OpenShift Container Platform 4.6

베어 메탈에 설치 프로그램이 프로비저닝 한 클러스터 배포

OpenShift Container Platform 베어메탈 클러스터 설치

Last Updated: 2022-09-18

OpenShift Container Platform 4.6 베어 메탈에 설치 프로그램이 프로비저닝 한 클러스터 배포

OpenShift Container Platform 베어메탈 클러스터 설치

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

법적 공지

초록

이 문서에서는 설치 프로그램을 사용하여 프로비저닝된 클러스터의 베어메탈 인프라에 OpenShift Container Platform 클러스터를 설치하는 방법을 설명합니다.

차례

1장. 베어 메탈에 설치 프로그램이 프로비저닝 한 클러스터 배포 4
1.1. 개요 4
1.2. 사전 요구 사항 4
1.2.1. 노드 요구 사항 5
1.2.2. 네트워크 요구 사항 5
1.2.3. 노드 설정 7
1.2.4. 대역 외 관리 9
1.2.5. 설치에 필요한 데이터 9
1.2.6. 노드의 유효성 검사 체크리스트 9
1.3. OPENSHIFT 설치를 위한 환경 설정 10
1.3.1. 프로비저너 노드에 RHEL 설치 10
1.3.2. OpenShift Container Platform 설치를 위한 provisioner 노드 준비 10
1.3.3. OpenShift Container Platform 설치 프로그램 검색 12
1.3.4. OpenShift Container Platform 설치 프로그램 추출 13
1.3.5. RHCOS 이미지 캐시 만들기 (선택 사항) 13
1.3.6. 구성 파일 15
1.3.6.1. install-config.yaml 파일을 생성합니다. 15
1.3.6.2. install-config.yaml 파일 내에서 프록시 설정 (선택 사항) 17
1.3.6.3. provisioning 네트워크가 없는 경우 install-config.yaml 파일 변경 (선택 사항) 17
1.3.6.4. 추가 install-config 매개 변수 18
1.3.6.5. BMC 주소 지정 21
1.3.6.6. 루트 장치 팁 24
1.3.6.7. OpenShift Container Platform 매니페스트 만들기 25
1.3.7. 비 연결 레지스트리 만들기 (선택 사항) 25
1.3.7.1. 미러링된 레지스트리를 호스팅할 레지스트리 노드 준비 (선택 사항) 26
1.3.7.2. 자체 서명 인증서 생성 (선택 사항) 26
1.3.7.3. 레지스트리 podman 컨테이너 만들기 (선택 사항) 27
1.3.7.4. 풀 시크릿(pull-secret) 복사 및 업데이트 (선택 사항) 27
1.3.7.5. 저장소 미러링 (선택 사항) 28
1.3.7.6. 비 연결 레지스트리를 사용하도록 install-config.yaml 파일 변경 (선택 사항) 29
1.3.8. 작업자 노드에 라우터 배치 29
1.3.9. 설치 확인 체크리스트 30
1.3.10. OpenShift Container Platform 설치 프로그램을 통해 클러스터 배포 30
1.3.11. 설치 후 30
1.3.12. 베어 메탈에 클러스터 재설치 준비 31
1.4. 문제 해결 31
1.4.1. 설치 프로그램 워크 플로우 문제 해결 31
1.4.2. install-config.yaml 문제 해결 33
1.4.3. 부트스트랩 VM 문제 34
1.4.3.1. 부트스트랩 VM은 클러스터 노드를 부팅할 수 없습니다. 35
1.4.3.2. 로그 검사 36
1.4.4. 클러스터 노드는 PXE 부팅 불가능 37
1.4.5. API에 액세스 불가능 37
1.4.6. 이전 설치 정리 38
1.4.7. 레지스트리 생성 문제 39
1.4.8. 기타 문제 39
1.4.8.1. runtime network not ready 오류 해결 39
1.4.8.2. DHCP를 통해 올바른 IPv6 주소를 얻지 못하는 클러스터 노드 40
1.4.8.3. DHCP를 통해 올바른 호스트 이름을 얻지 못하는 클러스터 노드 41
1.4.8.4. 루트가 엔드 포인트에 도달하지 않음 42
1.4.8.5. Firstboot 동안 Ignition 실패 43

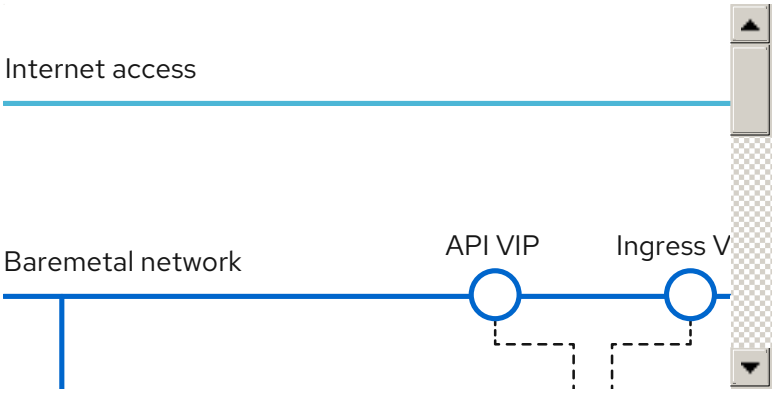
1.4.8.6. NTP가 동기화되지 않음 44
1.4.9. 설치 확인 46

1장. 베어 메탈에 설치 프로그램이 프로비저닝 한 클러스터 배포

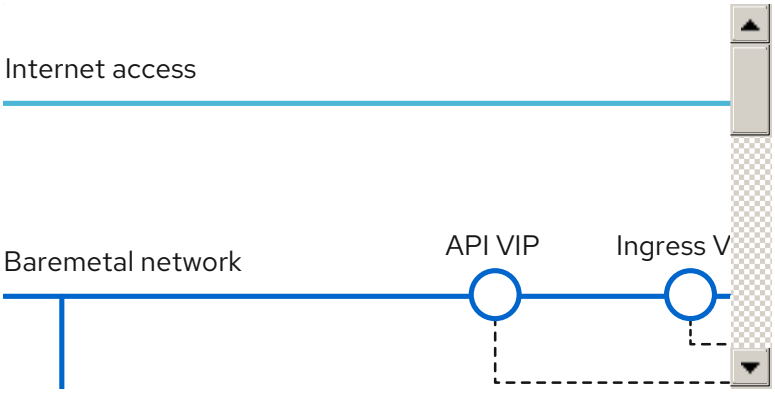
1.1. 개요

설치 프로그램에서 프로비저닝한 설치는 베어 메탈 노드에 OpenShift Container Platform 설치를 지원합니다. 이 문서에서는 설치를 성공적으로 수행하는 방법에 대해 설명합니다.

베어 메탈에 설치 프로비저닝를 설치하는 동안 **provisioner**라고 표시된 베어 메탈 노드의 설치 프로그램은 부트스트랩 가상 머신을 생성합니다. 부트스트랩 가상 머신의 역할은 OpenShift Container Platform 클러스터 배포 프로세스를 지원하는 것입니다. 부트스트랩 가상 머신은 네트워크 브리지를 통해 **baremetal** 네트워크 및 **provisioning** 네트워크 (있는 경우)에 연결됩니다.



OpenShift Container Platform 컨트롤 플레인 노드의 설치가 완료되고 완전히 작동하면 설치 프로그램이 부트스트랩 VM을 자동으로 제거하고 그에 따라 VIP (가상 IP 주소)를 적절한 노드로 이동합니다. API VIP는 컨트롤 플레인 노드로 이동하고 Ingress VIP는 작업자 노드로 이동합니다.



1.2. 사전 요구 사항

OpenShift Container Platform 설치 프로그램으로 프로비저닝된 설치에는 다음이 필요합니다.

1. Red Hat Enterprise Linux (RHEL) 8.x가 설치된 프로비저너 노드 1개
2. 컨트롤 플레인 노드 3 개
3. 각 노드의 베이스 보드 관리 컨트롤러 (BMC) 액세스
4. 하나 이상의 네트워크:
    a. 라우팅 가능한 **필수** 네트워크 1개
    b. 노드 프로비저닝을 위한 **선택적** 네트워크 1개

c. **선택적** 관리 네트워크 1개

OpenShift Container Platform 설치 프로그램으로 프로비저닝 설치를 시작하기 전에 하드웨어 환경이 다음 요구 사항을 충족하는지 확인합니다.

1.2.1. 노드 요구 사항

설치 프로그램에서 제공하는 설치에는 여러 하드웨어 노드 요구 사항이 있습니다.

- **CPU 아키텍처:** 모든 노드는 **x86_64** CPU 아키텍처를 사용해야 합니다.
- **유사한 노드:** Red Hat은 노드에서 역할별로 동일한 구성을 설정할 것을 권장합니다. 즉, Red Hat은 동일한 CPU, 메모리, 스토리지 설정의 브랜드 및 모델의 노드를 사용할 것을 권장하고 있습니다.
- **베이스 보드 관리 컨트롤러 : 프로비저너** 노드는 각 OpenShift Container Platform 클러스터 노드의 베이스 보드 관리 컨트롤러 (BMC)에 액세스할 수 있어야 합니다. IPMI, RedFish 또는 전용 프로토콜을 사용할 수 있습니다.
- **최신 생성:** 노드는 가장 최근 생성 중이어야 합니다. 설치 프로그램에서 제공하는 설치는 BMC 프로토콜에 의존하므로 하드웨어는 IPMI 암호화 제품군 17을 지원해야 합니다. 또한 RHEL 8에는 RAID 컨트롤러 용 최신 드라이버가 포함되어 있습니다. 노드가 RHEL 8을 실행하기 위해 **provisioner** 노드를 지원하고 RHCOS 8을 실행하기 위해 컨트롤 플레인 및 작업자 노드를 지원하기에 충분한 지 확인합니다.
- **레지스트리 노드:** 선택 사항: 연결이 끊긴 미러링된 레지스트리를 설정하는 경우 레지스트리가 자체 노드에 있는 것이 좋습니다.
- **프로비저너 노드:** 설치 프로그램에서 제공하는 설치에는 하나의 **프로비저너** 노드가 필요합니다.
- **컨트롤 플레인:** 설치 관리자 프로비저닝 설치에는 고가용성을 위해 세 개의 컨트롤 플레인 노드가 필요합니다.
- **작업자 노드:** 필요하지 않지만 일반적인 프로덕션 클러스터에는 하나 이상의 작업자 노드가 있습니다. 소규모 클러스터는 개발 및 테스트 중에 관리자와 개발자에게 더 많은 리소스를 제공합니다.
- **네트워크 인터페이스:** 각 노드에는 라우팅 가능한 **baremetal** 네트워크에 대해 하나 이상의 네트워크 인터페이스가 있어야합니다. 배포에 **provisioning** 네트워크를 사용할 때 각 노드에는 **provisioning** 네트워크에 대해 하나의 네트워크 인터페이스가 있어야합니다. **provisioning** 네트워크를 사용하는 것이 기본 구성입니다. 네트워크 인터페이스 이름 지정은 provisioning 네트워크의 컨트롤 플레인 노드에서 일관되어야 합니다. 예를 들어 컨트롤 플레인 노드에서 provisioning 네트워크에 **eth0** NIC를 사용하는 경우 다른 컨트롤 플레인 노드에서도 해당 NIC를 사용해야 합니다.
- **UEFI(Unified Extensible Firmware Interface):** 설치 관리자 프로비저닝 설치에는 provisioning 네트워크에서 IPv6 주소를 사용할 때 모든 OpenShift Container Platform 노드에서 UEFI 부팅이 필요합니다 **.** 또한 **provisioning** 네트워크 NIC에서 IPv6 프로토콜을 사용하도록 UEFI 장치 PXE 설정을 설정해야하지만 **provisioning 네트워크를 생략하면 이 요구 사항이 제거됩니다.**

1.2.2. 네트워크 요구 사항

OpenShift Container Platform의 설치 프로그램 프로비저닝 설치에는 기본적으로 몇 가지 네트워크 요구 사항이 있습니다. 먼저 설치 프로그램에서 프로비저닝하는 설치에는 각 베어 메탈 노드에서 운영 체제를 프로비저닝하기위한 라우팅 불가능한 **provisioning** 네트워크와 라우팅 가능한 **baremetal** 네트워크가 사

용됩니다. 설치 프로그램이 제공하는 설치는 **ironic-dnsmasq**를 배포하므로 네트워크에서 다른 DHCP 서버를 동일한 브로드 캐스트 도메인에서 실행할 수 없습니다. 네트워크 관리자는 OpenShift Container Platform 클러스터의 각 노드의 IP 주소를 예약해야 합니다.

Network Time Protocol (NTP)

클러스터의 각 OpenShift Container Platform 노드에 DHCP를 사용하여 검색 가능한 NTP(Network Time Protocol) 서버에 액세스하는 것이 좋습니다. NTP 서버가 없는 설치를 할 수 있지만 비동기 서버 클럭으로 인해 오류가 발생할 수 있습니다. NTP 서버를 사용하면 이 문제를 방지할 수 있습니다.

NIC 설정

OpenShift Container Platform은 다음 두 가지 네트워크를 사용하여 배포합니다.

- **프로비저닝**: provisioning **네트워크는** OpenShift Container Platform 클러스터의 일부인 각 노드에서 기본 운영 체제를 프로비저닝하는 데 사용되는 **선택 사항** 인 라우팅할 수 없는 네트워크입니다. **provisioning** 네트워크를 사용하여 배포하는 경우 각 노드의 첫 번째 NIC (**eth0** 또는 **eno1**)은 **provisioning** 네트워크와 상호 작용**해야합니다** .
- **baremetal**: **baremetal** 네트워크는 라우팅 가능한 네트워크입니다. **provisioning** 네트워크를 사용하여 배포하는 경우 각 노드의 두 번째 NIC (**eth1** 또는 **eno2**)은 **baremetal** 네트워크와 상호 작용**해야합니다**. **provisioning** 네트워크없이 배포하는 경우 **baremetal** 네트워크와 상호 작용하기 위해 각 노드에서 임의의 NIC를 사용할 수 있습니다.

중요

각 NIC는 적절한 네트워크에 해당하는 별도의 VLAN에 있어야 합니다.

