OpenShift Container Platform 4.6

インストーラーでプロビジョニングされるクラスターのベアメタルへのデプロイ

IPI OpenShift Container Platform ベアメタルクラスターのインストール

Last Updated: 2023-02-23

OpenShift Container Platform 4.6 インストーラーでプロビジョニングされるクラスターのベアメタルへのデプロイ

IPI OpenShift Container Platform ベアメタルクラスターのインストール

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.

法律上の通知

Copyright © 2022 | You need to change the HOLDER entity in the en-US/Deploying_installer-provisioned_clusters_on_bare_metal.ent file |.

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 ® is the registered trademark of Linus Torvalds in the United States and other countries.

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

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

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

Node.js ® 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 ® 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.

概要

本書では、インストーラーでプロビジョニングされるクラスターを含むベアメタルのインフラストラクチャーに OpenShift Container Platform クラスターをインストールする方法について説明します。

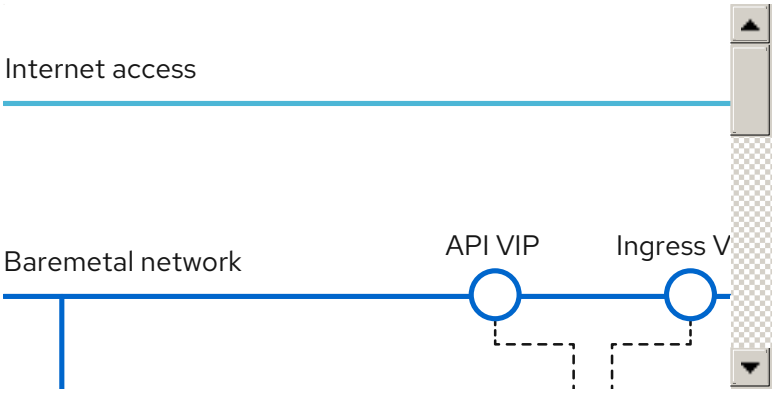
目次

第1章 インストーラーでプロビジョニングされるクラスターのベアメタルへのデプロイ

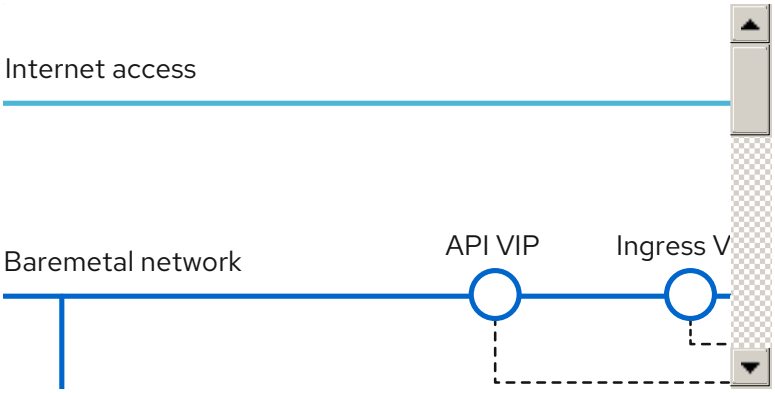
1.1. 概要

インストーラーでプロビジョニングされるインストールは、OpenShift Container Platform のベアメタルノードへのインストールのサポートを提供します。本書では、インストールを正常に実行するための方法について説明します。

ベアメタルでのインストーラーでプロビジョニングされるインストールの実行時に、**provisioner** というラベルの付けられたベアメタルノード上のインストーラーによりブートストラップ仮想マシンが作成されます。ブートストラップ仮想マシンのロールは、OpenShift Container Platform クラスターのデプロイプロセスを支援することにあります。ブートストラップ仮想マシンは、ネットワークブリッジを経由して **baremetal** ネットワークおよび **provisioning** ネットワークに接続されます (存在する場合)。



OpenShift Container Platform コントロールプレーンノードのインストールが完了し、完全に機能する状態になると、インストーラーはブートストラップ仮想マシンを自動的に破棄し、仮想 IP アドレス (VIP) を適切なノードに移動します。API VIP はコントロールプレーンノードに移動し、Ingress VIP はワーカーノードに移動します。



1.2. 前提条件

OpenShift Container Platform のインストーラーでプロビジョニングされるインストールには、以下が必要です。

1. 1 つの Red Hat Enterprise Linux (RHEL) 8.x がインストールされているプロビジョナーノード。
2. 3 つのコントロールプレーンノード
3. ベースボード管理コントローラー (BMC) の各ノードへのアクセス。

4. 1つ以上のネットワーク:

    a. 1つの **必須** のルーティング可能なネットワーク

    b. 1つの **オプション** のノードのプロビジョニング用のネットワーク

    c. 1つの **オプション** の管理ネットワーク

OpenShift Container Platform のインストーラーでプロビジョニングされるインストールを開始する前に、ハードウェア環境が以下の要件を満たしていることを確認してください。

1.2.1. ノードの要件

インストーラーでプロビジョニングされるインストールには、ハードウェアノードの各種の要件があります。

- **CPU アーキテクチャー:** すべてのノードで **x86_64** CPU アーキテクチャーを使用する必要があります。
- **同様のノード:** Red Hat では、ノードにロールごとに同じ設定を指定することを推奨しています。つまり Red Hat では、同じ CPU、メモリー、ストレージ設定の同じブランドおよびモデルのノードを使用することを推奨しています。
- **ベースボード管理コントローラー: provisioner** ノードは、各 OpenShift Container Platform クラスターノードのベースボード管理コントローラー (BMC) にアクセスできる必要があります。IPMI、RedFish、または独自のプロトコルを使用できます。
- **最新の生成:** ノードは最新の生成されたノードである必要があります。インストーラーでプロビジョニングされるインストールは BMC プロトコルに依存するため、ハードウェアは IPMI 暗号化スイート 17 をサポートしている必要があります。また、RHEL 8 は RAID コントローラーの最新のドライバーが同梱されています。ノードは **provisioner** ノード用に RHEL 8 を、コントローラープレーンおよびワーカーノード用に RHCOS 8 をサポートするのに必要な新しいバージョンのノードであることを確認します。
- **レジストリーノード:** オプション: 非接続のミラーリングされていないレジストリーを設定する場合、レジストリーは独自のノードに置くことが推奨されます。
- **プロビジョナーノード:** インストーラーでプロビジョニングされるインストールには1つの **provisioner** ノードが必要です。
- **コントロールプレーン:** インストーラーでプロビジョニングされるインストールには、高可用性を実現するために 3 つのコントロールプレーンノードが必要です。
- **ワーカーノード:** 必須ではありませんが、一般的な実稼働クラスターには1つまたは複数のワーカーノードがあります。小規模なクラスターでは、開発およびテスト時の管理者および開発者に対するリソース効率が高くなります。
- **ネットワークインターフェイス:** 各ノードでは、ルーティング可能な **baremetal** ネットワークに1つ以上のネットワークインターフェイスが必要です。デプロイメントに **provisioning** ネットワークを使用する場合、各ノードに **provisioning** ネットワーク用に1つのネットワークインターフェイスが必要になります。**provisioning** ネットワークの使用はデフォルト設定です。ネットワークインターフェイスの命名は、プロビジョニングネットワーク用のコントロールプレーンノード全体で一貫している必要があります。たとえば、コントロールプレーンノードがプロビジョニングネットワークに **eth0** NIC を使用する場合は、他のコントロールプレーンノードもこれを使用する必要があります。
- Unified Extensible Firmware Interface (UEFI):インストーラーでプロビジョニングされるイン

ストールでは、**provisioning** ネットワークで IPv6 アドレスを使用する場合に、すべての OpenShift Container Platform ノードで UEFI ブートが必要になります。さらに、UEFI Device PXE Settings (UEFI デバイス PXE 設定) は **provisioning** ネットワーク NIC で IPv6 プロトコルを使用するように設定する必要がありますが、 **provisioning ネットワークを省略すると、この要件はなくなります。**

1.2.2. ネットワーク要件

OpenShift Container Platform のインストーラーでプロビジョニングされるインストールには、デフォルトで複数のネットワーク要件があります。まず、インストーラーでプロビジョニングされるインストールでは、各ベアメタルノードにオペレーティングシステムをプロビジョニングするためのルーティング不可能な **provisioning** ネットワークおよびルーティング可能な **baremetal** ネットワークを使用します。インストーラーでプロビジョニングされるインストールは **ironic-dnsmasq** をデプロイするため、ネットワークではその他の DHCP サーバーを同じブロードキャストドメイン上で実行することはできません。ネットワーク管理者は、OpenShift Container Platform クラスターの各ノードの IP アドレスを予約する必要があります。

ネットワークタイムプロトコル (NTP)

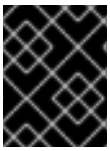
クラスターの各 OpenShift Container Platform ノードは、DHCP を使用して検出可能な Network Time Protocol (NTP) サーバーにアクセスできることが推奨されます。NTP サーバーなしでインストールすることは可能ですが、非同期サーバークロックはエラーを発生させる可能性があります。NTP サーバーを使用すると、この問題を防ぐことができます。

NIC の設定

OpenShift Container Platform は、2 つのネットワークを使用してデプロイします。

- **provisioning**: **provisioning** ネットワークは OpenShift Container Platform クラスターの一部である基礎となるオペレーティングシステムを各ノードにプロビジョニングするために使用される **オプションの** ルーティング不可能なネットワークです。**provisioning** ネットワークを使用してデプロイする場合、各ノードの最初の NIC (**eth0** または **eno1** など) は、**provisioning** ネットワークと対話する **必要があります**。

