



# OpenShift Container Platform 4.9

지원

OpenShift Container Platform 지원 요청



# OpenShift Container Platform 4.9 지원

---

OpenShift Container Platform 지원 요청

## 법적 공지

Copyright © 2023 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux<sup>®</sup> is the registered trademark of Linus Torvalds in the United States and other countries.

Java<sup>®</sup> is a registered trademark of Oracle and/or its affiliates.

XFS<sup>®</sup> is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL<sup>®</sup> is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js<sup>®</sup> is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack<sup>®</sup> Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

## 초록

OpenShift Container Platform에 대한 Red Hat 지원을 요청하는 방법에 대해 설명합니다. 또한 Telemetry 및 Insights Operator를 통한 원격 상태 모니터링에 대한 정보도 포함되어 있습니다. 이 문서에서는 원격 상태 모니터링이 제공하는 이점도 자세히 설명합니다.

## 차례

|   |           |
|---|-----------|
| <b>1장. 지원 개요</b> .....                                      | <b>3</b>  |
| 1.1. 지원 받기  | 3         |
| 1.2. 원격 상태 모니터링 문제  | 3         |
| 1.3. 클러스터에 대한 데이터 수집  | 3         |
| 1.4. 문제 해결  | 4         |
| <b>2장. 클러스터 리소스 관리</b> .....                                | <b>6</b>  |
| 2.1. 클러스터 리소스와 상호 작용  | 6         |
| <b>3장. 지원 요청</b> .....                                      | <b>7</b>  |
| 3.1. 지원 요청  | 7         |
| 3.2. RED HAT 지식베이스 정보                                       | 7         |
| 3.3. RED HAT 지식베이스 검색                                       | 7         |
| 3.4. 지원 케이스 제출  | 8         |
| 3.5. 추가 리소스   | 9         |
| <b>4장. 클러스터에 연결하여 원격 상태 모니터링</b> .....                      | <b>10</b> |
| 4.1. 원격 상태 모니터링 정보  | 10        |
| 4.2. 원격 상태 모니터링으로 수집된 데이터 표시                                | 14        |
| 4.3. 원격 상태 보고 비활성화  | 15        |
| 4.4. INSIGHTS를 사용하여 클러스터의 문제 식별                             | 18        |
| 4.5. INSIGHTS OPERATOR 사용                                   | 21        |
| 4.6. 네트워크가 제한된 환경에서 원격 상태 보고 사용                             | 22        |
| 4.7. INSIGHTS OPERATOR를 사용하여 간단한 콘텐츠 액세스 인증서 가져오기           | 27        |
| <b>5장. 클러스터에 대한 데이터 수집</b> .....                            | <b>30</b> |
| 5.1. MUST-GATHER 툴 정보                                       | 30        |
| 5.2. 클러스터 ID 검색   | 36        |
| 5.3. SOSREPORT 정보   | 37        |
| 5.4. OPENSIFT CONTAINER PLATFORM 클러스터 노드의 SOSREPORT 아카이브 생성 | 37        |
| 5.5. 부트 스트랩 노드의 저널 로그 쿼리                                    | 40        |
| 5.6. 클러스터 노드의 저널 로그 쿼리                                      | 40        |
| 5.7. OPENSIFT CONTAINER PLATFORM 노드 또는 컨테이너에서 네트워크 추적 수집    | 41        |
| 5.8. RED HAT 지원에 진단 데이터 제공                                  | 44        |
| 5.9. TOOLBOX정보  | 46        |
| <b>6장. 클러스터 사양 요약</b> .....                                 | <b>48</b> |
| 6.1. CLUSTERVERSION을 통해 클러스터 사양 요약                          | 48        |
| <b>7장. 문제 해결</b> .....                                      | <b>49</b> |
| 7.1. 설치 문제 해결   | 49        |
| 7.2. 노드 상태 확인   | 68        |
| 7.3. CRI-O 컨테이너 런타임 문제 해결                                   | 71        |
| 7.4. 운영 체제 문제 해결  | 75        |
| 7.5. 네트워크 문제 해결   | 78        |
| 7.6. OPERATOR 문제 해결   | 83        |
| 7.7. POD 문제 조사  | 95        |
| 7.8. SOURCE-TO-IMAGE 프로세스 문제 해결                             | 101       |
| 7.9. 스토리지 문제 해결   | 105       |
| 7.10. WINDOWS 컨테이너 워크로드 문제 해결                               | 105       |
| 7.11. 모니터링 문제 조사  | 109       |
| 7.12. OPENSIFT CLI (OC) 문제 진단                               | 114       |



# 1장. 지원 개요

Red Hat은 클러스터, 모니터링 및 문제 해결을 위한 데이터를 수집하기 위한 클러스터 관리자 툴을 제공합니다.

## 1.1. 지원 받기

**지원 받기:** Red Hat 고객 포털을 방문하여 기술 자료 문서를 검토하고, 지원 사례를 제출하고, 추가 제품 문서와 리소스를 검토합니다.

## 1.2. 원격 상태 모니터링 문제

**원격 상태 모니터링 문제:** OpenShift Container Platform은 클러스터에 대한 Telemetry 및 구성 데이터를 수집하고 Telemeter Client 및 Insights Operator를 사용하여 Red Hat에 보고합니다. Red Hat은 이 데이터를 사용하여 **연결된 클러스터**의 문제를 이해하고 해결합니다. 연결된 클러스터와 유사하게 **제한된 네트워크**에서 원격 상태 모니터링을 사용할 수 있습니다. OpenShift Container Platform은 다음을 사용하여 데이터를 수집하고 상태를 모니터링합니다.

- **Telemetry:** Telemetry Client는 4분 30초마다 지표 값을 수집하고 Red Hat에 업로드합니다. Red Hat은 이 데이터를 사용하여 다음을 수행합니다.
  - 클러스터 모니터링.
  - OpenShift Container Platform 업그레이드를 롤아웃합니다.
  - 업그레이드 환경을 개선합니다.
- **Insights Operator:** 기본적으로 OpenShift Container Platform은 2 시간마다 구성 및 구성 요소 오류 상태를 보고하는 Insight Operator를 설치하고 활성화합니다. Insight Operator는 다음을 지원합니다.
  - 잠재적인 클러스터 문제를 사전에 식별합니다.
  - Red Hat OpenShift Cluster Manager에서 솔루션 및 예방 조치를 제공합니다.

원격 분석 정보를 검토할 수 있습니다.

원격 상태 보고를 활성화한 경우 **Insights를 사용하여 문제를 확인합니다**. 선택적으로 원격 상태 보고를 비활성화할 수 있습니다.

## 1.3. 클러스터에 대한 데이터 수집

**클러스터에 대한 데이터 수집:** 지원 케이스를 열 때 디버깅 정보를 수집하는 것이 좋습니다. 이는 Red Hat 지원에서 근본 원인 분석을 수행하는 데 도움이 됩니다. 클러스터 관리자는 다음을 사용하여 클러스터에 대한 데이터를 수집할 수 있습니다.

- **must-gather 툴:** **must-gather** 툴을 사용하여 클러스터에 대한 정보를 수집하고 문제를 디버깅합니다.
- **sosreport:** 디버깅 목적으로 **sosreport** 도구를 사용하여 구성 세부 정보, 시스템 정보 및 진단 데이터를 수집합니다.
- **클러스터 ID:** Red Hat 지원에 정보를 제공할 때 클러스터의 고유 식별자를 받습니다.

- **부트스트랩 노드 저널 로그:** 부트스트랩 노드에서 **bootkube.service journald** 장치 로그 및 컨테이너 로그를 수집하여 부트스트랩 관련 문제를 해결합니다.
- **클러스터 노드 저널 로그:** 노드 관련 문제를 해결하기 위해 개별 클러스터 노드의 **/var/log** 내의 장치 로그와 로그를 수집합니다.
- **네트워크 추적:** 네트워크 관련 문제를 해결하기 위해 특정 OpenShift Container Platform 클러스터 노드 또는 컨테이너의 네트워크 패킷 추적을 Red Hat 지원에 제공합니다.
- **진단 데이터:** **redhat-support-tool** 명령을 사용하여 클러스터에 대한 진단 데이터를 수집합니다.

## 1.4. 문제 해결

클러스터 관리자는 다음 OpenShift Container Platform 구성 요소 문제를 모니터링하고 해결할 수 있습니다.

- **설치 문제:** OpenShift Container Platform 설치가 다양한 단계에 걸쳐 진행됩니다. 다음을 수행할 수 있습니다.
  - 설치 단계를 모니터링합니다.
  - 설치 문제가 발생하는 단계 확인.
  - 여러 설치 문제를 조사합니다.
  - 실패한 설치에서 로그를 수집합니다.
- **노드 문제:** 클러스터 관리자는 노드의 상태, 리소스 사용량 및 구성을 검토하여 노드 관련 문제를 확인하고 해결할 수 있습니다. 다음을 쿼리할 수 있습니다.
  - 노드의 kubelet 상태입니다.
  - 클러스터 노드 저널 로그.
- **crio 문제:** 클러스터 관리자는 각 클러스터 노드에서 CRI-O 컨테이너 런타임 엔진 상태를 확인할 수 있습니다. 컨테이너 런타임 문제가 발생하면 다음을 수행합니다.
  - CRI-O journald 장치 로그를 수집합니다.
  - CRI-O 스토리지 정리.
- **운영 체제 문제:** OpenShift Container Platform은 Red Hat Enterprise Linux CoreOS에서 실행됩니다. 운영 체제 문제가 발생하면 커널 크래시 프로시저를 조사할 수 있습니다. 다음을 확인합니다.
  - kdump 활성화.
  - kdump 구성을 테스트합니다.
  - 코어 덤프 분석.
- **네트워크 문제:** Open vSwitch 문제를 해결하려면 클러스터 관리자가 다음을 수행할 수 있습니다.
  - Open vSwitch 로그 수준을 일시적으로 구성합니다.
  - Open vSwitch 로그 수준을 영구적으로 구성합니다.
  - Open vSwitch 로그를 표시합니다.



- **Operator 문제:** 클러스터 관리자는 다음을 수행하여 Operator 문제를 해결할 수 있습니다.
  - Operator 서브스크립션 상태를 확인합니다.
  - Operator Pod 상태를 확인합니다.
  - Operator 로그를 수집합니다.
- **Pod 문제:** 클러스터 관리자는 Pod 상태를 검토하고 다음을 수행하여 Pod 관련 문제를 해결할 수 있습니다.
  - 포트 및 컨테이너 로그를 검토합니다.
  - 루트 액세스 권한으로 디버그 Pod를 시작합니다.
- **S2I 문제:** 클러스터 관리자는 S2I 단계를 관찰하여 S2I 프로세스에서 오류가 발생한 위치를 확인할 수 있습니다. 다음을 수집하여 S2I(Source-to-Image) 문제를 해결합니다.
  - S2I(Source-to-Image) 진단 데이터.
  - 애플리케이션 오류를 조사하기 위한 애플리케이션 진단 데이터.
- **스토리지 문제:** 실패한 노드에서 연결된 볼륨을 마운트 해제할 수 없기 때문에 새 노드에 볼륨을 마운트할 수 없는 경우 다중 연결 스토리지 오류가 발생합니다. 클러스터 관리자는 다음을 수행하여 다중 연결 스토리지 문제를 해결할 수 있습니다.
  - RWX 볼륨을 사용하여 여러 연결을 활성화합니다.
  - RWO 볼륨을 사용할 때 오류가 발생한 노드를 복구하거나 삭제합니다.
- **모니터링 문제:** 클러스터 관리자는 문제 해결 페이지의 절차에 따라 모니터링할 수 있습니다. 사용자 정의 프로젝트의 메트릭을 사용할 수 없거나 Prometheus가 많은 디스크 공간을 사용하는 경우 다음을 확인하십시오.
  - 사용자 정의 지표를 사용할 수 없는 이유를 조사합니다.
  - Prometheus가 많은 디스크 공간을 사용하는 이유를 확인합니다.
- **로깅 문제:** 클러스터 관리자는 OpenShift 로깅 문제에 대한 문제 해결 페이지에서 절차를 실행할 수 있습니다. 다음을 확인하여 로깅 문제를 해결합니다.
  - [Logging Operator의 상태입니다.](#)
  - [로그 저장소의 상태.](#)
  - [OpenShift 로깅 경고.](#)
  - [oc adm must-gather 명령을 사용하여 OpenShift 로깅 환경에 대한 정보입니다.](#)
- **OpenShift CLI (oc) 문제:** 로그 수준을 늘려 OpenShift CLI (oc) 문제를 조사합니다.

## 2장. 클러스터 리소스 관리

OpenShift Container Platform에 글로벌 구성 옵션을 적용할 수 있습니다. Operator는 클러스터 전체에 이러한 구성 설정을 적용합니다.

### 2.1. 클러스터 리소스와 상호 작용

OpenShift Container Platform에서 OpenShift CLI(oc)를 사용하여 클러스터 리소스와 상호 작용할 수 있습니다. **oc api-resources** 명령을 실행한 후 표시되는 클러스터 리소스를 편집할 수 있습니다.

#### 사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- 웹 콘솔에 액세스하거나 **oc** CLI 툴을 설치했습니다.

#### 프로세스

1. 적용된 구성 Operator를 보려면 다음 명령을 실행합니다.

```
$ oc api-resources -o name | grep config.openshift.io
```

2. 구성할 수 있는 클러스터 리소스를 보려면 다음 명령을 실행합니다.

```
$ oc explain <resource_name>.config.openshift.io
```

3. 클러스터에서 CRD(사용자 정의 리소스 정의) 오브젝트의 구성을 보려면 다음 명령을 실행합니다.

```
$ oc get <resource_name>.config -o yaml
```

4. 클러스터 리소스 구성을 편집하려면 다음 명령을 실행합니다.

```
$ oc edit <resource_name>.config -o yaml
```

## 3장. 지원 요청

### 3.1. 지원 요청

이 문서에 설명된 절차 또는 일반적으로 OpenShift Container Platform에 문제가 발생하는 경우 [Red Hat 고객 포털](#)에 액세스하십시오. 고객 포털에서 다음을 수행할 수 있습니다.

- Red Hat 제품과 관련된 기사 및 솔루션에 대한 Red Hat 지식베이스를 검색하거나 살펴볼 수 있습니다.
- Red Hat 지원에 대한 지원 케이스 제출할 수 있습니다.
- 다른 제품 설명서에 액세스 가능합니다.

클러스터 문제를 식별하려면 [OpenShift Cluster Manager](#)에서 Insights를 사용할 수 있습니다. Insights는 문제에 대한 세부 정보 및 문제 해결 방법에 대한 정보를 제공합니다.

이 문서를 개선하기 위한 제안이 있거나 오류를 발견한 경우 가장 관련 있는 문서 구성 요소에 대한 [Jira 문제](#)를 제출하십시오. 섹션 이름 및 OpenShift Container Platform 버전과 같은 구체적인 정보를 제공합니다.

### 3.2. RED HAT 지식베이스 정보

[Red Hat 지식베이스](#)는 Red Hat의 제품과 기술을 최대한 활용할 수 있도록 풍부한 콘텐츠를 제공합니다. Red Hat 지식베이스는 Red Hat 제품 설치, 설정 및 사용에 대한 기사, 제품 문서 및 동영상으로 구성되어 있습니다. 또한 알려진 문제에 대한 솔루션을 검색할 수 있으며, 간결한 근본 원인 설명 및 해결 단계를 제공합니다.

### 3.3. RED HAT 지식베이스 검색

OpenShift Container Platform 문제가 발생한 경우 초기 검색을 수행하여 솔루션이 이미 Red Hat Knowledgebase 내에 존재하는지 확인할 수 있습니다.

#### 사전 요구 사항

- Red Hat 고객 포털 계정이 있어야 합니다.

#### 프로세스

1. [Red Hat 고객 포털](#)에 로그인합니다.
2. 기본 Red Hat 고객 포털 검색 필드에 다음과 같이 문제와 관련된 키워드 및 문자열을 입력하십시오.
  - OpenShift Container Platform 구성 요소 (**etcd** 등)
  - 관련 절차 (예: **installation** 등)
  - 명시적 실패와 관련된 경고, 오류 메시지 및 기타 출력
3. **Search**를 클릭합니다
4. **OpenShift Container Platform** 제품 필터를 선택합니다.

5. Knowledgebase 콘텐츠 유형 필터를 선택합니다.

### 3.4. 지원 케이스 제출

#### 사전 요구 사항

- OpenShift CLI(**oc**)가 설치되어 있습니다.
- Red Hat 고객 포털 계정이 있어야 합니다.
- [OpenShift Cluster Manager](#) 에 액세스할 수 있습니다.

#### 프로세스

1. [Red Hat 고객 포털](#) 에 로그인하고 **SUPPORT CASES** → **Open a case**를 선택합니다.
2. 문제에 대한 적절한 카테고리(예: **Defect/Bug**), 제품(**OpenShift Container Platform**), 제품 버전(**4.9**, 자동 입력되어 있지 않은 경우)을 선택합니다.
3. 보고되는 문제와 관련이 있을 수 있는 권장 Red Hat 지식베이스 솔루션 목록을 확인합니다. 제안된 문서로 문제가 해결되지 않으면 **Continue**을 클릭합니다.
4. 문제의 증상 및 예상 동작에 대한 자세한 정보와 함께 간결하지만 구체적인 문제 요약을 입력합니다.
5. 보고되는 문제와 관련있는 제안된 Red Hat 지식베이스 솔루션 목록을 확인하십시오. 케이스 작성 과정에서 더 많은 정보를 제공하면 목록이 구체화됩니다. 제안된 문서로 문제가 해결되지 않으면 **Continue**을 클릭합니다.
6. 제시된 계정 정보가 정확한지 확인하고 필요한 경우 적절하게 수정합니다.
7. 자동 입력된 OpenShift Container Platform 클러스터 ID가 올바른지 확인합니다. 그렇지 않은 경우 클러스터 ID를 수동으로 가져옵니다.
  - OpenShift Container Platform 웹 콘솔을 사용하여 클러스터 ID를 수동으로 가져오려면 다음을 수행합니다.
    - a. **Home** → **Dashboards** → **Overview**로 이동합니다.
    - b. **Details** 섹션의 **Cluster ID** 필드에서 값을 찾습니다.
  - 또는 OpenShift Container Platform 웹 콘솔을 통해 새 지원 케이스를 열고 클러스터 ID를 자동으로 입력할 수 있습니다.
    - a. 툴바에서 **(?) Help** → **Open Support Case**로 이동합니다.
    - b. **Cluster ID** 값이 자동으로 입력됩니다.
  - OpenShift CLI (**oc**)를 사용하여 클러스터 ID를 얻으려면 다음 명령을 실행합니다.
 

```
$ oc get clusterversion -o jsonpath='{.items[].spec.clusterID}'
```
8. 프롬프트가 표시되면 다음 질문을 입력한 후 **Continue**를 클릭합니다.
  - 이 문제가 어디에서 발생했습니까? 어떤 시스템 환경을 사용하고 있습니까?

- 이 동작이 언제 발생했습니까? 발생 빈도는 어떻게 됩니까? 반복적으로 발생합니까? 특정 시간에만 발생합니까?
  - 이 문제의 발생 기간 및 비즈니스에 미치는 영향에 대한 정보를 제공해주십시오.
9. 관련 진단 데이터 파일을 업로드하고 **Continue**를 클릭합니다. **oc adm must-gather** 명령을 사용하여 수집된 데이터와 해당 명령으로 수집되지 않은 특정 문제와 관련된 데이터를 제공하는 것이 좋습니다
  10. 관련 케이스 관리 세부 정보를 입력하고 **Continue**를 클릭합니다.
  11. 케이스 세부 정보를 미리보고 **Submit**을 클릭합니다.

### 3.5. 추가 리소스

- 클러스터 문제 식별에 대한 자세한 내용은 [Insights를 사용하여 클러스터 문제 식별](#) 을 참조하십시오.

## 4장. 클러스터에 연결하여 원격 상태 모니터링

### 4.1. 원격 상태 모니터링 정보

OpenShift Container Platform은 클러스터에 대한 Telemetry 및 구성 데이터를 수집하고 Telemeter Client 및 Insights Operator를 사용하여 Red Hat에 보고합니다. Red Hat에 제공되는 데이터는 이 문서에 설명된 장점을 사용할 수 있습니다.

Telemetry 및 Insights Operator를 통해 Red Hat에 데이터를 보고하는 클러스터는 *연결 클러스터 (connected cluster)*라고 합니다.

**Telemetry**는 OpenShift Container Platform Telemeter Client에서 Red Hat으로 전송되는 정보를 설명하는 데 사용하는 용어입니다. 경량 속성은 연결된 클러스터에서 Red Hat으로 전송되어 서브스크립션 관리 자동화를 활성화하고, 클러스터의 상태를 모니터링하며, 지원 및 고객 환경을 개선합니다.

**Insights Operator**는 OpenShift Container Platform 구성 데이터를 수집하여 Red Hat으로 전송합니다. 데이터는 클러스터가 노출될 수 있는 문제에 대한 통찰력을 생성하는 데 사용됩니다. 이러한 통찰력은 [OpenShift Cluster Manager](#)에서 클러스터 관리자에게 전달됩니다.

이 두 프로세스에 대한 자세한 내용은 이 문서에 기재되어 있습니다.

#### Telemetry 및 Insights Operator의 이점

Telemetry 및 Insights Operator는 최종 사용자에게 다음과 같은 이점을 제공합니다.

- **문제 확인 및 해결 방법을 강화** Red Hat은 최종 사용자에게 정상적으로 표시될 수 있는 이벤트를 클러스터 단위로 보다 광범위한 관점에서 확인할 수 있습니다. 일부 문제는 이러한 관점에서 보다 신속하게 확인할 수 있으며 최종 사용자가 지원 케이스를 열거나 [Jira 문제](#)를 제출하지 않고 해결될 수 있습니다.
- **고급 릴리스 관리** OpenShift Container Platform은 **candidate, fast, stable** 릴리스 채널을 제공하므로 이를 통해 업데이트 전략을 선택할 수 있습니다. 릴리스를 **fast** 버전에서 **stable** 버전으로 업그레이드하는 것은 업데이트의 성공률 및 업그레이드 중에 발생하는 이벤트에 따라 달라집니다. 연결된 클러스터에서 제공하는 정보를 통해 Red Hat은 릴리스 품질을 **stable** 채널로 개선하고 **fast** 채널에 있는 문제에 신속하게 대응할 수 있습니다.
- **새로운 기능 및 기능의 우선 순위를 지정** 수집된 데이터는 OpenShift Container Platform의 가장 많이 사용되는 영역에 대한 정보를 제공합니다. 이러한 정보를 통해 Red Hat은 고객에게 가장 큰 영향을 미치는 새로운 기능 및 기능을 개발하는 데 중점을 둘 수 있습니다.
- **간소화된 지원 환경 제공** [Red Hat 고객 포털](#)에서 지원 티켓을 생성할 때 연결된 클러스터의 클러스터 ID를 지정할 수 있습니다. 이를 통해 Red Hat은 연결된 정보를 사용하여 클러스터 고유의 간소화된 지원 환경을 제공할 수 있습니다. 이 문서에서는 향상된 지원 환경에 대한 자세한 정보를 제공합니다.
- **예측 분석** [OpenShift Cluster Manager](#)에서 클러스터에 대해 표시된 Insights는 연결된 클러스터에서 수집된 정보로 활성화됩니다. Red Hat은 OpenShift Container Platform 클러스터가 노출되는 문제를 식별하는 데 도움이 되도록 딥 러닝, 머신 러닝, 인공지능 자동화에 중점을 두고 있습니다.

#### 4.1.1. Telemetry 정보

Telemetry는 업선된 클러스터 모니터링 지표의 일부를 Red Hat으로 보냅니다. Telemeter Client는 4분 30초마다 메트릭 값을 가져와 Red Hat에 데이터를 업로드합니다. 이러한 메트릭에 대한 설명은 이 설명서에서 제공됩니다.

Red Hat은 이러한 데이터 스트림을 사용하여 클러스터를 실시간으로 모니터링하고 필요에 따라 고객에게 영향을 미치는 문제에 대응합니다. 또한 Red Hat은 OpenShift Container Platform 업그레이드를 고객에게 제공하여 서비스 영향을 최소화하고 지속적으로 업그레이드 환경을 개선할 수 있습니다.

이러한 디버깅 정보는 Red Hat 지원 및 엔지니어링 팀에 제공되며, 지원 사례를 통해 보고된 데이터에 액세스하는 것과 동일한 제한 사항이 적용됩니다. Red Hat은 연결된 모든 클러스터 정보를 사용하여 OpenShift Container Platform을 개선하고 사용 편의성을 높입니다.

## 추가 리소스

- 클러스터 업데이트 또는 업그레이드에 대한 자세한 내용은 [OpenShift Container Platform 업데이트 설명서](#)를 참조하십시오.

### 4.1.1.1. Telemetry에서 수집하는 정보

Telemetry에서 수집되는 정보는 다음과 같습니다.

#### 4.1.1.1.1. 시스템 정보

- OpenShift Container Platform 클러스터 버전 및 업데이트 버전 가용성 확인에 사용되는 업데이트 세부 정보와 같은 버전 정보
- 클러스터당 사용 가능한 업데이트 수, 업데이트 진행 정보, 업데이트 진행 정보에 사용되는 채널 및 이미지 리포지터리, 업데이트에 발생하는 오류 수를 포함한 업데이트 정보
- 설치 중 생성된 임의의 고유 식별자
- Red Hat 지원이 클라우드 인프라 수준, 호스트 이름, IP 주소, Kubernetes Pod 이름, 네임스페이스 및 서비스의 노드 구성을 포함하여 고객에게 유용한 지원을 제공하는 데 도움이 되는 구성 세부 정보
- 클러스터 및 해당 조건 및 상태에 설치된 OpenShift Container Platform 프레임워크 구성 요소
- 성능이 저하된 Operator에 대해 "관련 개체"로 나열된 모든 네임스페이스에 대한 이벤트
- 성능 저하 소프트웨어에 대한 정보
- 인증서의 유효성에 대한 정보
- OpenShift Container Platform이 배포된 공급자 플랫폼의 이름 및 데이터 센터 위치

#### 4.1.1.1.2. 크기 조정 정보

- CPU 코어 수 및 각각에 사용된 RAM 용량을 포함한 클러스터, 시스템 유형 및 머신 크기에 대한 정보
- etcd 멤버 수 및 etcd 클러스터에 저장된 오브젝트 수
- 빌드 전략 유형별 애플리케이션 빌드 수

#### 4.1.1.1.3. 사용 정보

- 구성 요소, 기능 및 확장에 대한 사용 정보
- 기술 프리뷰 및 지원되지 않는 구성에 대한 사용량 세부 정보

Telemetry는 사용자 이름 또는 암호와 같은 식별 정보를 수집하지 않습니다. Red Hat은 개인 정보를 수집하지 않습니다. 개인 정보가 의도하지 않게 Red Hat에 수신된 경우 Red Hat은 이러한 정보를 삭제합니다. Telemetry 데이터가 개인 정보를 구성하는 범위까지, Red Hat의 개인정보 보호정책에 대한 자세한 내용은 [Red Hat 개인정보처리방침](#)을 참조하십시오.

## 추가 리소스

- OpenShift Container Platform의 Prometheus에서 Telemetry로 수집하는 속성을 나열하는 방법에 대한 자세한 내용은 [Telemetry에서 수집한 데이터 표시](#)를 참조하십시오.
- Telemetry가 Prometheus에서 수집하는 속성 목록은 [업스트림 cluster-monitoring-operator 소스 코드](#)를 참조하십시오.
- Telemetry는 기본적으로 설치 및 활성화되어 있습니다. 원격 상태 보고서를 사용하지 않는 경우 [원격 상태 보고서 비활성화](#)를 참조하십시오.

### 4.1.2. Insights Operator 정보

Insights Operator는 구성 및 구성 요소 오류 상태를 주기적으로 수집하고 기본적으로 이러한 데이터를 두 시간마다 Red Hat에 보고합니다. 이 정보를 통해 Red Hat은 구성 및 Telemetry를 통해 보고된 것보다 더 깊은 오류 데이터를 평가할 수 있습니다.

OpenShift Container Platform 사용자는 Red Hat Hybrid Cloud Console의 [Insights Advisor](#) 서비스에서 각 클러스터의 보고서를 표시할 수 있습니다. 문제가 확인된 경우 Insights는 추가 세부 정보와 가능한 경우 문제 해결 방법에 대한 단계를 제공합니다.

Insights Operator는 사용자 이름, 암호 또는 인증서와 같은 식별 정보를 수집하지 않습니다. Red Hat Insights 데이터 수집 및 제어에 대한 정보는 [Red Hat Insights Data & Application Security](#)를 참조하십시오.

Red Hat은 연결된 모든 클러스터 정보를 사용하여 다음을 수행합니다.

- Red Hat Hybrid Cloud Console의 [Insights Advisor](#) 서비스에서 잠재적인 클러스터 문제를 식별하고 솔루션 및 예방 조치 제공
- 제품 및 지원팀에 집계되는 중요한 정보를 제공하여 OpenShift Container Platform 개선
- OpenShift Container Platform의 직관성 향상

## 추가 리소스

- Insights Operator는 기본적으로 설치 및 활성화됩니다. 원격 상태 보고서를 사용하지 않는 경우 [원격 상태 보고서 비활성화](#)를 참조하십시오.

#### 4.1.2.1. Insights Operator에 의해 수집되는 정보

Insights Operator에서 수집되는 정보는 다음과 같습니다.

- OpenShift Container Platform 버전 및 환경과 관련된 문제를 확인하는 클러스터 및 해당 구성 요소에 대한 일반 정보
- 설정한 매개변수와 관련된 잘못된 설정 및 문제를 확인하는 클러스터 구성 파일(예: 이미지 레지스트리 구성)
- 클러스터 구성 요소에서 발생하는 오류



- 실행 중인 업데이트의 진행 상태 정보 및 구성 요소의 업그레이드 상태
- Amazon Web Services와 같이 OpenShift Container Platform이 배포된 플랫폼 및 클러스터가 위치한 리전의 세부 정보
- 별도의 TLS(Secure Hash Algorithm) 값으로 변환된 클러스터 워크로드 정보는 Red Hat이 중요한 세부 사항을 분리하지 않고 보안 및 버전 취약점에 대한 워크로드를 평가할 수 있습니다.
- Operator가 문제를 보고하면 **openshift-\*** 및 **kube-\*** 프로젝트의 코어 OpenShift Container Platform Pod에 대한 정보가 수집됩니다. 여기에는 상태, 리소스, 보안 컨텍스트, 불륨 정보 등이 포함됩니다.

#### 추가 리소스

- Insights Operator에 의해 수집된 데이터를 확인하는 방법에 대한 자세한 내용은 [Insights Operator에 의해 수집된 데이터 표시](#)를 참조하십시오.
- Insights Operator 소스 코드는 확인 및 제공할 수 있습니다. Insights Operator에서 수집한 항목 목록은 [Insights Operator 업스트림 프로젝트](#)를 참조하십시오.

### 4.1.3. Telemetry 및 Insights Operator 데이터 흐름 이해

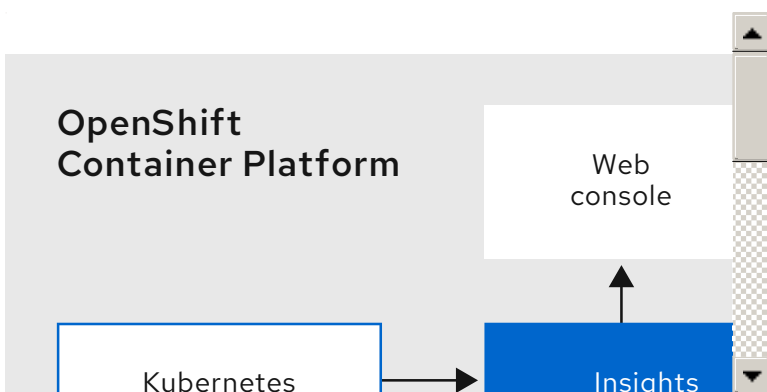
Telemeter Client는 Prometheus API에서 선택한 시계열 데이터를 수집합니다. 시계열 데이터는 처리하기 위해 4분 30초 마다 [api.openshift.com](#)에 업로드됩니다.

Insights Operator는 선택한 데이터를 Kubernetes API 및 Prometheus API에서 아카이브로 수집합니다. 아카이브는 처리를 위해 2시간마다 [OpenShift Cluster Manager](#)에 업로드됩니다. Insights Operator는 또한 [OpenShift Cluster Manager](#)에서 최신 Insights 분석을 다운로드합니다. OpenShift Container Platform 웹 콘솔의 **Overview** 페이지에 있는 **Insights status** 팝업을 설정하는 데 사용됩니다.

Red Hat과의 모든 통신은 TLS(Transport Layer Security) 및 상호 인증서 인증을 사용하여 암호화된 채널을 통해 이루어집니다. 모든 데이터는 전송 및 정지 상태에서 암호화됩니다.

고객 데이터를 처리하는 시스템에 대한 액세스는 다단계 인증 및 엄격한 인증 권한에 의해 제어됩니다. 필요에 따라 액세스 권한이 부여되며 필수 작업으로 제한됩니다.

#### Telemetry 및 Insights Operator 데이터 흐름



#### 추가 리소스

- OpenShift Container Platform [모니터링 스택에 대한 자세한 내용은 모니터링 개요](#)를 참조하십시오.

- 방화벽을 설정하고 Telemetry 및 Insights 엔드 포인트 활성화에 대한 자세한 내용은 [방화벽 구성](#)을 참조하십시오.

#### 4.1.4. 원격 상태 모니터링 데이터 사용 방법에 대한 추가 정보

원격 상태 모니터링을 사용하도록 수집된 정보는 [Telemetry에 의해 수집된 정보](#) 및 [Insights Operator에 의해 수집된 정보](#)에서 참조하십시오.

이 문서의 이전 섹션에 설명되어 있듯이 Red Hat은 지원 및 업그레이드, 성능 또는 구성 최적화, 서비스에 미치는 영향을 최소화, 위협 식별 및 문제 해결, 문제에 대한 대응 및 청구 등의 목적으로 Red Hat 제품 사용에 대한 데이터를 수집합니다.

#### 수집 보안 조치

Red Hat은 Telemetry 및 구성 데이터를 보호하기 위해 설계된 기술 및 제도 상의 조치를 사용합니다.

#### 공유

Red Hat은 사용자 환경을 개선하기 위해 Telemetry 및 Insights Operator에서 수집한 데이터를 내부적으로 공유할 수 있습니다. Red Hat은 Red Hat 제품 사용 및 고객의 사용을 보다 잘 이해할 수 있도록 돕거나 또는 파트너가 협력하여 제품의 지원을 성공적으로 통합하는 데 도움이 되는 집계 양식에서 Telemetry 및 설정 데이터를 공유할 수 있습니다.