DNS 서버 구성

클라이언트는 **baremetal** 네트워크를 통해 OpenShift Container Platform 클러스터 노드에 액세스합니다. 네트워크 관리자는 정식 이름 확장이 클러스터 이름인 하위 도메인 또는 하위 영역을 구성해야 합니다.

```
<cluster-name>.<domain-name>
```

예를 들면 다음과 같습니다.

```
test-cluster.example.com
```

DHCP 서버를 사용하여 노드의 IP 주소 예약

baremetal 네트워크의 경우 네트워크 관리자는 다음을 포함하여 여러 IP 주소를 예약해야 합니다.

1. 두 개의 가상 IP 주소
   - API 엔드 포인트에 대한 하나의 IP 주소
   - 와일드 카드 Ingress 엔드 포인트 중 IP 주소 1 개
2. 프로비저너 노드 중 하나의 IP 주소.
3. 각 컨트롤 플레인 (마스터) 노드의 하나의 IP 주소
4. 각 작업자 노드에 대해 하나의 IP 주소 (해당되는 경우)

다음 표에서는 정규화된 도메인 이름의 예를 보여줍니다. API 및 Nameserver 주소는 표준 이름 확장으로 시작됩니다. 컨트롤 플레인 및 작업자 노드의 호스트 이름은 예외이므로 원하는 호스트 이름 지정 규칙을 사용할 수 있습니다.

사용법	호스트 이름	IP
API	*api.<cluster-name>.<domain>*	*<ip>*
Ingress LB (apps)	**.apps.<cluster-name>.<domain>*	*<ip>*
Provisioner node	*provisioner.<cluster-name>.<domain>*	*<ip>*
Master-0	*openshift-master-0.<cluster-name>.<domain>*	*<ip>*
Master-1	*openshift-master-1.<cluster-name>-.<domain>*	*<ip>*
Master-2	*openshift-master-2.<cluster-name>.<domain>*	*<ip>*
Worker-0	*openshift-worker-0.<cluster-name>.<domain>*	*<ip>*
Worker-1	*openshift-worker-1.<cluster-name>.<domain>*	*<ip>*
Worker-n	*openshift-worker-n.<cluster-name>.<domain>*	*<ip>*

프로비저닝 네트워크가 없는 경우 추가 요구 사항

모든 설치 프로그램 프로비저닝 설치에는 **baremetal** 네트워크가 필요합니다. **baremetal** 네트워크는 외부의 외부 네트워크 액세스에 사용되는 라우팅 가능한 네트워크입니다. OpenShift Container Platform 클러스터 노드에 제공된 IP 주소 외에도 **provisioning** 네트워크가 없는 경우의 설치에는 다음이 필요합니다.

- **install-config.yaml** 설정 파일의 **bootstrapProvisioningIP** 설정에 **baremetal** 네트워크에서 사용 가능한 IP 주소를 설정합니다.
- **install-config.yaml** 설정 파일의 **provisioningHostIP** 설정에 **baremetal** 네트워크에서 사용 가능한 IP 주소를 설정합니다.
- RedFish Virtual Media/iDRAC Virtual Media를 사용하여 OpenShift Container Platform 클러스터를 배포합니다.

참고

provisioning 네트워크를 사용하는 경우 **bootstrapProvisioningIP** 및 **provisioningHostIP**에 추가 IP 주소를 설정할 필요가 없습니다.

대역 외 관리 IP 주소에 대한 포트 액세스

대역 외 관리 IP 주소는 노드와 별도의 네트워크에 있습니다. 대역 외 관리가 설치 중에 **baremetal** 노드와 통신할 수 있도록 대역 외 관리 IP 주소는 TCP 6180 포트에 대한 액세스 권한을 부여해야 합니다.

1.2.3. 노드 설정

provisioning 네트워크를 사용할 때 노드 설정

클러스터의 각 노드는 적절한 설치를 위해 다음과 같은 설정이 필요합니다.

주의

노드간에 일치하지 않으면 설치에 실패합니다.

클러스터 노드에는 두 개 이상의 NIC를 추가할 수 있지만 설치 프로세스는 처음 두 개의 NIC에만 추점을 둡니다.

NIC	네트워크	VLAN
NIC1	**provisioning**	<provisioning-vlan>
NIC2	**baremetal**	<baremetal-vlan>

NIC1은 OpenShift Container Platform 클러스터 설치에만 사용되는 라우팅 불가능한 네트워크 (**provisioning**)입니다.

Provisioner 노드의 RHEL (Red Hat Enterprise Linux) 8.x 설치 프로세스는 다를 수 있습니다. 로컬 Satellite 서버 PXE 서버 PXE 지원 NIC2를 사용하여 Red Hat Enterprise Linux (RHEL) 8.x를 설치하려면 다음과 같이합니다.

PXE	부팅 순서
NIC1 PXE 지원 **provisioning** 네트워크	1
NIC2 **baremetal** 네트워크 PXE 사용은 선택 사항입니다.	2

참고

다른 모든 NIC에서 PXE가 비활성화되어 있는지 확인합니다.

다음과 같이 컨트롤 플레인 및 작업자 노드를 설정합니다.

PXE	부팅 순서
NIC1 PXE 활성화 (프로비저닝 네트워크)	1

provisioning 네트워크없이 노드 설정

설치 프로세스에는 하나의 NIC가 필요합니다.

NIC	네트워크	VLAN
NICx	**baremetal**	<baremetal-vlan>

NICx는 OpenShift Container Platform 클러스터 설치에 사용되는 라우팅 가능한 네트워크 (**baremetal**)이며 인터넷으로 라우팅될 수 있습니다.

1.2.4. 대역 외 관리

일반적으로 노드에는 베이스 보드 관리 컨트롤러 (BMC)에서 사용하는 추가 NIC가 있습니다. 이러한 BMC는 **provisioner** 노드에서 액세스할 수 있습니다.

각 노드는 대역 외 관리를 통해 액세스할 수 있어야합니다. 대역 외 관리 네트워크를 사용할 때 **Provisioner** 노드는 OpenShift Container Platform 4를 성공적으로 설치하기 위해 대역 외 관리 네트워크에 액세스해야 합니다.

대역 외 관리 설정은 이 문서에서 다루지 않습니다. 대역 외 관리를 위해 별도의 관리 네트워크를 설정하는 것이 좋습니다. 그러나 **provisioning** 네트워크 또는 **baremetal** 네트워크를 사용하는 것은 유효한 옵션입니다.

1.2.5. 설치에 필요한 데이터

OpenShift Container Platform 클러스터를 설치하기 전에 모든 클러스터 노드에서 다음 정보를 수집하십시오.

- 대역 외 관리 IP
  - 예
    - Dell (iDRAC) IP
    - HP (iLO) IP

provisioning 네트워크를 사용하는 경우

- NIC1 (**provisioning**) MAC 주소
- NIC2 (**baremetal**) MAC 주소

provisioning 네트워크를 생략하는 경우

- NICx (**baremetal**) MAC 주소

1.2.6. 노드의 유효성 검사 체크리스트

provisioning 네트워크를 사용하는 경우

❑ NIC1 VLAN은 **provisioning** 네트워크에 대해 설정되어 있습니다.

❑ NIC2 VLAN은 **baremetal** 네트워크에 대해 설정되어 있습니다.

❑ NIC1은 프로비저너, 컨트롤 플레인 (마스터) 및 작업자 노드에서 PXE를 지원합니다.

❑ 다른 모든 NIC에서 PXE는 비활성화되어 있습니다.

❑ 컨트롤 플레인 및 작업자 노드가 설정되어 있습니다.

❑ 모든 노드는 대역 외 관리를 통해 액세스할 수 있습니다.

❑ 별도의 관리 네트워크가 생성되어 있습니다. (선택 사항)

❑ 설치에 필요한 데이터.

provisioning 네트워크를 생략하는 경우

❑ NICx VLAN은 **baremetal** 네트워크 용으로 설정되어 있습니다.

❑ 컨트롤 플레인 및 작업자 노드가 설정되어 있습니다.

❑ 모든 노드는 대역 외 관리를 통해 액세스할 수 있습니다.

❑ 별도의 관리 네트워크가 생성되어 있습니다. (선택 사항)

❑ 설치에 필요한 데이터.

1.3. OPENSHIFT 설치를 위한 환경 설정

1.3.1. 프로비저너 노드에 RHEL 설치

네트워킹 구성이 완료되면 다음 단계는 프로비저너 노드에 RHEL 8.x를 설치하는 것입니다. 설치 프로그램은 OpenShift Container Platform 클러스터를 설치하는 동안 프로비저너 노드를 오케스트레이터로 사용합니다. 이 문서에서는 프로비저너 노드에 RHEL을 설치하는 것은 다루지 않습니다. 선택 옵션에는 RHEL Satellite 서버, PXE 또는 설치 미디어 사용이 포함되지어 있지만 이에 국한되지는 않습니다.

1.3.2. OpenShift Container Platform 설치를 위한 provisioner 노드 준비

환경을 준비하려면 다음 단계를 수행하십시오.

프로세스

1. **ssh**를 통해 프로비저너 노드에 로그인합니다.
2. root가 아닌 사용자 (**kni**)를 만들고 해당 사용자에게 **sudo** 권한을 부여합니다.

```
# useradd kni
# passwd kni
# echo "kni ALL=(root) NOPASSWD:ALL" | tee -a /etc/sudoers.d/kni
# chmod 0440 /etc/sudoers.d/kni
```

3. 새 사용자에 대한 **ssh** 키를 만듭니다.

```
# su - kni -c "ssh-keygen -t ed25519 -f /home/kni/.ssh/id_rsa -N ''"
```

4. 프로비저너 노드에서 새 사용자로 로그인합니다.

```
# su - kni
$
```

5. Red Hat Subscription Manager를 사용하여 프로비저닝 노드를 등록합니다.

```
$ sudo subscription-manager register --username=<user> --password=<pass> --auto-attach
$ sudo subscription-manager repos --enable=rhel-8-for-x86_64-appstream-rpms --
enable=rhel-8-for-x86_64-baseos-rpms
```

참고

Red Hat Subscription Manager에 대한 자세한 내용은 Using and Configuring Red Hat Subscription Manager에서 참조하십시오.

6. 다음 패키지를 설치합니다.

```
$ sudo dnf install -y libvirt qemu-kvm mkisofs python3-devel jq ipmitool
```

7. 사용자를 변경하여 **libvirt** 그룹을 새로 만든 사용자에 추가합니다.

```
$ sudo usermod --append --groups libvirt <user>
```

8. **firewalld**를 다시 시작하고 **http** 서비스를 활성화합니다.

```
$ sudo systemctl start firewalld
$ sudo firewall-cmd --zone=public --add-service=http --permanent
$ sudo firewall-cmd --reload
```

9. **libvirtd** 서비스를 시작하고 활성화합니다.

```
$ sudo systemctl enable libvirtd --now
```

10. **default** 스토리지 풀을 생성하고 시작합니다.

```
$ sudo virsh pool-define-as --name default --type dir --target /var/lib/libvirt/images
$ sudo virsh pool-start default
$ sudo virsh pool-autostart default
```

11. 네트워킹을 설정합니다.

참고

이 단계는 웹 콘솔에서도 실행할 수 있습니다.

```
$ export PUB_CONN=<baremetal_nic_name>
$ export PROV_CONN=<prov_nic_name>
$ sudo nohup bash -c "
    nmcli con down \"$PROV_CONN\"
    nmcli con down \"$PUB_CONN\"
    nmcli con delete \"$PROV_CONN\"
    nmcli con delete \"$PUB_CONN\"
    # RHEL 8.1 appends the word \"System\" in front of the connection, delete in case it exists
    nmcli con down \"System $PUB_CONN\"
    nmcli con delete \"System $PUB_CONN\"
    nmcli connection add ifname provisioning type bridge con-name provisioning
```

```
    nmcli con add type bridge-slave ifname \"$PROV_CONN\" master provisioning
    nmcli connection add ifname baremetal type bridge con-name baremetal
    nmcli con add type bridge-slave ifname \"$PUB_CONN\" master baremetal
    pkill dhclient;dhclient baremetal
    nmcli connection modify provisioning ipv6.addresses fd00:1101::1/64 ipv6.method manual
    nmcli con down provisioning
    nmcli con up provisioning
"
```

참고

이 단계를 실행한 후 **ssh** 연결이 끊어 질 수 있습니다.

baremetal 네트워크를 통해 라우팅 할 수 없는 한 IPv6 주소는 모든 주소를 사용할 수 있습니다.

Pv6 주소를 사용하는 경우 UEFI가 활성화되고 UEFI PXE 설정이 IPv6 프로토콜로 설정되어 있는지 확인하십시오.

12. **provisioning** 네트워크 연결에서 IPv4 주소를 구성합니다.

```
$ nmcli connection modify provisioning ipv4.addresses 172.22.0.254/24 ipv4.method manual
```

13. **provisioner** 노드로 **ssh**를 실행합니다 (필요한 경우).

```
# ssh kni@provisioner.<cluster-name>.<domain>
```

14. 연결 브리지가 제대로 생성되었는지 확인합니다.

```
$ sudo nmcli con show
```

```
NAME               UUID                                  TYPE      DEVICE
baremetal          4d5133a5-8351-4bb9-bfd4-3af264801530  bridge    baremetal
provisioning       43942805-017f-4d7d-a2c2-7cb3324482ed  bridge    provisioning
virbr0             d9bca40f-eee1-410b-8879-a2d4bb0465e7  bridge    virbr0
bridge-slave-eno1  76a8ed50-c7e5-4999-b4f6-6d9014dd0812  ethernet  eno1
bridge-slave-eno2  f31c3353-54b7-48de-893a-02d2b34c4736  ethernet  eno2
```

15. **pull-secret.txt** 파일을 만듭니다.

```
$ vim pull-secret.txt
```

웹 브라우저에서 Install on Bare Metal with user-provisioned infrastructure 로 이동하고 **Downloads** 섹션 아래로 스크롤합니다. **Copy pull secret**을 클릭합니다. **pull-secret.txt** 파일에 내용을 붙여 넣고 **kni** 사용자의 홈 디렉터리에 저장합니다.