- **baremetal**: **baremetal** ネットワークはルーティング可能なネットワークです。**provisioning** ネットワークを使用してデプロイする場合、各ノードの 2 つ目の NIC (**eth1** または **eno2** など) は、**baremetal** ネットワークと対話する **必要があります**。**provisioning** ネットワークなしでデプロイする場合、**baremetal** ネットワークと対話するために各ノードで任意の NIC を使用することができます。



重要

それぞれの NIC は、適切なネットワークに対応する別個の VLAN 上にある必要があります。

DNS サーバーの設定

クライアントは、**baremetal** ネットワークで OpenShift Container Platform クラスターにアクセスします。ネットワーク管理者は、正規名の拡張がクラスター名であるサブドメインまたはサブゾーンを設定する必要があります。

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

以下に例を示します。

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

DHCP サーバーを使用するノードの IP アドレスの確保

baremetal ネットワークの場合、ネットワーク管理者は以下を含む多数の IP アドレスを予約する必要があります。

1. 2 つの仮想 IP アドレス
   - API エンドポイントの 1 つの IP アドレス
   - ワイルドカード Ingress エンドポイントの 1 つの IP アドレス
2. プロビジョナーノードの 1 つの IP アドレス
3. 各コントロールプレーン (マスター) ノード 1 つの IP アドレス
4. 各ワーカーノードの 1 つの IP アドレス (適用可能な場合)

以下の表は、完全修飾ドメイン名の具体例を示しています。API および Nameserver アドレスは、正式名の拡張子で始まります。コントロールプレーンおよびワーカーノードのホスト名は例であるため、任意のホストの命名規則を使用することができます。

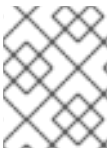
使用法	ホスト名	IP
API	**api.<cluster-name>.<domain>**	**<ip>**
Ingress LB (アプリケーション)	***.apps.<cluster-name>.<domain>**	**<ip>**
プロビジョナーノード	**provisioner.<cluster-name>.<domain>**	**<ip>**
Master-0	**openshift-master-0.<cluster-name>.<domain>**	**<ip>**
Master-1	**openshift-master-1.<cluster-name>-.<domain>**	**<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>**

プロビジョニングネットワークなしの追加要件

インストーラーでプロビジョニングされるすべてのインストールには、**baremetal** ネットワークが必要です。**baremetal** ネットワークは、外部への外部ネットワークアクセスに使用されるルーティング可能なネットワークです。OpenShift Container Platform クラスターノードに提供される IP アドレスに加えて、**provisioning** ネットワークがないインストールには以下が必要です。

- **baremetal** ネットワークからの利用可能な IP アドレスを、**install-config.yaml** 設定ファイル内の **bootstrapProvisioningIP** 設定に設定する。

- **baremetal** ネットワークからの利用可能な IP アドレスを、**install-config.yaml** 設定ファイル内の **provisioningHostIP** 設定に設定する。
- RedFish 仮想メディア/iDRAC 仮想メディアを使用した OpenShift Container Platform クラスターのデプロイ



注記

provisioning ネットワークを使用する場合、**bootstrapProvisioningIP** および **provisioningHostIP** に追加の IP アドレスを設定する必要はありません。

帯域外管理 IP アドレスのポートアクセス

帯域外管理 IP アドレスは、ノードとは別のネットワーク上にあります。インストール中に帯域外管理がベアメタルノードと通信できるようにするには、帯域外管理 IP アドレスアドレスに **TCP6180** ポートへのアクセスを許可する必要があります。

1.2.3. ノードの設定

provisioning ネットワークを使用する場合のノードの設定

クラスター内の各ノードには、適切なインストールを行うために以下の設定が必要です。

警告

ノード間で一致していないと、インストールに失敗します。

クラスターノードには 3 つ以上の NIC を追加できますが、インストールプロセスでは最初の 2 つの NIC のみに焦点が当てられます。

NIC	ネットワーク Network	VLAN
NIC1	**provisioning**	<provisioning-vlan>
NIC2	**baremetal**	<baremetal-vlan>

NIC1 は、OpenShift Container Platform クラスターのインストールにのみ使用されるルーティング不可能なネットワーク (**provisioning**) です。

プロビジョナーノードでの Red Hat Enterprise Linux (RHEL) 8.x インストールプロセスは異なる可能性があります。ローカルの Satellite サーバー、PXE サーバー、PXE 対応の NIC2 を使用して Red Hat Enterprise Linux (RHEL) 8.x をインストールするには、以下のようになります。

PXE	ブート順序
NIC1 PXE 対応の **provisioning** ネットワーク	1

NIC2 **baremetal** ネットワークPXE 対応はオプションです。	2

注記

他のすべての NIC で PXE が無効になっていることを確認します。

コントロールプレーンおよびワーカーノードを以下のように設定します。

PXE	ブート順序
NIC1 PXE 対応 (プロビジョニングネットワーク)	1

provisioning ネットワークを使用しないノードの設定

インストールプロセスには、1 つの NIC が必要です。

NIC	ネットワーク Network	VLAN
NICx	**baremetal**	<baremetal-vlan>

NICx は、OpenShift Container Platform クラスターのインストールに使用されるルーティング可能なネットワーク (**baremetal**) であり、インターネットにルーティング可能です。

1.2.4. アウトオブバンド管理 (Out-of-band Management: 帯域外管理)

ノードには通常、ベースボード管理コントローラー (BMC) が使用する追加の NIC があります。これらの BMC は **provisioner** ノードからアクセスできる必要があります。

各ノードは、アウトバウンド管理でアクセスできるようにする必要があります。アウトバウンド管理ネットワークを使用する場合、 **provisioner** ノードには、OpenShift Container Platform 4 の正常なインストールを実行するためにアウトバウンドネットワークへのアクセスが必要になります。

このアウトバウンド管理設定については、本書では扱いません。アウトバウンド管理には、別個の管理ネットワークを設定することを推奨します。ただし、**provisioning** ネットワークまたは **baremetal** ネットワークの使用は有効なオプションになります。

1.2.5. インストールに必要なデータ

OpenShift Container Platform クラスターのインストール前に、すべてのクラスターノードから以下の情報を収集します。

- アウトバウンド管理 IP
  - 例
    - Dell (iDRAC) IP
    - HP (iLO) IP

provisioning ネットワークを使用する場合

- NIC1 (**provisioning**) MAC アドレス
- NIC2 (**baremetal**) MAC アドレス

provisioning ネットワークを省略する場合

- NICx (**baremetal**) MAC アドレス

1.2.6. ノードの検証チェックリスト

provisioning ネットワークを使用する場合

❑ NIC1 VLAN が **provisioning** ネットワークについて設定されている。

❑ NIC2 VLAN が **baremetal** ネットワークについて設定されている。

❑ NIC1 がプロビジョナー、コントロールプレーン (マスター)、およびワーカーノードで PXE 対応として使用できる。

❑ PXE が他のすべての NIC で無効にされている。

❑ コントロールプレーンおよびワーカーノードが設定されている。

❑ すべてのノードがアウトオブバンド管理 (Out-of-band Management: 帯域外管理) でアクセス可能である。

❑ 別個の管理ネットワークが作成されている (オプション)。

❑ インストールに必要なデータ。

provisioning ネットワークを省略する場合

❑ NICx VLAN が **baremetal** ネットワークに設定されている。

❑ コントロールプレーンおよびワーカーノードが設定されている。

❑ すべてのノードがアウトオブバンド管理 (Out-of-band Management: 帯域外管理) でアクセス可能である。

❑ 別個の管理ネットワークが作成されている (オプション)。

❑ インストールに必要なデータ。

1.3. OPENSHIFT インストールの環境のセットアップ

1.3.1. RHEL のプロビジョナーノードへのインストール

ネットワーク設定が完了すると、次の手順では、プロビジョナーノードに RHEL 8.x をインストールします。インストーラーは、OpenShift Container Platform クラスターをインストールする間にプロビジョナーノードをオーケレーターとして使用します。本書の目的上、RHEL のプロビジョナーノードへのインストールは対象外です。ただし、オプションには、RHEL Satellite サーバー、PXE、またはインストールメディアの使用も含まれますが、これらに限定されません。

1.3.2. OpenShift Container Platform インストールのプロビジョナーノードの準備

環境を準備するには、以下の手順を実行します。

手順

1. **ssh** でプロビジョナーノードにログインします。
2. root 以外のユーザー (**kni**) を作成し、そのユーザーに **sudo** 権限を付与します。

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

3. 新規ユーザーの **ssh** キーを作成します。

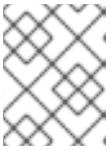
```
# su - kni -c "ssh-keygen -t ed25519 -f /home/kni/.ssh/id_rsa -N ""
```

4. プロビジョナーノードで新規ユーザーとしてログインします。

```
# su - kni
$
```

5. Red Hat Subscription Manager を使用してプロビジョナーノードを登録します。

```
$ 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
```



注記

Red Hat Subscription Manager についての詳細は、Using and Configuring Red Hat Subscription Manager を参照してください。

6. 以下のパッケージをインストールします。

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

7. ユーザーを変更して、新たに作成したユーザーに **libvirt** グループを追加します。

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

8. **firewalld** を再起動して、**http** サービスを有効にします。

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

9. **libvirtd** サービスを開始して、これを有効にします。

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

10. **default** ストレージプールを作成して、これを起動します。

```
$ 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. ネットワークの設定

注記

この手順は、Web コンソールから実行することもできます。

