Assisted Installer for OpenShift Container Platform 2024

Assisted Installer を使用した OpenShift Container Platform のインストール

ユーザーガイド

Last Updated: 2024-06-27

Assisted Installer for OpenShift Container Platform 2024 Assisted Installer を使用した OpenShift Container Platform のインストール

ユーザーガイド

法律上の通知

概要

Assisted Installer とその使用法に関する情報

目次

はじめに

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。この取り組みは膨大な作業を要するため、これらの変更による更新は可能な範囲で段階的に行われます。詳細は、Red Hat CTO である Chris Wright のメッセージ をご覧ください。

RED HAT ドキュメントへのフィードバック (英語のみ)

Jira で **Create Issue** フォームを送信することで、フィードバックを提供したり、エラーを報告したりできます。Jira の課題は、Red Hat Hybrid Cloud Infrastructure Jira プロジェクトに作成されるため、フィードバックの進行状況を追跡できます。

1. Jira にログインしていることを確認してください。Jira アカウントをお持ちでない場合は、アカウントを作成してフィードバックを送信してください。
2. **Create issue** をクリックします。
    a. **Summary** フィールドと **Description** フィールドに入力します。**Description** フィールドに、ドキュメントの URL、章またはセクション番号、および問題の詳しい説明を入力します。フォーム内の他のフィールドは変更しないでください。
3. **Create** をクリックします。

Red Hat ドキュメントに関するご意見やご感想をお寄せください。

第1章 ASSISTED INSTALLER について

Red Hat OpenShift Container Platform の Assisted Installer は、Red Hat Hybrid Cloud Console で提供されているユーザーフレンドリーなインストールソリューションです。Assisted Installer は、ベアメタル、Nutanix、vSphere、Oracle Cloud Infrastructure を中心に、さまざまなデプロイメントプラットフォームをサポートします。

次のプラットフォームでは、オプションの HTTP/S プロキシーを使用して、接続された環境にオンプレミスの OpenShift Container Platform をインストールできます。

- 高可用性 OpenShift Container Platform またはシングルノード OpenShift クラスター
- プラットフォームが完全に統合されたベアメタルまたは vSphere 上の OpenShift Container Platform、または統合されていない他の仮想化プラットフォーム
- オプション: OpenShift Virtualization と Red Hat OpenShift Data Foundation

1.1. 機能

Assisted Installer は、インストール機能をサービスとして提供します。この Software as a Service (SaaS) アプローチには、次のような特徴があります。

Web インターフェイス

- インストール設定ファイルを手動で作成する代わりに、Hybrid Cloud Console を使用してクラスターをインストールできます。

ブートストラップノードが不要

- ブートストラッププロセスがクラスター内のノードで実行されるため、ブートストラップノードが必要ありません。

効率的なインストールワークフロー

- クラスターをデプロイするために、OpenShift Container Platform に関する詳細な知識は必要ありません。Assisted Installer が適切なデフォルト設定を提供します。
- OpenShift Container Platform インストーラーをローカルで実行する必要はありません。
- 最新のテスト済み z-stream リリース用の最新の Assisted Installer にアクセスできます。

高度なネットワークオプション

- Assisted Installer は、SDN と OVN を使用した IPv4 ネットワーク、OVN のみを使用した IPv6 およびデュアルスタックネットワーク、NMState ベースの静的 IP アドレス指定、および HTTP/S プロキシーをサポートします。
- OVN は、OpenShift Container Platform 4.12 以降のデフォルトの Container Network Interface (CNI) です。
- SDN は OpenShift Container Platform 4.14 までサポートされており、OpenShift Container Platform 4.15 では非推奨となっています。

インストール前の検証

- インストールする前に、Assisted Installer は次の設定をチェックします。

- ネットワーク接続
- ネットワーク帯域幅
- レジストリーへの接続
- ドメイン名のアップストリーム DNS 解決
- クラスターノード間の時刻同期
- クラスターノードハードウェア
- インストール設定パラメーター

REST API

- Assisted Installer REST API を使用してインストールプロセスを自動化できます。

1.2. インストールのカスタマイズ

1 つまたは複数のオプションを選択してインストールをカスタマイズできます。

これらのオプションは Operators としてインストールされ、コントロールプレーン上のサービスとアプリケーションをパッケージ化、デプロイ、管理するために使用されます。詳細は、Operator ドキュメント を参照してください。

高度な設定オプションが必要な場合は、インストール後にこれらの Operator をデプロイできます。

OpenShift Virtualization

OpenShift Virtualization をデプロイして、次のタスクを実行できます。

- Linux および Windows 仮想マシン (VM) を作成および管理します。
- クラスター内で Pod と仮想マシンワークロードを並行して実行します。
- さまざまなコンソールと CLI ツールを使用して仮想マシンに接続します。
- 既存の仮想マシンをインポートして複製します。
- 仮想マシンに接続されたネットワークインターフェイスコントローラーとストレージディスクを管理します。
- ノード間で仮想マシンをライブ移行します。

詳細は、OpenShift Virtualization のドキュメント を参照してください。

Kubernetes 向けマルチクラスターエンジン

Kubernetes のマルチクラスターエンジンをデプロイすると、大規模なマルチクラスター環境で次のタスクを実行できます。

- 最初のクラスターから追加の Kubernetes クラスターをプロビジョニングおよび管理します。

- Hosted Control Plane を使用すると、コントロールプレーンとデータプレーンを分離して管理コストを削減し、クラスターのデプロイメントを最適化できます。詳細は、ホストされたコントロールプレーンの概要 を 参照してください。

- GitOps Zero Touch Provisioning を使用して、大規模なリモートエッジサイトを管理します。詳細は、エッジコンピューティング を参照してください。
すべての OpenShift Container Platform クラスターに、Red Hat OpenShift Data Foundation を使用してマルチクラスターエンジンをデプロイできます。

重要

マルチクラスターエンジンとストレージの設定

OpenShift Data Foundation を **使用せずに** マルチクラスターエンジンをデプロイすると、次のシナリオが発生します。

- マルチノードクラスター: ストレージは設定されていません。インストールプロセス後にストレージを設定する必要があります。

- シングルノード OpenShift: LVM ストレージがインストールされています。

Logical Volume Manager Storage

Logical Volume Manager Storage (LVM Storage) を使用すると、リソースが制限されたクラスター上でブロックストレージを動的にプロビジョニングできます。詳細は、論理ボリューム マネージャーストレージを使用した永続ストレージ を 参照してください。

Red Hat OpenShift Data Foundation

ファイル、ブロック、およびオブジェクトストレージに Red Hat OpenShift Data Foundation を使用できます。このストレージオプションは、すべての OpenShift Container Platform クラスターに推奨されます。OpenShift Data Foundation には、別のサブスクリプションが必要です。詳細は、OpenShift Data Foundation のドキュメント を参照してください。

1.3. API のサポートポリシー

Assisted Installer API は、非推奨の発表から少なくとも 3 カ月間サポートされます。

第2章 前提条件

Assisted Installer は、以下の前提条件を検証して、インストールが正常に行われるようにします。

ファイアウォールを使用する場合は、Assisted Installer が機能するために必要なリソースにアクセスできるようにファイアウォールを設定する必要があります。

2.1. サポートされている CPU アーキテクチャー

Assisted Installer は、次の CPU アーキテクチャーをサポートしています。

- x86_64
- arm64
- ppc64le
- s390x

2.2. リソース要件

このセクションでは、さまざまなクラスターとインストールオプションのリソース要件について説明します。

Kubernetes のマルチクラスターエンジンには関連情報が必要です。

OpenShift Data Foundation や LVM Storage などのストレージを備えたマルチクラスターエンジンをデプロイする場合は、各ノードに関連情報を割り当てる必要もあります。

2.2.1. マルチノードクラスターリソース要件

マルチノードクラスターのリソース要件は、インストールオプションによって異なります。

マルチノードクラスターの基本的なインストール

- コントロールプレーンノード
  - 4 CPU コア
  - 16 GB RAM
  - 100 GB のストレージ

注記

ディスクは適度に高速で、etcd **wal_fsync_duration_seconds** p99 の継続時間が 10 ミリ秒未満である必要があります。詳細は、Red Hat ナレッジベースのソリューション How to Use 'fio' to Check Etcd Disk Performance in OCP を参照してください。

- コンピュートノード
  - 2 CPU コア
  - 8 GB RAM。

- 100 GB のストレージ

マルチノードクラスター + マルチクラスターエンジン

- 追加の 4 CPU コア
- 追加の 16 GB RAM

注記

OpenShift Data Foundation を使用せずにマルチクラスターエンジンをデプロイすると、ストレージは設定されません。インストール後にストレージを設定します。

マルチノードクラスター + マルチクラスターエンジン + OpenShift Data Foundation または LVM Storage

- 追加の 75 GB ストレージ

2.2.2. シングルノード OpenShift リソース要件

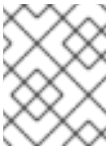
シングルノード OpenShift のリソース要件は、インストールオプションによって異なります。

シングルノード OpenShift の基本インストール

- 8 CPU コア
- 16 GB RAM
- 100 GB のストレージ

シングルノード OpenShift + マルチクラスターエンジン

- 追加の 8 CPU コア
- 追加の 32 GB RAM



注記

OpenShift Data Foundation を使用せずにマルチクラスターエンジンをデプロイすると、LVM Storage が有効になります。

シングルノード OpenShift + マルチクラスターエンジン + OpenShift Data Foundation または LVM Storage

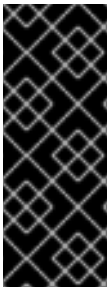
- 追加の 95 GB ストレージ

2.3. ネットワーク要件

ネットワークは次の要件を満たす必要があります。

- 静的 IP アドレス指定を使用しない場合は、DHCP サーバー。

- ベースドメイン名。次の要件が満たされていることを確認する必要があります。
  - ***.<cluster_name>.<base_domain>** などのワイルドカードがない場合、インストールは続行されません。
  - **api.<cluster_name>.<base_domain>** の DNS A/AAAA レコード。
  - ***.apps.<cluster_name>.<base_domain>** のワイルドカードを含む DNS A/AAAA レコード。
- ファイアウォールの外側のユーザーが **oc** CLI ツールを介してクラスターにアクセスできるようにする場合は、API URL 用にポート **6443** が開かれます。
- ファイアウォールの外側のユーザーがコンソールにアクセスできるようにする場合は、ポート **443** がコンソール用に開いています。
- ユーザー管理ネットワークを使用する場合、クラスター内の各ノードの DNS A/AAAA レコードがないと、インストールは続行されません。インストールの完了後にクラスター管理ネットワークを使用してクラスターに接続する場合は、クラスター内の各ノードに DNS A/AAAA レコードが必要ですが、Cluster Managed Networking を使用する場合は、A/AAAA レコードがなくてもインストールを続行できます。
- 静的 IP アドレス指定の使用時に事前設定されたホスト名で起動する場合は、クラスター内の各ノードの DNS PTR レコード。それ以外の場合、Assisted Installer には、ノードの名前をネットワークインターフェイスの MAC アドレスに変更する静的 IP アドレス指定を使用する場合のノードの自動名前変更機能があります。



重要

- トップレベルドメインレジストラーでの DNS A/AAAA レコードの設定は、更新にかなりの時間がかかる場合があります。インストールの遅延を防ぐために、インストールの前に A/AAAA レコードの DNS 設定が機能していることを確認してください。
- DNS レコードの例は、この章の **DNS 設定の例** を参照してください。

OpenShift Container Platform クラスターのネットワークは、以下の要件も満たしている必要があります。

- すべてのクラスターノード間の接続
- 各ノードのインターネットへの接続
- クラスターノード間の時刻同期のための NTP サーバーへのアクセス

2.4. DNS 設定の例

このセクションでは、Assisted Installer を使用して OpenShift Container Platform をデプロイするための DNS 要件を満たす A および PTR レコードの設定例を示します。サンプルは、特定の DNS ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、クラスター名は **ocp4** で、ベースドメインは **example.com** です。

2.4.1. DNS A レコードの設定例

BIND ゾーンファイルの以下の例は、このファイルはAssisted Installer を使用してインストールされたクラスターで名前解決するサンプル A レコードを示しています。