1.3.3. OpenShift Container Platform 설치 프로그램 검색

설치 프로그램의 **latest-4.x** 버전을 사용하여 최신 OpenShift Container Platform 릴리스 버전을 배포합니다.

```
$ export VERSION=latest-4.6
export RELEASE_IMAGE=$(curl -s https://mirror.openshift.com/pub/openshift-
v4/clients/ocp/$VERSION/release.txt | grep 'Pull From: quay.io' | awk -F ' ' '{print $3}')
```

추가 리소스

- 다양한 릴리스 채널에 대한 자세한 내용은 OpenShift Container Platform 업그레이드 채널 및 릴리스를 참조하십시오.

1.3.4. OpenShift Container Platform 설치 프로그램 추출

설치 프로그램을 가져온 후 다음 단계로 압축을 풉니다.

프로세스

1. 환경 변수를 설정합니다.

```
$ export cmd=openshift-baremetal-install
$ export pullsecret_file=~/pull-secret.txt
$ export extract_dir=$(pwd)
```

2. **oc** 바이너리를 가져옵니다.

```
$ curl -s https://mirror.openshift.com/pub/openshift-v4/clients/ocp/$VERSION/openshift-client-
linux.tar.gz | tar zxvf - oc
```

3. 설치 프로그램 압축을 풉니다.

```
$ sudo cp oc /usr/local/bin
$ oc adm release extract --registry-config "${pullsecret_file}" --command=$cmd --to
"${extract_dir}" ${RELEASE_IMAGE}
$ sudo cp openshift-baremetal-install /usr/local/bin
```

1.3.5. RHCOS 이미지 캐시 만들기 (선택 사항)

이미지 캐싱을 사용하려면 부트스트랩 VM에서 사용하는 RHCOS (Red Hat Enterprise Linux CoreOS) 이미지와 다른 노드를 프로비저닝하기 위해 설치 프로그램에서 사용하는 RHCOS 이미지의 두 가지 이미지를 다운로드해야 합니다. 이미지 캐싱은 선택 사항이지만 대역폭이 제한된 네트워크에서 설치 프로그램을 실행할 때 특히 유용합니다.

대역폭이 제한된 네트워크에서 설치 프로그램을 실행 중이고 RHCOS 이미지 다운로드에 15-20 분 이상 걸리는 경우 설치 프로그램이 시간 초과됩니다. 이러한 경우 웹 서버에서 이미지를 캐시할 수 있습니다.

다음 단계를 사용하여 이미지가 포함된 컨테이너를 설치합니다.

1. **podman**을 설치합니다.

```
$ sudo dnf install -y podman
```

2. RHCOS 이미지 캐싱에 사용할 방화벽 포트 **8080** 을 엽니 다.

```
$ sudo firewall-cmd --add-port=8080/tcp --zone=public --permanent
```

```
$ sudo firewall-cmd --reload
```

3. **bootstraposimage** 및 **clusterosimage** 를 저장할 디렉터리를 만듭니다.

```
$ mkdir /home/kni/rhcos_image_cache
```

4. 새로 생성된 디렉터리에 적절한 SELinux 컨텍스트를 설정합니다.

```
$ sudo semanage fcontext -a -t httpd_sys_content_t "/home/kni/rhcos_image_cache(/.*)?"
$ sudo restorecon -Rv rhcos_image_cache/
```

5. 설치 프로그램에서 커밋 ID를 가져옵니다. ID는 설치 프로그램이 다운로드해야하는 이미지를 결정합니다.

```
$ export COMMIT_ID=$(/usr/local/bin/openshift-baremetal-install version | grep '^built from commit' | awk '{print $4}')
```

6. 설치 프로그램이 노드에 배포할 RHCOS 이미지의 URI를 가져옵니다.

```
$ export RHCOS_OPENSTACK_URI=$(curl -s -S https://raw.githubusercontent.com/openshift/installer/$COMMIT_ID/data/data/rhcos.json  | jq .images.openstack.path | sed 's/"//g')
```

7. 설치 프로그램이 부트스트랩 VM에 배포할 RHCOS 이미지의 URI를 가져옵니다.

```
$ export RHCOS_QEMU_URI=$(curl -s -S https://raw.githubusercontent.com/openshift/installer/$COMMIT_ID/data/data/rhcos.json  | jq .images.qemu.path | sed 's/"//g')
```

8. 이미지가 게시되는 경로를 가져옵니다.

```
$ export RHCOS_PATH=$(curl -s -S https://raw.githubusercontent.com/openshift/installer/$COMMIT_ID/data/data/rhcos.json | jq .baseURI | sed 's/"//g')
```

9. 부트스트랩 VM에 배포된 RHCOS 이미지의 SHA 해시를 가져옵니다.

```
$ export RHCOS_QEMU_SHA_UNCOMPRESSED=$(curl -s -S https://raw.githubusercontent.com/openshift/installer/$COMMIT_ID/data/data/rhcos.json  | jq -r '.images.qemu["uncompressed-sha256"]')
```

10. 노드에 배포되는 RHCOS 이미지의 SHA 해시를 가져옵니다.

```
$ export RHCOS_OPENSTACK_SHA_COMPRESSED=$(curl -s -S https://raw.githubusercontent.com/openshift/installer/$COMMIT_ID/data/data/rhcos.json  | jq -r '.images.openstack.sha256')
```

11. 이미지를 다운로드하여 **/home/kni/rhcos_image_cache** 디렉터리에 저장하십시오.

```
$ curl -L ${RHCOS_PATH}${RHCOS_QEMU_URI} -o /home/kni/rhcos_image_cache/${RHCOS_QEMU_URI}
$ curl -L ${RHCOS_PATH}${RHCOS_OPENSTACK_URI} -o
```

```
/home/kni/rhcos_image_cache/${RHCOS_OPENSTACK_URI}
```

12. SELinux 유형이 새로 생성된 파일의 **httpd_sys_content_t**임을 확인합니다.

```
$ ls -Z /home/kni/rhcos_image_cache
```

13. Pod를 생성합니다.

```
$ podman run -d --name rhcos_image_cache \
-v /home/kni/rhcos_image_cache:/var/www/html \
-p 8080:8080/tcp \
quay.io/centos7/httpd-24-centos7:latest
```

1.3.6. 구성 파일

1.3.6.1. install-config.yaml 파일을 생성합니다.

install-config.yaml 파일에는 몇 가지 추가 정보가 필요합니다. 대부분의 정보는 설치된 클러스터가 사용 가능한 하드웨어를 완벽하게 관리하는 데 필요한 정보를 충분히 갖도록 설치 프로세스를 안내하는 데 사용됩니다.

1. **install-config.yaml**을 설정합니다. **pullSecret** 및 **sshKey** 등 환경에 맞게 적절한 변수를 변경합니다.

```
apiVersion: v1
baseDomain: <domain>
metadata:
  name: <cluster-name>
networking:
  machineCIDR: <public-cidr>
  networkType: OVNKubernetes
compute:
- name: worker
  replicas: 2 (1)
controlPlane:
  name: master
  replicas: 3
  platform:
    baremetal: {}
platform:
  baremetal:
    apiVIP: <api-ip>
    ingressVIP: <wildcard-ip>
    provisioningNetworkInterface: <NIC1>
    provisioningNetworkCIDR: <CIDR>
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: ipmi://<out-of-band-ip> (2)
          username: <user>
          password: <password>
        bootMACAddress: <NIC1-mac-address>
        hardwareProfile: default
```

```
      - name: <openshift-master-1>
        role: master
        bmc:
          address: ipmi://<out-of-band-ip> 3
          username: <user>
          password: <password>
        bootMACAddress: <NIC1-mac-address>
        hardwareProfile: default
      - name: <openshift-master-2>
        role: master
        bmc:
          address: ipmi://<out-of-band-ip> 4
          username: <user>
          password: <password>
        bootMACAddress: <NIC1-mac-address>
        hardwareProfile: default
      - name: <openshift-worker-0>
        role: worker
        bmc:
          address: ipmi://<out-of-band-ip> 5
          username: <user>
          password: <password>
        bootMACAddress: <NIC1-mac-address>
        hardwareProfile: unknown
      - name: <openshift-worker-1>
        role: worker
        bmc:
          address: ipmi://<out-of-band-ip>
          username: <user>
          password: <password>
        bootMACAddress: <NIC1-mac-address>
        hardwareProfile: unknown
pullSecret: '<pull_secret>'
sshKey: '<ssh_pub_key>'
```

1 OpenShift Container Platform 클러스터의 일부인 작업자 노드 수를 기준으로 작업자 머신을 스케일링합니다.

2 3 4 5 자세한 옵션은 BMC 주소 지정 섹션을 참조하십시오.

2. 클러스터 설정을 저장할 디렉터리를 만듭니다.

```
$ mkdir ~/clusterconfigs
$ cp install-config.yaml ~/clusterconfigs
```

3. OpenShift Container Platform 클러스터를 설치하기 전에 모든 베어 메탈 노드의 전원이 꺼져 있는지 확인합니다.

```
$ ipmitool -I lanplus -U <user> -P <password> -H <management-server-ip> power off
```

4. 이전 배포 시도에서 이전 부트스트랩 리소스가 남아 있는 경우 이전 부트스트랩 리소스를 삭제합니다.

```
for i in $(sudo virsh list | tail -n +3 | grep bootstrap | awk {'print $2'});
```

```
do
  sudo virsh destroy $i;
  sudo virsh undefine $i;
  sudo virsh vol-delete $i --pool $i;
  sudo virsh vol-delete $i.ign --pool $i;
  sudo virsh pool-destroy $i;
  sudo virsh pool-undefine $i;
done
```

1.3.6.2. install-config.yaml 파일 내에서 프록시 설정 (선택 사항)

프록시를 사용하여 OpenShift Container Platform 클러스터를 배포하려면 **install-config.yaml** 파일을 다음과 같이 변경합니다.

```
apiVersion: v1
baseDomain: <domain>
proxy:
  httpProxy: http://USERNAME:PASSWORD@proxy.example.com:PORT
  httpsProxy: https://USERNAME:PASSWORD@proxy.example.com:PORT
  noProxy: <WILDCARD_OF_DOMAIN>,<PROVISIONING_NETWORK/CIDR>,
<BMC_ADDRESS_RANGE/CIDR>
```

다음은 값을 포함하는 **noProxy**의 예입니다.

```
noProxy: .example.com,172.22.0.0/24,10.10.0.0/24
```

프록시가 활성화된 상태에서 해당 키 / 값 쌍에 적절한 프록시 값을 설정합니다.

주요 고려 사항:

- 프록시에 HTTPS 프록시가 없는 경우 **httpsProxy** 값을 **https://**에서 **http://**로 변경합니다.
- 프로비저닝 네트워크를 사용하는 경우 이를 **noProxy** 설정에 포함하십시오. 그렇지 않으면 설치 프로그램이 실패합니다.
- 모든 프록시 설정을 프로버저너 노드 내에서 환경 변수로 설정합니다. 예를 들어, **HTTP_PROXY, https_proxy**, **NO_PROXY**가 포함됩니다.

참고

IPv6으로 프로비저닝할 때 **noProxy** 설정에서 CIDR 주소 블록을 정의할 수 없습니다. 각 주소를 개별적으로 정의해야 합니다.

1.3.6.3. provisioning 네트워크가 없는 경우 install-config.yaml 파일 변경 (선택 사항)

provisioning 네트워크없이 OpenShift Container Platform 클러스터를 배포하려면 **install-config.yaml** 파일을 다음과 같이 변경하십시오.

```
platform:
  baremetal:
    apiVIP: <apiVIP>
    ingressVIP: <ingress/wildcard VIP>
```

```
provisioningNetwork: "Disabled"
provisioningHostIP: <baremetal_network_IP1>
bootstrapProvisioningIP: <baremetal_network_IP2>
```

참고

provisioningHostIP 및 **bootstrapProvisioningIP** 구성 설정의 **baremetal** 네트워크에서 두 개의 IP 주소를 지정하고 **provisioningBridge** 및 **provisioningNetworkCIDR** 구성 설정을 제거해야합니다.

1.3.6.4. 추가 install-config 매개 변수

install-config.yaml 파일의 필수 매개 변수 **hosts** 매개 변수 및 **bmc** 매개 변수는 다음 표를 참조하십시오.

표 1.1. 필수 매개 변수

매개 변수	기본	설명
baseDomain		클러스터의 도메인 이름입니다. 예: **example.com**
sshKey		**sshKey** 구성 설정에는 컨트롤 플레인 노드 및 작업자 노드에 액세스하는 데 필요한 **~/.ssh/id_rsa.pub** 파일의 키가 포함되어 있습니다. 일반적으로 이 키는 **provisioner** 노드에서 가져옵니다.
pullSecret		**pullSecret** 구성 설정에는 프로비저너 노드를 준비할 때 Install OpenShift on Bare Metal 페이지에서 다운로드한 풀 시크릿 사본이 포함되어 있습니다.
`metadata:` `  name:`		OpenShift Container Platform 클러스터에 지정되는 이름입니다. 예: **openshift**
`networking:` `  machineCIDR:`		외부 네트워크의 공개 CIDR (Classless Inter-Domain Routing) 입니다. 예: **10.0.0.0/24**
`compute:` `  - name: worker`		OpenShift Container Platform 클러스터에는 노드가 없는 경우에도 작업자 (또는 컴퓨팅) 노드에 이름을 지정해야 합니다.

매개 변수	기본	설명
compute: replicas: 2		복제는 OpenShift Container Platform 클러스터의 작업자 (또는 컴퓨팅) 노드 수를 설정합니다.
controlPlane: name: master		OpenShift Container Platform 클러스터에는 컨트롤 플레인 (마스터) 노드의 이름이 필요합니다.
controlPlane: replicas: 3		복제는 OpenShift Container Platform 클러스터의 일부로 포함된 컨트롤 플레인 (마스터) 노드의 수를 설정합니다.
provisioningNetworkInterface		프로비저닝 네트워크에 연결된 컨트롤 플레인 노드의 네트워크 인터페이스 이름입니다.
defaultMachinePlatform		플랫폼 구성없이 머신 풀에 사용되는 기본 설정입니다.
apiVIP	**api.<clustername.clusterdomain>**	내부 API 통신에 사용하는 VIP입니다. 이 설정은 기본 이름이 올바르게 확인되도록 DNS에서 지정되거나 사전에 설정해야 합니다.
disableCertificateVerification	**False**	**redfish** 및 **redfish-virtualmedia** 는 BMC 주소를 관리하기 위해 이 매개 변수가 필요합니다. BMC 주소에 자체 서명된 인증서를 사용하는 경우 값은 **True** 여야합니다.
ingressVIP	**test.apps.<clustername.clusterdomain>**	Ingress 트래픽에 사용할 VIP입니다.

