OpenShift Container Platform 4.7

클러스터 업데이트

OpenShift Container Platform 클러스터 업데이트

Last Updated: 2022-10-02

OpenShift Container Platform 4.7 클러스터 업데이트

OpenShift Container Platform 클러스터 업데이트

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 문서는 OpenShift Container Platform 클러스터 업데이트 또는 업그레이드에 대한 정보를 제공합니다. 클러스터 업데이트는 클러스터를 오프라인으로 전환할 필요없이 간단한 프로세스로 실행할 수 있습니다.

차례

1장. OPENSHIFT 업데이트 서비스 이해

인터넷에 액세스할 수 있는 클러스터의 경우 Red Hat은 공개 API 뒤에서 호스팅된 서비스로 실행되는 OpenShift Container Platform 업데이트 서비스를 통해 업데이트를 제공합니다.

참고

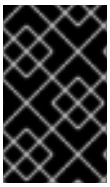
제한된 네트워크를 사용하고 클러스터가 공용 API에 액세스할 수 없는 경우 OpenShift 업데이트 서비스를 로컬로 설치할 수 있습니다. OpenShift 업데이트 서비스 설치 및 구성 을 참조하십시오.

1.1. OPENSHIFT 업데이트 서비스 정보

OSUS(OpenShift Update Service)는 Red Hat Enterprise Linux CoreOS(RHCOS)를 비롯한 OpenShift Container Platform에 대한 무선 업데이트를 제공합니다. 구성 요소 Operator의 *정점*과 이를 연결하는 *에지*를 포함하는 그래프 또는 다이어그램을 제공합니다. 그래프의 에지에는 안전하게 업데이트할 수 있는 버전이 표시됩니다. 정점은 관리형 클러스터 구성 요소의 상태를 지정하는 업데이트 페이로드입니다.

클러스터의 CVO (Cluster Version Operator)는 OpenShift Update Service를 확인하여 현재 구성 요소 버전 및 그래프의 정보를 기반으로 유효한 업데이트 및 업데이트 경로를 확인합니다. 업데이트를 요청하면 CVO는 해당 업데이트에 릴리스 이미지를 사용하여 클러스터를 업데이트합니다. 릴리스 아티팩트는 Quay에서 컨테이너 이미지로 호스팅됩니다.

OpenShift Update Service가 호환 가능한 업데이트만 제공할 수 있도록 자동화를 지원하는 버전 확인 파이프 라인이 제공됩니다. 각 릴리스 아티팩트는 지원되는 클라우드 플랫폼 및 시스템 아키텍처 및 기타 구성 요소 패키지와의 호환성 여부를 확인합니다. 파이프 라인에서 적용 가능한 버전이 있음을 확인한 후 OpenShift Update Service는 해당 버전 업데이트를 사용할 수 있음을 알려줍니다.



중요

OpenShift Update Service는 현재 클러스터에 권장되는 모든 업데이트를 표시합니다. OpenShift Update Service에서 업그레이드 경로를 권장하지 않는 경우 업데이트 또는 대상 릴리스와 관련된 알려진 문제로 인해 발생할 수 있습니다.

연속 업데이트 모드에서는 두 개의 컨트롤러가 실행됩니다. 하나의 컨트롤러는 페이로드 매니페스트를 지속적으로 업데이트하여 매니페스트를 클러스터에 적용한 다음 Operator의 제어된 롤아웃 상태를 출력하여 사용 가능한지, 업그레이드했는지 또는 실패했는지의 여부를 나타냅니다. 두 번째 컨트롤러는 OpenShift Update Service를 폴링하여 업데이트를 사용할 수 있는지 확인합니다.

중요

최신 버전으로의 업그레이드만 지원됩니다. 클러스터를 이전 버전으로 되돌리거나 롤백을 수행하는 것은 지원되지 않습니다. 업데이트에 실패하면 Red Hat 지원에 문의하십시오.

업데이트 프로세스 중에 MCO (Machine Config Operator)는 새 구성을 클러스터 머신에 적용합니다. MCO는 머신 설정 풀의 **maxUnavailable** 필드에 지정된 노드 수를 제한하고 이를 사용할 수없는 것으로 표시합니다. 기본적으로 이 값은 **1**로 설정됩니다. MCO는 새 설정을 적용하여 컴퓨터를 다시 시작합니다.

RHEL (Red Hat Enterprise Linux) 머신을 작업자로 사용하는 경우 먼저 시스템에서 OpenShift API를 업데이트해야하기 때문에 MCO는 이 머신에서 kubelet을 업데이트하지 않습니다.

새 버전의 사양이 이전 kubelet에 적용되므로 RHEL 머신을 **Ready** 상태로 되돌릴 수 없습니다. 컴퓨터를 사용할 수 있을 때까지 업데이트를 완료할 수 없습니다. 그러나 사용 불가능한 최대 노드 수를 설정하면 사용할 수 없는 머신의 수가 이 값을 초과하지 않는 경우에도 정상적인 클러스터 작업을 계속할 수 있습니다.

OpenShift Update Service는 Operator 및 하나 이상의 애플리케이션 인스턴스로 구성됩니다.

1.2. 관리되지 않는 OPERATOR에 대한 지원 정책

Operator의 *관리 상태*는 Operator가 설계 의도에 따라 클러스터의 해당 구성 요소에 대한 리소스를 적극적으로 관리하고 있는지 여부를 판별합니다. *Unmanaged* 상태로 설정된 Operator는 구성 변경에 응답하지 않고 업데이트되지도 않습니다.

비프로덕션 클러스터 또는 디버깅 중에는 이 기능이 유용할 수 있지만, Unmanaged 상태의 Operator는 지원되지 않으며 개별 구성 요소의 구성 및 업그레이드를 클러스터 관리자가 전적으로 통제하게 됩니다.

다음과 같은 방법으로 Operator를 Unmanaged 상태로 설정할 수 있습니다.

- **개별 Operator 구성**
  개별 Operator는 구성에 **managementState** 매개변수가 있습니다. Operator에 따라 다양한 방식으로 이 매개변수에 액세스할 수 있습니다. 예를 들어, Red HAt OpenShift Logging Operator는 관리 대상인 사용자 정의 리소스(CR)를 수정하여 이를 수행하는 반면 Cluster Samples Operator는 클러스터 전체의 구성 리소스를 사용합니다.

  managementState 매개변수를 **Unmanaged**로 변경하면 Operator가 리소스를 적극적으로 관리하지 않으며 해당하는 구성 요소와 관련된 조치도 수행하지 않습니다. 클러스터가 손상되고 수동 복구가 필요할 가능성이 있으므로 이 관리 상태를 지원하지 않는 Operator도 있습니다.

> **주의**
>
> 개별 Operator를 **Unmanaged** 상태로 변경하면 특정 구성 요소 및 기능이 지원되지 않습니다. 지원을 계속하려면 보고된 문제를 **Managed** 상태에서 재현해야 합니다.

- **Cluster Version Operator(CVO) 재정의**
  spec.overrides 매개변수를 CVO 구성에 추가하여 관리자가 구성 요소에 대한 CVO 동작에 대한 재정의 목록을 제공할 수 있습니다. 구성 요소에 대해 **spec.overrides[].unmanaged** 매개변수를 **true**로 설정하면 클러스터 업그레이드가 차단되고 CVO 재정의가 설정된 후 관리자에게 경고합니다.

> Disabling ownership via cluster version overrides prevents upgrades. Please remove overrides before continuing.

> **주의**
>
> CVO 재정의를 설정하면 전체 클러스터가 지원되지 않는 상태가 됩니다. 지원을 계속하려면 재정의를 제거한 후 보고된 문제를 재현해야 합니다.

2장. 클러스터 업데이트 개요

웹 콘솔 또는 OpenShift CLI (oc)를 사용하여 단일 작업으로 OpenShift Container Platform 4 클러스터를 업데이트할 수 있습니다.

2.1. OPENSHIFT 업데이트 서비스 이해

OpenShift 업데이트 서비스 이해: 인터넷에 액세스할 수 있는 클러스터의 경우 Red Hat은 공개 API 뒤에서 호스팅된 서비스로 실행되는 OpenShift Container Platform 업데이트 서비스를 통해 업데이트를 제공합니다. 자세한 내용은 다음을 참조하십시오.

- OpenShift 업데이트 서비스 정보
- 관리되지 않는 Operator에 대한 지원 정책 이해

2.2. OPENSHIFT 업데이트 서비스 설치 및 구성

OpenShift 업데이트 서비스 설치 및 구성: 인터넷에 액세스할 수 있는 클러스터는 공용 API에 액세스할 수 있습니다. 제한된 네트워크의 클러스터는 업데이트 정보를 위해 공용 API에 액세스할 수 없습니다. 제한된 네트워크에서 유사한 업그레이드 환경을 제공하기 위해 OpenShift 업데이트 서비스를 로컬로 설치 및 구성하여 연결이 끊긴 환경에서 사용할 수 있습니다. 자세한 내용은 다음을 참조하십시오.

- OpenShift Update Service Operator 설치
- OpenShift Update Service 애플리케이션 생성
- OpenShift Update Service 애플리케이션 삭제
- OpenShift Update Service Operator 설치 제거

2.3. 업그레이드 채널 및 릴리스 이해

업그레이드 채널 및 릴리스: 업그레이드 채널을 사용하면 업그레이드 전략을 선택할 수 있습니다. 업그레이드 채널은 OpenShift Container Platform의 마이너 버전에 따라 다릅니다. 업그레이드 채널은 릴리스 선택만 제어하고 설치하는 클러스터의 버전에 영향을 미치지 않습니다. OpenShift Container Platform의 특정 버전에 대한 **openshift-install** 바이너리 파일은 항상 해당 마이너 버전을 설치합니다. 자세한 내용은 다음을 참조하십시오.

- 버전 경로 업그레이드
- 빠르고 안정적인 채널 사용 및 전략 이해
- 제한된 네트워크 클러스터 이해
- 채널 간 전환

2.4. 웹 콘솔을 사용하여 클러스터 업데이트

웹 콘솔을 사용하여 클러스터 업데이트: 웹 콘솔을 사용하여 OpenShift Container Platform 클러스터를 업데이트할 수 있습니다. 다음 단계에서는 마이너 버전의 클러스터를 업데이트합니다. 마이너 버전 간에 클러스터를 업데이트하는 데 동일한 지침을 사용할 수 있습니다.

- 카나리아 롤아웃 업데이트 수행

- 웹 콘솔을 사용하여 클러스터 업데이트
- 웹 콘솔을 사용하여 업데이트 서버 변경

2.5. CLI를 사용하여 클러스터 업데이트

CLI를 사용하여 클러스터 업데이트: OpenShift CLI(oc)를 사용하여 마이너 버전에서 OpenShift Container Platform 클러스터를 업데이트할 수 있습니다. 다음 단계에서는 마이너 버전의 클러스터를 업데이트합니다. 마이너 버전 간에 클러스터를 업데이트하는 데 동일한 지침을 사용할 수 있습니다.

- CLI를 사용하여 클러스터 업데이트
- CLI를 사용하여 업데이트 서버 변경

2.6. 카나리아 롤아웃 업데이트 수행

카나리아 롤아웃 업데이트 수행: 작업자 노드에 대한 업데이트 롤아웃을 제어하면 업데이트 프로세스에서 애플리케이션이 실패하는 경우에도 미션 크리티컬 애플리케이션을 전체 업데이트 중에 계속 사용할 수 있도록 할 수 있습니다. 조직의 요구에 따라 소수의 작업자 노드를 업데이트하고 일정 기간 동안 클러스터 및 워크로드 상태를 평가한 다음 나머지 노드를 업데이트할 수 있습니다. 이를 *카나리아* 업데이트라고 합니다. 또는 한 번에 전체 클러스터를 업데이트하는 데 대규모 유지 관리 기간을 사용할 수 없는 경우 호스트 재부팅이 필요한 작업자 노드 업데이트를 작은 유지 관리 기간 내에 배치해야 할 수도 있습니다. 다음 절차를 수행할 수 있습니다.

- 카나리아 롤아웃 업데이트를 수행하기 위해 머신 구성 풀 생성
- 머신 구성 풀 정지
- 클러스터 업데이트 수행
- 머신 구성 풀 일시 정지 해제
- 노드를 원래 시스템 구성 풀로 이동

2.7. RHEL 컴퓨팅 시스템을 포함하는 클러스터 업데이트

RHEL 컴퓨팅 머신이 포함된 클러스터 업데이트: 클러스터에 RHEL (Red Hat Enterprise Linux) 시스템이 포함된 경우 추가 단계를 수행하여 해당 시스템을 업데이트해야 합니다. 다음 절차를 수행할 수 있습니다.

- 선택 사항: RHEL 시스템에서 Ansible 작업을 수행하기 위한 후크 추가
- 클러스터에서 RHEL 컴퓨팅 시스템 업데이트

2.8. 네트워크가 제한된 환경에서 클러스터 업데이트

네트워크가 제한된 환경의 클러스터 업데이트: 미러 호스트가 인터넷과 클러스터에 모두 액세스할 수 없는 경우 이미지를 해당 환경에서 연결이 끊긴 파일 시스템으로 미러링한 다음 호스트 또는 이동식 미디어를 가져올 수 있습니다. 로컬 컨테이너 레지스트리와 클러스터가 레지스트리의 미러 호스트에 연결된 경우 릴리스 이미지를 로컬 레지스트리로 직접 푸시할 수 있습니다.

- 미러 호스트 준비
- 이미지를 미러링할 수 있는 인증 정보 설정
- OpenShift Container Platform 이미지 저장소 미러링

- 이미지 서명 config map 생성
- 네트워크가 제한된 환경의 클러스터 업데이트
- 이미지 레지스트리 저장소 미러링 설정
- 클러스터 노드 재부팅 빈도를 줄이기 위해 미러 이미지 카탈로그의 범위 확장

3장. OPENSHIFT 업데이트 서비스 설치 및 구성

인터넷에 액세스할 수 있는 클러스터의 경우 Red Hat은 공개 API 뒤에서 호스팅된 서비스로 실행되는 OpenShift Container Platform 업데이트 서비스를 통해 업데이트를 제공합니다. 그러나 제한된 네트워크의 클러스터는 업데이트된 정보를 얻기 위해 공용 API에 액세스할 수 없습니다.

제한된 네트워크에서 유사한 업데이트 환경을 제공하기 위해 연결이 끊긴 환경에서 사용할 수 있도록 OpenShift 업데이트 서비스를 로컬로 설치 및 구성할 수 있습니다.

다음 섹션에서는 연결이 끊긴 클러스터 및 기본 운영 체제에 대해 무선 업데이트를 제공하는 방법을 설명합니다.

3.1. 사전 요구 사항

- Operator 설치에 대한 자세한 내용은 네임스페이스에 Operator 설치를 참조하십시오.

3.1.1. OpenShift 업데이트 서비스의 보안 레지스트리에 대한 액세스 구성

릴리스 이미지가 보안 레지스트리에 포함된 경우 업데이트 서비스에 대한 다음 변경과 함께 이미지 레지스트리 액세스를 위한 추가 신뢰 저장소 구성 단계를 완료합니다.

OpenShift Update Service Operator는 레지스트리 CA 인증서에 구성 맵 키 이름 **updateservice-registry**가 필요합니다.

업데이트 서비스에 대한 이미지 레지스트리 CA 구성 맵의 예

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: my-registry-ca
data:
  updateservice-registry: | 1
    -----BEGIN CERTIFICATE-----
    ...
    -----END CERTIFICATE-----
  registry-with-port.example.com..5000: | 2
    -----BEGIN CERTIFICATE-----
    ...
    -----END CERTIFICATE-----
```

1. OpenShift Update Service Operator에는 레지스트리 CA 인증서에 구성 맵 키 이름 updateservice-registry가 필요합니다.
2. 레지스트리에 **registry-with-port.example.com:5000** 같은 포트가 있는 경우 **:**이 **..**로 교체되어야 합니다.

3.1.2. 글로벌 클러스터 풀 시크릿 업데이트

현재 풀 시크릿을 교체하거나 새 풀 시크릿을 추가하여 클러스터의 글로벌 풀 시크릿을 업데이트할 수 있습니다.

사용자가 설치 중에 사용한 레지스트리보다 이미지를 저장하기 위해 별도의 레지스트리를 사용하는 경우 절차가 필요합니다.

주의

클러스터 리소스는 새로운 풀 시크릿에 맞게 조정되어야 하므로 일시적으로 클러스터 사용을 제한할 수 있습니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

프로세스

1. 선택 사항: 기존 풀 시크릿에 새 풀 시크릿을 추가하려면 다음 단계를 완료합니다.
   a. 다음 명령을 입력하여 풀 시크릿을 다운로드합니다.

```
$ oc get secret/pull-secret -n openshift-config --template='{{index .data
".dockerconfigjson" | base64decode}}' ><pull_secret_location> 1
```

   1 풀 시크릿 파일에 경로를 제공합니다.

   b. 다음 명령을 입력하여 새 풀 시크릿을 추가합니다.