#### 제3자

Red Hat은 Telemetry 및 구성 데이터의 수집, 분석 및 스토리지를 지원하기 위해 특정 제3자를 참여할 수 있습니다.

#### 사용자 컨트롤 / Telemetry 및 설정 데이터 수집 활성화 및 비활성화

[원격 상태 보고 비활성화 지침](#)에 따라 OpenShift Container Platform Telemetry 및 Insights Operator를 비활성화할 수 있습니다.

## 4.2. 원격 상태 모니터링으로 수집된 데이터 표시

관리자는 Telemetry 및 Insights Operator에서 수집한 메트릭을 검토할 수 있습니다.

### 4.2.1. Telemetry로 수집한 데이터 표시

Telemetry에서 캡처한 클러스터 및 구성 요소 시계열 데이터를 표시할 수 있습니다.

#### 사전 요구 사항

- OpenShift CLI(**oc**)를 설치합니다.
- **cluster-admin** 역할 또는 **cluster-monitoring-view** 역할 중 하나를 갖는 사용자로 클러스터에 로그인해야 합니다.

#### 프로세스

1. OpenShift Container Platform 클러스터에서 실행되는 Prometheus 서비스의 URL을 찾습니다.

```
$ oc get route prometheus-k8s -n openshift-monitoring -o jsonpath="{.spec.host}"
```

2. URL로 이동합니다.

- 이 쿼리를 **Expression** 입력 상자에 입력하고 **Execute**를 누릅니다.

```
{__name__=~"cluster:usage:.*|count:up0|count:up1|cluster_version|cluster_version_available_updates|cluster_operator_up|cluster_operator_conditions|cluster_version_payload|cluster_installer|cluster_infrastructure_provider|cluster_feature_set|instance:etcd_object_counts:sum|ALERTS|code:apiserver_request_total:rate:sum|cluster:capacity_cpu_cores:sum|cluster:capacity_memory_bytes:sum|cluster:cpu_usage_cores:sum|cluster:memory_usage_bytes:sum|openshift:cpu_usage_cores:sum|openshift:memory_usage_bytes:sum|workload:cpu_usage_cores:sum|workload:memory_usage_bytes:sum|cluster:virt_platform_nodes:sum|cluster:node_instance_type_count:sum|cnv:vmi_status_running:count|node_role_os_version_machine:cpu_capacity_cores:sum|node_role_os_version_machine:cpu_capacity_sockets:sum|subscription_sync_total|csv_succeeded|csv_abnormal|ceph_cluster_total_bytes|ceph_cluster_total_used_raw_bytes|ceph_health_status|job:ceph_osd_metadata:count|job:kube_pv:count|job:ceph_pools_iops:total|job:ceph_pools_iops_bytes:total|job:ceph_versions_running:count|job:noobaa_total_unhealthy_buckets:sum|job:noobaa_bucket_count:sum|job:noobaa_total_object_count:sum|noobaa_accounts_num|noobaa_total_usage|console_url|cluster:network_attachment_definition_instances:max|cluster:network_attachment_definition_enabled_instance_up:max|insightsclient_request_send_total|cam_app_workload_migrations|cluster:apiserver_current_inflight_requests:sum:max_over_time:2m|cluster:telemetry_selected_series:count",alertstate=~"firing"}
```

이 쿼리는 Telemetry가 실행 중인 OpenShift Container Platform 클러스터의 Prometheus 서비스에 대한 실행 요청을 복제하고 Telemetry에 의해 캡처된 전체 시계열 세트를 반환합니다.

#### 4.2.2. Insights Operator에 의해 수집된 데이터의 표시

Insights Operator가 수집한 데이터를 검토할 수 있습니다.

##### 사전 요구 사항

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

##### 프로세스

1. Insights Operator에 대해 현재 실행 중인 Pod의 이름을 검색합니다.

```
$ INSIGHTS_OPERATOR_POD=$(oc get pods --namespace=openshift-insights -o custom-columns=:metadata.name --no-headers --field-selector=status.phase=Running)
```

2. Insights Operator가 수집한 최근 데이터 아카이브를 복사합니다.

```
$ oc cp openshift-insights/$INSIGHTS_OPERATOR_POD:/var/lib/insights-operator ./insights-data
```

최신 Insights Operator 아카이브는 이제 **insights-data** 디렉토리에서 사용할 수 있습니다.

### 4.3. 원격 상태 보고 비활성화

클러스터의 상태 및 사용량 데이터 보고를 비활성화해야 할 수 있습니다.

원격 상태 보고서를 사용하지 않도록 선택하려면 다음을 수행해야 합니다.

1. 원격 상태 보고를 비활성화하려면 **글로벌 클러스터 풀 보안을 수정** 합니다.
2. 수정된 이 풀 시크릿을 사용하도록 **클러스터를 업데이트** 합니다.

### 4.3.1. 원격 상태 보고를 비활성화한 경우의 영향

OpenShift Container Platform에서 고객은 사용 정보를 보고하지 않도록 선택할 수 있습니다. 그러나 연결된 클러스터를 통해 Red Hat은 문제에 보다 신속하게 대응하고 효과적으로 고객에게 지원을 제공할 수 있을 뿐만 아니라 제품 업그레이드가 클러스터에 미치는 영향을 보다 잘 이해할 수 있습니다. 또한 연결된 클러스터는 서브스크립션 및 인타이틀먼트 프로세스를 단순화하고 OpenShift Cluster Manager 서비스에서 클러스터 및 서브스크립션 상태에 대한 개요를 제공할 수 있습니다.

Red Hat은 프로덕션 클러스터에서 이 기능을 비활성화해야 하는 경우에도 프로덕션 이전 환경 및 테스트 클러스터에서 상태 및 사용량 보고를 활성화된 상태로 둘 것을 강력히 권장합니다. 이를 통해 Red Hat은 고객 환경에서 OpenShift Container Platform의 제품 검증에 참여하고 제품 관련 문제에 신속하게 대응할 수 있습니다.

연결된 클러스터를 비활성화한 경우 다음과 같은 결과가 나타날 수 있습니다.

- Red Hat은 지원 케이스를 열지 않으면 제품 업그레이드 상태나 클러스터 상태를 모니터링할 수 없습니다.
- Red Hat은 구성 데이터를 사용하여 고객 지원 케이스를 보다 효과적으로 분류하고 고객이 중요하게 생각하는 구성을 식별할 수 없습니다.
- OpenShift Cluster Manager는 상태 및 사용량 정보를 포함하여 클러스터에 대한 데이터를 표시하지 않습니다.
- 사용 정보의 자동 보고 기능 없이 console.redhat.com을 통해 서브스크립션 인증 정보를 수동으로 입력해야 합니다.

네트워크가 제한된 환경에서 프록시를 올바르게 구성하여 Telemetry 및 Insights 데이터를 계속 보고할 수 있습니다.

### 4.3.2. 원격 상태 보고를 사용하지 않도록 글로벌 클러스터 풀 시크릿 수정

기존 글로벌 클러스터 풀 시크릿을 수정하여 원격 상태 보고를 비활성화할 수 있습니다. 이를 통해 Telemetry와 Insights Operator가 모두 비활성화됩니다.

#### 사전 요구 사항

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

#### 절차

1. 로컬 파일 시스템에 글로벌 클러스터 풀 시크릿을 다운로드합니다.

```
$ oc extract secret/pull-secret -n openshift-config --to=.
```

2. 텍스트 편집기에서 다운로드한 **.dockerconfigjson** 파일을 편집합니다.
3. 다음과 같이 **cloud.openshift.com** JSON 항목을 제거합니다.

```
"cloud.openshift.com":{"auth":"<hash>","email":"<email_address>"}
```

4. 파일을 저장합니다.

이러한 변경된 풀 시크릿을 사용하도록 클러스터를 업데이트할 수 있습니다.

### 4.3.3. 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.

#### 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.

## 4.4. INSIGHTS를 사용하여 클러스터의 문제 식별

Insights는 Insights Operator가 전송하는 데이터를 반복적으로 분석합니다. OpenShift Container Platform 사용자는 Red Hat Hybrid Cloud Console의 [Insights Advisor](#) 서비스에 보고서를 표시할 수 있습니다.

### 4.4.1. OpenShift Container Platform용 Red Hat Insights Advisor 정보

Insights Advisor를 사용하여 OpenShift Container Platform 클러스터의 상태를 평가하고 모니터링할 수 있습니다. 개별 클러스터 또는 전체 인프라에 대한 우려 여부에 관계없이 서비스 가용성, 내결함성, 성능 또는 보안에 영향을 줄 수 있는 문제에 대한 노출을 알고 있어야 합니다.

Insights는 Insights Operator가 *권장 사항* 데이터베이스를 사용하여 전송하는 데이터를 반복적으로 분석합니다. 이는 OpenShift Container Platform 클러스터를 위험하게 유지할 수 있는 조건 세트입니다. 그러면 다음 작업을 수행할 수 있는 Red Hat Hybrid Cloud Console의 Insights Advisor 서비스에 데이터가 업로드됩니다.

- 특정 권장 사항의 영향을 받는 클러스터를 참조하십시오.
- 강력한 필터링 기능을 사용하여 결과를 해당 권장 사항으로 조정합니다.
- 개별 권장 사항에 대해 자세히 알아보고 현재 위험을 자세히 알아보고 개별 클러스터에 맞는 해상도를 확보하십시오.
- 다른 이해 관계자와 결과를 공유하십시오.

### 4.4.2. Insights Advisor 권장 사항 이해

Insights Advisor는 클러스터의 서비스 가용성, 내결함성, 성능 또는 보안에 부정적인 영향을 미칠 수 있는 다양한 클러스터 상태 및 구성 요소에 대한 정보를 번들로 제공합니다. 이 정보 세트는 Insights Advisor에서 권장 사항이라고 하며 다음 정보가 포함되어 있습니다.

- **이름:** 권장 사항에 대한 간결한 설명입니다.
- **추가된:** Insights Advisor 아카이브에 권장 사항이 게시될 때
- **카테고리:** 문제가 서비스 가용성, 내결함성, 성능 또는 보안에 부정적인 영향을 미칠 가능성이 있는지 여부입니다.
- **총 위험:** 조건이 인프라에 부정적인 영향을 미칠 *가능성과 이러한 상황이 발생할 경우* 운영에 부정적인 영향을 미칠 가능성을 기반으로 얻은 값입니다.
- **클러스터:** 권장 사항이 감지되는 클러스터 목록
- **설명:** 클러스터에 미치는 영향을 포함하여 문제에 대한 간략한 개요
- **관련 주제 링크:** 문제에 대한 Red Hat의 추가 정보

### 4.4.3. 클러스터와 관련된 잠재적인 문제 표시

이 섹션에서는 [OpenShift Cluster Manager](#)의 [Insights Advisor](#)에 Insights 보고서를 표시하는 방법을 설명합니다.

Insights는 반복적으로 클러스터를 분석하여 최신 결과를 표시합니다. 예를 들어 문제를 해결하거나 새로운 문제가 발견된 경우 이러한 결과가 변경될 수 있습니다.

### 사전 요구 사항

- 클러스터가 [OpenShift Cluster Manager](#) 에 등록되어 있습니다.
- 원격 상태 보고가 활성화되어 있습니다 (기본값).
- [OpenShift Cluster Manager](#) 에 로그인되어 있습니다.

### 프로세스

1. [OpenShift Cluster Manager](#) 에서 **Advisor → Recommendations** 로 이동합니다.  
결과에 따라 Insights Advisor는 다음 중 하나를 표시합니다.
  - Insights에서 문제를 식별하지 않은 경우 일치하는 권장 사항을 찾을 수 없습니다.
  - Insights에서 감지한 문제 목록은 위험(낮음, 중간, 중요 및 심각)별로 그룹화되어 있습니다.
  - Insights가 클러스터를 아직 분석하지 않은 경우 클러스터가 아직 없습니다. 클러스터가 설치, 등록 및 인터넷에 연결된 직후 분석이 시작됩니다.
2. 문제가 표시되면 항목 앞의 > 아이콘을 클릭하여 자세한 내용을 확인합니다.  
문제에 따라 세부 정보에는 문제에 대한 Red Hat의 자세한 정보 링크가 포함되어 있을 수 있습니다.

#### 4.4.4. 모든 Insights Advisor 권장 사항 표시

권장 사항 보기는 기본적으로 클러스터에서 탐지된 권장 사항만 표시합니다. 그러나 Advisor 아카이브에서 모든 권장 사항을 볼 수 있습니다.

### 사전 요구 사항

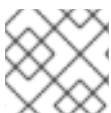
- 원격 상태 보고가 활성화되어 있습니다 (기본값).
- 클러스터는 Red Hat Hybrid Cloud Console에 [등록되어](#) 있습니다.
- [OpenShift Cluster Manager](#) 에 로그인되어 있습니다.

### 절차

1. [OpenShift Cluster Manager](#) 에서 **Advisor → Recommendations** 로 이동합니다.
2. 클러스터 영향 및 상태 필터 옆에 있는 X 아이콘을 클릭합니다.  
이제 클러스터에 대한 모든 잠재적 권장 사항을 확인할 수 있습니다.

#### 4.4.5. Insights Advisor 권장 사항 비활성화

더 이상 보고서에 표시되지 않도록 클러스터에 영향을 미치는 특정 권장 사항을 비활성화할 수 있습니다. 단일 클러스터 또는 모든 클러스터에 대한 권장 사항을 비활성화할 수 있습니다.



### 참고


모든 클러스터에 대한 권장 사항 비활성화는 향후 클러스터에도 적용됩니다.

### 사전 요구 사항

- 원격 상태 보고가 활성화되어 있습니다 (기본값).
- 클러스터가 [OpenShift Cluster Manager](#) 에 등록되어 있습니다.
- [OpenShift Cluster Manager](#) 에 로그인되어 있습니다.

절차

1. [OpenShift Cluster Manager](#) 에서 **Advisor → Recommendations** 로 이동합니다.
2. 비활성화할 권장 사항 이름을 클릭합니다. 단일 권장 사항 페이지로 이동합니다.
3. 단일 클러스터에 대한 권장 사항을 비활성화하려면 다음을 수행합니다.

a. 해당 클러스터의 옵션 메뉴  를 클릭한 다음 클러스터에 대한 비활성화 권장 사항을 클릭합니다.

b. 확인 노트를 입력하고 저장을 클릭합니다.

4. 모든 클러스터에 대한 권장 사항을 비활성화하려면 다음을 수행합니다.

a. 작업 → 권장 사항 비활성화를 클릭합니다.

b. 확인 노트를 입력하고 저장을 클릭합니다.

### 4.4.6. 이전에 비활성화된 Insights Advisor 권장 사항 활성화


모든 클러스터에 대해 권장 사항이 비활성화되면 더 이상 Insights Advisor의 권장 사항이 표시되지 않습니다. 이 동작을 변경할 수 있습니다.

사전 요구 사항

- 원격 상태 보고가 활성화되어 있습니다 (기본값).
- 클러스터가 [OpenShift Cluster Manager](#) 에 등록되어 있습니다.
- [OpenShift Cluster Manager](#) 에 로그인되어 있습니다.

절차

1. [OpenShift Cluster Manager](#) 에서 **Advisor → Recommendations** 로 이동합니다.
2. 상태 → Disabled 로 권장 사항을 필터링합니다.
3. 활성화할 권장 사항을 찾습니다.

4. 옵션 메뉴  를 클릭한 다음 권장 사항 사용을 클릭합니다.

### 4.4.7. 웹 콘솔에 Insights 상태 표시

Insights는 클러스터를 반복적으로 분석하고 OpenShift Container Platform 웹 콘솔에서 확인된 잠재적 클러스터 문제의 상태를 표시할 수 있습니다. 이 상태에는 다양한 카테고리의 문제 수가 표시되며 자세한 내용은 [OpenShift Cluster Manager](#) 의 보고서에 대한 링크입니다.



## 사전 요구 사항

- 클러스터는 [OpenShift Cluster Manager](#) 에 등록되어 있습니다.
- 원격 상태 보고가 활성화되어 있습니다 (기본값).
- OpenShift Container Platform 웹 콘솔에 로그인되어 있습니다.

## 절차

1. OpenShift Container Platform 웹 콘솔에서 **홈** → **개요**로 이동합니다.
2. **상태** 카드에서 **Insights**를 클릭합니다.  
팝업 창에 잠재적인 문제가 위험별로 그룹화되어 나열됩니다. 개별 카테고리를 클릭하거나 **Insights Advisor**의 모든 권장 사항 보기를 클릭하여 자세한 내용을 표시합니다.

## 4.5. INSIGHTS OPERATOR 사용

Insights Operator는 구성 및 구성 요소 오류 상태를 주기적으로 수집하고 기본적으로 이러한 데이터를 두 시간마다 Red Hat에 보고합니다. 이 정보를 통해 Red Hat은 구성 및 Telemetry를 통해 보고된 것보다 더 깊은 오류 데이터를 평가할 수 있습니다. OpenShift Container Platform 사용자는 Red Hat Hybrid Cloud Console의 [Insights Advisor](#) 서비스에 보고서를 표시할 수 있습니다.

### 추가 리소스

- Insights Operator는 기본적으로 설치 및 활성화됩니다. 원격 상태 보고서를 사용하지 않는 경우 [원격 상태 보고서 비활성화](#)를 참조하십시오.
- Insights Advisor를 사용하여 클러스터 문제 식별에 대한 자세한 내용은 [Insights를 사용하여 클러스터 문제 식별](#)을 참조하십시오.

### 4.5.1. Insights Operator 아카이브 다운로드

Insights Operator는 클러스터의 **openshift-insights** 네임스페이스에 있는 아카이브에 수집된 데이터를 저장합니다. Insights Operator에서 수집한 데이터를 다운로드하고 확인할 수 있습니다.

## 사전 요구 사항

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

## 절차

1. Insights Operator에 대해 실행 중인 Pod의 이름을 찾습니다.

```
$ oc get pods --namespace=openshift-insights -o custom-columns=:metadata.name --no-headers --field-selector=status.phase=Running
```

2. Insights Operator가 수집한 최근 데이터 아카이브를 복사합니다.

```
$ oc cp openshift-insights/<insights_operator_pod_name>:/var/lib/insights-operator ./insights-data 1
```

1 &lt;insights\_operator\_pod\_name>을 이전 명령의 Pod 이름 출력으로 바꿉니다.

최신 Insights Operator 아카이브는 이제 **insights-data** 디렉토리에서 사용할 수 있습니다.

## 4.5.2. Insights Operator 수집 기간 보기

Insights Operator가 아카이브에 포함된 정보를 수집하는 데 걸리는 시간을 볼 수 있습니다. 이를 통해 Insights Operator 리소스 사용과 Insights Advisor의 문제를 이해하는 데 도움이 됩니다.

### 사전 요구 사항

- Insights Operator 아카이브의 최근 사본입니다.

### 절차

1. 아카이브에서 **/insights-operator/gathers.json** 을 엽니다.  
파일에는 Insights Operator 수집 작업 목록이 포함되어 있습니다.

```
{
  "name": "clusterconfig/authentication",
  "duration_in_ms": 730, ①
  "records_count": 1,
  "errors": null,
  "panic": null
}
```

- ① **duration\_in\_ms** 는 각 수집 작업의 시간(밀리초)입니다.

2. 비정상에 대한 각 수집 작업을 검사합니다.

## 4.6. 네트워크가 제한된 환경에서 원격 상태 보고 사용

Insights Operator 아카이브를 수동으로 수집 및 업로드하여 제한된 네트워크에서 문제를 진단할 수 있습니다.

제한된 네트워크에서 Insights Operator를 사용하려면 다음을 수행해야 합니다.

- Insights Operator 아카이브 사본을 만듭니다.
- Insights Operator 아카이브를 [console.redhat.com](https://console.redhat.com)에 업로드합니다.

또한 업로드하기 전에 Insights Operator 데이터를 **난독 처리** 하도록 선택할 수 있습니다.

### 4.6.1. Insights Operator 수집 작업 실행

Insights Operator 아카이브를 생성하려면 수집 작업을 실행해야 합니다.

### 사전 요구 사항

- **cluster-admin**으로 OpenShift Container Platform 클러스터에 로그인되어 있습니다.

### 절차

1. 이 템플릿을 사용하여 **gather-job.yaml**이라는 파일을 생성합니다.

```
apiVersion: batch/v1
kind: Job
metadata:
  name: insights-operator-job
  annotations:
    config.openshift.io/inject-proxy: insights-operator
spec:
  backoffLimit: 6
  ttlSecondsAfterFinished: 600
  template:
    spec:
      restartPolicy: OnFailure
      serviceAccountName: operator
      nodeSelector:
        beta.kubernetes.io/os: linux
        node-role.kubernetes.io/master: ""
      tolerations:
        - effect: NoSchedule
          key: node-role.kubernetes.io/master
          operator: Exists
        - effect: NoExecute
          key: node.kubernetes.io/unreachable
          operator: Exists
          tolerationSeconds: 900
        - effect: NoExecute
          key: node.kubernetes.io/not-ready
          operator: Exists
          tolerationSeconds: 900
      volumes:
        - name: snapshots
          emptyDir: {}
        - name: service-ca-bundle
          configMap:
            name: service-ca-bundle
            optional: true
      initContainers:
        - name: insights-operator
          image: quay.io/openshift/origin-insights-operator:latest
          terminationMessagePolicy: FallbackToLogsOnError
          volumeMounts:
            - name: snapshots
              mountPath: /var/lib/insights-operator
            - name: service-ca-bundle
              mountPath: /var/run/configmaps/service-ca-bundle
              readOnly: true
      ports:
        - containerPort: 8443
          name: https
      resources:
        requests:
          cpu: 10m
          memory: 70Mi
      args:
        - gather
        - -v=4
        - --config=/etc/insights-operator/server.yaml
```

```
containers:
  - name: sleepy
    image: quay.io/openshift/origin-base:latest
    args:
      - /bin/sh
      - -c
      - sleep 10m
    volumeMounts: [{name: snapshots, mountPath: /var/lib/insights-operator}]
```

2. **insights-operator** 이미지 버전을 복사합니다.

```
$ oc get -n openshift-insights deployment insights-operator -o yaml
```

3. **gather-job.yaml**에 이미지 버전을 붙여 넣습니다.

```
initContainers:
  - name: insights-operator
    image: <your_insights_operator_image_version>
    terminationMessagePolicy: FallbackToLogsOnError
    volumeMounts:
```

4. 수집 작업을 생성합니다.

```
$ oc apply -n openshift-insights -f gather-job.yaml
```

5. 작업 Pod의 이름을 찾습니다.

```
$ oc describe -n openshift-insights job/insights-operator-job
```

#### 출력 예

```
Events:
  Type Reason      Age From          Message
  ---- -
  Normal SuccessfulCreate 7m18s job-controller Created pod: insights-operator-
  job-<your_job>
```

여기서 **insights-operator-job-<your\_job>**은 Pod의 이름입니다.

6. 작업이 완료되었는지 확인합니다.

```
$ oc logs -n openshift-insights insights-operator-job-<your_job> insights-operator
```

#### 출력 예

```
10407 11:55:38.192084 [1 diskrecorder.go:34] Wrote 108 records to disk in 33ms
```

7. 생성된 아카이브를 저장합니다.

```
$ oc cp openshift-insights/insights-operator-job-<your_job>:/var/lib/insights-operator
./insights-data
```

8. 작업을 정리합니다.

```
$ oc delete -n openshift-insights job insights-operator-job
```

#### 4.6.2. Insights Operator 아카이브 업로드

Insights Operator 아카이브를 [console.redhat.com](https://console.redhat.com)에 수동으로 업로드하여 잠재적인 문제를 진단할 수 있습니다.

##### 사전 요구 사항

- **cluster-admin**으로 OpenShift Container Platform 클러스터에 로그인되어 있습니다.
- 무제한 인터넷 액세스가 가능한 워크스테이션이 있습니다.
- Insights Operator 아카이브 사본을 생성했습니다.

##### 절차

1. **dockerconfig.json** 파일을 다운로드합니다.

```
$ oc extract secret/pull-secret -n openshift-config --to=.
```

2. **dockerconfig.json** 파일에서 **"cloud.openshift.com"** **"auth"** 토큰을 복사합니다.

```
{
  "auths": {
    "cloud.openshift.com": {
      "auth": "<your_token>",
      "email": "asd@redhat.com"
    }
  }
}
```

3. [console.redhat.com](https://console.redhat.com)에 아카이브를 업로드합니다.

```
$ curl -v -H "User-Agent: insights-operator/one10time200gather184a34f6a168926d93c330
cluster/<cluster_id>" -H "Authorization: Bearer <your_token>" -F
"upload=@<path_to_archive>; type=application/vnd.redhat.openshift.periodic-tar"
https://console.redhat.com/api/ingress/v1/upload
```

여기서 **<cluster\_id>**는 클러스터 ID이며 **<your\_token>**은 풀 시크릿의 토큰이며 **<path\_to\_archive>**는 Insights Operator 아카이브의 경로입니다.

작업이 성공하면 명령은 **"request\_id"** 및 **"account\_number"**를 반환합니다.

##### 출력 예


```
* Connection #0 to host console.redhat.com left intact
{"request_id":"393a7cf1093e434ea8dd4ab3eb28884c","upload":
{"account_number":"6274079"}}%
```

##### 검증 절차

1. <https://console.redhat.com/openshift>로 로그인합니다.
2. 왼쪽 창에서 **Clusters** 메뉴를 클릭합니다.
3. 클러스터 이름을 클릭하여 클러스터의 세부 사항을 표시합니다.
4. 클러스터의 **Insights** 권고 탭을 엽니다.  
업로드에 성공하면 탭에 다음 중 하나가 표시됩니다.
  - 클러스터에서 모든 권장 사항을 통과: Insights 권고에서 문제가 발견되지 않은 경우 표시됩니다.
  - Insights 권고에서 감지한 문제 목록으로 위험 (낮음, 중간, 중요 및 심각)에 따라 우선 순위가 지정됩니다.

### 4.6.3. Insights Operator 데이터 난독 처리

난독성을 활성화하여 Insights Operator가 [console.redhat.com](https://console.redhat.com)에 전송하는 중요하고 식별 가능한 IPv4 주소 및 클러스터 기본 도메인을 마스킹할 수 있습니다.



**주의**

이 기능을 사용할 수 있지만 Red Hat은 더욱 효과적인 지원 환경을 위해 난독 처리를 비활성할 것을 권장합니다.

난독 처리는 클러스터 IPv4 주소에 고유하지 않은 값을 할당하고, 메모리에 유지되는 변환 테이블을 사용하여 데이터를 [console.redhat.com](https://console.redhat.com)에 업로드하기 전에 Insights Operator 아카이브 전체의 난독 처리된 버전으로 IP 주소를 변경합니다.


클러스터 기본 도메인의 경우 난독 처리는 기본 도메인을 하드 코딩된 하위 문자열로 변경합니다. 예를 들어 **cluster-api.openshift.example.com**은 **cluster-api.<CLUSTER\_BASE\_DOMAIN>**이 됩니다.


다음 절차에서는 **openshift-config** 네임스페이스의 **support** 시크릿을 사용하여 난독화를 활성화합니다.

#### 사전 요구 사항

- **cluster-admin**으로 OpenShift Container Platform 웹 콘솔에 로그인되어 있습니다.

#### 절차

1. 워크로드 → 시크릿으로 이동합니다.
2. **openshift-config** 프로젝트를 선택합니다.
3. 이름으로 검색 필드를 사용하여 **지원** 시크릿을 검색합니다. 존재하지 않는 경우 **생성** → **키/값** 시크릿을 클릭하여 생성합니다.
4. 옵션 메뉴를  클릭한 다음 **시크릿 편집**을 클릭합니다.

5. 키/값 추가 클릭
6. 값이 **true**인 **enableGlobalObfuscation**이라는 키를 생성하고 **저장**을 클릭합니다.
7. **워크로드** → **Pod**로 이동합니다.
8. **openshift-insights** 프로젝트를 선택합니다.
9. **insights-operator** pod를 찾습니다.
10. **insights-operator** pod를 다시 시작하려면 **옵션** 메뉴  를 클릭한 다음 **Pod 삭제**를 클릭합니다.

#### 검증

1. **워크로드** → **시크릿**으로 이동합니다.
2. **openshift-insights** 프로젝트를 선택합니다.
3. **이름**으로 검색 필드를 사용하여 **난독화-translation-table** 시크릿을 검색합니다.

**obfuscation-translation-table** 시크릿이 있으면 난독이 활성화되고 작동합니다.

또는 Insights Operator 아카이브의 **/insights-operator/gathers.json** 값을 **"is\_global\_obfuscation\_enabled": true**로 검사할 수 있습니다.

#### 추가 리소스

- Insights Operator 아카이브 다운로드 방법에 대한 자세한 내용은 [Insights Operator가 수집한 데이터 표시](#)를 참조하십시오.

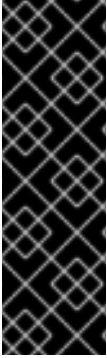
## 4.7. INSIGHTS OPERATOR를 사용하여 간단한 콘텐츠 액세스 인증서 가져오기

Insights Operator는 [Red Hat Hybrid Cloud Console](#) 에서 RHEL SCA(Simple Content Access) 인증서를 가져올 수 있습니다. SCA는 자격 톨링의 동작을 간소화하는 Red Hat 서브스크립션 톨의 기능입니다. 서브스크립션 톨 구성의 복잡성 없이 Red Hat 서브스크립션에서 제공하는 콘텐츠를 사용하는 것이 더 간편합니다. 인증서를 가져오면 **openshift-config-managed** 네임스페이스의 **etc-pki-entitlement** 시크릿에 저장됩니다.

Insights Operator는 기본적으로 8시간마다 SCA 인증서를 가져오지만 **openshift-config** 네임스페이스에서 **support** 시크릿을 사용하여 구성하거나 비활성화할 수 있습니다.

OpenShift Container Platform 4.9에서 이 기능은 기술 프리뷰로, **TechPreviewNoUpgrade** 기능 세트를 사용하여 활성화해야 합니다. 자세한 내용은 [FeatureGates를 사용하여 OpenShift Container Platform 기능 활성화](#)를 참조하십시오.

Simple Content Access 인증서에 대한 자세한 내용은 Red Hat Knowledgebase의 [Simple Content Access](#) 문서를 참조하십시오.



**중요**

**InsightsOperatorPullingSCA** 는 기술 프리뷰 기능 전용입니다. 기술 프리뷰 기능은 Red Hat 프로덕션 서비스 수준 계약(SLA)에서 지원되지 않으며 기능적으로 완전하지 않을 수 있습니다. 따라서 프로덕션 환경에서 사용하는 것은 권장하지 않습니다. 이러한 기능을 사용하면 향후 제품 기능을 조기에 이용할 수 있어 개발 과정에서 고객이 기능을 테스트하고 피드백을 제공할 수 있습니다.

Red Hat 기술 프리뷰 기능의 지원 범위에 대한 자세한 내용은 [기술 프리뷰 기능 지원 범위](#)를 참조하십시오.

**4.7.1. 간단한 콘텐츠 액세스 가져오기 간격 구성**


**openshift-config** 네임스페이스의 **support** 시크릿을 사용하여 Insights Operator가 RHEL SCA(Simple Content Access) 인증서를 가져오는 빈도를 구성할 수 있습니다. 인증서 가져오기는 일반적으로 8시간마다 발생하지만, Red Hat 서브스크립션 관리에서 SCA 구성을 업데이트하는 경우 이 간격을 단축할 수 있습니다.

다음 절차에서는 가져오기 간격을 1시간으로 업데이트하는 방법을 설명합니다.

**사전 요구 사항**

- **cluster-admin**으로 OpenShift Container Platform 웹 콘솔에 로그인되어 있습니다.


**절차**

1. 워크로드 → 시크릿으로 이동합니다.
2. **openshift-config** 프로젝트를 선택합니다.
3. 이름으로 검색 필드를 사용하여 **지원** 시크릿을 검색합니다. 존재하지 않는 경우 **생성** → **키/값** 시크릿을 클릭하여 생성합니다.
4. 옵션 메뉴를  클릭한 다음 **시크릿 편집**을 클릭합니다.
5. **키/값** 추가 클릭
6. 값이 **1h**인 **ocmlInterval**이라는 키를 만들고 **저장**을 클릭합니다.



**참고**

간격 **1h**는 60 분 동안 **60m**로 입력할 수도 있습니다.

7. 워크로드 → **Pod**로 이동합니다.
8. **openshift-insights** 프로젝트를 선택합니다.
9. **insights-operator** pod를 찾습니다.
10. **insights-operator** pod를 다시 시작하려면 옵션 메뉴  를 클릭한 다음 **Pod 삭제**를 클릭합니다.



## 4.7.2. 간단한 콘텐츠 액세스 가져오기 비활성화

**openshift-config** 네임스페이스에서 **support** 보안을 사용하여 RHEL SCA(Simple Content Access) 인증서 가져오기를 비활성화할 수 있습니다.

### 사전 요구 사항

- **cluster-admin**으로 OpenShift Container Platform 웹 콘솔에 로그인되어 있습니다.

### 절차

1. 워크로드 → 시크릿으로 이동합니다.
2. **openshift-config** 프로젝트를 선택합니다.
3. 이름으로 검색 필드를 사용하여 **지원** 시크릿을 검색합니다. 존재하지 않는 경우 **생성** → **키/값** 시크릿을 클릭하여 생성합니다.
4. 옵션 메뉴를  클릭한 다음 **시크릿 편집**을 클릭합니다.
5. 키/값 추가 클릭
6. 값이 **true**인 **ocmPullDisabled**라는 키를 생성하고 **저장**을 클릭합니다.
7. 워크로드 → **Pod**로 이동합니다.
8. **openshift-insights** 프로젝트를 선택합니다.
9. **insights-operator** pod를 찾습니다.
10. **insights-operator** pod를 다시 시작하려면 옵션 메뉴  를 클릭한 다음 **Pod 삭제**를 클릭합니다.

## 5장. 클러스터에 대한 데이터 수집

지원 사례를 여는 경우 클러스터에 대한 디버깅 정보를 Red Hat 지원에 제공하면 도움이 됩니다.

다음은 제공하는 것이 좋습니다.

- **oc adm must-gather** 명령을 사용하여 수집된 데이터
- 고유한 클러스터 ID

### 5.1. MUST-GATHER 툴 정보

**oc adm must-gather** CLI 명령은 다음을 포함하여 문제를 디버깅하는 데 필요할 가능성이 높은 클러스터에서 정보를 수집합니다.

- 리소스 정의
- 서비스 로그

기본적으로 **oc adm must-gather** 명령은 기본 플러그인 이미지를 사용하고 **./must-gather.local**에 씁니다.

또는 다음 섹션에 설명된 대로 적절한 인수를 사용하여 명령을 실행하여 특정 정보를 수집할 수 있습니다.