표 1.2. 선택적 매개변수

매개 변수	기본	설명
provisioningDHCPRange	**172.22.0.10,172.22.0.100**	**provisioning** 네트워크에서 노드의 IP 범위를 정의합니다.

매개 변수	기본	설명
provisioningNetworkCIDR	**172.22.0.0/24**	프로비저닝에 사용할 네트워크의 CIDR입니다. 이 옵션은 **provisioning** 네트워크에서 기본 주소 범위를 사용하지 않는 경우에 필요합니다.
clusterProvisioningIP	**provisioningNetworkCIDR**의 세 번째 IP 주소입니다.	프로비저닝 서비스가 실행되는 클러스터 내의 IP 주소입니다. 기본값은 **provisioning** 서브넷의 세 번째 IP 주소입니다. 예: **172.22.0.3**
bootstrapProvisioningIP	**provisioningNetworkCIDR**의 두 번째 IP 주소입니다.	설치 프로그램이 컨트롤 플레인 (마스터) 노드를 배포하는 동안 프로비저닝 서비스가 실행되는 부트스트랩 VM의 IP입니다. 기본값은 **provisioning** 서브넷의 두 번째 IP입니다. 예:**172.22.0.2** **provisioning** 네트워크를 사용하지 않는 경우 이 값을 **baremetal** 네트워크에서 사용 가능한 IP 주소로 설정합니다.
externalBridge	**baremetal**	**baremetal** 네트워크에 연결된 하이퍼 바이저의**baremetal** 브리지 이름입니다.
provisioningBridge	**provisioning**	**provisioning** 네트워크에 연결된**provisioner** 호스트의 **provisioning** 브리지 이름입니다.
defaultMachinePlatform		플랫폼 구성없이 머신 풀에 사용되는 기본 설정입니다.
bootstrapOSImage		부트스트랩 노드의 기본 운영 체제 이미지를 재정의하는 URL입니다. URL에는 이미지의 SHA-256 해시가 포함되어 있어야합니다. 예: **https://mirror.openshift.com/rhcos-<version>-qemu.qcow2.gz?sha256=<uncompressed_sha256>;** .
clusterOSImage		클러스터 노드의 기본 운영 체제를 재정의하는 URL입니다. URL에는 이미지의 SHA-256 해시가 포함되어 있어야 합니다. 예: **https://mirror.openshift.com/images/rhcos-<version>-openstack.qcow2.gz?sha256=<compressed_sha256>;**
provisioningNetwork		**provisioning** 네트워크의 요구 사항을 비활성화하려면 이 매개 변수를 **Disabled**로 설정합니다. 사용자는 가상 미디어 기반 프로비저닝 만 수행하거나 지원 설치를 사용하여 클러스터를 가져올 수 있습니다. 전원 관리를 사용하는 경우 시스템 네트워크에서 BMC에 액세스할 수 있습니다. 사용자는 프로비저닝 서비스에 사용되는 외부 네트워크에서 두 개의 IP 주소를 지정해야 합니다. DHCP, TFTP 등을 포함하여 프로비저닝 네트워크를 완전히 관리하려면 이 매개 변수를 기본값인 **managed**로 설정합니다. 이 매개 변수를 **unmanaged**로 설정하면 프로비저닝 네트워크를 계속 사용할 수 있지만 DHCP를 수동으로 설정해야 합니다. 가상 미디어의 프로비저닝이 권장되지만 필요한 경우 PXE를 계속 사용할 수 있습니다.

매개 변수	기본	설명
provisioningHostingIp		**provisioningNetwork** 구성 설정이 **Disabled** 로 설정된 경우 이 매개 변수를 **baremetal** 네트워크에서 사용 가능한 IP 주소로 설정합니다.
httpProxy		이 매개 변수를 환경 내에서 사용되는 적절한 HTTP 프록시로 설정합니다.
httpsProxy		이 매개 변수를 환경 내에서 사용되는 적절한 HTTPS 프록시로 설정합니다.
noProxy		이 매개 변수를 환경 내 프록시 사용에 대한 적절한 예외 목록으로 설정합니다.

호스트

hosts 매개 변수는 클러스터를 빌드하는 데 사용되는 별도의 베어 메탈 자산 목록입니다.

이름	기본	설명
name		세부 정보와 연결할 **BareMetalHost** 리소스의 이름입니다. 예: **openshift-master-0**
role		베어 메탈 노드의 역할입니다. **master** 또는 **worker** 중 하나입니다.
bmc		베이스 보드 관리 컨트롤러에 대한 연결 세부 정보입니다. 자세한 내용은 BMC 주소 지정 섹션을 참조하십시오.
bootMACAddress		호스트가 **provisioning** 네트워크에서 부팅하는 데 사용할 NIC의 MAC 주소입니다.

1.3.6.5. BMC 주소 지정

각 **bmc** 항목의 **address** 필드는 URL 체계의 컨트롤러 유형 및 네트워크에서의 위치를 포함하여 OpenShift Container Platform 클러스터 노드에 연결하기 위한 URL입니다.

IPMI

IPMI 호스트는 **ipmi://<out-of-band-ip>:<port>**를 사용하며 지정되지 않은 경우 포트 **623**에 기본 설정됩니다. 다음 예제는 **install-config.yaml** 파일 내의 IPMI 설정을 보여줍니다.

```
platform:
  baremetal:
    hosts:
```

```
      - name: openshift-master-0
        role: master
        bmc:
          address: ipmi://<out-of-band-ip>
          username: <user>
          password: <password>
```

HPE 용 RedFish

RedFish를 활성화하려면 **redfish://** 또는 **redfish+http://**를 사용하여 TLS를 비활성화합니다. 설치 프로그램에는 호스트 이름 또는 IP 주소와 시스템 ID 경로가 모두 필요합니다. 다음 예제는 **install-config.yaml** 파일 내의 RedFish 설정을 보여줍니다.

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: redfish://<out-of-band-ip>/redfish/v1/Systems/1
          username: <user>
          password: <password>
```

대역 외 관리 주소에 대한 권한 인증서를 사용하는 것이 좋지만 **disableCertificateVerification을 포함해야합니다. True** 자체 서명 인증서를 사용하는 경우 the **bmc** 구성에서. 다음 예제에서는 **disableCertificateVerification을 사용하는 RedFish 설정을 보여줍니다. true install-config.yaml** 파일 내의 설정 매개변수입니다.

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: redfish://<out-of-band-ip>/redfish/v1/Systems/1
          username: <user>
          password: <password>
          disableCertificateVerification: True
```

Dell 용 RedFish

RedFish를 활성화하려면 **redfish://** 또는 **redfish+http://**를 사용하여 TLS를 비활성화합니다. 설치 프로그램에는 호스트 이름 또는 IP 주소와 시스템 ID 경로가 모두 필요합니다. 다음 예제는 **install-config.yaml** 파일 내의 RedFish 설정을 보여줍니다.

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: redfish://<out-of-band-ip>/redfish/v1/Systems/System.Embedded.1
          username: <user>
          password: <password>
```

대역 외 관리 주소에 대한 권한 인증서를 사용하는 것이 좋지만 **disableCertificateVerification을 포함해야합니다. True** 자체 서명 인증서를 사용하는 경우 the **bmc** 구성에서. 다음 예제에서는 **disableCertificateVerification을 사용하는 RedFish 설정을 보여줍니다. true install-config.yaml** 파일 내의 설정 매개변수입니다.

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: redfish://<out-of-band-ip>/redfish/v1/Systems/System.Embedded.1
          username: <user>
          password: <password>
          disableCertificateVerification: True
```

참고

현재 RedFish는 베어 메탈 배포에서 OpenShift Container Platform의 설치 프로그램이 프로비저닝한 설치를 위해 iDRAC 펌웨어 버전 **4.20.20.20** 이상을 사용하는 Dell에서만 지원됩니다.

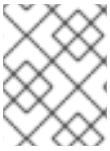
HPE 용 RedFish 가상 미디어

HPE 서버용 RedFish Virtual Media를 활성화하려면 **address** 설정에서 **redfish-virtualmedia://**를 사용합니다. 다음 예제는 **install-config.yaml** 파일 내에서 RedFish Virtual Media를 사용하는 방법을 보여줍니다.

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: redfish-virtualmedia://<out-of-band-ip>/redfish/v1/Systems/1
          username: <user>
          password: <password>
```

Dell 용 RedFish 가상 미디어

Dell 서버의 RedFish Virtual Media의 경우 **address** 설정에서 **idrac-virtualmedia://**를 사용합니다.



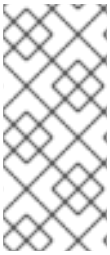
참고

Dell 서버의 RedFish Virtual Media에는 OpenShift Container Platform 4.6에서 알려진 문제가 있습니다. 4.6.1 포인트 릴리스에서 이 문제가 해결될 예정입니다.

다음 예는 **install-config.yaml** 파일 내에서 iDRAC 가상 미디어를 사용하는 방법을 보여줍니다.

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
```

```
address: idrac-virtualmedia://<out-of-band-ip>/redfish/v1/Systems/System.Embedded.1
username: <user>
password: <password>
```



참고

idrac-virtualmedia에는 iDRAC 펌웨어 버전 4.20.20.20 이상이 필요합니다.

OpenShift Container Platform 클러스터 노드에 iDRAC 콘솔을 통해 AutoAttach가 활성화되어 있는지 확인합니다. 메뉴 경로는 다음과 같습니다. **설정가상 미디어 --Attach 모드 AutoAttach**.

1.3.6.6. 루트 장치 팁

rootDeviceHints 매개 변수를 사용하면 설치 프로그램이 RHCOS (Red Hat Enterprise Linux CoreOS) 이미지를 특정 장치에 프로비저닝할 수 있습니다. 설치 프로그램은 장치를 검색한 순서대로 검사하고 검색된 값을 팁과 비교합니다. 설치 프로그램은 팁과 일치하는 첫 번째 검색된 장치를 사용합니다. 이 설정은 여러 팁을 결합할 수 있지만 장치는 설치 프로그램이이를 선택할 수 있도록 모든 팁과 일치해야 합니다.

표 1.3. 서브 필드

서브 필드	설명
deviceName	**/dev/vda**와 같은 Linux 장치 이름을 포함하는 문자열. 팁은 실제 값과 정확히 일치해야 합니다.
hctl	**0:0:0:0**과 같은 SCSI 버스 주소를 포함하는 문자열. 팁은 실제 값과 정확히 일치해야 합니다.
model	공급 업체별 장치 식별자가 포함된 문자열. 팁은 실제 값의 하위 문자열입니다.
vendor	장치의 공급 업체 또는 제조업체 이름이 포함된 문자열입니다. 팁은 실제 값의 하위 문자열입니다.
serialNumber	장치 일련 번호가 포함된 문자열입니다. 팁은 실제 값과 정확히 일치해야 합니다.
minSizeGigabytes	장치의 최소 크기 (기가 바이트)를 나타내는 정수입니다.
wwn	고유 저장소 식별자를 포함하는 문자열입니다. 팁은 실제 값과 정확히 일치해야 합니다.
wwnWithExtension	공급 업체 확장이 추가된 고유 한 저장소 식별자가 포함된 문자열입니다. 팁은 실제 값과 정확히 일치해야 합니다.
wwnVendorExtension	고유 공급 업체 저장소 식별자를 포함하는 문자열입니다. 팁은 실제 값과 정확히 일치해야 합니다.

서브 필드	설명
rotational	장치가 회전 디스크 여야하는지 (true) 아닌지 (false) 를 나타내는 부울 값입니다.

사용 예

```
 - name: master-0
   role: master
   bmc:
     address: ipmi://10.10.0.3:6203
     username: admin
     password: redhat
   bootMACAddress: de:ad:be:ef:00:40
   rootDeviceHints:
     deviceName: "/dev/sda"
```

1.3.6.7. OpenShift Container Platform 매니페스트 만들기

1. OpenShift Container Platform 매니페스트를 만듭니다.

```
$ ./openshift-baremetal-install --dir ~/clusterconfigs create manifests
```

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for
Scheduler cluster settings
WARNING Discarding the OpenShift Manifest that was provided in the target directory
because its dependencies are dirty and it needs to be regenerated
```

1.3.7. 비 연결 레지스트리 만들기 (선택 사항)

설치 레지스트리의 로컬 사본을 사용하여 OpenShift KNI 클러스터를 설치해야하는 경우가 있습니다. 이는 클러스터 노드가 인터넷에 액세스할 수 없는 네트워크에 있기 때문에 네트워크 효율성을 향상시키기 위한 것일 수 있습니다.

로컬 또는 미러링된 레지스트리 사본에는 다음이 필요합니다.

- 레지스트리 노드의 인증서. 자체 서명된 인증서를 사용할 수 있습니다.
- 웹 서버-이는 시스템의 컨테이너에 의해 제공됩니다.
- 인증서 및 로컬 저장소 정보를 포함하는 업데이트된 풀 시크릿.

참고

레지스트리 노드에 연결되지 않은 레지스트리를 만드는 것은 선택 사항입니다. 다음 섹션은 선택 사항으로 레지스트리 노드에 연결되지 않은 레지스트리를 만드는 경우에만 실행해야 하는 단계입니다. 레지스트리 노드에 연결되지 않은 레지스트리를 만들 때 "(선택 사항)"레이블이 붙은 모든 후속 하위 섹션을 실행해야 합니다.

1.3.7.1. 미러링된 레지스트리를 호스팅할 레지스트리 노드 준비 (선택 사항)

레지스트리 노드를 다음과 같이 변경합니다.

프로세스

1. 레지스트리 노드에서 방화벽 포트를 엽니다.

```
$ sudo firewall-cmd --add-port=5000/tcp --zone=libvirt  --permanent
$ sudo firewall-cmd --add-port=5000/tcp --zone=public   --permanent
$ sudo firewall-cmd --reload
```

2. 레지스트리 노드에 필요한 패키지를 설치합니다.