```
$ oc registry login --registry="<registry>" \ 1
--auth-basic="<username>:<password>" \ 2
--to=<pull_secret_location> 3
```

   1 새 레지스트리를 제공합니다. 동일한 레지스트리에 여러 리포지토리를 포함할 수 있습니다 (예: **--registry="<registry/my-namespace/my-repository>**;).

   2 새 레지스트리의 인증 정보를 제공합니다.

   3 풀 시크릿 파일에 경로를 제공합니다.

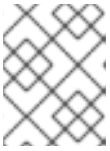
   또는 가져오기 시크릿 파일에 대한 수동 업데이트를 수행할 수 있습니다.

2. 다음 명령을 입력하여 클러스터의 글로벌 풀 시크릿을 업데이트합니다.

```
$ oc set data secret/pull-secret -n openshift-config --from-file=.dockerconfigjson=
<pull_secret_location> 1
```

   1 새 풀 시크릿 파일의 경로를 제공합니다.

   이 업데이트는 모든 노드로 롤아웃되며 클러스터 크기에 따라 작업에 약간의 시간이 걸릴 수 있습니다.

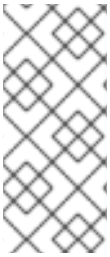


참고

OpenShift Container Platform 4.7.4부터 글로벌 풀 시크릿을 변경해도 더 이상 노드 드레이닝 또는 재부팅이 트리거되지 않습니다.

3.2. OPENSHIFT UPDATE SERVICE OPERATOR 설치

OpenShift Update Service를 설치하려면 먼저 OpenShift Container Platform 웹 콘솔 또는 CLI를 사용하여 OpenShift Update Service Operator를 설치해야 합니다.



참고

제한된 네트워크(연결이 끊기 클러스터)에 설치된 클러스터의 경우 Operator Lifecycle Manager는 기본적으로 원격 레지스트리에서 호스팅되는 Red Hat 제공 OperatorHub 소스에 액세스할 수 없습니다. 이러한 원격 소스에는 완전한 인터넷 연결이 필요하기 때문입니다. 자세한 내용은 제한된 네트워크에서 Operator Lifecycle Manager 사용 을 참조하십시오.

3.2.1. 웹 콘솔을 사용하여 OpenShift Update Service Operator 설치

웹 콘솔을 사용하여 OpenShift Update Service Operator를 설치할 수 있습니다.

프로세스

1. 웹 콘솔에서 **Operator → OperatorHub**를 클릭합니다.

참고

Update 서비스를 Filter by keyword...에 입력합니다. Operator를 더 빠르게 찾을 수 있는 필드입니다.

2. 사용 가능한 Operator 목록에서 **OpenShift Update Service**를 선택한 다음 **설치**를 클릭합니다.
   a. 채널 **v1**은 이 릴리스에서 사용할 수 있는 유일한 채널이므로 **업데이트 채널**로 선택됩니다.
   b. **설치 모드**에서 **클러스터의 특정 네임스페이스**를 선택합니다.
   c. **설치된 네임스페이스**의 네임스페이스를 선택하거나 권장 네임스페이스 **openshift-update-service**를 수락합니다.
   d. **승인 전략**을 선택합니다.
      - **자동** 전략을 사용하면 Operator 새 버전이 준비될 때 OLM(Operator Lifecycle Manager)이 자동으로 Operator를 업데이트할 수 있습니다.
      - **수동** 전략을 사용하려면 클러스터 관리자가 Operator 업데이트를 승인해야 합니다.
   e. **설치**를 클릭합니다.
3. **Operator → Installed Operator** 페이지로 전환하여 OpenShift Update Service Operator가 설치되었는지 확인합니다.
4. **OpenShift Update Service**가 선택한 네임스페이스에 **성공 상태**로 나열되어 있는지 확인합니다.

3.2.2. CLI를 사용하여 OpenShift Update Service Operator 설치

OpenShift CLI(**oc**)를 사용하여 OpenShift Update Service Operator를 설치할 수 있습니다.

프로세스

1. OpenShift OpenShift Update Service Operator의 네임스페이스를 생성합니다.
   a. OpenShift Update Service Operator에 대해 **Namespace** 오브젝트 YAML 파일 (예: **update-service-namespace.yaml**)을 만듭니다.

```
apiVersion: v1
kind: Namespace
metadata:
  name: openshift-update-service
  annotations:
    openshift.io/node-selector: ""
  labels:
    openshift.io/cluster-monitoring: "true" 1
```

   1 이 네임스페이스에서 Operator가 권장하는 클러스터 모니터링을 사용하도록 하려면 **openshift.io/cluster-monitoring** 레이블을 설정합니다.

   b. 네임스페이스를 생성합니다.

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

   예를 들면 다음과 같습니다.

```
$ oc create -f update-service-namespace.yaml
```

2. 다음 오브젝트를 생성하여 OpenShift Update Service Operator를 설치합니다.
   a. **OperatorGroup** 오브젝트 YAML 파일을 만듭니다 (예: **update-service-operator-group.yaml**).

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: update-service-operator-group
spec:
  targetNamespaces:
  - openshift-update-service
```

   b. **OperatorGroup** 오브젝트를 생성합니다.

```
$ oc -n openshift-update-service create -f <filename>.yaml
```

   예를 들면 다음과 같습니다.

```
$ oc -n openshift-update-service create -f update-service-operator-group.yaml
```

   c. **Subscription** 오브젝트 YAML 파일(예: **update-service-subscription.yaml**)을 생성합니다.

서브스크립션의 예

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: update-service-subscription
spec:
  channel: v1
  installPlanApproval: "Automatic"
  source: "redhat-operators" (1)
  sourceNamespace: "openshift-marketplace"
  name: "cincinnati-operator"
```

1. Operator를 제공하는 카탈로그 소스의 이름을 지정합니다. 사용자 정의 OLM(Operator Lifecycle Manager)을 사용하지 않는 클러스터의 경우 **redhat-operators**를 지정합니다. OpenShift Container Platform 클러스터가 제한된 네트워크(연결이 끊긴 클러스터)에 설치된 경우 OLM(Operator Lifecycle Manager)을 구성할 때 생성된 **CatalogSource** 오브젝트의 이름을 지정합니다.

d. **Subscription** 오브젝트를 생성합니다.

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

예를 들면 다음과 같습니다.

```
$ oc -n openshift-update-service create -f update-service-subscription.yaml
```

OpenShift Update Service Operator는 **openshift-update-service** 네임스페이스에 설치되고 **openshift-update-service** 네임스페이스를 대상으로 합니다.

3. Operator 설치를 확인합니다.

```
$ oc -n openshift-update-service get clusterserviceversions
```

출력 예

```
NAME                         DISPLAY                VERSION   REPLACES   PHASE
update-service-operator.v4.6.0   OpenShift Update Service   4.6.0             Succeeded
...
```

OpenShift Update Service Operator가 나열된 경우 설치에 성공한 것입니다. 버전 번호가 표시된 것과 다를 수 있습니다.

3.2.3. OpenShift Update Service 그래프 데이터 컨테이너 이미지 생성

OpenShift Update Service에는 그래프 데이터 컨테이너 이미지가 필요합니다. 이 이미지를 통해 OpenShift Update Service는 채널 멤버십에 및 차단된 업데이트 에지에 대한 정보를 검색합니다. 일반적으로 그래프 데이터는 업그레이드 그래프 데이터 리포지토리에서 직접 가져옵니다. 인터넷 연결이 불가능한 환경에서 init 컨테이너에서 이 정보를 로드하는 것도 OpenShift 업데이트 서비스에서 그래프 데이터를 사용할 수 있도록 하는 또 다른 방법입니다. init 컨테이너의 역할은 그래프 데이터의 로컬 사본을 제공하는 것이며 pod 초기화 중에 init 컨테이너가 서비스에서 액세스할 수 있는 볼륨에 데이터를 복사하는 것입니다.

프로세스

1. 다음을 포함하는 Dockerfile(예: **./Dockerfile**)을 생성합니다.

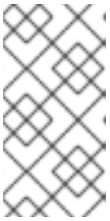
```
FROM registry.access.redhat.com/ubi8/ubi:8.1

RUN curl -L -o cincinnati-graph-data.tar.gz https://github.com/openshift/cincinnati-graph-data/archive/master.tar.gz

CMD exec /bin/bash -c "tar xvzf cincinnati-graph-data.tar.gz -C /var/lib/cincinnati/graph-data/ --strip-components=1"
```

2. 위 단계에서 생성된 Docker 파일을 사용하여 graph-data 컨테이너 이미지(예: **registry.example.com/openshift/graph-data:latest**)를 빌드합니다.

```
$ podman build -f ./Dockerfile -t registry.example.com/openshift/graph-data:latest
```

3. 이전 단계에서 만든 graph-data 컨테이너 이미지를 OpenShift Update Service에 액세스할 수 있는 리포지토리(예: **registry.example.com/openshift/graph-data:latest**)로 내보냅니다.

```
$ podman push registry.example.com/openshift/graph-data:latest
```

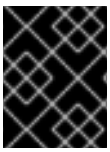


참고

제한된 네트워크의 로컬 레지스트리로 그래프 데이터 이미지를 내보내려면 이전 단계에서 생성한 graph-data 컨테이너 이미지를 OpenShift Update Service에 액세스할 수 있는 리포지토리로 복사합니다. 사용 가능한 옵션은 **oc image mirror --help** 를 실행합니다.

3.2.4. OpenShift Container Platform 이미지 저장소 미러링

OpenShift Update Service에는 업데이트 릴리스 페이로드가 포함된 로컬 액세스 레지스트리가 필요합니다.



중요

OpenShift Update Service 애플리케이션에서 과도한 메모리 사용을 방지하려면 다음 절차에 설명된 대로 릴리스 이미지를 별도의 저장소에 미러링하는 것이 좋습니다.

사전 요구 사항

- **OpenShift Container Platform 이미지 저장소 미러링** 섹션까지는 포함되지 않은 "연결이 끊긴 설치의 이미지 미러링"의 단계를 검토했습니다.
- 네트워크가 제한된 환경에서 사용할 미러 레지스트리를 설정하고 설정한 인증서 및 인증 정보에 액세스할 수 있습니다.
- Red Hat OpenShift Cluster Manager에서 풀 시크릿 을 다운로드하여 미러 저장소에 대한 인증을 포함하도록 수정했습니다.
- Subject Alternative Name을 설정하지 않는 자체 서명된 인증서를 사용하는 경우 이 절차의 **oc** 명령 앞에 **GODEBUG=x509ignoreCN=0**을 지정해야 합니다. 이 변수를 설정하지 않으면 **oc** 명령이 다음 오류로 인해 실패합니다.

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

프로세스

미러 호스트에서 다음 단계를 완료합니다.

1. OpenShift Container Platform 다운로드 페이지를 확인하여 업데이트할 OpenShift Container Platform 버전을 확인하고 Repository Tags 페이지에서 해당 태그를 확인합니다.
2. 필요한 환경 변수를 설정합니다.
   a. 릴리스 버전을 내보냅니다.

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

   <release_version>에 대해 설치할 OpenShift Container Platform 버전에 해당하는 태그를 지정합니다 (예: **4.6.4**).

   b. 로컬 레지스트리 이름 및 호스트 포트를 내보냅니다.

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

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

   c. 로컬 저장소 이름을 내보냅니다.

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

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

   d. 릴리스 이미지를 포함할 추가 로컬 리포지토리 이름을 내보냅니다.

   ```
   $
   LOCAL_RELEASE_IMAGES_REPOSITORY='<local_release_images_repository_name>'
   ```

   <local_release_images_repository_name>의 경우 레지스트리에 작성할 저장소 이름 (예: **ocp4/openshift4-release-images**)을 지정합니다.

   e. 미러링할 저장소 이름을 내보냅니다.

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

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

   f. 레지스트리 풀 시크릿의 경로를 내보냅니다.

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

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

   g. 릴리스 미러를 내보냅니다.

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

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

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

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

i. 미러링된 이미지를 호스트할 디렉터리의 경로를 내보냅니다.

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

1 초기 슬래시 (/) 문자를 포함하여 전체 경로를 지정합니다.

3. 미러 레지스트리에 버전 이미지를 미러링합니다.

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

i. 이동식 미디어를 인터넷에 연결된 시스템에 연결합니다.

ii. 미러링할 이미지 및 설정 매니페스트를 확인합니다.

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

iii. 이동식 미디어의 디렉터리에 이미지를 미러링합니다.

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

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

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

1 **REMOVABLE_MEDIA_PATH** 의 경우 이동식 미디어를 마운트한 경로를 사용해야 합니다.

v. 릴리스 이미지를 별도의 저장소에 미러링합니다.

```
$ oc image mirror -a ${LOCAL_SECRET_JSON}
${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-
```

```
${ARCHITECTURE}
${LOCAL_REGISTRY}/${LOCAL_RELEASE_IMAGES_REPOSITORY}:${OCP_REL
EASE}-${ARCHITECTURE}
```

- 로컬 컨테이너 레지스트리가 미러 호스트에 연결된 경우 릴리스 이미지를 로컬 레지스트리에 직접 푸시합니다.

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

3.3. OPENSHIFT UPDATE SERVICE 애플리케이션 생성

OpenShift Container Platform 웹 콘솔 또는 CLI를 사용하여 OpenShift Update Service 애플리케이션을 생성할 수 있습니다.

3.3.1. 웹 콘솔을 사용하여 OpenShift Update Service 애플리케이션 생성

OpenShift Container Platform 웹 콘솔을 사용하여 OpenShift Update Service Operator를 사용하여 OpenShift Update Service 애플리케이션을 생성할 수 있습니다.

사전 요구 사항

- OpenShift Update Service Operator가 설치되었습니다.
- OpenShift Update Service graph-data 컨테이너 이미지가 생성되어 OpenShift Update Service에서 액세스할 수 있는 리포지토리로 푸시되었습니다.
- 현재 릴리스 및 업데이트 대상 릴리스는 로컬에 액세스 가능한 레지스트리로 미러링되었습니다.

프로세스

1. 웹 콘솔에서 **Operator → Installed Operator**를 클릭합니다.
2. 설치된 Operator 목록에서 **OpenShift Update Service**를 선택합니다.
3. **Update Service** 탭을 클릭합니다.
4. **Create UpdateService**를 클릭합니다.
5. **Name** 필드에 이름을 입력합니다. (예: **service**)
6. **Graph Data Image** 필드에 "OpenShift Update Service 그래프 데이터 컨테이너 이미지 생성"에 생성된 graph-data 컨테이너 이미지에 로컬 pullspec을 입력합니다(예: **registry.example.com/openshift/graph-data:latest**).
7. **Releases** 필드에서 "OpenShift Container Platform 이미지 리포지토리 미러링"의 릴리스 이미지를 포함하도록 생성된 로컬 레지스트리 및 리포지토리를 입력합니다(예: **registry.example.com/ocp4/openshift4-release-images**).
8. **Replicas** 필드에 **2**를 입력합니다.

9. **Create**를 클릭하여 OpenShift Update Service 애플리케이션을 생성합니다.
10. OpenShift Update Service 애플리케이션 확인
    - **Update Service** 탭의 **UpdateServices** 목록에서 방금 만든 업데이트 서비스 애플리케이션을 클릭합니다.
    - **Resources** 탭을 클릭합니다.
    - 각 애플리케이션 리소스의 상태가 **Created**인지 확인합니다.

3.3.2. CLI를 사용하여 OpenShift Update Service 애플리케이션 생성

OpenShift CLI(**oc**)를 사용하여 OpenShift Update Service 애플리케이션을 생성할 수 있습니다.

사전 요구 사항

- OpenShift Update Service Operator가 설치되었습니다.
- OpenShift Update Service graph-data 컨테이너 이미지가 생성되어 OpenShift Update Service에서 액세스할 수 있는 리포지토리로 푸시되었습니다.
- 현재 릴리스 및 업데이트 대상 릴리스는 로컬에 액세스 가능한 레지스트리로 미러링되었습니다.

프로세스

1. OpenShift Update Service 대상 네임스페이스를 구성합니다(예: **openshift-update-service**).

   ```
   $ NAMESPACE=openshift-update-service
   ```

   네임스페이스는 Operator 그룹의 **targetNamespaces** 값과 일치해야 합니다.

2. OpenShift Update Service 애플리케이션의 이름을 구성합니다(예: **service**).

   ```
   $ NAME=service
   ```

3. "OpenShift Container Platform 이미지 리포지토리 미러링"에 구성된 릴리스 이미지의 로컬 레지스트리 및 리포지토리를 구성합니다(예: **registry.example.com/ocp4/openshift4-release-images**).

   ```
   $ RELEASE_IMAGES=registry.example.com/ocp4/openshift4-release-images
   ```

4. graph-data 이미지의 로컬 pullspec 을 "OpenShift Update Service 그래프 데이터 컨테이너 이미지 생성"에서 생성된 graph-data 컨테이너 이미지로 설정합니다(예: **registry.example.com/openshift/graph-data:latest**).

   ```
   $ GRAPH_DATA_IMAGE=registry.example.com/openshift/graph-data:latest
   ```

5. OpenShift Update Service 애플리케이션 오브젝트를 생성합니다.

   ```
   $ oc -n "${NAMESPACE}" create -f - <<EOF
   apiVersion: updateservice.operator.openshift.io/v1
   kind: UpdateService
   metadata:
   ```