```
$ 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
"
```

注記

このステップの実行後に **ssh** 接続が切断される可能性があります。

IPv6 アドレスには、**baremetal** ネットワーク経由でルーティング可能でない限り、任意のアドレスを使用できます。

IPv6 アドレスを使用する場合に UEFI PXE 設定が有効にされており、UEFI PXE 設定が IPv6 プロトコルに設定されていることを確認します。

12. **provisioning** ネットワーク接続で IPv4 アドレスが設定されている。

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

13. **provisioner** ノードに対して再度 **ssh** を実行します (必要な場合)。

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

14. 接続ブリッジが適切に作成されていることを確認します。

```
$ 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** ファイルを作成します。

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

Web ブラウザーで Install on Bare Metal with user-provisioned infrastructure に移動し、**Downloads** セクションまでスクロールダウンします。**Copy pull secret** をクリックします。**pull-secret.txt** ファイルにコンテンツを貼り付け、そのコンテンツを **kni** ユーザーのホームディレクトリーに保存します。

1.3.3. OpenShift Container Platform インストーラーの取得

インストーラーの **latest-4.x** バージョンを使用して、OpenShift Container Platform の最新の一般公開バージョンをデプロイします。

```
$ 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}')
```

関連情報

- 異なるリリースチャネルの概要については、OpenShift Container Platform アップグレードチャネルおよびリリース について参照してください。

1.3.4. OpenShift Container Platform インストールの展開

インストーラーの取得後、次のステップとしてこれを展開します。

手順

1. 環境変数を設定します。

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

2. **oc** バイナリーを取得します。

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

3. インストーラーを実行します。

```
$ 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 イメージキャッシュの作成 (オプション)

イメージのキャッシュを使用するには、ブートストラップ仮想マシンによって使用される Red Hat Enterprise Linux CoreOS (RHCOS) イメージと、異なるノードをプロビジョニングするためにインストーラーによって使用される RHCOS イメージという 2 つのイメージをダウンロードする必要があります。イメージのキャッシュはオプションですが、帯域幅が制限されたネットワークでインストーラーを実行する場合にとくに役立ちます。

帯域幅が制限されたネットワークでインストーラーを実行し、RHCOS イメージのダウンロードに 15 分から 20 分を超える時間がかかる場合、インストーラーはタイムアウトします。このような場合、Web サーバーでイメージをキャッシュすることができます。

イメージが含まれるコンテナーをインストールするには、以下の手順に従います。

1. **podman** をインストールします。

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

2. RHCOS イメージのキャッシュに使用されるファイアウォールのポート **8080** を開きます。

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

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

3. **bootstraposimage** および **clusterosimage** を保存するディレクトリーを作成します。

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

4. 新規に作成されたディレクトリーに適切な SELinux コンテキストを設定します。

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

5. インストーラーからコミット ID を取得します。ID は、インストーラーがダウンロードする必要のあるイメージを判別します。

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

6. インストーラーがノードにデプロイする RHCOS イメージの URI を取得します。

```
$ 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. インストーラーがブートストラップ仮想マシンにデプロイする RHCOS イメージの URI を取得します。

```
$ 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. イメージが公開されるパスを取得します。

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

9. ブートストラップ仮想マシンにデプロイされる RHCOS イメージの SHA ハッシュを取得します。

```
$ 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. ノードにデプロイされる RHCOS イメージの SHA ハッシュを取得します。

```
$ 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. イメージをダウンロードして、それらを **/home/kni/rhcos_image_cache** ディレクトリーに配置します。

```
$ 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 タイプが、新しく作成されたファイルの **httpd_sys_content_t** であることを確認します。

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

13. Pod を作成します。

```
$ 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. 設定ファイル

1.3.6.1. install-config.yaml ファイルの設定

install-config.yaml ファイルには、追加の詳細情報が必要です。それらの情報のほとんどは、インストーラーおよび結果として作成されるクラスターが利用可能なハードウェアを完全に管理するのに必要な利用可能なハードウェアについての十分な情報として提供されます。

1. **install-config.yaml** を設定します。**pullSecret** および **sshKey** など、環境に合わせて適切な変数を変更します。

```
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 クラスターの一部であるワーカーノードの数に基づいてワーカーマシンをスケーリングします。

2 3 4 5 その他のオプションについては、BMC アドレス指定のセクションを参照してください。

2. クラスター設定を保存するディレクトリーを作成します。

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

3. OpenShift Container Platform クラスターをインストールする前に、すべてのベアメタルノードの電源がオフになっていることを確認します。

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

4. 以前に試行したデプロイメントにより古いブートストラップリソースが残っている場合は、これを削除します。

```
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 ファイル内でのプロキシー設定 (オプション)

プロキシーを使用して OpenShift Container Platform クラスターをデプロイするには、 **install-config.yaml** ファイルに以下の変更を加えます。

```
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>
```

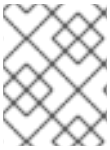
以下は、値を含む **noProxy** の例です。

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

プロキシーを有効な状態にして、対応するキー/値のペアでプロキシーの適切な値を設定します。

主な留意事項:

- プロキシーに HTTPS プロキシーがない場合、 **httpsProxy** の値を **https://** から **http://** に変更します。
- provisioning ネットワークを使用する場合、これを **noProxy** 設定に含めます。そうしない場合、インストーラーは失敗します。
- プロビジョナーノード内の環境変数としてすべてのプロキシー設定を設定します。たとえば、**HTTP_PROXY**、**HTTPS_PROXY**、および **NO_PROXY** が含まれます。



注記

IPv6 でプロビジョニングする場合、**noProxy** 設定で CIDR アドレスブロックを定義することはできません。各アドレスを個別に定義する必要があります。

1.3.6.3. provisioning ネットワークがない install-config.yaml ファイルの変更 (オプション)

provisioning ネットワークなしに OpenShift Container Platformmake をデプロイするには、 **install-config.yaml** ファイルに以下の変更を加えます。

```
platform:
  baremetal:
    apiVIP: <apiVIP>
    ingressVIP: <ingress/wildcard VIP>
    provisioningNetwork: "Disabled"
    provisioningHostIP: <baremetal_network_IP1>
    bootstrapProvisioningIP: <baremetal_network_IP2>
```

注記

provisioningHostIP および **bootstrapProvisioningIP** 設定の **baremetal** ネットワークからの 2 つの IP アドレスを指定し、 **provisioningBridge** および **provisioningNetworkCIDR** 設定を削除する必要があります。

1.3.6.4. 追加の install-config パラメーター

install-config.yaml ファイルに必要なパラメーター **hosts** パラメーターおよび **bmc** パラメーターについては、以下の表を参照してください。

表1.1 必須パラメーター

パラメーター	デフォルト	説明
baseDomain		クラスターのドメイン名。例: **example.com**
sshKey		**sshKey** 設定には、コントロールプレーンノードおよびワーカーノードにアクセスするために必要な **~/.ssh/id_rsa.pub** ファイルのキーが含まれます。通常、このキーは **provisionaer** ノードのキーです。
pullSecret		**pullSecret** 設定には、プロビジョナーノードの準備の際に Install OpenShift on Bare Metal ページからダウンロードしたプルシークレットのコピーが含まれます。
metadata: name:		OpenShift Container Platform クラスターに指定される名前。例: **openshift**
networking: machineCIDR:		外部ネットワークの公開 CIDR (Classless Inter-Domain Routing)。例: **10.0.0.0/24**
compute: - name: worker		OpenShift Container Platform クラスターでは、ノードがゼロであってもワーカー (またはコンピュート) ノードの名前を指定する必要があります。
compute: replicas: 2		レプリカは、OpenShift Container Platform クラスターのワーカー (またはコンピュート) ノードの数を設定します。
controlPlane: name: master		OpenShift Container Platform クラスターには、コントロールプレーン (マスター) ノードの名前が必要です。

パラメーター	デフォルト	説明
controlPlane: replicas: 3		レプリカは、OpenShift Container Platform クラスターの一部として含まれるコントロールプレーン (マスター) ノードの数を設定します。
provisioningNetworkInterface		プロビジョニングネットワークに接続されたコントロールプレーンノード上のネットワークインターフェイス名。
defaultMachinePlatform		プラットフォーム設定なしでマシンプールに使用されるデフォルト設定。
apiVIP	**api.<clustername.clusterdomain>**	内部 API 通信に使用する VIP。 この設定は、デフォルト名が正しく解決されるように DNS で指定するか、または事前に設定する必要があります。
disableCertificateVerification	**False**	**redfish** および **redfish-virtualmedia** では、このパラメーターで BMC アドレスを管理する必要があります。BMC アドレスに自己署名証明書を使用する場合は、値が **True** である必要があります。
ingressVIP	**test.apps.<clustername.clusterdomain>**	Ingress トラフィックに使用する VIP。

表1.2 オプションのパラメーター

パラメーター	デフォルト	詳細
provisioningDHCPRange	**172.22.0.10,172.22.0.100**	**provisioning** ネットワークでノードの IP 範囲を定義します。
provisioningNetworkCIDR	**172.22.0.0/24**	プロビジョニングに使用するネットワークの CIDR。このオプションは、**provisioning** ネットワークでデフォルトのアドレス範囲を使用しない場合に必要です。
clusterProvisioningIP	**provisioningNetworkCIDR** の 3 番目の IP アドレス。	プロビジョニングサービスが実行されるクラスター内の IP アドレス。デフォルトは、**provisioning** サブネットの 3 番目の IP アドレスに設定されます。例: **172.22.0.3**.

パラメーター	デフォルト	詳細
bootstrapProvisioningIP	**provisioningNetworkCIDR** の 2 番目の IP アドレス	インストーラーがコントロールプレーン (マスター) ノードをデプロイしている間にプロビジョニングサービスが実行されるブートストラップ仮想マシンの IP。デフォルトは、**provisioning** サブネットの 2 番目の IP に設定されます。例: **172.22.0.2** **provisioning** ネットワークを使用しない場合、この値を、**baremetal** ネットワークで利用可能な IP アドレスに設定します。
externalBridge	**baremetal**	**baremetal** ネットワークに接続されているハイパーバイザーの **baremetal** ブリッジの名前。
provisioningBridge	**provisioning**	**provisioning** ネットワークに接続されている **provisioner** ホストの **provisioning** ブリッジの名前。
defaultMachinePlatform		プラットフォーム設定なしでマシンプールに使用されるデフォルト設定。
bootstrapOSImage		ブートストラップノードのデフォルトのオペレーティングシステムイメージを上書きするための URL。URL にはイメージの SHA-256 ハッシュが含まれている必要があります。例: **https://mirror.openshift.com/rhcos-<version>-qemu.qcow2.gz?sha256=<uncompressed_sha256>**;
clusterOSImage		クラスターノードのデフォルトオペレーティングシステムを上書きするための URL。URL には、イメージの SHA-256 ハッシュを含める必要があります。例: **https://mirror.openshift.com/images/rhcos-<version>-openstack.qcow2.gz?sha256=<compressed_sha256>**;
provisioningNetwork		**provisioning** ネットワークの要件を無効にするには、このパラメーターを **Disabled** に設定します。ユーザーは仮想メディアベースのプロビジョニングのみを実行するか、またはアシストインストールを使用してクラスターを起動することができます。電源管理を使用する場合は、マシンネットワークから BMC にアクセスできる必要があります。ユーザーは、プロビジョニングサービスに使用する外部ネットワークに 2 つの IP アドレスを指定する必要があります。DHCP、TFTP などを含むプロビジョニングネットワークを完全に管理するには、このパラメーターを **managed** (デフォルト) に設定します。 このパラメーターを **unmanaged** に設定してプロビジョニングネットワークを引き続き有効にしますが、DHCP の手動設定を行います。仮想メディアのプロビジョニングが推奨されますが、必要に応じて PXE を引き続き利用できます。
provisioningHostingIp		**provisioningNetwork** 設定が **Disabled** に設定されている場合、このパラメーターを **baremetal** ネットワークで利用可能な IP アドレスに設定します。

パラメーター	デフォルト	詳細
httpProxy		このパラメーターを、環境内で使用する適切な HTTP プロキシーに設定します。
httpsProxy		このパラメーターを、環境内で使用する適切な HTTPS プロキシーに設定します。
noProxy		このパラメーターを、環境内のプロキシーの使用に対する例外の一覧に設定します。

ホスト

hosts パラメーターは、クラスターのビルドに使用される個別のベアメタルアセットの一覧です。

名前	デフォルト	詳細
name		詳細情報に関連付ける **BareMetalHost** リソースの名前。例: **openshift-master-0**
role		ベアメタルノードのロール。**master** または **worker** のいずれか。
bmc		ベースボード管理コントローラーの接続詳細。詳細は、BMC アドレスのセクションを参照してください。
bootMACAddress		ホストが **provisioning** ネットワークを起動するために使用する NIC の MAC アドレス。

1.3.6.5. BMC アドレス

それぞれの **bmc** エントリーの **address** フィールドは、URL スキーム内のコントローラーのタイプやネットワーク上のその場所を含む、OpenShift Container Platform クラスターノードに接続する URL です。

IPMI

IPMI ホストは **ipmi://<out-of-band-ip>:<port>** を使用し、指定されていない場合はポート **623** にデフォルトで設定されます。以下の例は、**install-config.yaml** ファイル内の IPMI 設定を示しています。

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