- 하나 이상의 특정 기능과 관련된 데이터를 수집하려면 다음 섹션에 나열된 대로 이미지에 **--image** 인수를 사용합니다. 예를 들면 다음과 같습니다.

```
$ oc adm must-gather --image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8:v4.9.0
```

- 감사 로그를 수집하려면 다음 섹션에 설명된 대로 **--/usr/bin/gather\_audit\_logs** 인수를 사용합니다. 예를 들면 다음과 같습니다.

```
$ oc adm must-gather --/usr/bin/gather_audit_logs
```



#### 참고

감사 로그는 파일의 크기를 줄이기 위해 기본 정보 세트의 일부로 수집되지 않습니다.

**oc adm must-gather** 를 실행하면 클러스터의 새 프로젝트에 임의의 이름이 있는 새 Pod가 생성됩니다. 해당 Pod에 대한 데이터가 수집되어 **must-gather.local**로 시작하는 새 디렉터리에 저장됩니다. 이 디렉터리는 현재 작업 중인 디렉터리에 생성되어 있습니다.

예를 들면 다음과 같습니다.

| NAMESPACE                   | NAME              | READY | STATUS  | RESTARTS | AGE |
|-----------------------------|-------------------|-------|---------|----------|-----|
| ...                         |                   |       |         |          |     |
| openshift-must-gather-5drcj | must-gather-bklx4 | 2/2   | Running | 0        | 72s |
| openshift-must-gather-5drcj | must-gather-s8sdh | 2/2   | Running | 0        | 72s |
| ...                         |                   |       |         |          |     |

### 5.1.1. Red Hat 지원을 위한 클러스터에 대한 데이터 수집

**oc adm must-gather** CLI 명령을 사용하여 클러스터에 대한 디버깅 정보를 수집할 수 있습니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift Container Platform CLI(**oc**)가 설치되어 있어야 합니다.

절차

1. **must-gather** 데이터를 저장하려는 디렉터리로 이동합니다.



#### 참고

클러스터에서 제한된 네트워크를 사용하는 경우 추가 단계를 수행해야 합니다. 미리 레지스트리에 신뢰할 수 있는 CA가 있는 경우 먼저 신뢰할 수 있는 CA를 클러스터에 추가해야 합니다. 제한된 네트워크의 모든 클러스터에 대해 기본 **must-gather** 이미지를 이미지 스트림으로 가져와야 합니다.

```
$ oc import-image is/must-gather -n openshift
```

2. **oc adm must-gather** 명령을 실행합니다.

```
$ oc adm must-gather
```



#### 참고

이 명령은 기본적으로 임의의 컨트롤 플레인 노드를 선택하기 때문에 **NotReady** 및 **SchedulingDisabled** 상태에 있는 컨트롤 플레인 노드에 Pod를 예약할 수 없습니다.

- a. 예를 들어 클러스터에서 Pod를 예약할 수 없는 경우 **oc adm inspect** 명령을 사용하여 특정 리소스에 대한 정보를 수집합니다.



#### 참고

권장되는 리소스를 얻으려면 Red Hat 지원에 문의하십시오.

3. 작업 디렉터리에서 생성된 **must-gather** 디렉터리에서 압축 파일을 만듭니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar cvaf must-gather.tar.gz must-gather.local.5421342344627712289/ 1
```

- 1 **must-gather-local.5421342344627712289/**를 실제 디렉터리 이름으로 교체하십시오.

4. 압축 파일을 [Red Hat 고객 포털](#)의 지원 케이스에 첨부합니다.

### 5.1.2. 특정 기능에 대한 데이터 수집

**oc adm must-gather** CLI 명령을 **--image** 또는 **--image-stream** 인수와 함께 사용하여 특정 기능에 대한 디버깅 정보를 수집할 수 있습니다. **must-gather** 툴은 여러 이미지를 지원하므로 단일 명령을 실행하여 둘 이상의 기능에 대한 데이터를 수집할 수 있습니다.

표 5.1. 지원되는 must-gather 이미지

| 이미지  | 목적  |
|--|---|
| <b>registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8:v4.9.7</b>                 | OpenShift Virtualization의 데이터 수집.           |
| <b>registry.redhat.io/openshift-serverless-1/svls-must-gather-rhel8</b>                                | OpenShift Serverless의 데이터 수집.               |
| <b>registry.redhat.io/openshift-service-mesh/istio-must-gather-rhel8:2.3</b>                           | Red Hat OpenShift Service Mesh의 데이터 수집      |
| <b>registry.redhat.io/rhmtc/openshift-migration-must-gather-rhel8:v1.7</b>                             | Migration Toolkit for Containers의 데이터 수집    |
| <b>registry.redhat.io/odf4/ocs-must-gather-rhel8:v4.9</b>  | Red Hat OpenShift Container Storage의 데이터 수집 |
| <b>registry.redhat.io/openshift-logging/cluster-logging-rhel8-operator</b>                             | OpenShift Logging의 데이터 수집.                  |
| <b>registry.redhat.io/openshift4/ose-local-storage-mustgather-rhel8:v&lt;installed_version_LSO&gt;</b> | Local Storage Operator의 데이터 수집              |
| <b>registry.redhat.io/openshift-sandboxed-containers-tech-preview/osc-must-gather-rhel8:1.1.0</b>      | OpenShift 샌드박스 컨테이너의 데이터 수집                 |



참고

특정 기능 데이터 외에도 기본 **must-gather** 데이터를 수집하려면 **--image-stream=openshift/must-gather** 인수를 추가하십시오.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift Container Platform CLI(**oc**)가 설치되어 있어야 합니다.

프로세스

1. **must-gather** 데이터를 저장하려는 디렉터리로 이동합니다.
2. **--image** 또는 **--image-stream** 인수를 하나 이상 사용하여 **oc adm must-gather** 명령을 실행합니다. 예를 들어 다음 명령은 기본 클러스터 데이터와 OpenShift Virtualization 관련 정보를 모두 수집합니다.

```
$ oc adm must-gather \
--image-stream=openshift/must-gather \ 1
--image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8:v4.9.7 2
```

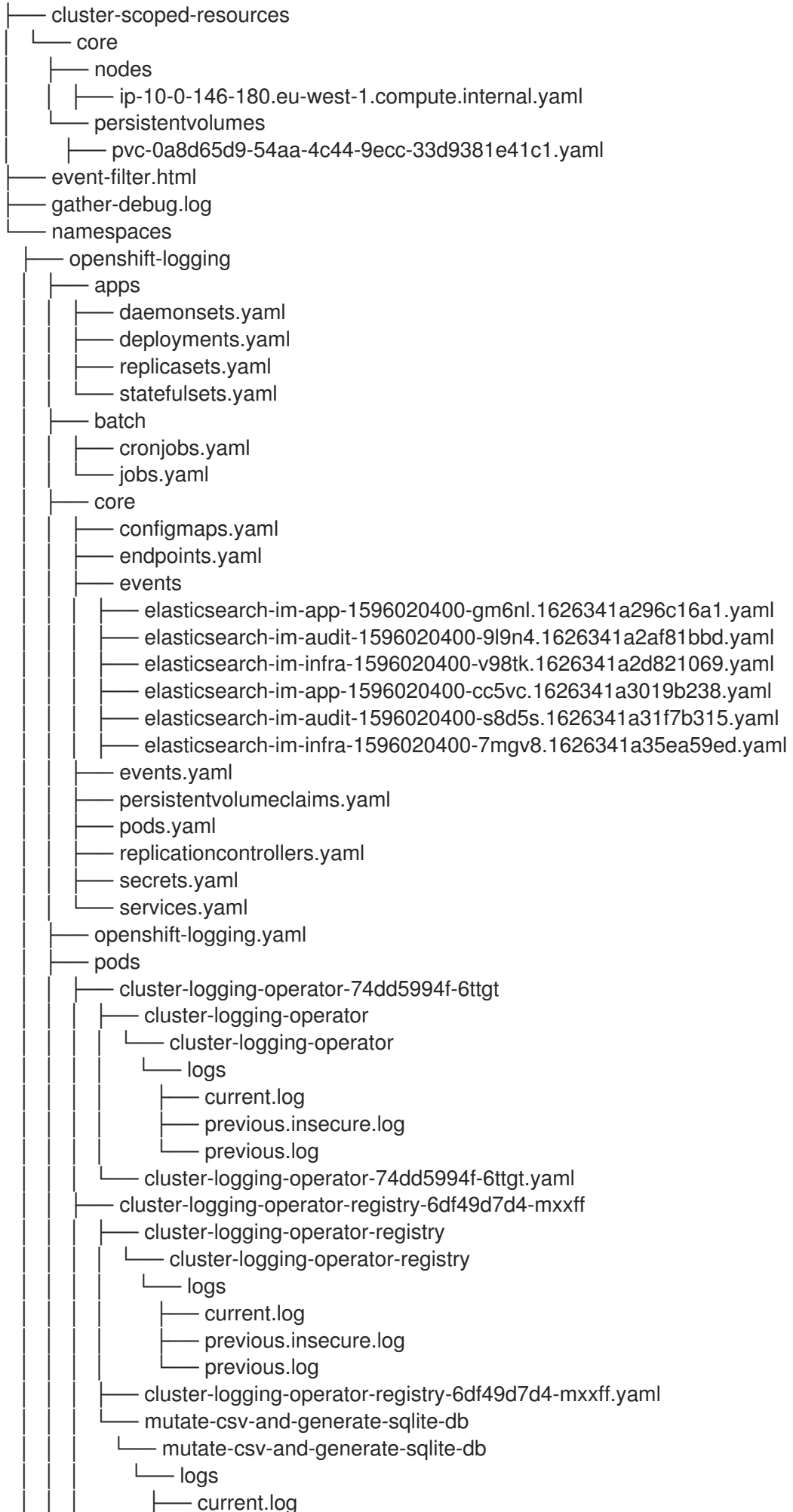
- 1 기본 OpenShift Container Platform **must-gather** 이미지
- 2 OpenShift Virtualization의 must-gather 이미지

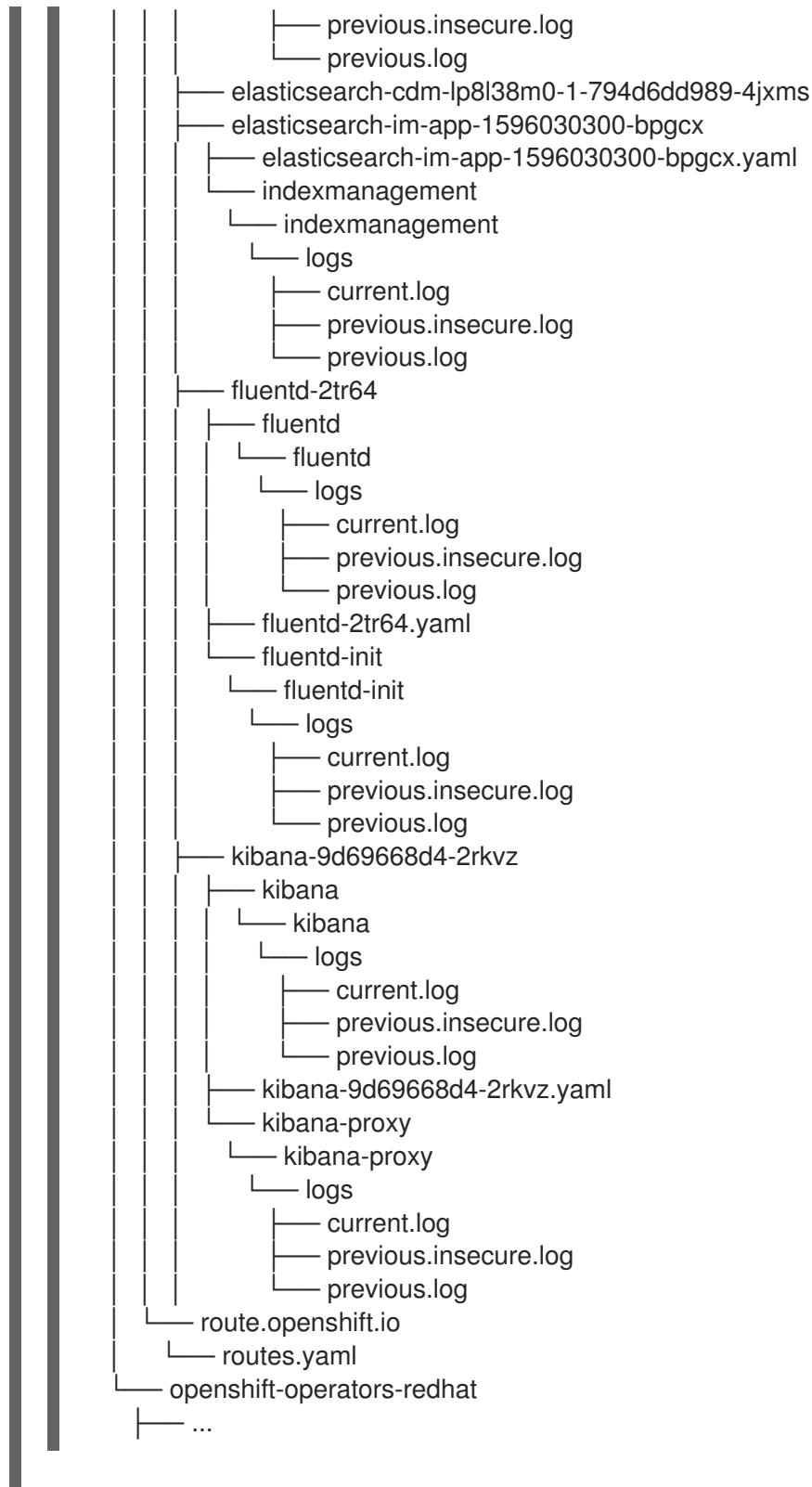
**must-gather** 툴을 추가 인수와 함께 사용하여 클러스터의 OpenShift 로깅 및 Red Hat OpenShift Logging Operator와 관련된 데이터를 수집할 수 있습니다. OpenShift 로깅의 경우 다음 명령을 실행합니다.

```
$ oc adm must-gather --image=$(oc -n openshift-logging get deployment.apps/cluster-logging-operator \
-o jsonpath='{.spec.template.spec.containers[?(@.name == "cluster-logging-operator")].image}')
```

### 예 5.1. OpenShift 로깅의 **must-gather** 출력 예

```
cluster-logging
├── clo
│   ├── cluster-logging-operator-74dd5994f-6ttgt
│   ├── clusterlogforwarder_cr
│   ├── cr
│   ├── csv
│   ├── deployment
│   └── logforwarding_cr
├── collector
│   └── fluentd-2tr64
├── eo
│   ├── csv
│   ├── deployment
│   └── elasticsearch-operator-7dc7d97b9d-jb4r4
├── es
│   ├── cluster-elasticsearch
│   │   ├── aliases
│   │   ├── health
│   │   ├── indices
│   │   ├── latest_documents.json
│   │   ├── nodes
│   │   ├── nodes_stats.json
│   │   └── thread_pool
│   ├── cr
│   ├── elasticsearch-cdm-lp8l38m0-1-794d6dd989-4jxms
│   └── logs
│       ├── elasticsearch-cdm-lp8l38m0-1-794d6dd989-4jxms
├── install
│   ├── co_logs
│   ├── install_plan
│   ├── olmo_logs
│   └── subscription
├── kibana
│   ├── cr
│   └── kibana-9d69668d4-2rkvz
```





3. **--image** 또는 **--image-stream** 인수를 하나 이상 사용하여 **oc adm must-gather** 명령을 실행합니다. 예를 들어 다음 명령은 기본 클러스터 데이터와 KubeVirt 관련 정보를 모두 수집합니다.

```

$ oc adm must-gather \
  --image-stream=openshift/must-gather \ 1
  --image=quay.io/kubevirt/must-gather 2

```

- 1 기본 OpenShift Container Platform **must-gather** 이미지
- 2 KubeVirt의 **must-gather** 이미지

4. 작업 디렉토리에서 생성된 **must-gather** 디렉토리에서 압축 파일을 만듭니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar cvaf must-gather.tar.gz must-gather.local.5421342344627712289/ 1
```

- 1 **must-gather-local.5421342344627712289/**를 실제 디렉터리 이름으로 교체하십시오.

5. 압축 파일을 [Red Hat 고객 포털](#)의 지원 케이스에 첨부합니다.

### 5.1.3. 감사 로그 수집

개별 사용자, 관리자 또는 시스템의 기타 구성 요소가 시스템에 영향을 준 활동 순서를 문서화하는 보안 관련 레코드 세트인 감사 로그를 수집할 수 있습니다. 다음에 대한 감사 로그를 수집할 수 있습니다.

- etcd 서버
- Kubernetes API 서버
- OpenShift OAuth API 서버
- OpenShift API 서버

#### 프로세스

1. `-- /usr/bin/gather_audit_logs` 플래그를 사용하여 `oc adm must-gather` 명령을 실행합니다.

```
$ oc adm must-gather -- /usr/bin/gather_audit_logs
```

2. 작업 디렉토리에서 생성된 **must-gather** 디렉토리에서 압축 파일을 만듭니다. 예를 들어 Linux 운영 체제를 사용하는 컴퓨터에서 다음 명령을 실행합니다.

```
$ tar cvaf must-gather.tar.gz must-gather.local.472290403699006248 1
```

- 1 **must-gather-local.4722403699006248**을 실제 디렉터리 이름으로 교체합니다.

3. 압축 파일을 [Red Hat 고객 포털](#)의 지원 케이스에 첨부합니다.

## 5.2. 클러스터 ID 검색

Red Hat 지원에 정보를 제공할 때 클러스터의 고유 식별자를 제공하는 것이 유용합니다. OpenShift Container Platform 웹 콘솔을 사용하여 클러스터 ID를 자동으로 입력할 수 있습니다. 웹 콘솔 또는 OpenShift CLI (**oc**)를 사용하여 클러스터 ID를 수동으로 검색할 수 있습니다.

#### 사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- 설치된 웹 콘솔 또는 OpenShift CLI (**oc**)에 액세스할 수 있어야 합니다.

#### 프로세스

- 웹 콘솔을 사용하여 지원 케이스를 열고 클러스터 ID를 자동으로 입력하려면 다음을 수행합니다.



- a. 톨바에서 (?) Help → Open Support Case로 이동합니다.
- b. Cluster ID 값이 자동으로 입력됩니다.
- 웹 콘솔을 사용하여 클러스터 ID를 수동으로 가져오려면 다음을 수행합니다.
  - a. Home → Dashboards → Overview로 이동합니다.
  - b. 값은 Details 섹션의 Cluster ID 필드에서 사용 가능합니다.
- OpenShift CLI (oc)를 사용하여 클러스터 ID를 얻으려면 다음 명령을 실행합니다.

```
$ oc get clusterversion -o jsonpath='{.items[].spec.clusterID}'
```

### 5.3. SOSREPORT 정보

**sosreport**는 RHEL(Red Hat Enterprise Linux) 및 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 구성 세부 정보, 시스템 정보, 진단 데이터를 수집하는 툴입니다. **sosreport**는 노드와 관련된 진단 정보를 수집하는 표준화된 방법을 제공합니다. 이러한 정보는 문제 진단을 위해 Red Hat 지원팀에 제공할 수 있습니다.

경우에 따라 Red Hat 지원팀에서 특정 OpenShift Container Platform 노드에 대한 **sosreport** 아카이브를 수집하도록 요청할 수 있습니다. 예를 들어, **oc adm must-gather**의 출력에 포함되지 않은 시스템 로그 또는 기타 노드 별 데이터를 확인해야 하는 경우가 있습니다.

### 5.4. OPENSIFT CONTAINER PLATFORM 클러스터 노드의 SOSREPORT 아카이브 생성

OpenShift Container Platform 4.9 클러스터 노드에 **sosreport**를 생성하는 방법으로 디버그 Pod를 사용하는 것이 좋습니다.

#### 사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- 호스트에 대한 SSH 액세스 권한이 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.
- Red Hat 표준 또는 프리미엄 서브스크립션이 있습니다.
- Red Hat 고객 포털 계정이 있어야 합니다.
- 기존 Red Hat 지원 케이스 ID가 있습니다.

#### 프로세스

1. 클러스터 노드 목록을 가져옵니다.

```
$ oc get nodes
```

2. 대상 노드에서 디버그 세션으로 들어갑니다. 이 단계는 **<node\_name>-debug**라는 디버그 Pod를 인스턴스화합니다.

```
$ oc debug node/my-cluster-node
```

**NoExecute** 효과로 테인트된 대상 노드에서 디버그 세션에 들어가려면 더미 네임스페이스에 허용 오차를 추가하고 더미 네임스페이스에서 디버그 Pod를 시작합니다.

```
$ oc new-project dummy
```

```
$ oc patch namespace dummy --type=merge -p '{"metadata": {"annotations": {"scheduler.alpha.kubernetes.io/defaultTolerations": [{"operator": "Exists"}]}}'
```

```
$ oc debug node/my-cluster-node
```

3. 디버그 셸 내에서 **/host**를 root 디렉터리로 설정합니다. 디버그 Pod는 Pod 내의 **/host**에 호스트의 루트 파일 시스템을 마운트합니다. root 디렉토리를 **/host**로 변경하면 호스트의 실행 경로에 포함된 바이너리를 실행할 수 있습니다.

```
# chroot /host
```



**참고**

RHCOS(Red Hat Enterprise Linux CoreOS)를 실행하는 OpenShift Container Platform 4.9 클러스터 노드는 변경할 수 없으며 Operator를 사용하여 클러스터 변경 사항을 적용합니다. SSH를 사용하여 클러스터 노드에 액세스하는 것은 권장되지 않으며 노드는 *accessed* 테인트로 표시됩니다. 그러나 OpenShift Container Platform API를 사용할 수 없거나 kubelet이 대상 노드에서 제대로 작동하지 않는 경우 **oc** 작업이 영향을 받습니다. 이러한 상황에서 대신 **ssh core @ <node>. <cluster\_name>.<base\_domain>**을 사용하여 노드에 액세스할 수 있습니다.

4. **sosreport** 를 실행하는 데 필요한 바이너리 및 플러그인이 포함된 **toolbox** 컨테이너를 시작합니다.

```
# toolbox
```



**참고**

기존 **toolbox** Pod가 이미 실행 중인 경우 **toolbox** 명령은 **'toolbox-' already exists**를 출력합니다. **Trying to start...**를 출력합니다. **podman rm toolbox-**에서 실행 중인 toolbox 컨테이너를 제거하고 새 toolbox 컨테이너를 생성하여 **sosreport** 플러그인 문제를 방지합니다.

5. **sosreport** 아카이브를 수집합니다.

- a. **sosreport** 명령을 실행하고 **crio.all** 및 **crio.logs** CRI-O 컨테이너 엔진 **sosreport** 플러그인을 활성화합니다.

```
# sosreport -k crio.all=on -k crio.logs=on 1
```

**1** **-k** 를 사용하면 기본값 외부에서 **sosreport** 플러그인 매개 변수를 정의할 수 있습니다.

- b. 프롬프트가 표시되면 **Enter**를 눌러 계속 진행합니다.

- c. Red Hat 지원 사례 ID를 제공합니다. **sosreport**는 아카이브의 파일 이름에 ID를 추가합니다.
- d. **sosreport** 출력은 아카이브의 위치와 체크섬을 제공합니다. 다음 예에서는 케이스 ID **01234567**을 참조합니다.

```
Your sosreport has been generated and saved in:
/host/var/tmp/sosreport-my-cluster-node-01234567-2020-05-28-eyjknxt.tar.xz 1
```

```
The checksum is: 382ffc167510fd71b4f12a4f40b97a4e
```

- 1 toolbox 컨테이너가 호스트의 root 디렉토리를 **/host**에 마운트하기 때문에 **sosreport** 아카이브의 파일 경로는 **chroot** 환경 외부에 있습니다.

6. 다음 방법 중 하나를 사용하여 분석을 위해 Red Hat 지원에 **sosreport** 아카이브를 제공합니다.

- OpenShift Container Platform 클러스터에서 직접 기존 Red Hat 지원 케이스에 파일을 업로드합니다.
  - a. toolbox 컨테이너 내에서 **redhat-support-tool** 을 실행하여 기존 Red Hat 지원 케이스에 아카이브를 직접 연결합니다. 이 예에서는 지원 사례 ID **01234567**을 사용합니다.

```
# redhat-support-tool addattachment -c 01234567 /host/var/tmp/my-sosreport.tar.xz
1
```

- 1 toolbox 컨테이너는 **/host**에 호스트의 root 디렉토리를 마운트합니다. **redhat-support-tool** 명령에 업로드할 파일을 지정할 때 **/host/**를 포함하여 toolbox 컨테이너의 root 디렉토리에서 절대 경로를 참조합니다.

- 기존 Red Hat 지원 케이스에 파일을 업로드합니다.
  - a. **oc debug node/<node\_name>** 명령을 실행하여 **sosreport** 아카이브를 연결하고 출력을 파일로 리디렉션합니다. 이 명령은 이전 **oc debug** 세션을 종료했다고 가정합니다.

```
$ oc debug node/my-cluster-node -- bash -c 'cat /host/var/tmp/sosreport-my-cluster-node-01234567-2020-05-28-eyjknxt.tar.xz' > /tmp/sosreport-my-cluster-node-01234567-2020-05-28-eyjknxt.tar.xz 1
```

- 1 디버그 컨테이너는 **/host**에 호스트의 root 디렉토리를 마운트합니다. 연결할 대상 파일을 지정할 때 **/host**를 포함하여 디버그 컨테이너의 root 디렉토리에서 절대 경로를 참조합니다.



### 참고

RHCOS(Red Hat Enterprise Linux CoreOS)를 실행하는 OpenShift Container Platform 4.9 클러스터 노드는 변경할 수 없으며 Operator를 사용하여 클러스터 변경 사항을 적용합니다. **scp**를 사용하여 클러스터 노드에서 **sosreport** 아카이브를 전송하는 것은 권장되지 않으며 노드는 accessed된 테인트로 표시됩니다. 그러나 OpenShift Container Platform API를 사용할 수 없거나 kubelet이 대상 노드에서 제대로 작동하지 않는 경우 **oc** 작업이 영향을 받습니다. 이러한 상황에서는 **scp core@<node>.<cluster\_name>.<base\_domain>:<file\_path> <local\_path>**를 실행하여 노드에서 **sosreport** 아카이브를 복사할 수 있습니다.

- b. <https://access.redhat.com/support/cases/> 내에서 기존 지원 케이스로 이동합니다.
- c. **Attach files**를 선택하고 메시지에 따라 파일을 업로드합니다.

## 5.5. 부트 스트랩 노드의 저널 로그 쿼리

부트 스트랩 관련 문제가 발생하는 경우 부트 스트랩 노드에서 **bootkube.service journald** 장치 로그 및 컨테이너 로그를 수집할 수 있습니다.

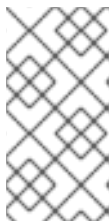
### 사전 요구 사항

- 부트 스트랩 노드에 대한 SSH 액세스 권한이 있어야 합니다.
- 부트 스트랩 노드의 정규화된 도메인 이름이 있어야 합니다.

### 프로세스

1. OpenShift Container Platform 설치 중에 부트 스트랩 노드에서 **bootkube.service journald** 장치 로그를 쿼리합니다. **<bootstrap\_fqdn>**을 부트 스트랩 노드의 정규화된 도메인 이름으로 바꿉니다.

```
$ ssh core@<bootstrap_fqdn> journalctl -b -f -u bootkube.service
```



### 참고

부트스트랩 노드의 **bootkube.service** 로그는 etcd **connection rejectd** 오류를 출력하고 부트스트랩 서버가 컨트롤 플레인 노드의 etcd에 연결할 수 없음을 나타냅니다. 각 컨트롤 플레인 노드에서 etcd를 시작하고 노드가 클러스터에 가입되면 오류가 중지됩니다.

2. 부트 스트랩 노드에서 **podman**을 사용하여 부트 스트랩 노드 컨테이너에서 로그를 수집합니다. **<bootstrap\_fqdn>**을 부트 스트랩 노드의 정규화된 도메인 이름으로 바꿉니다.

```
$ ssh core@<bootstrap_fqdn> 'for pod in $(sudo podman ps -a -q); do sudo podman logs $pod; done'
```

## 5.6. 클러스터 노드의 저널 로그 쿼리

개별 클러스터 노드의 **/var/log** 내에 **journald** 장치 로그 및 기타 로그를 수집할 수 있습니다.

### 사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- API 서비스가 작동하고 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.
- 호스트에 대한 SSH 액세스 권한이 있어야 합니다.

### 프로세스

1. OpenShift Container Platform 클러스터 노드에서 **kubelet journald** 장치 로그를 쿼리합니다. 다음 예제에서는 컨트롤 플레인 노드만 쿼리합니다.

```
$ oc adm node-logs --role=master -u kubelet 1
```

- 1 다른 장치 로그를 쿼리하려면 **kubelet**을 적절하게 대체합니다.

2. 클러스터 노드의 **/var/log/** 아래에있는 특정 하위 디렉터리에서 로그를 수집합니다.

- a. **/var/log/** 하위 디렉토리에 포함된 로그 목록을 검색합니다. 다음 예제는 모든 컨트롤 플레인 노드의 **/var/log/openshift-apiserver/**에 있는 파일을 나열합니다.

```
$ oc adm node-logs --role=master --path=openshift-apiserver
```

- b. **/var/log/** 하위 디렉터리 내의 특정 로그를 확인합니다. 다음 예제는 모든 컨트롤 플레인 노드에서 **/var/log/openshift-apiserver/audit.log** 내용을 출력합니다.

```
$ oc adm node-logs --role=master --path=openshift-apiserver/audit.log
```

- c. API가 작동하지 않는 경우 SSH를 사용하여 각 노드의 로그를 확인합니다. 다음은 **/var/log/openshift-apiserver/audit.log** 예제입니다.

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo tail -f /var/log/openshift-apiserver/audit.log
```



### 참고

RHCOS(Red Hat Enterprise Linux CoreOS)를 실행하는 OpenShift Container Platform 4.9 클러스터 노드는 변경할 수 없으며 Operator를 사용하여 클러스터 변경 사항을 적용합니다. SSH를 사용하여 클러스터 노드에 액세스하는 것은 권장되지 않으며 노드는 *accessed* 테인트로 표시됩니다. SSH를 통해 진단 데이터를 수집하기 전에 **oc adm must gather** 및 기타 **oc** 명령을 실행하여 충분한 데이터를 수집할 수 있는지 확인하십시오. 그러나 OpenShift Container Platform API를 사용할 수 없거나 kubelet이 대상 노드에서 제대로 작동하지 않는 경우 **oc** 작업이 영향을 받습니다. 이러한 상황에서 **ssh core@<node>.<cluster\_name>.<base\_domain>**을 사용하여 노드에 액세스할 수 있습니다.

## 5.7. OPENSIFT CONTAINER PLATFORM 노드 또는 컨테이너에서 네트워크 추적 수집

잠재적인 네트워크 관련 OpenShift Container Platform 문제를 조사할 때 Red Hat 지원은 특정 OpenShift Container Platform 클러스터 노드 또는 특정 컨테이너에서 네트워크 패킷 추적을 요청할 수 있습니다. OpenShift Container Platform에서 네트워크 추적을 캡처하는 데 권장되는 방법은 디버그 Pod를 사용하는 것입니다.

### 사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.
- Red Hat 표준 또는 프리미엄 서브스크립션이 있습니다.

- Red Hat 고객 포털 계정이 있어야 합니다.
- 기존 Red Hat 지원 케이스 ID가 있습니다.
- 호스트에 대한 SSH 액세스 권한이 있어야 합니다.

프로세스

1. 클러스터 노드 목록을 가져옵니다.

```
$ oc get nodes
```

2. 대상 노드에서 디버그 세션으로 들어갑니다. 이 단계는 `<node_name>-debug` 라는 디버그 Pod를 인스턴스화합니다.

```
$ oc debug node/my-cluster-node
```

3. 디버그 셸 내에서 `/host`를 root 디렉터리로 설정합니다. 디버그 Pod는 Pod 내의 `/host`에 호스트의 루트 파일 시스템을 마운트합니다. root 디렉토리를 `/host`로 변경하면 호스트의 실행 경로에 포함된 바이너리를 실행할 수 있습니다.

```
# chroot /host
```



참고

RHCOS(Red Hat Enterprise Linux CoreOS)를 실행하는 OpenShift Container Platform 4.9 클러스터 노드는 변경할 수 없으며 Operator를 사용하여 클러스터 변경 사항을 적용합니다. SSH를 사용하여 클러스터 노드에 액세스하는 것은 권장되지 않으며 노드는 `accessed` 태인트로 표시됩니다. 그러나 OpenShift Container Platform API를 사용할 수 없거나 kubelet이 대상 노드에서 제대로 작동하지 않는 경우 `oc` 작업이 영향을 받습니다. 이러한 상황에서 대신 `ssh core @ <node>.<cluster_name>.<base_domain>`을 사용하여 노드에 액세스할 수 있습니다.

4. `chroot` 환경 콘솔에서 노드의 인터페이스 이름을 가져옵니다.

```
# ip ad
```

5. `sosreport` 를 실행하는 데 필요한 바이너리 및 플러그인이 포함된 `toolbox` 컨테이너를 시작합니다.

```
# toolbox
```



참고

기존 `toolbox` Pod가 이미 실행 중인 경우 `toolbox` 명령은 `'toolbox-' already exists`를 출력합니다. `Trying to start...`를 출력합니다. `tcpdump` 문제를 방지하려면 `podman rm toolbox-`에서 실행 중인 `toolbox` 컨테이너를 제거하고 새 `toolbox` 컨테이너를 생성합니다.

6. 클러스터 노드에서 `tcpdump` 세션을 시작하고 출력을 캡처 파일로 리디렉션합니다. 이 예에서는 `ens5`를 인터페이스 이름으로 사용합니다.

```
$ tcpdump -nn -s 0 -i ens5 -w /host/var/tmp/my-cluster-node_$(date +%d_%m_%Y-%H_%M_%S-%Z).pcap 1
```

- 1 toolbox 컨테이너가 호스트의 root 디렉토리를 **/host**에 마운트하기 때문에 **tcpdump** 캡처 파일의 경로는 **chroot** 환경 외부에 있습니다.

7. 노드의 특정 컨테이너에 **tcpdump** 캡처가 필요한 경우 다음 단계를 따르십시오.

- a. 대상 컨테이너 ID를 확인합니다. toolbox 컨테이너가 호스트의 root 디렉토리를 **/host**에 마운트하기 때문에 **chroot host** 명령은 이 단계에서 **crictl** 명령 보다 우선합니다.

```
# chroot /host crictl ps
```

- b. 컨테이너의 프로세스 ID를 확인합니다. 이 예에서 컨테이너 ID는 **a7fe32346b120**입니다.

```
# chroot /host crictl inspect --output yaml a7fe32346b120 | grep 'pid' | awk '{print $2}'
```

- c. 컨테이너에서 **tcpdump** 세션을 시작하고 출력을 캡처 파일로 리디렉션합니다. 이 예는 컨테이너의 프로세스 ID로 **49,628**을 사용하고 인터페이스 이름으로 **ens5**를 사용합니다. **nsenter** 명령은 대상 프로세스의 네임 스페이스를 입력하고 해당 네임 스페이스를 사용하여 명령을 실행합니다. 이 예에서 대상 프로세스는 컨테이너의 프로세스 ID이므로 **tcpdump** 명령은 호스트에서 컨테이너 네임 스페이스를 사용하여 실행됩니다.

