠 없음
500	내부 서비스 오류	콘텐츠 없음
503	서비스를 사용할 수 없음	콘텐츠 없음

11.7.2.3.4. 태그

- **cluster.open-cluster-management.io**

11.7.2.4. PlacementDecision 삭제

```
DELETE /cluster.open-cluster-management.io/v1alpha1/namespaces/{namespace}/placementdecisions/{placementdecision_name}
```

11.7.2.4.1. 설명

단일 PlacementDecision 삭제

11.7.2.4.2. 매개 변수

유형	이름	설명	스키마
header	**COOKIE** *필요*	권한 부여: 베어러 {ACCESS_TOKEN}; ACCESS_TOKEN은 사용자 액세스 토큰입니다.	string
경로	**placementdecision_name** *required*	삭제할 PlacementDecision의 이름입니다.	string

11.7.2.4.3. 응답

HTTP 코드	설명	스키마
200	성공	콘텐츠 없음
403	액세스 금지	콘텐츠 없음
404	리소스를 찾을 수 없음	콘텐츠 없음
500	내부 서비스 오류	콘텐츠 없음
503	서비스를 사용할 수 없음	콘텐츠 없음

11.7.2.4.4. 태그

- **cluster.open-cluster-management.io**

11.7.3. 정의

11.7.3.1. PlacementDecision

이름	설명	스키마
apiVersion *필요*	PlacementDecision의 버전이 지정된 스키마입니다.	string
kind *필수*	REST 리소스를 나타내는 문자열 값입니다.	string
메타데이터 *필요*	PlacementDecision의 메타 데이터입니다.	object

11.8. 관리형 서비스 계정(기술 프리뷰)

11.8.1. 개요

이 문서는 **Kubernetes Operator**용 다중 클러스터 엔진의 **ManagedServiceAccount** 리소스에 대한 것입니다. **ManagedServiceAccount** 리소스에는 생성, 쿼리, 삭제, 업데이트 등 네 가지 요청이 있습니다.

11.8.1.1. URI 스키마

BasePath **: /kubernetes/apis**
Schemes **: HTTPS**

11.8.1.2. 태그

- **Managed serviceaccounts.multicluster.openshift.io': ManagedServiceAccounts생성 및 관리**

11.8.2. 경로

11.8.2.1. Create a ManagedServiceAccount

POST /apis/multicluster.openshift.io/v1alpha1/ManagedServiceAccounts

11.8.2.1.1. 설명

ManagedServiceAccount 를 생성합니다.

11.8.2.1.2. 매개 변수

유형	이름	설명	스키마
header	**COOKIE** *필요*	권한 부여: 베어러 {ACCESS_TOKEN} ; ACCESS_TOKEN은 사용자 액세스 토큰입니다.	string
body	**본문** *필요*	생성할 ManagedServiceAccount를 설명하는 매개변수입니다.	ManagedServiceAccount

11.8.2.1.3. 응답

HTTP 코드	설명	스키마
200	성공	콘텐츠 없음
403	액세스 금지	콘텐츠 없음
404	리소스를 찾을 수 없음	콘텐츠 없음
500	내부 서비스 오류	콘텐츠 없음
503	서비스를 사용할 수 없음	콘텐츠 없음