```
      address: ipmi://<out-of-band-ip>
      username: <user>
      password: <password>
```

HPE 用の Redfish

RedFish を有効にするには、**redfish://** または **redfish+http://** を使用して TLS を無効にします。インストーラーには、ホスト名または IP アドレスとシステム ID へのパスの両方が必要です。以下の例は、**install-config.yaml** ファイル内の RedFish 設定を示しています。

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

アウトバウンド管理のアドレスについて認証局の証明書を使用することが推奨されますが、自己署名証明書を使用する場合には、**bmc** 設定に **disableCertificateVerification: True** を含める必要があります。以下の例は、**install-config.yaml** ファイル内の **disableCertificateVerification: True** 設定パラメーターを使用する RedFish 設定を示しています。

```
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 用の Redfish

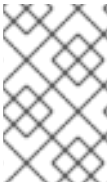
RedFish を有効にするには、**redfish://** または **redfish+http://** を使用して TLS を無効にします。インストーラーには、ホスト名または IP アドレスとシステム ID へのパスの両方が必要です。以下の例は、**install-config.yaml** ファイル内の RedFish 設定を示しています。

```
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>
```

アウトバウンド管理のアドレスについて認証局の証明書を使用することが推奨されますが、自己署名証明書を使用する場合には、**bmc** 設定に **disableCertificateVerification: True** を含める必要があります。以下の例は、**install-config.yaml** ファイル内の **disableCertificateVerification: True** 設定パラメー

ターを使用する RedFish 設定を示しています。

```
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
```



注記

現時点で RedFish は、ベアメタルデプロイメントにおける OpenShift Container Platform のインストーラーでプロビジョニングされるインストール向けに iDRAC ファームウェアのバージョン **4.20.20.20** 以上を搭載の Dell でのみサポートされます。

HPE 用の Redfish Virtual Media

HPE サーバーについて RedFish Virtual Media を有効にするには、**address** 設定で **redfish-virtualmedia://** を使用します。以下の例は、**install-config.yaml** ファイル内で RedFish Virtual Media を使用する方法を示しています。

```
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 用の Redfish Virtual Media

Dell サーバーの RedFish Virtual Media については、 **address** 設定で **idrac-virtualmedia://** を使用します。

注記

Dell サーバー上の Redfish Virtual Media には、OpenShift Container Platform 4.6 の既知の問題があります。4.6.1 ポイントリリースがこの問題を解決します。

以下の例は、**install-config.yaml** ファイル内で iDRAC 仮想メディアを使用する方法を示しています。

```
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>
```

注記

idrac-virtualmedia には iDRAC ファームウェアのバージョン 4.20.20.20 以降が必要です。

OpenShift Container Platform クラスターノードについて、iDRAC コンソールで AutoAttach が有効にされていることを確認します。メニューパスは以下のようになります。**Configuration→Virtual Media→Attach Mode→AutoAttach**

1.3.6.6. ルートデバイスのヒント

rootDeviceHints パラメーターは、インストーラーが Red Hat Enterprise Linux CoreOS (RHCOS) イメージを特定のデバイスにプロビジョニングできるようにします。インストーラーは、検出順にデバイスを検査し、検出された値をヒントの値と比較します。インストーラーは、ヒント値に一致する最初に検出されたデバイスを使用します。この設定は複数のヒントを組み合わせることができますが、デバイスは、インストーラーがこれを選択できるようにすべてのヒントに一致する必要があります。

表1.3 サブフィールド

サブフィールド	説明
deviceName	**/dev/vda** などの Linux デバイス名を含む文字列。ヒントは、実際の値と完全に一致する必要があります。
hctl	**0:0:0:0** などの SCSI バスアドレスを含む文字列。ヒントは、実際の値と完全に一致する必要があります。
model	ベンダー固有のデバイス識別子を含む文字列。ヒントは、実際の値のサブ文字列になります。
vendor	デバイスのベンダーまたは製造元の名前が含まれる文字列。ヒントは、実際の値のサブ文字列になります。
serialNumber	デバイスのシリアル番号を含む文字列。ヒントは、実際の値と完全に一致する必要があります。
minSizeGigabytes	デバイスの最小サイズ (ギガバイト単位) を表す整数。
wwn	一意のストレージ ID を含む文字列。ヒントは、実際の値と完全に一致する必要があります。
wwnWithExtension	ベンダー拡張が追加された一意のストレージ ID を含む文字列。ヒントは、実際の値と完全に一致する必要があります。

サブフィールド	説明
wwnVendorExtension	一意のベンダーストレージ ID を含む文字列。ヒントは、実際の値と完全に一致する必要があります。
rotational	デバイスがローテーションするディスクである (true) か、そうでないか (false) を示すブール値。

使用例

```
 - 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 マニフェストの作成

1. OpenShift Container Platform マニフェストを作成します。

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

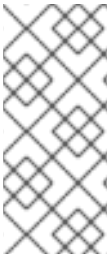
```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for
Scheduler cluster settings
WARNING Discarding the OpenShift Manifest that was provided in the target directory
because its dependencies are dirty and it needs to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

1.3.7. 非接続レジストリーの作成 (オプション)

インストールレジストリーのローカルコピーを使用して OpenShift KNI クラスターをインストールする必要がある場合があります。これにより、クラスターノードがインターネットにアクセスできないネットワーク上にあるため、ネットワークの効率が高まる可能性があります。

レジストリーのローカルまたはミラーリングされたコピーには、以下が必要です。

- レジストリーの証明書。これには、自己署名証明書を使用できます。
- Web サーバー: これは、システム上のコンテナーによって提供されます。
- 証明書およびローカルリポジトリーの情報が含まれる更新されたプルシークレット。



注記

レジストリーノードでの非接続レジストリーの作成はオプションです。以下のセクションでは、それらの手順はレジストリーノードで非接続レジストリーを作成する場合にのみ実行する必要があるためにオプションであることを示しています。レジストリーノードで非接続レジストリーを作成する際に、(optional) というラベルが付けられたすべての後続のサブセクションを実行する必要があります。

1.3.7.1. ミラーリングされたレジストリーをホストするためのレジストリーノードの準備 (オプション)

レジストリーノードに以下の変更を加えます。

手順

1. レジストリーノードでファイアウォールポートを開きます。