```
# nsenter -n -t 49628 -- tcpdump -nn -i ens5 -w /host/var/tmp/my-cluster-node-my-container_$(date +%d_%m_%Y-%H_%M_%S-%Z).pcap.pcap 1
```

- 1 toolbox 컨테이너가 호스트의 root 디렉토리를 **/host**에 마운트하기 때문에 **tcpdump** 캡처 파일의 경로는 **chroot** 환경 외부에 있습니다.

8. 분석을 위해 다음 방법 중 하나를 사용하여 **tcpdump** 캡처 파일을 Red Hat 지원팀에 제공합니다.

- OpenShift Container Platform 클러스터에서 직접 기존 Red Hat 지원 케이스에 파일을 업로드합니다.
  - a. toolbox 컨테이너 내에서 **redhat-support-tool**을 실행하여 기존 Red Hat 지원 케이스에 직접 파일을 첨부합니다. 이 예에서는 지원 사례 ID **01234567**을 사용합니다.

```
# redhat-support-tool addattachment -c 01234567 /host/var/tmp/my-tcpdump-capture-file.pcap 1
```

- 1 toolbox 컨테이너는 **/host**에 호스트의 root 디렉토리를 마운트합니다. **redhat-support-tool** 명령에 업로드할 파일을 지정할 때 **/host/**를 포함하여 toolbox 컨테이너의 root 디렉토리에서 절대 경로를 참조합니다.

- 기존 Red Hat 지원 케이스에 파일을 업로드합니다.
  - a. **oc debug node/<node\_name>** 명령을 실행하여 **sosreport** 아카이브를 연결하고 출력을 파일로 리디렉션합니다. 이 명령은 이전 **oc debug** 세션을 종료했다고 가정합니다.

```
$ oc debug node/my-cluster-node -- bash -c 'cat /host/var/tmp/my-tcpdump-capture-file.pcap' > /tmp/my-tcpdump-capture-file.pcap 1
```

- 1 디버그 컨테이너는 **/host**에 호스트의 root 디렉토리를 마운트합니다. 연결할 대상 파일을 지정할 때 **/host**를 포함하여 디버그 컨테이너의 root 디렉토리에서 절대 경



**참고**

RHCOS(Red Hat Enterprise Linux CoreOS)를 실행하는 OpenShift Container Platform 4.9 클러스터 노드는 변경할 수 없으며 Operator를 사용하여 클러스터 변경 사항을 적용합니다. **scp**를 사용하여 클러스터 노드에서 **tcpdump** 캡처 파일을 전송하는 것은 권장되지 않으며 노드는 *accessed* 테인트로 표시됩니다. 그러나 OpenShift Container Platform API를 사용할 수 없거나 kubelet이 대상 노드에서 제대로 작동하지 않는 경우 **oc** 작업이 영향을 받습니다. 이러한 상황에서는 **scp core@<node>.<cluster\_name>.<base\_domain>:<file\_path> <local\_path>**를 실행하여 노드에서 **tcpdump** 캡처 파일을 복사할 수 있습니다.

- b. <https://access.redhat.com/support/cases/> 내에서 기존 지원 케이스로 이동합니다.
- c. **Attach files**를 선택하고 메시지에 따라 파일을 업로드합니다.

### 5.8. RED HAT 지원에 진단 데이터 제공

OpenShift Container Platform 문제를 조사할 때 Red Hat 지원은 지원 케이스에 진단 데이터를 업로드하도록 요청할 수 있습니다. 파일은 Red Hat 고객 포털을 통해 지원 케이스에 업로드하거나 **redhat-support-tool** 명령을 사용하여 OpenShift Container Platform 클러스터에서 직접 업로드할 수 있습니다.

**사전 요구 사항**

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- 호스트에 대한 SSH 액세스 권한이 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.
- Red Hat 표준 또는 프리미엄 서브스크립션이 있습니다.
- Red Hat 고객 포털 계정이 있어야 합니다.
- 기존 Red Hat 지원 케이스 ID가 있습니다.

**프로세스**

- Red Hat 고객 포털을 통해 기존 Red Hat 지원 케이스에 진단 데이터를 업로드합니다.
- 1. **oc debug node/<node\_name>** 명령을 사용하여 OpenShift Container Platform 노드에 포함된 진단 파일을 연결하고 출력을 파일로 리디렉션합니다. 다음 예에서는 **/host/var/tmp/my-diagnostic-data.tar.gz**를 디버그 컨테이너에서 **/var/tmp/my-diagnostic-data.tar.gz**로 복사합니다.

```
$ oc debug node/my-cluster-node -- bash -c 'cat /host/var/tmp/my-diagnostic-data.tar.gz' > /var/tmp/my-diagnostic-data.tar.gz 1
```

- 1 디버그 컨테이너는 **/host**에 호스트의 root 디렉토리를 마운트합니다. 연결할 대상 파일을 지정할 때 **/host**를 포함하여 디버그 컨테이너의 root 디렉토리에서 절대 경로를 참조합니다.





### 참고

RHCOS(Red Hat Enterprise Linux CoreOS)를 실행하는 OpenShift Container Platform 4.9 클러스터 노드는 변경할 수 없으며 Operator를 사용하여 클러스터 변경 사항을 적용합니다. **scp**를 사용하여 클러스터 노드에서 파일을 전송하는 것은 권장되지 않으며 노드는 *accessed* 테이트로 표시됩니다. 그러나 OpenShift Container Platform API를 사용할 수 없거나 kubelet이 대상 노드에서 제대로 작동하지 않는 경우 **oc** 작업이 영향을 받습니다. 이러한 상황에서 **scp core@<node>.<cluster\_name>.<base\_domain>:<file\_path> <local\_path>**를 실행하여 노드에서 진단 파일을 복사할 수 있습니다.

2. <https://access.redhat.com/support/cases/> 내에서 기존 지원 케이스로 이동합니다.
  3. **Attach files**를 선택하고 메시지에 따라 파일을 업로드합니다.
- OpenShift Container Platform 클러스터에서 직접 기존 Red Hat 지원 케이스에 진단 데이터를 업로드합니다.
    1. 클러스터 노드 목록을 가져옵니다.

```
$ oc get nodes
```

2. 대상 노드에서 디버그 세션으로 들어갑니다. 이 단계는 **<node\_name>-debug**라는 디버그 Pod를 인스턴스화합니다.

```
$ oc debug node/my-cluster-node
```

3. 디버그 셸 내에서 **/host**를 root 디렉터리로 설정합니다. 디버그 Pod는 Pod 내의 **/host**에 호스트의 루트 파일 시스템을 마운트합니다. root 디렉토리를 **/host**로 변경하면 호스트의 실행 경로에 포함된 바이너리를 실행할 수 있습니다.

```
# chroot /host
```



### 참고

RHCOS(Red Hat Enterprise Linux CoreOS)를 실행하는 OpenShift Container Platform 4.9 클러스터 노드는 변경할 수 없으며 Operator를 사용하여 클러스터 변경 사항을 적용합니다. SSH를 사용하여 클러스터 노드에 액세스하는 것은 권장되지 않으며 노드는 *accessed* 테이트로 표시됩니다. 그러나 OpenShift Container Platform API를 사용할 수 없거나 kubelet이 대상 노드에서 제대로 작동하지 않는 경우 **oc** 작업이 영향을 받습니다. 이러한 상황에서 대신 **ssh core @ <node>.<cluster\_name>.<base\_domain>**을 사용하여 노드에 액세스할 수 있습니다.

4. **redhat-support-tool**을 실행하는 데 필요한 바이너리가 포함된 **toolbox** 컨테이너를 시작합니다.

```
# toolbox
```



## 참고

기존 **toolbox** Pod가 이미 실행 중인 경우 **toolbox** 명령은 **'toolbox-' already exists**를 출력합니다. **Trying to start...**를 출력합니다. **podman rm toolbox-**에서 실행 중인 toolbox 컨테이너를 제거하고 새 toolbox 컨테이너를 생성하여 문제를 방지합니다.

- a. **redhat-support-tool**을 실행하여 디버그 Pod의 파일을 기존 Red Hat 지원 케이스에 직접 첨부합니다. 이 예에서는 지원 케이스 ID '01234567'과 예제 파일 경로 **/host/var/tmp/my-diagnostic-data.tar.gz**를 사용합니다.

```
# redhat-support-tool addattachment -c 01234567 /host/var/tmp/my-diagnostic-data.tar.gz 1
```

- 1 toolbox 컨테이너는 **/host**에 호스트의 root 디렉토리를 마운트합니다. **redhat-support-tool** 명령에 업로드할 파일을 지정할 때 **/host/**를 포함하여 toolbox 컨테이너의 root 디렉토리에서 절대 경로를 참조합니다.

## 5.9. TOOLBOX 정보

**toolbox**는 RHCOS(Red Hat Enterprise Linux CoreOS) 시스템에서 컨테이너를 시작하는 툴입니다. 이 툴은 주로 **sosreport** 및 **redhat-support-tool** 과 같은 명령을 실행하는 데 필요한 필수 바이너리 및 플러그인이 포함된 컨테이너를 시작하는 데 사용됩니다.

**toolbox** 컨테이너의 주요 목적은 진단 정보를 수집하여 Red Hat 지원에 제공하는 것입니다. 그러나 추가 진단 도구가 필요한 경우 RPM 패키지를 추가하거나 표준 지원 도구 이미지의 대체 이미지를 실행할 수 있습니다.

### toolbox 컨테이너에 패키지 설치

기본적으로 **toolbox** 명령을 실행하면 **registry.redhat.io/rhel8/support-tools:latest** 이미지로 컨테이너를 시작합니다. 이 이미지에는 가장 자주 사용되는 지원 도구가 포함되어 있습니다. 이미지에 포함되지 않은 지원 툴이 필요한 노드별 데이터를 수집해야 하는 경우 추가 패키지를 설치할 수 있습니다.

### 사전 요구 사항

- **oc debug node/<node\_name>** 명령이 있는 노드에 액세스하고 있습니다.

### 프로세스

1. 디버그 셸 내에서 **/host**를 root 디렉터리로 설정합니다. 디버그 Pod는 Pod 내의 **/host**에 호스트의 루트 파일 시스템을 마운트합니다. root 디렉토리를 **/host**로 변경하면 호스트의 실행 경로에 포함된 바이너리를 실행할 수 있습니다.

```
# chroot /host
```

2. toolbox 컨테이너를 시작합니다.

```
# toolbox
```

3. **wget**과 같은 추가 패키지를 설치합니다.

```
# dnf install -y <package_name>
```

## toolbox를 사용하여 대체 이미지 시작

기본적으로 **toolbox** 명령을 실행하면 **registry.redhat.io/rhel8/support-tools:latest** 이미지로 컨테이너를 시작합니다. **.toolboxrc** 파일을 생성하고 실행할 이미지를 지정하여 대체 이미지를 시작할 수 있습니다.

### 사전 요구 사항

- **oc debug node/<node\_name>** 명령이 있는 노드에 액세스하고 있습니다.

### 프로세스

1. 디버그 셸 내에서 **/host**를 root 디렉터리로 설정합니다. 디버그 Pod는 Pod 내의 **/host**에 호스트의 루트 파일 시스템을 마운트합니다. root 디렉토리를 **/host**로 변경하면 호스트의 실행 경로에 포함된 바이너리를 실행할 수 있습니다.

```
# chroot /host
```

2. root 사용자 ID에 대한 홈 디렉터리에 **.toolboxrc** 파일을 생성합니다.

```
# vi ~/.toolboxrc
```

```
REGISTRY=quay.io           1
IMAGE=fedora/fedora:33-x86_64 2
TOOLBOX_NAME=toolbox-fedora-33 3
```

- 1 선택 사항: 대체 컨테이너 레지스트리를 지정합니다.
- 2 시작할 대체 이미지를 지정합니다.
- 3 선택 사항: toolbox 컨테이너의 대체 이름을 지정합니다.

3. 대체 이미지를 사용하여 toolbox 컨테이너를 시작합니다.

```
# toolbox
```



### 참고

기존 **toolbox** Pod가 이미 실행 중인 경우 **toolbox** 명령은 **'toolbox-' already exists**를 출력합니다. **Trying to start...**를 출력합니다. **podman rm toolbox-**에서 실행 중인 toolbox 컨테이너를 제거하고 새 toolbox 컨테이너를 생성하여 **sosreport** 플러그인 문제를 방지합니다.

## 6장. 클러스터 사양 요약

### 6.1. CLUSTERVERSION을 통해 클러스터 사양 요약

**clusterversion** 리소스를 쿼리하여 OpenShift Container Platform 클러스터 사양 요약을 가져올 수 있습니다.

#### 사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.

#### 프로세스

1. 클러스터 버전, 가용성, 가동 시간 및 일반 상태를 쿼리합니다.

```
$ oc get clusterversion
```

2. 클러스터 사양, 업데이트 가용성 및 업데이트 기록에 대한 자세한 요약을 가져옵니다.

```
$ oc describe clusterversion
```

## 7장. 문제 해결

### 7.1. 설치 문제 해결

#### 7.1.1. 설치 문제가 발생하는 위치 확인

OpenShift Container Platform 설치 문제를 해결할 때 설치 로그를 모니터링하여 문제가 발생하는 단계를 확인할 수 있습니다. 그런 다음 해당 단계와 관련된 진단 데이터를 검색합니다.

OpenShift Container Platform 설치하는 다음 단계에 따라 진행됩니다.

1. Ignition 구성 파일이 생성됩니다.
2. 부트스트랩 머신이 부팅되고 컨트롤 플레인 머신을 부팅하는 데 필요한 원격 리소스 호스팅이 시작됩니다.
3. 컨트롤 플레인 머신은 부트스트랩 머신에서 원격 리소스를 가져오고 부팅을 완료합니다.
4. 컨트롤 플레인 머신은 부트스트랩 머신을 사용하여 etcd 클러스터를 만듭니다.
5. 부트스트랩 머신은 새로운 etcd 클러스터를 사용하여 임시 Kubernetes 컨트롤 플레인을 시작합니다.
6. 임시 컨트롤 플레인은 프로덕션 컨트롤러 플레인을 컨트롤 플레인 머신에 예약합니다.
7. 임시 컨트롤 플레인이 종료되고 제어를 프로덕션 컨트롤 플레인에 전달합니다.
8. 부트스트랩 머신은 OpenShift Container Platform 구성 요소를 프로덕션 컨트롤 플레인에 추가합니다.
9. 설치 프로그램이 부트 스트랩 머신을 종료합니다.
10. 컨트롤 플레인이 작업자 노드를 설정합니다.
11. 컨트롤 플레인은 일련의 Operator 형태로 추가 서비스를 설치합니다.
12. 클러스터는 지원되는 환경에서 작업자 머신 생성을 포함하여 일상적인 작업에 필요한 나머지 구성 요소를 다운로드하고 구성합니다.

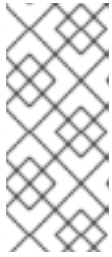
#### 7.1.2. 사용자 프로비저닝 인프라 설치 고려 사항

기본 설치 방법은 설치 프로그램에서 프로비저닝한 인프라를 사용하는 것입니다. 설치 프로그램에서 프로비저닝한 인프라 클러스터를 통해 OpenShift Container Platform은 운영 체제 자체를 포함하여 클러스터의 모든 측면을 관리합니다. 가능하면 이 기능을 사용하여 클러스터 인프라를 프로비저닝 및 유지 관리의 부담을 덜 수 있습니다.

사용자가 제공하는 인프라에 OpenShift Container Platform 4.9를 설치할 수도 있습니다. 이 설치 방법을 사용하는 경우 사용자가 제공하는 인프라 설치 설명서를 주의 깊게 따르십시오. 또한 설치 전에 다음 고려 사항을 확인하십시오.

- [RHEL \(Red Hat Enterprise Linux\) Ecosystem](#) 을 확인하고 선택한 서버 하드웨어 또는 가상화 기술에 대해 제공되는 RHCOS (Red Hat Enterprise Linux CoreOS) 지원 수준을 결정합니다.
- 많은 가상화 및 클라우드 환경에서는 게스트 운영 체제에 에이전트를 설치해야 합니다. 이러한 에이전트가 데몬 세트를 통해 배포되는 컨테이너화된 워크로드로 설치되어 있는지 확인합니다.

- 동적 스토리지, 온디맨드 서비스 라우팅, 노드 호스트 이름을 Kubernetes 호스트 이름으로 확인, 클러스터 자동 스케일링과 같은 기능을 사용하려면 클라우드 공급자 통합을 설치합니다.



**참고**

서로 다른 클라우드 공급자의 리소스를 결합하거나 여러 물리적 또는 가상 플랫폼에 걸쳐있는 OpenShift Container Platform 환경에서는 클라우드 공급자 통합을 활성화할 수 없습니다. 노드 라이프 사이클 컨트롤러는 기존 공급자 외부에 있는 노드를 클러스터에 추가하는 것을 허용하지 않으며 둘 이상의 클라우드 공급자 통합을 지정할 수 없습니다.

- 머신 세트 또는 자동 스케일링을 사용하여 OpenShift Container Platform 클러스터 노드를 자동으로 프로비저닝하려면 공급자별 Machine API를 구현해야 합니다.
- 선택한 클라우드 공급자가 초기 배포의 일부로 Ignition 구성 파일을 호스트에 삽입하는 방법을 제공하는지 확인합니다. 제공하지 않는 경우 HTTP 서버를 사용하여 Ignition 구성 파일을 호스팅해야 합니다. Ignition 구성 파일의 문제 해결 단계는 이 두 가지 방법 중 배포되는 방법에 따라 달라집니다.
- 포함된 컨테이너 레지스트리, Elasticsearch 또는 Prometheus와 같은 선택적 프레임워크 구성 요소를 활용하려면 스토리지를 수동으로 프로비저닝해야 합니다. 이를 명시적으로 구성하지 않는 한 기본 스토리지 클래스는 사용자 프로비저닝 인프라 설치에 정의되지 않습니다.
- 고가용성 OpenShift Container Platform 환경의 모든 컨트롤 플레인 노드에 API 요청을 분산하려면 로드 밸런서가 필요합니다. OpenShift Container Platform DNS 라우팅 및 포트 요구 사항을 충족하는 모든 TCP 기반 부하 분산 솔루션을 사용할 수 있습니다.

**7.1.3. OpenShift Container Platform 설치 전에 로드 밸런서 구성 확인**

OpenShift Container Platform 설치를 시작하기 전에 로드 밸런서 구성을 확인하십시오.

**사전 요구 사항**

- OpenShift Container Platform 설치를 준비하기 위해 선택한 외부 로드 밸런서를 구성하고 있어야 합니다. 다음 예에서는 HAProxy를 사용하여 클러스터에 로드 밸런싱 서비스를 제공하는 Red Hat Enterprise Linux (RHEL) 호스트를 기반으로 합니다.
- OpenShift Container Platform 설치를 준비하기 위해 DNS를 구성하고 있어야 합니다.
- 로드 밸런서에 SSH 액세스 권한이 있어야 합니다.

**프로세스**

1. **haproxy** systemd 서비스가 활성화되어 있는지 확인합니다.

```
$ ssh <user_name>@<load_balancer> systemctl status haproxy
```

2. 로드 밸런서가 필요한 포트에서 수신하고 있는지 확인합니다. 다음 예에서는 포트 **80, 443, 6443, 22623**을 참조합니다.

- RHEL (Red Hat Enterprise Linux) 6에서 실행되는 HAProxy 인스턴스의 경우 **netstat** 명령을 사용하여 포트 상태를 확인합니다.

```
$ ssh <user_name>@<load_balancer> netstat -nltupe | grep -E ':80|:443|:6443|:22623'
```

- RHEL (Red Hat Enterprise Linux) 7 또는 8에서 실행되는 HAProxy 인스턴스의 경우 **ss** 명령을 사용하여 포트 상태를 확인합니다.

```
$ ssh <user_name>@<load_balancer> ss -nltp | grep -E ':80|:443|:6443|:22623'
```



### 참고

Red Hat Enterprise Linux(RHEL) 7 이상에서는 **netstat** 대신 **ss** 명령을 사용하는 것이 좋습니다. **ss**는 iproute 패키지에서 제공합니다. **ss** 명령에 대한 자세한 내용은 [Red Hat Enterprise Linux \(RHEL\) 7 성능 조정 가이드](#) 를 참조하십시오.

3. 와일드 카드 DNS 레코드가 로드 밸런서로 해결되어 있는지 확인합니다.

```
$ dig <wildcard_fqdn> @<dns_server>
```

## 7.1.4. OpenShift Container Platform 설치 프로그램 로그 수준 지정

기본적으로 OpenShift Container Platform 설치 프로그램 로그 수준은 **info**로 설정됩니다. 실패한 OpenShift Container Platform 설치를 진단할 때 보다 자세한 로깅이 필요한 경우 다시 설치를 시작할 때 **openshift-install** 로그 수준을 **debug**로 높일 수 있습니다.

### 사전 요구 사항

- 설치 호스트에 대한 액세스 권한이 있어야 합니다.

### 프로세스

- 설치를 시작할 때 **debug**로 설치 로그 수준을 설정합니다.

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete --log-level debug
```

1

- 1 가능한 로그 수준에는 **info**, **warn**, **error**, **debug**가 있습니다.

## 7.1.5. openshift-install 명령 문제 해결

**openshift-install** 명령을 실행하는데 문제가 있는 경우 다음을 확인하십시오.

- 설치는 Ignition 구성 파일 생성 후 24 시간 이내에 시작되고 있습니다. 다음 명령을 실행하면 Ignition 파일이 생성됩니다.

```
$ ./openshift-install create ignition-configs --dir=./install_dir
```

- **install-config.yaml** 파일은 설치 프로그램과 동일한 디렉토리에 있습니다. **./openshift-install --dir** 옵션을 사용하여 대체 설치 경로를 선언한 경우 해당 디렉토리에 **install-config.yaml** 파일이 있는지 확인합니다.

## 7.1.6. 설치 진행 상황 모니터링

OpenShift Container Platform 설치가 진행됨에 따라 상위 수준 설치, 부트 스트랩 및 컨트롤 플레인 로그를 모니터링할 수 있습니다. 이렇게 하면 설치 진행 방식에 대한 가시성이 향상되고 설치 실패가 발생하는 단계를 식별할 수 있습니다.

### 사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.
- 호스트에 대한 SSH 액세스 권한이 있어야 합니다.
- 부트 스트랩 및 컨트롤 플레인 노드의 완전한 도메인 이름이 있어야 합니다.



#### 참고

초기 **kubeadmin** 암호는 설치 호스트의 **<install\_directory>/auth/kubeadmin-password**에서 찾을 수 있습니다.

### 프로세스

1. 설치가 진행되는 동안 설치 로그를 모니터링합니다.

```
$ tail -f ~/<installation_directory>/openshift_install.log
```

2. 부트스트랩 노드에서 **bootkube.service** journald 장치 로그를 모니터링합니다. 이를 통해 첫 번째 컨트롤 플레인의 부트 스트랩을 시각화할 수 있습니다. **<bootstrap\_fqdn>**을 부트 스트랩 노드의 정규화된 도메인 이름으로 바꿉니다.

```
$ ssh core@<bootstrap_fqdn> journalctl -b -f -u bootkube.service
```



#### 참고

부트스트랩 노드의 **bootkube.service** 로그는 **etcd connection rejectd** 오류를 출력하고 부트스트랩 서버가 컨트롤 플레인 노드의 **etcd**에 연결할 수 없음을 나타냅니다. 각 컨트롤 플레인 노드에서 **etcd**를 시작하고 노드가 클러스터에 가입되면 오류가 중지됩니다.

3. 컨트롤 플레인 노드가 부팅된 후 **kubelet.service** journald 장치 로그를 모니터링합니다. 이를 통해 컨트롤 플레인 노드 에이전트 활동을 시각화할 수 있습니다.

- a. **oc**를 사용하여 로그를 모니터링합니다.

```
$ oc adm node-logs --role=master -u kubelet
```

- b. API가 작동하지 않으면 대신 SSH를 사용하여 로그를 확인합니다. **<master-node>**. **<cluster\_name>**. **<base\_domain>**을 적절한 값으로 바꿉니다.

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> journalctl -b -f -u kubelet.service
```

4. 컨트롤 플레인 노드가 부팅된 후 **crio.service** journald 장치 로그를 모니터링합니다. 이를 통해 컨트롤 플레인 노드 CRI-O 컨테이너 런타임 활동을 시각화할 수 있습니다.



- a. **oc**를 사용하여 로그를 모니터링합니다.

```
$ oc adm node-logs --role=master -u crio
```

- b. API가 작동하지 않으면 대신 SSH를 사용하여 로그를 확인합니다. **<master-node>**, **<cluster\_name>**, **<base\_domain>**을 적절한 값으로 바꿉니다.

```
$ ssh core@master-N.cluster_name.sub_domain.domain journalctl -b -f -u crio.service
```

### 7.1.7. 부트 스트랩 노드 진단 데이터 수집

부트 스트랩 관련 문제가 발생하면 부트 스트랩 노드에서 **bootkube.service journald** 장치 로그 및 컨테이너 로그를 수집할 수 있습니다.

#### 사전 요구 사항

- 부트 스트랩 노드에 대한 SSH 액세스 권한이 있어야 합니다.
- 부트 스트랩 노드의 정규화된 도메인 이름이 있어야 합니다.
- HTTP 서버를 사용하여 Ignition 구성 파일을 호스팅하는 경우 HTTP 서버의 정규화된 도메인 이름과 포트 번호가 있어야 합니다. 또한 HTTP 호스트에 대한 SSH 액세스 권한이 있어야 합니다.

#### 프로세스

1. 부트 스트랩 노드의 콘솔에 액세스할 경우 노드가 로그인 프롬프트에 도달할 때까지 콘솔을 모니터링합니다.
2. Ignition 파일 구성을 확인합니다.
  - HTTP 서버를 사용하여 Ignition 구성 파일을 호스팅하는 경우:
    - a. 부트 스트랩 노드 Ignition 파일 URL을 확인합니다. **<http\_server\_fqdn>**을 HTTP 서버의 정규화된 도메인 이름으로 바꿉니다.

```
$ curl -I http://<http_server_fqdn>:<port>/bootstrap.ign 1
```

- 1 **-I** 옵션은 헤더만 반환합니다. 지정된 URL에서 Ignition 파일을 사용할 수 있는 경우 명령은 **200 OK** 상태를 반환합니다. 사용할 수 없는 경우 명령은 **404 file not found**를 반환합니다.

- b. 부트 스트랩 노드에서 Ignition 파일을 수신했는지 확인하려면 제공 호스트에서 HTTP 서버 로그를 쿼리합니다. 예를 들어 Apache 웹 서버를 사용하여 Ignition 파일을 제공하는 경우 다음 명령을 입력합니다.

```
$ grep -is 'bootstrap.ign' /var/log/httpd/access_log
```

부트 스트랩 Ignition 파일이 수신되면 연결된 **HTTP GET** 로그 메시지에 요청이 성공했음을 나타내는 **200 OK** 성공 상태가 포함됩니다.

- c. Ignition 파일이 수신되지 않은 경우 Ignition 파일이 존재하고 제공 호스트에 대한 적절한 파일 및 웹 서버 권한이 있는지 직접 확인합니다.

- 클라우드 공급자 메커니즘을 사용하여 초기 배포의 일부로 호스트에 Ignition 구성 파일을 삽입하는 경우:
  - a. 부트 스트랩 노드의 콘솔을 확인하고 부트 스트랩 노드 Ignition 파일을 삽입하는 메커니즘이 올바르게 작동하고 있는지 확인합니다.
- 3. 부트 스트랩 노드에 할당된 저장 장치의 가용성을 확인합니다.
- 4. 부트 스트랩 노드에 DHCP 서버의 IP 주소가 할당되었는지 확인합니다.
- 5. 부트 스트랩 노드에서 **bootkube.service** journald 장치 로그를 수집합니다. **<bootstrap\_fqdn>**을 부트 스트랩 노드의 정규화된 도메인 이름으로 바꿉니다.

```
$ ssh core@<bootstrap_fqdn> journalctl -b -f -u bootkube.service
```



**참고**

부트스트랩 노드의 **bootkube.service** 로그는 etcd **connection rejectd** 오류를 출력하고 부트스트랩 서버가 컨트롤 플레인 노드의 etcd에 연결할 수 없음을 나타냅니다. 각 컨트롤 플레인 노드에서 etcd를 시작하고 노드가 클러스터에 가입되면 오류가 중지됩니다.

- 6. 부트 스트랩 노드 컨테이너에서 로그를 수집합니다.
  - a. 부트 스트랩 노드에서 **podman** 을 사용하여 로그를 수집합니다. **<bootstrap\_fqdn>**을 부트 스트랩 노드의 정규화된 도메인 이름으로 바꿉니다.

```
$ ssh core@<bootstrap_fqdn> 'for pod in $(sudo podman ps -a -q); do sudo podman logs $pod; done'
```

- 7. 부트 스트랩 프로세스가 실패한 경우 다음을 확인하십시오.
  - 설치 호스트에서 **api.<cluster\_name>.<base\_domain>**을 확인할 수 있습니다.
  - 로드 밸런서는 부트 스트랩 및 컨트롤 플레인 노드에 포트 6443 연결을 프록시합니다. 프록시 구성이 OpenShift Container Platform 설치 요구 사항을 충족하는지 확인합니다.

**7.1.8. 컨트롤 플레인 노드 설치 문제 조사**

컨트롤 플레인 노드 설치에 문제가 발생하면 컨트롤 플레인 노드 OpenShift Container Platform 소프트웨어 정의 네트워크(SDN) 및 네트워크 Operator 상태를 확인합니다. **kubelet.service, crio.service** journald 장치 로그 및 컨트롤 플레인 노드 컨테이너 로그를 수집하여 컨트롤 플레인 노드 에이전트, CRI-O 컨테이너 런타임, Pod 활동에 대한 가시성을 확보합니다.

**사전 요구 사항**

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.
- 호스트에 대한 SSH 액세스 권한이 있어야 합니다.
- 부트 스트랩 및 컨트롤 플레인 노드의 완전한 도메인 이름이 있어야 합니다.

- HTTP 서버를 사용하여 Ignition 구성 파일을 호스팅하는 경우 HTTP 서버의 정규화된 도메인 이름과 포트 번호가 있어야 합니다. 또한 HTTP 호스트에 대한 SSH 액세스 권한이 있어야 합니다.



## 참고

초기 **kubeadmin** 암호는 설치 호스트의 **<install\_directory>/auth/kubeadmin-password**에서 찾을 수 있습니다.

## 절차

1. 컨트롤 플레인 노드의 콘솔에 액세스할 경우 노드가 로그인 프롬프트에 도달할 때 까지 콘솔을 모니터링합니다. 설치 중에 Ignition 로그 메시지가 콘솔에 출력됩니다.
2. Ignition 파일 설정을 확인합니다.

- HTTP 서버를 사용하여 Ignition 구성 파일을 호스팅하는 경우:

- a. 컨트롤 플레인 노드의 Ignition 파일 URL을 확인합니다. **<http\_server\_fqdn>**을 HTTP 서버의 정규화된 도메인 이름으로 바꿉니다.

```
$ curl -I http://<http_server_fqdn>:<port>/master.ign 1
```

- 1 **-I** 옵션은 헤더만 반환합니다. 지정된 URL에서 Ignition 파일을 사용할 수 있는 경우 명령은 **200 OK** 상태를 반환합니다. 사용할 수 없는 경우 명령은 **404 file not found**를 반환합니다.

- b. Ignition 파일이 컨트롤 플레인 노드에서 수신되었는지 확인하려면 제공 호스트에서 HTTP 서버 로그를 쿼리합니다. 예를 들어 Apache 웹 서버를 사용하여 Ignition 파일을 제공하는 경우 다음을 확인합니다.

```
$ grep -is 'master.ign' /var/log/httpd/access_log
```

마스터 Ignition 파일이 수신되면 연결된 **HTTP GET** 로그 메시지에 요청이 성공했음을 나타내는 **200 OK** 성공 상태가 포함됩니다.

- c. Ignition 파일이 수신되지 않은 경우 제공 호스트에 존재하는지 확인합니다. 적절한 파일 및 웹 서버 권한이 있는지 확인합니다.
- 클라우드 공급자 메커니즘을 사용하여 초기 배포의 일부로 호스트에 Ignition 구성 파일을 삽입하는 경우:
    - a. 컨트롤 플레인 노드의 콘솔을 확인하고 컨트롤 플레인 노드의 Ignition 파일을 삽입하는 메커니즘이 올바르게 작동하고 있는지 확인합니다.

3. 컨트롤 플레인 노드에 할당된 스토리지 장치의 가용성을 확인합니다.
4. 컨트롤 플레인 노드에 DHCP 서버의 IP 주소가 지정되었는지 확인합니다.
5. 컨트롤 플레인 노드 상태를 확인합니다.
  - a. 컨트롤 플레인 노드 상태를 쿼리합니다.

```
$ oc get nodes
```

- b. 컨트롤 플레인 노드 중 하나가 **Ready** 상태에 도달하지 않으면 자세한 노드 설명을 검색합니다.

```
$ oc describe node <master_node>
```



### 참고

설치 문제로 인해 OpenShift Container Platform API가 실행되지 않거나 kubelet이 각 노드에서 실행되지 않는 경우 **oc** 명령을 실행할 수 없습니다.

6. OpenShift Container Platform SDN의 상태를 확인합니다.

- a. **openshift-sdn** 네임스페이스에서 **sdn-controller**, **sdn**, **ovs** 데몬 세트 상태를 검토합니다.

```
$ oc get daemonsets -n openshift-sdn
```

- b. 해당 리소스가 **Not found**로 나열되어 있는 경우 **openshift-sdn** 네임스페이스의 Pod를 검토합니다.

```
$ oc get pods -n openshift-sdn
```

- c. **openshift-sdn** 네임스페이스에서 실패한 OpenShift Container Platform SDN Pod에 대한 로그를 검토합니다.

```
$ oc logs <sdn_pod> -n openshift-sdn
```

7. 클러스터의 네트워크 구성 상태를 확인합니다.

- a. 클러스터의 네트워크 구성이 존재하는지 확인합니다.

```
$ oc get network.config.openshift.io cluster -o yaml
```

- b. 설치 프로그램이 네트워크 구성을 만드는 데 실패한 경우 Kubernetes 매니페스트를 다시 생성하고 메시지 출력을 확인합니다.

```
$ ./openshift-install create manifests
```

- c. **openshift-network-operator** 네임스페이스에서 Pod 상태를 검토하고 CNO(Cluster Network Operator)가 실행 중인지 확인합니다.

```
$ oc get pods -n openshift-network-operator
```

- d. **openshift-network-operator** 네임스페이스에서 네트워크 Operator Pod 로그를 수집합니다.