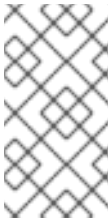
DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.1
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
control-plane0.ocp4.example.com. IN A 192.168.1.97 4
control-plane1.ocp4.example.com. IN A 192.168.1.98
control-plane2.ocp4.example.com. IN A 192.168.1.99
;
worker0.ocp4.example.com. IN A 192.168.1.11 5
worker1.ocp4.example.com. IN A 192.168.1.7
;
;EOF
```

1. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照します。

2. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照し、内部クラスター通信に使用されます。

3. ワイルドカードルートの名前解決を提供します。レコードは、アプリケーション Ingress ロードバランサーの IP アドレスを参照します。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress Controller Pod はデフォルトでワーカーマシンで実行されます。



注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

4. コントロールプレーンマシンの名前解決を提供します。

5 ワーカーマシンの名前解決を提供します。

2.4.2. DNS PTR レコードの設定例

BIND ゾーンファイルの例では、Assisted Installer を使用してインストールされたクラスターの逆引き名前解決の PTR レコードの例を示しています。

逆引きレコードの DNS ゾーンデータベースの例

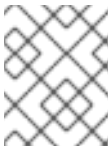
```
$$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
97.1.168.192.in-addr.arpa. IN PTR control-plane0.ocp4.example.com. 3
98.1.168.192.in-addr.arpa. IN PTR control-plane1.ocp4.example.com.
99.1.168.192.in-addr.arpa. IN PTR control-plane2.ocp4.example.com.
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 4
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com.
;
;EOF
```

1 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照します。

2 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照し、内部クラスター通信に使用されます。

3 コントロールプレーンマシンの逆引き DNS 解決を提供します。

4 ワーカーマシンの逆引き DNS 解決を提供します。



注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。

2.5. プリフライト検証

Assisted Installer は、インストール前にクラスターが前提条件を満たしていることを確認します。確認することで、インストール後の複雑なトラブルシューティングが不要になり、時間と労力が大幅に節約されます。ノードにソフトウェアをインストールする前に、Assisted Installer は次の検証を行います。

- ネットワーク接続の確保

- 十分なネットワーク帯域幅の確保
- レジストリーへの接続の確保
- アップストリーム DNS が必要なドメイン名を解決できるようにする手順
- クラスターノード間の時刻同期の確保
- クラスターノードが最小ハードウェア要件を満たしていることの確認
- インストール設定パラメーターの検証

Assisted Installer が前述の要件を正常に検証しない場合、インストールは続行されません。

第3章 アシステッドインストーラー WEB コンソールを使用したインストール

クラスターノードとネットワークの要件が満たされていることを確認したら、クラスターのインストールを開始できます。

3.1. インストール前の考慮事項

Assisted Installer を使用して OpenShift Container Platform をインストールする前に、以下の設定の選択を検討する必要があります。

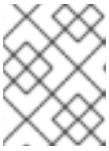
- 使用する基本ドメイン
- インストールする OpenShift Container Platform の製品バージョン
- フルクラスターまたは単一ノードの OpenShift をインストールするかどうか
- DHCP サーバーまたは静的ネットワーク設定を使用するかどうか
- IPv4 またはデュアルスタックネットワークを使用するかどうか
- OpenShift Virtualization をインストールするかどうか
- Red Hat OpenShift Data Foundation をインストールするかどうか
- Kubernetes のマルチクラスターエンジンをインストールするかどうか
- vSphere または Nutanix にインストールするときにプラットフォームと統合するかどうか
- 混合クラスターアーキテクチャーをインストールするかどうか

3.2. クラスターの詳細の設定

Assisted Installer Web ユーザーインターフェイスを使用してクラスターを作成するには、次の手順を使用します。

手順

1. Red Hat Hybrid Cloud コンソール にログインします。
2. **Red Hat OpenShift** タイルで、**Scale your applications** をクリックします。
3. メニューで、**Clusters** をクリックします。
4. **Create cluster** をクリックします。
5. **Datacenter** タブをクリックします。
6. **Assisted Installer** で、**Create cluster** をクリックします。
7. **Cluster Name** フィールドにクラスターの名前を入力します。
8. **Base domain** フィールドに、クラスターのベースドメインを入力します。クラスターのすべてのサブドメインは、この基本ドメインを使用します。



注記

ベースドメインは有効な DNS 名である必要があります。ベースドメインにワイルドカードドメインをセットアップしないでください。

9. インストールする OpenShift Container Platform のバージョンを選択します。

重要

- IBM Power および IBM zSystems プラットフォームの場合、OpenShift Container Platform 4.13 以降のみがサポートされます。
- 混合アーキテクチャークラスターのインストールの場合は、OpenShift Container Platform 4.12 以降を選択し、**-multi** オプションを使用します。混合アーキテクチャークラスターのインストール手順については、**関連情報** を参照してください。

10. オプション: OpenShift Container Platform をシングルノードにインストールする場合は、**Install single node Openshift (SNO)** を選択します。

注記

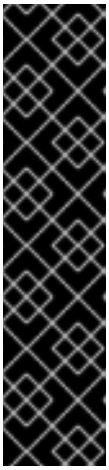
現在、SNO は IBM zSystems および IBM Power プラットフォームではサポートされていません。

11. オプション: Assisted Installer には、アカウントに関連付けられたプルシークレットがすでにあります。別のプルシークレットを使用する場合は、**Edit pull secret** を選択します。

12. オプション: OpenShift Container Platform をサードパーティープラットフォームにインストールする場合は、**Integrate with external parter platforms** リストから対象のプラットフォームを選択します。有効な値は **Nutanix**、**vSphere**、または **Oracle Cloud Infrastructure** です。Assisted Installer には、デフォルトのプラットフォーム統合はありません。

注記

各外部パートナー統合の詳細は、**関連情報** を参照してください。



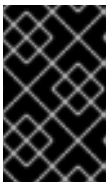
重要

Assisted Installer は、OpenShift Container Platform 4.14 以降の Oracle Cloud Infrastructure (OCI) 統合をサポートします。OpenShift Container Platform 4.14 の OCI 統合はテクノロジープレビュー機能でのみ利用できます。テクノロジープレビュー機能は、Red Hat 製品サポートのサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではない場合があります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビューの機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

13. オプション: Assisted Installer は、デフォルトで **x86_64** CPU アーキテクチャーを使用します。OpenShift Container Platform を別のアーキテクチャーにインストールする場合は、使用する

それぞれのアーキテクチャーを選択します。有効な値は、**arm64**、**ppc64le**、および **s390x** です。一部の機能は、**arm64**、**ppc64le**、および **s390x** CPU アーキテクチャーでは利用できないことに注意してください。



重要

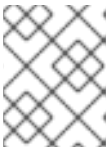
混合アーキテクチャーのクラスターインストールの場合は、デフォルトの **x86_64** アーキテクチャーを使用します。混合アーキテクチャークラスターのインストール手順については、**関連情報** を参照してください。

14. オプション: インストールに含めるカスタムマニフェストが少なくとも 1 つある場合は、**Include custom manifests** を選択します。カスタムマニフェストには、現在 Assisted Installer でサポートされていない追加の設定が含まれています。チェックボックスを選択すると、ウィザードに **Custom manifests** ページが追加され、そこでマニフェストをアップロードします。

重要

- OpenShift Container Platform を Oracle Cloud Infrastructure (OCI) サードパーティープラットフォームにインストールする場合は、Oracle が提供するカスタムマニフェストを追加することが必須です。
- すでにカスタムマニフェストを追加している場合は、**Include custom manifests** チェックボックスをオフにすると、それらはすべて自動的に削除されます。削除を確定するように求められます。

15. オプション: Assisted Installer はデフォルトで DHCP ネットワークに設定されます。DHCP 予約の代わりにクラスターノードに静的 IP 設定、ブリッジ、または結合を使用している場合は、**静的 IP、ブリッジ、および結合 を** 選択します。



注記

静的 IP 設定は、Oracle Cloud Infrastructure 上の OpenShift Container Platform インストールではサポートされていません。

16. オプション: インストールディスクの暗号化を有効にする場合は、インストールディスクの **暗号化を有効に** するで、単一ノード OpenShift の **コントロールプレーンノード、ワーカー** を選択できます。マルチノードクラスターの場合、**コントロールプレーンノード** を選択してコントロールプレーンノードのインストールディスクを暗号化し、**ワーカー** を選択してワーカーノードのインストールディスクを暗号化できます。

重要

インストールの開始後は、基本ドメイン、SNO チェックボックス、CPU アーキテクチャー、ホストのネットワーク設定、またはディスク暗号化を変更できません。

関連情報

- オプション: Nutanix へのインストール
- オプション: vSphere へのインストール
- オプション: Oracle Cloud Infrastructure (OCI) へのインストール

3.3. オプション: 静的ネットワークの設定

Assisted Installer は、SDN が OpenShift Container Platform 4.14 および OVN までの IPv4 ネットワークをサポートし、OVN のみを使用した IPv6 とデュアルスタックネットワークをサポートします。Assisted Installer は、IP アドレス/MAC アドレスマッピングを使用した静的ネットワークインターフェイスを使用したネットワークの設定をサポートしています。Assisted Installer は、ホスト用の宣言型ネットワークマネージャー API である NMState ライブラリーを使用したホストネットワークインターフェイスの設定もサポートしています。NMState を使用して、静的 IP アドレス指定、ボンディング、VLAN、およびその他の高度なネットワーク機能を備えたホストをデプロイできます。まず、ネットワーク全体の設定を設定する必要があります。次に、ホストごとにホスト固有の設定を作成する必要があります。

注記

z/VM を使用した IBM Z へのインストールの場合は、z/VM ノードと vSwitch が静的ネットワークと NMState に対して適切に設定されていることを確認してください。また、プールの MAC アドレスによって NMState の問題が発生する可能性があるため、z/VM ノードには固定の MAC アドレスが割り当てられている必要があります。

手順

1. インターネットプロトコルのバージョンを選択します。有効なオプションは **IPv4** と **Dual stack** です。
2. クラスターホストが共有 VLAN 上にある場合は、VLAN ID を入力します。
3. ネットワーク全体の IP アドレスを入力します。**Dual stack** ネットワークを選択した場合は、IPv4 と IPv6 の両方のアドレスを入力する必要があります。
    a. クラスターネットワークの IP アドレス範囲を CIDR 表記で入力します。
    b. デフォルトゲートウェイの IP アドレスを入力します。
    c. DNS サーバーの IP アドレスを入力します。
4. ホスト固有の設定を入力します。
    a. 単一のネットワークインターフェイスを使用する静的 IP アドレスのみを設定する場合は、フォームビューを使用して、各ホストの IP アドレスと MAC アドレスを入力します。
    b. 複数のインターフェイス、ボンディング、またはその他の高度なネットワーク機能を使用している場合は、YAML ビューを使用し、NMState 構文を使用して各ホストに必要なネットワーク状態を入力します。次に、ネットワーク設定で使用される各ホストインターフェイスの MAC アドレスとインターフェイス名を追加します。

関連情報

- NMState version 2.1.4

3.4. オプション: OPERATOR のインストール

この手順は任意です。

前提条件と設定オプションについては、製品ドキュメントを参照してください。

- OpenShift Virtualization

- Kubernetes 用のマルチクラスターエンジン
- Red Hat OpenShift Data Foundation
- Logical Volume Manager Storage

高度なオプションが必要な場合は、クラスターをインストールした後に Operator をインストールします。

手順

1. 次のオプションから 1 つ以上を選択します。
   - **Install OpenShift Virtualization**
   - **Install multicluster engine**
     すべての OpenShift Container Platform クラスターに、OpenShift Data Foundation を使用してマルチクラスターエンジンをデプロイできます。

     重要

     OpenShift Data Foundation を **使用せずに** マルチクラスターエンジンをデプロイすると、次のようなストレージ設定になります。

     - マルチノードクラスター: ストレージは設定されていません。インストール後にストレージを設定する必要があります。
     - シングルノード OpenShift: LVM ストレージがインストールされています。
   - **Logical Volume Manager Storage のインストール**
   - **Install OpenShift Data Foundation**
2. **Next** をクリックします。

3.5. クラスターへのホストの追加

1 つ以上のホストをクラスターに追加する必要があります。クラスターにホストを追加するには、検出 ISO を生成する必要があります。検出 ISO は、エージェントを使用して Red Hat Enterprise Linux CoreOS (RHCOS) インメモリーを実行します。

クラスター上の各ホストに対して次の手順を実行します。

手順

1. **Add hosts** ボタンをクリックし、プロビジョニングタイプを選択します。
   a. **Minimal image file: Provision with virtual media**を選択して、起動に必要なデータを取得する小さなイメージをダウンロードします。ノードには仮想メディア機能が必要です。これは、**x86_64** および **arm64** アーキテクチャーで推奨される方法です。
   b. **Full image file: Provision with physical media**を選択して、より大きなフルイメージをダウンロードします。これは、RHEL KVM を使用してインストールする場合の **ppc64le** アーキテクチャーおよび **s390x** アーキテクチャーの推奨方法です。

c. **iPXE: Provision from your network server**を選択して、iPXE を使用してホストを起動します。これは、z/VM ノードを使用する IBM Z で推奨される方法です。ISO ブートは、RHEL KVM インストールで推奨される方法です。

注記

- RHEL KVM にインストールする場合、KVM ホスト上の VM が初回起動時に再起動されず、手動での起動が必要になることがあります。
- Oracle Cloud Infrastructure に OpenShift Container Platform をインストールする場合は、**Minimal image file: Provision with virtual media only** を選択します。

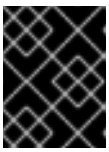
2. オプション: デフォルトのワーカーノードに加えて **Run workloads on control plane nodes**スイッチを有効にします。

注記

このオプションは、5 つ以上のノードのクラスターで使用できます。5 ノード未満のクラスターの場合、デフォルトでは、システムはコントロールプレーンノードでのみワークロードを実行します。詳細は、**関連情報** の **スケジュール可能なコントロールプレーンノードの設定** を参照してください。

3. オプション: クラスターホストがプロキシーの使用を必要とするファイアウォールの内側にある場合は、**Configure cluster-wide proxy settings**を選択します。プロキシーサーバーの HTTP および HTTPS URL のユーザー名、パスワード、IP アドレス、およびポートを入力します。

4. オプション: **core** ユーザーとしてクラスターノードに接続できるように、SSH 公開キーを追加します。クラスターノードにログインすると、インストール中にデバッグ情報を入手できます。



重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

a. ローカルマシンに既存の SSH キーペアがない場合は、クラスターノード SSH アクセス用のキーペアの生成 の手順に従います。

b. **SSH public key** フィールドで **Browse** をクリックして、SSH 公開鍵を含む **id_rsa.pub** ファイルをアップロードします。あるいは、ファイルマネージャーからファイルをフィールドにドラッグアンドドロップします。ファイルマネージャーでファイルを表示するには、メニューで **Show hidden files** を選択します。

5. オプション: クラスターホストが再暗号化中間者 (MITM) プロキシーを使用するネットワーク内にある場合、またはクラスターがコンテナーイメージレジストリーなどの他の目的で証明書を信頼する必要がある場合は、**Configure cluster-wide trusted certificates**を選択します。X.509 形式で追加の証明書を追加します。

6. 必要に応じて検出イメージを設定します。

7. オプション: 仮想化プラットフォームにインストールし、プラットフォームと統合する場合は、**Integrate with platform** を選択します。すべてのホストを起動し、それらがホストインベントリーに表示されることを確認する必要があります。すべてのホストが同じプラットフォー

ム上にある必要があります。

8. **Generate Discovery ISO** または **Generate Script File** をクリックします。
9. 検出 ISO または iPXE スクリプトをダウンロードします。
10. 検出イメージまたは iPXE スクリプトを使用してホストを起動します。

関連情報

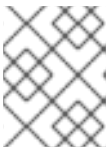
- 詳細は、検出イメージの設定 を参照してください。
- 詳細は、検出イメージを使用したホストの起動 を参照してください。
- 詳細は、Red Hat Enterprise Linux 9 - 仮想化の設定および管理 を参照してください。
- 詳細は、How to configure a VIOS Media Repository/Virtual Media Library を参照してください。
- Web コンソールを使用した Nutanix へのホストの追加
- vSphere へのホストの追加
- スケジュール可能なコントロールプレーンノードの設定

3.6. ホストの設定

検出 ISO を使用してホストを起動すると、ページの下部にあるテーブルにホストが表示されます。オプションで各ホストのホスト名およびロールを設定できます。必要に応じてホストを削除することもできます。

手順

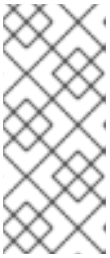
1. ホストの **Options** (⋮) メニューから ホスト名の **Change hostname** を選択します。必要に応じて、ホストの新しい名前を入力し、**Change** をクリックします。各ホストに有効で一意のホスト名があることを確認する必要があります。
   または、**Actions** リストから **Change hostname** を選択して、複数の選択したホストの名前を変更します。**Change Hostname** ダイアログで新しい名前を入力し、**{{n}}** を含めて各ホスト名を一意にします。次に、**Change** をクリックします。

   注記

   入力すると、**Preview** ペインに新しい名前が表示されます。名前は、ホストごとに 1 桁ずつ増加する点を除き、選択したすべてのホストで同一になります。

2. **Options** (⋮) メニューから、 **Delete host** を選択してホストを削除できます。**Delete** をクリックして削除を確定します。
   または、**Actions** リストから **Delete** を選択して、選択した複数のホストを同時に削除します。次に、**Delete hosts** をクリックします。



注記

通常のデプロイメントでは、クラスターには 3 つ以上のホストを含めることができ、これら 3 つはコントロールプレーンホストである必要があります。コントロールプレーンでもあるホストを削除した場合、またはホストが 2 つだけ残った場合は、システムの準備ができていないことを示すメッセージが表示されます。ホストを復元するには、検出 ISO からホストを再起動する必要があります。

3. ホストの **オプション** (⋮) メニューから、必要に応じて **View host events** を選択します。リスト内のイベントは時系列に表示されます。

4. マルチホストクラスターの場合、ホスト名の横にある **Role** 列で、メニューをクリックしてホストのロールを変更できます。
ロールを選択しない場合、Assisted Installer がロールを自動的に割り当てます。コントロールプレーンノードの最小ハードウェア要件は、ワーカーノードの要件を超えています。ホストにロールを割り当てる場合は、ハードウェアの最小要件を満たすホストにコントロールプレーンのロールを割り当てるようにしてください。

5. **Status** リンクをクリックして、ホストのハードウェア、ネットワーク、および Operator の検証を表示します。

6. ホスト名の左側にある矢印をクリックして、ホストの詳細を展開します。

すべてのクラスターホストが **Ready** のステータスで表示されたら、次の手順に進みます。

3.7. ストレージディスクの設定

ホスト検出中に取得されるホストごとに、複数のストレージディスクを指定できます。Assisted Installer ウィザードの **Storage** ページに、ストレージディスクがホストに一覧表示されます。

オプションで、各ディスクのデフォルト設定を変更できます。

インストールディスクの変更

Assisted Installer は、デフォルトでインストールディスクをランダムに割り当てます。ホストに複数のストレージディスクがある場合は、別のディスクを選択してインストールディスクとして機能させることができます。これにより、以前のディスクは自動的に割り当てが解除されます。

手順

1. ウィザードの **Storage** ページに移動します。

2. ホストを拡張して、関連するストレージディスクを表示します。

3. **Role** リストから **Installation disk** を選択します。

4. すべてのストレージディスクが **Ready** ステータスに戻る場合は、次の手順に進みます。

ディスクフォーマットの無効化

Assisted Installer は、インストールディスクとして定義されているかどうかに関係なく、デフォルトでインストールプロセス中にフォーマットするためにすべての起動可能なディスクをマークします。フォーマットすると、データが失われます。

特定のディスクのフォーマットを無効にすることもできます。これは、ブート可能なディスクが、主に起動順序の観点からインストールプロセスを干渉する可能性があるため、注意して実行する必要があります。

インストールディスクのフォーマットを無効にできません。

手順

1. ウィザードの **Storage** ページに移動します。
2. ホストを拡張して、関連するストレージディスクを表示します。
3. ディスクの **Format** をクリアします。
4. すべてのストレージディスクが **Ready** ステータスに戻る場合は、次の手順に進みます。

関連情報

- ホストの設定

3.8. ネットワークの設定

OpenShift Container Platform をインストールする前に、クラスターネットワークを設定する必要があります。

手順

1. **Networking** ページで、まだ選択されていない場合は、次のいずれかを選択します。
   - **クラスター管理ネットワーク**: クラスター管理ネットワークを選択すると、Assisted Installer は、API および Ingress VIP アドレスを管理するための **keepalived** および Virtual Router Redundancy Protocol (VRRP) を含む標準ネットワークトポロジーを設定することを意味します。

     注記

     - 現在、クラスター管理ネットワークは、OpenShift Container Platform バージョン 4.13 の IBM zSystems および IBM Power ではサポートされていません。
     - Oracle Cloud Infrastructure (OCI) は、ユーザー管理のネットワーク設定を備えた OpenShift Container Platform 4.14 でのみ使用できます。

   - **User-Managed Networking**: ユーザー管理のネットワークを選択すると、OpenShift Container Platform を非標準のネットワークトポロジーでデプロイできます。たとえば、**keepalived** や VRRP の代わりに外部ロードバランサーを使用してデプロイする場合や、多数の異なる L2 ネットワークセグメントにクラスターノードをデプロイする場合などです。
2. クラスター管理ネットワークの場合は、以下の設定を設定します。
   a. **マシンネットワーク** を定義します。デフォルトのネットワークを使用するか、サブネットを選択できます。
   b. **API 仮想 IP** を定義します。API 仮想 IP は、すべてのユーザーが対話し、プラットフォームを設定するためのエンドポイントを提供します。
   c. **Ingress 仮想 IP** を定義します。Ingress 仮想 IP は、クラスターの外部から流れるアプリケーショントラフィックのエンドポイントを提供します。

3. ユーザー管理のネットワークの場合は、次の設定を設定します。

    a. **Networking stack type** を選択します。

        - **IPv4** : ホストが IPv4 のみを使用している場合は、このタイプを選択します。
        - **デュアルスタック**: ホストが IPv4 と IPv6 を併用している場合、デュアルスタックを選択できます。

    b. **マシンネットワーク** を定義します。デフォルトのネットワークを使用するか、サブネットを選択できます。

    c. **API 仮想 IP** を定義します。API 仮想 IP は、すべてのユーザーが対話し、プラットフォームを設定するためのエンドポイントを提供します。

    d. **Ingress 仮想 IP** を定義します。Ingress 仮想 IP は、クラスターの外部から流れるアプリケーショントラフィックのエンドポイントを提供します。

    e. オプション: **Allocate IPs via DHCP server**を選択して、DHCP サーバーを使用して **API IP** と **Ingress IP** を自動的に割り当てることができます。

4. オプション: **Use advanced networking** を選択して、以下の高度なネットワークプロパティーを設定します。

    - **クラスターネットワーク CIDR**: Pod IP アドレスが割り当てられる IP アドレスブロックを定義します。
    - **クラスターネットワークホストプリフィックス**: 各ノードに割り当てるサブネットプリフィックス長を定義します。
    - **サービスネットワーク CIDR**: サービス IP アドレスに使用する IP アドレスを定義します。
    - **Network type**: 標準ネットワーク用の **Software-Defined Networking (SDN)** または IPv6、デュアルスタックネットワーク、Telco 機能用の **Open Virtual Networking (OVN)** のいずれかを選択します。OpenShift Container Platform 4.12 以降のリリースでは、OVN がデフォルトの Container Network Interface (CNI) です。OpenShift Container Platform 4.15 以降のリリースでは、**Software-Defined Networking (SDN)** はサポートされません。

関連情報

- ネットワーク設定

3.9. カスタムリポジトリーの追加

カスタムマニフェストは、Assisted Installer のユーザーインターフェイスで現在サポートされていない高度な設定が含まれる JSON または YAML ファイルです。カスタムマニフェストを作成することも、サードパーティーが提供するマニフェストを使用することもできます。

カスタムマニフェストは、ファイルシステムから **openshift** フォルダーまたは **manifests** フォルダーにアップロードできます。カスタムマニフェストファイルの数に制限はありません。

一度にアップロードできるファイルは 1 つだけです。ただし、アップロードされた各ファイルには複数のカスタムマニフェストを含めることができます。マルチドキュメントの YAML マニフェストのアップロードは、YAML ファイルを個別に追加するよりも高速です。

単一のカスタムマニフェストを含むファイルの場合は、使用可能なファイル拡張は、**.yaml**、**.yml**、または **.json** などです。

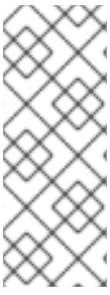
単一のカスタムマニフェストの例

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: master
  name: 99-openshift-machineconfig-master-kargs
spec:
  kernelArguments:
    - loglevel=7
```

複数のカスタムマニフェストを含むファイルの場合、使用可能なファイルタイプは、**.yaml** または **.yml** などです。

複数のカスタムマニフェストの例

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: master
  name: 99-openshift-machineconfig-master-kargs
spec:
  kernelArguments:
    - loglevel=7
---
apiVersion: machineconfiguration.openshift.io/v2
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-openshift-machineconfig-worker-kargs
spec:
  kernelArguments:
    - loglevel=5
```



注記

- OpenShift Container Platform を Oracle Cloud Infrastructure (OCI) 外部プラットフォームにインストールする場合は、Oracle が提供するカスタムマニフェストを追加する必要があります。vSphere や Nutanix などの追加の外部パートナー統合の場合、この手順はオプションです。
- カスタムマニフェストの詳細は、**関連情報** を参照してください。

Assisted Installer ユーザーインターフェイスでのカスタムマニフェストのアップロード

カスタムマニフェストをアップロードする場合は、マニフェスト名を入力し、宛先フォルダーを選択します。

前提条件

- ファイルシステムに少なくとも 1 つのカスタムマニフェストファイルが保存されている。

手順

1. ウィザードの **Cluster details** ページで、**Include custom manifests** チェックボックスを選択します。

2. **Custom manifest** ページの **folder** フィールドで、カスタムマニフェストファイルを保存する Assisted Installer フォルダーを選択します。オプションには **openshift** または **manifest** が含まれます。

3. **ファイル名** フィールドに、拡張子を含むマニフェストファイルの名前を入力します。たとえば、**manifest1.json** または **multiple1.yaml** です。

4. **Content** で **Upload** アイコンまたは **Browse** ボタンをクリックしてファイルをアップロードします。または、ファイルをファイルシステムから **Content** フィールドにドラッグします。

5. 別のマニフェストをアップロードするには、**Add another manifest** をクリックし、プロセスを繰り返します。これにより、以前にアップロードしたマニフェストが保存されます。

6. **Next** をクリックしてすべてのマニフェストを保存し、**Review and create** ページに移動します。アップロードしたカスタムマニフェストは **Custom manifests** の下に一覧表示されます。

Assisted Installer ユーザーインターフェイスでのカスタムマニフェストの変更

アップロードしたカスタムマニフェストのフォルダーとファイル名を変更できます。既存のマニフェストのコンテンツをコピーしたり、Chrome のダウンロード設定で定義されたフォルダーにダウンロードしたりすることもできます。

アップロードしたマニフェストのコンテンツは、変更できません。ただし、ファイルは上書きできます。

前提条件

- 1 つ以上のカスタムマニフェストファイルをアップロードしている。

手順

1. フォルダーを変更するには、**Folder** リストからマニフェスト用の別のフォルダーを選択します。

2. ファイル名を変更するには、**File name** フィールドにマニフェストの新しい名前を入力します。

3. マニフェストを上書きするには、新しいマニフェストを同じファイル名で保存します。

4. マニフェストをファイルとしてファイルシステムに保存するには、**Download** アイコンをクリックします。

5. マニフェストをコピーするには、**Copy to clipboard** アイコンをクリックします。

6. 変更を適用するには、**Add another manifest** または **Next** をクリックします。

Assisted Installer ユーザーインターフェイスでのカスタムマニフェストの削除

以下の 2 つの方法のいずれかで、インストール前にアップロードしたカスタムのマニフェストを削除できます。

- 1 つ以上のマニフェストを個別に削除する。

- すべてのマニフェストを一度に削除する。

マニフェストを削除すると、その操作を元に戻すことはできません。回避策は、マニフェストを再度アップロードすることです。

単一マニフェストの削除
一度に 1 つのマニフェストを削除できます。このオプションでは、最後に残ったマニフェストを削除できません。

前提条件

- 2 つ以上のカスタムマニフェストファイルをアップロードしている。

手順

1. **Custom manifests** ページに移動します。
2. マニフェスト名の上にマウスを置くと、**Delete** (マイナス) アイコンが表示されます。
3. アイコンをクリックし、ダイアログボックスの **Delete** をクリックします。

すべてのマニフェストの削除
すべてのカスタムマニフェストを一度に削除できます。これにより、**Custom manifest** ページも非表示になります。

前提条件

- 1 つ以上のカスタムマニフェストファイルをアップロードしている。

手順

1. ウィザードの **Cluster details** ページに移動します。
2. **Include custom manifests** チェックボックスをオフにします。
3. **Remove custom manifests** ダイアログボックスで、**Remove** をクリックします。

関連情報

- マニフェスト設定ファイル
- Multi-document YAML files

3.10. インストール前の検証

Assisted Installer は、インストール前にクラスターが前提条件を満たしていることを確認します。確認することで、インストール後の複雑なトラブルシューティングが不要になり、時間と労力が大幅に節約されます。クラスターをインストールする前に、クラスターと各ホストがインストール前の検証に合格していることを確認してください。

関連情報

- インストール前の検証

3.11. クラスターのインストール

設定が完了し、すべてのノードが **Ready** になったら、インストールを開始できます。インストールプロセスにはかなりの時間がかかりますが、Assisted Installer Web コンソールからインストールを監視できます。ノードはインストール中に再起動し、インストール後に初期化されます。

手順

1. **Begin installation** を押します。
2. 特定のホストのインストールステータスを表示するには、**Host Inventory** リストの **Status** 列のリンクをクリックします。

3.12. インストールの完了

クラスターがインストールされて初期化されると、Assisted Installer はインストールが完了したことを示します。Assisted Installer は、コンソール URL、**kubeadmin** のユーザー名とパスワード、および **kubeconfig** ファイルを提供します。さらに、Assisted Installer は、OpenShift Container Platform バージョン、ベースドメイン、CPU アーキテクチャー、API および Ingress IP アドレス、クラスターおよびサービスネットワーク IP アドレスを含むクラスターの詳細を提供します。

前提条件

- **oc** CLI ツールがインストールされている。

手順

1. **kubeadmin** のユーザー名とパスワードのコピーを作成します。
2. **kubeconfig** ファイルをダウンロードして、作業ディレクトリーの下の **auth** ディレクトリーにコピーします。

```
$ mkdir -p <working_directory>/auth
```

```
$ cp kubeadmin <working_directory>/auth
```

注記

kubeconfig ファイルは、インストールの完了後 24 時間はダウンロードできます。

3. **kubeconfig** ファイルをお使いの環境に追加します。

```
$ export KUBECONFIG=<your working directory>/auth/kubeconfig
```

4. **oc** CLI ツールでログインします。

```
$ oc login -u kubeadmin -p <password>
```

<password> を **kubeadmin** ユーザーのパスワードに置き換えます。

5. Web コンソールの URL をクリックするか、**Launch OpenShift Console** をクリックしてコンソールを開きます。

6. **kubeadmin** のユーザー名とパスワードを入力します。OpenShift Container Platform コンソールの指示に従って、アイデンティティープロバイダーを設定し、アラートレシーバーを設定します。

7. OpenShift Container Platform コンソールのブックマークを追加します。

8. インストール後のプラットフォーム統合手順を完了します。

関連情報

- Nutanix のインストール後の設定
- vSphere のインストール後の設定

第4章 ASSISTED INSTALLER API を使用したインストール

クラスターノードとネットワークの要件が満たされていることを確認したら、Assisted Installer API を使用してクラスターのインストールを開始できます。API を使用するには、次の手順を実行する必要があります。

- API 認証を設定します。
- プルシークレットを設定します。
- 新しいクラスター定義を登録します。
- クラスターのインフラストラクチャー環境を作成します。

これらの手順を実行すると、クラスター定義の変更、検出 ISO の作成、クラスターへのホストの追加、およびクラスターのインストールが可能になります。このドキュメントは Assisted Installer API のすべてのエンドポイントをカバーしているわけではありませんが、API ビューアー または swagger.yaml ファイルですべてのエンドポイントを確認できます。

4.1. オフライントークンの生成

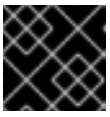
Assisted Installer Web コンソールからオフライントークンをダウンロードします。オフライントークンを使用して API トークンを設定します。

前提条件

- **jq** をインストールする。
- クラスター作成権限を持つユーザーとして OpenShift Cluster Manager にログインする。

手順

1. メニューで **Downloads** をクリックします。
2. **OpenShift Cluster Manager API Token** の下の **Tokens** セクションで、**View API Token** をクリックします。
3. **Load Token** をクリックします。



重要

ポップアップブロッカーを無効にします。

4. **Your API token** セクションで、オフライントークンをコピーします。
5. 端末で、オフライントークンを **OFFLINE_TOKEN** 変数に設定します。