```
$ 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. レジストリーノードに必要なパッケージをインストールします。

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

3. リポジトリー情報が保持されるディレクトリー構造を作成します。

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

1.3.7.2. 自己署名証明書の生成 (オプション)

レジストリーノードの自己署名証明書を生成し、これを **/opt/registry/certs** ディレクトリーに配置します。

手順

1. 必要に応じて証明書情報を調整します。

```
$ 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}"
```

注記

<Country Name> を置き換える場合には、2 文字のみを使用します。例: **US**

2. レジストリーノードの **ca-trust** を新規証明書で更新します。

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

1.3.7.3. レジストリー podman コンテナーの作成 (オプション)

レジストリーコンテナーは、証明書、認証ファイルの **/opt/registry** ディレクトリーを使用し、そのデータファイルを保存するためにこれを使用します。

レジストリーコンテナーは **httpd** を使用し、認証に **htpasswd** ファイルが必要になります。

手順

1. コンテナーが使用する **htpasswd** ファイルを **/opt/registry/auth** に作成します。

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

<user> をユーザー名に、**<passwd>** をパスワードに置き換えます。

2. レジストリーコンテナーを作成し、起動します。

```
$ 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. pull-secret のコピーおよび更新 (オプション)

プロビジョナーノードからレジストリーノードにプルシークレットファイルをコピーし、これを新規レジストリーノードの認証情報を含めるように変更します。

手順

1. **pull-secret.txt** ファイルをコピーします。

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

2. **host_fqdn** 環境変数をレジストリーノードの完全修飾ドメイン名で更新します。

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

3. **htpasswd** ファイルの作成に使用される **http** 認証情報の base64 エンコーディングで **b64auth** 環境変数を更新します。

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

<username> をユーザー名に、**<passwd>** をパスワードに置き換えます。

4. **base64** 承認文字列を使用するように **AUTHSTRING** 環境変数を設定します。**$USER** 変数は、現行ユーザーの名前が含まれる環境変数です。

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

5. **pull-secret.txt** ファイルを更新します。

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

1.3.7.5. リポジトリーのミラーリング (オプション)

手順

1. **oc** バイナリーをプロビジョナーノードからレジストリーノードにコピーします。

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

2. 必要な環境変数を設定します。
   a. リリースバージョンを設定します。

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

   <release_version> には、インストールする OpenShift Container Platform のバージョンに対応するタグ (**4.6** など) を指定します。

   b. ローカルレジストリー名とホストポートを設定します。

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

   <local_registry_host_name> については、ミラーレジストリーのレジストリードメイン名を指定し、**<local_registry_host_port>** については、コンテンツの送信に使用するポートを指定します。

   c. ローカルリポジトリー名を設定します。

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

<local_repository_name> については、**ocp4/openshift4** などのレジストリーに作成するリポジトリーの名前を指定します。

3. リモートインストールイメージをローカルリポジトリーにミラーリングします。

```
$ /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. install-config.yaml ファイルを、非接続レジストリーを使用するように変更します (オプション)。

プロビジョナーノードでは、**install-config.yaml** ファイルは **pull-secret-update.txt** ファイルから新たに作成された pull-secret を使用する必要があります。**install-config.yaml** ファイルには、非接続レジストリーノードの証明書およびレジストリー情報も含まれる必要があります。

手順

1. 非接続レジストリーノードの証明書を **install-config.yaml** ファイルに追加します。証明書は **"additionalTrustBundle: |"** 行に従い、通常は 2 つのスペースで適切にインデントされる必要があります。

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

2. レジストリーのミラー情報を **install-config.yaml** ファイルに追加します。

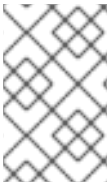
```
$ echo "imageContentSources:" >> install-config.yaml
$ echo "- mirrors:" >> install-config.yaml
$ echo "  - registry.example.com:5000/ocp4/openshift4" >> install-config.yaml
$ echo "  source: quay.io/openshift-release-dev/ocp-release" >> install-config.yaml
$ echo "- mirrors:" >> install-config.yaml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

注記

registry.example.com をレジストリーの完全修飾ドメイン名に置き換えます。

1.3.8. ワーカーノードでのルーターのデプロイ

インストール時に、インストーラーはルーター Pod をワーカーノードにデプロイします。デフォルトで、インストーラーは 2 つのルーター Pod をインストールします。初期クラスターにワーカーノードが 1 つしかない場合や、デプロイされたクラスターで、OpenShift Container Platform クラスター内のサービスに対して予定される外部トラフィックを処理する追加のルーターが必要な場合、**yaml** ファイルを作成して適切なルーターレプリカ数を設定できます。



注記

デフォルトで、インストーラーは 2 つのルーターをデプロイします。クラスターにワーカーノードが 2 つ以上ある場合、このセクションを省略できます。詳細は、Ingress Operator in OpenShift Container Platform を参照してください。

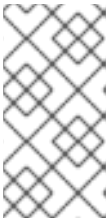
注記

クラスターにワーカーノードがない場合、インストーラーはデフォルトで 2 つのルーターをコントロールプレーンノードにデプロイします。クラスターにワーカーノードがない場合、このセクションを省略できます。

手順

1. **router-replicas.yaml** ファイルを作成します。