```
$ sudo yum -y install python3 podman httpd httpd-tools jq
```

3. 저장소 정보가 보관될 디렉터리 구조를 만듭니다.

```
$ sudo mkdir -p /opt/registry/{auth,certs,data}
```

1.3.7.2. 자체 서명 인증서 생성 (선택 사항)

레지스트리 노드의 자체 서명된 인증서를 생성하고 이를 **/opt/registry/certs** 디렉터리에 배치합니다.

프로세스

1. 인증서 정보를 적절하게 조정합니다.

```
$ host_fqdn=$( hostname --long )
$ cert_c="<Country Name>"   # Country Name (C, 2 letter code)
$ cert_s="<State>"          # Certificate State (S)
$ cert_l="<Locality>"       # Certificate Locality (L)
$ cert_o="<Organization>"   # Certificate Organization (O)
$ cert_ou="<Org Unit>"      # Certificate Organizational Unit (OU)
$ cert_cn="${host_fqdn}"    # Certificate Common Name (CN)

$ openssl req \
    -newkey rsa:4096 \
    -nodes \
    -sha256 \
    -keyout /opt/registry/certs/domain.key \
    -x509 \
    -days 365 \
    -out /opt/registry/certs/domain.crt \
    -addext "subjectAltName = DNS:${host_fqdn}" \
    -subj "/C=${cert_c}/ST=${cert_s}/L=${cert_l}/O=${cert_o}/OU=${cert_ou}/CN=${cert_cn}"
```

참고

<Country Name>을 바꾸는 경우에는 두 문자 만 사용합니다. 예: **US**

2. 새 인증서로 레지스트리 노드의 **ca-trust**를 업데이트합니다.

```
$ sudo cp /opt/registry/certs/domain.crt /etc/pki/ca-trust/source/anchors/
$ sudo update-ca-trust extract
```

1.3.7.3. 레지스트리 podman 컨테이너 만들기 (선택 사항)

레지스트리 컨테이너는 인증서, 인증 파일 및 데이터 파일 저장에 **/opt/registry** 디렉터리를 사용합니다.

레지스트리 컨테이너는 **httpd**를 사용하며 인증을 위해 **htpasswd** 파일이 필요합니다.

프로세스

1. 컨테이너에서 사용할 **htpasswd** 파일을 **/opt/registry/auth**에 만듭니다.

```
$ htpasswd -bBc /opt/registry/auth/htpasswd <user> <passwd>
```

   <user>를 사용자 이름으로, **<passwd>**를 암호로 바꿉니다.

2. 레지스트리 컨테이너를 만들고 시작합니다.

```
$ podman create \
  --name ocpdiscon-registry \
  -p 5000:5000 \
  -e "REGISTRY_AUTH=htpasswd" \
  -e "REGISTRY_AUTH_HTPASSWD_REALM=Registry" \
  -e "REGISTRY_HTTP_SECRET=ALongRandomSecretForRegistry" \
  -e "REGISTRY_AUTH_HTPASSWD_PATH=/auth/htpasswd" \
  -e "REGISTRY_HTTP_TLS_CERTIFICATE=/certs/domain.crt" \
  -e "REGISTRY_HTTP_TLS_KEY=/certs/domain.key" \
  -e "REGISTRY_COMPATIBILITY_SCHEMA1_ENABLED=true" \
  -v /opt/registry/data:/var/lib/registry:z \
  -v /opt/registry/auth:/auth:z \
  -v /opt/registry/certs:/certs:z \
  docker.io/library/registry:2
```

```
$ podman start ocpdiscon-registry
```

1.3.7.4. 풀 시크릿(pull-secret) 복사 및 업데이트 (선택 사항)

프로비저너 노드에서 레지스트리 노드로 풀 시크릿 파일을 복사하고 이를 새 레지스트리 노드의 인증 정보를 포함하도록 변경합니다.

프로세스

1. **pull-secret.txt** 파일을 복사합니다.

```
$ scp kni@provisioner:/home/kni/pull-secret.txt pull-secret.txt
```

2. **host_fqdn** 환경 변수를 레지스트리 노드의 완전한 도메인 이름으로 업데이트합니다.

```
$ host_fqdn=$( hostname --long )
```

3. **htpasswd** 파일을 만드는 데 사용되는 **http** 인증 정보의 base64 인코딩으로 **b64auth** 환경 변수를 업데이트합니다.

```
$ b64auth=$( echo -n '<username>:<passwd>' | openssl base64 )
```

<username>을 사용자 이름으로, **<passwd>**를 암호로 바꿉니다.

4. **base64** 승인 문자열을 사용하도록 **AUTHSTRING** 환경 변수를 설정합니다. **$USER** 변수는 현재 사용자의 이름을 포함하는 환경 변수입니다.

```
$ AUTHSTRING="{\"$host_fqdn:5000\": {\"auth\": \"$b64auth\",\"email\":
\"$USER@redhat.com\"}}"
```

5. **pull-secret.txt** 파일을 업데이트합니다.

```
$ jq ".auths += $AUTHSTRING" < pull-secret.txt > pull-secret-update.txt
```

1.3.7.5. 저장소 미러링 (선택 사항)

프로세스

1. 프로비저너 노드에서 레지스트리 노드에 **oc** 바이너리를 복사합니다.

```
$ sudo scp kni@provisioner:/usr/local/bin/oc /usr/local/bin
```

2. 필요한 환경 변수를 설정합니다.

   a. 릴리스 버전을 설정합니다.

```
$ VERSION=<release_version>
```

   <release_version> 은 설치할 OpenShift Container Platform 버전에 해당하는 태그를 지정합니다 (예: **4.6**).

   b. 로컬 레지스트리 이름 및 호스트 포트를 설정합니다.

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

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

   c. 로컬 리포지토리 이름을 설정합니다.

```
$ LOCAL_REPO='<local_repository_name>'
```

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

3. 원격 설치 이미지를 로컬 저장소에 미러링합니다.

```
$ /usr/local/bin/oc adm release mirror \
  -a pull-secret-update.txt \
  --from=$UPSTREAM_REPO \
```

```
--to-release-image=$LOCAL_REG/$LOCAL_REPO:${VERSION} \
--to=$LOCAL_REG/$LOCAL_REPO
```

1.3.7.6. 비 연결 레지스트리를 사용하도록 install-config.yaml 파일 변경 (선택 사항)

프로비저너 노드에서 **install-config.yaml** 파일은 **pull-secret-update.txt** 파일에서 새로 생성된 풀 시크릿(pull-secret)을 사용해야 합니다. **install-config.yaml** 파일에는 연결되지 않은 레지스트리 노드 인증서 및 레지스트리 정보도 포함되어야합니다.

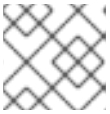
프로세스

1. 비 연결 레지스트리 노드의 인증서를 **install-config.yaml** 파일에 추가합니다. 인증서는 **"additionalTrustBundle:|"** 뒤에 있어야 하며 보통 두 개의 공백으로 적절하게 들여 쓰기합니다.

```
$ echo "additionalTrustBundle: |" >> install-config.yaml
$ sed -e 's/^/  /' /opt/registry/certs/domain.crt >> install-config.yaml
```

2. 레지스트리의 미러 정보를 **install-config.yaml** 파일에 추가합니다.

```
$ echo "imageContentSources:" >> install-config.yaml
$ echo "- mirrors:" >> install-config.yaml
$ echo "  - registry.example.com:5000/ocp4/openshift4" >> install-config.yaml
$ echo "  source: quay.io/openshift-release-dev/ocp-release" >> install-config.yaml
$ echo "- mirrors:" >> install-config.yaml
$ echo "  - registry.example.com:5000/ocp4/openshift4" >> install-config.yaml
$ echo "  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev" >> install-config.yaml
```



참고

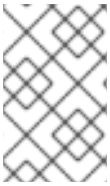
registry.example.com을 레지스트리의 정규화된 도메인 이름으로 바꿉니다.

1.3.8. 작업자 노드에 라우터 배치

설치 중에 설치 프로그램은 작업자 노드에 라우터 pod를 배포합니다. 기본적으로 설치 프로그램은 두 개의 라우터 pod를 설치합니다. 초기 클러스터에 작업자 노드가 하나만 있거나 배포된 클러스터에서 OpenShift Container Platform 클러스터 내의 서비스에 대해 외부 트래픽 로드를 처리하기 위해 추가 라우터가 필요한 경우 **yaml** 파일을 작성하여 적절한 라우터 복제 수를 설정할 수 있습니다.

참고

기본적으로 설치 프로그램은 두 개의 라우터를 배포합니다. 클러스터에 작업자 노드가 두 개 이상있는 경우 이 섹션을 생략할 수 있습니다. Ingress Operator에 대한 자세한 내용은 다음을 참조하십시오. OpenShift Container Platform의 Ingress Operator



참고

클러스터에 작업자 노드가 없는 경우 설치 프로그램은 기본적으로 컨트롤 플레인 노드에 두 개의 라우터를 배포합니다. 클러스터에 작업자 노드가없는 경우 이 섹션을 생략할 수 있습니다.

프로세스

1. **router-replicas.yaml** 파일을 만듭니다.

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: default
  namespace: openshift-ingress-operator
spec:
  replicas: <num-of-router-pods>
  endpointPublishingStrategy:
    type: HostNetwork
  nodePlacement:
    nodeSelector:
      matchLabels:
        node-role.kubernetes.io/worker: ""
```

참고

<num-of-router-pods>를 적절한 값으로 바꿉니다. 하나의 작업자 노드로만 작업하는 경우 **replicas :** 를 **1**로 설정합니다. 3 개 이상의 작업자 노드로 작업하는 경우 필요에 따라 **replicas:**을 기본값 **2**에서 늘릴 수 있습니다.

2. **router-replicas.yaml** 파일을 **clusterconfigs/openshift** 디렉터리에 저장 및 복사합니다.

```
cp ~/router-replicas.yaml clusterconfigs/openshift/99_router-replicas.yaml
```

1.3.9. 설치 확인 체크리스트

❑ OpenShift Container Platform 설치 프로그램이 검색되었습니다.

❑ OpenShift Container Platform 설치 프로그램이 추출되었습니다.

❑ **install-config.yaml**의 필수 매개 변수가 설정되었습니다.

❑ **install-config.yaml**의 **hosts** 매개 변수가 구성되었습니다.

❑ **install-config.yaml**의 **bmc** 매개 변수가 구성되었습니다.

❑ **bmc address** 필드에 구성된 값에 대한 규칙이 적용되었습니다.

❑ 비 연결 레지스트리를 생성했습니다 (선택 사항).

❑ (선택 사항) 비 연결 레지스트리 설정을 사용하고 있는 경우 이를 확인합니다.

❑ (선택 사항) 작업자 노드에 라우터를 배포했습니다.

1.3.10. OpenShift Container Platform 설치 프로그램을 통해 클러스터 배포

OpenShift Container Platform 설치 프로그램을 실행합니다.

```
$ ./openshift-baremetal-install --dir ~/clusterconfigs --log-level debug create cluster
```

1.3.11. 설치 후

배포 프로세스 중에 설치 디렉토리 폴더의 **.openshift_install.log** 로그 파일에 **tail** 명령을 실행하여 설치의 전체 상태를 확인할 수 있습니다.

```
$ tail -f /path/to/install-dir/.openshift_install.log
```

1.3.12. 베어 메탈에 클러스터 재설치 준비

베어 메탈에 클러스터를 다시 설치하기 전에 정리 작업을 수행해야 합니다.

프로세스

1. 부트스트랩, 컨트롤 플레인 (마스터라고도 함) 노드 및 작업자 노드의 디스크를 제거하거나 다시 포맷합니다. 하이퍼바이저 환경에서 작업하는 경우 제거한 디스크를 추가해야 합니다.
2. 이전 설치에서 생성한 아티팩트를 삭제합니다.

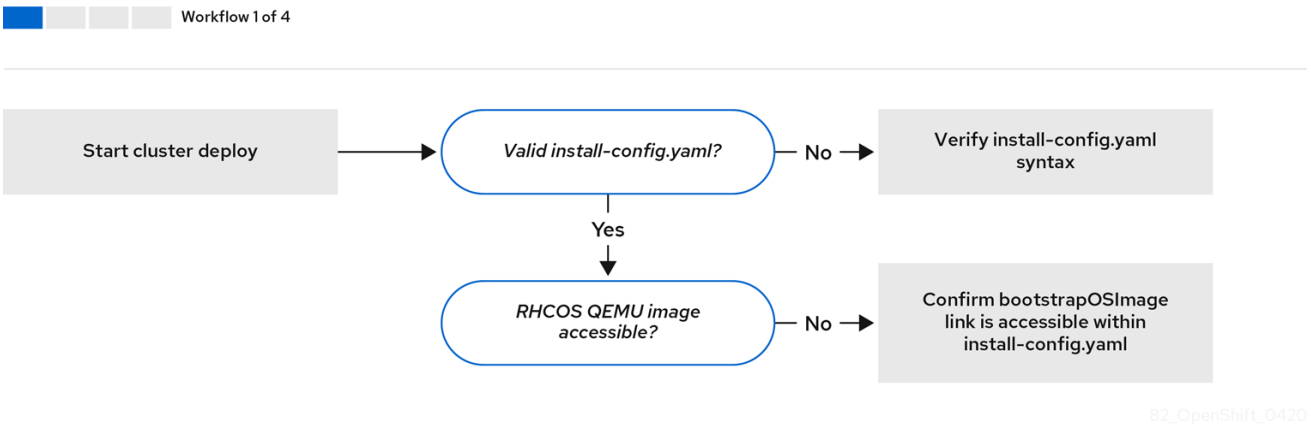
```
$ cd ; /bin/rm -rf auth/ bootstrap.ign master.ign worker.ign metadata.json \
.openshift_install.log .openshift_install_state.json
```

3. 새 매니페스트 및 Ignition 구성 파일을 생성합니다. 자세한 내용은 "Kubernetes 매니페스트 및 Ignition 구성 파일 생성"을 참조하십시오.
4. 설치 프로그램에서 생성한 새 부트스트랩, 컨트롤 플레인 및 컴퓨팅 노드 Ignition 구성 파일을 HTTP 서버에 업로드합니다. 이 명령은 이전 Ignition 파일을 덮어씁니다.