```
$ export OFFLINE_TOKEN=<copied_token>
```

ヒント

オフライントークンを永続的にするには、プロファイルに追加します。

6. (オプション) **OFFLINE_TOKEN** 変数の定義を確認します。

```
$ echo ${OFFLINE_TOKEN}
```

4.2. REST API を使用した認証

API 呼び出しには、API トークンによる認証が必要です。変数名として **API_TOKEN** を使用すると仮定すると、API 呼び出しに **-H "Authorization: Bearer ${API_TOKEN}"** を追加して、REST API で認証します。

注記

API トークンは 15 分後に期限切れになります。

前提条件

- **OFFLINE_TOKEN** 変数を生成している。

手順

1. コマンドラインターミナルで、**OFFLINE_TOKEN** を使用して **API_TOKEN** 変数を設定し、ユーザーを検証します。

```
$ export API_TOKEN=$( \
  curl \
  --silent \
  --header "Accept: application/json" \
  --header "Content-Type: application/x-www-form-urlencoded" \
  --data-urlencode "grant_type=refresh_token" \
  --data-urlencode "client_id=cloud-services" \
  --data-urlencode "refresh_token=${OFFLINE_TOKEN}" \
  "https://sso.redhat.com/auth/realms/redhat-external/protocol/openid-connect/token" \
  | jq --raw-output ".access_token" \
)
```

2. **API_TOKEN** 変数定義を確認します。

```
$ echo ${API_TOKEN}
```

3. トークン生成方法の 1 つのパスにスクリプトを作成します。以下に例を示します。

```
$ vim ~/.local/bin/refresh-token
```

```
export API_TOKEN=$( \
  curl \
  --silent \
  --header "Accept: application/json" \
  --header "Content-Type: application/x-www-form-urlencoded" \
  --data-urlencode "grant_type=refresh_token" \
  --data-urlencode "client_id=cloud-services" \
  --data-urlencode "refresh_token=${OFFLINE_TOKEN}" \
```

```
  "https://sso.redhat.com/auth/realms/redhat-external/protocol/openid-connect/token" \
  | jq --raw-output ".access_token" \
)
```

次に、ファイルを保存します。

4. ファイルモードを変更して実行可能にします。

```
$ chmod +x ~/.local/bin/refresh-token
```

5. API トークンを更新します。

```
$ source refresh-token
```

6. 次のコマンドを実行して、API にアクセスできることを確認します。

```
$ curl -s https://api.openshift.com/api/assisted-install/v2/component-versions -H
"Authorization: Bearer ${API_TOKEN}" | jq
```

出力例

```
{
  "release_tag": "v2.11.3",
  "versions": {
    "assisted-installer": "registry.redhat.io/rhai-tech-preview/assisted-installer-rhel8:v1.0.0-
211",
    "assisted-installer-controller": "registry.redhat.io/rhai-tech-preview/assisted-installer-
reporter-rhel8:v1.0.0-266",
    "assisted-installer-service": "quay.io/app-sre/assisted-service:78d113a",
    "discovery-agent": "registry.redhat.io/rhai-tech-preview/assisted-installer-agent-
rhel8:v1.0.0-195"
  }
}
```

4.3. プルシークレットの設定

Assisted Installer API 呼び出しの多くは、プルシークレットを必要とします。プルシークレットをファイルにダウンロードして、API 呼び出しで参照できるようにします。プルシークレットは、リクエストの JSON オブジェクト内の値として含まれる JSON オブジェクトです。プルシークレットの JSON は、引用符をエスケープするようにフォーマットする必要があります。以下に例を示します。

更新前

```
{"auths":{"cloud.openshift.com": ...
```

更新後

```
{\"auths\":{\"cloud.openshift.com\": ...
```

手順

1. メニューで **OpenShift** をクリックします。

2. サブメニューで **Downloads** をクリックします。
3. **Pull secret** の下の **Tokens** セクションで、**Download** をクリックします。
4. シェル変数からプルシークレットを使用するには、次のコマンドを実行します。

```
$ export PULL_SECRET=$(cat ~/Downloads/pull-secret.txt | jq -R .)
```

5. **jq** を使用してプルシークレットファイルを丸呑みするには、**pull_secret** 変数で参照し、値を **tojson** にパイプして、エスケープされた JSON として適切にフォーマットされていることを確認します。以下に例を示します。

```
$ curl https://api.openshift.com/api/assisted-install/v2/clusters \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d "$(jq --null-input \
        --slurpfile pull_secret ~/Downloads/pull-secret.txt ' 1
    {
        "name": "testcluster",
        "high_availability_mode": "None",
        "openshift_version": "4.11",
        "pull_secret": $pull_secret[0] | tojson, 2
        "base_dns_domain": "example.com"
    }
')"
```

1. プルシークレットファイルを丸呑みします。
2. プルシークレットをエスケープされた JSON 形式にフォーマットします。

6. **PULL_SECRET** 変数の定義を確認します。

```
$ echo ${PULL_SECRET}
```

4.4. オプション: SSH 公開鍵の生成

OpenShift Container Platform のインストール中に、オプションでインストールプログラムに SSH 公開鍵を指定できます。これは、インストールエラーのトラブルシューティングを行うときに、リモートノードへの SSH 接続を開始するのに役立ちます。

認証に使用するローカルマシンに既存の SSH キーペアがない場合は、ここで作成します。

前提条件

- **OFFLINE_TOKEN** および **API_TOKEN** 変数を生成している。

手順

1. ターミナルで、root ユーザーから SSH 公開鍵を取得します。

```
$ cat /root/.ssh/id_rsa.pub
```

2. SSH 公開鍵を **CLUSTER_SSHKEY** 変数に設定します。

```
$ CLUSTER_SSHKEY=<downloaded_ssh_key>
```

3. **CLUSTER_SSHKEY** 変数の定義を確認します。

```
$ echo ${CLUSTER_SSHKEY}
```

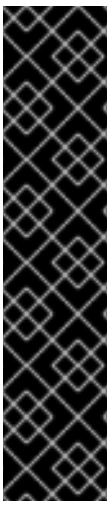
4.5. 新しいクラスターの登録

API を使用して新しいクラスター定義を登録するには、/v2/clusters エンドポイントを使用します。新しいクラスターを登録するには、次の設定が必要です。

- **name**
- **openshift-version**
- **pull_secret**
- **cpu_architecture**

新しいクラスターを登録するときに設定できるフィールドの詳細は、API ビューアー の **cluster-create-params** モデルを参照してください。**olm_operators** フィールドを設定する場合の Operator のインストールに関する詳細は、**関連情報** を参照してください。

クラスター定義を作成したら、クラスター定義を変更し、追加設定の値を指定できます。



重要

- 特定のインストールプラットフォームおよび OpenShift Container Platform バージョンでは、同じクラスター上で 2 つの異なるアーキテクチャーを組み合わせて、混合アーキテクチャークラスターを作成することもできます。詳細は、**関連情報** を参照してください。
- OpenShift Container Platform をサードパーティーのプラットフォームにインストールする場合は、関連する手順について **関連情報** を参照してください。
- 5 - 10 ノードのクラスターの場合、クラスターを登録するときに、ワーカーノードに加えてコントロールプレーンノードでワークロードを実行するようにスケジュールすることを選択できます。詳細は、**関連情報** の **スケジュール可能なコントロールプレーンノードの設定** を参照してください。

前提条件

- 有効な **API_TOKEN** を生成している。トークンは 15 分ごとに期限切れになります。
- プルシークレットをダウンロードしている。
- オプション: プルシークレットを **$PULL_SECRET** 変数に割り当て済みである。

手順

1. API トークンを更新します。

```
$ source refresh-token
```

2. 新しいクラスターを登録します。

   a. オプション: リクエストでプルシークレットファイルを一気に読み込むことで、新しいクラスターを登録できます。

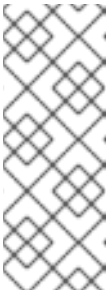
```
$ curl -s -X POST https://api.openshift.com/api/assisted-install/v2/clusters \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d "$(jq --null-input \
    --slurpfile pull_secret ~/Downloads/pull-secret.txt '
{
    "name": "testcluster",
    "openshift_version": "4.11",
    "cpu_architecture" : "<architecture_name>", 1
    "high_availability_mode": "<cluster_type>", 2
    "base_dns_domain": "example.com",
    "pull_secret": $pull_secret[0] | tojson
}
')" | jq '.id'
```

注記

1. 次の値のいずれかを使用します: **x86_64**、**arm64**、**ppc64le**、**s390x**、**multi**。混合アーキテクチャーのクラスターの場合、**multi** のみを使用します。
2. マルチノード OpenShift Container Platform クラスターを表すにはデフォルト値 **Full** を使用し、シングルノード OpenShift クラスターを表すには **None** を使用します。複数のマスターノード上に **高可用性** クラスターを **すべて** インストールし、インストールされたクラスターの可用性を保証します。**none** は、1 つのノードに完全なクラスターをインストールします。

   b. オプション: 設定を JSON ファイルに書き込み、それをリクエストで参照することにより、新しいクラスターを登録できます。

```
cat << EOF > cluster.json
{
  "name": "testcluster",
  "openshift_version": "4.11",
  "high_availability_mode": "<cluster_type>", 1
  "base_dns_domain": "example.com",
  "network_type": "examplenetwork",
  "cluster_network_cidr":"11.111.1.0/14"
  "cluster_network_host_prefix": 11,
  "service_network_cidr": "111.11.1.0/16",
  "api_vips":[{"ip": ""}],
  "ingress_vips": [{"ip": ""}],
  "vip_dhcp_allocation": false,
  "additional_ntp_source": "clock.redhat.com,clock2.redhat.com",
  "ssh_public_key": "$CLUSTER_SSHKEY",
  "pull_secret": $PULL_SECRET
}
EOF
```



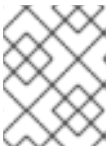
注記

1 マルチノード OpenShift Container Platform クラスターを表すにはデフォルト値 **Full** を使用し、シングルノード OpenShift クラスターを表すには **None** を使用します。複数のマスターノード上に **高可用性** クラスターを **すべて** インストールし、インストールされたクラスターの可用性を保証します。**none** は、1 つのノードに完全なクラスターをインストールします。

```
$ curl -s -X POST "https://api.openshift.com/api/assisted-install/v2/clusters" \
  -d @./cluster.json \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $API_TOKEN" \
  | jq '.id'
```

3. 返された **cluster_id** を **CLUSTER_ID** 変数に割り当て、エクスポートします。

```
$ export CLUSTER_ID=<cluster_id>
```



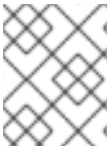
注記

ターミナルセッションを閉じる場合は、新しいターミナルセッションで **CLUSTER_ID** 変数を再度エクスポートする必要があります。

4. 新しいクラスターのステータスを確認します。

```
$ curl -s -X GET "https://api.openshift.com/api/assisted-install/v2/clusters/$CLUSTER_ID" \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $API_TOKEN" \
  | jq
```

新しいクラスター定義を登録したら、クラスターのインフラ環境を作成します。



注記

インフラストラクチャー環境を作成するまで、Assisted Installer ユーザーインターフェイスにクラスター設定を表示することはできません。

関連情報

- クラスターの変更
- 混合アーキテクチャークラスターのインストール
- オプション: Nutanix へのインストール
- オプション: vSphere へのインストール
- オプション: Oracle Cloud Infrastructure へのインストール

4.5.1. オプション: Operator のインストール

新しいクラスターを登録するときに、次の Operator をインストールできます。

- OpenShift Virtualization Operator

注記

現在、OpenShift Virtualization は IBM zSystems および IBM Power ではサポートされていません。

- multicluster engine Operator
- OpenShift Data Foundation Operator
- LVM Storage Operator

高度なオプションが必要な場合は、クラスターをインストールした後に Operator をインストールします。

手順

- 以下のコマンドを実行します。

```
$ curl -s -X POST https://api.openshift.com/api/assisted-install/v2/clusters \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d "$(jq --null-input \
   --slurpfile pull_secret ~/Downloads/pull-secret.txt '
{
   "name": "testcluster",
   "openshift_version": "4.15",
   "cpu_architecture" : "x86_64",
   "base_dns_domain": "example.com",
  "olm_operators": [
  { "name": "mce" } 1
  ,
  { "name": "odf" } 2
    ]
   "pull_secret": $pull_secret[0] | tojson
}
')" | jq '.id'
```

1 OpenShift Virtualization の場合は **cnv**、マルチクラスターエンジンの場合は **mce**、OpenShift Data Foundation の場合は **odf**、LVM ストレージの場合は **lvm** を指定します。

2 この例では、マルチノードクラスターに multicluster engine と OpenShift Data Foundation をインストールします。シングルノード OpenShift クラスターの場合は、**mce** と **lvm** を指定します。

関連情報

- OpenShift Virtualization ドキュメント
- Red Hat OpenShift Cluster Manager ドキュメント
- Red Hat OpenShift Data Foundation ドキュメント

- Logical Volume Manager Storage ドキュメント

4.6. クラスターの変更

API を使用してクラスター定義を変更するには、/v2/clusters/{cluster_id} エンドポイントを使用します。クラスターリソースの変更は、ネットワークの種類の変更やユーザー管理ネットワークの有効化などの設定を追加するための一般的な操作です。クラスター定義を変更するときに設定できるフィールドの詳細については、API ビューアー の **v2-cluster-update-params** モデルを参照してください。

すでに登録されているクラスターリソースに Operator を追加または削除できます。

注記

ノード上にパーティションを作成するには、OpenShift Container Platform ドキュメントの ノード上でのストレージの設定 を参照してください。

前提条件

- 新しいクラスターリソースを作成した。

手順

1. API トークンを更新します。

```
$ source refresh-token
```

2. クラスターを変更します。たとえば、SSH 鍵を変更します。

```
$ curl https://api.openshift.com/api/assisted-install/v2/clusters/${CLUSTER_ID} \
-X PATCH \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d '
{
    "ssh_public_key": "ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQDZrD4LMkAEeoU2vShhF8VM+cCZtVRgB7tqtsMx
ms2q3TOJZAgfuqReKYWm+OLOZTD+DO3Hn1pah/mU3u7uJfTUg4wEX0Le8zBu9xJVym0B
VmSFkzHfIJVTn6SfZ81NqcalisGWkpmkKXVCdnVAX6RsbHfpGKk9YPQarmRCn5KzkelJK4hrS
WpBPjdzkFXaIpf64JBZtew9XVYA3QeXkIcFuq7NBuUH9BonroPEmIXNOa41PUP1IWq3mERN
gzHZiuU8Ks/pFuU5HCMvv4qbTOIhiig7vidImHPpqYT/TCkuVi5w0ZZgkkBeLnxWxH0ldrfzgFBY
AxnpTU8Ih/4VhG538Ix1hxPaM6cXds2ic71mBbtbSrk+zjtNPaeYk1O7UpcCw4jjHspU/rVV/DY51
D5gSiiuaFPBMucnYPgUxy4FMBFfGrmGLIzTKiLzcz0DiSz1jBeTQOX++1nz+KDLBD8CPdi5k4d
q7lLkapRk85qdEvgaG5RlHMSPSS3wDrQ51fD8= user@hostname"
}
' | jq
```

4.6.1. Operator の変更

以前のインストールの一部としてすでに登録されているクラスターリソースから Operator を追加または削除できます。これは、OpenShift Container Platform のインストールを開始する前にのみ可能です。

/v2/clusters/{cluster_id} エンドポイントの PATCH メソッドを使用して、必要な Operator 定義を設定します。

前提条件

- API トークンを更新した。
- **CLUSTER_ID** を環境変数としてエクスポートした。

手順

- Operator を変更するには、次のコマンドを実行します。

```
$ curl https://api.openshift.com/api/assisted-install/v2/clusters/${CLUSTER_ID} \
-X PATCH \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d '
{
   "olm_operators": [{"name": "mce"}, {"name": "cnv"}], 1
}
' | jq '.id'
```

1 OpenShift Virtualization の場合は **cnv**、multicluster engine の場合は **mce**、Red Hat OpenShift Data Foundation の場合は **odf**、Logical Volume Manager Storage の場合は **lvm** を指定します。以前にインストールされた Operator を削除するには、これを値の一覧から除外します。以前にインストールされたすべての Operator を削除するには、空の配列 (**"olm_operators": []**) を指定します。

出力例

```
{
 <various cluster properties>,
 "monitored_operators": [
  {
   "cluster_id": "b5259f97-be09-430e-b5eb-d78420ee509a",
   "name": "console",
   "operator_type": "builtin",
   "status_updated_at": "0001-01-01T00:00:00.000Z",
   "timeout_seconds": 3600
  },
  {
   "cluster_id": "b5259f97-be09-430e-b5eb-d78420ee509a",
   "name": "cvo",
   "operator_type": "builtin",
   "status_updated_at": "0001-01-01T00:00:00.000Z",
   "timeout_seconds": 3600
  },
  {
   "cluster_id": "b5259f97-be09-430e-b5eb-d78420ee509a",
   "name": "mce",
   "namespace": "multicluster-engine",
   "operator_type": "olm",
   "status_updated_at": "0001-01-01T00:00:00.000Z",
   "subscription_name": "multicluster-engine",
   "timeout_seconds": 3600
  },
  {
```

```
    "cluster_id": "b5259f97-be09-430e-b5eb-d78420ee509a",
    "name": "cnv",
    "namespace": "openshift-cnv",
    "operator_type": "olm",
    "status_updated_at": "0001-01-01T00:00:00.000Z",
    "subscription_name": "hco-operatorhub",
    "timeout_seconds": 3600
  },
  {
    "cluster_id": "b5259f97-be09-430e-b5eb-d78420ee509a",
    "name": "lvm",
    "namespace": "openshift-local-storage",
    "operator_type": "olm",
    "status_updated_at": "0001-01-01T00:00:00.000Z",
    "subscription_name": "local-storage-operator",
    "timeout_seconds": 4200
  }
],
<more cluster properties>
```

注記

この出力は、新しいクラスターの状態の説明になります。出力の **monitored_operators** プロパティーには、次の 2 つのタイプの Operator が含まれます。

- **"operator_type": "builtin"**: このタイプの Operator は、OpenShift Container Platform の不可欠な部分です。
- **"Operator_type": "olm"**: このタイプの Operator は、ユーザーによって手動で追加されるか、依存関係として自動的に追加されます。この例では、LVM Storage Operator が OpenShift Virtualization の依存関係として自動的に追加されます。

4.7. 新しいインフラ環境の登録

Assisted Installer API を使用して新しいクラスター定義を登録したら、v2/infra-envs エンドポイントを使用してインフラストラクチャー環境を作成します。新しいインフラストラクチャー環境を登録するには、次の設定が必要です。

- **name**
- **pull_secret**
- **cpu_architecture**

新しいインフラストラクチャー環境を登録するときに設定できるフィールドの詳細は、API ビューアーの **infra-env-create-params** モデルを参照してください。インフラストラクチャー環境は、作成後に変更できます。ベストプラクティスとして、新しいインフラストラクチャー環境を作成するときに **cluster_id** を含めることを検討してください。**cluster_id** は、インフラストラクチャー環境をクラスター定義に関連付けます。新しいインフラストラクチャー環境を作成するとき、Assisted Installer は検出 ISO も生成します。

前提条件

- 有効な **API_TOKEN** を生成している。トークンは 15 分ごとに期限切れになります。
- プルシークレットをダウンロードしている。
- オプション: 新しいクラスター定義を登録し、**cluster_id** をエクスポートした。

手順

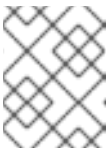
1. API トークンを更新します。

```
$ source refresh-token
```

2. 新しいインフラストラクチャー環境を登録します。できればクラスター名を含む名前を指定します。この例では、クラスター ID を提供して、インフラストラクチャー環境をクラスターリソースに関連付けます。次の例では、**image_type** を指定しています。**full-iso** または **minimum-** iso のいずれかを指定できます。デフォルト値は **minimal-iso** です。

   a. オプション: リクエストでプルシークレットファイルを丸呑みすることで、新しいインフラストラクチャー環境を登録できます。

```
$ curl https://api.openshift.com/api/assisted-install/v2/infra-envs \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d "$(jq --null-input \
  --slurpfile pull_secret ~/Downloads/pull-secret.txt \
  --arg cluster_id ${CLUSTER_ID} '
    {
      "name": "testcluster-infra-env",
      "image_type":"full-iso",
      "cluster_id": $cluster_id,
      "cpu_architecture" : "<architecture_name>", 1
      "pull_secret": $pull_secret[0] | tojson
    }
')" | jq '.id'
```



注記

1 有効な値を指定してくださ
い。**x86_64**、**arm64**、**ppc64le**、**s390x**、**multi** が有効です。

   b. オプション: 設定を JSON ファイルに書き込み、それを要求で参照することにより、新しいインフラストラクチャー環境を登録できます。

```
$ cat << EOF > infra-envs.json
{
 "name": "testcluster",
 "pull_secret": $PULL_SECRET,
 "proxy": {
    "http_proxy": "",
    "https_proxy": "",
    "no_proxy": ""
 },
 "ssh_authorized_key": "$CLUSTER_SSHKEY",
 "image_type": "full-iso",
```

```
  "cluster_id": "${CLUSTER_ID}",
  "openshift_version": "4.11"
}
EOF
```

```
$ curl -s -X POST "https://api.openshift.com/api/assisted-install/v2/infra-envs"
 -d @./infra-envs.json
 -H "Content-Type: application/json"
 -H "Authorization: Bearer $API_TOKEN"
 | jq '.id'
```

3. 返された **ID** を **INFRA_ENV_ID** 変数に割り当て、エクスポートします。