```
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: ""
```



注記

<num-of-router-pods> を適切な値に置き換えます。1 つのワーカーノードのみを使用している場合、**replicas:** を **1** に設定します。4 つ以上のワーカーノードを使用している場合、**replicas:** のデフォルトの値 **2** を随時増やすことができます。

2. **router-replicas.yaml** ファイルを保存し、これを **clusterconfigs/openshift** ディレクトリーにコピーします。

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

1.3.9. インストールの検証チェックリスト

❑ OpenShift Container Platform インストーラーが取得されている。

❑ OpenShift Container Platform インストーラーが展開されている。

❑ **install-config.yaml** の必須パラメーターが設定されている。

❑ **install-config.yaml** の **hosts** パラメーターが設定されている。

❑ **install-config.yaml** の **bmc** パラメーターが設定されている。

❑ **bmc address** フィールドで設定されている値の変換が適用されている。

❑ 非接続レジストリーを作成している (オプション)。

❑ (オプション) 非接続レジストリー設定が使用されている場合にこれを検証する。

❑ (オプション) ルーターをワーカーノードにデプロイしている。

1.3.10. OpenShift Container Platform インストーラーを使用したクラスターのデプロイ

OpenShift Container Platform インストーラーを実行します。

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

1.3.11. インストール後

デプロイメントプロセスで、**tail** コマンドを install ディレクトリーフォルダーの **.openshift_install.log** ログファイルに対して実行して、インストールの全体のステータスを確認できます。

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

1.3.12. ベアメタルにクラスターを再インストールする準備

ベアメタルにクラスターを再インストールする前に、クリーンアップ操作を実行する必要があります。

手順

1. ブートストラップ、コントロールプレーンノード (マスターとも呼ばれる)、およびワーカーノードのディスクを削除または再フォーマットします。ハイパーバイザー環境で作業している場合は、削除したディスクを追加する必要があります。
2. 以前のインストールで生成されたアーティファクトを削除します。

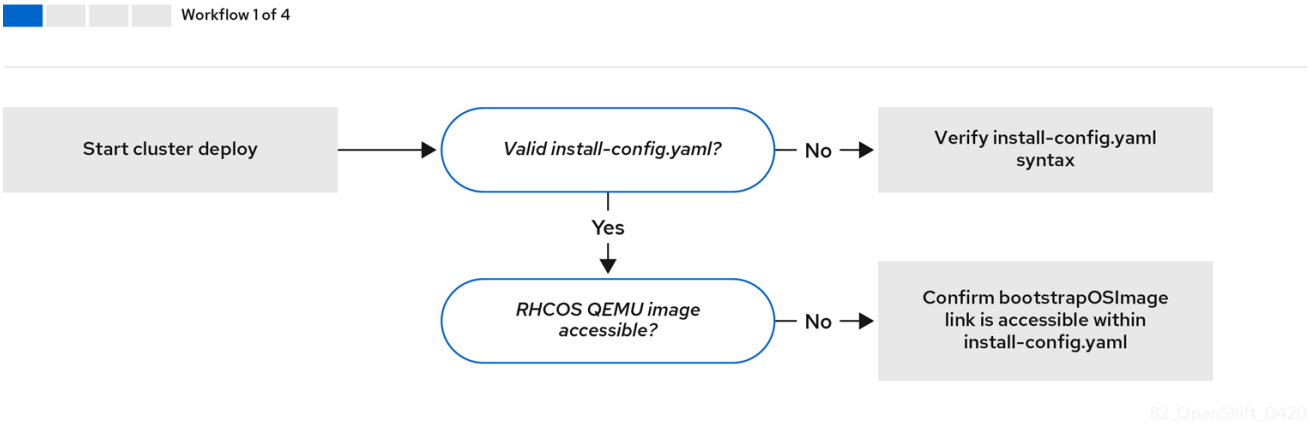
```
$ cd ; /bin/rm -rf auth/ bootstrap.ign master.ign worker.ign metadata.json \
.openshift_install.log .openshift_install_state.json
```

3. 新しいマニフェストと Ignition 設定ファイルを生成します。詳細は、Kubernetes マニフェストおよび Ignition 設定ファイルの作成を参照してください。
4. インストールプログラムが作成した新規ブートストラップ、コントロールプレーン、およびコンピュートノード Ignition 設定ファイルを HTTP サーバーにアップロードします。これにより、以前の Ignition ファイルが上書きされます。

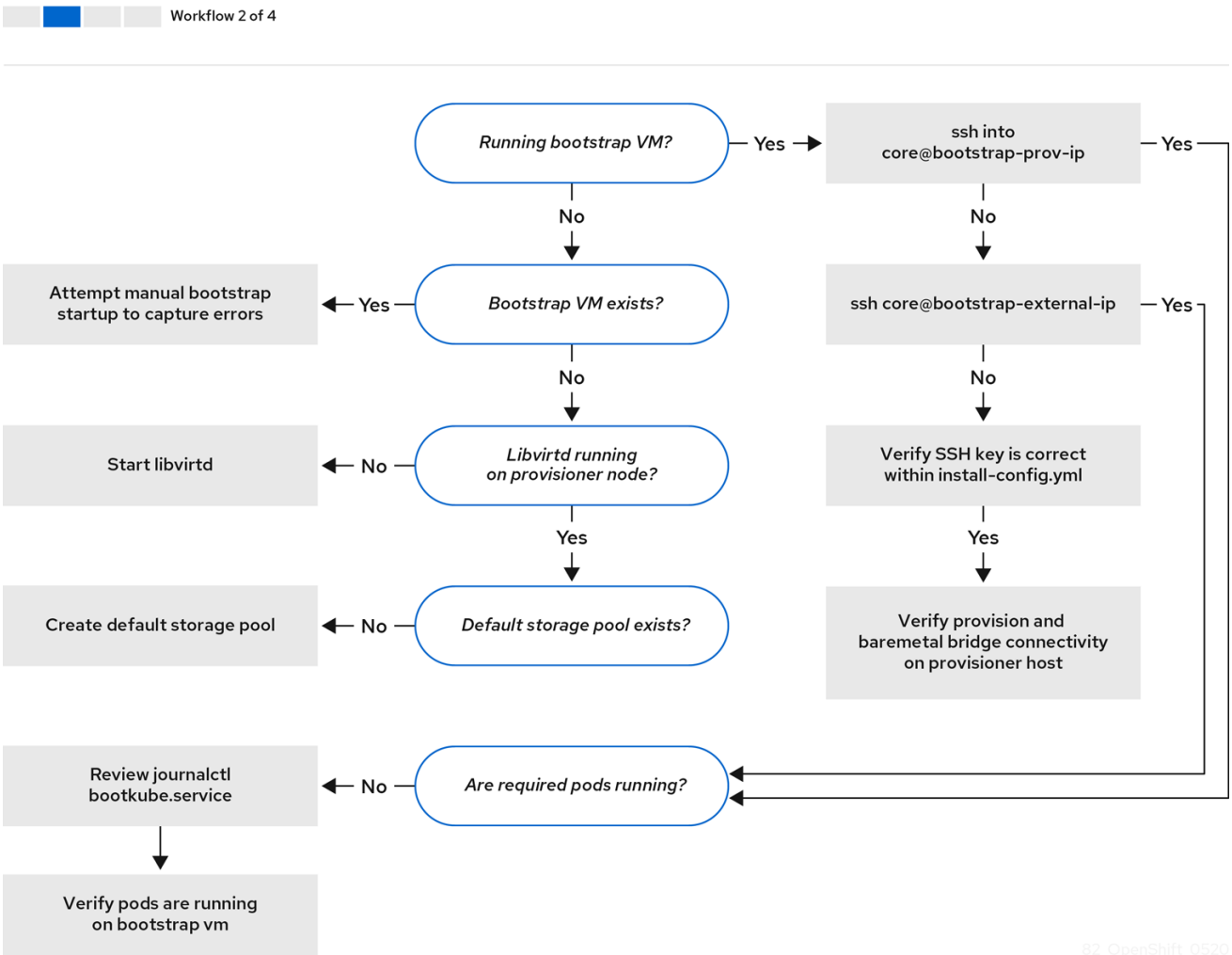
1.4. トラブルシューティング

1.4.1. インストーラーワークフローのトラブルシューティング

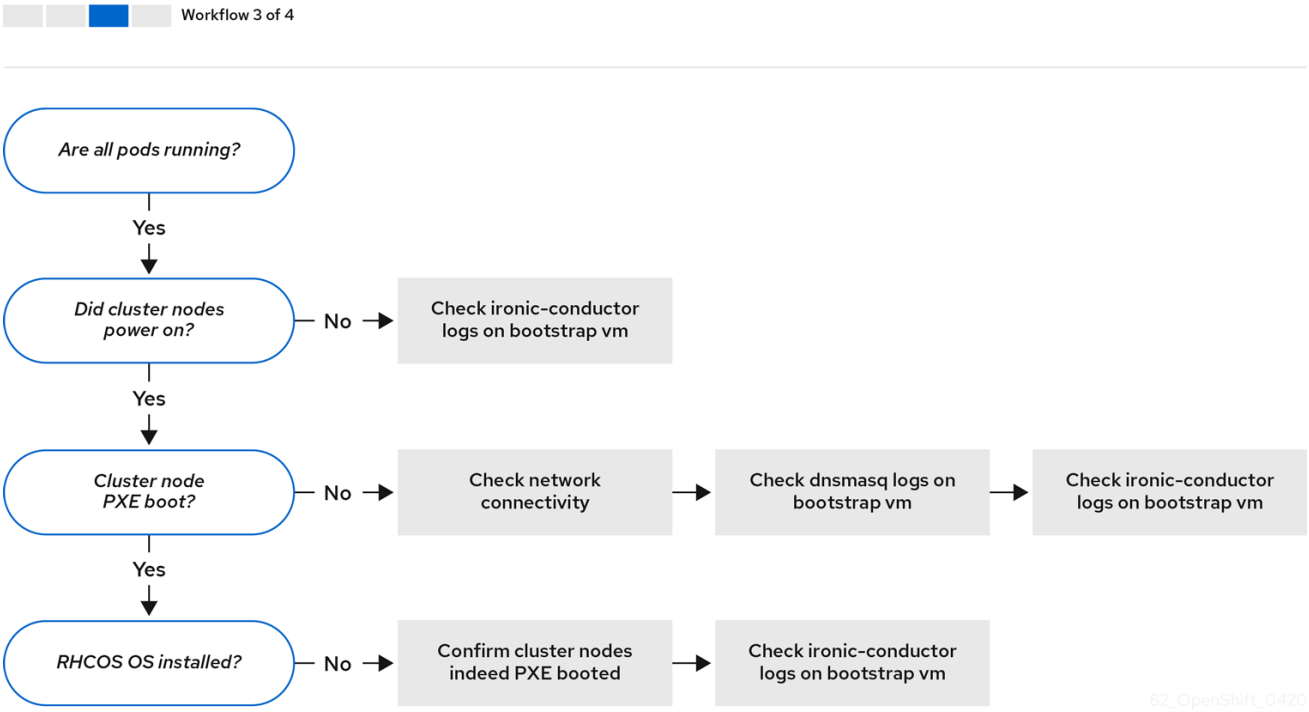
インストール環境のトラブルシューティングを行う前に、ベアメタルへのインストーラーでプロビジョニングされるインストールの全体的なフローを理解することは重要です。以下の図は、環境におけるステップごとのトラブルシューティングフローを示しています。



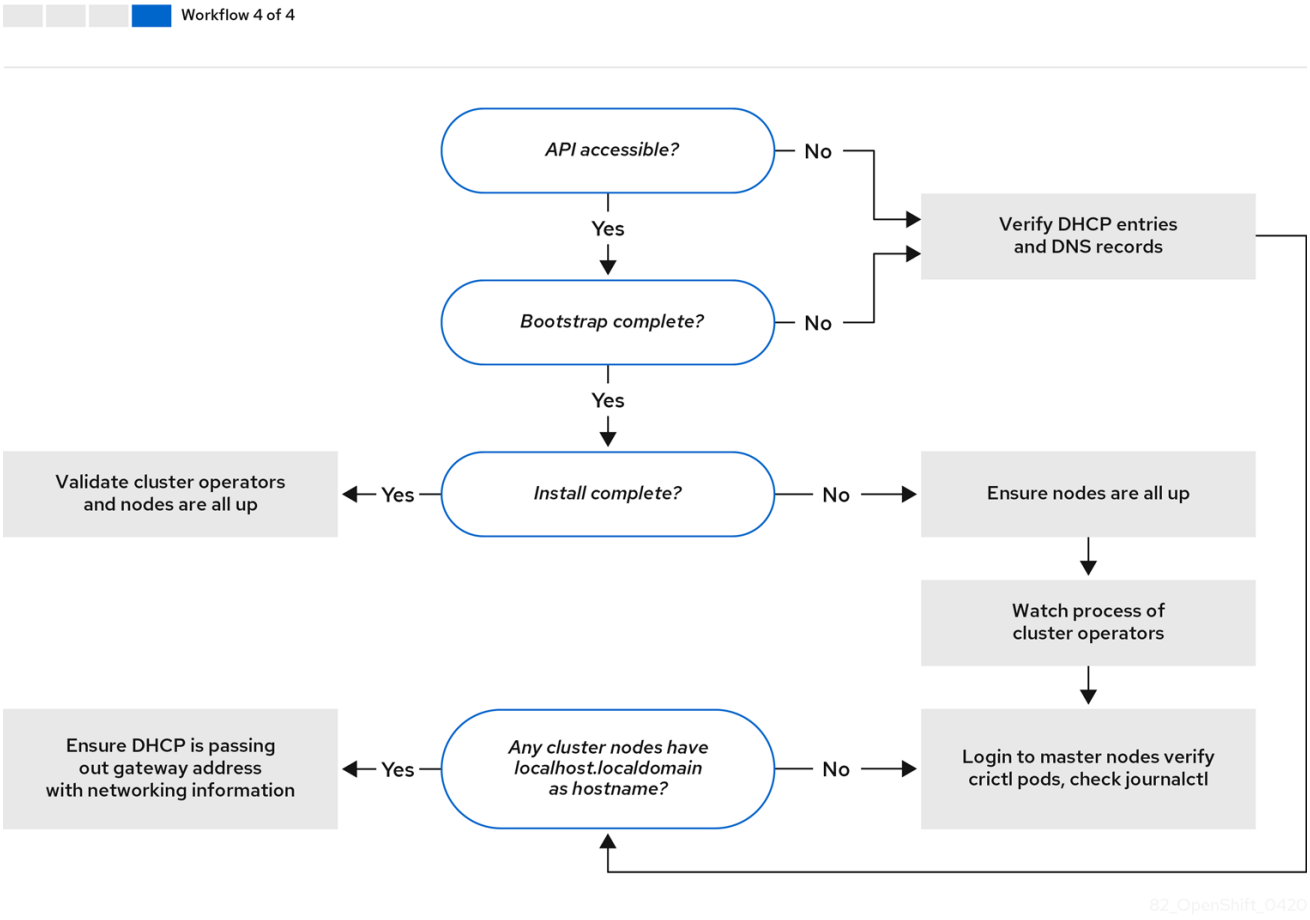
ワークフロー 1/4 は、**install-config.yaml** ファイルにエラーがある場合や Red Hat Enterprise Linux CoreOS (RHCOS) イメージにアクセスできない場合のトラブルシューティングのワークフローを説明しています。トラブルシューティングについての提案は、Troubleshooting **install-config.yaml** を参照してください。



ワークフロー 2/4 は、ブートストラップ仮想マシンの問題、クラスターノードを起動できないブートストラップ仮想マシン、および ログの検査 についてのトラブルシューティングのワークフローを説明しています。**provisioning** ネットワークなしに OpenShift Container Platform クラスターをインストールする場合は、このワークフローは適用されません。



ワークフロー 3/4 は、PXE ブートしないクラスターノード のトラブルシューティングのワークフローを説明しています。



ワークフロー 4/4 は、アクセスできない API から 検証済みのインストール までのトラブルシューティングのワークフローを説明します。

1.4.2. install-config.yaml のトラブルシューティング

install-config.yaml 設定ファイルは、OpenShift Container Platform クラスターの一部であるすべてのノードを表します。このファイルには、**apiVersion**、**baseDomain**、**imageContentSources**、および仮想 IP アドレスのみで設定されるがこれらに制限されない必要なオプションが含まれます。OpenShift Container Platform クラスターのデプロイメントの初期段階でエラーが発生した場合、エラーは **install-config.yaml** 設定ファイルにある可能性があります。

手順

1. YAML-tips のガイドラインを使用します。
2. syntax-check を使用して YAML 構文が正しいことを確認します。
3. Red Hat Enterprise Linux CoreOS (RHCOS) QEMU イメージが適切に定義され、**install-config.yaml** で提供される URL 経由でアクセスできることを確認します。以下に例を示します。