```
$ oc logs pod/<network_operator_pod_name> -n openshift-network-operator
```

8. 컨트롤 플레인 노드가 부팅된 후 **kubelet.service** journald 장치 로그를 모니터링합니다. 이를 통해 컨트롤 플레인 노드 에이전트 활동을 시각화할 수 있습니다.

- a. **oc**를 사용하여 로그를 검색합니다.

```
$ oc adm node-logs --role=master -u kubelet
```

- b. API가 작동하지 않으면 대신 SSH를 사용하여 로그를 확인합니다. <master-node>. <cluster\_name>.<base\_domain>을 적절한 값으로 바꿉니다.

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> journalctl -b -f -u
kubelet.service
```



### 참고

RHCOS(Red Hat Enterprise Linux CoreOS)를 실행하는 OpenShift Container Platform 4.9 클러스터 노드는 변경할 수 없으며 Operator를 사용하여 클러스터 변경 사항을 적용합니다. SSH를 사용하여 클러스터 노드에 액세스하는 것은 권장되지 않으며 노드는 *accessed* 테인트로 표시됩니다. SSH를 통해 진단 데이터를 수집하기 전에 **oc adm must gather** 및 기타 **oc** 명령을 실행하여 충분한 데이터를 수집할 수 있는지 확인하십시오. 그러나 OpenShift Container Platform API를 사용할 수 없거나 kubelet이 대상 노드에서 제대로 작동하지 않는 경우 **oc** 작업이 영향을 받습니다. 이러한 상황에서 **ssh core@<node>.<cluster\_name>.<base\_domain>**을 사용하여 노드에 액세스할 수 있습니다.

9. 컨트롤 플레인 노드가 부팅 된 후 **crio.service** journald 장치 로그를 검색합니다. 이를 통해 컨트롤 플레인 노드 CRI-O 컨테이너 런타임 활동을 시각화할 수 있습니다.

- a. **oc**를 사용하여 로그를 검색합니다.

```
$ oc adm node-logs --role=master -u crio
```

- b. API가 작동하지 않으면 대신 SSH를 사용하여 로그를 확인합니다.

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> journalctl -b -f -u
crio.service
```

10. 컨트롤 플레인 노드의 **/var/log/** 아래에있는 특정 하위 디렉터리에서 로그를 수집합니다.

- a. **/var/log/** 하위 디렉토리에 포함된 로그 목록을 검색합니다. 다음 예제는 모든 컨트롤 플레인 노드의 **/var/log/openshift-apiserver/**에 있는 파일을 나열합니다.

```
$ oc adm node-logs --role=master --path=openshift-apiserver
```

- b. **/var/log/** 하위 디렉터리 내의 특정 로그를 확인합니다. 다음 예제는 모든 컨트롤 플레인 노드에서 **/var/log/openshift-apiserver/audit.log** 내용을 출력합니다.

```
$ oc adm node-logs --role=master --path=openshift-apiserver/audit.log
```

- c. API가 작동하지 않는 경우 SSH를 사용하여 각 노드의 로그를 확인합니다. 다음은 **/var/log/openshift-apiserver/audit.log** 예제입니다.

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo tail -f
/var/log/openshift-apiserver/audit.log
```

11. SSH를 사용하여 컨트롤 플레인 노드 컨테이너 로그를 확인합니다.

- a. 컨테이너를 나열합니다.

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl ps -a
```

- b. **crictrl**를 사용하여 컨테이너의 로그를 검색합니다.

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictrl logs -f
<container_id>
```

12. 컨트롤 플레인 노드 구성 문제가 발생하는 경우 MCO, MCO 엔드 포인트 및 DNS 레코드가 작동하는지 확인합니다. MCO (Machine Config Operator)는 설치 시 운영 체제 구성을 관리합니다. 또한 시스템 클럭의 정확성과 인증서의 유효성을 확인하십시오.

- a. MCO 엔드 포인트를 사용할 수 있는지 테스트합니다. **<cluster\_name>**을 적절한 값으로 바꿉니다.

```
$ curl https://api-int.<cluster_name>:22623/config/master
```

- b. 엔드 포인트가 응답하지 않는 경우 로드 밸런서 구성을 확인합니다. 엔드 포인트가 포트 22623에서 실행되도록 구성되었는지 확인합니다.

- c. MCO 엔드 포인트의 DNS 레코드가 구성되어 로드 밸런서 문제를 해결하고 있는지 확인합니다.

- i. 정의된 MCO 엔드 포인트 이름에 대한 DNS 조회를 실행합니다.

```
$ dig api-int.<cluster_name> @<dns_server>
```

- ii. 로드 밸런서에서 할당된 MCO IP 주소에 대한 역방향 조회를 실행합니다.

```
$ dig -x <load_balancer_mco_ip_address> @<dns_server>
```

- d. MCO가 부트 스트랩 노드에서 직접 작동하는지 확인합니다. **<bootstrap\_fqdn>**을 부트 스트랩 노드의 정규화된 도메인 이름으로 바꿉니다.

```
$ ssh core@<bootstrap_fqdn> curl https://api-int.<cluster_name>:22623/config/master
```

- e. 시스템 클럭은 부트 스트랩, 마스터 및 작업자 노드 간에 동기화되어야 합니다. 각 노드의 시스템 클럭 참조 시간 및 시간 동기화 통계를 확인합니다.

```
$ ssh core@<node>.<cluster_name>.<base_domain> chronyc tracking
```

- f. 인증서의 유효성을 확인합니다.

```
$ openssl s_client -connect api-int.<cluster_name>:22623 | openssl x509 -noout -text
```

### 7.1.9. etcd 설치 문제 조사

설치 중에 etcd 문제가 발생하면 etcd Pod 상태를 확인하고 etcd Pod 로그를 수집할 수 있습니다. etcd DNS 레코드를 확인하고 컨트롤 플레인 노드에서 DNS 가용성을 확인할 수도 있습니다.

#### 사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.

- 호스트에 대한 SSH 액세스 권한이 있어야 합니다.
- 컨트롤 플레인 노드의 정규화된 도메인 이름이 있어야 합니다.

## 절차

1. etcd Pod의 상태를 확인합니다.

a. **openshift-etcd** 네임스페이스에서 Pod 상태를 검토합니다.

```
$ oc get pods -n openshift-etcd
```

b. **openshift-etcd-operator** 네임스페이스에서 Pod 상태를 검토합니다.

```
$ oc get pods -n openshift-etcd-operator
```

2. 이전 명령으로 나열된 Pod 중 **Running** 또는 **Completed** 상태가 표시되지 않는 경우 Pod의 진단 정보를 수집합니다.

a. Pod의 이벤트를 검토합니다.

```
$ oc describe pod/<pod_name> -n <namespace>
```

b. Pod의 로그를 검사합니다.

```
$ oc logs pod/<pod_name> -n <namespace>
```

c. Pod에 여러 컨테이너가 있는 경우 위의 명령에서 오류가 생성되고 컨테이너 이름이 오류 메시지에 제공됩니다. 각 컨테이너의 로그를 검사합니다.

```
$ oc logs pod/<pod_name> -c <container_name> -n <namespace>
```

3. API가 작동하지 않는 경우 대신 SSH를 사용하여 각 컨트롤 플레인 노드에서 etcd Pod 및 컨테이너 로그를 검토합니다. **<master-node>.<cluster\_name>.<base\_domain>**을 적절한 값으로 바꿉니다.

a. 각 컨트롤 플레인 노드에 etcd Pod를 나열합니다.

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl pods --name=etcd-
```

b. **Ready** 상태가 표시되지 않는 Pod의 경우 Pod 상태를 자세히 검사합니다. **<pod\_id>**를 이전 명령의 출력에 나열된 Pod ID로 교체합니다.

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl inspectp <pod_id>
```

c. Pod와 관련된 컨테이너를 나열합니다.

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl ps | grep '<pod_id>'
```

- d. **Ready** 상태가 표시되지 않는 컨테이너의 경우 컨테이너 상태를 자세히 검사합니다. `<container_id>`를 이전 명령의 출력에 나열된 컨테이너 ID로 바꿉니다.

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl inspect <container_id>
```

- e. **Ready** 상태가 표시되지 않는 컨테이너의 로그를 확인합니다. `<container_id>`를 이전 명령의 출력에 나열된 컨테이너 ID로 바꿉니다.

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl logs -f <container_id>
```



**참고**

RHCOS(Red Hat Enterprise Linux CoreOS)를 실행하는 OpenShift Container Platform 4.9 클러스터 노드는 변경할 수 없으며 Operator를 사용하여 클러스터 변경 사항을 적용합니다. SSH를 사용하여 클러스터 노드에 액세스하는 것은 권장되지 않으며 노드는 *accessed* 테인트로 표시됩니다. SSH를 통해 진단 데이터를 수집하기 전에 **oc adm must gather** 및 기타 **oc** 명령을 실행하여 충분한 데이터를 수집할 수 있는지 확인하십시오. 그러나 OpenShift Container Platform API를 사용할 수 없거나 kubelet이 대상 노드에서 제대로 작동하지 않는 경우 **oc** 작업이 영향을 받습니다. 이러한 상황에서 **ssh core@<node>.<cluster\_name>.<base\_domain>**을 사용하여 노드에 액세스할 수 있습니다.

- 4. 컨트롤 플레인 노드에서 기본 및 보조 DNS 서버 연결을 확인합니다.

**7.1.10. 컨트롤 플레인 노드 kubelet 및 API 서버 문제 조사**

설치 중에 컨트롤 플레인 노드 kubelet 및 API 서버 문제를 조사하려면 DNS, DHCP 및 로드 밸런서 기능을 확인합니다. 또한 인증서가 만료되지 않았는지 확인합니다.

**사전 요구 사항**

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.
- 호스트에 대한 SSH 액세스 권한이 있어야 합니다.
- 컨트롤 플레인 노드의 정규화된 도메인 이름이 있어야 합니다.

**절차**

1. API 서버의 DNS 레코드가 컨트롤 플레인 노드의 kubelet을 **https://api-int.<cluster\_name>.<base\_domain>:6443**으로 전송하는지 확인합니다. 레코드가 로드 밸런서를 참조하는지 확인합니다.
2. 로드 밸런서의 포트 6443 정의가 각 컨트롤 플레인 노드를 참조하는지 확인합니다.
3. DHCP에서 고유한 컨트롤 플레인 노드 호스트 이름이 지정되어 있는지 확인합니다.
4. 각 컨트롤 플레인 노드에서 **kubelet.service** journald 장치 로그를 검사합니다.
  - a. **oc**를 사용하여 로그를 검색합니다.



```
$ oc adm node-logs --role=master -u kubelet
```

- b. API가 작동하지 않으면 대신 SSH를 사용하여 로그를 확인합니다. <master-node>. <cluster\_name>.<base\_domain>을 적절한 값으로 바꿉니다.

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> journalctl -b -f -u kubelet.service
```



### 참고

RHCOS(Red Hat Enterprise Linux CoreOS)를 실행하는 OpenShift Container Platform 4.9 클러스터 노드는 변경할 수 없으며 Operator를 사용하여 클러스터 변경 사항을 적용합니다. SSH를 사용하여 클러스터 노드에 액세스하는 것은 권장되지 않으며 노드는 *accessed* 테인트로 표시됩니다. SSH를 통해 진단 데이터를 수집하기 전에 **oc adm must gather** 및 기타 **oc** 명령을 실행하여 충분한 데이터를 수집할 수 있는지 확인하십시오. 그러나 OpenShift Container Platform API를 사용할 수 없거나 kubelet이 대상 노드에서 제대로 작동하지 않는 경우 **oc** 작업이 영향을 받습니다. 이러한 상황에서 **ssh core@<node>.<cluster\_name>.<base\_domain>**을 사용하여 노드에 액세스할 수 있습니다.

5. 컨트롤 플레인 노드 kubelet 로그에서 인증서 만료 메시지를 확인합니다.

- a. **oc**를 사용하여 로그를 검색합니다.

```
$ oc adm node-logs --role=master -u kubelet | grep -is 'x509: certificate has expired'
```

- b. API가 작동하지 않으면 대신 SSH를 사용하여 로그를 확인합니다. <master-node>. <cluster\_name>.<base\_domain>을 적절한 값으로 바꿉니다.

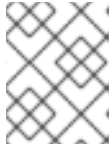
```
$ ssh core@<master-node>.<cluster_name>.<base_domain> journalctl -b -f -u kubelet.service | grep -is 'x509: certificate has expired'
```

### 7.1.11. 작업자 노드 설치 문제 조사

작업자 노드 설치 문제가 발생하는 경우 작업자 노드 상태를 확인할 수 있습니다. **kubelet.service**, **crio.service** journald 장치 로그 및 작업자 노드 컨테이너 로그를 수집하여 작업자 노드 에이전트, CRI-O 컨테이너 런타임, Pod 활동에 대한 가시성을 확보합니다. 또한 Ignition 파일 및 Machine API Operator 기능을 확인할 수 있습니다. 작업자 노드 설치 후 구성이 실패하면 MCO (Machine Config Operator) 및 DNS 기능을 확인합니다. 또한 부트 스트랩, 마스터 및 작업자 노드 간의 시스템 클럭 동기화를 확인하고 인증서의 유효성을 확인할 수도 있습니다.

#### 사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.
- 호스트에 대한 SSH 액세스 권한이 있어야 합니다.
- 부트 스트랩 및 작업자 노드의 완전한 도메인 이름이 있어야 합니다.
- HTTP 서버를 사용하여 Ignition 구성 파일을 호스팅하는 경우 HTTP 서버의 정규화된 도메인 이름과 포트 번호가 있어야 합니다. 또한 HTTP 호스트에 대한 SSH 액세스 권한이 있어야 합니다.



## 참고

초기 **kubeadmin** 암호는 설치 호스트의 `<install_directory>/auth/kubeadmin-password`에서 찾을 수 있습니다.

## 프로세스

1. 작업자 노드의 콘솔에 액세스할 경우 노드가 로그인 프롬프트에 도달할 때 까지 콘솔을 모니터링합니다. 설치 중에 Ignition 로그 메시지가 콘솔에 출력됩니다.
2. Ignition 파일 설정을 확인합니다.

- HTTP 서버를 사용하여 Ignition 구성 파일을 호스팅하는 경우:

- a. 작업자 노드의 Ignition 파일 URL을 확인합니다. `<http_server_fqdn>`을 HTTP 서버의 정규화된 도메인 이름으로 바꿉니다.

```
$ curl -I http://<http_server_fqdn>:<port>/worker.ign 1
```

- 1 **-I** 옵션은 헤더만 반환합니다. 지정된 URL에서 Ignition 파일을 사용할 수 있는 경우 명령은 **200 OK** 상태를 반환합니다. 사용할 수 없는 경우 명령은 **404 file not found**를 반환합니다.

- b. Ignition 파일이 작업자 노드에서 수신된 것을 확인하려면 HTTP 호스트의 HTTP 서버 로그를 쿼리합니다. 예를 들어 Apache 웹 서버를 사용하여 Ignition 파일을 제공하는 경우 다음을 확인합니다.

```
$ grep -is 'worker.ign' /var/log/httpd/access_log
```

작업자 Ignition 파일이 수신되면 연결된 **HTTP GET** 로그 메시지에 요청이 성공했음을 나타내는 **200 OK** 성공 상태가 포함됩니다.

- c. Ignition 파일이 수신되지 않은 경우 제공 호스트에 존재하는지 확인합니다. 적절한 파일 및 웹 서버 권한이 있는지 확인합니다.
- 클라우드 공급자 메커니즘을 사용하여 초기 배포의 일부로 호스트에 Ignition 구성 파일을 삽입하는 경우:
    - a. 작업자 노드의 콘솔을 확인하고 작업자 노드의 Ignition 파일을 삽입하는 메커니즘이 올바르게 작동하고 있는지 확인합니다.

3. 작업자 노드에 할당된 스토리지 장치의 가용성을 확인합니다.
4. 작업자 노드에 DHCP 서버의 IP 주소가 지정되었는지 확인합니다.
5. 작업자 노드 상태를 확인합니다.

- a. 노드의 상태를 쿼리합니다.

```
$ oc get nodes
```

- b. **Ready** 상태를 표시하지 않는 작업자 노드에 대한 자세한 노드 설명을 가져옵니다.

```
$ oc describe node <worker_node>
```



### 참고

설치 문제로 인해 OpenShift Container Platform API가 실행되지 않거나 kubelet이 각 노드에서 실행되지 않는 경우 **oc** 명령을 실행할 수 없습니다.

6. 컨트롤 플레인 노드와 달리 작업자 노드는 Machine API Operator를 사용하여 배포 및 조정됩니다. Machine API Operator의 상태를 확인합니다.

- a. Machine API Operator Pod의 상태를 검토합니다.

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

- b. Machine API Operator Pod의 상태가 **Ready**가 아닌 경우 Pod의 이벤트를 자세히 설명합니다.

```
$ oc describe pod/<machine_api_operator_pod_name> -n openshift-machine-api
```

- c. **machine-api-operator** 컨테이너 로그를 검사합니다. 컨테이너는 **machine-api-operator** Pod 내에서 실행됩니다.

```
$ oc logs pod/<machine_api_operator_pod_name> -n openshift-machine-api -c machine-api-operator
```

- d. 또한 **kube-rbac-proxy** 컨테이너 로그도 검사합니다. 컨테이너는 **machine-api-operator** Pod 내에서도 실행됩니다.

```
$ oc logs pod/<machine_api_operator_pod_name> -n openshift-machine-api -c kube-rbac-proxy
```

7. 작업자 노드가 부팅된 후 작업자 노드에서 **kubelet.service** journald 장치 로그를 모니터링합니다. 이를 통해 작업자 노드 에이전트 활동을 시각화할 수 있습니다.

- a. **oc**를 사용하여 로그를 검색합니다.

```
$ oc adm node-logs --role=worker -u kubelet
```

- b. API가 작동하지 않으면 대신 SSH를 사용하여 로그를 확인합니다. **<worker-node>**. **<cluster\_name>**. **<base\_domain>**을 적절한 값으로 바꿉니다.

```
$ ssh core@<worker-node>.<cluster_name>.<base_domain> journalctl -b -f -u kubelet.service
```



### 참고

RHCOS(Red Hat Enterprise Linux CoreOS)를 실행하는 OpenShift Container Platform 4.9 클러스터 노드는 변경할 수 없으며 Operator를 사용하여 클러스터 변경 사항을 적용합니다. SSH를 사용하여 클러스터 노드에 액세스하는 것은 권장되지 않으며 노드는 **accessed** 테인트로 표시됩니다. SSH를 통해 진단 데이터를 수집하기 전에 **oc adm must gather** 및 기타 **oc** 명령을 실행하여 충분한 데이터를 수집할 수 있는지 확인하십시오. 그러나 OpenShift Container Platform API를 사용할 수 없거나 kubelet이 대상 노드에서 제대로 작동하지 않는 경우 **oc** 작업이 영향을 받습니다. 이러한 상황에서 **ssh core@<node>**. **<cluster\_name>**. **<base\_domain>**을 사용하여 노드에 액세스할 수 있습니다.

8. 작업자 노드가 부팅 된 후 **crio.service** journald 장치 로그를 검색합니다. 이를 통해 작업자 노드 CRI-O 컨테이너 런타임 활동을 시각화할 수 있습니다.

- a. **oc**를 사용하여 로그를 검색합니다.

```
$ oc adm node-logs --role=worker -u crio
```

- b. API가 작동하지 않으면 대신 SSH를 사용하여 로그를 확인합니다.

```
$ ssh core@<worker-node>.<cluster_name>.<base_domain> journalctl -b -f -u crio.service
```

9. 작업자 노드의 **/var/log/** 아래에 있는 특정 하위 디렉토리에서 로그를 수집합니다.

- a. **/var/log/** 하위 디렉토리에 포함된 로그 목록을 검색합니다. 다음 예제는 모든 작업자 노드의 **/var/log/sss**에 있는 파일을 나열합니다.

```
$ oc adm node-logs --role=worker --path=sss
```

- b. **/var/log/** 하위 디렉토리 내의 특정 로그를 확인합니다. 다음 예제는 모든 작업자 노드에서 **/var/log/sss/audit.log** 콘텐츠를 출력합니다.

```
$ oc adm node-logs --role=worker --path=sss/sss.log
```

- c. API가 작동하지 않는 경우 SSH를 사용하여 각 노드의 로그를 확인합니다. 다음은 **/var/log/sss/sss.log**의 예제입니다.

```
$ ssh core@<worker-node>.<cluster_name>.<base_domain> sudo tail -f /var/log/sss/sss.log
```

10. SSH를 사용하여 작업자 노드 컨테이너 로그를 확인합니다.

- a. 컨테이너를 나열합니다.

```
$ ssh core@<worker-node>.<cluster_name>.<base_domain> sudo crictl ps -a
```

- b. **crictl**를 사용하여 컨테이너의 로그를 검색합니다.

```
$ ssh core@<worker-node>.<cluster_name>.<base_domain> sudo crictl logs -f <container_id>
```

11. 작업자 노드 구성 문제가 발생하는 경우 MCO, MCO 엔드 포인트 및 DNS 레코드가 작동하는지 확인합니다. MCO (Machine Config Operator)는 설치 시 운영 체제 구성을 관리합니다. 또한 시스템 클럭의 정확성과 인증서의 유효성을 확인합니다.

- a. MCO 엔드 포인트를 사용할 수 있는지 테스트합니다. **<cluster\_name>**을 적절한 값으로 바꿉니다.

```
$ curl https://api-int.<cluster_name>:22623/config/worker
```

- b. 엔드 포인트가 응답하지 않는 경우 로드 밸런서 구성을 확인합니다. 엔드 포인트가 포트 22623에서 실행되도록 구성되었는지 확인합니다.

- c. MCO 엔드 포인트의 DNS 레코드가 구성되어 로드 밸런서 문제를 해결하고 있는지 확인합니다.

- i. 정의된 MCO 엔드 포인트 이름에 대한 DNS 조회를 실행합니다.

```
$ dig api-int.<cluster_name> @<dns_server>
```

- ii. 로드 밸런서에서 할당된 MCO IP 주소에 대한 역방향 조회를 실행합니다.

```
$ dig -x <load_balancer_mco_ip_address> @<dns_server>
```

- d. MCO가 부트 스트랩 노드에서 직접 작동하는지 확인합니다. **<bootstrap\_fqdn>**을 부트 스트랩 노드의 정규화된 도메인 이름으로 바꿉니다.

```
$ ssh core@<bootstrap_fqdn> curl https://api-int.<cluster_name>:22623/config/worker
```

- e. 시스템 클럭은 부트 스트랩, 마스터 및 작업자 노드 간에 동기화되어야 합니다. 각 노드의 시스템 클럭 참조 시간 및 시간 동기화 통계를 확인합니다.

```
$ ssh core@<node>.<cluster_name>.<base_domain> chronyc tracking
```

- f. 인증서의 유효성을 확인합니다.

```
$ openssl s_client -connect api-int.<cluster_name>:22623 | openssl x509 -noout -text
```

### 7.1.12. 설치 후 Operator 상태 쿼리

설치가 끝나면 Operator의 상태를 확인할 수 있습니다. 사용할 수 없는 Operator에 대한 진단 데이터를 검색합니다. **Pending** 중으로 나열되거나 오류 상태인 Operator Pod의 로그를 검토합니다. 문제가 있는 Pod에서 사용하는 기본 이미지의 유효성을 검사합니다.

#### 사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.

#### 프로세스

1. 설치가 끝나면 클러스터 Operator가 모두 사용 가능한 상태인지 확인합니다.

```
$ oc get clusteroperators
```

2. 필요한 모든 CSR(인증서 서명 요청)이 모두 승인되었는지 확인합니다. 일부 노드는 **Ready** 상태로 이동하지 않을 수 있으며 보류 중인 CSR이 있는 경우 일부 클러스터 Operator를 사용하지 못할 수 있습니다.

- a. CSR의 상태를 검토하고 클러스터에 추가된 각 시스템에 대해 **Pending** 또는 **Approved** 상태의 클라이언트 및 서버 요청이 표시되어 있는지 확인합니다.

```
$ oc get csr
```

#### 출력 예

| NAME      | AGE   | REQUESTOR   | CONDITION        |
|-----------|-------|---|------------------|
| csr-8b2br | 15m   | system:serviceaccount:openshift-machine-config-operator:node-bootstrapper | Pending <b>1</b> |
| csr-8vnps | 15m   | system:serviceaccount:openshift-machine-config-operator:node-bootstrapper | Pending          |
| csr-bfd72 | 5m26s | system:node:ip-10-0-50-126.us-east-2.compute.internal                     | Pending <b>2</b> |
| csr-c57lv | 5m26s | system:node:ip-10-0-95-157.us-east-2.compute.internal                     | Pending          |
| ...       |       |   |                  |

**1** 클라이언트 요청 CSR.

**2** 서버 요청 CSR.

예에서는 두 시스템이 클러스터에 참여하고 있습니다. 목록에는 승인된 CSR이 더 많이 나타날 수도 있습니다.

- b. CSR이 승인되지 않은 경우, 추가된 시스템에 대한 모든 보류 중인 CSR이 **Pending** 상태로 전환된 후 클러스터 시스템의 CSR을 승인합니다.



### 참고

CSR은 교체 주기가 자동으로 만료되므로 클러스터에 시스템을 추가한 후 1시간 이내에 CSR을 승인하십시오. 한 시간 내에 승인하지 않으면 인증서가 교체되고 각 노드에 대해 두 개 이상의 인증서가 표시됩니다. 이러한 인증서를 모두 승인해야 합니다. 첫 번째 CSR을 승인한 후 후속 노드 클라이언트 CSR은 클러스터 **kube-controller-manager**에 의해 자동으로 승인됩니다.



### 참고

베어 메탈 및 기타 사용자 프로비저닝 인프라와 같이 머신 API를 사용하도록 활성화되지 않는 플랫폼에서 실행되는 클러스터의 경우 CSR(Kubelet service Certificate Request)을 자동으로 승인하는 방법을 구현해야 합니다. 요청이 승인되지 않으면 API 서버가 kubelet에 연결될 때 서비스 인증서가 필요하므로 **oc exec, oc rsh, oc logs** 명령을 성공적으로 수행할 수 없습니다. Kubelet 엔드포인트에 연결하는 모든 작업을 수행하려면 이 인증서 승인이 필요합니다. 이 방법은 새 CSR을 감시하고 CSR이 **system:node** 또는 **system:admin** 그룹의 **node-bootstrapper** 서비스 계정에 의해 제출되었는지 확인하고 노드의 ID를 확인합니다.

- 개별적으로 승인하려면 유효한 CSR 각각에 대해 다음 명령을 실행하십시오.

```
$ oc adm certificate approve <csr_name> 1
```

**1** **<csr\_name>**은 현재 CSR 목록에 있는 CSR의 이름입니다.

- 보류 중인 CSR을 모두 승인하려면 다음 명령을 실행하십시오.

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

3. Operator 이벤트를 표시합니다.

```
$ oc describe clusteroperator <operator_name>
```

4. Operator의 네임스페이스 내에서 Operator Pod의 상태를 검토합니다.

```
$ oc get pods -n <operator_namespace>
```

5. 상태가 **Running**이 아닌 Pod에 대한 자세한 설명을 가져옵니다.

```
$ oc describe pod/<operator_pod_name> -n <operator_namespace>
```

6. Pod 로그를 검사합니다.

```
$ oc logs pod/<operator_pod_name> -n <operator_namespace>
```

7. Pod 기본 이미지 관련 문제가 발생하면 기본 이미지 상태를 검토합니다.

a. 문제가 있는 Pod에서 사용되는 기본 이미지의 세부 정보를 가져옵니다.

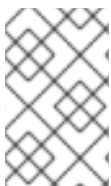
```
$ oc get pod -o "jsonpath={range .status.containerStatuses[*]}{.name}{'\t'}{.state}{'\t'}{.image}{'\n'}{end}" <operator_pod_name> -n <operator_namespace>
```

b. 기본 이미지 릴리스 정보를 나열합니다.

```
$ oc adm release info <image_path>:<tag> --commits
```

### 7.1.13. 실패한 설치에서 로그 수집

설치 프로그램에 SSH 키를 지정한 경우 실패한 설치에 대한 데이터를 수집할 수 있습니다.



#### 참고

실패한 설치에 대한 로그를 수집하는데 사용되는 명령은 실행 중인 클러스터에서 로그를 수집할 때 사용되는 명령과 다릅니다. 실행 중인 클러스터에서 로그를 수집해야 하는 경우 **oc adm must-gather** 명령을 사용하십시오.

#### 사전 요구 사항

- 부트스트랩 프로세스가 완료되기 전에 OpenShift Container Platform 설치에 실패합니다. 부트스트랩 노드가 실행 중이며 SSH를 통해 액세스할 수 있습니다.
- ssh-agent** 프로세스는 컴퓨터에서 활성화되어 있으며 **ssh-agent** 프로세스 및 설치 프로그램에 동일한 SSH 키를 제공하고 있습니다.
- 프로비저닝하는 인프라에 클러스터를 설치하려는 경우 부트스트랩 및 컨트롤 플레인 노드의 정규화된 도메인 이름이 있어야 합니다.

#### 프로세스

- 부트스트랩 및 컨트롤 플레인 시스템에서 설치 로그를 가져오는데 필요한 명령을 생성합니다.

- 설치 관리자 프로비저닝 인프라를 사용한 경우 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

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

1 **installation\_directory**는 **./openshift-install create cluster**를 실행할 때 지정한 디렉터리입니다. 이 디렉터리에는 설치 프로그램이 생성한 OpenShift Container Platform 정의 파일이 포함되어 있습니다.

설치 프로그램이 프로비저닝한 인프라의 경우 설치 프로그램은 클러스터에 대한 정보를 저장하므로 호스트 이름 또는 IP 주소를 지정할 필요가 없습니다.

- 자체적으로 프로비저닝한 인프라를 사용한 경우 설치 프로그램이 포함된 디렉터리로 변경하고 다음 명령을 실행합니다.

```
$ ./openshift-install gather bootstrap --dir <installation_directory> \ 1
--bootstrap <bootstrap_address> \ 2
--master <master_1_address> \ 3
--master <master_2_address> \ 4
--master <master_3_address>" 5
```

1 **installation\_directory**의 경우 **./openshift-install create cluster**를 실행할 때 지정한 것과 동일한 디렉터리를 지정합니다. 이 디렉터리에는 설치 프로그램이 생성한 OpenShift Container Platform 정의 파일이 포함되어 있습니다.

2 **<bootstrap\_address>**는 클러스터 부트스트랩 시스템의 정규화된 도메인 이름 또는 IP 주소입니다.

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



**참고**

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

**출력 예**

```
INFO Pulling debug logs from the bootstrap machine
INFO Bootstrap gather logs captured here "<installation_directory>/log-bundle-
<timestamp>.tar.gz"
```

설치 실패에 대한 Red Hat 지원 케이스를 만들 경우 압축된 로그를 케이스에 포함해야 합니다.

**7.1.14. 추가 리소스**

- OpenShift Container Platform 설치 유형 및 프로세스에 대한 자세한 내용은 [설치 프로세스](#)를 참조하십시오.

**7.2. 노드 상태 확인**



### 7.2.1. 노드 상태, 리소스 사용량 및 구성 확인

클러스터 노드 상태, 리소스 사용량 통계 및 노드 로그를 확인합니다. 또한 개별 노드에서 **kubelet** 상태를 쿼리합니다.

#### 사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.

#### 프로세스

- 클러스터의 모든 노드 이름, 상태 및 역할을 나열합니다.

```
$ oc get nodes
```

- 클러스터 내의 각 노드에 대한 CPU 및 메모리 사용량을 요약합니다.

```
$ oc adm top nodes
```

- 특정 노드의 CPU 및 메모리 사용량을 요약합니다.

```
$ oc adm top node my-node
```

### 7.2.2. 노드에서 kubelet의 상태 쿼리

클러스터 노드 상태, 리소스 사용량 통계 및 노드 로그를 확인할 수 있습니다. 또한 개별 노드에서 **kubelet** 상태를 쿼리할 수 있습니다.

#### 사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- API 서비스가 작동하고 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.

#### 프로세스

1. kubelet은 각 노드에서 systemd 서비스를 사용하여 관리됩니다. 디버그 Pod 내에서 **kubelet** systemd 서비스를 쿼리하여 kubelet의 상태를 검토합니다.
  - a. 노드의 디버그 Pod를 시작합니다.

```
$ oc debug node/my-node
```



#### 참고

컨트롤 플레인 노드에서 **oc debug** 를 실행하는 경우 **/etc/kubernetes/static-pod-resources/kube-apiserver-certs/secrets/node-kubeconfigs** 디렉터리에서 관리 **kubeconfig** 파일을 찾을 수 있습니다.

- b. 디버그 셸 내에서 **/host**를 root 디렉터리로 설정합니다. 디버그 Pod는 Pod 내의 **/host**에 호스트의 루트 파일 시스템을 마운트합니다. root 디렉토리를 **/host**로 변경하면 호스트의 실행 경로에 포함된 바이너리를 실행할 수 있습니다.

```
# chroot /host
```



**참고**

Red Hat Enterprise Linux CoreOS (RHCOS)를 실행하는 OpenShift Container Platform 클러스터 노드는 변경할 수 없으며 Operator를 통해 클러스터 변경 사항을 적용합니다. SSH를 사용하여 클러스터 노드에 액세스하는 것은 권장되지 않으며 노드는 *accessed* 테인트로 표시됩니다. 그러나 OpenShift Container Platform API를 사용할 수 없거나 **kubelet**이 대상 노드에서 제대로 작동하지 않는 경우 **oc** 작업이 영향을 받습니다. 이러한 상황에서 대신 **ssh core @ <node>.<cluster\_name>.<base\_domain>**을 사용하여 노드에 액세스할 수 있습니다.

- c. **kubelet** systemd 서비스가 노드에서 활성화되어 있는지 여부를 확인합니다.

```
# systemctl is-active kubelet
```

- d. 더 자세한 **kubelet.service** 상태 요약을 출력합니다.

```
# systemctl status kubelet
```

**7.2.3. 클러스터 노드의 저널 로그 쿼리**

개별 클러스터 노드의 **/var/log** 내에 **journald** 장치 로그 및 기타 로그를 수집할 수 있습니다.

**사전 요구 사항**

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- API 서비스가 작동하고 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.
- 호스트에 대한 SSH 액세스 권한이 있어야 합니다.