```
$ export INFRA_ENV_ID=<id>
```

注記

インフラストラクチャー環境を作成し、**cluster_id** を介してクラスター定義に関連付けると、Assisted Installer Web ユーザーインターフェイスでクラスター設定を確認できます。ターミナルセッションを閉じる場合は、新しいターミナルセッションで **ID** を再エクスポートする必要があります。

4.8. インフラストラクチャー環境の変更

/v2/infra-envs/{infra_env_id} エンドポイントを使用してインフラストラクチャー環境を変更できます。インフラストラクチャー環境の変更は、ネットワーク、SSH キー、イグニッション設定のオーバーライドなどの設定を追加するための一般的な操作です。

インフラストラクチャー環境を変更するときに設定できるフィールドの詳細については、API ビューアー の **infra-env-update-params** モデルを参照してください。新しいインフラストラクチャー環境を変更する場合、Assisted Installer は検出 ISO も再生成します。

前提条件

- 新しいインフラストラクチャー環境が作成された。

手順

1. API トークンを更新します。

```
$ source refresh-token
```

2. インフラストラクチャー環境を変更します。

```
$ curl https://api.openshift.com/api/assisted-install/v2/infra-envs/${INFRA_ENV_ID} \
-X PATCH \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d "$(jq --null-input \
  --slurpfile pull_secret ~/Downloads/pull-secret.txt '
    {
      "image_type":"minimal-iso",
```

```
      "pull_secret": $pull_secret[0] | tojson
    }
')" | jq
```

4.8.1. オプション: カーネル引数の追加

Assisted Installer を介してカーネル引数を Red Hat Enterprise Linux CoreOS (RHCOS)カーネルに指定すると、特に検出 ISO のカーネルパラメーターをカスタマイズできない場合に、特定のパラメーターまたはオプションをカーネルに渡します。カーネルパラメーターは、ハードウェアの対話、システムのパフォーマンス、および機能に影響を与えるカーネルの動作とオペレーティングシステムの設定のさまざまな側面を制御できます。カーネル引数は、ハードウェア設定、デバッグ設定、システムサービス、およびその他の低レベルの設定についてノードの RHCOS カーネルをカスタマイズまたは通知するために使用されます。

RHCOS インストーラーの **kargs modify** コマンドは、**append**、**delte**、および **replace** のオプションをサポートします。

/v2/infra-envs/{infra_env_id} エンドポイントを使用してインフラストラクチャー環境を変更できます。新しいインフラストラクチャー環境を変更する場合、Assisted Installer は検出 ISO も再生成します。

手順

1. API トークンを更新します。

```
$ source refresh-token
```

2. カーネル引数を変更します。

```
$ curl https://api.openshift.com/api/assisted-install/v2/infra-envs/${INFRA_ENV_ID} \
-X PATCH \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d "$(jq --null-input \
  --slurpfile pull_secret ~/Downloads/pull-secret.txt '
    {
      "kernel_arguments": [{ "operation": "append", "value": "<karg>=<value>" }], (1)
      "image_type":"minimal-iso",
      "pull_secret": $pull_secret[0] | tojson
    }
')" | jq
```

(1) **<karg>** は、カーネル引数に、**<value>** はカーネル引数の値に置き換えます。例: **rd.net.timeout.carrier=60**。カーネル引数ごとに JSON オブジェクトを追加することで、複数のカーネル引数を指定できます。

4.9. ホストの追加

クラスターリソースとインフラストラクチャー環境を設定したら、検出 ISO イメージをダウンロードします。次の 2 つのイメージから選択できます。

- **完全な ISO イメージ**: ブートを自己完結型にする必要がある場合は、完全な ISO イメージを使用します。このイメージには、Assisted Installer エージェントを起動して開始するために必要

なすべてが含まれています。ISO イメージのサイズは約 1GB です。これは、RHEL KVM を使用してインストールする場合の **s390x** アーキテクチャーで推奨の方法です。

- **最小限の ISO イメージ:** 仮想メディア接続の帯域幅が制限されている場合は、最小限の ISO イメージを使用します。これはデフォルト設定です。このイメージには、ネットワークを使用してホストを起動するために必要なものだけが含まれています。コンテンツの大部分は、起動時にダウンロードされます。ISO イメージのサイズは約 100MB です。

注記

現在、ISO イメージは、z/VM を使用した IBM Z (**s390x**) へのインストールではサポートされていません。詳細は、iPXE を使用したホストの起動 を参照してください。

3 つの方法を使用して、検出イメージでホストを起動できます。詳細は、検出イメージを使用したホストの起動 を参照してください。

前提条件

- クラスターが作成済みである。
- インフラストラクチャー環境を作成した。
- 設定が完了した。
- クラスターホストがプロキシーの使用を必要とするファイアウォールの背後にある場合、プロキシーサーバーの HTTP および HTTPS URL のユーザー名、パスワード、IP アドレス、およびポートを設定済みである。
- イメージタイプを選択したか、デフォルトの **minimum-iso** を使用します。

手順

1. 必要に応じて検出イメージを設定します。詳細は、検出イメージの設定 を参照してください。
2. API トークンを更新します。

```
$ source refresh-token
```

3. ダウンロード URL を取得します。

```
$ curl -H "Authorization: Bearer ${API_TOKEN}" \
https://api.openshift.com/api/assisted-install/v2/infra-envs/${INFRA_ENV_ID}/downloads/image-url
```

出力例

```
{
  "expires_at": "2024-02-07T20:20:23.000Z",
  "url": "https://api.openshift.com/api/assisted-images/bytoken/<TOKEN>/<OCP_VERSION>/<CPU_ARCHITECTURE>/<FULL_OR_MINIMAL_IMAGE>.iso"
}
```

4. 検出イメージをダウンロードします。

```
$ wget -O discovery.iso <url>
```

<url> を前の手順のダウンロード URL に置き換えます。

5. 検出イメージを使用してホストを起動します。
6. ホストにロールを割り当てます。

関連情報

- 検出イメージの設定
- 検出イメージを使用したホストの起動
- API を使用した Nutanix へのホストの追加
- vSphere へのホストの追加
- ホストへのロールの割り当て
- iPXE を使用したホストの起動

4.10. ホストの変更

ホストを追加したら、必要に応じてホストを変更します。最も一般的な変更は、**host_name** および **host_role** パラメーターに対するものです。

/v2/infra-envs/{infra_env_id}/hosts/{host_id} エンドポイントを使用してホストを変更できます。ホストの変更時に設定できるフィールドの詳細は、API ビューアー の **host-update-params** モデルを参照してください。

ホストは、次の 2 つのロールのいずれかになります。

- **master**: **マスター** ロールを持つホストは、コントロールプレーンホストとして動作します。
- **worker**: **worker** ロールを持つホストは、ワーカーホストとして動作します。

デフォルトでは、Assisted Installer はホストを **auto-assign** に設定します。これは、ホストが **master** ロールか **worker** ロールかをインストールプログラムが自動的に判断することを意味します。以下の手順を使用して、ホストのロールを設定します。

前提条件

- ホストをクラスターに追加した。

手順

1. API トークンを更新します。

```
$ source refresh-token
```

2. ホスト ID を取得します。

```
$ curl -s -X GET "https://api.openshift.com/api/assisted-install/v2/clusters/$CLUSTER_ID" \
--header "Content-Type: application/json" \
```

```
  -H "Authorization: Bearer $API_TOKEN" \
| jq '.host_networks[].host_ids'
```

3. ホストを変更します。

```
$ curl https://api.openshift.com/api/assisted-install/v2/infra-
envs/${INFRA_ENV_ID}/hosts/<host_id> \ 1
-X PATCH \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d '
   {
     "host_role":"worker"
     "host_name" : "worker-1"
   }
' | jq
```

1 **<host_id>** をホストの ID に置き換えます。

4.10.1. ストレージディスク設定の変更

ホスト検出中に取得された各ホストは、複数のストレージディスクを指定できます。オプションで、各ディスクのデフォルト設定を変更できます。

前提条件

- クラスターを設定し、ホストを検出している。詳細は、**関連情報** を参照してください。

ストレージディスクの表示
クラスター内のホストおよび各ホストのディスクを表示できます。これにより、特定のディスクに対してアクションを実行できます。

手順

1. API トークンを更新します。

```
$ source refresh-token
```

2. クラスターのホスト ID を取得します。

```
$ curl -s "https://api.openshift.com/api/assisted-install/v2/clusters/$CLUSTER_ID" \
  -H "Authorization: Bearer $API_TOKEN" \
| jq '.host_networks[].host_ids'
```

出力例

```
$ "1022623e-7689-8b2d-7fbd-e6f4d5bb28e5"
```

注記

これは、単一ホストの ID です。複数のホスト ID はコンマで区切られます。

3. 特定のホストのディスクを取得します。

```
$ curl https://api.openshift.com/api/assisted-install/v2/infra-
envs/${INFRA_ENV_ID}/hosts/<host_id> \ 1
-H "Authorization: Bearer ${API_TOKEN}" \
| jq '.inventory | fromjson | .disks'
```

1 **<host_id>** は、該当するホストの ID に置き換えます。

出力例

```
$ [
  {
    "by_id": "/dev/disk/by-id/wwn-0x6c81f660f98afb002d3adc1a1460a506",
    "by_path": "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:0:0",
    "drive_type": "HDD",
    "has_uuid": true,
    "hctl": "1:2:0:0",
    "id": "/dev/disk/by-id/wwn-0x6c81f660f98afb002d3adc1a1460a506",
    "installation_eligibility": {
      "eligible": true,
      "not_eligible_reasons": null
    },
    "model": "PERC_H710P",
    "name": "sda",
    "path": "/dev/sda",
    "serial": "0006a560141adc3a2d00fb8af960f681",
    "size_bytes": 6595056500736,
    "vendor": "DELL",
    "wwn": "0x6c81f660f98afb002d3adc1a1460a506"
  }
]
```

注記

これは、1 つのディスクの出力です。これには、ディスクの **disk_id** および **installation_eligibility** プロパティーが含まれます。

インストールディスクの変更

Assisted Installer は、デフォルトでインストールディスクをランダムに割り当てます。ホストに複数のストレージディスクがある場合は、別のディスクを選択してインストールディスクとして機能させることができます。これにより、以前のディスクは自動的に割り当てが解除されます。

Installation_eligibility プロパティーが **eligible: true** のディスクを選択して、インストールディスクとして指定できます。

手順

1. ホストおよびストレージディスク ID を取得します。詳細は、**ストレージディスクの表示** を参照してください。

2. オプション: 現在のインストールディスクを特定します。

```
$ curl https://api.openshift.com/api/assisted-install/v2/infra-
envs/${INFRA_ENV_ID}/hosts/<host_id> \ 1
-H "Authorization: Bearer ${API_TOKEN}" \
| jq '.installation_disk_id'
```

1 **<host_id>** は、該当するホストの ID に置き換えます。

3. 新規インストールディスクを割り当てます。

```
$ curl https://api.openshift.com/api/assisted-install/v2/infra-
envs/${INFRA_ENV_ID}/hosts/<host_id> \ 1
-X PATCH \
-H "Content-Type: application/json" \
-H "Authorization: Bearer ${API_TOKEN}" \

{
  "disks_selected_config": [
    {
      "id": "<disk_id>", 2
      "role": "install"
    }
  ]
}
```

注記

1 **<host_id>** をホストの ID に置き換えます。
2 &lt **;disk_id&gt**; を新規インストールディスクの ID に置き換えます。

ディスクフォーマットの無効化
Assisted Installer は、インストールディスクとして定義されているかどうかに関係なく、デフォルトでインストールプロセス中にフォーマットするためにすべての起動可能なディスクをマークします。フォーマットすると、データが失われます。

特定のディスクのフォーマットを無効にすることもできます。これは、ブート可能なディスクが、主に起動順序の観点からインストールプロセスを干渉する可能性があるため、注意して実行する必要があります。

インストールディスクのフォーマットを無効にできません。

手順

1. ホストおよびストレージディスク ID を取得します。詳細は、**ストレージディスクの表示** を参照してください。

2. 以下のコマンドを実行します。

```
$  curl https://api.openshift.com/api/assisted-install/v2/infra-
envs/${INFRA_ENV_ID}/hosts/<host_id> \ 1
-X PATCH \
-H "Content-Type: application/json" \
-H "Authorization: Bearer ${API_TOKEN}" \
```

```
{
 "disks_skip_formatting": [
  {
   "disk_id": "<disk_id>", (2)
   "skip_formatting": true (3)
  }
 ]
}
```

注記

(1) **<host_id>** をホストの ID に置き換えます。
(2) **<disk_id>** は、ディスクの ID に置き換えます。複数のディスクがある場合は、ID をコンマで区切ります。
(3) フォーマットを再度有効にするには、値を **false** に変更します。

4.11. カスタムリポジトリーの追加

カスタムマニフェストは、Assisted Installer のユーザーインターフェイスで現在サポートされていない高度な設定が含まれる JSON または YAML ファイルです。カスタムマニフェストを作成することも、サードパーティーが提供するマニフェストを使用することもできます。API でカスタムマニフェストを作成するには、/v2/clusters/$CLUSTER_ID/manifests エンドポイントを使用します。

base64 でエンコードされたカスタムマニフェストを、**openshift** フォルダーまたは Assisted Installer API を使用して **manifests** フォルダーにアップロードすることができます。許可されるカスタムのマニフェストの数に制限はありません。

一度にアップロードできる base64 でエンコードされた JSON マニフェストは 1 つだけです。ただし、アップロードされた各 base64 でエンコードされた YAML ファイルには、複数のカスタムマニフェストを含めることができます。マルチドキュメントの YAML マニフェストのアップロードは、YAML ファイルを個別に追加するよりも高速です。

単一のカスタムマニフェストを含むファイルの場合は、使用可能なファイル拡張は、**.yaml**、**.yml**、または **.json** などです。

単一のカスタムマニフェストの例

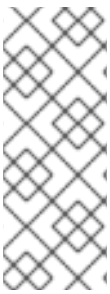
```
{
    "apiVersion": "machineconfiguration.openshift.io/v1",
    "kind": "MachineConfig",
    "metadata": {
        "labels": {
            "machineconfiguration.openshift.io/role": "primary"
        },
        "name": "10_primary_storage_config"
    },
    "spec": {
        "config": {
            "ignition": {
                "version": "3.2.0"
            },
            "storage": {
                "disks": [
                    {
```

```
                "device": "</dev/xxyN>",
                "partitions": [
                  {
                    "label": "recovery",
                    "startMiB": 32768,
                    "sizeMiB": 16384
                  }
                ]
              }
            ],
            "filesystems": [
              {
                "device": "/dev/disk/by-partlabel/recovery",
                "label": "recovery",
                "format": "xfs"
              }
            ]
          }
        }
    }
}
```

複数のカスタムマニフェストを含むファイルの場合、使用可能なファイルタイプは、**.yaml** または **.yml** などです。

複数のカスタムマニフェストの例

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: master
  name: 99-openshift-machineconfig-master-kargs
spec:
  kernelArguments:
    - loglevel=7
---
apiVersion: machineconfiguration.openshift.io/v2
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-openshift-machineconfig-worker-kargs
spec:
  kernelArguments:
    - loglevel=5
```



注記

- OpenShift Container Platform を Oracle Cloud Infrastructure (OCI) 外部プラットフォームにインストールする場合は、Oracle が提供するカスタムマニフェストを追加する必要があります。vSphere や Nutanix などの追加の外部パートナー統合の場合、この手順はオプションです。
- カスタムマニフェストの詳細は、**関連情報** を参照してください。

前提条件

- 有効な **API_TOKEN** を生成している。トークンは 15 分ごとに期限切れになります。
- 新しいクラスター定義を登録し、**cluster_id** を **$CLUSTER_ID** BASH 変数にエクスポートしている。

手順

1. カスタムマニフェストファイルを作成します。
2. ファイル形式に適した拡張子を使用して、カスタムマニフェストファイルを保存します。
3. API トークンを更新します。

```
$ source refresh-token
```

4. 以下のコマンドを実行して、カスタムマニフェストをクラスターに追加します。

```
$ curl -X POST "https://api.openshift.com/api/assisted-
install/v2/clusters/$CLUSTER_ID/manifests" \
    -H "Authorization: Bearer $API_TOKEN" \
    -H "Content-Type: application/json" \
    -d '{
          "file_name":"manifest.json",
          "folder":"manifests",
          "content":"""$(base64 -w 0 ~/manifest.json)"""
    }' | jq
```

manifest.json はマニフェストファイルの名前に置き換えます。**manifest.json** の 2 番目のインスタンスは、ファイルへのパスです。パスが正しいことを確認します。

出力例

```
{
  "file_name": "manifest.json",
  "folder": "manifests"
}
```

注記

base64 -w 0 コマンドは、マニフェストを文字列として base64 エンコードし、キャリッジリターンを返します。キャリッジリターンを含むエンコーディングは例外を生成します。

5. Assisted Installer がマニフェストを追加したことを確認します。

```
curl -X GET "https://api.openshift.com/api/assisted-
install/v2/clusters/$CLUSTER_ID/manifests/files?folder=manifests&file_name=manifest.json"
-H "Authorization: Bearer $API_TOKEN"
```

manifest.json はマニフェストファイルの名前に置き換えます。

関連情報

- マニフェスト設定ファイル
- Multi-document YAML files

4.12. インストール前の検証

Assisted Installer は、インストール前にクラスターが前提条件を満たしていることを確認します。確認することで、インストール後の複雑なトラブルシューティングが不要になり、時間と労力が大幅に節約されます。クラスターをインストールする前に、クラスターと各ホストがインストール前の検証に合格していることを確認してください。

関連情報

- インストール前の検証

4.13. クラスターのインストール

クラスターホストの検証が完了したら、クラスターをインストールできます。

前提条件

- クラスターとインフラストラクチャー環境を作成しました。
- インフラストラクチャー環境にホストを追加しました。
- ホストはバリデーションにパスしました。

手順

1. API トークンを更新します。

```
$ source refresh-token
```

2. クラスターをインストールします。

```
$ curl -H "Authorization: Bearer $API_TOKEN" \
-X POST \
https://api.openshift.com/api/assisted-install/v2/clusters/$CLUSTER_ID/actions/install | jq
```

3. インストール後のプラットフォーム統合手順を完了します。

関連情報

- Nutanix のインストール後の設定
- vSphere のインストール後の設定

第5章 オプション: ディスク暗号化の有効化

TPM v2 または Tang 暗号化モードを使用して、インストールディスクの暗号化を有効にすることができます。

注記

状況によっては、ベアメタルホストのファームウェアで TPM ディスク暗号化を有効にし、Assisted Installer で生成した ISO から起動すると、クラスターのデプロイメントが停止することがあります。これは、ホスト上の以前のインストールからの TPM 暗号化キーが残っている場合に発生する可能性があります。詳細は、BZ#2011634 を参照してください。この問題が発生した場合は、Red Hat サポートに連絡してください。

5.1. TPM V2 暗号化の有効化

前提条件

- 各ホストの BIOS で TPM v2 暗号化が有効になっているかどうかを確認します。ほとんどの Dell システムでこれが必要です。コンピューターのマニュアルを確認してください。Assisted Installer は、ファームウェアで TPM が有効になっていることも検証します。詳細は、Assisted Installer API の **disk-encruption** モデルを参照してください。

重要

TPM v2 暗号化チップが各ノードにインストールされ、ファームウェアで有効になっていることを確認します。

手順

1. オプション: UI を使用して、ユーザーインターフェイスウィザードの **クラスターの詳細** ステップで、コントロールプレーンノード、ワーカー、またはその両方で TPM v2 暗号化を有効にすることを選択します。

2. オプション: API を使用して、ホストの変更手順に従います。**disk_encryption.enable_on** 設定を **all**、**masters**、または **worker** に設定します。**disk_encryption.mode** 設定を **tpmv2** に設定します。

   a. API トークンを更新します。

   ```
   $ source refresh-token
   ```

   b. TPM v2 暗号化を有効にします。

   ```
   $ curl https://api.openshift.com/api/assisted-install/v2/clusters/${CLUSTER_ID} \
   -X PATCH \
   -H "Authorization: Bearer ${API_TOKEN}" \
   -H "Content-Type: application/json" \
   -d '
   {
     "disk_encryption": {
       "enable_on": "none",
       "mode": "tpmv2"
   ```

```
  }
}
' | jq
```

enable_on の有効な設定は、**all**、**master**、**worker**、または **none** です。

5.2. TANG 暗号化を有効にする

前提条件

- Tang 交換キーのサムプリントの生成に使用できる Red Hat Enterprise Linux (RHEL) 8 マシンにアクセスできる。

手順

1. Tang サーバーを設定するか、既存のサーバーにアクセスします。手順については、NBDE (Network-Bound Disk Encryption) を参照してください。複数の Tang サーバーを設定できますが、Assisted Installer はインストール中にすべてのサーバーに接続できる必要があります。
2. Tang サーバーで、**tang-show-keys** を使用して Tang サーバーのサムプリントを取得します。

```
$ tang-show-keys <port>
```

オプション: **<port>** ポート番号に置き換えます。デフォルトのポート番号は **80** です。

サムプリントの例

```
1gYTN_LpU9ZMB35yn5IbADY5OQ0
```

3. オプション: **jose** を使用して Tang サーバーのサムプリントを取得します。
   a. **jose** が Tang サーバーにインストールされていることを確認します。

```
$ sudo dnf install jose
```

   b. Tang サーバーで、**jose** を使用してサムプリントを取得します。

```
$ sudo jose jwk thp -i /var/db/tang/<public_key>.jwk
```