```
  name: ${NAME}
spec:
  replicas: 2
  releases: ${RELEASE_IMAGES}
  graphDataImage: ${GRAPH_DATA_IMAGE}
EOF
```

6. OpenShift Update Service 애플리케이션 확인

   a. 다음 명령을 사용하여 정책 엔진 경로를 가져옵니다.

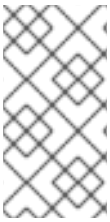
   ```
   $ while sleep 1; do POLICY_ENGINE_GRAPH_URI="$(oc -n "${NAMESPACE}" get -o
   jsonpath='{.status.policyEngineURI}/api/upgrades_info/v1/graph{"\n"}' updateservice
   "${NAME}")"; SCHEME="${POLICY_ENGINE_GRAPH_URI%%:*}"; if test "${SCHEME}"
   = http -o "${SCHEME}" = https; then break; fi; done
   ```

   명령이 성공할 때까지 폴링해야 할 수도 있습니다.

   b. 정책 엔진에서 그래프를 검색합니다. **channel**에 유효한 버전을 지정해야 합니다. 예를 들어 OpenShift Container Platform 4.7에서 실행하는 경우 **stable-4.7** 을 사용합니다.

   ```
   $ while sleep 10; do HTTP_CODE="$(curl --header Accept:application/json --output
   /dev/stderr --write-out "%{http_code}" "${POLICY_ENGINE_GRAPH_URI}?
   channel=stable-4.6")"; if test "${HTTP_CODE}" -eq 200; then break; fi; echo
   "${HTTP_CODE}"; done
   ```

   이 경우 그래프 요청이 성공할 때까지 폴링되지만 미러링된 릴리스 이미지에 따라 결과 그래프가 비어 있을 수 있습니다.



참고

정책 엔진 경로 이름은 RFC-1123을 기반으로 63자 이하여야 합니다. **host must conform to DNS 1123 naming convention and must be no more than 63 characters**로 인해 **CreateRouteFailed** 이유와 함께 **ReconcileCompleted** 상태가 **false**인 경우 더 짧은 이름으로 업데이트 서비스를 생성하십시오.

3.3.3. Cluster Version Operator (CVO) 구성

OpenShift Update Service Operator가 설치되고 OpenShift Update Service 애플리케이션이 생성된 후 로컬에 설치된 OpenShift Update Service에서 그래프 데이터를 가져오도록 CVO(Cluster Version Operator)를 업데이트할 수 있습니다.

사전 요구 사항

- OpenShift Update Service Operator가 설치되었습니다.
- OpenShift Update Service graph-data 컨테이너 이미지가 생성되어 OpenShift Update Service에서 액세스할 수 있는 리포지토리로 푸시되었습니다.
- 현재 릴리스 및 업데이트 대상 릴리스는 로컬에 액세스 가능한 레지스트리로 미러링되었습니다.
- OpenShift Update Service 애플리케이션이 생성되었습니다.

프로세스

1. OpenShift Update Service 대상 네임스페이스를 설정합니다(예: **openshift-update-service**).

```
$ NAMESPACE=openshift-update-service
```

2. OpenShift Update Service 애플리케이션의 이름을 설정합니다(예: **service**).

```
$ NAME=service
```

3. 정책 엔진 경로를 가져옵니다.

```
$ POLICY_ENGINE_GRAPH_URI="$(oc -n "${NAMESPACE}" get -o
jsonpath='{.status.policyEngineURI}/api/upgrades_info/v1/graph{"\n"}' updateservice
"${NAME}")"
```

4. 풀 그래프 데이터의 패치를 설정합니다.

```
$ PATCH="{\"spec\":{\"upstream\":\"${POLICY_ENGINE_GRAPH_URI}\"}}"
```

5. CVO를 패치하여 로컬 OpenShift Update Service를 사용합니다.

```
$ oc patch clusterversion version -p $PATCH --type merge
```

참고

업데이트 서버를 신뢰하도록 CA를 구성하려면 클러스터 전체의 프록시 활성화 를 참조하십시오.

3.4. OPENSHIFT UPDATE SERVICE 애플리케이션 삭제

OpenShift Container Platform 웹 콘솔 또는 CLI를 사용하여 OpenShift Update Service 애플리케이션을 삭제할 수 있습니다.

3.4.1. 웹 콘솔을 사용하여 OpenShift Update Service 애플리케이션 삭제

OpenShift Container Platform 웹 콘솔을 사용하여 OpenShift Update Service Operator로 OpenShift Update Service 애플리케이션을 삭제할 수 있습니다.

사전 요구 사항

- OpenShift Update Service Operator가 설치되었습니다.

프로세스

1. 웹 콘솔에서 **Operator → Installed Operator**를 클릭합니다.
2. 설치된 Operator 목록에서 **OpenShift Update Service**를 선택합니다.
3. **Update Service** 탭을 클릭합니다.
4. 설치된 OpenShift Update Service 애플리케이션 목록에서 삭제할 애플리케이션을 선택한 다음**Delete UpdateService**를 클릭합니다.

5. **Delete UpdateService?** 확인 프롬프트에서 **Delete**를 클릭하여 삭제를 확인합니다.

3.4.2. CLI를 사용하여 OpenShift Update Service 애플리케이션 삭제

OpenShift CLI(**oc**)를 사용하여 OpenShift Update Service 애플리케이션을 삭제할 수 있습니다.

프로세스

1. OpenShift Update Service 애플리케이션이 생성된 네임스페이스(예: **openshift-update-service**)를 사용하여 OpenShift Update Service 애플리케이션 이름을 가져옵니다.

```
$ oc get updateservice -n openshift-update-service
```

출력 예

```
NAME      AGE
service   6s
```

2. 이전 단계의 **NAME** 값과 OpenShift Update Service 애플리케이션이 생성된 네임스페이스(예: **openshift-update-service**)를 사용하여 OpenShift Update Service 애플리케이션을 삭제합니다.

```
$ oc delete updateservice service -n openshift-update-service
```

출력 예

```
updateservice.updateservice.operator.openshift.io "service" deleted
```

3.5. OPENSHIFT UPDATE SERVICE OPERATOR 설치 제거

OpenShift Update Service를 설치 제거하려면 먼저 OpenShift Container Platform 웹 콘솔 또는 CLI를 사용하여 모든 OpenShift Update Service 애플리케이션을 삭제해야 합니다.

3.5.1. 웹 콘솔을 사용하여 OpenShift Update Service Operator 설치 제거

OpenShift Container Platform 웹 콘솔을 사용하여 OpenShift Update Service Operator를 설치 제거할 수 있습니다.

사전 요구 사항

- 모든 OpenShift Update Service 애플리케이션이 삭제되어 있어야 합니다.

프로세스

1. 웹 콘솔에서 **Operator → Installed Operator**를 클릭합니다.
2. 설치된 Operator 목록에서 **OpenShift Update Service**을 선택하고 **Uninstall Operator**를 클릭합니다.
3. **Uninstall Operator?** 확인 대화 상자에서 **Uninstall**를 클릭하고 제거를 확인합니다.

3.5.2. CLI를 사용하여 OpenShift Update Service Operator 설치 제거

OpenShift CLI(**oc**)를 사용하여 OpenShift Update Service Operator를 제거할 수 있습니다.

사전 요구 사항

- 모든 OpenShift Update Service 애플리케이션이 삭제되어 있어야 합니다.

프로세스

1. OpenShift Update Service Operator가 포함된 프로젝트로 변경합니다(예: **openshift-update-service**).

```
$ oc project openshift-update-service
```

출력 예

```
Now using project "openshift-update-service" on server "https://example.com:6443".
```

2. OpenShift Update Service Operator Operator 그룹의 이름을 가져옵니다.

```
$ oc get operatorgroup
```

출력 예

```
NAME                              AGE
openshift-update-service-fprx2   4m41s
```

3. operator 그룹을 삭제합니다(예: **openshift-update-service-fprx2**).

```
$ oc delete operatorgroup openshift-update-service-fprx2
```

출력 예

```
operatorgroup.operators.coreos.com "openshift-update-service-fprx2" deleted
```

4. OpenShift Update Service Operator 서브스크립션의 이름을 가져옵니다.

```
$ oc get subscription
```

출력 예

```
NAME                      PACKAGE                   SOURCE                       CHANNEL
update-service-operator   update-service-operator   updateservice-index-catalog   v1
```

5. 이전 단계의 **Name** 값을 사용하여 **currentCSV** 필드에서 구독한 OpenShift Update Service Operator의 현재 버전을 확인합니다.

```
$ oc get subscription update-service-operator -o yaml | grep " currentCSV"
```

출력 예

```
  currentCSV: update-service-operator.v0.0.1
```

6. 서브스크립션을 삭제합니다(예: **update-service-operator**).

```
$ oc delete subscription update-service-operator
```

출력 예

```
subscription.operators.coreos.com "update-service-operator" deleted
```

7. 이전 단계의 **currentCSV** 값을 사용하여 OpenShift Update Service Operator의 CSV를 삭제합니다.

```
$ oc delete clusterserviceversion update-service-operator.v0.0.1
```

출력 예

```
clusterserviceversion.operators.coreos.com "update-service-operator.v0.0.1" deleted
```

4장. 업그레이드 채널 및 릴리스 이해

OpenShift Container Platform 4.1에서 Red Hat은 클러스터 업데이트에 적절한 릴리스 버전을 권장하기 위해 채널 개념을 도입했습니다. 업데이트 속도를 제어하면 이러한 업그레이드 채널을 통해 업데이트 전략을 선택할 수 있습니다. 업그레이드 채널은 OpenShift Container Platform의 마이너 버전과 연결되어 있습니다. 예를 들어 OpenShift Container Platform 4.7 업그레이드 채널에서는 4.7에 대한 업데이트 및 4.7 내에서 업데이트를 권장합니다. 또한 4.6의 클러스터가 4.7으로 업데이트될 수 있도록 4.6 및 4.6에서 4.7로 업데이트하는 것이 좋습니다. 4.8 이상의 릴리스에 대한 업데이트는 권장하지 않습니다. 이 전략을 사용하면 관리자가 OpenShift Container Platform의 다음 마이너 버전으로 명시적으로 업데이트하기로 결정할 수 있습니다.

업그레이드 채널은 릴리스 선택만 제어하며 설치한 클러스터 버전에는 영향을 미치지 않습니다. OpenShift Container Platform 특정 버전의 **openshift-install** 바이너리 파일은 항상 해당 버전을 설치합니다.

OpenShift Container Platform 4.7은 다음과 같은 업그레이드 채널을 제공합니다.

- **candidate-4.7**
- **fast-4.7**
- **stable-4.7**
- **EUS-4.y** (4.6과 같이 번호가 매겨진 4.y 클러스터 릴리스를 실행하는 경우에만)

주의

Red Hat은 Openshift Update Service에서 제안하는 버전으로의 업그레이드를 권장합니다. 마이너 버전 업데이트의 경우 버전이 연속되어야 합니다. Red Hat은 비지속 버전에 대한 업데이트를 테스트하지 않으며 이전 버전과의 호환성을 보장할 수 없습니다.

4.1. 업그레이드 채널 및 릴리스 경로

클러스터 관리자는 웹 콘솔에서 업그레이드 채널을 구성할 수 있습니다.

4.1.1. candidate-4.7 채널

candidate-4.7 채널에 는 z-stream (4.7.z) 및 이전 마이너 버전 릴리스에 대한 후보 빌드가 포함되어 있습니다. 릴리스 후보 버전에는 제품의 모든 기능이 포함되어 있지만 지원되지는 않습니다. 릴리스 후보 버전을 사용하여 새 버전의 기능을 테스트하고 다음 OpenShift Container Platform 버전이 시스템에 적합한지 확인할 수 있습니다. 릴리스 후보는 이름에 **-rc**와 같은 사전 릴리스 버전이 포함되어 있지 않은 후보 채널에서 사용 가능한 빌드를 말합니다. 후보 채널에서 버전을 사용할 수 있게 되면 더 많은 품질 테스트가 수행됩니다. 품질 기준을 충족하는 경우 **fast-4.7** 또는 **stable-4.7** 채널로 확장됩니다. 이로 인해 특정 릴리스가 **candidate-4.7** 채널과 **fast-4.7** 또는 **stable-4.7**채널 모두에서 사용 가능한 경우 이는 Red Hat에서 지원되는 버전임을 의미합니다. **candidate-4.7** 채널에는 채널에 권장되는 업데이트가없는 릴리스 버전이 포함되어 있을 수 있습니다.

candidate-4.7 채널을 사용하여 OpenShift Container Platform의 이전 마이너 버전에서 업데이트할 수 있습니다.

4.1.2. fast-4.7 채널

Red Hat이 특정 버전이 공식적으로 릴리스된다고 선언하면 **fast-4.7** 채널이 4.7의 신규 및 이전 마이너 버전으로 업데이트됩니다. 이와 같이 이 릴리스는 완전히 지원되고, 프로덕션 환경에 적합한 품질이며, **candidate-4.7** 채널에서 릴리스 후보로 사용 가능한 동안 문제 없이 제대로 작동하는 것으로 표시됩니다. **fast-4.7**채널에 릴리스가 표시된 후 **stable-4.7** 채널에 추가됩니다. 릴리스는 **fast-4.7** 채널에 표시되기 전에 **stable-4.7** 채널에 표시되지 않습니다.

fast-4.7 채널을 사용하여 OpenShift Container Platform의 이전 마이너 버전에서 업데이트할 수 있습니다.

4.1.3. stable-4.7 채널

에라타가 출시되면 곧 **fast-4.7** 채널에 표시되지만 릴리스는 지연 후 **stable-4.7** 채널에 추가됩니다. 이 지연 시간 동안 Connected Customer Program에 참여하는 Red Hat SRE 팀, Red Hat 지원 서비스, 사전 프로덕션 및 프로덕션 환경에서 릴리스의 안정성에 대한 데이터가 수집됩니다. **stable-4.7** 채널을 사용하여 OpenShift Container Platform의 이전 마이너 버전에서 업데이트할 수 있습니다.

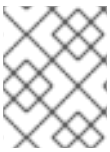
4.1.4. EUS-4.y 채널

stable 채널 외에도 모든 OpenShift Container Platform 마이너 버전은 EUS (Extended Update Support) 를 제공합니다. 이러한 EUS 버전은 표준 및 프리미엄 서브스크립션을 보유한 고객의 완전 및 유지 관리 지원 단계를 18 개월로 확장합니다.

OpenShift Container Platform 4. **y가 EUS 단계로 전환될 때까지 stable -4. y 및 eus-4.** y 채널에는 차이가 없지만 eus-4.y 채널이 사용 가능하게 되는 즉시 **eus-4.y** 채널로 전환할 수 있습니다.

다음 EUS 채널 업데이트가 제공되면 다음 EUS 버전에 도달할 때까지 다음 EUS 채널로 전환할 수 있습니다.

이 업데이트 프로세스는 **eus-4.6** 채널에는 적용되지 않습니다.



참고

표준 및 비 EUS 가입자 모두 모든 EUS 리포지토리 및 필요한 RPM (**rhel-*-eus-rpms**)에 액세스하여 드라이버 디버깅 및 빌드와 같은 중요한 목적을 지원할 수 있습니다.

4.1.5. 업그레이드 버전 경로

OpenShift Container Platform은 설치된 OpenShift Container Platform 버전과 다음 릴리스로 액세스하기 위해 선택한 채널의 경로를 확인할 수 있는 업그레이드 권장 서비스를 제공합니다.

fast-4.7 채널에서는 다음을 확인할 수 있습니다.

- 4.7.0
- 4.7.1
- 4.7.3
- 4.7.4

이 서비스는 테스트되었으며 심각한 문제가 없는 업데이트만 권장합니다. 알려진 취약점이 포함된 OpenShift Container Platform 버전으로 업데이트할 것을 권장하지 않습니다. 예를 들어 클러스터가 4.7.1에 있고 OpenShift Container Platform에서 4.7.4를 권장하는 경우 .4.7.1에서 .4.7.4로 안전하게 업데이트

할 수 있습니다. 연속적인 패치 번호에 의존하지 않도록하십시오. 이 예에서 4.7.2는 채널에서 사용 불가능합니다.

업데이트의 안정성은 채널에 따라 다릅니다. **candidate-4.7** 채널에 업데이트 권장 사항이 있다고해서 해당 업데이트가 지원되는 것은 아닙니다. 업데이트와 관련하여 아직 심각한 문제가 발견되지 않았지만 업데이트를 통한 트래픽이 많지 않을 경우 안정성이 확인되지 않을 수 있습니다. **fast-4.7 또는 stable-4.7 채널에** 업데이트 권장 사항이 있다는 것은 업데이트가 지원됨을 나타냅니다. 릴리스는 채널에서 제거되지 않지만 심각한 문제가있는 업데이트 권장 사항은 모든 채널에서 제거됩니다. 업데이트 권장 사항이 제거된 후에도 시작된 업데이트는 계속 지원됩니다.

Red Hat은 **fast-4.7** 또는 **stable-4.7** 채널에서 지원되는 모든 릴리스에서 4.7.z의 최신 릴리스로 지원되는 업데이트 경로를 제공합니다. 그러나 문제가 발생한 릴리스로부터 안전한 경로를 구축하고 확인하는 동안 지연이 발생할 수 있습니다.

4.1.6. 빠르고 안정적인 채널 사용 및 전략

fast-4.7 및 **stable-4.7** 채널에서는 공식 버전이 릴리스되는 대로 이를 즉시 수신하거나 Red Hat이 해당 업데이트의 롤아웃을 제어하는 것을 허용할 지 여부를 선택할 수 있습니다. 롤아웃 중 또는 이후에 문제가 발견되면 해당 버전에 대한 업데이트가 **fast-4.7** 및 **stable-4.7** 채널에서 차단될 수 있으며 새로 권장되는 업데이트 대상의 새 버전이 도입될 수 있습니다.

고객은 **fast-4.7** 채널에서 사전 프로덕션 시스템을 설정하고, **stable-4.7** 채널에서 프로덕션 시스템을 설정한 뒤, Red Hat연결 고객 프로그램에 참여하여 고객의 프로세스를 개선할 수 있습니다. Red Hat은 이 프로그램을 사용하여 사용자의 특정 하드웨어 및 소프트웨어 설정에 대한 업데이트의 영향을 관찰합니다. 향후 릴리스에서는 업데이트가 **fast-4.7**에서 **stable-4.7** 채널로 이동하는 속도가 향상되거나 변경될 수 있습니다.

4.1.7. 네트워크가 제한된 환경의 클러스터

OpenShift Container Platform 클러스터의 컨테이너 이미지를 직접 관리하는 경우 제품 릴리스와 관련된 Red Hat 에라타를 참조하고 업데이트에 영향을 미치는 영향을 고려해야 합니다. 업데이트 중에 사용자 인터페이스에서 이러한 버전 간 전환에 대해 경고할 수 있으므로 이러한 경고를 무시하기 전에 적절한 버전을 선택했는지 확인해야 합니다.

4.1.8. 채널 간 전환

채널은 웹 콘솔에서 또는 **패치** 명령을 통해 전환할 수 있습니다.