**프로세스**

1. OpenShift Container Platform 클러스터 노드에서 **kubelet journald** 장치 로그를 쿼리합니다. 다음 예제에서는 컨트롤 플레인 노드만 쿼리합니다.

```
$ oc adm node-logs --role=master -u kubelet 1
```

**1** 다른 장치 로그를 쿼리하려면 **kubelet**을 적절하게 대체합니다.

2. 클러스터 노드의 **/var/log/** 아래에있는 특정 하위 디렉터리에서 로그를 수집합니다.
  - a. **/var/log/** 하위 디렉터리에 포함된 로그 목록을 검색합니다. 다음 예제는 모든 컨트롤 플레인 노드의 **/var/log/openshift-apiserver/**에 있는 파일을 나열합니다.

```
$ oc adm node-logs --role=master --path=openshift-apiserver
```

- b. `/var/log/` 하위 디렉터리 내의 특정 로그를 확인합니다. 다음 예제는 모든 컨트롤 플레인 노드에서 `/var/log/openshift-apiserver/audit.log` 내용을 출력합니다.

```
$ oc adm node-logs --role=master --path=openshift-apiserver/audit.log
```

- c. API가 작동하지 않는 경우 SSH를 사용하여 각 노드의 로그를 확인합니다. 다음은 `/var/log/openshift-apiserver/audit.log` 예제입니다.

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo tail -f /var/log/openshift-apiserver/audit.log
```



### 참고

RHCOS(Red Hat Enterprise Linux CoreOS)를 실행하는 OpenShift Container Platform 4.9 클러스터 노드는 변경할 수 없으며 Operator를 사용하여 클러스터 변경 사항을 적용합니다. SSH를 사용하여 클러스터 노드에 액세스하는 것은 권장되지 않으며 노드는 `accessed` 테인트로 표시됩니다. SSH를 통해 진단 데이터를 수집하기 전에 **oc adm must gather** 및 기타 **oc** 명령을 실행하여 충분한 데이터를 수집할 수 있는지 확인하십시오. 그러나 OpenShift Container Platform API를 사용할 수 없거나 kubelet이 대상 노드에서 제대로 작동하지 않는 경우 **oc** 작업이 영향을 받습니다. 이러한 상황에서 **ssh core@<node>.<cluster\_name>.<base\_domain>**을 사용하여 노드에 액세스할 수 있습니다.

## 7.3. CRI-O 컨테이너 런타임 문제 해결

### 7.3.1. CRI-O 컨테이너 런타임 엔진 정보

CRI-O는 Kubernetes 네이티브 컨테이너 런타임 구현으로 운영 체제와 긴밀하게 통합되는 효율적이고 최적화된 Kubernetes 환경을 제공합니다. CRI-O는 컨테이너 실행, 중지 및 다시 시작 기능을 제공합니다.

CRI-O 컨테이너 런타임 엔진은 각 OpenShift Container Platform 클러스터 노드에서 `systemd` 서비스를 사용하여 관리됩니다. 컨테이너 런타임 문제가 발생하면 각 노드에서 **crio** `systemd` 서비스의 상태를 확인합니다. 컨테이너 런타임 문제가 발생하면 노드에서 CRI-O `journald` 장치 로그를 수집합니다.

### 7.3.2. CRI-O 런타임 엔진 상태 확인

각 클러스터 노드에서 CRI-O 컨테이너 런타임 엔진 상태를 확인할 수 있습니다.

#### 사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.

#### 절차

1. 디버그 Pod 내에서 노드의 **crio** `systemd` 서비스를 쿼리하여 CRI-O 상태를 검토합니다.
  - a. 노드의 디버그 Pod를 시작합니다.

```
$ oc debug node/my-node
```

- b. 디버그 셸 내에서 **/host**를 root 디렉터리로 설정합니다. 디버그 Pod는 Pod 내의 **/host**에 호스트의 루트 파일 시스템을 마운트합니다. root 디렉토리를 **/host**로 변경하면 호스트의 실행 경로에 포함된 바이너리를 실행할 수 있습니다.

```
# chroot /host
```



**참고**

RHCOS(Red Hat Enterprise Linux CoreOS)를 실행하는 OpenShift Container Platform 4.9 클러스터 노드는 변경할 수 없으며 Operator를 사용하여 클러스터 변경 사항을 적용합니다. SSH를 사용하여 클러스터 노드에 액세스하는 것은 권장되지 않으며 노드는 *accessed* 테인트로 표시됩니다. 그러나 OpenShift Container Platform API를 사용할 수 없거나 kubelet이 대상 노드에서 제대로 작동하지 않는 경우 **oc** 작업이 영향을 받습니다. 이러한 상황에서 대신 **ssh core @ <node>.<cluster\_name>.<base\_domain>**을 사용하여 노드에 액세스할 수 있습니다.

- c. **crio** systemd 서비스가 노드에서 활성 상태인지 확인합니다.

```
# systemctl is-active crio
```

- d. 더 자세한 **crio.service** 상태 요약을 출력합니다.

```
# systemctl status crio.service
```

**7.3.3. CRI-O journald 장치 로그 수집**

CRI-O 문제가 발생하는 경우 노드에서 CRI-O journald 장치 로그를 얻을 수 있습니다.

**사전 요구 사항**

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- API 서비스가 작동하고 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.
- 컨트롤 플레인 또는 컨트롤 플레인 시스템의 정규화된 도메인 이름이 있어야 합니다.

**절차**

1. CRI-O journald 장치 로그를 수집합니다. 다음 예제는 클러스터 내의 모든 컨트롤 플레인 노드에서 로그를 수집합니다.

```
$ oc adm node-logs --role=master -u crio
```

2. 특정 노드에서 CRI-O journald 장치 로그를 수집합니다.

```
$ oc adm node-logs <node_name> -u crio
```

- API가 작동하지 않으면 대신 SSH를 사용하여 로그를 확인합니다. `<node>.<cluster_name>.<base_domain>`을 적절한 값으로 바꿉니다.

```
$ ssh core@<node>.<cluster_name>.<base_domain> journalctl -b -f -u crio.service
```



### 참고

RHCOS(Red Hat Enterprise Linux CoreOS)를 실행하는 OpenShift Container Platform 4.9 클러스터 노드는 변경할 수 없으며 Operator를 사용하여 클러스터 변경 사항을 적용합니다. SSH를 사용하여 클러스터 노드에 액세스하는 것은 권장되지 않으며 노드는 `accessed` 테인트로 표시됩니다. SSH를 통해 진단 데이터를 수집하기 전에 **oc adm must gather** 및 기타 **oc** 명령을 실행하여 충분한 데이터를 수집할 수 있는지 확인하십시오. 그러나 OpenShift Container Platform API를 사용할 수 없거나 kubelet이 대상 노드에서 제대로 작동하지 않는 경우 **oc** 작업이 영향을 받습니다. 이러한 상황에서 **ssh core@<node>.<cluster\_name>.<base\_domain>**을 사용하여 노드에 액세스할 수 있습니다.

### 7.3.4. CRI-O 스토리지 정리

다음 문제가 발생하는 경우 CRI-O 임시 스토리지를 수동으로 삭제할 수 있습니다.

- 노드는 모든 Pod에서 실행할 수 없으며 다음과 같은 오류가 나타납니다.

```
Failed to create pod sandbox: rpc error: code = Unknown desc = failed to mount container XXX: error recreating the missing symlinks: error reading name of symlink for XXX: open /var/lib/containers/storage/overlay/XXX/link: no such file or directory
```

- 작업 노드에 새 컨테이너를 생성할 수 없으며 "can't stat lower layer" 오류가 나타납니다.

```
can't stat lower layer ... because it does not exist. Going through storage to recreate the missing symlinks.
```

- 클러스터 업그레이드 후 또는 재부팅을 시도하는 경우 노드는 **NotReady** 상태입니다.
- crio**(컨테이너 런타임 구현)가 제대로 작동하지 않습니다.
- 컨테이너 런타임 인스턴스(**crio**)가 작동하지 않기 때문에 **oc debug node/<nodename>**을 사용하여 노드에서 디버그 셸을 시작할 수 없습니다.

CRI-O 스토리지를 완전히 지우고 오류를 해결하려면 다음 프로세스를 따르십시오.

#### 사전 요구 사항:

- cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.

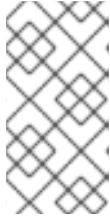
#### 절차

- 노드에서 **cordons**를 사용합니다. 이는 노드가 **Ready** 상태가 되면 워크로드가 예약되지 않도록 하기 위한 것입니다. **SchedulingDisabled**가 상태 섹션에 있으면 예약이 비활성화되어 있음을 알 수 있습니다.

```
$ oc adm cordon <nodename>
```

- 노드를 cluster-admin으로 드레이닝합니다.

```
$ oc adm drain <nodename> --ignore-daemonsets --delete-emptydir-data
```



### 참고

Pod 또는 Pod 템플릿의 **terminationGracePeriodSeconds** 속성은 정상 종료 기간을 제어합니다. 이 속성은 기본적으로 30초이지만 필요에 따라 애플리케이션별로 사용자 지정할 수 있습니다. 90초 이상 설정하면 Pod가 **SIGKILLED** 로 표시되고 성공적으로 종료되지 않을 수 있습니다.

- 노드가 반환되면 SSH 또는 콘솔을 통해 노드에 다시 연결합니다. 그런 다음 root 사용자에게 연결합니다.

```
$ ssh core@node1.example.com
$ sudo -i
```

- kubelet을 수동으로 중지합니다.

```
# systemctl stop kubelet
```

- 컨테이너 및 pod를 중지합니다.

- 다음 명령을 사용하여 HostNetwork에 없는 포드를 중지 **합니다**. 제거는 HostNetwork에 있는 네트워킹 플러그인 pod에 의존하기 때문에 먼저 제거해야 **합니다**.

```
.. for pod in $(crictl pods -q); do if [[ "$(crictl inspectp $pod | jq -r
.status.linux.namespaces.options.network)" != "NODE" ]]; then crictl rmp -f $pod; fi; done
```

- 다른 모든 Pod를 중지합니다.

```
# crictl rmp -fa
```

- crio 서비스를 수동으로 중지합니다.

```
# systemctl stop crio
```

- 이러한 명령을 실행한 후 임시 스토리지를 완전히 초기화할 수 있습니다.

```
# crio wipe -f
```

- crio 및 kubelet 서비스를 시작합니다.

```
# systemctl start crio
# systemctl start kubelet
```

- crio 및 kubelet 서비스가 시작되고 노드가 **Ready** 상태에 있으면 정리가 작동했는지 알 수 있습니다.

```
$ oc get nodes
```

출력 예

```

NAME          STATUS          ROLES    AGE    VERSION
ci-ln-tkbxyft-f76d1-nwvhr-master-1 Ready, SchedulingDisabled master 133m v1.22.0-rc.0+75ee307

```

10. 노드를 예약 가능으로 표시합니다. **SchedulingDisabled** 상태가 더 이상 아닐 때 예약이 활성화 되어 있음을 알 수 있습니다.

```
$ oc adm uncordon <nodename>
```

#### 출력 예

```

NAME          STATUS    ROLES    AGE    VERSION
ci-ln-tkbxyft-f76d1-nwvhr-master-1 Ready      master 133m v1.22.0-rc.0+75ee307

```

## 7.4. 운영 체제 문제 해결

OpenShift Container Platform은 RHCOS에서 실행됩니다. 다음 절차에 따라 운영 체제와 관련된 문제를 해결할 수 있습니다.

### 7.4.1. 커널 크래시 조사

**kexec-tools**에 포함된 **kdump** 서비스는 크래시 덤프 메커니즘을 제공합니다. 이 서비스를 사용하여 이후 분석을 위해 시스템 메모리의 콘텐츠를 저장할 수 있습니다.

**kdump** 서비스는 기술 프리뷰 기능으로만 사용할 수 있습니다. 기술 프리뷰 기능은 Red Hat 프로덕션 서비스 수준 계약(SLA)에서 지원되지 않으며 기능적으로 완전하지 않을 수 있습니다. 따라서 프로덕션 환경에서 사용하는 것은 권장하지 않습니다. 이러한 기능을 사용하면 향후 제품 기능을 조기에 이용할 수 있어 개발 과정에서 고객이 기능을 테스트하고 피드백을 제공할 수 있습니다.

Red Hat 기술 프리뷰 기능의 지원 범위에 대한 자세한 내용은 <https://access.redhat.com/support/offerings/techpreview/>를 참조하십시오.

#### 7.4.1.1. kdump 활성화

RHCOS는 **kexec-tools**와 함께 제공되지만 **kdump**를 사용하려면 수동으로 구성해야 합니다.

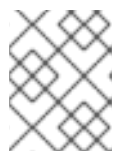
#### 절차

다음 단계를 수행하여 RHCOS에서 **kdump**를 활성화합니다.

1. 첫 번째 커널 부팅 중에 크래시 커널의 메모리를 예약하려면 다음 명령을 입력하여 커널 인수를 제공합니다.

```
# rpm-ostree kargs --append='crashkernel=256M'
```

2. 선택 사항: 기본 로컬 **/var/crash** 위치 대신 크래시 덤프를 네트워크 또는 다른 위치에 작성하려면 **/etc/kdump.conf** 구성 파일을 편집합니다.



#### 참고

LUKS를 사용할 때는 네트워크 덤프가 필요합니다. **kdump**는 LUKS 암호화 장치에서 로컬 크래시 덤프를 지원하지 않습니다.

**kdump** 서비스 구성에 대한 자세한 내용은 `/etc/sysconfig/kdump`, `/etc/kdump.conf`, **kdump.conf** 메뉴얼 페이지의 주석을 참조하십시오. 덤프 대상 구성에 대한 자세한 내용은 [RHEL kdump 문서를 참조하십시오](#).

3. **kdump** systemd 서비스를 활성화합니다.

```
# systemctl enable kdump.service
```

4. 시스템을 재부팅합니다.

```
# systemctl reboot
```

5. **kdump.service**가 성공적으로 시작 및 종료하고 `cat /sys/kernel/kexec_crash_loaded`가 1을 출력하는 것을 확인하여 **kdump**가 크래시 커널을 로드하는지 확인합니다.

### 7.4.1.2. Day-1에 kdump 활성화

**kdump** 서비스는 노드별로 커널 문제를 디버그하도록 사용하도록 설정되어 있습니다. **kdump**를 활성화하는 데 드는 비용이 있고 추가 **kdump** 지원 노드마다 비용이 누적되므로 필요에 따라 **kdump**를 각 노드에서만 사용하도록 설정하는 것이 좋습니다. 각 노드에서 **kdump**를 활성화할 수 있는 비용은 다음과 같습니다.

- 크래시 커널에 대해 예약된 메모리로 인해 사용 가능한 RAM이 줄어듭니다.
- 커널이 코어를 덤프하는 동안 노드를 사용할 수 없습니다.
- 크래시 덤프를 저장하는 데 추가 스토리지 공간이 사용됩니다.
- **kdump** 서비스가 [기술 프리뷰](#)에 있기 때문에 프로덕션에 준비가 되지 않습니다.

**kdump** 서비스 활성화의 단점 및 장단점을 알고 있는 경우 클러스터 전체에서 **kdump**를 활성화할 수 있습니다. 시스템별 머신 구성은 아직 지원되지 않지만 **MachineConfig** 개체의 **systemd** 단위를 통해 이전 단계를 day-1에 수행하고 클러스터의 모든 노드에서 **kdump**를 사용하도록 설정할 수 있습니다.

**MachineConfig** 개체를 생성하고 해당 개체를 클러스터 설정 중에 Ignition에서 사용하는 매니페스트 파일 세트에 삽입할 수 있습니다. Ignition 구성 사용 방법에 대한 자세한 내용은 [설치 → 설치 구성 섹션의 "노드 정의"](#)를 참조하십시오.

### 절차

클러스터 전체 구성을 위한 **MachineConfig** 오브젝트를 생성합니다.

1. **kdump**를 구성하고 활성화하는 Butane 구성 파일 **99-worker-kdump.bu**를 생성합니다.

```
variant: openshift
version: 4.9.0
metadata:
  name: 99-worker-kdump ①
  labels:
    machineconfiguration.openshift.io/role: worker ②
openshift:
  kernel_arguments: ③
    - crashkernel=256M
storage:
  files:
    - path: /etc/kdump.conf ④
```



```

mode: 0644
overwrite: true
contents:
  inline: |
    path /var/crash
    core_collector makedumpfile -l --message-level 7 -d 31

```

```

- path: /etc/sysconfig/kdump 5
  mode: 0644
  overwrite: true
  contents:
    inline: |
      KDUMP_COMMANDLINE_REMOVE="hugepages hugepagesz slub_debug quiet
log_buf_len swiotlb"
      KDUMP_COMMANDLINE_APPEND="irqpoll nr_cpus=1 reset_devices
cgroup_disable=memory mce=off numa=off udev.children-max=2 panic=10 rootflags=nofail
acpi_no_memhotplug transparent_hugepage=never nokaslr novmcoredd hest_disable"
      KEXEC_ARGS="-s"
      KDUMP_IMG="vmlinuz"

systemd:
  units:
    - name: kdump.service
      enabled: true

```

- 1 2** 컨트롤 플레인 노드에 **MachineConfig** 개체를 생성할 때 두 위치 모두에서 **worker**를 **master**로 바꿉니다.
- 3** 크래시 커널용 메모리를 예약하기 위해 커널 인수를 제공합니다. 필요한 경우 다른 커널 인수를 추가할 수 있습니다.
- 4** `/etc/kdump.conf`의 내용을 기본값에서 변경하려면 이 섹션을 포함하고 그에 따라 **inline** 하위 섹션을 수정합니다.
- 5** `/etc/sysconfig/kdump`의 내용을 기본값에서 변경하려면 이 섹션을 포함하고 이에 따라 **inline** 하위 섹션을 수정합니다.

2. Butane을 사용하여 노드에 전달할 구성이 포함된 시스템 구성 YAML 파일 **99-worker-kdump.yaml**을 생성합니다.

```
$ butane 99-worker-kdump.bu -o 99-worker-kdump.yaml
```

3. 클러스터 설정 중에 YAML 파일을 매니페스트에 배치합니다. YAML 파일을 사용하여 클러스터 설정 후 이 **MachineConfig** 오브젝트를 생성할 수도 있습니다.

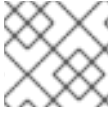
```
$ oc create -f ./99-worker-kdump.yaml
```

### 7.4.1.3. kdump 설정 테스트

kdump에 대한 내용은 RHEL 문서의 [kdump 설정 테스트](#) 섹션을 참조하십시오.

### 7.4.1.4. 코어 덤프 분석

kdump에 대한 내용은 RHEL 문서의 [코어 덤프 분석](#) 섹션을 참조하십시오.



## 참고

별도의 RHEL 시스템에서 vmcore 분석을 수행하는 것이 좋습니다.

### 추가 리소스

- [RHEL에서 kdump 설정](#)
- [kdump용 Linux 커널 문서](#)
- [kdump.conf\(5\)](#) - 사용 가능한 옵션 전체에 대한 문서가 포함된 **/etc/kdump.conf** 설정 파일의 man 페이지
- [kexec\(8\)](#) - **kexec** 패키지의 매뉴얼 페이지
- [kexec 및 kdump에 대한 Red Hat Knowledgebase 문서](#)

## 7.4.2. Ignition 실패 디버깅

머신을 프로비저닝할 수 없는 경우 Ignition이 실패하고 RHCOS가 긴급 셸로 부팅됩니다. 디버깅 정보를 얻으려면 다음 절차를 사용하십시오.

### 절차

1. 다음 명령을 실행하여 어떤 서비스 단위가 실패했는지 표시합니다.

```
$ systemctl --failed
```

2. 선택 사항: 개별 서비스 장치에서 다음 명령을 실행하여 자세한 정보를 확인합니다.

```
$ journalctl -u <unit>.service
```

## 7.5. 네트워크 문제 해결

### 7.5.1. 네트워크 인터페이스 선택 방법

베어 메탈 또는 둘 이상의 NIC(네트워크 인터페이스 컨트롤러)가 있는 가상 머신에 설치할 경우 OpenShift Container Platform이 Kubernetes API 서버와의 통신에 사용하는 NIC는 노드가 부팅될 때 systemd에서 실행하는 **nodeip-configuration.service** 서비스 유닛에 의해 결정됩니다. 서비스는 노드의 네트워크 인터페이스와 서브넷으로 구성된 첫 번째 네트워크 인터페이스를 반복합니다. OpenShift Container Platform 통신에 대해 API 서버의 IP 주소를 호스팅할 수 있습니다.

**nodeip-configuration.service** 서비스에서 올바른 NIC를 확인한 후 서비스는 **/etc/systemd/system/kubelet.service.d/20-nodenet.conf** 파일을 만듭니다. **20-nodenet.conf** 파일은 **KUBELET\_NODE\_IP** 환경 변수를 서비스가 선택한 IP 주소로 설정합니다.

kubelet 서비스가 시작되면 **20-nodenet.conf** 파일에서 환경 변수 값을 읽고 IP 주소를 **--node-ip** kubelet 명령줄 인수 값으로 설정합니다. 결과적으로 kubelet 서비스는 선택한 IP 주소를 노드 IP 주소로 사용합니다.

설치 후 하드웨어 또는 네트워킹을 재구성하는 경우 재부팅 후 **nodeip-configuration.service** 서비스에서 다른 NIC를 선택할 수 있습니다. 경우에 따라 **oc get nodes -o wide** 명령의 출력에서 **INTERNAL-IP** 열을 검토하여 다른 NIC가 선택되었는지 감지할 수 있습니다.

다른 NIC가 선택되어 네트워크 통신이 중단되거나 잘못 구성된 경우 선택 프로세스를 재정의하는 한 가지 전략은 올바른 IP 주소를 명시적으로 설정하는 것입니다. 다음 목록은 높은 수준의 단계 및 고려 사항을 나타냅니다.

- OpenShift Container Platform 통신에 사용할 IP 주소를 결정하는 셸 스크립트를 생성합니다. 스크립트가 `/etc/systemd/system/kubelet.service.d/98-nodenet-override.conf`와 같은 사용자 지정 유닛 파일을 생성하도록 합니다. 사용자 지정 유닛 파일인 `nmcli-nodenet-override.conf`를 사용하여 `KUBELET_NODE_IP` 환경 변수를 IP 주소로 설정합니다.
- `/etc/systemd/system/kubelet.service.d/20-nodenet.conf` 파일을 덮어쓰지 마십시오. 동일한 디렉토리 경로에서 `98-nodenet-override.conf`와 같이 숫자가 더 높은 파일 이름을 지정합니다. 목표는 `20-nodenet.conf` 다음에 사용자 지정 유닛 파일을 실행하고 환경 변수의 값을 재정의하는 것입니다.
- 셸 스크립트를 base64로 인코딩된 문자열로 사용하여 머신 구성 오브젝트를 생성하고 Machine Config Operator를 사용하여 `/usr/local/bin/override-node-ip.sh`와 같은 파일 시스템 경로에 있는 노드에 스크립트를 배포합니다.
- 셸 스크립트가 실행된 후 `systemctl daemon-reload`가 실행되는지 확인합니다. 가장 간단한 방법은 다음 샘플과 같이 시스템 구성에 `ExecStart=systemctl daemon-reload`를 지정하는 것입니다.

### kubelet의 네트워크 인터페이스를 재정의하는 샘플 머신 구성

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-nodenet-override
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;charset=utf-8;base64,<encoded_script>
          mode: 0755
          overwrite: true
          path: /usr/local/bin/override-node-ip.sh
    systemd:
      units:
      - contents: |
          [Unit]
          Description=Override node IP detection
          Wants=network-online.target
          Before=kubelet.service
          After=network-online.target
          [Service]
          Type=oneshot
          ExecStart=/usr/local/bin/override-node-ip.sh
          ExecStart=systemctl daemon-reload
          [Install]
```

```
WantedBy=multi-user.target
enabled: true
name: nodenet-override.service
```

## 7.5.2. Open vSwitch 문제 해결

일부 OVS(Open vSwitch) 문제를 해결하려면 자세한 정보를 포함하도록 로그 수준을 구성해야 할 수 있습니다.

노드에서 로그 수준을 일시적으로 수정하는 경우 다음 예와 같이 노드의 머신 구성 데몬에서 로그 메시지를 받을 수 있습니다.

```
E0514 12:47:17.998892 2281 daemon.go:1350] content mismatch for file /etc/systemd/system/ovs-vswitchd.service: [Unit]
```

불일치와 관련된 로그 메시지를 방지하려면 문제 해결을 완료한 후 로그 수준 변경 사항을 되돌립니다.

### 7.5.2.1. 일시적으로 Open vSwitch 로그 수준 구성

단기 문제 해결을 위해 OVS(Open vSwitch) 로그 수준을 일시적으로 구성할 수 있습니다. 다음 절차에 따라 노드를 재부팅할 필요가 없습니다. 또한 노드를 재부팅할 때 마다 구성 변경 사항이 유지되지 않습니다.

이 절차를 수행하여 로그 수준을 변경하면 **ovs-vswitchd.service**에 대한 콘텐츠 불일치가 있음을 나타내는 머신 구성 데몬에서 로그 메시지를 수신할 수 있습니다. 로그 메시지를 방지하려면 이 절차를 반복하고 로그 수준을 원래 값으로 설정합니다.

#### 사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.

#### 절차

1. 노드의 디버그 Pod를 시작합니다.

```
$ oc debug node/<node_name>
```

2. 디버그 셸 내에서 **/host**를 root 디렉터리로 설정합니다. 디버그 Pod는 Pod 내의 **/host**에 호스트의 root 파일 시스템을 마운트합니다. root 디렉토리를 **/host**로 변경하면 호스트 파일 시스템에서 바이너리를 실행할 수 있습니다.

```
# chroot /host
```

3. OVS 모듈의 현재 syslog 수준을 확인합니다.

```
# ovs-appctl vlog/list
```

다음 예제 출력은 **info**로 설정된 syslog의 로그 수준을 보여줍니다.

#### 출력 예

```
console syslog file
```

```

-----
backtrace      OFF      INFO     INFO
bfd            OFF      INFO     INFO
bond           OFF      INFO     INFO
bridge         OFF      INFO     INFO
bundle         OFF      INFO     INFO
bundles        OFF      INFO     INFO
cfm            OFF      INFO     INFO
collectors     OFF      INFO     INFO
command_line  OFF      INFO     INFO
connmgr        OFF      INFO     INFO
conntack       OFF      INFO     INFO
conntack_tp    OFF      INFO     INFO
coverage       OFF      INFO     INFO
ct_dpif        OFF      INFO     INFO
daemon         OFF      INFO     INFO
daemon_unix    OFF      INFO     INFO
dns_resolve    OFF      INFO     INFO
dpdk           OFF      INFO     INFO
...

```

4. `/etc/systemd/system/ovs-vswitchd.service.d/10-ovs-vswitchd-restart.conf` 파일에서 로그 수준을 지정합니다.

```

Restart=always
ExecStartPre=-/bin/sh -c '/usr/bin/chown -R :${OVS_USER_ID##*:*} /var/lib/openvswitch'
ExecStartPre=-/bin/sh -c '/usr/bin/chown -R :${OVS_USER_ID##*:*} /etc/openvswitch'
ExecStartPre=-/bin/sh -c '/usr/bin/chown -R :${OVS_USER_ID##*:*} /run/openvswitch'
ExecStartPost=-/usr/bin/ovs-appctl vlog/set syslog:dbg
ExecReload=-/usr/bin/ovs-appctl vlog/set syslog:dbg

```

이전 예에서 로그 수준은 **dbg**로 설정됩니다. **syslog:<log\_level>**을 **off,emer,err,warn,info** 또는 **dbg**로 설정하여 마지막 두 행을 변경합니다. **off** 로그 수준은 모든 로그 메시지를 필터링합니다.

5. 서비스를 다시 시작하십시오.

```

# systemctl daemon-reload

# systemctl restart ovs-vswitchd

```

### 7.5.2.2. 영구적으로 Open vSwitch 로그 수준 구성

OVS(Open vSwitch) 로그 수준을 장기적으로 변경할 경우 로그 수준을 영구적으로 변경할 수 있습니다.

#### 사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.

#### 절차

1. 다음 예제와 같은 **MachineConfig** 오브젝트를 사용하여 **99-change-ovs-loglevel.yaml**과 같은 파일을 생성합니다.

```

apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: master ❶
  name: 99-change-ovs-loglevel
spec:
  config:
    ignition:
      version: 3.2.0
    systemd:
      units:
        - dropins:
          - contents: |
              [Service]
              ExecStartPost=-/usr/bin/ovs-appctl vlog/set syslog:dbg ❷
              ExecReload=-/usr/bin/ovs-appctl vlog/set syslog:dbg
            name: 20-ovs-vswitchd-restart.conf
          name: ovs-vswitchd.service

```

- ❶ 이 절차를 수행하여 컨트롤 플레인 노드를 구성한 후 절차를 반복하고 역할을 **worker**로 설정하여 작업자 노드를 구성합니다.
- ❷ **syslog:<log\_level>** 값을 설정합니다. 로그 수준은 **off,emer,err,warn,info** 또는 **dbg**입니다. 값을 **off**로 설정하면 모든 로그 메시지가 필터링됩니다.

2. 머신 구성을 적용합니다.

```
$ oc apply -f 99-change-ovs-loglevel.yaml
```

추가 리소스

- [Machine Config Operator 이해](#)
- [Machine config pool 상태 확인](#)

### 7.5.2.3. Open vSwitch 로그 표시

다음 절차에 따라 OVS(Open vSwitch) 로그를 표시합니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.

절차

- 다음 명령 중 하나를 실행합니다.
  - 클러스터 외부에서 **oc** 명령을 사용하여 로그를 표시합니다.

```
$ oc adm node-logs <node_name> -u ovs-vswitchd
```

- 클러스터의 노드에 로그인한 후 로그를 표시합니다.

```
# journalctl -b -f -u ovs-vswitchd.service
```

노드에 로그인하는 한 가지 방법은 `oc debug node/<node_name>` 명령을 사용하는 것입니다.

## 7.6. OPERATOR 문제 해결

Operator는 OpenShift Container Platform 애플리케이션을 패키징, 배포 및 관리하는 방법입니다. Operator는 소프트웨어 공급 업체의 엔지니어링 팀의 확장 기능으로 OpenShift Container Platform 환경을 모니터링하고 현재 상태를 사용하여 실시간으로 의사 결정을 내립니다. Operator는 업그레이드를 원활하게 처리하고 오류 발생에 자동으로 대응하며 시간을 절약하기 위해 소프트웨어 백업 프로세스를 생략하는 것과 같은 바로가기를 실행하지 않습니다.

OpenShift Container Platform 4.9에는 클러스터가 제대로 작동하는 데 필요한 기본 Operator 세트가 포함되어 있습니다. 이러한 기본 운영자는 CVO (Cluster Version Operator)에 의해 관리됩니다.

클러스터 관리자는 OpenShift Container Platform 웹 콘솔 또는 CLI를 사용하여 OperatorHub에서 애플리케이션 Operator를 설치할 수 있습니다. 그런 다음 Operator를 하나 이상의 네임 스페이스에 가입시켜 클러스터의 개발자가 사용할 수 있도록 합니다. 애플리케이션 Operator는 OLM (Operator Lifecycle Manager)에서 관리합니다.

Operator 문제가 발생하면 Operator 서브스크립션 상태를 확인하십시오. 클러스터 전체에서 Operator Pod 상태를 확인하고 진단을 위해 Operator 로그를 수집합니다.

### 7.6.1. Operator 서브스크립션 상태 유형

서브스크립션은 다음 상태 유형을 보고할 수 있습니다.

표 7.1. 서브스크립션 상태 유형

| 상태                             | 설명                                      |
|--------------------------------|---|
| <b>CatalogSourcesUnhealthy</b> | 해결에 사용되는 일부 또는 모든 카탈로그 소스가 정상 상태가 아닙니다. |
| <b>InstallPlanMissing</b>      | 서브스크립션 설치 계획이 없습니다.                     |
| <b>InstallPlanPending</b>      | 서브스크립션 설치 계획이 설치 대기 중입니다.               |
| <b>InstallPlanFailed</b>       | 서브스크립션 설치 계획이 실패했습니다.                   |



#### 참고

기본 OpenShift Container Platform 클러스터 Operator는 CVO(Cluster Version Operator)에서 관리하며 **Subscription** 오브젝트가 없습니다. 애플리케이션 Operator는 OLM(Operator Lifecycle Manager)에서 관리하며 **Subscription** 오브젝트가 있습니다.

### 7.6.2. CLI를 사용하여 Operator 서브스크립션 상태 보기

CLI를 사용하여 Operator 서브스크립션 상태를 볼 수 있습니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.

절차

1. Operator 서브스크립션을 나열합니다.

```
$ oc get subs -n <operator_namespace>
```

2. **oc describe** 명령을 사용하여 **Subscription** 리소스를 검사합니다.

```
$ oc describe sub <subscription_name> -n <operator_namespace>
```

3. 명령 출력에서 Operator 서브스크립션 조건 유형의 상태에 대한 **Conditions** 섹션을 확인합니다. 다음 예에서 사용 가능한 모든 카탈로그 소스가 정상이므로 **CatalogSourcesUnhealthy** 조건 유형의 상태가 **false**입니다.

출력 예

```
Conditions:
  Last Transition Time: 2019-07-29T13:42:57Z
  Message:             all available catalogsources are healthy
  Reason:              AllCatalogSourcesHealthy
  Status:              False
  Type:                CatalogSourcesUnhealthy
```



참고

기본 OpenShift Container Platform 클러스터 Operator는 CVO(Cluster Version Operator)에서 관리하며 **Subscription** 오브젝트가 없습니다. 애플리케이션 Operator는 OLM(Operator Lifecycle Manager)에서 관리하며 **Subscription** 오브젝트가 있습니다.

7.6.3. CLI를 사용하여 Operator 카탈로그 소스 상태 보기

CLI를 사용하여 Operator 카탈로그 소스의 상태를 볼 수 있습니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.

절차

1. 네임스페이스의 카탈로그 소스를 나열합니다. 예를 들어 클러스터 전체 카탈로그 소스에 사용되는 **openshift-marketplace** 네임스페이스를 확인할 수 있습니다.

```
$ oc get catalogsources -n openshift-marketplace
```

출력 예



| NAME                | DISPLAY             | TYPE | PUBLISHER   | AGE   |
|---------------------|---------------------|------|-------------|-------|
| certified-operators | Certified Operators | grpc | Red Hat     | 55m   |
| community-operators | Community Operators | grpc | Red Hat     | 55m   |
| example-catalog     | Example Catalog     | grpc | Example Org | 2m25s |
| redhat-marketplace  | Red Hat Marketplace | grpc | Red Hat     | 55m   |
| redhat-operators    | Red Hat Operators   | grpc | Red Hat     | 55m   |

2. **oc describe** 명령을 사용하여 카탈로그 소스에 대한 자세한 내용 및 상태를 가져옵니다.

```
$ oc describe catalogsource example-catalog -n openshift-marketplace
```

#### 출력 예

```
Name:      example-catalog
Namespace: openshift-marketplace
...
Status:
  Connection State:
    Address:      example-catalog.openshift-marketplace.svc:50051
    Last Connect: 2021-09-09T17:07:35Z
    Last Observed State: TRANSIENT_FAILURE
  Registry Service:
    Created At:   2021-09-09T17:05:45Z
    Port:         50051
    Protocol:     grpc
    Service Name: example-catalog
    Service Namespace: openshift-marketplace
```

앞의 예제 출력에서 마지막으로 관찰된 상태는 **TRANSIENT\_FAILURE**입니다. 이 상태는 카탈로그 소스에 대한 연결을 설정하는 데 문제가 있음을 나타냅니다.