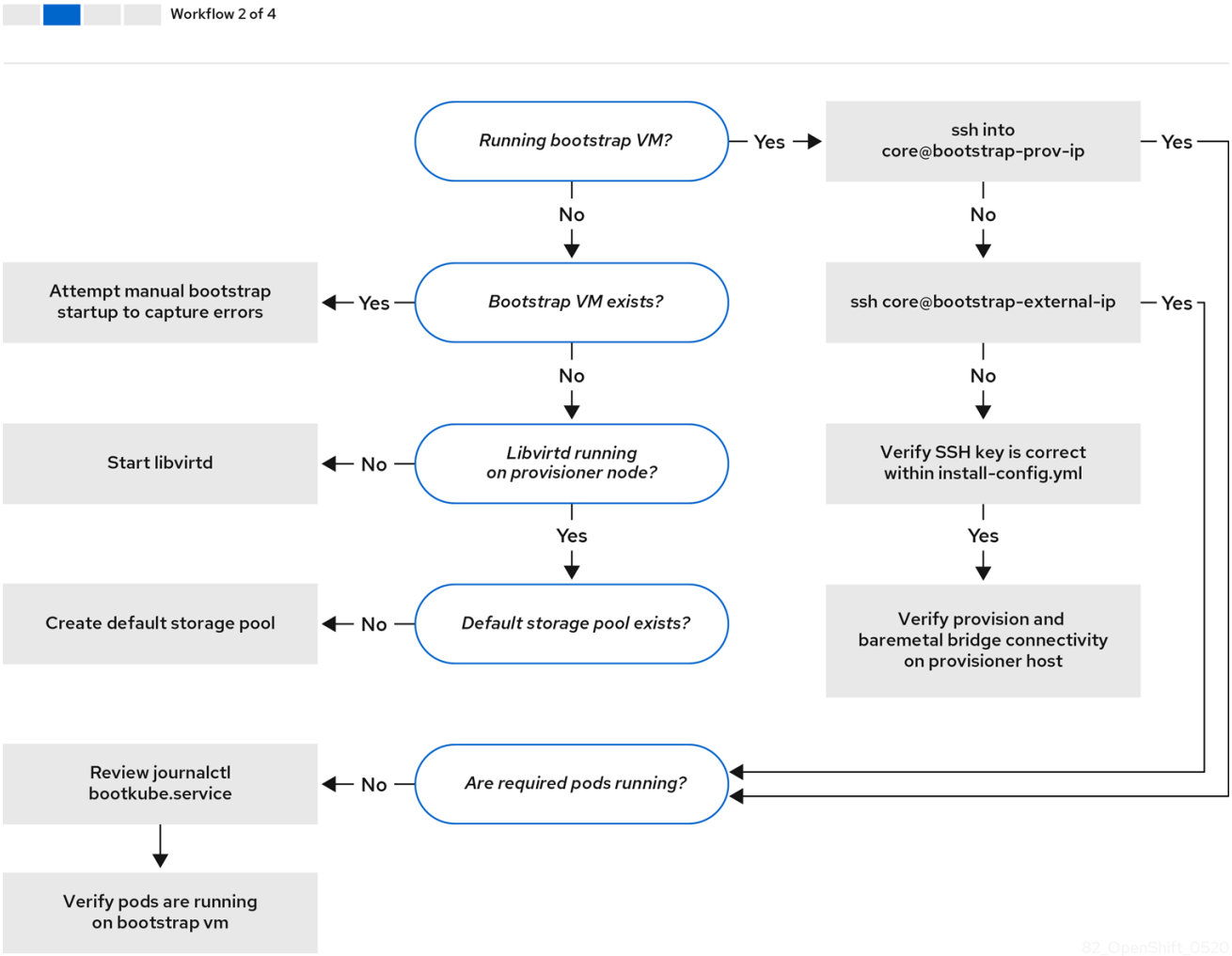
1.4. 문제 해결

1.4.1. 설치 프로그램 워크 플로우 문제 해결

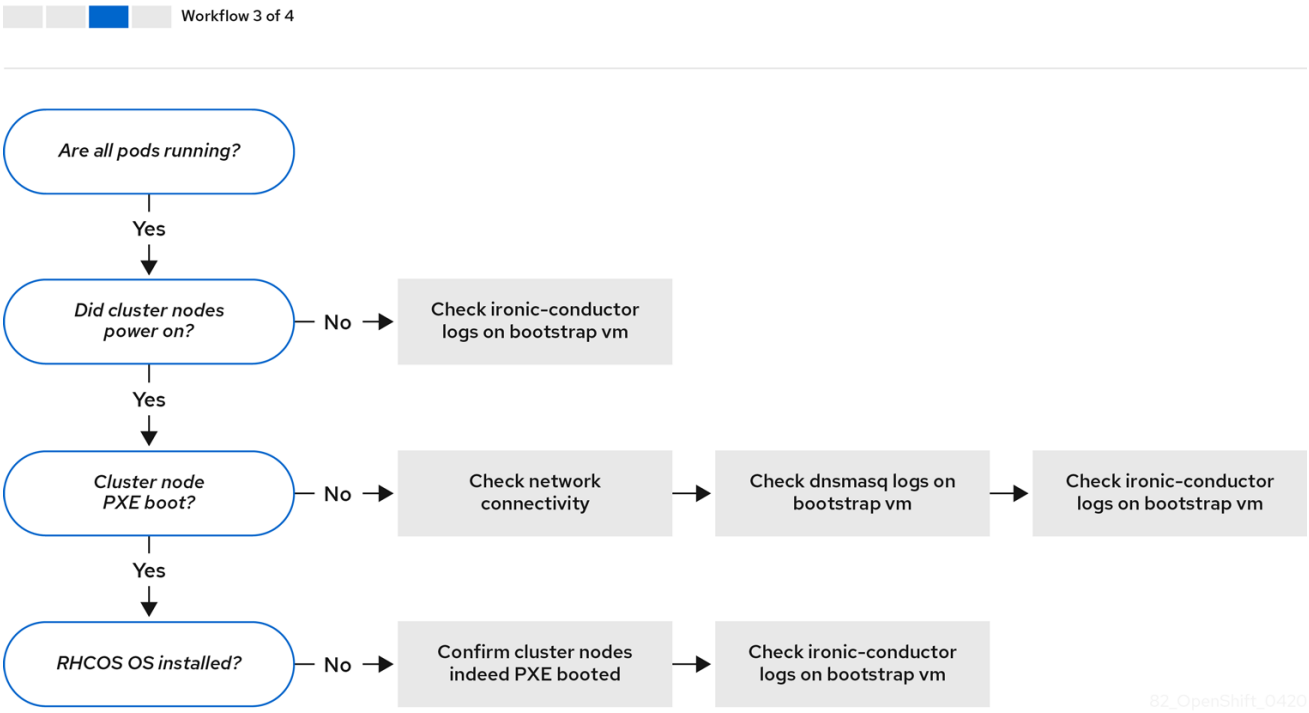
설치 환경의 문제를 해결하기 전에 베어 메탈에서 설치 프로그램이 제공하는 설치의 전체 흐름을 이해하는 것이 중요합니다. 아래 다이어그램은 환경에서 단계별 분석과 함께 문제 해결 흐름을 보여줍니다.



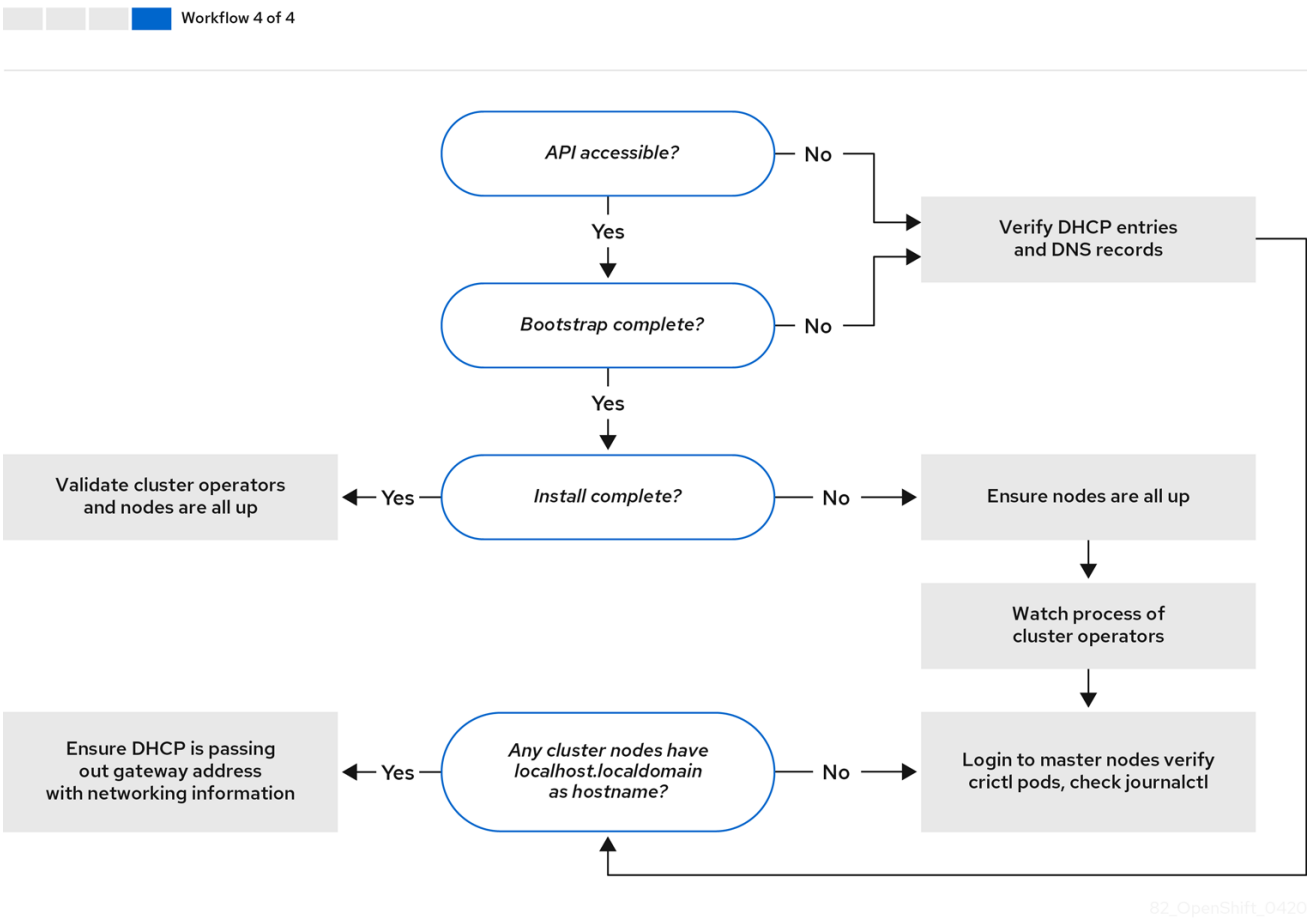
*워크 플로우 1/4*은 **install-config.yaml** 파일에 오류가 있거나 RHCOS (Red Hat Enterprise Linux CoreOS) 이미지에 액세스할 수 없는 경우 문제 해결의 워크 플로우를 보여줍니다. 문제 해결에 대한 제안은 Troubleshooting **install-config.yaml**에서 참조하십시오.



*워크 플로우 2/4*는 부트스트랩 VM 문제, 클러스터 노드를 부팅할 수 없는 부트스트랩 VM 및 로그 검사에 대한 문제 해결 워크 플로우를 보여줍니다. **provisioning** 네트워크없이 OpenShift Container Platform 클러스터를 설치할 때 이 워크플로가 적용되지 않습니다.



*워크 플로우 3/4*은 PXE 부팅이 되지 않는 클러스터 노드 에 대한 문제 해결의 워크 플로우를 보여줍니다.



*워크 플로우 4/4*는 액세스할 수 없는 API부터 검증된 설치까지의 문제 해결 워크 플로우를 보여줍니다.

1.4.2. install-config.yaml 문제 해결

install-config.yaml 설정 파일은 OpenShift Container Platform 클러스터의 일부인 모든 노드를 나타냅니다. 이 파일에는 **apiVersion**, **baseDomain**, **imageContentSources** 및 가상 IP 주소로 구성되지만 이에 국한되지 않는 필수 옵션이 포함되어 있습니다. OpenShift Container Platform 클러스터 배포 초기에 오류가 발생하면 **install-config.yaml** 구성 파일에 오류가 있을 수 있습니다.

프로세스

1. YAML-tips의 지침을 사용합니다.
2. syntax-check를 사용하여 YAML 구문이 올바른지 확인합니다.
3. RHCOS (Red Hat Enterprise Linux CoreOS) QEMU 이미지가 올바르게 정의되어 있고 **install-config.yaml**에서 제공되는 URL을 통해 액세스할 수 있는지 확인합니다. 예를 들면 다음과 같습니다.

```
$ curl -s -o /dev/null -I -w "%{http_code}\n" http://webserver.example.com:8080/rhcos-44.81.202004250133-0-qemu.x86_64.qcow2.gz?sha256=7d884b46ee54fe87bbc3893bf2aa99af3b2d31f2e19ab5529c60636fbd0f1ce7
```

출력이 **200**이면 부트스트랩 VM 이미지를 저장하는 웹 서버의 유효한 응답이 있습니다.

1.4.3. 부트스트랩 VM 문제

OpenShift Container Platform 설치 프로그램은 OpenShift Container Platform 클러스터 노드의 프로비저닝을 처리하는 부트스트랩 노드 가상 머신을 시작합니다.

프로세스

1. 설치 프로그램을 트리거한 후 약 10~15 분 후에 **virsh** 명령을 사용하여 부트스트랩 VM이 작동하는지 확인합니다.

```
$ sudo virsh list
```

```
 Id    Name                           State
----------------------------------------------------
 12    openshift-xf6fq-bootstrap      running
```

참고

부트스트랩 VM의 이름은 항상 클러스터 이름으로 시작하여 그 뒤에 임의의 문자 집합이 있고 "bootstrap"이라는 단어로 끝납니다.

부트스트랩 VM이 10-15 분 후에도 실행되지 않으면 실행되지 않은 이유에 대한 문제를 해결합니다. 발생 가능한 문제는 다음과 같습니다.