```
$ oc patch clusterversion version --type json -p '[{"op": "add", "path": "/spec/channel", "value": "
<channel>"}]'
```

현재 릴리스를 포함하지 않는 채널로 전환하면 웹 콘솔에 경고가 표시됩니다. 웹 콘솔은 현재 릴리스가 없는 채널에서 업데이트를 권장하지 않습니다. 하지만 언제든지 원래 채널로 돌아갈 수 있습니다.

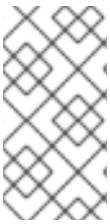
채널을 변경하면 클러스터의 지원 가능성에 영향을 미칠 수 있습니다. 다음과 같은 조건이 적용될 수 있습니다.

- **stable-4.7** 채널에서 **fast-4.7** 채널로 변경해도 클러스터는 계속 지원됩니다.
- **candidate-4.7 채널로** 전환할 수 있지만 이 채널의 일부 릴리스는 지원되지 않을 수 있습니다.
- 현재 릴리스가 정식 사용 버전 릴리스인 경우**candidate-4.7** 채널에서 **fast-4.7** 채널로 전환할 수 있습니다.

- 항상 **fast-4.7 채널에서 stable-4.7 채널로** 전환할 수 있습니다 **.** 현재 릴리스가 최근에 승격된 경우 **릴리스가 stable-4.7** 로 승격될 때까지 최대 하루가 지연될 수 있습니다.

5장. 웹 콘솔을 사용하여 클러스터 업데이트

웹 콘솔을 사용하여 OpenShift Container Platform 클러스터를 업데이트하거나 업그레이드할 수 있습니다. 다음 단계에서는 마이너 버전의 클러스터를 업데이트합니다. 마이너 버전 간에 클러스터를 업데이트하는 데 동일한 지침을 사용할 수 있습니다.



참고

oc를 사용하여 업데이트 채널을 변경하기가 어렵기 때문에 웹 콘솔을 사용하여 업데이트 채널을 변경합니다. 웹 콘솔 내에서 업데이트 프로세스를 완료하는 것이 좋습니다. 4.7 채널을 변경한 후 업데이트를 완료하기 위해 CLI를 사용하여 클러스터 업데이트 단계를 실행할 수 있습니다.

5.1. 전제 조건

- **admin** 권한이 있는 사용자로 클러스터에 액세스합니다. RBAC를 사용하여 권한 정의 및 적용 을 참조하십시오.
- 업데이트가 실패하는 경우 최근 etcd 백업이 있고 클러스터를 이전 상태로 복원해야 합니다.
- OLM(Operator Lifecycle Manager)을 통해 이전에 설치된 모든 Operator가 최신 채널의 최신 버전으로 업데이트되었는지 확인합니다. Operator를 업데이트하면 클러스터 업데이트 중에 기본 OperatorHub 카탈로그가 현재 마이너 버전에서 다음 버전으로 전환할 때 유효한 업그레이드 경로를 사용할 수 있습니다. 자세한 정보는 설치된 Operator 업그레이드를 참조하십시오.
- 모든 MCP(Machine config pool)가 실행 중이고 일시 중지되지 않는지 확인합니다. 업데이트 프로세스 중에 일시 중지된 MCP와 연결된 노드를 건너뜁니다. 카나리아 롤아웃 업데이트 전략을 수행하는 경우 MCP를 일시 중지할 수 있습니다.
- 클러스터에서 수동으로 유지 관리되는 인증 정보를 사용하는 경우 CCO (Cloud Credential Operator)가 업그레이드 가능한 상태인지 확인합니다. 자세한 내용은 AWS, Azure 또는 GCP에 대해 *수동으로 유지 관리되는 인증 정보를 사용하여 클러스터 업그레이드* 참조하십시오.

중요

- 업데이트가 완료되지 않으면 CVO(Cluster Version Operator)는 업데이트를 조정하는 동안 모든 차단 구성 요소의 상태를 보고합니다. 클러스터를 이전 버전으로 롤백하는 것은 지원되지 않습니다. 업데이트가 완료되지 않으면 Red Hat 지원팀에 문의하십시오.
- **unsupportedConfigOverrides** 섹션을 사용하여 Operator 설정을 변경하는 것은 지원되지 않으며 클러스터 업데이트를 차단할 수 있습니다. 클러스터를 업데이트하려면 먼저 이 설정을 제거해야 합니다.

추가 리소스

- 관리되지 않는 Operator에 대한 지원 정책

5.2. 카나리아 롤아웃 업데이트 수행

일부 특정 사용 사례에서는 클러스터의 나머지 부분과 동시에 특정 노드를 업데이트하지 않도록 보다 제어된 업데이트 프로세스를 원할 수 있습니다. 이러한 사용 사례에는 다음이 포함되지만 이에 국한되지는 않습니다.

- 업데이트 중에 사용할 수 없는 미션크리티컬 애플리케이션이 있습니다. 업데이트 후 노드의 애플리케이션을 소규모로 천천히 테스트할 수 있습니다.
- 유지 보수 기간이 짧아서 모든 노드를 업데이트할 수 없거나 유지 보수 기간이 여러 개일 수 있습니다.

롤링 업데이트 프로세스는 일반적인 업데이트 워크플로우가 **아닙니다**. 대규모 클러스터를 사용하면 여러 명령을 실행해야 하는 시간이 많이 소요될 수 있습니다. 이러한 복잡성으로 인해 전체 클러스터에 영향을 줄 수 있는 오류가 발생할 수 있습니다. 롤링 업데이트를 사용할지 여부를 신중하게 고려하고 시작하기 전에 프로세스 구현을 신중하게 계획하는 것이 좋습니다.

이 주제에서 설명하는 롤링 업데이트 프로세스에는 다음이 포함됩니다.

- 하나 이상의 사용자 지정 MCP(Machine config pool) 생성.
- 해당 노드를 사용자 지정 MCP로 이동하기 위해 즉시 업데이트하지 않으려는 각 노드에 레이블을 지정.
- 해당 노드에 대한 업데이트를 방지하는 사용자 지정 MCP를 일시 중지.
- 클러스터 업데이트 수행.
- 해당 노드에서 업데이트를 트리거하는 하나의 사용자 지정 MCP를 일시 중지 해제.
- 해당 노드에서 애플리케이션을 테스트하여 새로 업데이트된 해당 노드에서 애플리케이션이 예상대로 작동하는지 확인.
- 선택적으로 나머지 노드에서 사용자 지정 레이블을 소규모 배치로 제거하고 해당 노드에서 애플리케이션을 테스트.

참고

MCP를 일시 중지하면 Machine Config Operator에서 연결된 노드에 구성 변경 사항을 적용하지 못합니다. MCP를 일시 중지하면 자동으로 순환된 인증서가 **kube-apiserver-to-kubelet-signer** CA 인증서의 자동 CA 순환을 포함하여 관련 노드로 푸시되지 않습니다. **kube-apiserver-to-kubelet-signer** CA 인증서가 만료되고 MCO가 인증서를 자동으로 갱신하려고 하면 새 인증서가 생성되지만 해당 머신 구성 풀의 노드에 적용되지 않습니다. 이로 인해 **oc debug**, **oc logs**, **oc exec**, **oc attach**를 포함하여 여러 **oc** 명령이 실패합니다. MCP 일시 중지는 **kube-apiserver-to-kubelet-signer** CA 인증서 만료에 대해 신중하게 고려하여 단기간 동안만 수행해야 합니다.

카나리아 롤아웃 업데이트 프로세스를 사용하려면 카나리아 롤아웃 업데이트 수행 을 참조하십시오.

5.3. 웹 콘솔을 사용하여 클러스터 업데이트

사용 가능한 업데이트가 있으면 웹 콘솔에서 클러스터를 업데이트할 수 있습니다.

사용 가능한 OpenShift Container Platform 권고 및 업데이트는 고객 포털의 에라타 섹션을 참조하십시오.

전제 조건

- **admin** 권한이있는 사용자로 웹 콘솔에 액세스합니다.

절차

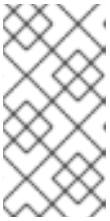
1. 웹 콘솔에서 **Administration → Cluster Settings**을 클릭하고 **Details** 탭의 내용을 확인합니다.
2. 프로덕션 클러스터의 경우 **Channel** 이 **stable-4.7** 등 업데이트하려는 버전에 대한 올바른 채널로 설정되어 있는지 확인합니다.

중요

프로덕션 클러스터의 경우 **stable-*** 또는 **fast-*** 채널에 가입해야합니다.

- **Update status** 가 **Updates available** 이 아닌 경우 클러스터를 업데이트할 수 없습니다.
- **Select channel**은 클러스터가 실행 중이거나 업데이트 중인 클러스터 버전을 나타냅니다.

3. 업데이트할 버전을 선택하고 **Save(저장)**를 클릭합니다.
입력 채널 **Update Status**가 **Update to <product-version> in progress**로 변경되고 Operator 및 노드의 진행률을 확인하여 클러스터 업데이트의 진행 상황을 검토할 수 있습니다.



참고

클러스터를 버전 4.y+1과 같이 다음 마이너 버전으로 업그레이드하는 경우 새 기능에 의존하는 워크로드를 배포하기 전에 노드가 업데이트되었는지 확인하는 것이 좋습니다. 아직 업데이트되지 않은 작업자 노드가 있는 풀은 **클러스터 설정** 페이지에 표시됩니다.

4. 업데이트가 완료되고 Cluster Version Operator가 사용 가능한 업데이트를 새로 고침한 후 현재 채널에서 사용 가능한 추가 업데이트가 있는지 확인합니다.
   - 업데이트가 있는 경우 더 이상 업데이트할 수 없을 때까지 현재 채널에서 업데이트를 계속 수행합니다.
   - 사용 가능한 업데이트가 없는 경우 **Channel** 을 다음 마이너 버전의 **stable-*** 또는 **fast-*** 채널로 변경하고 해당 채널에서 원하는 버전으로 업데이트합니다.

   필요한 버전에 도달할 때까지 여러 중간 업데이트를 수행해야 할 수도 있습니다.

5.4. 웹 콘솔을 사용하여 업데이트 서버 변경

업데이트 서버 변경은 선택 사항입니다. 로컬에 설치되어 구성된 OSUS(OpenShift Update Service)가 있는 경우 업데이트 중에 로컬 서버를 사용하도록 서버의 URL을 **upstream**으로 설정해야 합니다.

절차

1. **관리 → 클러스터 설정**으로 이동하여 **버전**을 클릭합니다.
2. **YAML** 탭을 클릭한 다음 **업스트림** 매개변수 값을 편집합니다.

출력 예

```
...
spec:
  clusterID: db93436d-7b05-42cc-b856-43e11ad2d31a
  upstream: '<update-server-url>' 1
...
```

1 **<update-server-url>** 변수는 업데이트 서버의 URL을 지정합니다.

기본 **upstream**은 **https://api.openshift.com/api/upgrades_info/v1/graph**입니다.

3. **저장**을 클릭합니다.

6장. CLI를 사용하여 클러스터 업데이트

OpenShift CLI (**oc**)를 사용하여 마이너 버전에서 OpenShift Container Platform 클러스터를 업데이트하거나 업그레이드할 수 있습니다. 동일한 지침에 따라 마이너 버전 간에 클러스터를 업데이트할 수도 있습니다.

6.1. 전제 조건

- **admin** 권한이 있는 사용자로 클러스터에 액세스합니다. RBAC를 사용하여 권한 정의 및 적용 을 참조하십시오.
- 업데이트가 실패하는 경우 최근 etcd 백업이 있고 클러스터를 이전 상태로 복원해야 합니다.
- OLM(Operator Lifecycle Manager)을 통해 이전에 설치된 모든 Operator가 최신 채널의 최신 버전으로 업데이트되었는지 확인합니다. Operator를 업데이트하면 클러스터 업데이트 중에 기본 OperatorHub 카탈로그가 현재 마이너 버전에서 다음 버전으로 전환할 때 유효한 업그레이드 경로를 사용할 수 있습니다. 자세한 정보는 설치된 Operator 업그레이드를 참조하십시오.
- 모든 MCP(Machine config pool)가 실행 중이고 일시 중지되지 않는지 확인합니다. 업데이트 프로세스 중에 일시 중지된 MCP와 연결된 노드를 건너뜁니다. 카나리아 롤아웃 업데이트 전략을 수행하는 경우 MCP를 일시 중지할 수 있습니다.
- 클러스터에서 수동으로 유지 관리되는 인증 정보를 사용하는 경우 CCO (Cloud Credential Operator)가 업그레이드 가능한 상태인지 확인합니다. 자세한 내용은 AWS, Azure 또는 GCP에 대해 *수동으로 유지 관리되는 인증 정보를 사용하여 클러스터 업그레이드* 참조하십시오.

중요

- 업데이트가 완료되지 않으면 CVO(Cluster Version Operator)는 업데이트를 조정하는 동안 모든 차단 구성 요소의 상태를 보고합니다. 클러스터를 이전 버전으로 롤백하는 것은 지원되지 않습니다. 업데이트가 완료되지 않으면 Red Hat 지원팀에 문의하십시오.
- **unsupportedConfigOverrides** 섹션을 사용하여 Operator 설정을 변경하는 것은 지원되지 않으며 클러스터 업데이트를 차단할 수 있습니다. 클러스터를 업데이트하려면 먼저 이 설정을 제거해야 합니다.

추가 리소스

- 관리되지 않는 Operator에 대한 지원 정책

6.2. CLI를 사용하여 클러스터 업데이트

업데이트가있는 경우 OpenShift CLI (**oc**)를 사용하여 클러스터를 업데이트할 수 있습니다.

사용 가능한 OpenShift Container Platform 권고 및 업데이트는 고객 포털의 에라타 섹션을 참조하십시오.

전제 조건

- 업데이트된 버전과 일치하는 OpenShift CLI (**oc**)를 설치합니다.
- **cluster-admin** 권한이 있는 사용자로 클러스터에 로그인합니다.

- **jq** 패키지를 설치합니다.

절차

1. 클러스터를 사용할 수 있는지 확인합니다.

```
$ oc get clusterversion
```

출력 예

```
NAME     VERSION   AVAILABLE   PROGRESSING   SINCE   STATUS
version   4.6.9     True        False         158m    Cluster version is 4.6.9
```

2. 현재 업데이트 채널 정보를 확인하고 채널이 **stable-4.7**로 설정되어 있는지 확인합니다.

```
$ oc get clusterversion -o json|jq ".items[0].spec"
```

출력 예

```
{
  "channel": "stable-4.7",
  "clusterID": "990f7ab8-109b-4c95-8480-2bd1deec55ff"
}
```

중요

프로덕션 클러스터의 경우 **stable-*** 또는 **fast-*** 채널에 가입해야합니다.

3. 사용 가능한 업데이트를 확인하고 적용하려는 업데이트의 버전 번호를 기록해 둡니다.

```
$ oc adm upgrade
```

출력 예

```
Cluster version is 4.1.0

Updates:

VERSION IMAGE
4.1.2   quay.io/openshift-release-dev/ocp-
release@sha256:9c5f0df8b192a0d7b46cd5f6a4da2289c155fd5302dec7954f8f06c878160b8b
```

4. 업데이트를 적용합니다.

- 최신 버전으로 업데이트하려면 다음을 수행합니다.