<public_key> を Tang サーバーの公開交換キーに置き換えます。

サムプリントの例

```
1gYTN_LpU9ZMB35yn5IbADY5OQ0
```

4. オプション: ユーザーインターフェイスウィザードの **クラスターの詳細** ステップで、コントロールプレーンノード、ワーカー、またはその両方で Tang 暗号化を有効にすることを選択します。Tang サーバーの URL と拇印を入力する必要があります。
5. オプション: API を使用して、ホストの変更手順に従います。
   a. API トークンを更新します。

```
-
```

```
$ source refresh-token
```

b. **disk_encryption.enable_on** 設定を **all**、**masters**、または **worker** に設定します。**disk_encryption.mode** 設定を **tang** に設定します。**disk_encyrption.tang_servers** を設定して、1 つ以上の Tang サーバーに関する URL と拇印の詳細を提供します。

```
$ curl https://api.openshift.com/api/assisted-install/v2/clusters/${CLUSTER_ID} \
-X PATCH \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d '
{
  "disk_encryption": {
    "enable_on": "all",
    "mode": "tang",
    "tang_servers": "
[{\"url\":\"http://tang.example.com:7500\",\"thumbprint\":\"PLjNyRdGw03zlRoGjQYMahSZGu9\"},
{\"url\":\"http://tang2.example.com:7500\",\"thumbprint\":\"XYjNyRdGw03zlRoGjQYMahSZGu3\"}]"
  }
}
' | jq
```

enable_on の有効な設定は、**all**、**master**、**worker**、または **none** です。**tang_servers** 値内で、オブジェクト内の引用符をコメントアウトします。

5.3. 関連情報

- ホストの変更

第6章 オプション: スケジュール可能なコントロールプレーンノードの設定

高可用性のデプロイメントでは、3 つ以上のノードがコントロールプレーンを設定します。コントロールプレーンノードは、OpenShift Container Platform の管理と OpenShift コンテナーの実行に使用されます。残りのノードはワーカーであり、顧客のコンテナーとワークロードを実行するために使用されます。ワーカーノードの数は 1 から数千までの範囲になります。

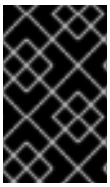
単一ノードの OpenShift クラスター、または最大 4 つのノードで設定されるクラスターの場合、システムはコントロールプレーンノードで実行されるワークロードを自動的にスケジュールします。

5 - 10 ノードのクラスターの場合、ワーカーノードに加えてコントロールプレーンノードでワークロードを実行するようにスケジュール設定できます。このオプションは、効率を高め、十分に活用されていないリソースがないようにする場合に推奨されます。このオプションは、インストールのセットアップ時またはインストール後の手順の一部として選択できます。

10 ノードを超える大規模なクラスターの場合、このオプションは推奨されません。

このセクションでは、インストールセットアップの一環として、Assisted Installer UI と API を使用して、コントロールプレーンノードで実行されるワークロードをスケジュールする方法について説明します。

インストール後にスケジュール可能なコントロールプレーンノードを設定する方法は、OpenShift Container Platform ドキュメント の コントロールプレーンノードをスケジュール可能として設定 を参照してください。



重要

コントロールプレーンノードをデフォルトのスケジュール不可からスケジュール可に設定するには、追加のサブスクリプションが必要です。これは、コントロールプレーンノードがワーカーノードになるためです。

6.1. WEB コンソールを使用したスケジュール可能なコントロールプレーンの設定

前提条件

- クラスターの詳細を設定している。
- OpenShift Container Platform 4.14 以降がインストールされている。

手順

1. Red Hat Hybrid Cloud Console にログインし、Assisted Installer UI を使用して OpenShift Container Platform をインストールするための手順に従います。詳細については、**関連情報 のアシステッドインストーラー Web コンソールを使用したインストール** を参照してください。
2. **Host discovery** ページに到達したら、**Add hosts** をクリックします。
3. 必要に応じて、**Provisioning type** と追加設定を変更します。すべてのオプションは、スケジュール可能なコントロールプレーンと互換性があります。
4. ISO をダウンロードするには、**Generate Discovery ISO** をクリックします。

5. **コントロールプレーンノードで Run workloads を on に設定し**ます。

注記

ノードが 4 つ以下の場合、このスイッチは自動的に有効になり、変更することはできません。

6. **Next** をクリックします。

6.2. API を使用したスケジュール可能なコントロールプレーンの設定

schedulable_masters 属性を使用して、コントロールプレーンノードでワークロードを実行できるようにします。

前提条件

- 有効な **API_TOKEN** を生成している。トークンは 15 分ごとに期限切れになる。
- **$PULL_SECRET** 変数が作成されている。
- OpenShift Container Platform 4.14 以降がインストールされている。

手順

1. Assisted Installer API を使用して Assisted Installer をインストールするための手順に従います。詳細は、**関連情報** の **Assisted Installer API を使用したインストール** を参照してください。
2. 新しいクラスターを登録する手順に到達したら、**schedulable_masters** 属性を次のように設定します。

```
$ curl https://api.openshift.com/api/assisted-install/v2/clusters/${CLUSTER_ID} \
-X PATCH \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d '
{
  "schedulable_masters": true (1)
}
' | jq
```

(1) コントロールプレーンノードでのワークロードのスケジューリングを有効にします。

6.3. 関連情報

- アシステッドインストーラー Web コンソールを使用したインストール
- Assisted Installer API を使用したインストール

第7章 検出イメージの設定

Assisted Installer は初期イメージを使用して、OpenShift Container Platform のインストールを試行する前にハードウェアおよびネットワークの検証を実行するエージェントを実行します。Ignition を使用して、検出イメージをカスタマイズできます。

注記

検出イメージへの変更は、システムに保持されません。

7.1. IGNITION 設定ファイルの作成

Ignition は低レベルのシステム設定ユーティリティーであり、一時的な初期ルートファイルシステムである **initramfs** の一部です。最初の起動時に Ignition が実行されると、Ignition 設定ファイルで設定データが検出され、**switch_root** が呼び出されてホストのルートファイルシステムにピボットされる前に、それがホストに適用されます。

Ignition は、JSON 設定仕様 ファイルを使用して、最初の起動時に発生する一連の変更を表します。

重要

3.2 より新しいバージョンの Ignition はサポートされておらず、エラーが発生します。

手順

1. Ignition ファイルを作成し、設定仕様のバージョンを指定します。

```
$ vim ~/ignition.conf
```

```
{
  "ignition": { "version": "3.1.0" }
}
```

2. 設定データを Ignition ファイルに追加します。たとえば、**core** ユーザーにパスワードを追加します。

   a. パスワードハッシュを生成します。

```
$ openssl passwd -6
```

   b. 生成されたパスワードハッシュを **core** ユーザーに追加します。

```
{
  "ignition": { "version": "3.1.0" },
  "passwd": {
    "users": [
      {
        "name": "core",
        "passwordHash":
"$6$spam$M5LGSMGyVD.9XOboxcwrsnwNdF4irpJdAWy.1Ry55syyUiUssIzIAHaOrUHr2z
g6ruD8YNBPW9kW0H8EnKXyc1"
      }
```

```
    ]
  }
}
```

3. Ignition ファイルを保存し、**IGNITION_FILE** 変数にエクスポートします。

```
$ export IGNITION_FILE=~/ignition.conf
```

7.2. IGNITION を使用した検出イメージの変更

Ignition 設定ファイルを作成したら、Assisted Installer API を使用してインフラストラクチャー環境にパッチを適用することにより、検出イメージを変更できます。

前提条件

- Web コンソールを使用してクラスターを作成した場合は、API 認証が設定されています。
- インフラストラクチャー環境があり、インフラストラクチャー環境 **ID** を **INFRA_ENV_ID** 変数にエクスポートしました。
- 有効な Ignition ファイルがあり、ファイル名を **$IGNITION_FILE** としてエクスポートしました。

手順

1. **lightning_config_override** JSON オブジェクトを作成し、ファイルにリダイレクトします。

```
$ jq -n \
  --arg IGNITION "$(jq -c . $IGNITION_FILE)" \
  '{ignition_config_override: $IGNITION}' \
  > discovery_ignition.json
```

2. API トークンを更新します。

```
$ source refresh-token
```

3. インフラストラクチャー環境にパッチを適用します。

```
$ curl \
  --header "Authorization: Bearer $API_TOKEN" \
  --header "Content-Type: application/json" \
  -XPATCH \
  -d @discovery_ignition.json \
  https://api.openshift.com/api/assisted-install/v2/infra-envs/$INFRA_ENV_ID | jq
```

lightning_config_override オブジェクトは、Ignition ファイルを参照します。

4. 更新された検出イメージをダウンロードします。

第8章 検出イメージを使用したホストの起動

Assisted Installer は初期イメージを使用して、OpenShift Container Platform のインストールを試行する前にハードウェアおよびネットワークの検証を実行するエージェントを実行します。次の 3 つの方法を使用して、検出イメージでホストを起動できます。

- USB ドライブ
- Redfish 仮想メディア
- iPXE

8.1. USB ドライブに ISO イメージを作成する

検出 ISO イメージを含む USB ドライブを使用して、Assisted Installer エージェントをインストールできます。USB ドライブを使用してホストを起動すると、ソフトウェアをインストールするためのホストの準備が整います。

手順

1. 管理ホストで、USB ドライブを USB ポートに挿入します。
2. ISO イメージを USB ドライブにコピーします。次に例を示します。

```
# dd if=<path_to_iso> of=<path_to_usb> status=progress
```

ここでは、以下のようになります。

<path_to_iso>
ダウンロードした検出 ISO ファイルへの相対パスです (たとえば、**discovery.iso**)。

<path_to_usb>
/dev/sdb など、接続された USB ドライブの場所です。
ISO が USB ドライブにコピーされたら、USB ドライブを使用してクラスターホストに Assisted Installer エージェントをインストールできます。

8.2. USB ドライブでの起動

起動可能な USB ドライブを使用して Assisted Installer にノードを登録するには、次の手順を使用します。

手順

1. RHCOS ディスカバリー ISO USB ドライブをターゲットホストに挿入します。
2. サーバーのファームウェア設定で起動ドライブの順序を設定し、アタッチされた検出 ISO から起動して、サーバーを再起動します。
3. ホストが起動するまで待ちます。
   a. Web コンソールのインストールの場合、管理ホストでブラウザーに戻ります。ホストが、検出されたホストのリストに表示されるまで待ちます。

b. API インストールの場合、トークンを更新し、有効なホスト数を確認して、ホスト ID を収集します。

```
$ source refresh-token
```

```
$ curl -s -X GET "https://api.openshift.com/api/assisted-
install/v2/clusters/$CLUSTER_ID" \
--header "Content-Type: application/json" \
  -H "Authorization: Bearer $API_TOKEN" \
| jq '.enabled_host_count'
```

```
$ curl -s -X GET "https://api.openshift.com/api/assisted-
install/v2/clusters/$CLUSTER_ID" \
--header "Content-Type: application/json" \
  -H "Authorization: Bearer $API_TOKEN" \
| jq '.host_networks[].host_ids'
```

出力例

```
[
  "1062663e-7989-8b2d-7fbb-e6f4d5bb28e5"
]
```

8.3. REDFISH API を使用した HTTP ホスト ISO イメージからの起動

Redfish Baseboard Management Controller (BMC) API を使用してインストールした ISO を使用して、ネットワーク内のホストをプロビジョニングできます。

前提条件

- インストール Red Hat Enterprise Linux CoreOS (RHCOS) ISO をダウンロードしている。

手順

1. ネットワークでアクセス可能な HTTP サーバーに ISO ファイルをコピーします。
2. ホストされている ISO ファイルからホストを起動します。以下に例を示します。

   a. 次のコマンドを実行して、redfish API を呼び出し、ホストされている ISO を **VirtualMedia** ブートメディアとして設定します。

```
$ curl -k -u <bmc_username>:<bmc_password> \
-d '{"Image":"<hosted_iso_file>", "Inserted": true}' \
-H "Content-Type: application/json" \
-X POST
<host_bmc_address>/redfish/v1/Managers/iDRAC.Embedded.1/VirtualMedia/CD/Actions/Vi
rtualMedia.InsertMedia
```

   詳細は以下のようになります。

   <bmc_username>:<bmc_password>

   ターゲットホスト BMC のユーザー名とパスワードです。

<hosted_iso_file>

ホストされたインストール ISO の URL です (例: **https://example.com/rhcos-live-minimal.iso**)。ISO は、ターゲットホストマシンからアクセスできる必要があります。

<host_bmc_address>

ターゲットホストマシンの BMC IP アドレスです。

b. 次のコマンドを実行して、**VirtualMedia** デバイスから起動するようにホストを設定します。

```
$ curl -k -u <bmc_username>:<bmc_password> \
-X PATCH -H 'Content-Type: application/json' \
-d '{"Boot": {"BootSourceOverrideTarget": "Cd", "BootSourceOverrideMode": "UEFI",
"BootSourceOverrideEnabled": "Once"}}' \
<host_bmc_address>/redfish/v1/Systems/System.Embedded.1
```

c. ホストを再起動します。

```
$ curl -k -u <bmc_username>:<bmc_password> \
-d '{"ResetType": "ForceRestart"}' \
-H 'Content-type: application/json' \
-X POST
<host_bmc_address>/redfish/v1/Systems/System.Embedded.1/Actions/ComputerSystem.Reset
```

d. オプション: ホストの電源がオフになっている場合は、**{"ResetType": "On"}** スイッチを使用して起動できます。以下のコマンドを実行します。

```
$ curl -k -u <bmc_username>:<bmc_password> \
-d '{"ResetType": "On"}' -H 'Content-type: application/json' \
-X POST
<host_bmc_address>/redfish/v1/Systems/System.Embedded.1/Actions/ComputerSystem.Reset
```

8.4. IPXE を使用したホストの起動

Assisted Installer は、インフラストラクチャー環境の検出イメージを起動するために必要なすべての成果物を含む iPXE スクリプトを提供します。現在の iPXE の HTTPS 実装には制限があるため、HTTP サーバーで必要なアーティファクトをダウンロードして公開することを推奨します。現在、iPXE が HTTPS プロトコルをサポートしていても、サポートされているアルゴリズムは古く、推奨されていません。

サポートされている暗号の完全なリストは https://ipxe.org/crypto にあります。

前提条件

- API を使用してインフラストラクチャー環境を作成したか、UI を使用してクラスターを作成しました。
- インフラストラクチャー環境 ID がシェルに **$INFRA_ENV_ID** としてエクスポートされている。
- API にアクセスするときに使用する認証情報があり、シェルで **$API_TOKEN** としてトークンをエクスポートしました。

注記

Web コンソールを使用して iPXE を設定すると、**$INFRA_ENV_ID** および **$API_TOKEN** 変数が事前設定されています。

- イメージをホストする HTTP サーバーがあります。

注記

IBM Power は PXE のみをサポートします。また、IBM Power では、**/var/lib/tftpboot に grub2 をインストール** し、PXE 用の DHCP および TFTP をインストールする必要もあります。

手順

1. iPXE スクリプトを Web コンソールから直接ダウンロードするか、アシステッドインストーラーから iPXE スクリプトを取得します。

```
$ curl \
  --silent \
  --header "Authorization: Bearer $API_TOKEN" \
  https://api.openshift.com/api/assisted-install/v2/infra-
envs/$INFRA_ENV_ID/downloads/files?file_name=ipxe-script > ipxe-script
```

例

```
#!ipxe
initrd --name initrd http://api.openshift.com/api/assisted-images/images/<infra_env_id>/pxe-
initrd?arch=x86_64&image_token=<token_string>&version=4.10
kernel http://api.openshift.com/api/assisted-images/boot-artifacts/kernel?
arch=x86_64&version=4.10 initrd=initrd
coreos.live.rootfs_url=http://api.openshift.com/api/assisted-images/boot-artifacts/rootfs?
arch=x86_64&version=4.10 random.trust_cpu=on rd.luks.options=discard ignition.firstboot
ignition.platform.id=metal console=tty1 console=ttyS1,115200n8 coreos.inst.persistent-
kargs="console=tty1 console=ttyS1,115200n8"
boot
```

2. **ipxe-script** から URL を抽出して、必要なアーティファクトをダウンロードします。

   a. 初期 RAM ディスクをダウンロードします。

   ```
   $ awk '/^initrd /{print $NF}' ipxe-script | curl -o initrd.img
   ```

   b. Linux カーネルをダウンロードします。

   ```
   $ awk '/^kernel /{print $2}' ipxe-script | curl -o kernel
   ```

   c. ルートファイルシステムをダウンロードします。

   ```
   $ grep ^kernel ipxe-script | xargs -n1| grep ^coreos.live.rootfs_url | cut -d = -f 2- | curl -o
   rootfs.img
   ```

3. URL を **ipxe-script**` 内のさまざまなアーティファクトに変更して、ローカル HTTP サーバーに一致させます。以下に例を示します。

```
#!ipxe
set webserver http://192.168.0.1
initrd --name initrd $webserver/initrd.img
kernel $webserver/kernel initrd=initrd coreos.live.rootfs_url=$webserver/rootfs.img
random.trust_cpu=on rd.luks.options=discard ignition.firstboot ignition.platform.id=metal
console=tty1 console=ttyS1,115200n8 coreos.inst.persistent-kargs="console=tty1
console=ttyS1,115200n8"
boot
```

4. オプション: IBM zSystems に RHEL KVM を使用してインストールする場合は、追加のカーネル引数を指定してホストを起動する必要があります。

```
random.trust_cpu=on rd.luks.options=discard ignition.firstboot ignition.platform.id=metal
console=tty1 console=ttyS1,115200n8 coreos.inst.persistent-kargs="console=tty1
console=ttyS1,115200n8
```

注記

iPXE を使用して RHEL KVM にインストールする場合、VM ホスト上の VM が初回起動時に再起動されず、手動での起動が必要になることがあります。

5. オプション: IBM Power にインストールする場合は、次のように intramfs、カーネル、および root をダウンロードする必要があります。

   a. initrd.img と kernel.img を PXE ディレクトリー `/var/lib/tftpboot/rhcos` にコピーします。

   b. rootfs.img を HTTPD ディレクトリー `/var/www/html/install` にコピーします。

   c. 次のエントリーを `/var/lib/tftpboot/boot/grub2/grub.cfg` に追加します。

```
if [ ${net_default_mac} == fa:1d:67:35:13:20 ]; then
default=0
fallback=1
timeout=1
menuentry "CoreOS (BIOS)" {
echo "Loading kernel"
linux "/rhcos/kernel.img" ip=dhcp rd.neednet=1 ignition.platform.id=metal ignition.firstboot
coreos.live.rootfs_url=http://9.114.98.8:8000/install/rootfs.img
echo "Loading initrd"
initrd "/rhcos/initrd.img"
}
fi
```

第9章 ホストへのロールの割り当て

検出されたホストにロールを割り当てることができます。これらのロールはクラスター内のホストの機能を定義します。ロールは、標準の Kubernetes タイプのいずれかにすることができます: **control plane (master)** または **worker**。

ホストは、選択したロールの最小要件を満たしている必要があります。このドキュメントの前提条件セクションを参照するか、プリフライト要件 API を使用して、ハードウェア要件を見つけることができます。

ロールを選択しない場合は、システムが自動的に選択します。インストールの開始前であれば、いつでもロールを変更できます。

9.1. WEB コンソールを使用してロールを選択する

ホストが検出を終了した後、ロールを選択できます。

手順

1. **Host Discovery** タブに移動し、**Host Inventory** テーブルまで下にスクロールします。
2. 必要なホストの **Auto-assign** ドロップダウンを選択します。
3. **Control plane node** を選択して、このホストにコントロールプレーンロールを割り当てます。
4. **Worker** を選択して、このホストにワーカーロールを割り当てます。
5. バリデーションステータスを確認します。

9.2. API を使用したロールの選択

/v2/infra-envs/{infra_env_id}/hosts/{host_id} エンドポイントを使用して、ホストのロールを選択できます。ホストは、次の 2 つのロールのいずれかになります。

- **master**: **マスター** ロールを持つホストは、コントロールプレーンホストとして動作します。
- **worker**: **worker** ロールを持つホストは、ワーカーホストとして動作します。

デフォルトでは、Assisted Installer はホストを **auto-assign** に設定します。これは、ホストが **master** ロールか **worker** ロールかをインストーラが自動的に判断することを意味します。この手順を使用して、ホストのロールを設定します。

前提条件

- ホストをクラスターに追加した。

手順

1. API トークンを更新します。

```
$ source refresh-token
```

2. ホスト ID を取得します。

```
$ curl -s -X GET "https://api.openshift.com/api/assisted-install/v2/clusters/$CLUSTER_ID" \
--header "Content-Type: application/json" \
  -H "Authorization: Bearer $API_TOKEN" \
| jq '.host_networks[].host_ids'
```

出力例

```
[
  "1062663e-7989-8b2d-7fbb-e6f4d5bb28e5"
]
```

3. **host_role** 設定を変更します。

```
$ curl https://api.openshift.com/api/assisted-install/v2/infra-
envs/${INFRA_ENV_ID}/hosts/<host_id> \
-X PATCH \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d '
    {
      "host_role":"worker"
    }
' | jq
```

<host_id> をホストの ID に置き換えます。

9.3. ロールの自動割り当て

自分でロールを割り当てない場合、Assisted Installer はホストのロールを自動的に選択します。ロールの選択メカニズムでは、ホストのメモリー、CPU、およびディスク容量が考慮されます。これは、コントロールプレーンノードの最小要件を満たす最も弱い 3 つのホストにコントロールプレーンのロールを割り当てることを目的としています。他のすべてのホストは、デフォルトでワーカーノードになります。目標は、コントロールプレーンを実行するのに十分なリソースを提供し、実際のワークロードを実行するために容量集約型のホストを予約することです。

自動割り当ての決定は、インストール前にいつでも上書きできます。

検証により、自動選択が有効なものであることが確認されます。

9.4. 関連情報

前提条件

第10章 インストール前の検証

10.1. インストール前の検証の定義

Assisted Installer は、クラスターのインストールを可能な限り単純かつ効率的でエラーのないものにすることを目的としています。Assisted Installer は、インストールを開始する前に、設定と収集されたテレメトリーに対してバリデーションチェックを実行します。

Assisted Installer は、コントロールプレーントポロジー、ネットワーク設定、ホスト名など、インストール前に提供された情報を使用します。また、インストールしようとしているホストからのリアルタイムテレメトリーも使用します。

ホストが検出 ISO を起動すると、ホスト上でエージェントが開始されます。エージェントは、ホストの状態に関する情報を Assisted Installer に送信します。

Assisted Installer は、このすべての情報を使用して、インストール前のリアルタイムの検証を計算します。すべての検証は、インストールに対してブロッキングまたは非ブロッキングのいずれかです。