11.8.2.1.4. Use

- **managedserviceaccount/yaml**

11.8.2.1.5. 태그

- **managedserviceaccount.multicluster.openshift.io**

11.8.2.1.5.1. 요청 본문


```
{
  "apiVersion": "apiextensions.k8s.io/v1",
  "kind": "CustomResourceDefinition",
  "metadata": {
    "annotations": {
      "controller-gen.kubebuilder.io/version": "v0.4.1"
    },
    "creationTimestamp": null,
    "name": "managedserviceaccount.authentication.open-cluster-management.io"
  },
  "spec": {
    "group": "authentication.open-cluster-management.io",
    "names": {
      "kind": "ManagedServiceAccount",
      "listKind": "ManagedServiceAccountList",
      "plural": "managedserviceaccounts",
      "singular": "managedserviceaccount"
    },
    "scope": "Namespaced",
    "versions": [
      {
        "name": "v1alpha1",
        "schema": {
          "openAPIV3Schema": {
            "description": "ManagedServiceAccount is the Schema for the
```

```
managedserviceaccounts\nAPI",
          "properties": {
           "apiVersion": {
            "description": "APIVersion defines the versioned schema of this representation\nof an object. Servers should convert recognized schemas to the latest\ninternal value, and may reject unrecognized values. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources",
            "type": "string"
           },
           "kind": {
            "description": "Kind is a string value representing the REST resource this\nobject represents. Servers may infer this from the endpoint the client\nsubmits requests to. Cannot be updated. In CamelCase. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds",
            "type": "string"
           },
           "metadata": {
            "type": "object"
           },
           "spec": {
            "description": "ManagedServiceAccountSpec defines the desired state of ManagedServiceAccount",
            "properties": {
             "rotation": {
              "description": "Rotation is the policy for rotation the credentials.",
              "properties": {
               "enabled": {
                "default": true,
                "description": "Enabled prescribes whether the ServiceAccount token\nwill be rotated from the upstream",
                "type": "boolean"
               },
               "validity": {
                "default": "8640h0m0s",
                "description": "Validity is the duration for which the signed ServiceAccount\ntoken is valid.",
                "type": "string"
               }
              },
              "type": "object"
             },
             "ttlSecondsAfterCreation": {
              "description": "ttlSecondsAfterCreation limits the lifetime of a ManagedServiceAccount.\nIf the ttlSecondsAfterCreation field is set, the ManagedServiceAccount\nwill be automatically deleted regardless of the ManagedServiceAccount's\nstatus. When the ManagedServiceAccount is deleted, its lifecycle\nguarantees (e.g. finalizers) will be honored. If this field is unset,\nthe ManagedServiceAccount won't be automatically deleted. If this\nfield is set to zero, the ManagedServiceAccount becomes eligible\nfor deletion immediately after its creation. In order to use ttlSecondsAfterCreation,\nthe EphemeralIdentity feature gate must be enabled.",
              "exclusiveMinimum": true,
              "format": "int32",
              "minimum": 0,
              "type": "integer"
             }
            },
```

```
            "required": [
              "rotation"
            ],
            "type": "object"
          },
          "status": {
            "description": "ManagedServiceAccountStatus defines the observed state
of\nManagedServiceAccount",
            "properties": {
              "conditions": {
                "description": "Conditions is the condition list.",
                "items": {
                  "description": "Condition contains details for one aspect of the current\nstate of
this API Resource. --- This struct is intended for direct\nuse as an array at the field path
.status.conditions.  For example,\ntype FooStatus struct{     // Represents the observations of
a\nfoo's current state.     // Known .status.conditions.type are:\n\"Available\", \"Progressing\",
and \"Degraded\"     // +patchMergeKey=type\n    // +patchStrategy=merge     // +listType=map
// +listMapKey=type\n    Conditions []metav1.Condition
`json:\"conditions,omitempty\"\npatchStrategy:\"merge\" patchMergeKey:\"type\"
protobuf:\"bytes,1,rep,name=conditions\"`\n\n     // other fields }",
                  "properties": {
                    "lastTransitionTime": {
                      "description": "lastTransitionTime is the last time the condition\ntransitioned
from one status to another. This should be when\nthe underlying condition changed.  If that is
not known, then\nusing the time when the API field changed is acceptable.",
                      "format": "date-time",
                      "type": "string"
                    },
                    "message": {
                      "description": "message is a human readable message indicating\ndetails
about the transition. This may be an empty string.",
                      "maxLength": 32768,
                      "type": "string"
                    },
                    "observedGeneration": {
                      "description": "observedGeneration represents the .metadata.generation\nthat
the condition was set based upon. For instance, if .metadata.generation\nis currently 12, but
the .status.conditions[x].observedGeneration\nis 9, the condition is out of date with respect to
the current\nstate of the instance.",
                      "format": "int64",
                      "minimum": 0,
                      "type": "integer"
                    },
                    "reason": {
                      "description": "reason contains a programmatic identifier indicating\nthe
reason for the condition's last transition. Producers\nof specific condition types may define
expected values and\nmeanings for this field, and whether the values are considered\na
guaranteed API. The value should be a CamelCase string.\nThis field may not be empty.",
                      "maxLength": 1024,
                      "minLength": 1,
                      "pattern": "^[A-Za-z]([A-Za-z0-9_,:]*[A-Za-z0-9_])?$",
                      "type": "string"
                    },
                    "status": {
                      "description": "status of the condition, one of True, False, Unknown.",
                      "enum": [
```

```
                    "True",
                    "False",
                    "Unknown"
                   ],
                   "type": "string"
                  },
                  "type": {
                   "description": "type of condition in CamelCase or in
foo.example.com/CamelCase.\n--- Many .condition.type values are consistent across
resources\nlike Available, but because arbitrary conditions can be useful\n(see
.node.status.conditions), the ability to deconflict is\nimportant. The regex it matches is
(dns1123SubdomainFmt/)?(qualifiedNameFmt)",
                   "maxLength": 316,
                   "pattern": "^([a-z0-9]([-a-z0-9]*[a-z0-9])?(\\.[a-z0-9]([-a-z0-9]*[a-z0-9])?)*/)?(([A-
Za-z0-9][-A-Za-z0-9_.]*)?[A-Za-z0-9])$",
                   "type": "string"
                  }
                 },
                 "required": [
                  "lastTransitionTime",
                  "message",
                  "reason",
                  "status",
                  "type"
                 ],
                 "type": "object"
                },
                "type": "array"
               },
               "expirationTimestamp": {
                "description": "ExpirationTimestamp is the time when the token will expire.",
                "format": "date-time",
                "type": "string"
               },
               "tokenSecretRef": {
                "description": "TokenSecretRef is a reference to the corresponding
ServiceAccount's\nSecret, which stores the CA certficate and token from the
managed\ncluster.",
                "properties": {
                 "lastRefreshTimestamp": {
                  "description": "LastRefreshTimestamp is the timestamp indicating\nwhen the
token in the Secret is refreshed.",
                  "format": "date-time",
                  "type": "string"
                 },
                 "name": {
                  "description": "Name is the name of the referenced secret.",
                  "type": "string"
                 }
                },
                "required": [
                 "lastRefreshTimestamp",
                 "name"
                ],
                "type": "object"
               }
```

```
            },
            "type": "object"
          }
        },
        "type": "object"
      }
    },
    "served": true,
    "storage": true,
    "subresources": {
      "status": {}
    }
  }
 ]
},
"status": {
 "acceptedNames": {
  "kind": "",
  "plural": ""
 },
 "conditions": [],
 "storedVersions": []
}
}
```