```
$ oc adm upgrade --to-latest=true 1
```

- 특정 버전으로 업데이트하려면 다음을 수행합니다.

```
$ oc adm upgrade --to=<version> 1
```

1 1 **<version>**은 이전 명령의 출력에서 얻을 수 있는 업데이트 버전입니다.

5. 클러스터 버전 Operator의 상태를 확인합니다.

```
$ oc get clusterversion -o json|jq ".items[0].spec"
```

출력 예

```
{
  "channel": "stable-4.7",
  "clusterID": "990f7ab8-109b-4c95-8480-2bd1deec55ff",
  "desiredUpdate": {
    "force": false,
    "image": "quay.io/openshift-release-dev/ocp-
release@sha256:9c5f0df8b192a0d7b46cd5f6a4da2289c155fd5302dec7954f8f06c878160b8b",

    "version": "4.7.0" 1
  }
}
```

1 **desiredUpdate** 부분의 **version** 번호가 지정한 값과 일치하는 경우 업데이트가 진행 중입니다.

6. 클러스터 버전 상태 기록에서 업데이트 상태를 모니터링합니다. 모든 개체가 업데이트를 완료하는 데 시간이 걸릴 수 있습니다.

```
$ oc get clusterversion -o json|jq ".items[0].status.history"
```

출력 예

```
[
  {
    "completionTime": null,
    "image": "quay.io/openshift-release-dev/ocp-
release@sha256:b8fa13e09d869089fc5957c32b02b7d3792a0b6f36693432acc0409615ab23b
7",
    "startedTime": "2021-01-28T20:30:50Z",
    "state": "Partial",
    "verified": true,
    "version": "4.7.0"
  },
  {
    "completionTime": "2021-01-28T20:30:50Z",
    "image": "quay.io/openshift-release-dev/ocp-
release@sha256:b8fa13e09d869089fc5957c32b02b7d3792a0b6f36693432acc0409615ab23b
7",
    "startedTime": "2021-01-28T17:38:10Z",
    "state": "Completed",
    "verified": false,
    "version": "4.7.0"
  }
]
```

기록에는 클러스터에 적용된 최신 버전 목록이 포함되어 있습니다. 이 값은 CVO가 업데이트를 적용할 때 업데이트됩니다. 목록은 날짜순으로 정렬되며 최신 업데이트가 목록의 맨 처음에 표시됩니다. 롤아웃이 완료되면 기록의 업데이트 상태는 **Completed**로 표시되고 업데이트가 실패하거나 완료되지 않은 경우 **Partial**로표시됩니다.

7. 업데이트가 완료되면 클러스터 버전이 새 버전으로 업데이트되었는지 확인합니다.

```
$ oc get clusterversion
```

출력 예

```
NAME     VERSION     AVAILABLE   PROGRESSING   SINCE     STATUS
version   4.7.0       True        False         2m        Cluster version is 4.7.0
```

8. 클러스터를 버전 4.y에서 4.(y+1)로 업그레이드하는 경우 새 기능을 사용하는 워크로드를 배포하기 전에 노드가 업그레이드되었는지 확인하는 것이 좋습니다.

```
$ oc get nodes
```

출력 예

```
NAME                         STATUS   ROLES    AGE   VERSION
ip-10-0-168-251.ec2.internal   Ready    master   82m   v1.20.0
ip-10-0-170-223.ec2.internal   Ready    master   82m   v1.20.0
ip-10-0-179-95.ec2.internal    Ready    worker   70m   v1.20.0
ip-10-0-182-134.ec2.internal   Ready    worker   70m   v1.20.0
ip-10-0-211-16.ec2.internal    Ready    master   82m   v1.20.0
ip-10-0-250-100.ec2.internal   Ready    worker   69m   v1.20.0
```

6.3. CLI를 사용하여 업데이트 서버 변경

업데이트 서버 변경은 선택 사항입니다. 로컬에 설치되어 구성된 OSUS(OpenShift Update Service)가 있는 경우 업데이트 중에 로컬 서버를 사용하도록 서버의 URL을 **upstream**으로 설정해야 합니다. **upstream**의 기본값은 **https://api.openshift.com/api/upgrades_info/v1/graph**입니다.

프로세스

- 클러스터 버전에서 **upstream** 매개변수 값을 변경합니다.

```
$ oc patch clusterversion/version --patch '{"spec":{"upstream":"<update-server-url>"}}' --
type=merge
```

<update-server-url> 변수는 업데이트 서버의 URL을 지정합니다.

출력 예

```
clusterversion.config.openshift.io/version patched
```

7장. 카나리아 롤아웃 업데이트 수행

업데이트 프로세스로 인해 애플리케이션이 실패하더라도 전체 업데이트 중에 미션크리티컬 애플리케이션을 계속 사용할 수 있도록 작업자 노드에 대한 업데이트 롤아웃을 보다 제어해야 하는 몇 가지 시나리오가 있을 수 있습니다. 조직의 요구에 따라 작업자 노드의 작은 하위 집합을 업데이트하고 일정 기간 동안 클러스터 및 워크로드 상태를 평가한 다음 나머지 노드를 업데이트할 수 있습니다. 이를 *카나리아* 업데이트라고 합니다. 또는 호스트 재부팅이 필요한 작업자 노드 업데이트를 한 번에 전체 클러스터를 업데이트할 수 없는 경우 정의된 더 작은 유지 관리 기간으로 전환해야 할 수도 있습니다.

이러한 시나리오에서는 클러스터를 업데이트할 때 특정 작업자 노드가 업데이트되지 않도록 여러 MCP(사용자 정의 머신 구성 풀)를 생성할 수 있습니다. 나머지 클러스터가 업데이트되면 적절한 시간에 배치로 해당 작업자 노드를 업데이트할 수 있습니다.

예를 들어 초과 용량이 10%인 노드가 100개 있는 클러스터가 있는 경우 유지 관리 기간이 4시간을 넘지 않아야 하며 작업자 노드를 드레이닝하고 재부팅하는 데 8분이 걸리기 때문에 MCP를 활용하여 목표를 달성할 수 있습니다. 예를 들어 각각 10, 30개, 30개의 노드가 있는 **workerpool-canary**, **workerpool-A**, **workerpool-B**, **workerpool-C**라는 4개의 MCP를 정의할 수 있습니다.

첫 번째 유지 관리 기간 동안 **workerpool-A**, **workerpool-B**, **workerpool-C**에 대한 MCP를 일시 중지한 다음 클러스터 업데이트를 시작합니다. 이 경우 해당 풀이 일시 중지되지 않았기 때문에 OpenShift Container Platform에서 실행되는 구성 요소와 **workerpool-canary** MCP의 멤버인 10개의 노드가 업데이트되었습니다. 나머지 3개의 MCP는 일시 중지되었으므로 업데이트되지 않습니다. 어떠한 이유로 **workerpool-canary** 업데이트의 클러스터 또는 워크로드 상태가 부정적인 영향을 미치는 경우 문제를 진단할 때까지 충분한 용량을 유지 관리하면서 해당 풀의 모든 노드를 차단하고 드레이닝합니다. 모든 항목이 예상대로 작동하면 일시 중지 해제를 결정하기 전에 클러스터 및 워크로드 상태를 평가하여 각 추가 유지 관리 기간 동안 연속으로 **workerpool-A**, **workerpool-B**, **workerpool-C**를 업데이트합니다.

사용자 지정 MCP를 사용하여 작업자 노드 업데이트를 관리하는 것은 유연성을 제공하지만 여러 명령을 실행해야 하는 시간이 많이 걸리는 프로세스일 수 있습니다. 이러한 복잡성으로 인해 전체 클러스터에 영향을 줄 수 있는 오류가 발생할 수 있습니다. 시작하기 전에 조직의 요구 사항을 신중하게 고려하고 프로세스 구현을 신중하게 계획하는 것이 좋습니다.

참고

MCP를 다른 OpenShift Container Platform 버전으로 업데이트하지 않는 것이 좋습니다. 예를 들어 한 MCP를 4.y.10에서 4.y.11로 다른 MCP를 4.y.12로 업데이트하지 마십시오. 이 시나리오는 테스트되지 않아 정의되지 않은 클러스터 상태가 될 수 있습니다.

중요

머신 구성 풀을 일시 중지하면 Machine Config Operator가 연결된 노드에 구성 변경 사항을 적용할 수 없습니다. MCP를 일시 중지하면 자동으로 순환된 인증서가 **kube-apiserver-to-kubelet-signer** CA 인증서의 자동 CA 순환을 포함하여 관련 노드로 푸시되지 않습니다. **kube-apiserver-to-kubelet-signer** CA 인증서가 만료되고 MCO가 인증서를 자동으로 갱신하려고 하면 새 인증서가 생성되지만 해당 머신 구성 풀의 노드에 적용되지 않습니다. 이로 인해 **oc debug**, **oc logs**, **oc exec**, **oc attach**를 포함하여 여러 **oc** 명령이 실패합니다. MCP 일시 중지는 **kube-apiserver-to-kubelet-signer** CA 인증서 만료에 대해 신중하게 고려하여 단기간 동안만 수행해야 합니다.

7.1. 카나리아 롤아웃 업데이트 프로세스 및 MCP 정보

OpenShift Container Platform에서 노드는 개별적으로 간주되지 않습니다. 노드는 MCP(Machine config pool)로 그룹화됩니다. 기본 OpenShift Container Platform 클러스터에는 두 개의 MCP가 있습니다. 하나는 컨트롤 플레인 노드용이고 하나는 작업자 노드용입니다. OpenShift Container Platform 업데이트는 모든 MCP에 동시에 영향을 미칩니다.

업데이트 중에 MCO (Machine Config Operator)는 기본적으로 **1**에서 지정된 **maxUnavailable** 노드 수(지정된 경우)까지 MCP 내의 모든 노드를 드레이닝하고 차단합니다. 노드를 드레이닝하고 차단하면 노드의 모든 Pod 예약이 취소되고 노드가 예약 불가능으로 표시됩니다. 노드를 드레이닝한 후 Machine Config Daemon은 OS(운영 체제) 업데이트를 포함할 수 있는 새 머신 구성을 적용합니다. OS를 업데이트하려면 호스트가 재부팅해야 합니다.

특정 노드가 업데이트되지 않도록 하려면 드레이닝, 차단 및 업데이트되지 않도록 사용자 지정 MCP를 생성할 수 있습니다. 그런 다음 해당 MCP를 일시 중지하여 해당 MCP와 연결된 노드가 업데이트되지 않았는지 확인합니다. MCO는 일시 중지된 MCP를 업데이트하지 않습니다. 하나 이상의 사용자 지정 MCP를 생성하여 해당 노드를 업데이트하는 순서에 대해 더 많은 제어 권한을 부여할 수 있습니다. 첫 번째 MCP에서 노드를 업데이트한 후 애플리케이션 호환성을 확인한 다음 나머지 노드를 새 버전으로 점진적으로 업데이트할 수 있습니다.

참고

컨트롤 플레인의 안정성을 보장하기 위해 컨트롤 플레인 노드 (마스터 노드라고도 함)에서 사용자 정의 MCP를 생성하는 것은 지원되지 않습니다. MCO(Machine Config Operator)는 컨트롤 플레인 노드에 대해 생성된 사용자 정의 MCP를 무시합니다.

워크로드 배포 토폴로지에 따라 생성하는 MCP 수와 각 MCP의 노드 수를 신중하게 고려해야 합니다. 예를 들어 특정 유지 관리 창에 업데이트를 조정해야 하는 경우 창 내에서 OpenShift Container Platform을 업데이트할 수 있는 노드 수를 알아야 합니다. 이 숫자는 고유한 클러스터 및 워크로드 특성에 따라 달라집니다.

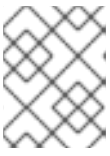
또한 클러스터에서 사용할 수 있는 추가 용량을 고려해야 합니다. 예를 들어 업데이트된 노드에서 애플리케이션이 예상대로 작동하지 않는 경우 풀의 해당 노드를 차단하고 드레이닝하여 애플리케이션 pod를 다른 노드로 이동할 수 있습니다. 필요한 사용자 지정 MCP 수와 각 MCP에 있는 노드 수를 확인하기 위해 사용 가능한 추가 용량을 고려해야 합니다. 예를 들어 두 개의 사용자 지정 MCP를 사용하고 노드의 50%가 각 풀에 있는 경우 노드의 50%를 실행하면 애플리케이션에 충분한 QoS(Quality-of-service)를 제공하는지 확인해야 합니다.

이 업데이트 프로세스를 문서화된 모든 OpenShift Container Platform 업데이트 프로세스와 함께 사용할 수 있습니다. 그러나 이 프로세스는 Ansible 플레이북을 사용하여 업데이트되는 RHEL(Red Hat Enterprise Linux) 시스템에서는 작동하지 않습니다.

7.2. 카나리아 롤아웃 업데이트 수행 정보

다음에서는 카나리아 롤아웃 업데이트 프로세스의 일반적인 워크플로에 대해 설명합니다. 워크플로의 각 작업을 수행하는 단계는 다음 섹션에 설명되어 있습니다.

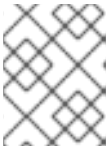
1. 작업자 풀을 기반으로 MCP를 생성합니다. 각 MCP의 노드 수는 각 MCP의 유지 관리 기간 및 클러스터에서 사용할 수 있는 추가 작업자 노드를 의미하는 예약 용량과 같은 몇 가지 요소에 따라 다릅니다.

   참고

   MCP에서 **maxUnavailable** 설정을 변경하여 언제든지 업데이트할 수 있는 머신 수를 지정할 수 있습니다. 기본값은 1입니다.

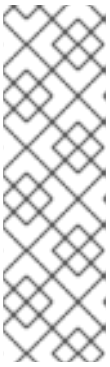
2. 사용자 지정 MCP에 노드 선택기를 추가합니다. 나머지 클러스터와 동시에 업데이트하지 않으려는 각 노드에 대해 일치하는 레이블을 노드에 추가합니다. 이 레이블은 노드를 MCP에 연결합니다.



참고

노드에서 기본 작업자 레이블을 제거하지 마십시오. 노드에는 클러스터에서 제대로 작동하려면 역할 레이블이 **있어야 합니다**.

3. 업데이트 프로세스의 일부로 업데이트하지 않으려는 MCP를 일시 중지합니다.



참고

MCP를 일시 중지하면 kube-apiserver-to-kubelet-signer 자동 CA 인증서 교체도 일시 중지합니다. 새 CA 인증서는 설치 날짜로부터 292일로 생성되며 이전 인증서는 설치 날짜로부터 365일에 제거됩니다. 다음 자동 CA 인증서 교체까지 남은 시간을 알아보려면 Red Hat OpenShift 4의 CA 인증서 자동 갱신 이해 을 참조하십시오. CA 인증서 교체가 수행되면 풀이 일시 중지 해제되었는지 확인합니다. MCP가 일시 중지되면 인증서 교체가 발생하지 않으므로 클러스터의 성능이 저하되고 **oc debug**, **oc logs**, **oc exec**, **oc attach**를 포함하여 여러 **oc** 명령에 오류가 발생합니다.

4. 클러스터 업데이트를 수행합니다. 업데이트 프로세스는 컨트롤 플레인 노드(마스터 노드라고도 함)를 포함하여 일시 중지되지 않은 MCP를 업데이트합니다.

5. 업데이트된 노드에서 애플리케이션을 테스트하여 애플리케이션이 예상대로 작동하는지 확인합니다.

6. 나머지 MCP를 하나씩 일시 중지 해제하고 모든 작업자 노드가 업데이트될 때까지 해당 노드에서 애플리케이션을 테스트합니다. MCP의 일시 중지를 해제하면 해당 MCP와 연결된 노드의 업데이트 프로세스가 시작됩니다. **관리 → 클러스터 설정**을 클릭하여 웹 콘솔에서 업데이트 진행 상황을 확인할 수 있습니다. 또는 **oc get machineconfigpools** CLI 명령을 사용합니다.

7. 선택적으로 업데이트된 노드에서 사용자 지정 레이블을 제거하고 사용자 지정 MCP를 삭제합니다.

7.3. 카나리아 롤아웃 업데이트를 수행할 머신 구성 풀 생성

카나리아 롤아웃 업데이트를 수행하는 첫 번째 작업은 MCP(Machine config pool)를 하나 이상 생성하는 것입니다.

1. 작업자 노드에서 MCP를 생성합니다.

   a. 클러스터의 작업자 노드를 나열합니다.

   ```
   $ oc get -l 'node-role.kubernetes.io/master!=' -o 'jsonpath={range .items[*]}
   {.metadata.name}{"\n"}{end}' nodes
   ```

   출력 예

   ```
   ci-ln-pwnll6b-f76d1-s8t9n-worker-a-s75z4
   ci-ln-pwnll6b-f76d1-s8t9n-worker-b-dglj2
   ci-ln-pwnll6b-f76d1-s8t9n-worker-c-lldbm
   ```

   b. 지연할 노드의 경우 사용자 지정 라벨을 노드에 추가합니다.

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

예를 들면 다음과 같습니다.