10.2. ブロッキング検証と非ブロッキング検証

ブロッキング検証により、インストールの進行が妨げられます。つまり、続行するには、問題を解決してブロッキング検証に合格する必要があります。

ノンブロッキング検証は警告であり、問題の原因となる可能性があることを通知します。

10.3. バリデーションの種類

Assisted Installer は、次の 2 種類のバリデーションを実行します。

ホスト

ホストのバリデーションにより、特定のホストの設定がインストールに対して有効であることを確認します。

クラスター

クラスターのバリデーションにより、クラスター全体の設定がインストールに対して有効であることを確認します。

10.4. ホストのバリデーション

10.4.1. REST API を使用してホストバリデーションを取得する

注記

Web コンソールを使用する場合、これらの検証の多くは名前で表示されません。ラベルと一致する検証のリストを取得するには、次の手順を使用します。

前提条件

- **jq** ユーティリティーをインストールした。

- API を使用してインフラストラクチャー環境を作成したか、UI を使用してクラスターを作成している。
- ホストが検出 ISO で起動されている
- シェルでクラスター ID を **CLUSTER_ID** としてエクスポートした。
- API にアクセスするときに使用する認証情報があり、トークンをシェルで **API_TOKEN** としてエクスポートした。

手順

1. API トークンを更新します。

```
$ source refresh-token
```

2. すべてのホストのすべてのバリデーションを取得します。

```
$ curl \
  --silent \
  --header "Authorization: Bearer $API_TOKEN" \
  https://api.openshift.com/api/assisted-install/v2/clusters/$CLUSTER_ID/hosts \
  | jq -r .[].validations_info \
  | jq 'map(.[])'
```

3. すべてのホストのパスしていないバリデーションを取得します。

```
$ curl \
  --silent \
  --header "Authorization: Bearer $API_TOKEN" \
  https://api.openshift.com/api/assisted-install/v2/clusters/$CLUSTER_ID/hosts \
  | jq -r .[].validations_info \
  | jq 'map(.[]) | map(select(.status=="failure" or .status=="pending")) | select(length>0)'
```

10.4.2. ホストバリデーションの詳細

パラメーター	バリデーションタイプ	説明
connected	非ブロッキング	ホストが最近 Assisted Installer と通信したことを確認します。
has-inventory	非ブロッキング	Assisted Installer がホストからインベントリーを受信したことを確認します。
has-min-cpu-cores	非ブロッキング	CPU コアの数が最小要件を満たしていることを確認します。
has-min-memory	非ブロッキング	メモリーの量が最小要件を満たしていることを確認します。

パラメーター	バリデーションタイプ	説明
has-min-valid-disks	非ブロッキング	少なくとも 1 つの使用可能なディスクが適格基準を満たしていることを確認します。
has-cpu-cores-for-role	ブロッキング	コアの数がホストのロールの最小要件を満たしていることを確認します。
has-memory-for-role	ブロッキング	メモリーの量がホストのロールの最小要件を満たしていることを確認します。
ignition-downloadable	ブロッキング	Day 2 ホストの場合、ホストが Day 1 クラスターからイグニッション設定をダウンロードできることを確認します。
belongs-to-majority-group	ブロッキング	マジョリティグループは、クラスター上で最大のフルメッシュ接続グループであり、すべてのメンバーが他のすべてのメンバーと通信できます。この検証では、マルチノードの Day 1 クラスター内のホストが過半数グループに属していることを確認します。
valid-platform-network-settings	ブロッキング	プラットフォームがネットワーク設定に対して有効であることを確認します。
ntp-synced	非ブロッキング	ホストで時刻を同期するために NTP サーバーが正常に使用されたかどうかを確認します。
container-images-available	非ブロッキング	コンテナーイメージがイメージレジストリーから正常にプルされたかどうかを確認します。
sufficient-installation-disk-speed	ブロッキング	以前のインストールのディスク速度メトリックが要件を満たしていることを確認します (存在する場合)。
sufficient-network-latency-requirement-for-role	ブロッキング	クラスター内のホスト間の平均ネットワーク遅延が要件を満たしていることを確認します。
sufficient-packet-loss-requirement-for-role	ブロッキング	クラスター内のホスト間のネットワークパケット損失が要件を満たしていることを確認します。
has-default-route	ブロッキング	ホストにデフォルトルートが設定されていることを確認します。
api-domain-name-resolved-correctly	ブロッキング	ユーザー管理ネットワークを使用するマルチノードクラスターの場合。ホストがクラスターの API ドメイン名を解決できることを確認します。

パラメーター	バリデーションタイプ	説明
api-int-domain-name-resolved-correctly	ブロッキング	ユーザー管理ネットワークを使用するマルチノードクラスターの場合。ホストがクラスターの内部 API ドメイン名を解決できることを確認します。
apps-domain-name-resolved-correctly	ブロッキング	ユーザー管理ネットワークを使用するマルチノードクラスターの場合。ホストがクラスターの内部アプリドメイン名を解決できることを確認します。
compatible-with-cluster-platform	非ブロッキング	ホストがクラスタープラットフォームと互換性があることを確認します
dns-wildcard-not-configured	ブロッキング	OpenShift で既知の問題が発生するため、ワイルドカード DNS *.<cluster_name>.<base_domain> が設定されていないことを確認します。
disk-encryption-requirements-satisfied	非ブロッキング	設定されているホストとディスクの暗号化のタイプが要件を満たしていることを確認します。
non-overlapping-subnets	ブロッキング	このホストに重複するサブネットがないことを確認します。
hostname-unique	ブロッキング	ホスト名がクラスター内で一意であることを確認します。
hostname-valid	ブロッキング	ホスト名の有効性をチェックします。つまり、ホスト名の一般的な形式と一致し、禁止されていないことを意味します。
belongs-to-machine-cidr	ブロッキング	ホスト IP がマシン CIDR のアドレス範囲内にあることを確認します。
lso-requirements-satisfied	ブロッキング	クラスターがローカルストレージ Operator の要件を満たしていることを検証します。

パラメーター	バリデーションタイプ	説明
odf-requirements-satisfied	ブロッキング	クラスターが Openshift Data Foundation Operator の要件を満たしていることを検証します。 ● クラスターには最低 3 つのホストがあります。 ● クラスターには 3 つのマスターのみ、または少なくとも 3 つのワーカーがあります。 ● クラスターには 3 つの適格なディスクがあり、各ホストには適格なディスクが必要です。 ● 3 つ以上のホストを持つクラスターでは、ホストのロールを自動割り当てにしないでください。
cnv-requirements-satisfied	ブロッキング	クラスターがコンテナーネイティブ仮想化の要件を満たしていることを検証します。 ● ホストの BIOS で CPU 仮想化が有効になっている必要があります。 ● ホストには、コンテナーネイティブ仮想化に使用できる十分な CPU コアと RAM が必要です。 ● 必要に応じてホストパスプロビジョナーを検証します。
lvm-requirements-satisfied	ブロッキング	クラスターが論理ボリュームマネージャー Operator の要件を満たしていることを検証します。 ● ホストには、パーティション化もフォーマットもされていない、少なくとも 1 つの追加の空のディスクがあります。
vsphere-disk-uuid-enabled	非ブロッキング	有効な各ディスクで **disk.EnableUUID** が **true** に設定されていることを確認します。vSphere では、これにより各ディスクに UUID が割り当てられます。
compatible-agent	ブロッキング	検出エージェントのバージョンがエージェントの Docker イメージのバージョンと互換性があることを確認します。
no-skip-installation-disk	ブロッキング	インストールディスクがディスクフォーマットをスキップしていないことを確認します。

パラメーター	バリデーションタイプ	説明
no-skip-missing-disk	ブロッキング	フォーマットをスキップするようにマークされたすべてのディスクがインベントリーにあることを確認します。ディスク ID は再起動時に変更される可能性があり、このバリデーションにより、それによって引き起こされる問題が防止されます。
media-connected	ブロッキング	ホストへのインストールメディアの接続を確認します。
machine-cidr-defined	非ブロッキング	クラスターのマシンネットワーク定義が存在することを確認します。
id-platform-network-settings	ブロッキング	プラットフォームがネットワーク設定と互換性があることを確認します。一部のプラットフォームは、Single Node Openshift をインストールする場合、またはユーザー管理ネットワークを使用する場合にのみ許可されます。

10.5. クラスターのバリデーション

10.5.1. REST API を使用してクラスターバリデーションを取得する

Web コンソールを使用する場合、これらの検証の多くは名前で表示されません。ラベルと一致するバリデーションのリストを取得するには、次の手順を使用します。

前提条件

- **jq** ユーティリティーをインストールした。
- API を使用してインフラストラクチャー環境を作成したか、UI を使用してクラスターを作成している。
- シェルでクラスター ID を **CLUSTER_ID** としてエクスポートした。
- API にアクセスするときに使用する認証情報があり、トークンをシェルで **API_TOKEN** としてエクスポートした。

手順

1. API トークンを更新します。

```
$ source refresh-token
```

2. すべてのクラスターバリデーションを取得します。

```
$ curl \
  --silent \
  --header "Authorization: Bearer $API_TOKEN" \
```

```
https://api.openshift.com/api/assisted-install/v2/clusters/$CLUSTER_ID \
| jq -r .validations_info \
| jq 'map(.[])'
```

3. パスしなかったクラスターバリデーションを取得します。

```
$ curl \
  --silent \
  --header "Authorization: Bearer $API_TOKEN" \
  https://api.openshift.com/api/assisted-install/v2/clusters/$CLUSTER_ID \
  | jq -r .validations_info \
  | jq '. | map(.[] | select(.status=="failure" or .status=="pending")) | select(length>0)'
```

10.5.2. クラスターバリデーションの詳細

パラメーター	バリデーションタイプ	説明
machine-cidr-defined	非ブロッキング	クラスターのマシンネットワーク定義が存在することを確認します。
cluster-cidr-defined	非ブロッキング	クラスターのクラスターネットワーク定義が存在することを確認します。
service-cidr-defined	非ブロッキング	クラスターのサービスネットワーク定義が存在することを確認します。
cidrs の重複なし	ブロッキング	定義されたネットワークが重複していないことを確認します。
networks-same-address-families	ブロッキング	定義されたネットワークが同じアドレスファミリーを共有していることを確認します (有効なアドレスファミリーは IPv4、IPv6 です)。
network-prefix-valid	ブロッキング	クラスターネットワーク 接頭辞をチェックして、それが有効であり、すべてのホストに十分なアドレス空間を許可していることを確認します。
machine-cidr-equals-to-calculated-cidr	ブロッキング	非ユーザー管理のネットワーククラスターの場合。**apiVIP** または **ingressVIP** が存在する場合、それらがマシン CIDR のメンバーであることを確認します。
api-vips-defined	非ブロッキング	非ユーザー管理のネットワーククラスターの場合。**apiVIP** が存在することを確認します。
api-vips-valid	ブロッキング	非ユーザー管理のネットワーククラスターの場合。**apiVIPs** がマシン CIDR に属しており、使用されていないかどうかを確認します。

パラメーター	バリデーションタイプ	説明
ingress-vips-defined	ブロッキング	非ユーザー管理のネットワーククラスターの場合。**ingressVIP** が存在することを確認します。
ingress-vips-valid	非ブロッキング	非ユーザー管理のネットワーククラスターの場合。**ingressVIP** がマシンの CIDR に属しており、使用されていないかどうかを確認します。
all-hosts-are-ready-to-install	ブロッキング	クラスター内のすべてのホストがインストール準備完了ステータスにあることを確認します。
sufficient-masters-count	ブロッキング	この検証は、マルチノードクラスターにのみ適用されます。 ● クラスターには正確に 3 つのマスターが必要です。 ● クラスターにワーカーノードがある場合は、少なくとも 2 つのワーカーノードが存在する必要があります。
dns-domain-defined	非ブロッキング	クラスターのベース DNS ドメインが存在することを確認します。
pull-secret-set	非ブロッキング	プルシークレットが存在することを確認します。プルシークレットが有効または承認されていることを確認しません。
ntp-server-configured	ブロッキング	各ホストクロックの同期が 4 分以内であることを確認します。
lso-requirements-satisfied	ブロッキング	クラスターがローカルストレージ Operator の要件を満たしていることを検証します。
odf-requirements-satisfied	ブロッキング	クラスターが Openshift Data Foundation Operator の要件を満たしていることを検証します。 ● クラスターには最低 3 つのホストがあります。 ● クラスターには 3 つのマスターのみ、または少なくとも 3 つのワーカーがあります。 ● クラスターには 3 つの適格なディスクがあり、各ホストには適格なディスクが必要です。

パラメーター	バリデーションタイプ	説明
cnv-requirements-satisfied	ブロッキング	クラスターがコンテナーネイティブ仮想化の要件を満たしていることを検証します。 ● クラスターの CPU アーキテクチャーは x86 です
lvm-requirements-satisfied	ブロッキング	クラスターが論理ボリュームマネージャー Operator の要件を満たしていることを検証します。 ● クラスターはシングルノードである必要があります。 ● クラスターは Openshift >= 4.11.0 を実行している必要があります。
network-type-valid	ブロッキング	ネットワークタイプが存在する場合、その有効性をチェックします。 ● ネットワークタイプは OpenshiftSDN または OVNKubernetes である必要があります。 ● OpenshiftSDN は、IPv6 または単一ノードの Openshift をサポートしていません。OpenShiftSDN は、OpenShift Container Platform 4.15 以降のリリースではサポートされません。 ● OVNKubernetes は VIP DHCP 割り当てをサポートしていません。

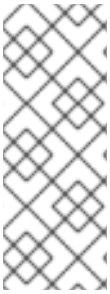
第11章 ネットワーク設定

このセクションでは、Assisted Installer を使用したネットワーク設定の基本について説明します。

11.1. CLUSTER NETWORKING

OpenShift で使用されるさまざまなネットワークタイプとアドレスがあり、以下の表に一覧表示されています。

型	DNS	説明
clusterNetwork		Pod IP アドレスの割り当てに使用する IP アドレスプール。
serviceNetwork		サービスの IP アドレスプール。
machineNetwork		クラスターを形成するマシンの IP アドレスブロック。
apiVIP	**api.<clustername.clusterdomain>**	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、事前に設定する必要があります。デュアルスタックネットワークを使用してデプロイする場合、これは IPv4 アドレスである必要があります。
apiVIPs	**api.<clustername.clusterdomain>**	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、事前に設定する必要があります。デュアルスタックネットワークを使用する場合、最初のアドレスは IPv4 アドレスで、2 番目のアドレスは IPv6 アドレスである必要があります。**apiVIP** 設定も行う必要があります。
ingressVIP	***.apps.<clustername.clusterdomain>**	Ingress トラフィックに使用する VIP。デュアルスタックネットワークを使用してデプロイする場合、これは IPv4 アドレスである必要があります。
ingressVIPs	***.apps.<clustername.clusterdomain>**	Ingress トラフィックに使用する VIP。デュアルスタックネットワークを使用してデプロイする場合、最初のアドレスは IPv4 アドレスで、2 番目のアドレスは IPv6 アドレスである必要があります。**ingressVIP** 設定も行う必要があります。



注記

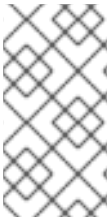
OpenShift Container Platform 4.12 では、デュアルスタックネットワークで複数の IP アドレスを受け入れる新しい **apiVIPs** および **ingressVIPs** 設定が導入されています。デュアルスタックネットワークを使用する場合、最初の IP アドレスは IPv4 アドレスで、2 番目の IP アドレスは IPv6 アドレスである必要があります。**apiVIP** と **IngressVIP** は新しい設定に置き換えられますが、API を使用して設定を変更する場合は、新しい設定と古い設定の両方を設定する必要があります。

目的のネットワークスタックに応じて、さまざまなネットワークコントローラーを選択できます。現在、Assisted Service は、以下の設定のいずれかを使用して OpenShift Container Platform クラスターをデプロイできます。

- IPv4
- IPv6
- Dual-stack (IPv4 + IPv6)

サポートされるネットワークコントローラーは、選択したスタックによって異なります。以下の表にまとめられています。詳細な Container Network Interface (CNI) ネットワークプロバイダー機能の比較は、OCP Networking のドキュメント を参照してください。

Stack	SDN	OVN
IPv4	はい	はい
IPv6	いいえ	はい
デュアルスタック	いいえ	はい



注記

OVN は、OpenShift Container Platform 4.12 以降のリリースにおいてデフォルトの Container Network Interface (CNI) です。SDN は OpenShift Container Platform 4.14 までサポートされますが、OpenShift Container Platform 4.15 以降のリリースでは対応していません。

11.1.1. 制限事項

11.1.1.1. SDN

- SDN コントローラーは、シングルノード OpenShift ではサポートされていません。
- SDN コントローラーは IPv6 をサポートしていません。
- SDN コントローラーは OpenShift Container Platform 4.15 以降のリリースではサポートされません。詳細は、OpenShift Container Platform リリースノートの OpenShift SDN ネットワークプラグインの非推奨化 を参照してください。

11.1.1.2. OVN-Kubernetes

OCP ドキュメントの OVN-Kubernetes の制限に関するセクション を参照してください。

11.1.2. クラスターネットワーク

クラスターネットワークは、クラスターにデプロイされたすべての Pod が IP アドレスを取得するネットワークです。ワークロードがクラスターを形成する多くのノードにまたがって存在する可能性があることを考えると、ネットワークプロバイダーが Pod の IP アドレスに基づいて個々のノードを簡単に見つけられることが重要です。これを行うために、**clusterNetwork.cidr** は、**clusterNetwork.hostPrefix** で定義されたサイズのサブネットにさらに分割されます。

ホスト 接頭辞は、クラスター内の個々のノードに割り当てられるサブネットの長さを指定します。クラスターがマルチノードクラスターにアドレスを割り当てる方法の例:

```
---
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
---
```

上記のスニペットを使用して 3 ノードクラスターを作成すると、次のネットワークトポロジーが作成される場合があります。

- ノード #1 でスケジュールされた Pod は **10.128.0.0/23** から IP を取得します
- ノード #2 でスケジュールされた Pod は **10.128.2.0/23** から IP を取得します
- ノード #3 でスケジュールされた Pod は、**10.128.4.0/23** から IP を取得します

OVN-K8 内部の説明はこのドキュメントの範囲外ですが、上記のパターンは、Pod とそれに対応するノード間のマッピングの大きなリストを保持することなく、異なるノード間で Pod-to-Pod トラフィックをルーティングする方法を提供します。

11.1.3. マシンネットワーク

マシンネットワークは、クラスターを設定するすべてのホストが相互に通信するために使用するネットワークです。これは、API と Ingress VIP を含める必要があるサブネットでもあります。

11.1.4. マルチノードクラスターと比較した SNO

シングルノード OpenShift をデプロイするか、マルチノードクラスターをデプロイするかによって、異なる値が必須になります。以下の表で、これについて詳しく説明します。

パラメーター	SNO	DHCP モードのマルチノードクラスター	DHCP モードを使用しないマルチノードクラスター
clusterNetwork	必須	必須	必須
serviceNetwork	必須	必須	必須
machineNetwork	自動割り当て可能 (*)	自動割り当て可能 (*)	自動割り当て可能 (*)

パラメーター	SNO	DHCP モードのマルチノードクラスター	DHCP モードを使用しないマルチノードクラスター
apiVIP	禁止されている	禁止されている	必須
apiVIPs	禁止されている	禁止されている	4.12 以降のリリースで必須
ingressVIP	禁止されている	禁止されている	必須
ingressVIPs	禁止されている	禁止されている	4.12 以降のリリースで必須

(*) マシンネットワーク CIDR の自動割り当ては、ホストネットワークが 1 つしかない場合に発生します。それ以外の場合は、明示的に指定する必要があります。

11.1.5. エアギャップ環境

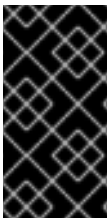
インターネットアクセスなしでクラスターをデプロイメントするためのワークフローには、このドキュメントの範囲外の前提条件がいくつかあります。いくつかの洞察については、Zero Touch Provisioning the hard way Git リポジトリー を参照してください。

11.2. VIP DHCP 割り当て

VIP DHCP 割り当ては、DHCP サーバーからこれらの IP アドレスを自動的に割り当てるサービスの機能を活用することで、ユーザーが API および Ingress 用の仮想 IP を手動で提供する要件をスキップできるようにする機能です。

この機能を有効にすると、クラスター設定から **api_vip** と **ingress_vip** を使用する代わりに、サービスはリース割り当てリクエストを送信し、応答に基づいて VIP を適宜使用します。サービスは、マシンネットワークから IP アドレスを割り当てます。

これは OpenShift Container Platform の機能ではなく、設定を容易にするために Assisted Service に実装されていることに注意してください。



重要

VIP DHCP 割り当ては現在、OpenShift Container Platform SDN ネットワークタイプに制限されています。SDN は、OpenShift Container Platform バージョン 4.15 以降ではサポートされません。したがって、VIP DHCP 割り当てのサポートも OpenShift Container Platform 4.15 以降で終了します。

11.2.1. 自動割り当てを有効にするペイロードの例

```
---
{
  "vip_dhcp_allocation": true,
  "network_type": "OVNKubernetes",
  "user_managed_networking": false,
  "cluster_networks": [
```

```
    {
      "cidr": "10.128.0.0/14",
      "host_prefix": 23
    }
  ],
  "service_networks": [
    {
      "cidr": "172.30.0.0/16"
    }
  ],
  "machine_networks": [
    {
      "cidr": "192.168.127.0/24"
    }
  ]
}
---
```

11.2.2. 自動割り当てを無効にするペイロードの例

```
---
{
  "api_vips": [
    {
        "ip": "192.168.127.100"
    }
  ],
  "ingress_vips": [
    {
        "ip": "192.168.127.101"
    }
  ],
  "vip_dhcp_allocation": false,
  "network_type": "OVNKubernetes",
  "user_managed_networking": false,
  "cluster_networks": [
    {
      "cidr": "10.128.0.0/14",
      "host_prefix": 23
    }
  ],
  "service_networks": [
    {
      "cidr": "172.30.0.0/16"
    }
  ]
}
---
```

11.3. 関連情報

- ベアメタル IPI のドキュメント には、VIP アドレスの構文に関する追加の説明が記載されています。

11.4. ユーザー管理ネットワークとクラスター管理ネットワークの違いについて

ユーザー管理ネットワーキングは、非標準のネットワークトポロジーを使用する顧客が OpenShift Container Platform クラスターをデプロイできるようにする、Assisted Installer の機能です。たとえば、以下のとおりです。

- VIP アドレスの処理に **keepalived** と VRRP を使用したくない外部ロードバランサーをお持ちのお客様。
- 多くの異なる L2 ネットワークセグメントに分散されたクラスターノードを使用したデプロイメント。