3. 카탈로그 소스가 생성된 네임스페이스의 Pod를 나열합니다.

```
$ oc get pods -n openshift-marketplace
```

#### 출력 예

| NAME                                  | READY | STATUS           | RESTARTS | AGE   |
|---------------------------------------|-------|------------------|----------|-------|
| certified-operators-cv9nn             | 1/1   | Running          | 0        | 36m   |
| community-operators-6v8lp             | 1/1   | Running          | 0        | 36m   |
| marketplace-operator-86bfc75f9b-jkgbc | 1/1   | Running          | 0        | 42m   |
| example-catalog-bwt8z                 | 0/1   | ImagePullBackOff | 0        | 3m55s |
| redhat-marketplace-57p8c              | 1/1   | Running          | 0        | 36m   |
| redhat-operators-smxx8                | 1/1   | Running          | 0        | 36m   |

카탈로그 소스가 네임스페이스에 생성되면 해당 네임스페이스에 카탈로그 소스의 Pod가 생성됩니다. 위 예제 출력에서 **example-catalog-bwt8z** pod의 상태는 **ImagePullBackOff**입니다. 이 상태는 카탈로그 소스의 인덱스 이미지를 가져오는 데 문제가 있음을 나타냅니다.

4. 자세한 정보는 **oc describe** 명령을 사용하여 Pod를 검사합니다.

```
$ oc describe pod example-catalog-bwt8z -n openshift-marketplace
```

출력 예

```

Name:      example-catalog-bwt8z
Namespace: openshift-marketplace
Priority:   0
Node:      ci-ln-jyryyg2-f76d1-ggdbq-worker-b-vsxd/10.0.128.2
...
Events:
  Type    Reason          Age          From          Message
  ----    -
Normal   Scheduled       48s         default-scheduler Successfully assigned openshift-marketplace/example-catalog-bwt8z to ci-ln-jyryyf2-f76d1-fgdbq-worker-b-vsxd
Normal   AddedInterface  47s         multus        Add eth0 [10.131.0.40/23] from openshift-sdn
Normal   BackOff        20s (x2 over 46s) kubelet       Back-off pulling image "quay.io/example-org/example-catalog:v1"
Warning  Failed         20s (x2 over 46s) kubelet       Error: ImagePullBackOff
Normal   Pulling        8s (x3 over 47s) kubelet       Pulling image "quay.io/example-org/example-catalog:v1"
Warning  Failed         8s (x3 over 47s) kubelet       Failed to pull image "quay.io/example-org/example-catalog:v1": rpc error: code = Unknown desc = reading manifest v1 in quay.io/example-org/example-catalog: unauthorized: access to the requested resource is not authorized
Warning  Failed         8s (x3 over 47s) kubelet       Error: ErrImagePull
    
```

앞의 예제 출력에서 오류 메시지는 권한 부여 문제로 인해 카탈로그 소스의 인덱스 이미지를 성공적으로 가져오지 못한 것으로 표시됩니다. 예를 들어 인덱스 이미지는 로그인 인증 정보가 필요한 레지스트리에 저장할 수 있습니다.

추가 리소스

- [Operator Lifecycle Manager 개념 및 리소스 → 카탈로그 소스](#)
- gRPC 문서: [연결 상태](#)
- [프라이빗 레지스트리에서 Operator용 이미지에 액세스](#)

7.6.4. Operator Pod 상태 쿼리

클러스터 내의 Operator Pod 및 해당 상태를 나열할 수 있습니다. 자세한 Operator Pod 요약은 수집할 수도 있습니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- API 서비스가 작동하고 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.

절차

1. 클러스터에서 실행 중인 Operator를 나열합니다. 출력에는 Operator 버전, 가용성 및 가동 시간 정보가 포함됩니다.

```
$ oc get clusteroperators
```

- Operator의 네임스페이스에서 실행 중인 Operator Pod와 Pod 상태, 재시작, 경과 시간을 표시합니다.

```
$ oc get pod -n <operator_namespace>
```

- 자세한 Operator Pod 요약을 출력합니다.

```
$ oc describe pod <operator_pod_name> -n <operator_namespace>
```

- Operator 문제가 노드와 관련된 경우 해당 노드에서 Operator 컨테이너 상태를 쿼리합니다.

- 노드의 디버그 Pod를 시작합니다.

```
$ oc debug node/my-node
```

- 디버그 셸 내에서 **/host**를 root 디렉터리로 설정합니다. 디버그 Pod는 Pod 내의 **/host**에 호스트의 루트 파일 시스템을 마운트합니다. root 디렉토리를 **/host**로 변경하면 호스트의 실행 경로에 포함된 바이너리를 실행할 수 있습니다.

```
# chroot /host
```



### 참고

RHCOS(Red Hat Enterprise Linux CoreOS)를 실행하는 OpenShift Container Platform 4.9 클러스터 노드는 변경할 수 없으며 Operator를 사용하여 클러스터 변경 사항을 적용합니다. SSH를 사용하여 클러스터 노드에 액세스하는 것은 권장되지 않으며 노드는 *accessed* 데인트로 표시됩니다. 그러나 OpenShift Container Platform API를 사용할 수 없거나 kubelet이 대상 노드에서 제대로 작동하지 않는 경우 **oc** 작업이 영향을 받습니다. 이러한 상황에서 대신 **ssh core @ <node>.<cluster\_name>.<base\_domain>**을 사용하여 노드에 액세스할 수 있습니다.

- 상태 및 관련 Pod ID를 포함하여 노드의 컨테이너에 대한 세부 정보를 나열합니다.

```
# crictl ps
```

- 노드의 특정 Operator 컨테이너에 대한 정보를 나열합니다. 다음 예는 **network-operator** 컨테이너에 대한 정보를 나열합니다.

```
# crictl ps --name network-operator
```

- 디버그 셸을 종료합니다.

## 7.6.5. Operator 로그 수집

Operator 문제가 발생하면 Operator Pod 로그에서 자세한 진단 정보를 수집할 수 있습니다.

### 사전 요구 사항

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

- API 서비스가 작동하고 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.
- 컨트롤 플레인 또는 컨트롤 플레인 시스템의 정규화된 도메인 이름이 있어야 합니다.

## 절차

1. Operator의 네임스페이스에서 실행 중인 Operator Pod와 Pod 상태, 재시작, 경과 시간을 표시합니다.

```
$ oc get pods -n <operator_namespace>
```

2. Operator Pod의 로그를 검토합니다.

```
$ oc logs pod/<pod_name> -n <operator_namespace>
```

Operator Pod에 컨테이너가 여러 개 있는 경우 위 명령에 의해 각 컨테이너의 이름이 포함된 오류가 생성됩니다. 개별 컨테이너의 로그를 쿼리합니다.

```
$ oc logs pod/<operator_pod_name> -c <container_name> -n <operator_namespace>
```

3. API가 작동하지 않는 경우 대신 SSH를 사용하여 각 컨트롤 플레인 노드에서 Operator Pod 및 컨테이너 로그를 검토합니다. **<master-node>.<cluster\_name>.<base\_domain>**을 적절한 값으로 바꿉니다.

- a. 각 컨트롤 플레인 노드에 Pod를 나열합니다.

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl pods
```

- b. **Ready** 상태가 표시되지 않는 Operator Pod의 경우 Pod 상태를 자세히 검사합니다. **<operator\_pod\_id>**를 이전 명령의 출력에 나열된 Operator Pod의 ID로 교체합니다.

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl inspectp <operator_pod_id>
```

- c. Operator Pod와 관련된 컨테이너를 나열합니다.

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl ps --pod= <operator_pod_id>
```

- d. **Ready** 상태가 표시되지 않는 Operator 컨테이너의 경우 컨테이너 상태를 자세히 검사합니다. **<container\_id>**를 이전 명령의 출력에 나열된 컨테이너 ID로 바꿉니다.

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl inspect <container_id>
```

- e. **Ready** 상태가 표시되지 않는 Operator 컨테이너의 로그를 확인합니다. **<container\_id>**를 이전 명령의 출력에 나열된 컨테이너 ID로 바꿉니다.

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl logs -f <container_id>
```



## 참고

RHCOS(Red Hat Enterprise Linux CoreOS)를 실행하는 OpenShift Container Platform 4.9 클러스터 노드는 변경할 수 없으며 Operator를 사용하여 클러스터 변경 사항을 적용합니다. SSH를 사용하여 클러스터 노드에 액세스하는 것은 권장되지 않으며 노드는 `accessed` 테인트로 표시됩니다. SSH를 통해 진단 데이터를 수집하기 전에 **oc adm must gather** 및 기타 **oc** 명령을 실행하여 충분한 데이터를 수집할 수 있는지 확인하십시오. 그러나 OpenShift Container Platform API를 사용할 수 없거나 kubelet이 대상 노드에서 제대로 작동하지 않는 경우 **oc** 작업이 영향을 받습니다. 이러한 상황에서 **ssh core@<node>.<cluster\_name>.<base\_domain>**을 사용하여 노드에 액세스할 수 있습니다.

### 7.6.6. Machine Config Operator가 자동으로 재부팅되지 않도록 비활성화

Machine Config Operator(MCO)에서 구성을 변경한 경우 변경 사항을 적용하려면 RHCOS(Red Hat Enterprise Linux CoreOS)를 재부팅해야 합니다. 구성 변경이 자동이든 수동이든 RHCOS 노드는 일시 중지되지 않는 한 자동으로 재부팅됩니다.

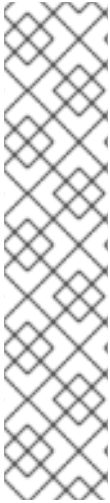


## 참고

다음 수정 사항에서는 노드 재부팅이 트리거되지 않습니다.

- MCO가 다음 변경 사항을 감지하면 노드를 드레이닝하거나 재부팅하지 않고 업데이트를 적용합니다.
  - 머신 구성의 **spec.config.passwd.users.sshAuthorizedKeys** 매개변수에서 SSH 키 변경
  - **openshift-config** 네임 스페이스에서 글로벌 풀 시크릿 또는 풀 시크릿 관련 변경 사항
  - Kubernetes API Server Operator의 **/etc/kubernetes/kubelet-ca.crt** 인증 기관(CA) 자동 교체
- MCO가 **ImageDigestMirrorSet** 또는 **ImageTagMirrorSet** 오브젝트 추가 또는 편집과 같은 **/etc/containers/registries.conf** 파일의 변경 사항을 감지하면 해당 노드를 비우고 변경 사항을 적용하고 노드를 분리합니다. 노드 드레이닝은 다음과 같은 변경에 대해 발생하지 않습니다.
  - 각 미러에 대해 설정된 **pull-from-mirror = "digest-only"** 매개변수를 사용하여 레지스트리를 추가합니다.
  - 레지스트리에 설정된 **pull-from-mirror = "digest-only"** 매개변수를 사용하여 미러를 추가합니다.
  - **unqualified-search-registries** 목록에 항목이 추가되었습니다.

원치 않는 중단을 방지하기 위해 Operator에서 머신 구성을 변경한 후 자동 재부팅되지 않도록 머신 구성 풀(MCP)을 수정할 수 있습니다.



**참고**

MCP를 일시 중지하면 MCO가 연결된 노드에 구성 변경 사항을 적용하지 못합니다. MCP를 일시 중지하면 자동으로 순환된 인증서가 **kube-apiserver-to-kubelet-signer** CA 인증서의 자동 순환을 포함하여 관련 노드로 푸시되지 않습니다. **kube-apiserver-to-kubelet-signer** CA 인증서가 만료되고 MCO가 자동으로 인증서를 갱신하려고 하면 새 인증서가 생성되지만 일시 중지된 MCP의 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.

새 CA 인증서는 설치 날짜로부터 292일 후에 생성되며 해당 날짜로부터 365일 후에 제거됩니다. 다음 자동 CA 인증서 교체를 확인하려면 [Red Hat OpenShift 4의 Understand CA 인증서 자동 갱신](#)을 참조하십시오.

**7.6.6.1. 콘솔을 사용하여 Machine Config Operator가 자동으로 재부팅되지 않도록 비활성화**

MCO(Machine Config Operator)에서 원하지 않는 변경을 방지하기 위해 OpenShift Container Platform 웹 콘솔을 사용하여 MCO가 해당 풀의 노드를 변경하지 않도록 MCP(머신 구성 풀)를 수정할 수 있습니다. 이렇게 하면 일반적으로 MCO 업데이트 프로세스의 일부인 재부팅을 방지할 수 있습니다.



**참고**

[Machine Config Operator 비활성화에 대한 두 번째 참고 사항](#)은 자동으로 재부팅 되지 않습니다.

**사전 요구 사항**

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

**절차**

자동 MCO 업데이트 재부팅을 일시 중지하거나 일시 중지 해제하려면 다음을 수행합니다.

- 자동 재부팅 프로세스를 일시 중지합니다.
  1. **cluster-admin** 역할의 사용자로 OpenShift Container Platform 웹 콘솔에 로그인합니다.
  2. **Compute** → **MachineConfigPools**를 클릭합니다.
  3. **MachineConfigPools** 페이지에서 재부팅을 일시 중지할 노드에 따라 **마스터** 또는 **작업자**를 클릭합니다.
  4. **마스터** 또는 **작업자** 페이지에서 **YAML**을 클릭합니다.
  5. YAML에서 **spec.paused** 필드를 **true**로 업데이트합니다.

**샘플 MachineConfigPool 오브젝트**

```

apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfigPool
...
spec:
...
  paused: true 1
    
```

1 **spec.paused** 필드를 **true**로 업데이트하여 재부팅을 일시 중지합니다.

6. MCP가 일시 중지되었는지 확인하려면 **MachineConfigPools** 페이지로 돌아갑니다. **MachineConfigPools** 페이지에서 **Paused** 열은 수정한 MCP에 대한 **True** 를 보고합니다.

일시 중지된 동안 MCP에 보류 중인 변경 사항이 있는 경우 **Updated** 열은 **False**이고 **Updating** 열은 **False**입니다. **Updated**이 **True**이고 **Updating**이 **False**인 경우 보류 중인 변경 사항이 없습니다.



### 중요

**Updated** 및 **Updating** 열이 모두 **False**인 보류 중인 변경 사항이 있는 경우 최대한 빨리 재부팅할 수 있도록 유지 관리 시간을 예약하는 것이 좋습니다. 자동 재부팅 프로세스를 일시 중지 해제하려면 다음 단계를 사용하여 마지막 재부팅 이후 대기열에 있는 변경 사항을 적용합니다.

- 자동 재부팅 프로세스의 일시 중지를 해제합니다.
  1. **cluster-admin** 역할의 사용자로 OpenShift Container Platform 웹 콘솔에 로그인합니다.
  2. **Compute** → **MachineConfigPools**를 클릭합니다.
  3. **MachineConfigPools** 페이지에서 재부팅을 일시 중지할 노드에 따라 **마스터** 또는 **작업자** 를 클릭합니다.
  4. **마스터** 또는 **작업자** 페이지에서 **YAML**을 클릭합니다.
  5. YAML에서 **spec.paused** 필드를 **false**로 업데이트합니다.

### 샘플 MachineConfigPool 오브젝트

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfigPool
...
spec:
...
  paused: false 1
```

1 재부팅을 허용하도록 **spec.paused** 필드를 **false**로 업데이트합니다.



### 참고

MCP의 일시 정지를 해제하면 MCO는 모든 일시 중지된 변경 사항이 필요에 따라 RHCOS(Red Hat Enterprise Linux CoreOS) 재부팅을 적용합니다.

6. MCP가 일시 중지되었는지 확인하려면 **MachineConfigPools** 페이지로 돌아갑니다. **MachineConfigPools** 페이지에서 **Paused** 열은 수정한 MCP에 대한 **False** 를 보고합니다.

MCP가 보류 중인 변경 사항을 적용하는 경우 **Updated** 열은 **False**이고 **Updating**인 열은 **True**입니다. 업데이트됨이 **True**이고 업데이트 중이 **False**인 경우 추가 변경 사항이 적용되지 않습니다.

### 7.6.6.2. CLI를 사용하여 Machine Config Operator가 자동으로 재부팅되지 않도록 비활성화

MCO(Machine Config Operator)의 변경으로 인한 원치 않는 중단을 방지하려면 OpenShift CLI(oc)를 사용하여 MCP(Machine Config Pool)를 수정하여 MCO가 해당 풀의 노드를 변경하지 못하도록 할 수 있습니다. 이렇게 하면 일반적으로 MCO 업데이트 프로세스의 일부인 재부팅을 방지할 수 있습니다.



#### 참고

Machine Config Operator 비활성화에 대한 두 번째 [참고 사항](#)은 자동으로 재부팅 되지 않습니다.

#### 사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.

#### 절차

자동 MCO 업데이트 재부팅을 일시 중지하거나 일시 중지 해제하려면 다음을 수행합니다.

- 자동 재부팅 프로세스를 일시 중지합니다.
  1. **MachineConfigPool** 사용자 정의 리소스를 업데이트하여 **spec.paused** 필드를 **true**로 설정합니다.

#### 컨트롤 플레인 (마스터) 노드

```
$ oc patch --type=merge --patch='{"spec":{"paused":true}}' machineconfigpool/master
```

#### 작업자 노드

```
$ oc patch --type=merge --patch='{"spec":{"paused":true}}' machineconfigpool/worker
```

2. MCP가 일시 중지되었는지 확인합니다.

#### 컨트롤 플레인 (마스터) 노드

```
$ oc get machineconfigpool/master --template='{{.spec.paused}}'
```

#### 작업자 노드

```
$ oc get machineconfigpool/worker --template='{{.spec.paused}}'
```

#### 출력 예

```
true
```

**spec.paused** 필드가 **true**이고 MCP가 일시 중지되었습니다.

3. MCP에 보류 중인 변경 사항이 있는지 확인합니다.

```
# oc get machineconfigpool
```



## 출력 예

| NAME   | CONFIG  | UPDATED | UPDATING |
|--------|---|---------|----------|
| master | rendered-master-33cf0a1254318755d7b48002c597bf91  | True    | False    |
| worker | rendered-worker-e405a5bdb0db1295acea08bccca33fa60 | False   | False    |

**UPDATED** 열이 **False**이고 **UPDATING**이 **False**이면 보류 중인 변경 사항이 있습니다.

**UPDATED**가 **True**이고 **UPDATING**이 **False**인 경우 보류 중인 변경 사항이 없습니다. 이전 예에서 작업자 노드에 보류 중인 변경 사항이 있습니다. 컨트롤 플레인 노드에는 보류 중인 변경 사항이 없습니다.



## 중요

**Updated** 및 **Updating** 열이 모두 **False**인 보류 중인 변경 사항이 있는 경우 최대한 빨리 재부팅할 수 있도록 유지 관리 기간을 예약하는 것이 좋습니다. 자동 재부팅 프로세스를 일시 중지 해제하려면 다음 단계를 사용하여 마지막 재부팅 이후 대기열에 있는 변경 사항을 적용합니다.

- 자동 재부팅 프로세스의 일시 중지를 해제합니다.
  1. **MachineConfigPool** 사용자 정의 리소스에서 **spec.paused** 필드를 **false**로 업데이트합니다.

## 컨트롤 플레인 (마스터) 노드

```
$ oc patch --type=merge --patch='{"spec":{"paused":false}}' machineconfigpool/master
```

## 작업자 노드

```
$ oc patch --type=merge --patch='{"spec":{"paused":false}}' machineconfigpool/worker
```



## 참고

MCP의 일시 정지를 해제하면 MCO는 일시 중지된 모든 변경 사항을 적용하고 필요에 따라 RHCOS(Red Hat Enterprise Linux CoreOS)를 재부팅합니다.

2. MCP가 일시 중지되지 않았는지 확인합니다.

## 컨트롤 플레인 (마스터) 노드

```
$ oc get machineconfigpool/master --template='{{.spec.paused}}'
```

## 작업자 노드

```
$ oc get machineconfigpool/worker --template='{{.spec.paused}}'
```

## 출력 예

```
false
```

**spec.paused** 필드가 **false**이고 MCP가 일시 중지되지 않습니다.

3. MCP에 보류 중인 변경 사항이 있는지 확인합니다.

```
$ oc get machineconfigpool
```

출력 예

```
NAME CONFIG UPDATED UPDATING
master rendered-master-546383f80705bd5aeaba93 True False
worker rendered-worker-b4c51bb33ccaae6fc4a6a5 False True
```

MCP에서 보류 중인 변경 사항을 적용하는 경우 **UPDATED** 열은 **False**이고 **UPDATING** 열은 **True**입니다. **UPDATED**가 **True**이고 **UPDATING**이 **False**이면 추가 변경 사항이 없습니다. 이전 예에서 MCO는 작업자 노드를 업데이트하고 있습니다.

### 7.6.7. 실패한 서브스크립션 새로 고침

OLM(Operator Lifecycle Manager)에서는 네트워크상에서 액세스할 수 없는 이미지를 참조하는 Operator를 구독하는 경우 **openshift-marketplace** 네임스페이스에 다음 오류로 인해 실패하는 작업을 확인할 수 있습니다.

출력 예

```
ImagePullBackOff for
Back-off pulling image "example.com/openshift4/ose-elasticsearch-operator-
bundle@sha256:6d2587129c846ec28d384540322b40b05833e7e00b25cca584e004af9a1d292e"
```

출력 예

```
rpc error: code = Unknown desc = error pinging docker registry example.com: Get
"https://example.com/v2/": dial tcp: lookup example.com on 10.0.0.1:53: no such host
```

결과적으로 서브스크립션이 이러한 장애 상태에 고착되어 Operator를 설치하거나 업그레이드할 수 없습니다.

서브스크립션, CSV(클러스터 서비스 버전) 및 기타 관련 오브젝트를 삭제하여 실패한 서브스크립션을 새로 고칠 수 있습니다. 서브스크립션을 다시 생성하면 OLM에서 올바른 버전의 Operator를 다시 설치합니다.

#### 사전 요구 사항

- 액세스할 수 없는 번들 이미지를 가져올 수 없는 실패한 서브스크립션이 있습니다.
- 올바른 번들 이미지에 액세스할 수 있는지 확인했습니다.

#### 프로세스

1. Operator가 설치된 네임스페이스에서 **Subscription** 및 **ClusterServiceVersion** 오브젝트의 이름을 가져옵니다.

```
$ oc get sub, csv -n <namespace>
```

출력 예

■

| NAME   | PACKAGE | SOURCE                 | CHANNEL |
|--|---------|------------------------|---------|
| subscription.operators.coreos.com/elasticsearch-operator-operators 5.0 |         | elasticsearch-operator | redhat- |

| NAME  | DISPLAY   | VERSION   |
|---|-----------|-----------|
| REPLACES PHASE<br>clusterserviceversion.operators.coreos.com/elasticsearch-operator.5.0.0-65<br>Elasticsearch Operator 5.0.0-65 | Succeeded | OpenShift |

- 서브스크립션을 삭제합니다.

```
$ oc delete subscription <subscription_name> -n <namespace>
```

- 클러스터 서비스 버전을 삭제합니다.

```
$ oc delete csv <csv_name> -n <namespace>
```

- openshift-marketplace** 네임스페이스에서 실패한 모든 작업 및 관련 구성 맵의 이름을 가져옵니다.

```
$ oc get job,configmap -n openshift-marketplace
```

#### 출력 예

| NAME  | COMPLETIONS | DURATION | AGE   |
|---|-------------|----------|-------|
| job.batch/1de9443b6324e629ddf31fed0a853a121275806170e34c926d69e53a7fcbccb | 1/1         | 26s      | 9m30s |

| NAME  | DATA | AGE   |
|---|------|-------|
| configmap/1de9443b6324e629ddf31fed0a853a121275806170e34c926d69e53a7fcbccb | 3    | 9m30s |

- 작업을 삭제합니다.

```
$ oc delete job <job_name> -n openshift-marketplace
```

이렇게 하면 액세스할 수 없는 이미지를 가져오려는 Pod가 다시 생성되지 않습니다.

- 구성 맵을 삭제합니다.

```
$ oc delete configmap <configmap_name> -n openshift-marketplace
```

- 웹 콘솔에서 OperatorHub를 사용하여 Operator를 다시 설치합니다.

#### 검증

- Operator가 제대로 다시 설치되었는지 확인합니다.

```
$ oc get sub,csv,installplan -n <namespace>
```

## 7.7. POD 문제 조사

OpenShift Container Platform은 호스트에 함께 배포되는 하나 이상의 컨테이너인 Pod의 Kubernetes 개념을 활용합니다. Pod는 OpenShift Container Platform 4.9에서 정의, 배포 및 관리할 수 있는 최소 컴퓨팅 단위입니다.

Pod가 정의되면 컨테이너가 종료될 때까지 또는 제거될 때까지 노드에서 실행되도록 할당됩니다. 정책 및 종료 코드에 따라 Pod는 종료 후 제거되거나 해당 로그에 액세스할 수 있도록 유지됩니다.

Pod 문제 발생 시 가장 먼저 Pod의 상태를 확인합니다. Pod의 명시적인 오류가 발생한 경우에는 Pod의 오류 상태를 확인하여 특정 이미지, 컨테이너 또는 Pod 네트워크 문제를 파악합니다. 오류 상태에 따라 진단 데이터를 수집합니다. Pod 이벤트 메시지와 Pod 및 컨테이너 로그 정보를 확인합니다. 명령줄에서 실행 중인 Pod에 액세스하여 문제를 동적으로 진단하거나 문제가 있는 Pod의 배포 구성을 기반으로 루트 액세스 권한으로 디버그 Pod를 시작합니다.

### 7.7.1. Pod 오류 상태 이해

Pod에서 오류가 발생하면 명시적 오류 상태를 반환하며 **oc get Pods** 출력의 **status** 필드에서 확인할 수 있습니다. Pod 오류 상태에는 이미지, 컨테이너 및 컨테이너 네트워크 관련 오류가 포함됩니다.

다음 표에는 Pod 오류 상태 및 설명이 기재되어 있습니다.

표 7.2. Pod 오류 상태

| Pod 오류 상태                     | 설명   |
|-------------------------------|--|
| <b>ErrImagePull</b>           | 일반 이미지 검색 오류입니다.   |
| <b>ErrImagePullBackOff</b>    | 이미지 검색에 실패하여 백 오프되었습니다.  |
| <b>ErrInvalidImageName</b>    | 지정된 이미지 이름이 잘못되었습니다.   |
| <b>ErrImageInspect</b>        | 이미지 검사에 실패했습니다.  |
| <b>ErrImageNeverPull</b>      | <b>PullPolicy</b> 는 <b>NeverPullImage</b> 에 설정된 대상 이미지는 호스트에서 로컬로 표시되지 않습니다. |
| <b>ErrRegistryUnavailable</b> | 레지스트리에서 이미지 검색을 시도할 때 HTTP 오류가 발생했습니다.                                       |
| <b>ErrContainerNotFound</b>   | 지정된 컨테이너가 선언된 Pod에 존재하지 않거나 kubelet에 의해 관리되지 않습니다.                           |
| <b>ErrRunInitContainer</b>    | 컨테이너 초기화에 실패했습니다.  |
| <b>ErrRunContainer</b>        | Pod의 컨테이너가 정상적으로 시작되지 않았습니다.   |
| <b>ErrKillContainer</b>       | Pod의 컨테이너가 정상적으로 종료되지 않았습니다.   |

| Pod 오류 상태                  | 설명                                      |
|----------------------------|---|
| <b>ErrCrashLoopBackOff</b> | 컨테이너가 종료되었습니다. kubelet은 재시작을 시도하지 않습니다. |
| <b>ErrVerifyNonRoot</b>    | 컨테이너 또는 이미지가 root 권한으로 실행하려고 했습니다.      |
| <b>ErrCreatePodSandbox</b> | Pod 샌드 박스 생성에 실패했습니다.                   |
| <b>ErrConfigPodSandbox</b> | Pod 샌드 박스 구성을 가져오지 못했습니다.               |
| <b>ErrKillPodSandbox</b>   | Pod의 샌드박스가 정상적으로 중지되지 않았습니다.            |
| <b>ErrSetupNetwork</b>     | 네트워크 초기화에 실패했습니다.                       |
| <b>ErrTeardownNetwork</b>  | 네트워크 종료에 실패했습니다.                        |

### 7.7.2. Pod 상태 검토

Pod 상태 및 오류 상태를 쿼리할 수 있습니다. Pod의 관련 배포 구성을 쿼리하고 기본 이미지 가용성을 검토할 수 있습니다.

#### 사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.
- **skopeo**가 설치되어 있어야 합니다.

#### 프로세스

1. 프로젝트로 전환합니다.

```
$ oc project <project_name>
```

2. 네임스페이스 내에서 실행 중인 Pod와 Pod 상태, 오류 상태, 재시작, 경과 시간을 표시합니다.

```
$ oc get pods
```

3. 네임 스페이스가 배포 구성에 의해 관리되는지 확인합니다.

```
$ oc status
```

네임 스페이스가 배포 구성으로 관리되는 경우 출력에 배포 구성 이름과 기본 이미지 참조가 포함됩니다.

- 이전 명령의 출력에서 참조되는 기본 이미지를 검사합니다.

```
$ skopeo inspect docker://<image_reference>
```

- 기본 이미지 참조가 올바르지 않으면 배치 구성에서 참조를 업데이트합니다.

```
$ oc edit deployment/my-deployment
```

- 배포 구성이 완료된 후 변경되면 구성이 자동으로 다시 배포됩니다. 배포가 진행되는 동안 Pod 상태를 확인하여 문제가 해결되었는지 확인합니다.

```
$ oc get pods -w
```

- Pod 실패와 관련된 진단 정보를 보려면 네임스페이스 내의 이벤트를 검토합니다.

```
$ oc get events
```

### 7.7.3. Pod 및 컨테이너 로그 검사

Pod 및 컨테이너 로그에서 명시적 Pod 실패와 관련된 경고 및 오류 메시지를 검사할 수 있습니다. 정책 및 종료 코드에 따라 Pod가 종료된 후에도 Pod 및 컨테이너 로그를 계속 사용할 수 있습니다.

#### 사전 요구 사항

- cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- API 서비스가 작동하고 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.

#### 프로세스

- 특정 Pod의 로그를 쿼리합니다.

```
$ oc logs <pod_name>
```

- Pod에서 특정 컨테이너의 로그를 쿼리합니다.

```
$ oc logs <pod_name> -c <container_name>
```

이전 **oc logs** 명령을 사용하여 검색된 로그는 Pod 또는 컨테이너 내에서 stdout으로 전송된 메시지로 구성됩니다.

- Pod에서 **/var/log/**에 포함된 로그를 검사합니다.

- Pod에서 **/var/log**에 포함된 로그 파일 및 하위 디렉터리를 나열합니다.

```
$ oc exec <pod_name> ls -alh /var/log
```

- Pod에서 **/var/log**에 포함된 특정 로그 파일을 쿼리합니다.

```
$ oc exec <pod_name> cat /var/log/<path_to_log>
```

- c. 특정 컨테이너의 **/var/log**에 포함된 로그 파일 및 하위 디렉토리를 나열합니다.

```
$ oc exec <pod_name> -c <container_name> ls /var/log
```

- d. 특정 컨테이너의 **/var/log**에 포함된 특정 로그 파일을 쿼리합니다.

```
$ oc exec <pod_name> -c <container_name> cat /var/log/<path_to_log>
```

#### 7.7.4. 실행 중인 Pod에 액세스

Pod 내에서 셸을 열거나 포트 전달을 통해 네트워크 액세스 권한을 취득하여 실행 중인 Pod를 동적으로 확인할 수 있습니다.

##### 사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- API 서비스가 작동하고 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.

##### 프로세스

1. 액세스하려는 Pod가 포함된 프로젝트로 전환합니다. 이는 **oc rsh** 명령이 **-n namespace** 옵션을 허용하지 않기 때문에 필요합니다.

```
$ oc project <namespace>
```

2. Pod에서 원격 셸을 시작합니다.

```
$ oc rsh <pod_name> ①
```

- ① Pod에 컨테이너가 여러 개 있는 경우 **-c <container\_name>**을 지정하지 않으면 **oc rsh**는 첫 번째 컨테이너로 기본 설정됩니다.

3. Pod에서 특정 컨테이너로 원격 셸을 시작합니다.

```
$ oc rsh -c <container_name> pod/<pod_name>
```

4. Pod에서 포트로의 포트 전달 세션을 만듭니다.

```
$ oc port-forward <pod_name> <host_port>:<pod_port> ①
```

- ① **Ctrl+C**를 입력하여 포트 전달 세션을 취소합니다.

#### 7.7.5. 루트 액세스 권한으로 디버그 Pod 시작

문제가 있는 Pod 배포 또는 배포 구성에 따라 루트 액세스 권한으로 디버그 Pod를 시작할 수 있습니다. 일반적으로 Pod 사용자는 루트가 아닌 권한으로 실행되지만 임시 루트 권한으로 문제 해결 Pod를 실행하면 문제 해결에 유용할 수 있습니다.

#### 사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- API 서비스가 작동하고 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.

#### 프로세스

1. 배포에 따라 루트 액세스 권한으로 디버그 Pod를 시작합니다.

a. 프로젝트의 배포 이름을 가져옵니다.

```
$ oc get deployment -n <project_name>
```

b. 배포에 따라 루트 권한으로 디버그 Pod를 시작합니다.

```
$ oc debug deployment/my-deployment --as-root -n <project_name>
```

2. 배포 구성에 따라 루트 액세스 권한으로 디버그 Pod를 시작합니다.

a. 프로젝트의 배포 구성 이름을 가져옵니다.

```
$ oc get deploymentconfigs -n <project_name>
```

b. 배포 구성에 따라 루트 권한으로 디버그 Pod를 시작합니다.

```
$ oc debug deploymentconfig/my-deployment-configuration --as-root -n <project_name>
```



#### 참고

대화형 셸을 실행하는 대신 **-<command>**를 이전 **oc debug** 명령에 추가하여 디버그 Pod 내에서 개별 명령을 실행할 수 있습니다.

### 7.7.6. Pod 및 컨테이너 간 파일 복사

Pod 간에 파일을 복사하여 구성 변경을 테스트하거나 진단 정보를 수집할 수 있습니다.

#### 사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- API 서비스가 작동하고 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.

#### 프로세스

1. 파일을 Pod에 복사합니다.



```
$ oc cp <local_path> <pod_name>:/<path> -c <container_name> 1
```

1 -c 옵션이 지정되지 않은 경우 Pod의 첫 번째 컨테이너가 선택됩니다.

2. Pod에서 파일을 복사합니다.

```
$ oc cp <pod_name>:/<path> -c <container_name><local_path> 1
```

1 -c 옵션이 지정되지 않은 경우 Pod의 첫 번째 컨테이너가 선택됩니다.