2. **libvirtd**가 시스템에서 실행 중인지 확인합니다.

```
$ systemctl status libvirtd
```

```
● libvirtd.service - Virtualization daemon
   Loaded: loaded (/usr/lib/systemd/system/libvirtd.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2020-03-03 21:21:07 UTC; 3 weeks 5 days ago
     Docs: man:libvirtd(8)
           https://libvirt.org
 Main PID: 9850 (libvirtd)
    Tasks: 20 (limit: 32768)
   Memory: 74.8M
   CGroup: /system.slice/libvirtd.service
           ├─ 9850 /usr/sbin/libvirtd
```

부트스트랩 VM이 작동하는 경우 해당 VM에 로그인합니다.

3. **virsh console** 명령을 사용하여 부트스트랩 VM의 IP 주소를 찾습니다.

```
$ sudo virsh console example.com
```

```
Connected to domain example.com
Escape character is ^]

Red Hat Enterprise Linux CoreOS 43.81.202001142154.0 (Ootpa) 4.3
SSH host key: SHA256:BRWJktXZgQQRY5zjuAV0IKZ4WM7i4TiUyMVanqu9Pqg (ED25519)
SSH host key: SHA256:7+iKGA7VtG5szmk2jB5gl/5EZ+SNcJ3a2g23o0lnIio (ECDSA)
SSH host key: SHA256:DH5VWhvhvagOTaLsYiVNse9ca+ZSW/30OOMed8rIGOc (RSA)
```

```
ens3:  fd35:919d:4042:2:c7ed:9a9f:a9ec:7
ens4: 172.22.0.2 fe80::1d05:e52e:be5d:263f
localhost login:
```

중요

provisioning 네트워크 없이 OpenShift Container Platform 클러스터를 배포하는 경우, **172.22.0.2**와 같은 개인 IP 주소가 아닌 공용 IP 주소를 사용해야 합니다.

4. IP 주소를 받으면 **ssh** 명령을 사용하여 부트스트랩 VM에 로그인합니다.

참고

이전 단계의 콘솔 출력에서 **ens3**에서 제공되는 IPv6 IP 주소 또는 **ens4**에서 제공되는 IPv4 IP를 사용할 수 있습니다.

```
$ ssh core@172.22.0.2
```

부트스트랩 VM에 성공적으로 로그인하지 못한 경우 다음 시나리오 중 하나가 발생했을 가능성이 있습니다.

- **172.22.0.0/24** 네트워크에 연결할 수 없습니다. 프로비저닝 호스트, 특히 **provisioning** 네트워크 브리지의 네트워크 연결을 확인합니다. **provisioning** 네트워크를 사용하지 않는 경우에는 이 문제가 발생하지 않습니다.
- 공용 네트워크를 통해 부트스트랩 VM에 연결할 수 없습니다. **baremetal** 네트워크에서 SSH를 시도할 때 **provisioner** 호스트, 특히 **baremetal** 네트워크 브리지의 연결을 확인합니다.
- **Permission denied (publickey, password, keyboard-interactive)**문제가 발생했습니다. 부트스트랩 VM에 액세스하려고하면 **Permission denied** 오류가 발생할 수 있습니다. VM에 로그인하려는 사용자의 SSH 키가 **install-config.yaml** 파일에 설정되어 있는지 확인합니다.

1.4.3.1. 부트스트랩 VM은 클러스터 노드를 부팅할 수 없습니다.

배포 중에 부트스트랩 VM이 클러스터 노드를 부팅하지 못하여 VM이 RHCOS 이미지로 노드를 프로비저닝하지 못할 수 있습니다. 이 시나리오는 다음과 같은 이유로 발생할 수 있습니다.

- **install-config.yaml** 파일 관련 문제
- 베어 메탈 네트워크를 통한 대역 외 네트워크 액세스 문제

이 문제를 확인하기 위해 **ironic**과 관련된 세 가지 컨테이너를 사용할 수 있습니다.

- **ironic-api**
- **ironic-conductor**
- **ironic-inspector**

프로세스

1. 부트스트랩 VM에 로그인합니다.

```
$ ssh core@172.22.0.2
```

2. 컨테이너 로그를 확인하려면 다음을 실행합니다.

```
[core@localhost ~]$ sudo podman logs -f <container-name>
```

<container-name>을 **ironic-api**, **ironic-conductor** 또는 **ironic-inspector** 중 하나로 바꿉니다. 컨트롤 플레인 노드가 PXE를 통해 부팅되지 않는 문제가 발생하면 **ironic-conductor** Pod를 확인하십시오. **ironic-conductor** Pod는 IPMI를 통해 노드에 로그인을 시도하기 때문에 클러스터 노드 부팅 시도에 대한 가장 자세한 정보가 포함되어 있습니다.

가능한 이유

배포가 시작되면 클러스터 노드가 **ON** 상태 일 수 있습니다.

해결책

IPMI를 통해 설치를 시작하기 전에 OpenShift Container Platform 클러스터 노드의 전원을 끄십시오.

```
$ ipmitool -I lanplus -U root -P <password> -H <out-of-band-ip> power off
```

1.4.3.2. 로그 검사

RHCOS 이미지를 다운로드하거나 액세스하는 데 문제가 발생하면 먼저 **install-config.yaml** 구성 파일에서 URL이 올바른지 확인합니다.

RHCOS 이미지를 호스팅하는 내부 웹 서버의 예

```
bootstrapOSImage: http://<ip:port>/rhcos-43.81.202001142154.0-qemu.x86_64.qcow2.gz?
sha256=9d999f55ff1d44f7ed7c106508e5deecd04dc3c06095d34d36bf1cd127837e0c
clusterOSImage: http://<ip:port>/rhcos-43.81.202001142154.0-openstack.x86_64.qcow2.gz?
sha256=a1bda656fa0892f7b936fdc6b6a6086bddaed5dafacedcd7a1e811abb78fe3b0
```

ipa-downloader 및 **coreos-downloader** 컨테이너는 **install-config.yaml** 구성 파일에 지정된 웹 서버 또는 외부 quay.io 레지스트리에서 리소스를 다운로드합니다. 다음 두 컨테이너가 실행 중인지 확인하고 필요에 따라 로그를 검사합니다.

- **ipa-downloader**
- **coreos-downloader**

프로세스

1. 부트스트랩 VM에 로그인합니다.

```
$ ssh core@172.22.0.2
```

2. 부트스트랩 VM 내에서 **ipa-downloader** 및 **coreos-downloader** 컨테이너의 상태를 확인합니다.

```
[core@localhost ~]$ sudo podman logs -f ipa-downloader
```

```
[core@localhost ~]$ sudo podman logs -f coreos-downloader
```

부트스트랩 VM이 이미지의 URL에 액세스할 수 없는 경우 **curl** 명령을 사용하여 VM이 이미지에 액세스할 수 있는지 확인합니다.

3. 배포 단계에서 모든 컨테이너가 시작되었는지 여부를 나타내는 **bootkube** 로그를 검사하려면 다음을 실행합니다.

```
[core@localhost ~]$ journalctl -xe
```

```
[core@localhost ~]$ journalctl -b -f -u bootkube.service
```

4. **dnsmasq**, **mariadb**, **httpd** 및 **ironic**를 포함한 모든 Pod가 실행 중인지 확인합니다.

```
[core@localhost ~]$ sudo podman ps
```

5. Pod에 문제가 있는 경우 문제가있는 컨테이너의 로그를 확인합니다. **ironic-api**의 로그를 확인하려면 다음을 실행합니다.

```
[core@localhost ~]$ sudo podman logs <ironic-api>
```

1.4.4. 클러스터 노드는 PXE 부팅 불가능

OpenShift Container Platform 클러스터 노드가 PXE 부팅을 하지 않는 경우 PXE 부팅이 되지 않는 클러스터 노드에서 다음 검사를 실행합니다. 이 절차는 **provisioning** 네트워크없이 OpenShift Container Platform 클러스터를 설치할 때 적용되지 않습니다.

프로세스

1. **provisioning** 네트워크에 대한 네트워크 연결을 확인하십시오.
2. **provisioning** 네트워크의 NIC에서 PXE가 활성화되어 있고 다른 모든 NIC에 대해 PXE가 비활성화되어 있는지 확인합니다.
3. **install-config.yaml** 구성 파일에 **provisioning** 네트워크에 연결된 NIC에 대한 적절한 하드웨어 프로필과 부팅 MAC 주소가 있는지 확인합니다. 예를 들면 다음과 같습니다.

컨트롤 플레인 노드 설정

```
bootMACAddress: 24:6E:96:1B:96:90 # MAC of bootable provisioning NIC
hardwareProfile: default          #control plane node settings
```

작업자 노드 설정

```
bootMACAddress: 24:6E:96:1B:96:90 # MAC of bootable provisioning NIC
hardwareProfile: unknown          #worker node settings
```

1.4.5. API에 액세스 불가능

클러스터가 실행 중이고 클라이언트가 API에 액세스할 수 없는 경우 도메인 이름 확인 문제가 API에 대한 액세스를 방해할 수 있습니다.

절차

1. **호스트 이름 확인:** 클러스터 노드를 확인하여 **localhost.localdomain** 뿐만 아니라 정규화된 도메인 이름이 있는지 확인합니다. 예를 들면 다음과 같습니다.

```
$ hostname
```

호스트 이름이 설정되지 않은 경우 올바른 호스트 이름을 설정합니다. 예를 들면 다음과 같습니다.

```
$ hostnamectl set-hostname <hostname>
```

2. **잘못된 이름 확인: dig** 및 **nslookup** 을 사용하여 DNS 서버에서 각 노드의 이름 확인이 올바른지 확인합니다. 예를 들면 다음과 같습니다.

```
$ dig api.<cluster-name>.example.com
```

```
; <<>> DiG 9.11.4-P2-RedHat-9.11.4-26.P2.el8 <<>> api.<cluster-name>.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 37551
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 866929d2f8e8563582af23f05ec44203d313e50948d43f60 (good)
;; QUESTION SECTION:
;api.<cluster-name>.example.com. IN A

;; ANSWER SECTION:
api.<cluster-name>.example.com. 10800 IN A 10.19.13.86

;; AUTHORITY SECTION:
<cluster-name>.example.com. 10800 IN NS <cluster-name>.example.com.

;; ADDITIONAL SECTION:
<cluster-name>.example.com. 10800 IN A 10.19.14.247

;; Query time: 0 msec
;; SERVER: 10.19.14.247#53(10.19.14.247)
;; WHEN: Tue May 19 20:30:59 UTC 2020
;; MSG SIZE  rcvd: 140
```

이 예의 출력은 **api. <cluster-name> .example.com** VIP의 적절한 IP 주소가 **10.19.13.86**임을 보여줍니다. 이 IP 주소는 **baremetal** 네트워크에 있어야합니다.

1.4.6. 이전 설치 정리

이전 배포에 실패한 경우 OpenShift Container Platform을 다시 배포하기 전에 실패한 시도에서 아티팩트를 제거합니다.

프로세스

1. OpenShift Container Platform 클러스터를 설치하기 전에 모든 베어 메탈 노드의 전원을 끕니다.

```
$ ipmitool -I lanplus -U <user> -P <password> -H <management-server-ip> power off
```

2. 이전 배포에서 남은 모든 이전 부트스트랩 리소스를 제거합니다.

```
for i in $(sudo virsh list | tail -n +3 | grep bootstrap | awk {'print $2'});
do
  sudo virsh destroy $i;
  sudo virsh undefine $i;
  sudo virsh vol-delete $i --pool $i;
  sudo virsh vol-delete $i.ign --pool $i;
  sudo virsh pool-destroy $i;
  sudo virsh pool-undefine $i;
done
```

3. Terraform이 실패하지 않도록 **clusterconfigs** 디렉터리에서 다음을 제거합니다.

```
$ rm -rf ~/clusterconfigs/auth ~/clusterconfigs/terraform* ~/clusterconfigs/tls
~/clusterconfigs/metadata.json
```

1.4.7. 레지스트리 생성 문제

비 연결 레지스트리를 만들 때 레지스트리 미러링을 시도하는 경우 "User Not Authorized" 오류가 발생할 수 있습니다. 이 오류는 기존 **pull-secret.txt** 파일에 새 인증을 추가할 수 없는 경우 발생할 수 있습니다.

프로세스

1. 인증이 성공했는지 확인합니다.

```
$ /usr/local/bin/oc adm release mirror \
  -a pull-secret-update.json
  --from=$UPSTREAM_REPO \
  --to-release-image=$LOCAL_REG/$LOCAL_REPO:${VERSION} \
  --to=$LOCAL_REG/$LOCAL_REPO
```

참고

설치 이미지 미러링에 사용되는 변수의 출력 예:

```
UPSTREAM_REPO=${RELEASE_IMAGE}
LOCAL_REG=<registry_FQDN>:<registry_port>
LOCAL_REPO='ocp4/openshift4'
```

RELEASE_IMAGE 및 **VERSION**의 값은 **OpenShift 설치 환경 설정** 섹션의 **OpenShift 설치 프로그램 가져오기** 단계에서 설정됩니다.

2. 레지스트리를 미러링 후 연결이 끊긴 환경에서 이에 액세스할 수 있는지 확인합니다.

```
$ curl -k -u <user>:<password> https://registry.example.com:<registry-port>/v2/_catalog
{"repositories":["<Repo-Name>"]}
```

1.4.8. 기타 문제

1.4.8.1. runtime network not ready 오류 해결

클러스터 배포 후 다음과 같은 오류가 발생할 수 있습니다.

```
`runtime network not ready: NetworkReady=false reason:NetworkPluginNotReady message:Network plugin returns error: Missing CNI default network`
```

Cluster Network Operator 설치 프로그램이 생성한 특수 개체에 대응하여 네트워킹 구성 요소를 배포해야 합니다. 컨트롤 플레인 (마스터) 노드가 시작된 후 부트스트랩 컨트롤 플레인이 중지되기 전에 설치 프로세스 초기에 실행됩니다. 컨트롤 플레인 (마스터) 노드를 시작할 때 오랜 지연이나 **apiserver** 통신 문제와 같은 미묘한 설치 프로그램 문제가 표시될 수 있습니다.

프로세스

1. **openshift-network-operator** 네임 스페이스에서 Pod를 검사합니다.