11.8.2.2. 하나의 ManagedServiceAccount 쿼리

```
GET /cluster.open-cluster-management.io/v1alpha1/namespaces/{namespace}/managedserviceaccounts/{managedserviceaccount_name}
```

11.8.2.2.1. 설명

자세한 내용은 하나의 **ManagedServiceAccount** 를 쿼리합니다.

11.8.2.2.2. 매개 변수

유형	이름	설명	스키마
header	COOKIE *필요*	권한 부여: 베어러 {ACCESS_TOKEN} ; ACCESS_TOKEN은 사용자 액세스 토큰입니다.	string
경로	managedserviceaccount_name *required*	쿼리할 **ManagedServiceAccount** 의 이름입니다.	string

11.8.2.2.3. 응답

HTTP 코드	설명	스키마
200	성공	콘텐츠 없음
403	액세스 금지	콘텐츠 없음
404	리소스를 찾을 수 없음	콘텐츠 없음
500	내부 서비스 오류	콘텐츠 없음
503	서비스를 사용할 수 없음	콘텐츠 없음

11.8.2.2.4. 태그

- **cluster.open-cluster-management.io**

11.8.2.3. ManagedServiceAccount삭제

> **DELETE /cluster.open-cluster-management.io/v1alpha1/namespaces/{namespace}/managedserviceaccounts/{managedserviceaccount_name}**

11.8.2.3.1. 설명

하나의 **ManagedServiceAccount** 를 삭제합니다.

11.8.2.3.2. 매개 변수

유형	이름	설명	스키마
header	**COOKIE** *필요*	권한 부여: 베어러 {ACCESS_TOKEN} ; ACCESS_TOKEN은 사용자 액세스 토큰입니다.	string
경로	**managedserviceaccount_name** *required*	삭제할 **ManagedServiceAccount** 의 이름입니다.	string

11.8.2.3.3. 응답

HTTP 코드	설명	스키마
200	성공	콘텐츠 없음
403	액세스 금지	콘텐츠 없음
404	리소스를 찾을 수 없음	콘텐츠 없음
500	내부 서비스 오류	콘텐츠 없음
503	서비스를 사용할 수 없음	콘텐츠 없음

11.8.2.3.4. 태그

- **cluster.open-cluster-management.io**

11.8.3. 정의

11.8.3.1. ManagedServiceAccount

이름	설명	스키마
apiVersion *필요*	**ManagedServiceAccount** 의 버전이 지정된 스키마입니다.	string
kind *필수*	REST 리소스를 나타내는 문자열 값입니다.	string
메타데이터 *필요*	**ManagedServiceAccount** 의 메타 데이터입니다.	object
spec *필수*	**ManagedServiceAccount** 사양입니다.	