```
$ oc label node ci-ln-0qv1yp2-f76d1-kl2tq-worker-a-j2ssz node-
role.kubernetes.io/workerpool-canary=
```

출력 예

```
node/ci-ln-gtrwm8t-f76d1-spbl7-worker-a-xk76k labeled
```

c. 새 MCP를 생성합니다.

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfigPool
metadata:
  name: workerpool-canary 1
spec:
  machineConfigSelector:
    matchExpressions: 2
      - {
        key: machineconfiguration.openshift.io/role,
        operator: In,
        values: [worker,workerpool-canary]
        }
  nodeSelector:
    matchLabels:
      node-role.kubernetes.io/workerpool-canary: "" 3
```

1. MCP의 이름을 지정합니다.
2. **worker** 및 사용자 지정 MCP 이름을 지정합니다.
3. 이 풀에서 원하는 노드에 추가한 사용자 지정 라벨을 지정합니다.

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

출력 예

```
machineconfigpool.machineconfiguration.openshift.io/workerpool-canary created
```

d. 클러스터의 MCP 목록과 현재 상태를 확인합니다.

```
$ oc get machineconfigpool
```

출력 예

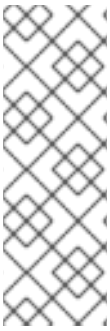
```
NAME            CONFIG                                             UPDATED   UPDATING
DEGRADED   MACHINECOUNT   READYMACHINECOUNT
UPDATEDMACHINECOUNT   DEGRADEDMACHINECOUNT   AGE
master          rendered-master-b0bb90c4921860f2a5d8a2f8137c1867           True
False      False      3            3                3                   0                    97m
workerpool-canary rendered-workerpool-canary-87ba3dec1ad78cb6aecebf7fbb476a36
```

```
True      False      False      1          1          1          0          2m42s
worker    rendered-worker-87ba3dec1ad78cb6aecebf7fbb476a36      True
False     False      2          2          2          0          97m
```

새 머신 구성 풀 **workerpool-canary**가 생성되고 사용자 정의 레이블을 추가한 노드 수가 머신 수에 표시됩니다. 작업자 MCP 머신 수는 동일한 수만큼 줄어듭니다. 시스템 수를 업데이트하는 데 몇 분이 걸릴 수 있습니다. 이 예에서는 하나의 노드가 **worker** MCP에서 **workerpool-canary** MCP로 이동되었습니다.

7.4. 머신 구성 풀 일시 중지

이 카나리아 롤아웃 업데이트 프로세스에서 나머지 OpenShift Container Platform 클러스터로 업데이트하지 않을 노드에 레이블을 지정한 후 MCP(Machine config pool)를 생성하고 해당 MCP를 일시 중지합니다. MCP를 일시 중지하면 MCO(Machine Config Operator)가 해당 MCP와 연결된 노드를 업데이트할 수 없습니다.



참고

MCP를 일시 중지하면 kube-apiserver-to-kubelet-signer 자동 CA 인증서 교체도 일시 중지합니다. 새 CA 인증서는 설치 날짜로부터 292일로 생성되며 이전 인증서는 설치 날짜로부터 365일에 제거됩니다. 다음 자동 CA 인증서 교체까지 남은 시간을 알아보려면 Red Hat OpenShift 4의 CA 인증서 자동 갱신 이해을 참조하십시오. CA 인증서 교체가 수행되면 풀이 일시 중지 해제되었는지 확인합니다. MCP가 일시 중지되면 인증서 교체가 발생하지 않으므로 클러스터의 성능이 저하되고 **oc debug**, **oc logs**, **oc exec**, **oc attach**를 포함하여 여러 **oc** 명령에 오류가 발생합니다.

MCP를 일시 중지하려면 다음을 수행합니다.

1. 일시 중지하려는 MCP를 패치합니다.

```
$ oc patch mcp/<mcp_name> --patch '{"spec":{"paused":true}}' --type=merge
```

예를 들면 다음과 같습니다.

```
$  oc patch mcp/workerpool-canary --patch '{"spec":{"paused":true}}' --type=merge
```

출력 예

```
machineconfigpool.machineconfiguration.openshift.io/workerpool-canary patched
```

7.5. 클러스터 업데이트 수행

MCP가 준비 상태가 되면 클러스터 업데이트를 포맷할 수 있습니다. 클러스터에 적합한 다음 업데이트 방법 중 하나를 참조하십시오.

- 웹 콘솔을 사용하여 클러스터 업데이트
- CLI를 사용하여 클러스터 업데이트

업데이트가 완료되면 MCP의 일시 중지를 하나씩 해제할 수 있습니다.

7.6. 머신 구성 풀 일시 중지 해제

이 카나리아 롤아웃 업데이트 프로세스에서 OpenShift Container Platform 업데이트가 완료된 후 사용자 정의 MCP를 하나씩 일시 중지 해제합니다. MCP의 일시 중지를 해제하면 MCO(Machine Config Operator)가 해당 MCP와 연결된 노드를 업데이트할 수 있습니다.

MCP 일시 중지를 해제하려면 다음을 수행합니다.

1. 일시 중지 해제할 MCP를 패치합니다.

```
$ oc patch mcp/<mcp_name> --patch '{"spec":{"paused":false}}' --type=merge
```

예를 들면 다음과 같습니다.

```
$  oc patch mcp/workerpool-canary --patch '{"spec":{"paused":false}}' --type=merge
```

출력 예

```
machineconfigpool.machineconfiguration.openshift.io/workerpool-canary patched
```

oc get machineconfigpools 명령을 사용하여 업데이트 진행 상황을 확인할 수 있습니다.

2. 업데이트된 노드에서 애플리케이션을 테스트하여 애플리케이션이 예상대로 작동하는지 확인합니다.

3. 일시 중지된 다른 MCP를 하나씩 일시 중지 해제하고 애플리케이션이 작동하는지 확인합니다.

7.6.1. 애플리케이션 장애 발생 시

업데이트된 노드에서 작동하지 않는 애플리케이션 등의 오류가 발생하는 경우 풀의 노드를 차단하고 드레이닝하여 애플리케이션 pod를 다른 노드로 이동하여 애플리케이션의 서비스 품질을 유지 관리할 수 있습니다. 첫 번째 MCP는 초과 용량보다 크지 않아야 합니다.

7.7. 노드를 원래 머신 구성 풀로 이동

이 카나리아 롤아웃 업데이트 프로세스에서 사용자 정의 MCP(MCP)를 일시 중지 해제하고 해당 MCP와 연결된 노드의 애플리케이션이 예상대로 작동하는지 확인한 후 노드에 추가한 사용자 정의 레이블을 제거하여 노드를 원래 MCP로 다시 이동해야 합니다.

중요

노드에는 클러스터에서 제대로 작동하는 역할이 있어야 합니다.

노드를 원래 MCP로 이동하려면 다음을 수행합니다.

1. 노드에서 사용자 지정 레이블을 제거합니다.

```
$ oc label node <node_name> node-role.kubernetes.io/<custom-label>-
```

예를 들면 다음과 같습니다.

```
$ oc label node ci-ln-0qv1yp2-f76d1-kl2tq-worker-a-j2ssz node-
role.kubernetes.io/workerpool-canary-
```

출력 예

```
node/ci-ln-0qv1yp2-f76d1-kl2tq-worker-a-j2ssz labeled
```

MCO는 노드를 원래 MCP로 다시 이동하고 노드를 MCP 구성으로 조정합니다.

2. 클러스터의 MCP 목록과 현재 상태를 확인합니다.

```
$oc get mcp
```

```
NAME              CONFIG                                        UPDATED   UPDATING
DEGRADED   MACHINECOUNT   READYMACHINECOUNT   UPDATEDMACHINECOUNT
DEGRADEDMACHINECOUNT   AGE
master            rendered-master-1203f157d053fd987c7cbd91e3fbc0ed        True      False
False      3              3                   3                   0                   61m
workerpool-canary   rendered-mcp-noupdate-5ad4791166c468f3a35cd16e734c9028   True
False      False      0              0                   0                   0                   21m
worker            rendered-worker-5ad4791166c468f3a35cd16e734c9028        True      False
False      3              3                   3                   0                   61m
```

노드는 사용자 지정 MCP에서 제거되고 원래 MCP로 다시 이동합니다. 시스템 수를 업데이트하는 데 몇 분이 걸릴 수 있습니다. 이 예에서는 하나의 노드가 제거된 **workerpool-canary** MCP에서 'worker'MCP로 이동되었습니다.

3. 선택 사항: 사용자 지정 MCP를 삭제합니다.

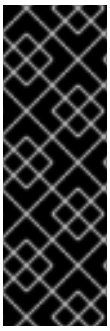
```
$ oc delete mcp <mcp_name>
```

8장. RHEL 컴퓨팅 시스템을 포함하는 클러스터 업데이트

OpenShift Container Platform 클러스터를 업데이트하거나 업그레이드할 수 있습니다. 클러스터에 RHEL (Red Hat Enterprise Linux) 시스템이 포함된 경우 해당 시스템을 업데이트하기 위해 추가 단계를 수행해야합니다.

8.1. 전제 조건

- **admin** 권한이 있는 사용자로 클러스터에 액세스합니다. RBAC를 사용하여 권한 정의 및 적용 을 참조하십시오.
- 업데이트가 실패하는 경우 최근 etcd 백업이 있고 클러스터를 이전 상태로 복원해야 합니다.
- 클러스터에서 수동으로 유지 관리되는 인증 정보를 사용하는 경우 CCO (Cloud Credential Operator)가 업그레이드 가능한 상태인지 확인합니다. 자세한 내용은 AWS, Azure 또는 GCP에 대해 *수동으로 유지 관리되는 인증 정보를 사용하여 클러스터 업그레이드* 참조하십시오.



중요

Prometheus의 PVC로 연결된 클러스터 모니터링을 실행 중인 경우 클러스터 업데이트 중에 OOM이 종료될 수 있습니다. Prometheus에 영구 스토리지를 사용하는 경우 클러스터 업데이트 중 및 업데이트가 완료된 후 몇 시간 동안 Prometheus 메모리 사용량이 두 배로 증가합니다. OOM 종료 문제가 발생하지 않도록 하려면 업데이트 전에 사용 가능한 메모리 크기의 두 배인 작업자 노드를 허용합니다. 예를 들어 최소 권장 노드(8GB RAM이 있는 코어 2개)에서 모니터링을 실행 중인 경우 메모리를 16GB로 늘립니다. 자세한 내용은 BZ#1925061을 참조하십시오.

추가 리소스

- 관리되지 않는 Operator에 대한 지원 정책

8.2. 웹 콘솔을 사용하여 클러스터 업데이트

사용 가능한 업데이트가 있으면 웹 콘솔에서 클러스터를 업데이트할 수 있습니다.

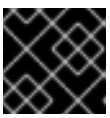
사용 가능한 OpenShift Container Platform 권고 및 업데이트는 고객 포털의 에라타 섹션을 참조하십시오.

전제 조건

- **admin** 권한이있는 사용자로 웹 콘솔에 액세스합니다.

절차

1. 웹 콘솔에서 **Administration → Cluster Settings**을 클릭하고 **Details** 탭의 내용을 확인합니다.
2. 프로덕션 클러스터의 경우 **Channel** 이 **stable-4.7** 등 업데이트하려는 버전에 대한 올바른 채널로 설정되어 있는지 확인합니다.

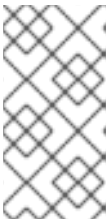


중요

프로덕션 클러스터의 경우 **stable-*** 또는 **fast-*** 채널에 가입해야합니다.

- **Update status** 가 **Updates available** 이 아닌 경우 클러스터를 업데이트할 수 없습니다.
- **Select channel**은 클러스터가 실행 중이거나 업데이트 중인 클러스터 버전을 나타냅니다.

3. 업데이트할 버전을 선택하고 **Save(저장)**를 클릭합니다.
입력 채널 **Update Status**가 **Update to <product-version> in progress**로 변경되고 Operator 및 노드의 진행률을 확인하여 클러스터 업데이트의 진행 상황을 검토할 수 있습니다.

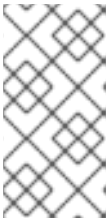


참고

클러스터를 버전 4.y+1과 같이 다음 마이너 버전으로 업그레이드하는 경우 새 기능에 의존하는 워크로드를 배포하기 전에 노드가 업데이트되었는지 확인하는 것이 좋습니다. 아직 업데이트되지 않은 작업자 노드가 있는 풀은 **클러스터 설정** 페이지에 표시됩니다.

4. 업데이트가 완료되고 Cluster Version Operator가 사용 가능한 업데이트를 새로 고침한 후 현재 채널에서 사용 가능한 추가 업데이트가 있는지 확인합니다.
   - 업데이트가 있는 경우 더 이상 업데이트할 수 없을 때까지 현재 채널에서 업데이트를 계속 수행합니다.
   - 사용 가능한 업데이트가 없는 경우 **Channel** 을 다음 마이너 버전의 **stable-*** 또는 **fast-*** 채널로 변경하고 해당 채널에서 원하는 버전으로 업데이트합니다.

   필요한 버전에 도달할 때까지 여러 중간 업데이트를 수행해야 할 수도 있습니다.



참고

RHEL (Red Hat Enterprise Linux) 작업자 시스템이 포함된 클러스터를 업데이트하면 업데이트 프로세스 중에 해당 작업자를 일시적으로 사용할 수 없게 됩니다. 클러스터가 **NotReady** 상태가 되면 각 RHEL 시스템에 대해 업데이트 플레이북을 실행하여 업데이트를 완료해야 합니다.

8.3. 선택 사항: RHEL 시스템에서 ANSIBLE 작업을 수행하기 위한 후크 추가

OpenShift Container Platform을 업데이트할 때 *후크*를 사용하여 RHEL 컴퓨팅 시스템에서 Ansible 작업을 실행할 수 있습니다.

8.3.1. 업그레이드를 위한 Ansible Hook 사용 정보

OpenShift Container Platform을 업데이트할 때 *후크*를 사용하여 특정 작업 중에 RHEL (Red Hat Enterprise Linux) 노드에서 사용자 정의 작업을 실행할 수 있습니다. 후크를 사용하면 특정 업데이트 작업 전후에 실행할 작업을 정의하는 파일을 지정할 수 있습니다. OpenShift Container Platform 클러스터에서 RHEL 컴퓨팅 노드를 업데이트할 때 후크를 사용하여 사용자 정의 인프라의 유효성을 검사하거나 변경할 수 있습니다.

후크가 실패하면 작업도 실패하므로 후크를 여러 번 실행하고 동일한 결과를 얻도록 설계해야합니다.

후크에는 다음과 같은 제한이 있습니다.-후크에는 정의되거나 버전이 지정된 인터페이스가 없습니다. 후크는 내부 **openshift-ansible** 변수를 사용할 수 있지만 이러한 변수는 향후 OpenShift Container Platform 릴리스에서 수정되거나 제거될 수 있습니다. -후크에는 오류 처리 기능이 없으므로 후크에 오류가 발생하면 업데이트 프로세스가 중단됩니다. 오류가 발생하면 문제를 해결한 다음 업그레이드를 다시 시작해야합니다.

8.3.2. 후크를 사용하도록 Ansible 인벤토리 파일 설정

all : vars 섹션 아래의 **hosts** 인벤토리 파일에서 작업자 시스템이라고도 하는 RHEL (Red Hat Enterprise Linux) 컴퓨팅 시스템을 업데이트할 때 사용할 후크를 정의합니다.

전제 조건

- RHEL 컴퓨팅 시스템 클러스터를 추가하는 데 사용되는 컴퓨터에 액세스할 수 있어여 합니다. RHEL 시스템을 정의하는 **hosts** Ansible 인벤토리 파일에 액세스할 수 있어야 합니다.

절차

1. 후크를 설계한 후 Ansible 작업을 정의하는 YAML 파일을 만듭니다. 이 파일은 다음 예와 같이 Playbook이 아닌 일련의 작업으로 구성되어 있어야 합니다.

```
---
# Trivial example forcing an operator to acknowledge the start of an upgrade
# file=/home/user/openshift-ansible/hooks/pre_compute.yml

- name: note the start of a compute machine update
  debug:
     msg: "Compute machine upgrade of {{ inventory_hostname }} is about to start"

- name: require the user agree to start an upgrade
  pause:
     prompt: "Press Enter to start the compute machine update"
```

2. **hosts** Ansible 인벤토리 파일을 수정하여 후크 파일을 지정합니다. 후크 파일은 다음과 같이 **[all : vars]** 섹션에서 매개 변수 값으로 지정됩니다.

   인벤토리 파일에서 후크 정의의 예

```
[all:vars]
openshift_node_pre_upgrade_hook=/home/user/openshift-ansible/hooks/pre_node.yml
openshift_node_post_upgrade_hook=/home/user/openshift-ansible/hooks/post_node.yml
```

   후크 경로의 모호성을 피하려면 정의에서 상대 경로 대신 절대 경로를 사용합니다.