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

```
NAME                                READY STATUS            RESTARTS   AGE
pod/network-operator-69dfd7b577-bg89v   0/1   ContainerCreating 0          149m
```

2. **provisioner** 노드에서 네트워크 구성이 존재하는지 확인합니다.

```
$ kubectl get network.config.openshift.io cluster -oyaml
```

```
apiVersion: config.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  serviceNetwork:
  - 172.30.0.0/16
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  networkType: OpenShiftSDN
```

존재하지 않는 경우 설치 프로그램이 이를 생성하지 않은 것입니다. 설치 프로그램이 생성하지 않은 이유를 확인하려면 다음을 실행합니다.

```
$ openshift-install create manifests
```

3. **network-operator**가 실행되고 있는지 확인합니다.

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

4. 로그를 검색합니다.

```
$ kubectl -n openshift-network-operator logs -l "name=network-operator"
```

3 개 이상의 컨트롤 플레인 (마스터) 노드가 있는 고가용성 클러스터에서 Operator는 리더의 선택을 실행하고 다른 Operator는 절전 모드로 전환합니다. 자세한 내용은 Troubleshooting을 참조하십시오.

1.4.8.2. DHCP를 통해 올바른 IPv6 주소를 얻지 못하는 클러스터 노드

클러스터 노드가 DHCP를 통해 올바른 IPv6 주소를 얻지 못하는 경우 다음을 확인합니다.

1. 예약 된 IPv6 주소가 DHCP 범위 밖에 있는지 확인합니다.
2. DHCP 서버의 IP 주소 예약에서 예약에 올바른 DHCP 고유 식별자 (DUID)가 지정되어 있는지 확인합니다. 예를 들면 다음과 같습니다.

```
# This is a dnsmasq dhcp reservation, 'id:00:03:00:01' is the client id and '18:db:f2:8c:d5:9f' is the MAC Address for the NIC
id:00:03:00:01:18:db:f2:8c:d5:9f,openshift-master-1,[2620:52:0:1302::6]
```

3. 경로 알림 (Route Announcement)이 제대로 작동하는지 확인합니다.
4. DHCP 서버가 IP 주소 범위를 제공하는 데 필요한 인터페이스에서 수신하고 있는지 확인합니다.

1.4.8.3. DHCP를 통해 올바른 호스트 이름을 얻지 못하는 클러스터 노드

IPv6 배포 중에 클러스터 노드는 DHCP를 통해 호스트 이름을 검색해야 합니다. 경우에 따라 **NetworkManager**가 호스트 이름을 즉시 할당하지 않을 수 있습니다. 컨트롤 플레인 (마스터) 노드는 다음과 같은 오류를 보고할 수 있습니다.

```
Failed Units: 2
  NetworkManager-wait-online.service
  nodeip-configuration.service
```

이 오류는 클러스터 노드가 DHCP 서버에서 호스트 이름을 받지 않고 부팅되었을 가능성이 있음을 나타냅니다. 이로 인해 **kubelet**이 **localhost.localdomain** 호스트 이름으로 부팅됩니다. 이 문제를 해결하려면 노드가 호스트 이름을 업데이트하도록 합니다.

프로세스

1. **hostname**을 검색합니다.

```
[core@master-X ~]$ hostname
```

   호스트 이름이 **localhost**인 경우 다음 단계를 진행합니다.

참고

여기서 **X**는 컨트롤 플레인 노드(마스터 노드라고도 함) 번호입니다.

2. 클러스터 노드가 DHCP 임대를 갱신하도록 합니다.

```
[core@master-X ~]$ sudo nmcli con up "<bare-metal-nic>"
```

   <bare-metal-nic>을 **baremetal** 네트워크에 해당하는 유선 연결로 바꿉니다.

3. **hostname** 다시 확인하십시오.

```
[core@master-X ~]$ hostname
```

4. 호스트 이름이 여전히 **localhost.localdomain**인 경우 **NetworkManager**를 다시 시작합니다.

```
[core@master-X ~]$ sudo systemctl restart NetworkManager
```

5. 호스트 이름이 여전히 **localhost.localdomain** 인 경우 몇 분 기다린 후 다시 확인하십시오. 호스트 이름이 **localhost.localdomain**으로 남아 있으면 이전 단계를 반복합니다.

6. **nodeip-configuration** 서비스를 다시 시작합니다.

```
[core@master-X ~]$ sudo systemctl restart nodeip-configuration.service
```

   이 서비스는 올바른 호스트 이름 참조를 사용하여 **kubelet** 서비스를 재구성합니다.

7. 이전 단계에서 kubelet이 변경되었으므로 단위 파일 정의를 다시 로드하십시오.

```
[core@master-X ~]$ sudo systemctl daemon-reload
```

8. **kubelet** 서비스를 다시 시작합니다.

```
[core@master-X ~]$ sudo systemctl restart kubelet.service
```

9. **kubelet** 이 올바른 호스트 이름으로 부팅되었는지 확인합니다.

```
[core@master-X ~]$ sudo journalctl -fu kubelet.service
```

클러스터 가동 후 (예: 클러스터를 다시 시작) 클러스터 노드가 DHCP를 통해 올바른 호스트 이름을 얻지 못하는 경우 클러스터에 **csr**은 보류 처리됩니다. **csr**을 승인 **하지 마십시오.** 그렇지 않으면 다른 문제가 발생할 수 있습니다.

CSR 처리

1. 클러스터에서 CSR을 가져옵니다.

```
$ oc get csr
```

2. 보류중인 **CSR** 에 **Subject Name: localhost.localdomain**이 포함되어 있는지 확인합니다.

```
$ oc get csr <pending_csr> -o jsonpath='{.spec.request}' | base64 --decode | openssl req -noout -text
```

3. **Subject Name: localhost.localdomain**이 포함된 모든 **csr**을 제거합니다.

```
$ oc delete csr <wrong_csr>
```

1.4.8.4. 루트가 엔드 포인트에 도달하지 않음

설치 프로세스 중에 VRRP (Virtual Router Redundancy Protocol) 충돌이 발생할 수 있습니다. 특정 클러스터 이름을 사용하여 클러스터 배포의 일부였던 이전에 사용된 OpenShift Container Platform 노드가 여전히 실행 중이지만 동일한 클러스터 이름을 사용하는 현재 OpenShift Container Platform 클러스터 배포의 일부가 아닌 경우 이러한 충돌이 발생할 수 있습니다. 예를 들어 클러스터는 클러스터 이름 **openshift**를 사용하여 3 개의 컨트롤 플레인 (마스터) 노드와 3 개의 작업자 노드를 배포합니다. 나중에 다른 설치에서 동일한 클러스터 이름 **openshift**를 사용하지만 이 재배포에서는 3 개의 컨트롤 플레인 (마스터) 노드 만 설치하여 이전 배포의 작업자 노드 3 개를 **ON** 상태로 유지합니다. 이로 인해 VRID (Virtual Router Identifier) 충돌 및 VRRP 충돌이 발생할 수 있습니다.

1. 루트를 가져옵니다.

```
$ oc get route oauth-openshift
```

2. 서비스 엔드 포인트를 확인합니다.

```
$ oc get svc oauth-openshift
```

```
NAME              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
oauth-openshift   ClusterIP   172.30.19.162   <none>        443/TCP   59m
```

3. 컨트롤 플레인 (마스터) 노드에서 서비스에 연결을 시도합니다.

```
[core@master0 ~]$ curl -k https://172.30.19.162
```

```
{
  "kind": "Status",
  "apiVersion": "v1",
  "metadata": {
  },
  "status": "Failure",
  "message": "forbidden: User \"system:anonymous\" cannot get path \"/\"",
  "reason": "Forbidden",
  "details": {
  },
  "code": 403
```

4. **provisioner** 노드에서 **authentication-operator** 오류를 식별합니다.

```
$ oc logs deployment/authentication-operator -n openshift-authentication-operator
```

```
Event(v1.ObjectReference{Kind:"Deployment", Namespace:"openshift-authentication-operator", Name:"authentication-operator", UID:"225c5bd5-b368-439b-9155-5fd3c0459d98", APIVersion:"apps/v1", ResourceVersion:"", FieldPath:""}): type: 'Normal' reason: 'OperatorStatusChanged' Status for clusteroperator/authentication changed: Degraded message changed from "IngressStateEndpointsDegraded: All 2 endpoints for oauth-server are reporting"
```

해결책

1. 모든 배포의 클러스터 이름이 고유한지 확인하여 충돌이 발생하지 않도록합니다.

2. 동일한 클러스터 이름을 사용하는 클러스터 배포의 일부가 아닌 잘못된 노드 모두를 종료합니다. 그렇지 않으면 OpenShift Container Platform 클러스터의 인증 pod가 정상적으로 시작되지 않을 수 있습니다.

1.4.8.5. Firstboot 동안 Ignition 실패

Firstboot 중에 Ignition 설정이 실패할 수 있습니다.

프로세스

1. Ignition 설정이 실패한 노드에 연결합니다.

```
Failed Units: 1
  machine-config-daemon-firstboot.service
```

2. **machine-config-daemon-firstboot** 서비스를 다시 시작합니다.

```
[core@worker-X ~]$ sudo systemctl restart machine-config-daemon-firstboot.service
```

1.4.8.6. NTP가 동기화되지 않음

OpenShift Container Platform 클러스터를 배포하려면 클러스터 노드 간의 NTP 시계가 동기화되어야합니다. 동기화된 시계가 없으면 시간 차이가 2 초보다 크면 클럭 드리프트로 인해 배포 실패할 수 있습니다.

프로세스

1. 클러스터 노드의 **AGE** 차이를 확인하십시오. 예를 들면 다음과 같습니다.

```
$ oc get nodes
```

```
NAME                         STATUS   ROLES    AGE   VERSION
master-0.cloud.example.com   Ready    master   145m   v1.16.2
master-1.cloud.example.com   Ready    master   135m   v1.16.2
master-2.cloud.example.com   Ready    master   145m   v1.16.2
worker-2.cloud.example.com   Ready    worker   100m   v1.16.2
```

2. 클럭 드리프트로 인한 일관성없는 시간 지연을 확인하십시오. 예를 들면 다음과 같습니다.

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

```
master-1    error registering master-1  ipmi://<out-of-band-ip>
```

```
$ sudo timedatectl
```

```
               Local time: Tue 2020-03-10 18:20:02 UTC
           Universal time: Tue 2020-03-10 18:20:02 UTC
                 RTC time: Tue 2020-03-10 18:36:53
                Time zone: UTC (UTC, +0000)
System clock synchronized: no
              NTP service: active
          RTC in local TZ: no
```

기존 클러스터에서 클럭 드리프트 처리

1. **chrony.conf** 파일의 내용을 작성하고 **base64** 문자열로 인코딩하십시오. 예를 들면 다음과 같습니다.

```
$ cat << EOF | base 64
server <NTP-server> iburst 1
stratumweight 0
driftfile /var/lib/chrony/drift
```

```
rtcsync
makestep 10 3
bindcmdaddress 127.0.0.1
bindcmdaddress ::1
keyfile /etc/chrony.keys
commandkey 1
generatecommandkey
noclientlog
logchange 0.5
logdir /var/log/chrony
EOF
```

1 **<NTP-server>**를 NTP 서버의 IP 주소로 바꿉니다. 출력을 복사합니다.

```
[text-in-base-64]
```

2. **MachineConfig** 개체를 만들고 **base64** 문자열을 이전 단계의 출력에서 생성된 **[text-in-base-64]** 문자열로 바꿉니다. 다음 예제는 컨트롤 플레인 (마스터) 노드에 파일을 추가합니다. 작업자 노드의 파일을 변경하거나 작업자 역할에 대한 추가 시스템 설정을 만들 수 있습니다.

```
$ cat << EOF > ./99_masters-chrony-configuration.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  creationTimestamp: null
  labels:
    machineconfiguration.openshift.io/role: master
  name: 99-master-etc-chrony-conf
spec:
  config:
    ignition:
      config: {}
      security:
        tls: {}
      timeouts: {}
      version: 3.1.0
    networkd: {}
    passwd: {}
    storage:
      files:
      - contents:
          source: data:text/plain;charset=utf-8;base64,[text-in-base-64]
        group:
          name: root
        mode: 420
        overwrite: true
        path: /etc/chrony.conf
        user:
          name: root
  osImageURL: ""
```

1 **[text-in-base-64]**를 base64 문자열로 바꿉니다.

3. 설정 파일의 백업 사본을 만듭니다. 예를 들면 다음과 같습니다.

```
$ cp 99_masters-chrony-configuration.yaml 99_masters-chrony-configuration.yaml.backup
```

4. 설정 파일을 적용합니다.

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

5. **System clock synchronized** 값이 **yes** 인지 확인하십시오.

```
$ sudo timedatectl
```

```
               Local time: Tue 2020-03-10 19:10:02 UTC
           Universal time: Tue 2020-03-10 19:10:02 UTC
                 RTC time: Tue 2020-03-10 19:36:53
                Time zone: UTC (UTC, +0000)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

배포 전에 클럭 동기화를 설정하려면 매니페스트 파일을 생성하고이 파일을 **openshift** 디렉터리에 추가합니다. 예를 들면 다음과 같습니다.

```
$ cp chrony-masters.yaml ~/clusterconfigs/openshift/99_masters-chrony-configuration.yaml
```

그런 다음 계속해서 클러스터를 만듭니다.

1.4.9. 설치 확인

설치 후 설치 프로그램이 노드 및 pod를 성공적으로 배포했는지 확인합니다.

프로세스

1. OpenShift Container Platform 클러스터 노드가 적절하게 설치되면 **STATUS** 열에 **Ready** 상태가 표시됩니다.

```
$ oc get nodes
```

```
NAME                 STATUS   ROLES          AGE  VERSION
master-0.example.com   Ready    master,worker   4h   v1.16.2
master-1.example.com   Ready    master,worker   4h   v1.16.2
master-2.example.com   Ready    master,worker   4h   v1.16.2
```

2. 설치 프로그램이 모든 pod를 성공적으로 배포했는지 확인합니다. 다음 명령은 아직 실행 중이거나 출력의 일부로 완료된 모든 pod를 제거합니다.

```
$ oc get pods --all-namespaces | grep -iv running | grep -iv complete
```