### 참고

**oc cp** 가 작동하려면 컨테이너 내에서 **tar** 바이너리를 사용할 수 있어야 합니다.

## 7.8. SOURCE-TO-IMAGE 프로세스 문제 해결

### 7.8.1. Source-to-Image 문제 해결을 위한 전략

Source-to-Image (S2I)를 사용하여 재현 가능한 Docker 형식의 컨테이너 이미지를 빌드합니다. 컨테이너 이미지에 애플리케이션 소스 코드를 삽입하고 새 이미지를 어셈블하여 바로 실행할 수 있는 이미지를 만들 수 있습니다. 새 이미지는 기본 이미지 (빌더)와 빌드된 소스를 결합합니다.

S2I 프로세스에서 오류가 발생한 위치를 확인하기 위해 다음 S2I 단계 관련 Pod의 상태를 확인할 수 있습니다.

1. **빌드 구성 단계에서** 빌드 Pod는 기본 이미지 및 애플리케이션 소스 코드에서 애플리케이션 컨테이너 이미지를 만드는 데 사용됩니다.
2. **배포 구성 단계에서** 배포 Pod는 빌드 구성 단계에서 빌드된 애플리케이션 컨테이너 이미지에서 애플리케이션 Pod를 배포하는 데 사용됩니다. 배포 Pod는 서비스 및 경로와 같은 다른 리소스도 배포합니다. 배포 구성은 빌드 구성이 성공한 후에 시작됩니다.
3. **배포 Pod에서 애플리케이션 Pod를 시작한 후** 실행 중인 애플리케이션 Pod 내에서 애플리케이션 오류가 발생할 수 있습니다. 예를 들어 애플리케이션 Pod가 **Running** 상태인 경우에도 애플리케이션이 예상대로 작동하지 않을 수 있습니다. 이 시나리오에서는 실행 중인 애플리케이션 Pod에 액세스하여 Pod 내의 애플리케이션 오류를 조사할 수 있습니다.

S2I 문제를 해결할 때 다음 전략을 따르십시오.

1. 빌드, 배포 및 애플리케이션 Pod 상태 모니터링
2. 문제가 발생한 S2I 프로세스 단계 확인
3. 실패한 단계에 해당하는 로그 확인

### 7.8.2. Source-to-Image 진단 데이터 수집

S2I 툴은 빌드 Pod와 배포 Pod를 순서대로 실행합니다. 배포 Pod는 빌드 단계에서 생성된 애플리케이션 컨테이너 이미지를 기반으로 애플리케이션 Pod를 배포합니다. 빌드, 배포 및 애플리케이션 Pod 상태를 모니터링하여 S2I 프로세스에서 오류가 발생하는 위치를 확인합니다. 다음은 이에 따라 진단 데이터를 수집합니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- API 서비스가 작동하고 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.

프로세스

1. S2I 프로세스 전체에서 Pod의 상태를 확인하고 오류가 발생하는 단계를 확인합니다.

```
$ oc get pods -w 1
```

**1** **Ctrl+C**를 사용하여 명령을 종료할 때까지 **-w**를 사용하여 Pod의 변경 사항을 모니터링합니다.

2. 실패한 Pod 로그에서 오류가 있는지 확인합니다.

- 빌드 Pod가 실패하면 빌드 Pod의 로그를 검토합니다.

```
$ oc logs -f pod/<application_name>-<build_number>-build
```



참고

또는 **oc logs -f bc/<application\_name>**을 사용하여 빌드 구성의 로그를 확인할 수 있습니다. 빌드 구성의 로그에는 빌드 Pod의 로그가 포함됩니다.

- 배포 Pod가 실패하면 배포 Pod의 로그를 검토합니다.

```
$ oc logs -f pod/<application_name>-<build_number>-deploy
```



참고

또는 **oc logs -f dc/<application\_name>**을 사용하여 배포 구성의 로그를 확인할 수 있습니다. 이렇게 하면 배포 Pod가 성공적으로 완료될 때까지 배포 Pod의 로그가 출력됩니다. 이 명령을 배포 Pod가 완료된 후 실행하면 애플리케이션 Pod에서 로그를 출력합니다. 배포 Pod가 완료된 후에도 **oc logs -f pod/<application\_name>-<build\_number>-deploy**를 실행하여 로그에 계속 액세스할 수 있습니다.

- 애플리케이션 Pod가 실패하거나 애플리케이션이 실행 중인 애플리케이션 Pod 내에서 예상대로 작동하지 않으면 애플리케이션 Pod의 로그를 확인합니다.

```
$ oc logs -f pod/<application_name>-<build_number>-<random_string>
```

7.8.3. 애플리케이션 오류 조사를 위한 애플리케이션 진단 데이터 수집

실행 중인 애플리케이션 Pod 내에서 애플리케이션 오류가 발생할 수 있습니다. 이러한 상태에서 다음 전략을 사용하여 진단 정보를 검색할 수 있습니다.

- 애플리케이션 Pod와 관련된 이벤트를 검토합니다.

- OpenShift Logging 프레임워크에서 수집하지 않는 애플리케이션별 로그 파일을 포함하여 애플리케이션 Pod의 로그를 검토합니다.
- 애플리케이션 기능을 대화 형으로 테스트하고 애플리케이션 컨테이너에서 진단 도구를 실행합니다.

### 사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.

### 프로세스

1. 특정 애플리케이션 Pod와 관련된 이벤트를 나열합니다. 다음 예에서는 **my-app-1-akdlg**라는 애플리케이션 Pod의 이벤트를 검색합니다.

```
$ oc describe pod/my-app-1-akdlg
```

2. 애플리케이션 Pod에서 로그를 검토합니다.

```
$ oc logs -f pod/my-app-1-akdlg
```

3. 실행 중인 애플리케이션 Pod 내에서 특정 로그를 쿼리합니다. stdout으로 전송되는 로그는 OpenShift Logging 프레임 워크에서 수집되며 위의 명령의 출력에 포함됩니다. 다음 쿼리는 stdout으로 전송되지 않은 로그에만 필요합니다.

- a. Pod 내에서 루트 권한 없이 애플리케이션 로그에 액세스할 수 있는 경우 다음과 같이 로그 파일을 연결합니다.

```
$ oc exec my-app-1-akdlg -- cat /var/log/my-application.log
```

- b. 애플리케이션 로그를 보기 위해 root 액세스가 필요한 경우 root 권한으로 디버그 컨테이너를 시작한 다음 컨테이너 내에서 로그 파일을 볼 수 있습니다. 프로젝트의 **DeploymentConfig** 개체에서 디버그 컨테이너를 시작합니다. 일반적으로 Pod 사용자는 루트 이외의 권한으로 실행되지만 임시 루트 권한으로 문제 해결 Pod를 실행하면 문제 해결에 유용할 수 있습니다.

```
$ oc debug dc/my-deployment-configuration --as-root -- cat /var/log/my-application.log
```



### 참고

-- **<command>**를 추가하지 않고 **oc debug dc/<deployment\_configuration> --as-root**를 실행하면 디버그 Pod에서 루트 액세스 권한으로 대화형 셸에 액세스할 수 있습니다.

4. 대화형 셸이 있는 애플리케이션 컨테이너에서 대화형으로 애플리케이션 기능을 테스트하고 진단 도구를 실행합니다.

- a. 애플리케이션 컨테이너에서 대화형 셸을 시작합니다.

```
$ oc exec -it my-app-1-akdlg /bin/bash
```

- b. 셸에서 대화형으로 애플리케이션 기능을 테스트합니다. 예를 들어 컨테이너의 엔트리 포인트

명령을 실행하고 결과를 확인할 수 있습니다. 그런 다음 S2I 프로세스를 통해 소스 코드를 업데이트하고 애플리케이션 컨테이너를 다시 빌드하기 전에 명령 줄에서 직접 변경 사항을 테스트합니다.

- c. 컨테이너에서 사용 가능한 진단 바이너리를 실행합니다.



**참고**

일부 진단 바이너리를 실행하려면 root 권한이 필요합니다. 이러한 상황에서는 **oc debug dc/<deployment\_configuration> --as-root**를 실행하여 문제가 있는 Pod의 **DeploymentConfig** 개체에 따라 루트 액세스 권한으로 디버그 Pod를 시작할 수 있습니다. 그런 다음 디버그 Pod 내에서 루트로 진단 바이너리를 실행할 수 있습니다.

- 5. 컨테이너 내에서 진단 바이너리를 사용할 수 없는 경우 **nsenter**를 사용하여 컨테이너의 네임 스페이스에서 호스트의 진단 바이너리를 실행할 수 있습니다. 다음 예제는 호스트의 **IP** 바이너리를 사용하여 컨테이너의 네임 스페이스에서 **ip ad**를 실행합니다.

- a. 대상 노드에서 디버그 세션으로 들어갑니다. 이 단계는 **<node\_name>-debug**라는 디버그 Pod를 인스턴스화합니다.

```
$ oc debug node/my-cluster-node
```

- b. 디버그 셸 내에서 **/host**를 root 디렉터리로 설정합니다. 디버그 Pod는 Pod 내의 **/host**에 호스트의 루트 파일 시스템을 마운트합니다. root 디렉토리를 **/host**로 변경하면 호스트의 실행 경로에 포함된 바이너리를 실행할 수 있습니다.

```
# chroot /host
```



**참고**

RHCOS(Red Hat Enterprise Linux CoreOS)를 실행하는 OpenShift Container Platform 4.9 클러스터 노드는 변경할 수 없으며 Operator를 사용하여 클러스터 변경 사항을 적용합니다. SSH를 사용하여 클러스터 노드에 액세스하는 것은 권장되지 않으며 노드는 *accessed* 테인트로 표시됩니다. 그러나 OpenShift Container Platform API를 사용할 수 없거나 kubelet이 대상 노드에서 제대로 작동하지 않는 경우 **oc** 작업이 영향을 받습니다. 이러한 상황에서 대신 **ssh core @ <node>.<cluster\_name>.<base\_domain>**을 사용하여 노드에 액세스할 수 있습니다.

- c. 대상 컨테이너 ID를 확인합니다.

```
# crictl ps
```

- d. 컨테이너의 프로세스 ID를 확인합니다. 이 예에서 대상 컨테이너 ID는 **a7fe32346b120**입니다.

```
# crictl inspect a7fe32346b120 --output yaml | grep 'pid:' | awk '{print $2}'
```

- e. 호스트의 **ip** 바이너리를 사용하여 컨테이너의 네임 스페이스에서 **ip ad**를 실행합니다. 이 예에서는 컨테이너의 프로세스 ID로 **31150**을 사용합니다. **nsenter** 명령은 대상 프로세스의 네임 스페이스를 입력하고 네임 스페이스에서 명령을 실행합니다. 이 경우 대상 프로세스는 컨테이너의 프로세스 ID이므로 **ip ad** 명령은 호스트의 컨테이너 네임 스페이스에서 실행됩니다.

```
# nsenter -n -t 31150 -- ip ad
```



### 참고

컨테이너의 네임 스페이스에서 호스트의 진단 바이너리를 실행하는 것은 디버그 노드와 같은 권한 있는 컨테이너를 사용하는 경우에만 가능합니다.

#### 7.8.4. 추가 리소스

- S2I 빌드 전략에 대한 자세한 내용은 [S2I\(Source-to-Image\) 빌드](#)를 참조하십시오.

## 7.9. 스토리지 문제 해결

### 7.9.1. 다중 연결 오류 해결

노드가 예기치 않게 중단되거나 종료되면 연결된 RWO(ReadWriteOnce) 볼륨이 노드에서 마운트 해제되어 다른 노드에서 예약된 Pod에서 사용할 수 있습니다.

그러나 오류가 발생한 노드가 연결된 볼륨을 마운트 해제할 수 없기 때문에 새 노드에 마운트할 수 없습니다.

다중 연결 오류가 보고됩니다.

#### 출력 예

```
Unable to attach or mount volumes: unmounted volumes=[sso-mysql-pvol], unattached volumes=[sso-mysql-pvol default-token-x4rzc]: timed out waiting for the condition
Multi-Attach error for volume "pvc-8837384d-69d7-40b2-b2e6-5df86943eef9" Volume is already used by pod(s) sso-mysql-1-ns6b4
```

#### 프로세스

다중 연결 문제를 해결하려면 다음 해결 방법 중 하나를 사용합니다.

- RWX 볼륨을 사용하여 여러 연결을 활성화합니다.  
대부분의 스토리지 솔루션의 경우 RWX (ReadWriteMany) 볼륨을 사용하여 다중 연결 오류를 방지할 수 있습니다.
- RWO 볼륨을 사용할 때 오류가 발생한 노드를 복구하거나 삭제합니다.  
VMware vSphere와 같이 RWX를 지원하지 않는 스토리지의 경우 RWO 볼륨을 대신 사용해야 합니다. 그러나 RWO 볼륨은 여러 노드에 마운트할 수 없습니다.

RWO 볼륨에 다중 연결 오류 메시지가 표시되면 종료되거나 충돌한 노드에서 pod를 강제로 삭제하여 동적 영구 볼륨이 연결된 경우와 같이 중요한 워크로드의 데이터 손실을 방지합니다.

```
$ oc delete pod <old_pod> --force=true --grace-period=0
```

이 명령은 종료되거나 중단된 노드에서 멈춘 볼륨을 6분 후 삭제합니다.

## 7.10. WINDOWS 컨테이너 워크로드 문제 해결

### 7.10.1. Windows Machine Config Operator가 설치되지 않음

WMCO(Windows Machine Config Operator) 설치 프로세스를 완료했지만 Operator가 **InstallWaiting** 단계에서 멈춘 경우 네트워킹 문제로 인해 문제가 발생한 것일 수 있습니다.

WMCO를 사용하려면 OVN-Kubernetes를 사용하는 하이브리드 네트워킹으로 OpenShift Container Platform 클러스터를 구성해야 합니다. WMCO는 사용 가능한 하이브리드 네트워킹이 없으면 설치 프로세스를 완료할 수 없습니다. 이는 다중 운영 체제(OS) 및 OS 변형에서 노드를 관리하는 데 필요합니다. 클러스터를 설치하는 동안 완료해야 합니다.

자세한 내용은 [하이브리드 네트워킹 구성](#)을 참조하십시오.

### 7.10.2. Windows 머신이 컴퓨팅 노드가 되지 않는 이유 조사

Windows 머신이 컴퓨팅 노드가 되지 않는 데에는 다양한 이유가 있습니다. 이 문제를 조사하는 가장 좋은 방법은 WMCO(Windows Machine Config Operator) 로그를 수집하는 것입니다.

#### 사전 요구 사항

- OLM(Operator Lifecycle Manager)을 사용하여 WMCO(Windows Machine Config Operator)를 설치했습니다.
- Windows 머신 세트를 생성했습니다.

#### 프로세스

- 다음 명령을 실행하여 WMCO 로그를 수집합니다.

```
$ oc logs -f deployment/windows-machine-config-operator -n openshift-windows-machine-config-operator
```

### 7.10.3. Windows 노드에 액세스

**oc debug node** 명령을 사용하여 Windows 노드에 액세스할 수 없습니다. 이 명령을 사용하려면 노드에서 권한 있는 Pod를 실행해야 하는데, 이 기능은 아직 Windows에서 지원되지 않습니다. 대신 SSH(Secure Shell) 또는 RDP(Remote Desktop Protocol)를 사용하여 Windows 노드에 액세스할 수 있습니다. 두 방법 모두 SSH bastion이 필요합니다.

#### 7.10.3.1. SSH를 사용하여 Windows 노드에 액세스

SSH(Secure Shell)를 사용하여 Windows 노드에 액세스할 수 있습니다.

#### 사전 요구 사항

- OLM(Operator Lifecycle Manager)을 사용하여 WMCO(Windows Machine Config Operator)를 설치했습니다.
- Windows 머신 세트를 생성했습니다.
- **cloud-private-key** 보안에 사용한 키와 클러스터를 만들 때 사용한 키를 ssh-agent에 추가했습니다. 보안상의 이유로 사용 후에는 ssh-agent에서 키를 제거해야 합니다.
- **ssh-bastion Pod**를 사용하여 Windows 노드에 연결했습니다.

#### 프로세스

- 다음 명령을 실행하여 Windows 노드에 액세스합니다.

```
$ ssh -t -o StrictHostKeyChecking=no -o ProxyCommand='ssh -A -o
StrictHostKeyChecking=no \
  -o ServerAliveInterval=30 -W %h:%p core@$(oc get service --all-namespaces -l run=ssh-
bastion \
  -o go-template="{{ with (index (index .items 0).status.loadBalancer.ingress 0) }}{{ or
.hostname .ip }}{{end}}"' <username>@<windows_node_internal_ip> 1 2
```

- 1 AWS(Amazon Web Services)의 **Administrator** 또는 Microsoft Azure의 **capi**와 같은 클라우드 공급자 사용자 이름을 지정합니다.
- 2 노드의 내부 IP 주소를 지정합니다. 이 IP 주소는 다음 명령을 실행하여 검색할 수 있습니다.

```
$ oc get nodes <node_name> -o jsonpath={.status.addresses[?
(@.type=="InternalIP").address}
```

### 7.10.3.2. RDP를 사용하여 Windows 노드에 액세스

RDP(Remote Desktop Protocol)를 사용하여 Windows 노드에 액세스할 수 있습니다.

#### 사전 요구 사항

- OLM(Operator Lifecycle Manager)을 사용하여 WMCO(Windows Machine Config Operator)를 설치했습니다.
- Windows 머신 세트를 생성했습니다.
- **cloud-private-key** 보안에 사용한 키와 클러스터를 만들 때 사용한 키를 ssh-agent에 추가했습니다. 보안상의 이유로 사용 후에는 ssh-agent에서 키를 제거해야 합니다.
- **ssh-bastion Pod**를 사용하여 Windows 노드에 연결했습니다.

#### 절차

1. 다음 명령을 실행하여 SSH 터널을 설정합니다.

```
$ ssh -L 2020:<windows_node_internal_ip>:3389 \ 1
core@$(oc get service --all-namespaces -l run=ssh-bastion -o go-template="{{ with (index
(index .items 0).status.loadBalancer.ingress 0) }}{{ or .hostname .ip }}{{end}}")
```

- 1 노드의 내부 IP 주소를 지정합니다. 이 IP 주소는 다음 명령을 실행하여 검색할 수 있습니다.

```
$ oc get nodes <node_name> -o jsonpath={.status.addresses[?
(@.type=="InternalIP").address}
```

2. 결과 셸 내에서 SSH를 통해 Windows 노드에 연결하고 다음 명령을 실행하여 사용자 암호를 만듭니다.

```
C:\> net user <username> * 1
```

- 1 클라우드 공급자 사용자 이름을 지정합니다(예: AWS의 **Administrator**, Azure의 **capi**).

이제 RDP 클라이언트를 사용하여 **localhost:2020**에서 Windows 노드에 원격으로 액세스할 수 있습니다.

#### 7.10.4. Windows 컨테이너에 대한 Kubernetes 노드 로그 수집

Windows 컨테이너 로깅은 Linux 컨테이너 로깅과 다르게 작동합니다. Windows 워크로드에 대한 Kubernetes 노드 로그는 기본적으로 **C:\var\logs** 디렉터리로 스트리밍됩니다. 따라서 해당 디렉터리에서 Windows 노드 로그를 수집해야 합니다.

##### 사전 요구 사항

- OLM(Operator Lifecycle Manager)을 사용하여 WMCO(Windows Machine Config Operator)를 설치했습니다.
- Windows 머신 세트를 생성했습니다.

##### 프로세스

1. **C:\var\logs**의 모든 디렉터리에 있는 로그를 보려면 다음 명령을 실행합니다.

```
$ oc adm node-logs -l kubernetes.io/os=windows --path= \
  /ip-10-0-138-252.us-east-2.compute.internal containers \
  /ip-10-0-138-252.us-east-2.compute.internal hybrid-overlay \
  /ip-10-0-138-252.us-east-2.compute.internal kube-proxy \
  /ip-10-0-138-252.us-east-2.compute.internal kubelet \
  /ip-10-0-138-252.us-east-2.compute.internal pods
```

2. 동일한 명령을 사용하여 디렉터리의 파일을 나열하고 개별 로그 파일을 볼 수 있습니다. 예를 들어 kubelet 로그를 보려면 다음 명령을 실행합니다.

```
$ oc adm node-logs -l kubernetes.io/os=windows --path=/kubelet/kubelet.log
```

#### 7.10.5. Windows 애플리케이션 이벤트 로그 수집

kubelet **logs** 끝점의 **Get-WinEvent** shim을 사용하여 Windows 머신에서 애플리케이션 이벤트 로그를 수집할 수 있습니다.

##### 사전 요구 사항

- OLM(Operator Lifecycle Manager)을 사용하여 WMCO(Windows Machine Config Operator)를 설치했습니다.
- Windows 머신 세트를 생성했습니다.

##### 프로세스

- Windows 머신의 이벤트 로그에 로깅하는 모든 애플리케이션의 로그를 보려면 다음을 실행합니다.

```
$ oc adm node-logs -l kubernetes.io/os=windows --path=journal
```

**oc adm must-gather**로 로그를 수집할 때와 같은 명령이 실행됩니다.



-u 플래그를 사용하여 각 서비스를 지정하면 이벤트 로그의 다른 Windows 애플리케이션 로그도 수집할 수 있습니다. 예를 들어 다음 명령을 실행하면 Docker 런타임 서비스에 대한 로그를 수집할 수 있습니다.

```
$ oc adm node-logs -l kubernetes.io/os=windows --path=journal -u docker
```

### 7.10.6. Windows 컨테이너에 대한 Docker 로그 수집

Windows Docker 서비스는 로그를 stdout에 스트리밍하지 않고 대신 Windows의 이벤트 로그에 기록합니다. Docker 이벤트 로그를 확인하여 Windows Docker 서비스로 발생할 수 있는 문제를 조사할 수 있습니다.

#### 사전 요구 사항

- OLM(Operator Lifecycle Manager)을 사용하여 WMCO(Windows Machine Config Operator)를 설치했습니다.
- Windows 머신 세트를 생성했습니다.

#### 프로세스

1. Windows 노드에 SSH로 연결하고 PowerShell로 이동합니다.

```
C:\> powershell
```

2. 다음 명령을 실행하여 Docker 로그를 확인합니다.

```
C:\> Get-EventLog -LogName Application -Source Docker
```

### 7.10.7. 추가 리소스

- [Windows의 컨테이너 문제 해결](#)
- [호스트 및 컨테이너 이미지 불일치 문제 해결](#)
- [Windows의 Docker 문제 해결](#)
- [Windows의 일반적인 Kubernetes 문제](#)

## 7.11. 모니터링 문제 조사

OpenShift Container Platform에는 주요 플랫폼 구성 요소를 모니터링할 수 있는 사전 구성, 사전 설치 및 자체 업데이트 모니터링 스택이 포함되어 있습니다. OpenShift Container Platform 4.9에서 클러스터 관리자는 선택 옵션으로 사용자 정의 프로젝트에 대한 모니터링을 활성화할 수 있습니다.

자체 메트릭을 사용할 수 없거나 Prometheus가 많은 디스크 공간을 사용하는 경우 다음 단계를 수행할 수 있습니다.

### 7.11.1. 사용자 정의 메트릭을 사용할 수 없는 이유 확인

**ServiceMonitor** 리소스를 사용하면 사용자 정의 프로젝트에서 서비스에 의해 노출되는 메트릭을 사용하는 방법을 확인할 수 있습니다. **ServiceMonitor** 리소스를 생성했지만 메트릭 UI에서 해당 메트릭을 볼 수 없는 경우 이 프로세스에 설명된 단계를 수행하십시오.

## 사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.
- 사용자 정의 워크로드에 대한 모니터링을 활성화 및 구성하고 있어야 합니다.
- **user-workload-monitoring-config ConfigMap** 오브젝트가 생성되어 있습니다.
- **ServiceMonitor** 리소스가 생성되어 있습니다.

## 프로세스

1. 서비스 및 **ServiceMonitor** 리소스 구성에서 해당 라벨이 일치하는지 확인합니다.
  - a. 서비스에 정의된 라벨을 가져옵니다. 다음 예제에서는 **ns1** 프로젝트의 **prometheus-example-app** 서비스를 쿼리합니다.

```
$ oc -n ns1 get service prometheus-example-app -o yaml
```

### 출력 예

```
labels:
  app: prometheus-example-app
```

- b. **ServiceMonitor** 리소스의 **matchLabels app** 라벨이 이전 단계의 라벨 출력과 일치하는지 확인합니다.

```
$ oc -n ns1 get servicemonitor prometheus-example-monitor -o yaml
```

### 출력 예

```
spec:
  endpoints:
    - interval: 30s
      port: web
      scheme: http
  selector:
    matchLabels:
      app: prometheus-example-app
```



### 참고

프로젝트 보기 권한이 있는 개발자로서 서비스 및 **ServiceMonitor** 리소스 라벨을 확인할 수 있습니다.

2. **openshift-user-workload-monitoring** 프로젝트에서 **Prometheus Operator**의 로그를 검사합니다.
  - a. **openshift-user-workload-monitoring** 프로젝트의 Pod를 나열합니다.

```
$ oc -n openshift-user-workload-monitoring get pods
```

## 출력 예

```

NAME                                READY STATUS RESTARTS AGE
prometheus-operator-776fcbbd56-2nbfm 2/2   Running 0      132m
prometheus-user-workload-0           5/5   Running 1      132m
prometheus-user-workload-1           5/5   Running 1      132m
thanos-ruler-user-workload-0         3/3   Running 0      132m
thanos-ruler-user-workload-1         3/3   Running 0      132m

```

- b. **prometheus-operator** pod의 **prometheus-operator** 컨테이너에서 로그를 가져옵니다. 다음 예에서 Pod는 **prometheus-operator-776fcbbd56-2nbfm**입니다.

```
$ oc -n openshift-user-workload-monitoring logs prometheus-operator-776fcbbd56-2nbfm -c prometheus-operator
```

서비스 모니터에 문제가 있는 경우 로그에 다음과 유사한 오류가 포함될 수 있습니다.

```

level=warn ts=2020-08-10T11:48:20.906739623Z caller=operator.go:1829
component=prometheusoperator msg="skipping servicemonitor" error="it accesses file
system via bearer token file which Prometheus specification prohibits"
servicemonitor=eagle/eagle namespace=openshift-user-workload-monitoring
prometheus=user-workload

```

3. Prometheus UI에서 프로젝트의 대상 상태를 직접 확인하십시오.

- a. **openshift-user-workload-monitoring** 프로젝트에서 Prometheus 인스턴스에 대한 포트 전달을 설정합니다.

```
$ oc port-forward -n openshift-user-workload-monitoring pod/prometheus-user-workload-0 9090
```

- b. 웹 브라우저에서 <http://localhost:9090/targets>을 열고 Prometheus UI에서 직접 프로젝트의 대상 상태를 확인합니다. 대상과 관련된 오류 메시지를 확인합니다.

4. **openshift-user-workload-monitoring** 프로젝트에서 Prometheus Operator의 디버그 수준 로깅을 구성합니다.

- a. **openshift-user-workload-monitoring** 프로젝트에서 **user-workload-monitoring-config ConfigMap** 오브젝트를 편집합니다.

```
$ oc -n openshift-user-workload-monitoring edit configmap user-workload-monitoring-config
```

- b. **prometheusOperator**의 **logLevel:debug**를 **data / config.yaml** 아래에 추가하여 로그 수준을 **debug**로 설정합니다.

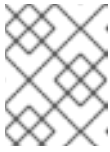
```

apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:

```

```
config.yaml: |
  prometheusOperator:
    logLevel: debug
```

- c. 파일을 저장하여 변경 사항을 적용합니다.



### 참고

**openshift-user-workload-monitoring** 프로젝트의 **prometheus-operator**는 로그 수준 변경을 적용하면 자동으로 다시 시작됩니다.

- d. **openshift-user-workload-monitoring** 프로젝트의 **prometheus-operator** 배포에 **debug** 로그 수준이 적용되었는지 확인합니다.

```
$ oc -n openshift-user-workload-monitoring get deploy prometheus-operator -o yaml |
grep "log-level"
```

### 출력 예

```
- --log-level=debug
```

디버그 수준 로깅은 Prometheus Operator가 수행한 모든 호출을 표시합니다.

- e. **prometheus-operator** Pod가 실행되고 있는지 확인합니다.

```
$ oc -n openshift-user-workload-monitoring get pods
```



### 참고

구성 맵에 인식할 수 없는 Prometheus Operator **loglevel** 값이 포함된 경우 **prometheus-operator** Pod가 재시작되지 않을 수 있습니다.

- f. 디버그 로그를 검토하여 Prometheus Operator에서 **ServiceMonitor** 리소스를 사용하고 있는지 확인합니다. 기타 관련 오류에 대한 로그를 확인합니다.

## 추가 리소스

- [사용자 정의 워크로드 모니터링 구성 맵 생성](#)
- 서비스 모니터 또는 Pod 모니터를 만드는 방법에 대한 자세한 내용은 [서비스 모니터링 방법 지정](#)에서 참조하십시오.

## 7.11.2. Prometheus가 많은 디스크 공간을 소비하는 이유 확인

개발자는 라벨을 생성하여 키-값 쌍의 형식으로 메트릭의 속성을 정의할 수 있습니다. 잠재적인 키-값 쌍의 수는 속성에 사용 가능한 값의 수에 해당합니다. 무제한의 잠재적인 값이 있는 속성을 바인딩되지 않은 속성이라고 합니다. 예를 들어, **customer\_id** 속성은 무제한 가능한 값이 있기 때문에 바인딩되지 않은 속성입니다.

할당된 모든 키-값 쌍에는 고유한 시계열이 있습니다. 라벨에 있는 바인딩되지 않은 많은 속성을 사용하면 생성되는 시계열 수가 기하급수적으로 증가할 수 있습니다. 이는 Prometheus 성능에 영향을 미칠 수 있으며 많은 디스크 공간을 소비할 수 있습니다.

Prometheus가 많은 디스크를 사용하는 경우 다음 조치를 사용할 수 있습니다.

- 수집 중인 스크랩 샘플 수를 확인합니다.
- 가장 많은 시계열을 생성하는 라벨에 대한 자세한 내용은 **Prometheus UI에서 시계열 데이터베이스(TSDB) 상태를 확인**합니다. 여기에는 클러스터 관리자 권한이 필요합니다.
- 사용자 정의 메트릭에 할당되는 바인딩되지 않은 속성의 수를 줄임으로써 **생성되는 고유의 시계열 수를 감소**합니다.



#### 참고

사용 가능한 값의 제한된 집합에 바인딩되는 속성을 사용하면 가능한 키-값 쌍 조합의 수가 줄어듭니다.

- 사용자 정의 프로젝트에서 **스크랩할 수 있는 샘플 수를 제한**합니다. 여기에는 클러스터 관리자 권한이 필요합니다.

#### 사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.

#### 절차

1. 관리자 관점에서 **Observe → Metrics** 로 이동합니다.
2. **Expression** 필드에서 다음 Prometheus Query Language (PromQL) 쿼리를 실행합니다. 이렇게 하면 스크랩 샘플 수가 가장 많은 10개의 메트릭이 반환됩니다.

```
topk(10,count by (job)({__name__=~".+"}))
```

3. 예상 스크랩 샘플 수 보다 많은 메트릭에 할당된 바인딩되지 않은 라벨 값의 수를 조사합니다.
  - **메트릭이 사용자 정의 프로젝트와 관련된 경우** 위크로드에 할당된 메트릭의 키-값 쌍을 확인합니다. 이는 애플리케이션 수준에서 Prometheus 클라이언트 라이브러리를 통해 구현됩니다. 라벨에서 참조되는 바인딩되지 않은 속성의 수를 제한하십시오.
  - **메트릭이 OpenShift Container Platform의 주요 프로젝트와 관련된 경우** [Red Hat Customer Portal](#)에서 Red Hat 지원 케이스를 생성하십시오.
4. Prometheus UI에서 TSDB 상태를 확인합니다.
  - a. 관리자 관점에서 **네트워킹 → 라우트**로 이동합니다.
  - b. **Project:** 목록에서 **openshift-monitoring** 프로젝트를 선택합니다.
  - c. **prometheus-k8s** 열에서 URL을 선택하여 Prometheus UI의 로그인 페이지를 엽니다.
  - d. OpenShift Container Platform 인증 정보를 사용하여 로그인할 **OpenShift로 로그인**을 선택합니다.
  - e. Prometheus UI에서 **상태 → TSDB 상태**로 이동합니다.

- 스크랩 샘플 제한을 설정하고 관련 알림 규칙을 생성하는 방법에 대한 자세한 내용은 [사용자 정의 프로젝트의 스크랩 샘플 제한 설정](#)을 참조하십시오.

## 7.12. OPENSIFT CLI (oc) 문제 진단

### 7.12.1. OpenShift CLI (oc) 로그 수준 이해

OpenShift CLI (**oc**)를 사용하면 터미널에서 애플리케이션을 생성하고 OpenShift Container Platform 프로젝트를 관리할 수 있습니다.

**oc** 명령 관련 문제가 발생하면 **oc** 로그 수준을 높여서 명령으로 생성된 API 요청, API 응답 및 **curl** 요청 세부 정보를 출력합니다. 이를 통해 특정 **oc** 명령의 기본 작업에 대한 세부적인 보기를 통해 오류 특성에 대한 통찰력을 제공할 수 있습니다.

**oc** 로그 수준은 1에서 10까지 있습니다. 다음 표에서는 **oc** 로그 수준을 설명합니다.

표 7.3. OpenShift CLI (oc) 로그 수준

| 로그 수준 | 설명   |
|-------|--|
| 1~5   | stderr에 대한 추가 로깅이 없습니다.  |
| 6     | stderr에 API 요청을 기록합니다.   |
| 7     | stderr에 API 요청 및 헤더를 기록합니다.  |
| 8     | stderr에 API 요청, 헤더 및 본문과 API 응답 헤더 및 본문을 기록합니다.                      |
| 9     | stderr에 API 요청, 헤더 및 본문, API 응답 헤더 및 본문, <b>curl</b> 요청을 기록합니다.      |
| 10    | stderr에 API 요청, 헤더 및 본문, API 응답 헤더 및 본문, <b>curl</b> 요청을 상세하게 기록합니다. |

### 7.12.2. OpenShift CLI (oc) 로그 수준 지정

명령의 로그 수준을 높여 OpenShift CLI (**oc**) 문제를 조사할 수 있습니다.

#### 사전 요구 사항

- OpenShift CLI(**oc**)를 설치합니다.

#### 절차

1. **oc** 명령을 실행할 때 **oc** 로그 레벨을 지정합니다.

```
$ oc <options> --loglevel <log_level>
```

2. 일반적으로 OpenShift Container Platform 사용자의 현재 세션 토큰은 필요한 경우 로깅된 **curl** 요청에 포함됩니다. **oc** 명령의 기본 프로세스 측면을 단계별로 테스트할 때 사용하기 위해 현재 사용자의 세션 토큰을 수동으로 가져올 수도 있습니다.

---

■ \$ oc whoami -t