12장. 설치 제거

Kubernetes용 다중 클러스터 엔진을 제거하면 프로세스의 두 가지 수준인 *사용자 정의 리소스 제거* 및 *전체 Operator 제거가 표시됩니다.* 제거 프로세스를 완료하는 데 최대 **5**분이 걸릴 수 있습니다.

- 사용자 정의 리소스 제거는 **MultiClusterEngine** 인스턴스의 사용자 정의 리소스를 제거하는 가장 기본적인 제거 유형이지만 다른 필수 **Operator** 리소스를 남겨 둡니다. 이 설치 제거 수준은 동일한 설정 및 구성 요소를 사용하여 다시 설치하려는 경우 유용합니다.

- 두 번째 수준은 사용자 정의 리소스 정의와 같은 구성 요소를 제외하고 대부분의 **Operator** 구성 요소를 제거하는 더 완전한 설치 제거입니다. 이 단계를 계속 수행하면 사용자 정의 리소스 제거로 제거되지 않은 모든 구성 요소와 서브스크립션이 제거됩니다. 이 제거 후 사용자 정의 리소스를 다시 설치하기 전에 **Operator**를 다시 설치해야 합니다.

12.1. 사전 요구 사항: DETACH ENABLED SERVICES

Kubernetes 엔진의 다중 클러스터 엔진을 설치 제거하려면 먼저 해당 엔진에서 관리하는 모든 클러스터를 분리해야 합니다. 오류를 방지하려면 엔진에서 여전히 관리하는 모든 클러스터를 분리한 다음 제거를 다시 시도합니다.

- 관리 클러스터가 연결된 경우 다음 메시지가 표시될 수 있습니다.

```
Cannot delete MultiClusterEngine resource because ManagedCluster resource(s) exist
```

클러스터 분리에 대한 자세한 내용은 *클러스터 생성에서 공급자에 대한 정보를 선택하여 관리에서 클러스터 제거 섹션을* 참조하십시오.

12.2. 명령을 사용하여 리소스 제거

1. 아직 없는 경우 **OpenShift Container Platform CLI**가 **oc** 명령을 실행하도록 구성되어 있는지 확인합니다. **oc** 명령을 구성하는 방법에 대한 자세한 내용은 **OpenShift Container Platform** 설명서에서 **OpenShift CLI** 시작하기 를 참조하십시오.

2. 다음 명령을 입력하여 프로젝트 네임스페이스로 변경합니다. ***namespace*** 를 프로젝트 네임스페이스의 이름으로 변경합니다.

```
oc project <namespace>
```

3. 다음 명령을 입력하여 **MultiClusterEngine** 사용자 정의 리소스를 제거합니다.

```
oc delete multiclusterengine --all
```

다음 명령을 입력하여 진행 상황을 볼 수 있습니다.

```
oc get multiclusterengine -o yaml
```

4. 다음 명령을 입력하여 설치된 네임스페이스에서 멀티 클러스터 엔진 **ClusterServiceVersion**을 삭제합니다.

```
❯ oc get csv
NAME                 DISPLAY                          VERSION   REPLACES   PHASE
multicluster-engine.v2.0.0   multicluster engine for Kubernetes   2.0.0              Succeeded

❯ oc delete clusterserviceversion multicluster-engine.v2.0.0
❯ oc delete sub multicluster-engine
```

여기에 표시된 **CSV** 버전은 다를 수 있습니다.

12.3. 콘솔을 사용하여 구성 요소 삭제

RedHat OpenShift Container Platform 콘솔을 사용하여 제거할 때 **Operator**를 제거합니다. 콘솔을 사용하여 설치 제거하려면 다음 단계를 완료합니다.

1. **OpenShift Container Platform** 콘솔 탐색에서 **Operator > 설치된 Operator > Kubernetes**용 다중 클러스터 엔진을 선택합니다.

2. **MultiClusterEngine** 사용자 정의 리소스를 제거합니다.

   a. ***Multiclusterengine*** 탭을 선택합니다.

   b. **MultiClusterEngine** 사용자 정의 리소스의 *옵션* 메뉴를 선택합니다.

   c. **Delete MultiClusterEngine** 을 선택합니다.

3. 다음 섹션의 절차에 따라 정리 스크립트를 실행합니다.

팁: Kubernetes 버전에 대해 동일한 다중 클러스터 엔진을 다시 설치하려는 경우 이 절차의 나머지 단계를 건너뛰고 사용자 정의 리소스를 다시 설치할 수 있습니다.

4. 설치된 Operator로 이동합니다.