8.3.3. RHEL 컴퓨팅 시스템에서 사용 가능한 후크

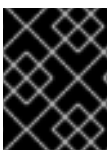
OpenShift Container Platform 클러스터에서 RHEL (Red Hat Enterprise Linux) 컴퓨팅 시스템을 업데이트 할 때 다음 후크를 사용할 수 있습니다.

후크 이름	설명

후크 이름	설명
openshift_node_pre_cordon_hook	• 각 노드가 차단 (cordon)되기 **이전**에 실행됩니다. • 이 후크는 **각 노드**에 대해 연속적으로 실행됩니다. • 작업이 다른 호스트에서 실행되어야하는 경우 해당 작업은 **delegate_to** 또는 **local_action**을 사용해야 합니다.
openshift_node_pre_upgrade_hook	• 각 노드가 차단(cordon)된 **후** 업데이트되기 **전**에 실행됩니다. • 이 후크는 **각 노드**에 대해 연속적으로 실행됩니다. • 작업이 다른 호스트에서 실행되어야하는 경우 해당 작업은 **delegate_to** 또는 **local_action**을 사용해야 합니다.
openshift_node_pre_uncordon_hook	• 각 노드가 업데이트 된 **후** 차단 해제 (uncordon)되기 **전**에 실행됩니다. • 이 후크는 **각 노드**에 대해 연속적으로 실행됩니다. • 작업이 다른 호스트에서 실행되어야하는 경우, 해당 작업은 **delegate_to** 또는 **local_action**을 사용해야 합니다.
openshift_node_post_upgrade_hook	• 각 노드가 차단 해제 (uncordon) **후에** 실행됩니다. 이는 **마지막** 노드 업데이트 작업입니다. • 이 후크는 **각 노드**에 대해 연속적으로 실행됩니다. • 작업이 다른 호스트에서 실행되어야하는 경우 해당 작업은 **delegate_to** 또는 **local_action**을 사용해야 합니다.

8.4. 클러스터에서 RHEL 컴퓨팅 시스템 업데이트

클러스터를 업데이트한 후 클러스터의 RHEL (Red Hat Enterprise Linux) 컴퓨팅 시스템을 업데이트해야 합니다.



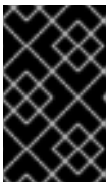
중요

작업자 (compute) 머신에는 RHEL (Red Hat Enterprise Linux) 버전 7.9 이상만 지원되므로 RHEL 작업자 시스템을 버전 8로 업그레이드해서는 안 됩니다.

RHEL을 운영 체제로 사용하는 경우 컴퓨팅 머신을 다른 OpenShift Container Platform 마이너 버전으로 업데이트할 수도 있습니다. 마이너 버전 업데이트를 수행할 때 RHEL에서 RPM 패키지를 제외할 필요가 없습니다.

전제 조건

- 클러스터가 업데이트되었습니다.



중요

RHEL 시스템에 업데이트 프로세스를 완료하기 위해 클러스터에서 생성한 자산이 필요하므로 RHEL 작업자 시스템을 업데이트하기 전에 클러스터를 업데이트해야 합니다.

- RHEL 컴퓨팅 머신을 클러스터에 추가하는 데 사용한 로컬 시스템에 액세스할 수 있습니다. RHEL 시스템 및 **upgrade** Playbook을 정의하는 **hosts** Ansible 인벤토리 파일에 액세스할 수 있어야 합니다.
- 마이너 버전을 업데이트하기 위해 RPM 리포지토리는 클러스터에서 실행 중인 동일한 버전의 OpenShift Container Platform을 사용하고 있습니다.

프로세스

1. 호스트에서 firewalld를 중지하고 비활성화합니다.

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

참고

기본적으로 "최소" 설치 옵션이 있는 기본 OS RHEL에서는 firewalld 서비스를 활성화합니다. 호스트에서 firewalld 서비스를 활성화하면 작업자의 OpenShift Container Platform 로그에 액세스할 수 없습니다. 작업자의 OpenShift Container Platform 로그에 계속 액세스하려면 나중에 firewalld를 활성화하지 마십시오.

2. OpenShift Container Platform 4.7에 필요한 리포지토리를 활성화합니다.
   a. Ansible Playbook을 실행하는 컴퓨터에서 필요한 리포지토리를 업데이트합니다.

```
# subscription-manager repos --disable=rhel-7-server-ose-4.6-rpms \
                           --enable=rhel-7-server-ansible-2.9-rpms \
                           --enable=rhel-7-server-ose-4.7-rpms
```

   b. Ansible Playbook을 실행하는 시스템에서 **openshift-ansible**을 포함하여 필요한 패키지를 업데이트합니다.

```
# yum update openshift-ansible openshift-clients
```

   c. 각 RHEL 컴퓨팅 노드에 필요한 리포지토리를 업데이트합니다.

```
# subscription-manager repos --disable=rhel-7-server-ose-4.6-rpms \
                           --enable=rhel-7-server-ose-4.7-rpms  \
                           --enable=rhel-7-fast-datapath-rpms   \
                           --enable=rhel-7-server-optional-rpms
```

3. RHEL 작업자 시스템을 업데이트합니다.

   a. 현재 노드 상태를 확인하고 업데이트할 RHEL 작업자를 결정합니다.

   ```
   # oc get node
   ```

   출력 예

   ```
   NAME                 STATUS                     ROLES    AGE    VERSION
   mycluster-control-plane-0   Ready                      master   145m   v1.20.0
   mycluster-control-plane-1   Ready                      master   145m   v1.20.0
   mycluster-control-plane-2   Ready                      master   145m   v1.20.0
   mycluster-rhel7-0        NotReady,SchedulingDisabled   worker   98m
   v1.14.6+97c81d00e
   mycluster-rhel7-1        Ready                      worker   98m    v1.14.6+97c81d00e
   mycluster-rhel7-2        Ready                      worker   98m    v1.14.6+97c81d00e
   mycluster-rhel7-3        Ready                      worker   98m    v1.14.6+97c81d00e
   ```

   NotReady, SchedulingDisabled 상태인 시스템을 확인합니다.

   b. 다음 예와 같이 **/<path>/inventory/hosts**에서 Ansible 인벤토리 파일을 확인하고 **NotReady, SchedulingDisabled** 상태인 시스템만 **[workers]** 섹션에 나열되도록 내용을 업데이트합니다.

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

   openshift_kubeconfig_path="~/.kube/config"

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

   c. **openshift-ansible** 디렉토리로 변경합니다.

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

   d. **upgrade** Playbook을 실행합니다.

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

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

   참고

   upgrade 플레이북은 OpenShift Container Platform 패키지만 업그레이드합니다. 운영 체제 패키지를 업데이트하지 않습니다.

4. 이전 단계의 프로세스에 따라 클러스터의 각 RHEL 작업자 시스템을 업데이트합니다.

5. 모든 작업자를 업데이트한 후 모든 클러스터 노드가 새 버전으로 업데이트되었는지 확인합니다.

```
# oc get node
```

출력 예

```
NAME                        STATUS                        ROLES    AGE    VERSION
mycluster-control-plane-0   Ready                         master   145m   v1.20.0
mycluster-control-plane-1   Ready                         master   145m   v1.20.0
mycluster-control-plane-2   Ready                         master   145m   v1.20.0
mycluster-rhel7-0           NotReady,SchedulingDisabled   worker   98m    v1.20.0
mycluster-rhel7-1           Ready                         worker   98m    v1.20.0
mycluster-rhel7-2           Ready                         worker   98m    v1.20.0
mycluster-rhel7-3           Ready                         worker   98m    v1.20.0
```

6. 선택 사항: **업그레이드** 플레이북에서 업데이트하지 않은 운영 체제 패키지를 업데이트합니다. 4.7에 없는 패키지를 업데이트하려면 다음 명령을 사용합니다.

```
# yum update
```

참고

4.7을 설치할 때 사용한 것과 동일한 RPM 리포지토리를 사용하는 경우 RPM 패키지를 제외할 필요가 없습니다.

9장. 네트워크가 제한된 환경에서 클러스터 업데이트

oc CLI(명령줄 인터페이스)를 사용하여 제한된 네트워크 OpenShift Container Platform 클러스터를 업데이트할 수 있습니다.

네트워크가 제한된 환경은 클러스터 노드가 인터넷에 액세스할 수 없는 환경입니다. 따라서 레지스트리에 설치 이미지를 입력해야합니다. 레지스트리 호스트가 인터넷과 클러스터에 모두에 액세스할 수 없는 경우 이미지를 해당 환경에서 분리된 파일 시스템으로 미러링한 다음 호스트 또는 이동식 미디어를 가져올 수 있습니다. 로컬 컨테이너 레지스트리와 클러스터가 미러 레지스트리의 호스트에 연결된 경우 릴리스 이미지를 로컬 레지스트리로 직접 푸시할 수 있습니다.

네트워크가 제한된 환경에 여러 개의 클러스터가 있는 경우 필요한 릴리스 이미지를 단일 컨테이너 이미지 레지스트리에 미러링하고 해당 레지스트리를 사용하여 모든 클러스터를 업데이트합니다.

9.1. 전제 조건

- 필요한 컨테이너 이미지를 얻으려면 인터넷에 액세스할 수 있어야 합니다.
- 네트워크가 제한된 환경에서 컨테이너 레지스트리에 대한 쓰기 권한이 있어야 이미지를 푸시하고 가져올 수 있습니다. 컨테이너 레지스트리는 Docker 레지스트리 API v2와 호환되어야합니다.
- **oc** 명령 줄 인터페이스 (CLI) 툴이 설치되어 있어야합니다.
- **admin** 권한이 있는 사용자로 클러스터에 액세스합니다. RBAC를 사용하여 권한 정의 및 적용 을 참조하십시오.
- 업데이트가 실패하는 경우 최근 etcd 백업이 있고 클러스터를 이전 상태로 복원해야 합니다.
- 모든 MCP(Machine config pool)가 실행 중이고 일시 중지되지 않는지 확인합니다. 업데이트 프로세스 중에 일시 중지된 MCP와 연결된 노드를 건너뜁니다. 카나리아 롤아웃 업데이트 전략을 수행하는 경우 MCP를 일시 중지할 수 있습니다.
- 클러스터에서 수동으로 유지 관리되는 인증 정보를 사용하는 경우 CCO (Cloud Credential Operator)가 업그레이드 가능한 상태인지 확인합니다. 자세한 내용은 AWS, Azure 또는 GCP에 대해 *수동으로 유지 관리되는 인증 정보를 사용하여 클러스터 업그레이드* 참조하십시오.

중요

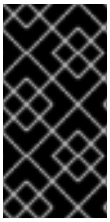
Prometheus의 PVC로 연결된 클러스터 모니터링을 실행 중인 경우 클러스터 업데이트 중에 OOM이 종료될 수 있습니다. Prometheus에 영구 스토리지를 사용하는 경우 클러스터 업데이트 중 및 업데이트가 완료된 후 몇 시간 동안 Prometheus 메모리 사용량이 두 배로 증가합니다. OOM 종료 문제가 발생하지 않도록 하려면 업데이트 전에 사용 가능한 메모리 크기의 두 배인 작업자 노드를 허용합니다. 예를 들어 최소 권장 노드(8GB RAM이 있는 코어 2개)에서 모니터링을 실행 중인 경우 메모리를 16GB로 늘립니다. 자세한 내용은 BZ#1925061을 참조하십시오.

9.2. 미러 호스트 준비

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

9.2.1. 바이너리를 다운로드하여 OpenShift CLI 설치

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



중요

이전 버전의 **oc**를 설치한 경우, OpenShift Container Platform 4.7의 모든 명령을 완료하는 데 해당 버전을 사용할 수 없습니다. 새 버전의 **oc**를 다운로드하여 설치합니다. 네트워크가 제한된 환경에서 클러스터를 업그레이드하는 경우 업그레이드하려는 **oc** 버전을 설치합니다.

9.2.1.1. Linux에서 OpenShift CLI 설치

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

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.7 Linux Client(OpenShift v4.7 Linux 클라이언트)** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 풉니다.

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

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

9.2.1.2. Windows에서 OpenSfhit CLI 설치

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

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.7 Windows Client** 항목 옆에 있는 **지금 다운로드**를 클릭하고 파일을 저장합니다.
4. ZIP 프로그램으로 아카이브의 압축을 풉니다.
5. **oc** 바이너리를 **PATH**에 있는 디렉터리로 이동합니다.
PATH를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

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

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

9.2.1.3. macOS에 OpenShift CLI 설치

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

프로세스

1. Red Hat 고객 포털에서 OpenShift Container Platform 다운로드 페이지 로 이동합니다.
2. **버전** 드롭다운 메뉴에서 적절한 버전을 선택합니다.
3. **OpenShift v4.7 MacOSX Client** 항목 옆에 있는 **지금 다운로드를** 클릭하고 파일을 저장합니다.
4. 아카이브의 압축을 해제하고 압축을 풉니다.
5. **oc** 바이너리 PATH의 디렉터리로 이동합니다.
   PATH를 확인하려면 터미널을 열고 다음 명령을 실행합니다.

   ```
   $ echo $PATH
   ```

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

```
$ oc <command>
```

9.3. 이미지를 미러링할 수 있는 인증 정보 설정

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

주의

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

주의

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

전제 조건

- 네트워크가 제한된 환경에서 사용하도록 미러 레지스트리가 설정되어 있습니다.
- 미러 레지스트리에서 이미지를 미러링할 이미지 저장소 위치를 확인했습니다.
- 이미지를 해당 이미지 저장소에 업로드할 수 있는 미러 레지스트리 계정을 제공하고 있습니다.

프로세스

설치 호스트에서 다음 단계를 수행합니다.

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

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

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

3. 풀 시크릿을 JSON 형식으로 복사합니다.

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

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

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

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

5. 새 파일을 편집하고 레지스트리를 설명하는 섹션을 추가합니다.

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

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

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

파일은 다음 예제와 유사합니다.

```
{
  "auths": {
    "registry.example.com": {
      "auth": "BGVtbYk3ZHAtqXs=",
      "email": "you@example.com"
    },
    "cloud.openshift.com": {
      "auth": "b3BlbnNo...",
      "email": "you@example.com"
    },
    "quay.io": {
      "auth": "b3BlbnNo...",
      "email": "you@example.com"
    },
    "registry.connect.redhat.com": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    },
    "registry.redhat.io": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    }
  }
}
```

9.4. OPENSHIFT CONTAINER PLATFORM 이미지 저장소 미러링

네트워크가 제한된 환경에서 프로비저닝하는 인프라에서 클러스터를 업데이트하기 전에 필요한 컨테이너 이미지를 해당 환경에 미러링해야 합니다. 이 프로세스를 무제한 네트워크에서 사용하여 클러스터가 외부 컨텐츠에 대해 조직의 제어 조건을 충족하는 컨테이너 이미지만 사용하도록 할 수 있습니다.

프로세스

1. Red Hat OpenShift Container Platform Upgrade Graph 시각화 프로그램 및 업데이트 플래너 를 사용하여 한 버전에서 다른 버전으로의 업데이트를 계획합니다. OpenShift Upgrade Graph는 채널 그래프와 현재 클러스터 버전과 예약된 클러스터 버전 간의 업데이트 경로가 있는지 확인하는 방법을 제공합니다.

2. 필요한 환경 변수를 설정합니다.

a. 릴리스 버전을 내보냅니다.

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

& **lt;release_version** >에 대해 업데이트할 OpenShift Container Platform 버전에 해당하는 태그를 지정합니다 (예: **4.5.4**) .

b. 로컬 레지스트리 이름 및 호스트 포트를 내보냅니다.

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

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

c. 로컬 저장소 이름을 내보냅니다.

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

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

d. 미러링할 저장소 이름을 내보냅니다.

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

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

e. 레지스트리 풀 시크릿의 경로를 내보냅니다.

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

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

참고

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

f. 릴리스 미러를 내보냅니다.

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

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

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

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

h. 미러링된 이미지를 호스트할 디렉터리의 경로를 내보냅니다.

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

1 초기 슬래시 (/) 문자를 포함하여 전체 경로를 지정합니다.

3. 미러링할 이미지 및 설정 매니페스트를 확인합니다.

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

4. 버전 이미지를 미러 레지스트리에 미러링합니다.

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

  i. 이동식 미디어를 인터넷에 연결된 시스템에 연결합니다.

  ii. 이미지 및 설정 매니페스트를 이동식 미디어의 디렉토리에 미러링합니다.

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

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

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

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

  iv. **oc** CLI(명령줄 인터페이스)를 사용하여 업그레이드 중인 클러스터에 로그인합니다.

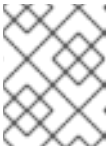
  v. 미러링된 릴리스 이미지 서명 config map을 연결된 클러스터에 적용합니다.

```
$ oc apply -f ${REMOVABLE_MEDIA_PATH}/mirror/config/<image_signature_file>
1
```

1 < **image_signature_file** >은 파일의 경로와 이름을 지정합니다(예: **signature-sha256-81154f5c03294534.yaml**).

- 로컬 컨테이너 레지스트리와 클러스터가 미러 호스트에 연결된 경우 릴리스 이미지를 로컬 레지스트리로 직접 푸시하고 다음 명령을 사용하여 config map을 클러스터에 적용합니다.