```
$ 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
```

出力が **200** の場合、ブートストラップ仮想マシンイメージを保存する Web サーバーからの有効な応答があります。

1.4.3. ブートストラップ仮想マシンの問題

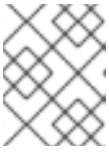
OpenShift Container Platform インストーラーは、OpenShift Container Platform クラスターノードのプロビジョニングを処理するブートストラップノードの仮想マシンを起動します。

手順

1. インストーラーをトリガー後の約 10 分から 15 分後に、**virsh** コマンドを使用してブートストラップ仮想マシンが機能していることを確認します。

```
$ sudo virsh list
```

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



注記

ブートストラップ仮想マシンの名前は常にクラスター名で始まり、その後にランダムな文字セットが続き、bootstrap という単語で終わります。

ブートストラップ仮想マシンが 10 - 15 分後に実行されていない場合は、実行されない理由についてトラブルシューティングします。発生する可能性のある問題には以下が含まれます。

2. **libvirtd** がシステムで実行されていることを確認します。

```
$ 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
```

ブートストラップ仮想マシンが動作している場合は、これにログインします。

3. **virsh console** コマンドを使用して、ブートストラップ仮想マシンの IP アドレスを見つけます。

```
$ 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:
```

重要

provisioning ネットワークなしで OpenShift Container Platform クラスターをデプロイする場合、**172.22.0.2** などのプライベート IP アドレスではなく、パブリック IP アドレスを使用する必要があります。

4. IP アドレスを取得したら、**ssh** コマンドを使用してブートストラップ仮想マシンにログインします。

注記

直前の手順のコンソール出力では、**ens3** で提供される IPv6 IP アドレスまたは **ens4** で提供される IPv4 IP を使用できます。

```
$ ssh core@172.22.0.2
```

ブートストラップ仮想マシンへのログインに成功しない場合は、以下いずれかのシナリオが発生した可能性があります。

- **172.22.0.0/24** ネットワークにアクセスできない。プロビジョナーホスト、とくに **provisioning** ネットワークブリッジに関連するネットワークの接続を確認します。**provisioning** ネットワークを使用していない場合は、この問題はありません。

- パブリックネットワーク経由でブートストラップ仮想マシンにアクセスできない。**baremetal** ネットワークで SSH を試行する際に、**provisioner** ホストの、とくに **baremetal** ネットワークブリッジについて接続を確認します。

- **Permission denied (publickey,password,keyboard-interactive)** が出される。ブートストラップ仮想マシンへのアクセスを試行すると、**Permission denied** エラーが発生する可能性があります。仮想マシンへのログインを試行するユーザーの SSH キーが **install-config.yaml** ファイル内で設定されていることを確認します。

1.4.3.1. ブートストラップ仮想マシンがクラスターノードを起動できない

デプロイメント時に、ブートストラップ仮想マシンがクラスターノードの起動に失敗する可能性があり、これにより、仮想マシンがノードに RHCOS イメージをプロビジョニングできなくなります。このシナリオは、以下の原因で発生する可能性があります。

- **install-config.yaml** ファイルに関連する問題。

- ベアメタルネットワーク経由のアウトオブバンド (out-of-band) ネットワークアクセスに関する問題

この問題を確認するには、**ironic** に関連する 3 つのコンテナーを使用できます。

- **ironic-api**

- **ironic-conductor**

- **ironic-inspector**

手順

1. ブートストラップ仮想マシンにログインします。

```
$ ssh core@172.22.0.2
```

2. コンテナーログを確認するには、以下を実行します。

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

<container-name> を、**ironic-api**、**ironic-conductor**、または **ironic-inspector** のいずれかに置き換えます。コントロールプレーンノードが PXE 経由で起動しない問題が発生した場合には、**ironic-conductor** Pod を確認してください。**ironic-conductor** Pod には、IPMI 経由でノードへのログインを試みるため、クラスターノードのブートの試行についての最も詳細な情報が含まれます。

考えられる理由

クラスターノードは、デプロイメントの開始時に **ON** 状態にある可能性があります。

解決策

IPMI でのインストールを開始する前に、OpenShift Container Platform クラスターノードの電源をオフにします。

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

1.4.3.2. ログの検査

RHCOS イメージのダウンロードまたはアクセスに問題が発生した場合には、最初に **install-config.yaml** 設定ファイルで URL が正しいことを確認します。

RHCOS イメージをホストする内部 Web サーバーの例

```
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 および **coreos-downloader** コンテナーは、**install-config.yaml** 設定ファイルで指定されている Web サーバーまたは外部の quay.io レジストリーからリソースをダウンロードします。以下の 2 つのコンテナーが稼働していることを確認し、必要に応じてログを検査します。

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

手順

1. ブートストラップ仮想マシンにログインします。

   ```
   $ ssh core@172.22.0.2
   ```

2. ブートストラップ仮想マシン内の **ipa-downloader** および **coreos-downloader** コンテナーのステータスを確認します。

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

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

   ブートストラップ仮想マシンがイメージへの URL にアクセスできない場合、**curl** コマンドを使用して、仮想マシンがイメージにアクセスできることを確認します。

3. すべてのコンテナーがデプロイメントフェーズで起動されているかどうかを示す **bootkube** ログを検査するには、以下を実行します。

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

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

4. **dnsmasq**、**mariadb**、**httpd**、および **ironic** を含むすべての Pod が実行中であることを確認します。

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

5. Pod に問題がある場合には、問題のあるコンテナーのログを確認します。**ironic-api** のログを確認するには、以下を実行します。

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

1.4.4. クラスターノードが PXE ブートしない

OpenShift Container Platform クラスターノードが PXE ブートしない場合、PXE ブートしないクラスターノードで以下のチェックを実行します。この手順は、**provisioning** ネットワークなしで OpenShift Container Platform クラスターをインストールする場合には適用されません。

手順

1. **provisioning** ネットワークへのネットワークの接続を確認します。
2. PXE が **provisioning** ネットワークの NIC で有効にされており、PXE がその他のすべての NIC について無効にされていることを確認します。
3. **install-config.yaml** 設定ファイルに、適切なハードウェアプロファイルと **provisioning** ネットワークに接続された NIC のブート MAC アドレスが含まれることを確認します。以下に例を示します。

コントロールプレーンノードの設定

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

ワーカーノード設定

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

1.4.5. API にアクセスできない

クラスターが実行されており、クライアントが API にアクセスできない場合、ドメイン名の解決の問題により API へのアクセスが妨げられる可能性があります。

手順

1. **Hostname Resolution:** クラスターノードに **localhost.localdomain** だけでなく、完全修飾ドメイン名があることを確認します。以下に例を示します。

```
$ hostname
```