5. *옵션* 메뉴를 선택하고 **Uninstall operator** 를 선택하여 **Kubernetes_ operator**용 _ 다중 클러스터 엔진을 제거합니다.

12.4. 문제 해결 UNINSTALL

다중 클러스터 엔진 사용자 정의 리소스가 제거되지 않은 경우 정리 스크립트를 실행하여 남아 있는 잠재적인 아티팩트를 제거하십시오.

a. 다음 스크립트를 파일에 복사합니다.

```
#!/bin/bash
oc delete apiservice v1.admission.cluster.open-cluster-management.io
v1.admission.work.open-cluster-management.io
oc delete validatingwebhookconfiguration multiclusterengines.multicluster.openshift.io
oc delete mce --all
```

13장. 문제를 해결하기 위해 MUST-GATHER 명령 실행

문제 해결을 시작하려면 사용자가 **must-gather** 명령을 실행하여 문제를 디버깅할 수 있는 문제 해결 시나리오에 대해 알아보고 명령을 사용하여 시작하는 절차를 참조하십시오.

필수 액세스 권한: 클러스터 관리자

13.1. MUST-GATHER 시나리오

- 시나리오 1: *문제 해결 방법에 대한 자세한 내용은 문서화된 문제 해결* 섹션을 참조하십시오. 이 가이드는 제품의 주요 기능에 따라 구성됩니다.

  이 시나리오에서는 가이드에서 해결 방법이 문서에 있는지 확인합니다.

- 시나리오 2: 해결 단계에 문제가 문서화되지 않은 경우 **must-gather** 명령을 실행하고 출력을 사용하여 문제를 디버깅합니다.

- 시나리오 3: **must-gather** 명령의 출력을 사용하여 문제를 디버깅할 수 없는 경우 **Red Hat** 지원과 출력을 공유하십시오.

13.2. MUST-GATHER 절차

must-gather 명령 사용을 시작하려면 다음 절차를 참조하십시오.

1. **must-gather** 명령에 대해 알아보고 **RedHat OpenShift Container Platform** 설명서에서 클러스터에 대한 데이터를 수집하는 데 필요한 사전 요구 사항을 설치합니다.

2. 클러스터에 로그인합니다. 일반적인 사용 사례의 경우 *엔진* 클러스터에 로그인하는 동안 **must-gather** 를 실행해야 합니다.

   참고: 관리 클러스터를 확인하려면 **cluster-scoped-resources** 디렉터리에 있는 **gather-managed.log** 파일을 찾습니다.

   ```
   <your-directory>/cluster-scoped-resources/gather-managed.log>
   ```

iPXEED 및 **AVAILABLE** 열에 **True** 가 설정되지 않은 관리형 클러스터가 있는지 확인합니다. **True** 상태로 연결되지 않은 클러스터에서 **must-gather** 명령을 실행할 수 있습니다.

3. 데이터 및 디렉터리를 수집하는 데 사용되는 **Kubernetes** 이미지의 다중 클러스터 엔진을 추가합니다. 출력에 대한 이미지와 디렉터리를 삽입한 다음 명령을 실행합니다.

```
oc adm must-gather --image=registry.redhat.io/multicluster-engine/must-gather-rhel8:v2.0.0 --dest-dir=<directory>
```

4. 지정된 디렉터리로 이동하여 다음 수준으로 구성된 출력을 확인합니다.

- 두 개의 피어 수준: **cluster-scoped-resources** 및 **namespace** 리소스입니다.
- 각 하위 수준: 클러스터 범위 및 네임스페이스 범위 리소스의 사용자 정의 리소스 정의의 **API** 그룹입니다.
- 각각에 대한 다음 수준: **YAML** 파일은 종류 별로 정렬됩니다.

13.3. 연결이 끊긴 환경의 MUST-GATHER

연결이 끊긴 환경에서 **must-gather** 명령을 실행하려면 다음 단계를 완료합니다.

1. 연결이 끊긴 환경에서 **Red Hat Operator** 카탈로그 이미지를 미러 레지스트리에 미러링합니다. 자세한 내용은 연결이 끊긴 네트워크에 설치를 참조하십시오.

2. 다음 명령을 실행하여 미러 레지스트리에서 이미지를 참조하는 로그를 추출합니다.

```
REGISTRY=registry.example.com:5000
IMAGE=$REGISTRY/multicluster-engine/must-gather-rhel8@sha256:ff9f37eb400dc1f7d07a9b6f2da9064992934b69847d17f59e385783c071b9d8

oc adm must-gather --image=$IMAGE --dest-dir=./data
```

여기에서 제품 팀의 버그를 열 수 있습니다.