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



참고

--apply-release-image-signature 옵션이 포함된 경우 이미지 서명 확인을 위해 config map을 작성하지 않습니다.

9.5. 이미지 서명 CONFIG MAP 생성

클러스터를 업데이트하기 전에 사용하는 릴리스 이미지의 서명이 포함된 config map을 수동으로 생성해야합니다. 이 서명을 통해 CVO (Cluster Version Operator)는 예상되는 이미지와 실제 이미지 서명을 비교하여 릴리스 이미지가 변경되지 않았는지 확인할 수 있습니다.

버전 4.4.8 이상에서 업그레이드하는 경우 **oc** CLI를 사용하여 config map을 생성할 수 있습니다. 이전 버전에서 업그레이드하는 경우 수동 방법을 사용해야합니다.

9.5.1. 이미지 서명 config map 수동으로 생성

이미지 서명 config map을 만들고 업데이트하려는 클러스터에 적용합니다.

참고

클러스터를 업데이트할 때 마다 다음 단계를 수행해야합니다.

프로세스

1. OpenShift Container Platform 업그레이드 경로 지식 베이스 문서를 참조하여 클러스터의 유효한 업데이트 경로를 결정합니다.
2. **OCP_RELEASE_NUMBER** 환경 변수에 버전을 추가합니다.

```
$ OCP_RELEASE_NUMBER=<release_version> 1
```

1 **<release_version>**에 대해 클러스터를 업데이트하려는 OpenShift Container Platform 버전에 해당하는 태그를 지정합니다 (예: **4.4.0**).

3. **ARCHITECTURE** 환경 변수에 클러스터의 시스템 아키텍처를 추가합니다.

```
$ ARCHITECTURE=<server_architecture> 1
```

1 **server_architecture**에 대해 **x86_64**과 같은 서버 아키텍처를 지정합니다.

4. Quay 에서 릴리스 이미지 다이제스트를 가져옵니다.

```
$ DIGEST="$(oc adm release info quay.io/openshift-release-dev/ocp-release:${OCP_RELEASE_NUMBER}-${ARCHITECTURE} | sed -n 's/Pull From: .*@//p')"
```

5. 다이제스트 알고리즘을 설정합니다.

```
$ DIGEST_ALGO="${DIGEST%%:*}"
```

6. 다이제스트 서명을 설정합니다.

```
$ DIGEST_ENCODED="${DIGEST#*:}"
```

7. mirror.openshift.com 웹 사이트에서 이미지 서명을 가져옵니다.

```
$ SIGNATURE_BASE64=$(curl -s "https://mirror.openshift.com/pub/openshift-
v4/signatures/openshift/release/${DIGEST_ALGO}=${DIGEST_ENCODED}/signature-1" |
base64 -w0 && echo)
```

8. config map을 생성합니다.

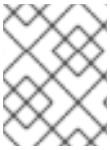
```
$ cat >checksum-${OCP_RELEASE_NUMBER}.yaml <<EOF
apiVersion: v1
kind: ConfigMap
metadata:
  name: release-image-${OCP_RELEASE_NUMBER}
  namespace: openshift-config-managed
  labels:
    release.openshift.io/verification-signatures: ""
binaryData:
  ${DIGEST_ALGO}-${DIGEST_ENCODED}: ${SIGNATURE_BASE64}
EOF
```

9. 업데이트할 클러스터에 config map을 적용합니다.

```
$ oc apply -f checksum-${OCP_RELEASE_NUMBER}.yaml
```

9.6. 네트워크가 제한된 환경의 클러스터 업데이트

네트워크가 제한된 환경의 클러스터를 다운로드한 릴리스 이미지의 OpenShift Container Platform 버전으로 업데이트합니다.



참고

로컬 OpenShift Update Service가 있는 경우 이 절차 대신 연결된 웹 콘솔 또는 CLI 지침을 사용하여 업데이트할 수 있습니다.

전제 조건

- 새 릴리스의 이미지를 레지스트리에 미러링하고 있습니다.
- 새 릴리스의 릴리스 이미지 서명 ConfigMap을 클러스터에 적용하고 있습니다.
- 이미지 서명 ConfigMap에서 릴리스의 sha256 합계 값을 얻을 수 있습니다.
- OpenShift CLI (**oc**), 4.4.8 이상 버전을 설치합니다.

프로세스

- 클러스터를 업데이트합니다.

```
$ oc adm upgrade --allow-explicit-upgrade --to-image
${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}<sha256_sum_value> (1)
```

1 **<sha256_sum_value>** 값은 이미지 서명 ConfigMap에서 릴리스에 대한 sha256 합계입니다 (예:

미러 레지스트리에 **ImageContentSourcePolicy**를 사용하는 경우 **LOCAL_REGISTRY** 대신 표준 레지스트리 이름을 사용할 수 있습니다.

참고

ImageContentSourcePolicy 개체가 있는 클러스터에 대한 글로벌 풀 시크릿만 구성할 수 있습니다. 프로젝트에 풀 시크릿을 추가할 수 없습니다.

9.7. 이미지 레지스트리 저장소 미러링 설정

컨테이너 레지스트리 저장소 미러링을 설정하면 다음을 수행할 수 있습니다.

- 소스 이미지 레지스트리의 저장소에서 이미지를 가져오기 위해 요청을 리디렉션하고 미러링된 이미지 레지스트리의 저장소에서 이를 해석하도록 OpenShift Container Platform 클러스터를 설정합니다.
- 하나의 미러가 다운된 경우 다른 미러를 사용할 수 있도록 각 대상 저장소에 대해 여러 미러링된 저장소를 확인합니다.

다음은 OpenShift Container Platform의 저장소 미러링의 몇 가지 속성입니다.

- 이미지 풀은 레지스트리 다운타임에 탄력적으로 대처할 수 있습니다.
- 제한된 네트워크의 클러스터는 중요한 위치 (예: quay.io)에서 이미지를 가져오도록 요청할 수 있으며 회사의 방화벽 뒤의 레지스트리에서 요청된 이미지를 제공하도록 할 수 있습니다.
- 이미지 가져오기 요청이 있으면 특정한 레지스트리 순서로 가져오기를 시도하며 일반적으로 영구 레지스트리는 마지막으로 시도합니다.
- 입력한 미러링 정보는 OpenShift Container Platform 클러스터의 모든 노드에서 **/etc/containers/registries.conf** 파일에 추가됩니다.
- 노드가 소스 저장소에서 이미지를 요청하면 요청된 컨텐츠를 찾을 때 까지 미러링된 각 저장소를 차례로 시도합니다. 모든 미러가 실패하면 클러스터는 소스 저장소를 시도합니다. 성공하면 이미지를 노드로 가져올 수 있습니다.

저장소 미러링은 다음과 같은 방법으로 설정할 수 있습니다.

- OpenShift Container Platform 설치 시
  OpenShift Container Platform에 필요한 컨테이너 이미지를 가져온 다음 해당 이미지를 회사 방화벽 뒤에 배치하면 제한된 네트워크에 있는 데이터 센터에 OpenShift Container Platform을 설치할 수 있습니다.
- OpenShift Container Platform 설치 후
  OpenShift Container Platform 설치 시 미러링을 설정하지 않고 **ImageContentSourcePolicy** 개체를 사용하여 나중에 설정할 수 있습니다.

다음 절차에서는 설치 후 미러 구성을 제공합니다. 이때 다음을 식별하는 **ImageContentSourcePolicy** 오브젝트를 생성할 수 있습니다.

- 미러링하려는 컨테이너 이미지 저장소의 소스

- 소스 저장소에서 요청된 컨텐츠를 제공하는 각 미러 저장소에 대한 개별 항목

참고

ImageContentSourcePolicy 개체가 있는 클러스터에 대한 글로벌 풀 시크릿만 구성할 수 있습니다. 프로젝트에 풀 시크릿을 추가할 수 없습니다.

전제 조건

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

프로세스

1. 미러링된 저장소를 설정합니다.

   - Red Hat Quay Repository Mirroring 에 설명된대로 Red Hat Quay를 사용하여 미러링된 저장소를 설정합니다. Red Hat Quay를 사용하면 한 저장소에서 다른 저장소로 이미지를 복사하고 시간이 지남에 따라 해당 저장소를 반복해서 자동으로 동기화할 수 있습니다.

   - **skopeo**와 같은 툴을 사용하여 소스 디렉토리에서 미러링된 저장소로 이미지를 수동으로 복사합니다.
   예를 들어, Red Hat Enterprise Linux(RHEL) 7 또는 RHEL 8 시스템에 skopeo RPM 패키지를 설치한 후 다음 예와 같이 **skopeo** 명령을 사용합니다.

     ```
     $ skopeo copy \
     docker://registry.access.redhat.com/ubi8/ubi-minimal@sha256:5cfbaf45ca96806917830c183e9f37df2e913b187adb32e89fd83fa455ebaa6 \
     docker://example.io/example/ubi-minimal
     ```

     이 예제에는 **example.io**라는 컨테이너 이미지 레지스트리가 있으며, **registry.access.redhat.com**에서 **ubi8/ubi-minimal** 이미지를 복사할 **example**이라는 이미지 저장소가 있습니다. 레지스트리를 생성한 후 OpenShift Container Platform 클러스터를 설정하여 소스 저장소의 요청을 미러링된 저장소로 리디렉션할 수 있습니다.

2. OpenShift Container Platform 클러스터에 로그인합니다.

3. **ImageContentSourcePolicy** 파일(예: **registryrepomirror.yaml**)을 생성하고 소스 및 미러를 특정 레지스트리 및 저장소 쌍과 이미지로 교체합니다.

   ```
   apiVersion: operator.openshift.io/v1alpha1
   kind: ImageContentSourcePolicy
   metadata:
     name: ubi8repo
   spec:
     repositoryDigestMirrors:
     - mirrors:
       - example.io/example/ubi-minimal (1)
       source: registry.access.redhat.com/ubi8/ubi-minimal (2)
     - mirrors:
       - example.com/example/ubi-minimal
       source: registry.access.redhat.com/ubi8/ubi-minimal
     - mirrors:
       - mirror.example.com/redhat
       source: registry.redhat.io/openshift4 (3)
   ```

-

1. 이미지 레지스트리 및 저장소의 이름을 가리킵니다.
2. 미러링된 컨텐츠를 포함하는 레지스트리 및 저장소를 가리킵니다.
3. 해당 네임스페이스의 이미지를 사용하도록 레지스트리 내에서 네임스페이스를 구성할 수 있습니다. 레지스트리 도메인을 소스로 사용하는 경우 **ImageContentSourcePolicy** 리소스가 레지스트리의 모든 리포지토리에 적용됩니다.

4. 새 **ImageContentSourcePolicy** 개체를 생성합니다.

```
$ oc create -f registryrepomirror.yaml
```

ImageContentSourcePolicy 개체가 생성된 후 새 설정이 각 노드에 배포된 클러스터는 소스 저장소에 대한 요청에 미러링된 저장소를 사용하기 시작합니다.

5. 미러링된 설정이 적용되었는지 확인하려면 노드 중 하나에서 다음을 수행하십시오.

   a. 노드를 나열합니다.

```
$ oc get node
```

   출력 예

```
NAME                         STATUS                     ROLES    AGE  VERSION
ip-10-0-137-44.ec2.internal   Ready                      worker   7m   v1.20.0
ip-10-0-138-148.ec2.internal   Ready                      master   11m  v1.20.0
ip-10-0-139-122.ec2.internal   Ready                      master   11m  v1.20.0
ip-10-0-147-35.ec2.internal   Ready,SchedulingDisabled   worker   7m   v1.20.0
ip-10-0-153-12.ec2.internal   Ready                      worker   7m   v1.20.0
ip-10-0-154-10.ec2.internal   Ready                      master   11m  v1.20.0
```

   변경 사항이 적용되어 있기 때문에 각 작업자 노드의 예약이 비활성화되어 있음을 알 수 있습니다.

   b. 디버깅 프로세스를 시작하고 노드에 액세스합니다.

```
$ oc debug node/ip-10-0-147-35.ec2.internal
```

   출력 예

```
Starting pod/ip-10-0-147-35ec2internal-debug ...
To use host binaries, run `chroot /host`
```

   c. 노드의 파일에 액세스합니다.

```
sh-4.2# chroot /host
```

   d. **/etc/containers/registries.conf** 파일을 체크하여 변경 사항이 적용되었는지 확인합니다.

```
sh-4.2# cat /etc/containers/registries.conf
```

   출력 예

```
unqualified-search-registries = ["registry.access.redhat.com", "docker.io"]
[[registry]]
  location = "registry.access.redhat.com/ubi8/"
  insecure = false
  blocked = false
  mirror-by-digest-only = true
  prefix = ""

  [[registry.mirror]]
    location = "example.io/example/ubi8-minimal"
    insecure = false

  [[registry.mirror]]
    location = "example.com/example/ubi8-minimal"
    insecure = false
```

e. 소스의 이미지 다이제스트를 노드로 가져와 실제로 미러링에 의해 해결되는지 확인합니다. **ImageContentSourcePolicy** 개체는 이미지 태그가 아닌 이미지 다이제스트만 지원합니다.

```
sh-4.2# podman pull --log-level=debug registry.access.redhat.com/ubi8/ubi-minimal@sha256:5cfbaf45ca96806917830c183e9f37df2e913b187adb32e89fd83fa455ebaa6
```

저장소 미러링 문제 해결

저장소 미러링 절차가 설명대로 작동하지 않는 경우 저장소 미러링 작동 방법에 대한 다음 정보를 사용하여 문제를 해결하십시오.

- 가져온 이미지는 첫 번째 작동 미러를 사용하여 공급합니다.
- 주요 레지스트리는 다른 미러가 작동하지 않는 경우에만 사용됩니다.
- 시스템 컨텍스트에서 **Insecure** 플래그가 폴백으로 사용됩니다.
- **/etc/containers/registries.conf** 파일 형식이 최근에 변경되었습니다. 현재 버전은 TOML 형식의 버전 2입니다.

9.8. 클러스터 노드 재부팅 빈도를 줄이기 위해 미러 이미지 카탈로그의 범위 확장

미러링된 이미지 카탈로그의 범위를 저장소 수준 또는 더 넓은 레지스트리 수준에서 지정할 수 있습니다. 광범위한 **ImageContentSourcePolicy** 리소스는 리소스 변경에 따라 노드를 재부팅해야 하는 횟수를 줄입니다.

ImageContentSourcePolicy 리소스에서 미러 이미지 카탈로그의 범위를 확장하려면 다음 절차를 수행합니다.

전제 조건

- OpenShift Container Platform CLI **oc**를 설치합니다.
- **cluster-admin** 권한이 있는 사용자로 로그인합니다.
- 연결이 끊긴 클러스터에서 사용할 미러링된 이미지 카탈로그를 구성합니다.

절차

1. **<local_registry>**, **<pull_spec>**, 및 **<pull_secret_file>**에 대한 값을 지정하여 다음 명령을 실행합니다.

```
$ oc adm catalog mirror <local_registry>/<pull_spec> <local_registry> -a <pull_secret_file> --icsp-scope=registry
```

   다음과 같습니다.

   <local_registry>

   연결이 끊긴 클러스터에 대해 구성한 로컬 레지스트리입니다 (예: **local.registry:5000**).

   <pull_spec>

   연결이 끊긴 레지스트리에 구성된 풀 사양입니다(예: **redhat/redhat-operator-index:v4.7**).

   <pull_secret_file>

   은 **.json** 파일 형식의 **registry.redhat.io** 풀 시크릿입니다. Red Hat OpenShift Cluster Manager에서 풀 시크릿을 다운로드할 수 있습니다.

   oc adm catalog mirror 명령은 **/redhat-operator-index-manifests** 디렉터리를 생성하고 **imageContentSourcePolicy.yaml**,**catalogSource.yaml** 및 **mapping.txt** 파일을 생성합니다.

2. 새 **ImageContentSourcePolicy** 리소스를 클러스터에 적용합니다.

```
$ oc apply -f imageContentSourcePolicy.yaml
```

검증

- **oc apply**가 **ImageContentSourcePolicy**에 변경 사항을 성공적으로 적용했는지 확인합니다.

```
$ oc get ImageContentSourcePolicy -o yaml
```

출력 예

```
apiVersion: v1
items:
- apiVersion: operator.openshift.io/v1alpha1
  kind: ImageContentSourcePolicy
  metadata:
    annotations:
      kubectl.kubernetes.io/last-applied-configuration: |

{"apiVersion":"operator.openshift.io/v1alpha1","kind":"ImageContentSourcePolicy","metadata":{"annotations":{},"name":"redhat-operator-index"},"spec":{"repositoryDigestMirrors":[{"mirrors":["local.registry:5000"],"source":"registry.redhat.io"}]}}
...
```

ImageContentSourcePolicy 리소스를 업데이트한 후 OpenShift Container Platform은 새 설정을 각 노드에 배포하고 클러스터는 소스 저장소에 대한 요청에 미러링된 저장소를 사용하기 시작합니다.

9.9. 추가 리소스

- 제한된 네트워크에서 Operator Lifecycle Manager 사용

- 머신 구성 개요
- OpenShift 업데이트 서비스 설치 및 구성