ホスト名が設定されていない場合、正しいホスト名を設定します。以下に例を示します。

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

2. **正しくない名前の解決:** 各ノードに **dig** および **nslookup** を使用して DNS サーバーに正しい名前の解決があることを確認します。以下に例を示します。

```
$ 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
```

前述の例の出力は、**api.<cluster-name>.example.com** VIP の適切な IP アドレスが **10.19.13.86** であることを示しています。この IP アドレスは **baremetal** 上にある必要があります。

1.4.6. 以前のインストールのクリーンアップ

以前のデプロイメントに失敗した場合、OpenShift Container Platform のデプロイを再試行する前に、失敗した試行からアーティファクトを削除します。

手順

1. OpenShift Container Platform クラスターをインストールする前に、すべてのベアメタルノードの電源をオフにします。

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

2. 以前に試行したデプロイメントにより古いブートストラップリソースが残っている場合は、これらをすべて削除します。

```
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. 以下を **clusterconfigs** ディレクトリーから削除し、Terraform が失敗することを防ぎます。

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

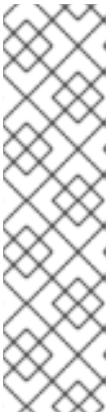
1.4.7. レジストリーの作成に関する問題

非接続レジストリーの作成時に、レジストリーのミラーリングを試行する際に User Not Authorized エラーが発生する場合があります。このエラーは、新規の認証を既存の **pull-secret.txt** ファイルに追加できない場合に生じる可能性があります。

手順

1. 認証が正常に行われていることを確認します。

```
$ /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
```



注記

インストールイメージのミラーリングに使用される変数の出力例:

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

RELEASE_IMAGE および **VERSION** の値は、**OpenShift インストールの環境のセットアップ**セクションの **OpenShift Installer の取得** の手順で設定されています。

2. レジストリーのミラーリング後に、非接続環境でこれにアクセスできることを確認します。

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

1.4.8. その他の問題点

1.4.8.1. runtime network not ready エラーへの対応

クラスターのデプロイメント後に、以下のエラーが発生する可能性があります。

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

Cluster Network Operator は、インストーラーによって作成される特別なオブジェクトに対応してネットワークコンポーネントをデプロイします。これは、コントロールプレーン (マスター) ノードが起動した後、ブートストラップコントロールプレーンが停止する前にインストールプロセスの初期段階で実行されます。これは、コントロールプレーン (マスター) ノードの起動の長い遅延や **apiserver** 通信の問題などの、より判別しづらいインストーラーの問題を示すことができます。

手順

1. **openshift-network-operator** namespace の Pod を検査します。

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

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

2. **provisioner** ノードで、ネットワーク設定が存在することを判別します。

```
$ 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
```

存在しない場合には、インストーラーはこれを作成していません。インストーラーがこれを作成しなかった理由を判別するには、以下のコマンドを実行します。

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

3. **network-operator** が実行されていることを確認します。

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

4. ログを取得します。

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

3 つ以上のコントロールプレーン (マスター) ノードを持つ高可用性クラスターの場合、Operator はリーダーの選択を実行し、他の Operator はすべてスリープ状態になります。詳細は、Troubleshooting を参照してください。

1.4.8.2. クラスターノードが DHCP 経由で正しい IPv6 アドレスを取得しない

クラスターノードが DHCP 経由で正しい IPv6 アドレスを取得しない場合は、以下の点を確認してください。

1. 予約された IPv6 アドレスが DHCP 範囲外にあることを確認します。
2. DHCP サーバーの IP アドレス予約では、予約で正しい DUID (DHCP 固有識別子) が指定されていることを確認します。以下に例を示します。

```
# 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. Route Announcement が機能していることを確認します。
4. DHCP サーバーが、IP アドレス範囲を提供する必要なインターフェイスでリッスンしていることを確認します。

1.4.8.3. クラスターノードが DHCP 経由で正しいホスト名を取得しない

IPv6 のデプロイメント時に、クラスターノードは DHCP でホスト名を取得する必要があります。**NetworkManager** はホスト名をすぐに割り当てない場合があります。コントロールプレーン (マスター) ノードは、以下のようなエラーを報告する可能性があります。

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

このエラーは、最初に DHCP サーバーからホスト名を受信せずにクラスターノードが起動する可能性があることを示しています。これにより、**kubelet** が **localhost.localdomain** ホスト名で起動します。エラーに対処するには、ノードによるホスト名の更新を強制します。

手順

1. **hostname** を取得します。

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

   ホスト名が **localhost** の場合は、以下の手順に進みます。

   注記

   X は、コントロールプレーンノード (別名マスターノード) 番号になります。

2. クラスターノードによる DHCP リースの更新を強制します。

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

   <bare-metal-nic> を、**baremetal** ネットワークに対応する有線接続に置き換えます。

3. **hostname** を再度確認します。

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

4. ホスト名が **localhost.localdomain** の場合は、**NetworkManager** を再起動します。

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

5. ホスト名がまだ **localhost.localdomain** の場合は、数分待機してから再度確認します。ホスト名が **localhost.localdomain** のままの場合は、直前の手順を繰り返します。
6. **nodeip-configuration** サービスを再起動します。

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

   このサービスは、正しいホスト名の参照で **kubelet** サービスを再設定します。

7. kubelet が直前の手順で変更された後にユニットファイル定義を再読み込みします。

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

8. **kubelet** サービスを再起動します。

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

9. **kubelet** が正しいホスト名で起動されていることを確認します。

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

再起動時など、クラスターの稼働後にクラスターノードが正しいホスト名を取得しない場合、クラスターの **csr** は保留中になります。**csr** は承認 **しません**。承認すると、他の問題が生じる可能性があります。

csr の対応

1. クラスターで CSR を取得します。

```
$ oc get csr
```

2. 保留中の **csr** に **Subject Name: localhost.localdomain** が含まれているかどうかを確認します。

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

3. **Subject Name: localhost.localdomain** が含まれる **csr** を削除します。

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

1.4.8.4. ルートがエンドポイントに到達しない

インストールプロセス時に、VRRP (Virtual Router Redundancy Protocol) の競合が発生する可能性があります。この競合は、特定のクラスター名を使用してクラスターデプロイメントの一部であった、以前に使用された OpenShift Container Platform ノードが依然として実行中であるものの、同じクラスター名を使用した現在の OpenShift Container Platform クラスターデプロイメントの一部ではない場合に発生する可能性があります。たとえば、クラスターはクラスター名 **openshift** を使用してデプロイされ、3 つのコントロールプレーン (マスター) ノードと 3 つのワーカーノードをデプロイします。後に、別のインストールで同じクラスター名 **openshift** が使用されますが、この再デプロイメントは 3 つのコントロールプレーン (マスター) ノードのみをインストールし、以前のデプロイメントの 3 つのワーカーノードを **ON** 状態のままにします。これにより、VRID (Virtual Router Identifier) の競合が発生し、VRRP が競合する可能性があります。

1. ルートを取得します。

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

2. サービスエンドポイントを確認します。

```
$ 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. コントロールプレーン (マスター) ノードからサービスへのアクセスを試行します。

```
[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** ノードからの **authentication-operator** エラーを特定します。

```
$ 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"
```

解決策

1. すべてのデプロイメントのクラスター名が一意であり、競合が発生しないことを確認します。
2. 同じクラスター名を使用するクラスターデプロイメントの一部ではない不正なノードをすべてオフにします。そうしないと、OpenShift Container Platform クラスターの認証 Pod が正常に起動されなくなる可能性があります。

1.4.8.5. 初回起動時の Ignition の失敗

初回起動時に、Ignition 設定が失敗する可能性があります。

手順

1. Ignition 設定が失敗したノードに接続します。

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

2. **machine-config-daemon-firstboot** サービスを再起動します。

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

1.4.8.6. NTP が同期しない

OpenShift Container Platform クラスターのデプロイメントは、クラスターノード間の NTP の同期クロックによって異なります。同期クロックがない場合、時間の差が 2 秒を超えるとクロックのドリフトによりデプロイメントが失敗する可能性があります。

手順

1. クラスターノードの **AGE** の差異の有無を確認します。以下に例を示します。

```
$ 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. クロックのドリフトによる一貫性のないタイミングの遅延について確認します。以下に例を示します。

```
$ 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
```

既存のクラスターでのクロックドリフトへの対応

1. **chrony.conf** ファイルを作成し、これを **base64** 文字列としてエンコードします。以下に例を示します。

```
$ 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>** を NTP サーバーの IP アドレスに置き換えます。出力をコピーします。

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

2. **MachineConfig** オブジェクトを作成します。**base64** 文字列を直前の手順の出力で生成される **[text-in-base-64]** 文字列に置き換えます。以下の例では、ファイルをコントロールプレーン (マスター) ノードに追加します。ワーカーノードのファイルを変更するか、またはワーカーロールの追加のマシン設定を作成できます。

```
$ 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]** を base64 文字列に置き換えます。

3. 設定ファイルのバックアップコピーを作成します。以下に例を示します。

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

4. 設定ファイルを適用します。

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

5. **System clock synchronized** の値が **yes** であることを確認します。

```
$ 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
```

デプロイメントの前にクロック同期を設定するには、マニフェストファイルを生成し、このファイルを **openshift** ディレクトリーに追加します。以下に例を示します。

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

クラスターの作成を継続します。

1.4.9. インストールの確認

インストール後に、インストーラーがノードおよび Pod を正常にデプロイしていることを確認します。

手順

1. OpenShift Container Platform クラスターノードが適切にインストールされると、以下の **Ready** 状態が **STATUS** 列に表示されます。

```
$ 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. インストーラーによりすべての Pod が正常にデプロイされたことを確認します。以下のコマンドは、実行中の Pod、または出力の一部として完了した Pod を削除します。

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