11.4.1. 検証

Assisted Installer では、インストールの開始を許可する前に、さまざまなネットワーク検証が行われます。ユーザー管理ネットワークを有効にすると、次の検証が変更されます。

- L2 チェック (ARP) の代わりに L3 接続チェック (ICMP) が実行されます。

11.5. 静的ネットワーク設定

検出 ISO を生成または更新するときに、静的ネットワーク設定を使用できます。

11.5.1. 前提条件

- あなたは NMState に精通しています。

11.5.2. NMState 設定

YAML 形式の NMState ファイルは、ホストに必要なネットワーク設定を指定します。これには、検出時にインターフェイスの実際の名前に置き換えられるインターフェイスの論理名があります。

11.5.2.1. NMState 設定の例

```
---
dns-resolver:
  config:
    server:
    - 192.168.126.1
interfaces:
- ipv4:
    address:
    - ip: 192.168.126.30
      prefix-length: 24
    dhcp: false
    enabled: true
  name: eth0
  state: up
  type: ethernet
- ipv4:
    address:
    - ip: 192.168.141.30
      prefix-length: 24
```

```
    dhcp: false
    enabled: true
  name: eth1
  state: up
  type: ethernet
routes:
  config:
  - destination: 0.0.0.0/0
    next-hop-address: 192.168.126.1
    next-hop-interface: eth0
    table-id: 254
---
```

11.5.3. MAC インターフェイスのマッピング

MAC インターフェイスマップは、NMState 設定で定義された論理インターフェイスを、ホスト上に存在する実際のインターフェイスにマップする属性です。

マッピングでは、ホストに存在する物理インターフェイスを常に使用する必要があります。たとえば、NMState 設定がボンドまたは VLAN を定義する場合、マッピングには親インターフェイスのエントリーのみを含める必要があります。

11.5.3.1. MAC インターフェイスマッピングの例

```
---
mac_interface_map: [
    {
      mac_address: 02:00:00:2c:23:a5,
      logical_nic_name: eth0
    },
    {
      mac_address: 02:00:00:68:73:dc,
      logical_nic_name: eth1
    }
]
---
```

11.5.4. 追加の NMState 設定の例

以下の例は、部分的な設定を示すことのみを目的としています。そのまま使用することを意図したものではなく、使用する環境に合わせて常に調整する必要があります。誤って使用すると、マシンがネットワークに接続できなくなる可能性があります。

11.5.4.1. タグ VLAN

```
---
    interfaces:
    - ipv4:
        address:
        - ip: 192.168.143.15
          prefix-length: 24
        dhcp: false
        enabled: true
      ipv6:
```

```
      enabled: false
    name: eth0.404
    state: up
    type: vlan
    vlan:
      base-iface: eth0
      id: 404
      reorder-headers: true
---
```

11.5.4.2. ネットワークボンド

```
---
  interfaces:
  - ipv4:
      address:
      - ip: 192.168.138.15
        prefix-length: 24
      dhcp: false
      enabled: true
    ipv6:
      enabled: false
    link-aggregation:
      mode: active-backup
      options:
        all_slaves_active: delivered
        miimon: "140"
      slaves:
      - eth0
      - eth1
    name: bond0
    state: up
    type: bond
---
```

11.6. API を使用した静的ネットワーク設定の適用

Assisted Installer API を使用して静的ネットワーク設定を適用できます。

前提条件

1. API を使用してインフラストラクチャー環境を作成したか、UI を使用してクラスターを作成している。
2. インフラストラクチャー環境 ID がシェルに **$INFRA_ENV_ID** としてエクスポートされている。
3. API にアクセスするときに使用する認証情報があり、シェルで **$API_TOKEN** としてトークンをエクスポートしました。
4. **server-a.yaml** および **server-b.yaml** として利用可能な静的ネットワーク設定を持つ YAML ファイルがあります。

手順

1. API リクエストを含む一時ファイル **/tmp/request-body.txt** を作成します。

```
---
jq -n --arg NMSTATE_YAML1 "$(cat server-a.yaml)" --arg NMSTATE_YAML2 "$(cat server-b.yaml)" \
'{
  "static_network_config": [
    {
      "network_yaml": $NMSTATE_YAML1,
      "mac_interface_map": [{"mac_address": "02:00:00:2c:23:a5", "logical_nic_name": "eth0"}, {"mac_address": "02:00:00:68:73:dc", "logical_nic_name": "eth1"}]
    },
    {
      "network_yaml": $NMSTATE_YAML2,
      "mac_interface_map": [{"mac_address": "02:00:00:9f:85:eb", "logical_nic_name": "eth1"}, {"mac_address": "02:00:00:c8:be:9b", "logical_nic_name": "eth0"}]
    }
  ]
}' >> /tmp/request-body.txt
---
```

2. API トークンを更新します。

```
$ source refresh-token
```

3. Assisted Service API エンドポイントにリクエストを送信します。

```
---
$ curl -H "Content-Type: application/json" \
-X PATCH -d @/tmp/request-body.txt \
-H "Authorization: Bearer ${API_TOKEN}" \
https://api.openshift.com/api/assisted-install/v2/infra-envs/$INFRA_ENV_ID
---
```

11.7. 関連情報

- Web コンソールを使用した静的ネットワーク設定の適用

11.8. デュアルスタックネットワークへの変換

デュアルスタック IPv4/IPv6 設定により、Pod が IPv4 と IPv6 の両方のサブネットに存在するクラスターのデプロイが可能になります。

11.8.1. 前提条件

- OVN-K8s のドキュメント を理解している。

11.8.2. シングルノード OpenShift のペイロードの例

```
---
{
  "network_type": "OVNKubernetes",
  "user_managed_networking": false,
```

```
  "cluster_networks": [
    {
      "cidr": "10.128.0.0/14",
      "host_prefix": 23
    },
    {
      "cidr": "fd01::/48",
      "host_prefix": 64
    }
  ],
  "service_networks": [
    {"cidr": "172.30.0.0/16"}, {"cidr": "fd02::/112"}
  ],
  "machine_networks": [
    {"cidr": "192.168.127.0/24"},{"cidr": "1001:db8::/120"}
  ]
}
---
```

11.8.3. 多くのノードで設定される OpenShift Container Platform クラスターのペイロードの例

```
---
{
  "vip_dhcp_allocation": false,
  "network_type": "OVNKubernetes",
  "user_managed_networking": false,
  "api_vips": [
    {
      "ip": "192.168.127.100"
    },
    {
      "ip": "2001:0db8:85a3:0000:0000:8a2e:0370:7334"
    }
  ],
  "ingress_vips": [
    {
      "ip": "192.168.127.101"
    },
    {
      "ip": "2001:0db8:85a3:0000:0000:8a2e:0370:7335"
    }
  ],
  "cluster_networks": [
    {
      "cidr": "10.128.0.0/14",
      "host_prefix": 23
    },
    {
      "cidr": "fd01::/48",
      "host_prefix": 64
    }
  ],
  "service_networks": [
    {"cidr": "172.30.0.0/16"}, {"cidr": "fd02::/112"}
```

```
  ],
  "machine_networks": [
    {"cidr": "192.168.127.0/24"},{"cidr": "1001:db8::/120"}
  ]
}
---
```

11.8.4. 制限事項

デュアルスタックネットワークを使用する場合、**api_vips** IP アドレスと **ingress_vips** IP アドレスの設定は、プライマリー IP アドレスファミリーである必要があり、IPv4 アドレスである必要があります。現在、Red Hat は、IPv6 をプライマリー IP アドレスファミリーとして使用するデュアルスタック VIP またはデュアルスタックネットワークをサポートしていません。Red Hat は、IPv4 をプライマリー IP アドレスファミリーとして、IPv6 をセカンダリー IP アドレスファミリーとして使用するデュアルスタックネットワークをサポートしています。したがって、IP アドレス値を入力するときは、IPv6 エントリーの前に IPv4 エントリーを配置する必要があります。

11.9. 関連情報

- OpenShift ネットワーキングについて
- OpenShift SDN - CNI ネットワークプロバイダー
- OVN-Kubernetes - CNI ネットワークプロバイダー
- デュアルスタックサービス設定シナリオ
- ベアメタル OCP へのインストール
- Cluster Network Operator の設定

第12章 クラスターの拡張

ユーザーインターフェイスまたは API を使用してホストを追加して、Assisted Installer でインストールしたクラスターを拡張できます。

関連情報

- クラスターにノード追加時の API 接続の失敗
- OpenShift クラスターでのマルチアーキテクチャーのコンピュートマシンの設定

12.1. マルチアーキテクチャーサポートの確認

異なるアーキテクチャーのノードを追加する前に、クラスターが複数のアーキテクチャーをサポートできることを確認する必要があります。

手順

1. CLI を使用してクラスターにログインします。
2. 次のコマンドを実行して、クラスターがアーキテクチャーペイロードを使用していることを確認します。

```
$ oc adm release info -o json | jq .metadata.metadata
```

検証

- 次の出力が表示された場合、クラスターは複数のアーキテクチャーをサポートしています。

```
{
  "release.openshift.io/architecture": "multi"
}
```

12.2. マルチアーキテクチャークラスターのインストール

x86_64 コントロールプレーンを備えたクラスターは、2 つの異なる CPU アーキテクチャーを持つワーカーノードをサポートできます。混合アーキテクチャークラスターは、各アーキテクチャーの長所を組み合わせて、さまざまなワークロードをサポートします。

たとえば、**x86_64** の既存の OpenShift Container Platform クラスターに、 **arm64**、**IBM Power**、または **IBM zSystems** ワーカーノードを追加できます。

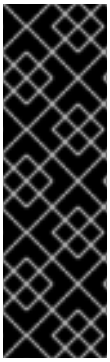
インストールの主な手順は次のとおりです。

1. マルチアーキテクチャークラスターを作成して登録します。
2. **x86_64** インフラストラクチャー環境を作成し、**x86_64** の ISO 検出イメージをダウンロードして、コントロールプレーンを追加します。コントロールプレーンには **x86_64** アーキテクチャーが必要です。
3. **arm64**、**IBM Power** または **IBM zSystems** インフラストラクチャー環境を作成し、**arm64**、**IBM Power** または **IBM zSystems** の ISO 検出イメージをダウンロードして、ワーカーノードを追加します。

サポート対象のプラットフォーム

以下の表に、OpenShift Container Platform の各バージョンの混合アーキテクチャークラスターをサポートするプラットフォームをリストします。インストールするバージョンに適したプラットフォームを使用してください。

OpenShift Container Platform バージョン	サポート対象のプラットフォーム	Day 1 コントロールプレーンアーキテクチャー	Day 2 ノードアーキテクチャー
4.12.0	• Microsoft Azure (TP)	• **x86_64**	• **arm64**
4.13.0	• Microsoft Azure • Amazon Web Services • ベアメタル (TP)	• **x86_64** • **x86_64** • **x86_64**	• **arm64** • **arm64** • **arm64**
4.14.0	• Microsoft Azure • Amazon Web Services • ベアメタル • Google Cloud Platform • IBM® Power® • IBM Z®	• **x86_64** • **x86_64** • **x86_64** • **x86_64** • **x86_64** • **x86_64**	• **arm64** • **arm64** • **arm64** • **arm64** • **ppc64le** • **s390x**



重要

テクノロジープレビュー (TP) 機能は、Red Hat 製品サポートのサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではない場合があります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

主なステップ

1. API を使用して OpenShift Container Platform をインストールする手順を開始します。詳細は、**関連情報** セクションの **Assisted Installer API を使用したインストール**を参照してください。

2. インストールの "Registering a new cluster" のステップに到達したら、クラスターを **multi-architecture** クラスターとして登録します。

```
$ curl -s -X POST https://api.openshift.com/api/assisted-install/v2/clusters \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d "$(jq --null-input \
  --slurpfile pull_secret ~/Downloads/pull-secret.txt '
{
  "name": "testcluster",
  "openshift_version": "<version-number>-multi", (1)
  "cpu_architecture" : "multi" (2)
  "high_availability_mode": "full" (3)
  "base_dns_domain": "example.com",
  "pull_secret": $pull_secret[0] | tojson
}
')" | jq '.id'
```

注記

1. OpenShift Container Platform のバージョン番号には **複数の** オプションを使用します (例: **"4.12-multi"**)。
2. CPU アーキテクチャーを **multi-** に設定します。
3. **full** の値を使用して、マルチノードの OpenShift Container Platform を指定します。

3. インストールの "Registering a new infrastructure environment" ステップに到達したら、**cpu_architecture** を **x86_64** に設定します。

```
$ curl https://api.openshift.com/api/assisted-install/v2/infra-envs \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d "$(jq --null-input \
 --slurpfile pull_secret ~/Downloads/pull-secret.txt \
 --arg cluster_id ${CLUSTER_ID} '
   {
     "name": "testcluster-infra-env",
     "image_type":"full-iso",
     "cluster_id": $cluster_id,
     "cpu_architecture" : "x86_64"
     "pull_secret": $pull_secret[0] | tojson
   }
')" | jq '.id'
```

4. インストールの "Adding hosts" ステップに到達したら、**host_role** を **master** に設定します。

注記

詳細は、**関連情報** の **ホストへのロールの割り当て** を参照してください。

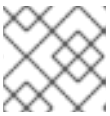
```
$ curl https://api.openshift.com/api/assisted-install/v2/infra-
envs/${INFRA_ENV_ID}/hosts/<host_id> \
-X PATCH \
```

```
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d '
  {
    "host_role":"master"
  }
' | jq
```

5. **x86_64** アーキテクチャーの検出イメージをダウンロードします。
6. 生成された検出イメージを使用して **x86_64** アーキテクチャーホストを起動します。
7. インストールを開始し、クラスターが完全にインストールされるまで待ちます。
8. インストールの "Registering a new infrastructure environment" 手順を繰り返します。今回は、**cpu_architecture** を **ppc64le** (IBM Power の場合)、**s390x** (IBM Z の場合)、または **arm64** のいずれかに設定します。以下に例を示します。

```
$ curl -s -X POST https://api.openshift.com/api/assisted-install/v2/clusters \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d "$(jq --null-input \
  --slurpfile pull_secret ~/Downloads/pull-secret.txt '
{
  "name": "testcluster",
  "openshift_version": "4.12",
  "cpu_architecture" : "arm64"
  "high_availability_mode": "full"
  "base_dns_domain": "example.com",
  "pull_secret": $pull_secret[0] | tojson
}
')" | jq '.id'
```

9. インストールの "Adding hosts" 手順を繰り返します。今回は、**host_role** を **worker** に設定します。



注記

詳細は、**関連情報** の **ホストへのロールの割り当て** を参照してください。

```
$ curl https://api.openshift.com/api/assisted-install/v2/infra-
envs/${INFRA_ENV_ID}/hosts/<host_id> \
-X PATCH \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d '
  {
    "host_role":"worker"
  }
' | jq
```

10. **arm64**、**ppc64le**、または **s390x** アーキテクチャーの検出イメージをダウンロードします。
11. 生成された検出イメージを使用してアーキテクチャーホストを起動します。

12. インストールを開始し、クラスターが完全にインストールされるまで待ちます。

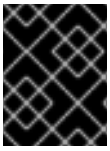
検証

- 次のコマンドを実行して、クラスター内の **arm64**、**ppc64le**、または **s390x** ワーカーノードを表示します。

```
$ oc get nodes -o wide
```

12.3. WEB コンソールを使用したホストの追加

Assisted Installer を使用して作成されたクラスターにホストを追加できます。



重要

Assisted Installer クラスターへのホストの追加は、OpenShift Container Platform バージョン 4.11 以降を実行しているクラスターでのみサポートされます。

手順

1. OpenShift Cluster Manager にログインし、拡張するクラスターをクリックします。
2. **Add hosts** をクリックし、新規ホストの検出 ISO をダウンロードし、必要に応じて SSH 公開鍵を追加し、クラスター全体のプロキシーを設定します。
3. オプション: 必要に応じて、Ignition ファイルを変更します。
4. 検出 ISO を使用してターゲットホストを起動し、ホストがコンソールで検出されるまで待ちます。
5. ホストロールを選択します。**worker** または **control plane** ホストのいずれかになります。
6. インストールを開始します。
7. インストールが進行すると、ホストに対して保留中の証明書署名要求 (CSR) が生成されます。プロンプトが表示されたら、保留中の CSR を承認してインストールを完了します。
ホストが正常にインストールされると、クラスター Web コンソールにホストとしてリストされます。

重要

新しいホストは、元のクラスターと同じ方法を使用して暗号化されます。

12.4. API を使用したホストの追加

Assisted Installer の REST API を使用して、ホストをクラスターに追加できます。

前提条件

- OpenShift Cluster Manager CLI (**ocm**) をインストールしている。
- クラスター作成権限を持つユーザーで OpenShift Cluster Manager にログインする。
- **jq** をインストールする。

- 拡張するクラスターに必要なすべての DNS レコードが存在することを確認する。

手順

1. Assisted Installer の REST API に対して認証し、セッションの API トークンを生成します。生成されたトークンは 15 分間のみ有効です。

2. 次のコマンドを実行して、**$API_URL** 変数を設定します。

```
$ export API_URL=<api_url> 1
```

1 **<api_url>** を Assisted Installer API の URL に置き換えます (例: **https://api.openshift.com**)。

3. 以下のコマンドを実行してクラスターをインポートします。

   a. **$CLUSTER_ID** 変数を設定します。クラスターにログインし、次のコマンドを実行します。

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

   b. クラスターのインポートに使用される **$CLUSTER_REQUEST** 変数を設定します。

```
$ export CLUSTER_REQUEST=$(jq --null-input --arg openshift_cluster_id
"$CLUSTER_ID" '{
  "api_vip_dnsname": "<api_vip>", 1
  "openshift_cluster_id": $CLUSTER_ID,
  "name": "<openshift_cluster_name>" 2
}')
```

1 **<api_vip>** をクラスターの API サーバーのホスト名に置き換えます。これは、API サーバーの DNS ドメイン、またはホストが到達できる単一ノードの IP アドレスになります。たとえば、**api.compute-1.example.com** です。

2 **<openshift_cluster_name>** をクラスターのプレーンテキスト名に置き換えます。クラスター名は、Day 1 クラスターのインストール中に設定されたクラスター名と一致する必要があります。

   c. クラスターをインポートし、**$CLUSTER_ID** 変数を設定します。以下のコマンドを実行します。

```
$ CLUSTER_ID=$(curl "$API_URL/api/assisted-install/v2/clusters/import" -H
"Authorization: Bearer ${API_TOKEN}" -H 'accept: application/json' -H 'Content-Type:
application/json' \
  -d "$CLUSTER_REQUEST" | tee /dev/stderr | jq -r '.id')
```

4. 次のコマンドを実行して、クラスターの **InfraEnv** リソースを生成し、**$INFRA_ENV_ID** 変数を設定します。

   a. console.redhat.com の Red Hat OpenShift Cluster Manager からプルシークレットファイルをダウンロードします。

   b. **$INFRA_ENV_REQUEST** 変数を設定します。

```
export INFRA_ENV_REQUEST=$(jq --null-input \
    --slurpfile pull_secret <path_to_pull_secret_file> \ 1
    --arg ssh_pub_key "$(cat <path_to_ssh_pub_key>)" \ 2
    --arg cluster_id "$CLUSTER_ID" '{
  "name": "<infraenv_name>", 3
  "pull_secret": $pull_secret[0] | tojson,
  "cluster_id": $cluster_id,
  "ssh_authorized_key": $ssh_pub_key,
  "image_type": "<iso_image_type>" 4
}')
```

1. **<path_to_pull_secret_file>** は、console.redhat.com の Red Hat OpenShift Cluster Manager からダウンロードしたプルシークレットを含むローカルファイルへのパスに置き換えます。
2. **<path_to_ssh_pub_key>** を、ホストへのアクセスに必要な公開 SSH キーへのパスに置き換えます。この値を設定しないと、検出モードでホストにアクセスできません。
3. **<infraenv_name>** を **InfraEnv** リソースのプレーンテキスト名に置き換えます。
4. **<iso_image_type>** を **full-iso** または **minimal-iso** のいずれかの ISO イメージタイプに置き換えます。

c. **$INFRA_ENV_REQUEST** を /v2/infra-envs API に送信し、**$INFRA_ENV_ID** 変数を設定します。

```
$ INFRA_ENV_ID=$(curl "$API_URL/api/assisted-install/v2/infra-envs" -H "Authorization: Bearer ${API_TOKEN}" -H 'accept: application/json' -H 'Content-Type: application/json' -d "$INFRA_ENV_REQUEST" | tee /dev/stderr | jq -r '.id')
```

5. 次のコマンドを実行して、クラスターホストの検出 ISO の URL を取得します。

```
$ curl -s "$API_URL/api/assisted-install/v2/infra-envs/$INFRA_ENV_ID" -H "Authorization: Bearer ${API_TOKEN}" | jq -r '.download_url'
```

出力例

```
https://api.openshift.com/api/assisted-images/images/41b91e72-c33e-42ee-b80f-b5c5bbf6431a?arch=x86_64&image_token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2NTYwMjYzNzEsInN1YiI6IjQxYjkxZTcyLWMzM2UtNDJlZS1iODBmLWI1YzViYmY2NDMxYSJ9.1EX_VGaMNejMhrAvVRBS7PDPIQtbOOc8LtG8OukE1a4&type=minimal-iso&version=4.12
```

6. ISO をダウンロードします。

```
$ curl -L -s '<iso_url>' --output rhcos-live-minimal.iso 1
```

1. **<iso_url>** を前の手順の ISO の URL に置き換えます。

7. ダウンロードした **rhcos-live-minimal.iso** から新しいワーカーホストを起動します。

8. **インストールされていない** クラスター内のホストのリストを取得します。新しいホストが表示されるまで、次のコマンドを実行し続けます。

```
$ curl -s "$API_URL/api/assisted-install/v2/clusters/$CLUSTER_ID" -H "Authorization: Bearer ${API_TOKEN}" | jq -r '.hosts[] | select(.status != "installed").id'
```

出力例

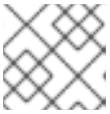
```
2294ba03-c264-4f11-ac08-2f1bb2f8c296
```

9. 新しいホストの **$HOST_ID** 変数を設定します。以下に例を示します。

```
$ HOST_ID=<host_id> 1
```

1 **<host_id>** を前の手順のホスト ID に置き換えます。

10. 以下のコマンドを実行して、ホストがインストールできる状態であることを確認します。



注記

完全な **jq** 式を含むコマンド全体をコピーしてください。

```
$ curl -s $API_URL/api/assisted-install/v2/clusters/$CLUSTER_ID -H "Authorization: Bearer ${API_TOKEN}" | jq '
def host_name($host):
    if (.suggested_hostname // "") == "" then
        if (.inventory // "") == "" then
            "Unknown hostname, please wait"
        else
            .inventory | fromjson | .hostname
        end
    else
        .suggested_hostname
    end;

def is_notable($validation):
    ["failure", "pending", "error"] | any(. == $validation.status);

def notable_validations($validations_info):
    [
        $validations_info // "{}"
        | fromjson
        | to_entries[].value[]
        | select(is_notable(.))
    ];

{
    "Hosts validations": {
        "Hosts": [
            .hosts[]
            | select(.status != "installed")
            | {
                "id": .id,
```

```
                "name": host_name(.),
                "status": .status,
                "notable_validations": notable_validations(.validations_info)
            }
        ]
    },
    "Cluster validations info": {
        "notable_validations": notable_validations(.validations_info)
    }
}
' -r
```

出力例

```
{
  "Hosts validations": {
    "Hosts": [
      {
        "id": "97ec378c-3568-460c-bc22-df54534ff08f",
        "name": "localhost.localdomain",
        "status": "insufficient",
        "notable_validations": [
          {
            "id": "ntp-synced",
            "status": "failure",
            "message": "Host couldn't synchronize with any NTP server"
          },
          {
            "id": "api-domain-name-resolved-correctly",
            "status": "error",
            "message": "Parse error for domain name resolutions result"
          },
          {
            "id": "api-int-domain-name-resolved-correctly",
            "status": "error",
            "message": "Parse error for domain name resolutions result"
          },
          {
            "id": "apps-domain-name-resolved-correctly",
            "status": "error",
            "message": "Parse error for domain name resolutions result"
          }
        ]
      }
    ]
  },
  "Cluster validations info": {
    "notable_validations": []
  }
}
```

11. 前のコマンドでホストの準備ができていることが示されたら、次のコマンドを実行し、/v2/infra-envs/{infra_env_id}/hosts/{host_id}/actions/install API を使用してインストールを開始します。

```
$ curl -X POST -s "$API_URL/api/assisted-install/v2/infra-envs/$INFRA_ENV_ID/hosts/$HOST_ID/actions/install"  -H "Authorization: Bearer ${API_TOKEN}"
```

12. インストールが進行すると、ホストに対して保留中の証明書署名要求 (CSR) が生成されます。

重要

インストールを完了するには、CSR を承認する必要があります。

次の API 呼び出しを実行し続けて、クラスターのインストールを監視します。

```
$ curl -s "$API_URL/api/assisted-install/v2/clusters/$CLUSTER_ID" -H "Authorization: Bearer ${API_TOKEN}" | jq '{
    "Cluster day-2 hosts":
        [
            .hosts[]
            | select(.status != "installed")
            | {id, requested_hostname, status, status_info, progress, status_updated_at, updated_at, infra_env_id, cluster_id, created_at}
        ]
}'
```

出力例

```
{
  "Cluster day-2 hosts": [
    {
      "id": "a1c52dde-3432-4f59-b2ae-0a530c851480",
      "requested_hostname": "control-plane-1",
      "status": "added-to-existing-cluster",
      "status_info": "Host has rebooted and no further updates will be posted. Please check console for progress and to possibly approve pending CSRs",
      "progress": {
        "current_stage": "Done",
        "installation_percentage": 100,
        "stage_started_at": "2022-07-08T10:56:20.476Z",
        "stage_updated_at": "2022-07-08T10:56:20.476Z"
      },
      "status_updated_at": "2022-07-08T10:56:20.476Z",
      "updated_at": "2022-07-08T10:57:15.306369Z",
      "infra_env_id": "b74ec0c3-d5b5-4717-a866-5b6854791bd3",
      "cluster_id": "8f721322-419d-4eed-aa5b-61b50ea586ae",
      "created_at": "2022-07-06T22:54:57.161614Z"
    }
  ]
}
```

13. オプション: 次のコマンドを実行して、クラスターのすべてのイベントを表示します。

```
$ curl -s "$API_URL/api/assisted-install/v2/events?cluster_id=$CLUSTER_ID" -H "Authorization: Bearer ${API_TOKEN}" | jq -c '.[] | {severity, message, event_time, host_id}'
```

出力例

```
{"severity":"info","message":"Host compute-0: updated status from insufficient to known (Host is ready to be installed)","event_time":"2022-07-08T11:21:46.346Z","host_id":"9d7b3b44-1125-4ad0-9b14-76550087b445"}
{"severity":"info","message":"Host compute-0: updated status from known to installing (Installation is in progress)","event_time":"2022-07-08T11:28:28.647Z","host_id":"9d7b3b44-1125-4ad0-9b14-76550087b445"}
{"severity":"info","message":"Host compute-0: updated status from installing to installing-in-progress (Starting installation)","event_time":"2022-07-08T11:28:52.068Z","host_id":"9d7b3b44-1125-4ad0-9b14-76550087b445"}
{"severity":"info","message":"Uploaded logs for host compute-0 cluster 8f721322-419d-4eed-aa5b-61b50ea586ae","event_time":"2022-07-08T11:29:47.802Z","host_id":"9d7b3b44-1125-4ad0-9b14-76550087b445"}
{"severity":"info","message":"Host compute-0: updated status from installing-in-progress to added-to-existing-cluster (Host has rebooted and no further updates will be posted. Please check console for progress and to possibly approve pending CSRs)","event_time":"2022-07-08T11:29:48.259Z","host_id":"9d7b3b44-1125-4ad0-9b14-76550087b445"}
{"severity":"info","message":"Host: compute-0, reached installation stage Rebooting","event_time":"2022-07-08T11:29:48.261Z","host_id":"9d7b3b44-1125-4ad0-9b14-76550087b445"}
```

14. クラスターにログインし、保留中の CSR を承認してインストールを完了します。

検証

- 新規ホストが **Ready** のステータスでクラスターに正常に追加されたことを確認します。

```
$ oc get nodes
```

出力例

```
NAME                         STATUS   ROLES          AGE   VERSION
control-plane-1.example.com  Ready    master,worker  56m   v1.25.0
compute-1.example.com        Ready    worker         11m   v1.25.0
```

12.5. 正常なクラスターへのプライマリーコントロールプレーンノードのインストール

この手順では、プライマリーコントロールプレーンノードを正常な OpenShift Container Platform クラスターにインストールする方法について説明します。

クラスターが正常でない場合、管理する前に追加の操作が必要になります。詳細は、**関連情報** を参照してください。

前提条件

- 少なくとも 3 つのノードを含む正常なクラスターがインストール済みである。
- シングルノードに **role: master** を 割り当て済み である。

手順

1. 保留中の **CertificateSigningRequests** (CSR) を取得します。

```
$ oc get csr | grep Pending
```

出力例

```
csr-5sd59   8m19s   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   <none>
Pending
csr-xzqts   10s     kubernetes.io/kubelet-serving                 system:node:worker-6
<none>              Pending
```

2. 保留中の CSR を承認します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}{{end}}
{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

重要

インストールを完了するには、CSR を承認する必要があります。

3. プライマリーノードが **Ready** ステータスであることを確認します。

```
$ oc get nodes
```

出力例

```
NAME       STATUS   ROLES    AGE     VERSION
master-0   Ready    master   4h42m   v1.24.0+3882f8f
worker-1   Ready    worker   4h29m   v1.24.0+3882f8f
master-2   Ready    master   4h43m   v1.24.0+3882f8f
master-3   Ready    master   4h27m   v1.24.0+3882f8f
worker-4   Ready    worker   4h30m   v1.24.0+3882f8f
master-5   Ready    master   105s    v1.24.0+3882f8f
```

注記

機能する Machine API を使用してクラスターを実行する場合、**etcd-operator** に、新しいノードを参照する **Machine** カスタムリソース (CR) が必要です。

4. **Machine** CR を **BareMetalHost** および **Node** にリンクします。

   a. 一意の **.metadata.name** の値を使用して **BareMetalHost** CR を作成します。

```
apiVersion: metal3.io/v1alpha1
kind: BareMetalHost
metadata:
  name: custom-master3
  namespace: openshift-machine-api
  annotations:
spec:
  automatedCleaningMode: metadata
```

```
  bootMACAddress: 00:00:00:00:00:02
  bootMode: UEFI
  customDeploy:
    method: install_coreos
  externallyProvisioned: true
  online: true
  userData:
    name: master-user-data-managed
    namespace: openshift-machine-api
```

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

b. **BareMetalHost** CR を適用します。

```
$ oc apply -f <filename>
```

c. 一意の **.machine.name** 値を使用して **Machine** CR を作成します。

```
apiVersion: machine.openshift.io/v1beta1
kind: Machine
metadata:
  annotations:
    machine.openshift.io/instance-state: externally provisioned
    metal3.io/BareMetalHost: openshift-machine-api/custom-master3
  finalizers:
  - machine.machine.openshift.io
  generation: 3
  labels:
    machine.openshift.io/cluster-api-cluster: test-day2-1-6qv96
    machine.openshift.io/cluster-api-machine-role: master
    machine.openshift.io/cluster-api-machine-type: master
  name: custom-master3
  namespace: openshift-machine-api
spec:
  metadata: {}
  providerSpec:
    value:
      apiVersion: baremetal.cluster.k8s.io/v1alpha1
      customDeploy:
        method: install_coreos
      hostSelector: {}
      image:
        checksum: ""
        url: ""
      kind: BareMetalMachineProviderSpec
      metadata:
        creationTimestamp: null
      userData:
        name: master-user-data-managed
```

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

d. **Machine** CR を適用します。

```
$ oc apply -f <filename>
```

e. **link-machine-and-node.sh** スクリプトを使用して、**BareMetalHost**、**Machine**、および **Node** をリンクします。

```
#!/bin/bash

# Credit goes to https://bugzilla.redhat.com/show_bug.cgi?id=1801238.
# This script will link Machine object and Node object. This is needed
# in order to have IP address of the Node present in the status of the Machine.

# set -x
set -e

machine="$1"
node="$2"

if [ -z "$machine" ] || [ -z "$node" ]; then
    echo "Usage: $0 MACHINE NODE"
    exit 1
fi

# uid=$(echo "${node}" | cut -f1 -d':')
node_name=$(echo "${node}" | cut -f2 -d':')

oc proxy &
proxy_pid=$!
function kill_proxy {
    kill $proxy_pid
}
trap kill_proxy EXIT SIGINT

HOST_PROXY_API_PATH="http://localhost:8001/apis/metal3.io/v1alpha1/namespaces/op
enshift-machine-api/baremetalhosts"

function print_nics() {
    local ips
    local eob
    declare -a ips

    readarray -t ips < <(echo "${1}" \
                        | jq '.[] | select(. | .type == "InternalIP") | .address' \
                        | sed 's/"//g')

    eob=','
    for (( i=0; i<${#ips[@]}; i++ )); do
        if [ $((i+1)) -eq ${#ips[@]} ]; then
            eob=""
        fi
        cat <<- EOF
          {
            "ip": "${ips[$i]}",
            "mac": "00:00:00:00:00:00",
            "model": "unknown",
            "speedGbps": 10,
```

```
            "vlanId": 0,
            "pxe": true,
            "name": "eth1"
          }${eob}
EOF
    done
}

function wait_for_json() {
    local name
    local url
    local curl_opts
    local timeout

    local start_time
    local curr_time
    local time_diff

    name="$1"
    url="$2"
    timeout="$3"
    shift 3
    curl_opts="$@"
    echo -n "Waiting for $name to respond"
    start_time=$(date +%s)
    until curl -g -X GET "$url" "${curl_opts[@]}" 2> /dev/null | jq '.' 2> /dev/null > /dev/null;
do
        echo -n "."
        curr_time=$(date +%s)
        time_diff=$((curr_time - start_time))
        if [[ $time_diff -gt $timeout ]]; then
            printf '\nTimed out waiting for %s' "${name}"
            return 1
        fi
        sleep 5
    done
    echo " Success!"
    return 0
}
wait_for_json oc_proxy "${HOST_PROXY_API_PATH}" 10 -H "Accept: application/json"
-H "Content-Type: application/json"

addresses=$(oc get node -n openshift-machine-api "${node_name}" -o json | jq -c
'.status.addresses')

machine_data=$(oc get machines.machine.openshift.io -n openshift-machine-api -o json
"${machine}")
host=$(echo "$machine_data" | jq '.metadata.annotations["metal3.io/BareMetalHost"]' |
cut -f2 -d/ | sed 's/"//g')

if [ -z "$host" ]; then
    echo "Machine $machine is not linked to a host yet." 1>&2
    exit 1
fi

# The address structure on the host doesn't match the node, so extract
```

```
# the values we want into separate variables so we can build the patch
# we need.
hostname=$(echo "${addresses}" | jq '.[] | select(. | .type == "Hostname") | .address' | sed
's/"//g')

set +e
read -r -d '' host_patch << EOF
{
  "status": {
    "hardware": {
      "hostname": "${hostname}",
      "nics": [
$(print_nics "${addresses}")
      ],
      "systemVendor": {
        "manufacturer": "Red Hat",
        "productName": "product name",
        "serialNumber": ""
      },
      "firmware": {
        "bios": {
          "date": "04/01/2014",
          "vendor": "SeaBIOS",
          "version": "1.11.0-2.el7"
        }
      },
      "ramMebibytes": 0,
      "storage": [],
      "cpu": {
        "arch": "x86_64",
        "model": "Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz",
        "clockMegahertz": 2199.998,
        "count": 4,
        "flags": []
      }
    }
  }
}
EOF
set -e

echo "PATCHING HOST"
echo "${host_patch}" | jq .

curl -s \
    -X PATCH \
    "${HOST_PROXY_API_PATH}/${host}/status" \
    -H "Content-type: application/merge-patch+json" \
    -d "${host_patch}"

oc get baremetalhost -n openshift-machine-api -o yaml "${host}"
```

```
$ bash link-machine-and-node.sh custom-master3 worker-5
```

5. **etcd** メンバーを確認します。

-

```
$ oc rsh -n openshift-etcd etcd-worker-2
etcdctl member list -w table
```

出力例

```
+--------+---------+--------+-------------+-------------+---------+
|   ID   | STATUS |  NAME  |  PEER ADDRS  | CLIENT ADDRS | LEARNER |
+--------+---------+--------+-------------+-------------+---------+
|2c18942f| started |worker-3|192.168.111.26|192.168.111.26|  false  |
|61e2a860| started |worker-2|192.168.111.25|192.168.111.25|  false  |
|ead4f280| started |worker-5|192.168.111.28|192.168.111.28|  false  |
+--------+---------+--------+-------------+-------------+---------+
```

6. **etcd-operator** 設定がすべてのノードに適用されていることを確認します。

```
$ oc get clusteroperator etcd
```

出力例

```
NAME   VERSION   AVAILABLE   PROGRESSING   DEGRADED   SINCE   MESSAGE
etcd   4.11.5    True        False         False      5h54m
```

7. **etcd-operator** の正常性を確認します。

```
$ oc rsh -n openshift-etcd etcd-worker-0
etcdctl endpoint health
```

出力例

```
192.168.111.26 is healthy: committed proposal: took = 11.297561ms
192.168.111.25 is healthy: committed proposal: took = 13.892416ms
192.168.111.28 is healthy: committed proposal: took = 11.870755ms
```

8. ノードの正常性を確認します。

```
$ oc get nodes
```

出力例

```
NAME       STATUS   ROLES    AGE     VERSION
master-0   Ready    master   6h20m   v1.24.0+3882f8f
worker-1   Ready    worker   6h7m    v1.24.0+3882f8f
master-2   Ready    master   6h20m   v1.24.0+3882f8f
master-3   Ready    master   6h4m    v1.24.0+3882f8f
worker-4   Ready    worker   6h7m    v1.24.0+3882f8f
master-5   Ready    master   99m     v1.24.0+3882f8f
```

9. **ClusterOperators** の正常性を確認します。

```
$ oc get ClusterOperators
```

出力例

```
NAME                                       VERSION AVAILABLE PROGRESSING DEGRADED SINCE
MSG
authentication                             4.11.5  True      False       False    5h57m
baremetal                                  4.11.5  True      False       False    6h19m
cloud-controller-manager                   4.11.5  True      False       False    6h20m
cloud-credential                           4.11.5  True      False       False    6h23m
cluster-autoscaler                         4.11.5  True      False       False    6h18m
config-operator                            4.11.5  True      False       False    6h19m
console                                    4.11.5  True      False       False    6h4m
csi-snapshot-controller                    4.11.5  True      False       False    6h19m
dns                                        4.11.5  True      False       False    6h18m
etcd                                       4.11.5  True      False       False    6h17m
image-registry                             4.11.5  True      False       False    6h7m
ingress                                    4.11.5  True      False       False    6h6m
insights                                   4.11.5  True      False       False    6h12m
kube-apiserver                             4.11.5  True      False       False    6h16m
kube-controller-manager                    4.11.5  True      False       False    6h16m
kube-scheduler                             4.11.5  True      False       False    6h16m
kube-storage-version-migrator              4.11.5  True      False       False    6h19m
machine-api                                4.11.5  True      False       False    6h15m
machine-approver                           4.11.5  True      False       False    6h19m
machine-config                             4.11.5  True      False       False    6h18m
marketplace                                4.11.5  True      False       False    6h18m
monitoring                                 4.11.5  True      False       False    6h4m
network                                    4.11.5  True      False       False    6h20m
node-tuning                                4.11.5  True      False       False    6h18m
openshift-apiserver                        4.11.5  True      False       False    6h8m
openshift-controller-manager               4.11.5  True      False       False    6h7m
openshift-samples                          4.11.5  True      False       False    6h12m
operator-lifecycle-manager                 4.11.5  True      False       False    6h18m
operator-lifecycle-manager-catalog         4.11.5  True      False       False    6h19m
operator-lifecycle-manager-pkgsvr          4.11.5  True      False       False    6h12m
service-ca                                 4.11.5  True      False       False    6h19m
storage                                    4.11.5  True      False       False    6h19m
```

10. **ClusterVersion** を確認します。

```
$ oc get ClusterVersion
```

出力例

```
NAME      VERSION   AVAILABLE   PROGRESSING   SINCE   STATUS
version   4.11.5    True        False         5h57m   Cluster version is 4.11.5
```

11. 古いコントロールプレーンノードを削除します。

    a. **BareMetalHost** CR を削除します。

    ```
    $ oc delete bmh -n openshift-machine-api custom-master3
    ```

    b. **Machine** が正常でないことを確認します。

```
$ oc get machine -A
```

出力例

```
NAMESPACE               NAME                              PHASE    AGE
openshift-machine-api   custom-master3                    Running  14h
openshift-machine-api   test-day2-1-6qv96-master-0        Failed   20h
openshift-machine-api   test-day2-1-6qv96-master-1        Running  20h
openshift-machine-api   test-day2-1-6qv96-master-2        Running  20h
openshift-machine-api   test-day2-1-6qv96-worker-0-8w7vr  Running  19h
openshift-machine-api   test-day2-1-6qv96-worker-0-rxddj  Running  19h
```

c. **Machine** CR を削除します。

```
$ oc delete machine -n openshift-machine-api   test-day2-1-6qv96-master-0
machine.machine.openshift.io "test-day2-1-6qv96-master-0" deleted
```

d. **Node** CR の削除を確認します。

```
$ oc get nodes
```

出力例

```
NAME       STATUS   ROLES    AGE   VERSION
worker-1   Ready    worker   19h   v1.24.0+3882f8f
master-2   Ready    master   20h   v1.24.0+3882f8f
master-3   Ready    master   19h   v1.24.0+3882f8f
worker-4   Ready    worker   19h   v1.24.0+3882f8f
master-5   Ready    master   15h   v1.24.0+3882f8f
```

12. **etcd-operator** ログをチェックして、**etcd** クラスターのステータスを確認します。

```
$ oc logs -n openshift-etcd-operator etcd-operator-8668df65d-lvpjf
```

出力例

```
E0927 07:53:10.597523       1 base_controller.go:272] ClusterMemberRemovalController
reconciliation failed: cannot remove member: 192.168.111.23 because it is reported as
healthy but it doesn't have a machine nor a node resource
```

13. 物理マシンを削除して、**etcd-operator** がクラスターメンバーを調整できるようにします。

```
$ oc rsh -n openshift-etcd etcd-worker-2
etcdctl member list -w table; etcdctl endpoint health
```

出力例

```
+--------+---------+--------+--------------+--------------+---------+
|   ID   | STATUS |  NAME  |  PEER ADDRS  | CLIENT ADDRS | LEARNER |
+--------+---------+--------+--------------+--------------+---------+
|2c18942f| started |worker-3|192.168.111.26|192.168.111.26|  false  |
|61e2a860| started |worker-2|192.168.111.25|192.168.111.25|  false  |
```

```
|ead4f280| started |worker-5|192.168.111.28|192.168.111.28|  false  |
+--------+---------+--------+--------------+--------------+---------+
192.168.111.26 is healthy: committed proposal: took = 10.458132ms
192.168.111.25 is healthy: committed proposal: took = 11.047349ms
192.168.111.28 is healthy: committed proposal: took = 11.414402ms
```

関連情報

- 正常でないクラスターへのプライマリーコントロールプレーンノードのインストール

12.6. 正常でないクラスターへのプライマリーコントロールプレーンノードのインストール

この手順では、正常でない OpenShift Container Platform クラスターにプライマリーコントロールプレーンノードをインストールする方法について説明します。

前提条件

- 少なくとも 3 つのノードを含む正常なクラスターがインストール済みである。
- コントロールプレーンがインストール済みである。
- シングルノードに **role: master** を 割り当て済み である。

手順

1. クラスターの初期状態を確認します。

```
$ oc get nodes
```

出力例

```
NAME       STATUS     ROLES    AGE   VERSION
worker-1   Ready      worker   20h   v1.24.0+3882f8f
master-2   NotReady   master   20h   v1.24.0+3882f8f
master-3   Ready      master   20h   v1.24.0+3882f8f
worker-4   Ready      worker   20h   v1.24.0+3882f8f
master-5   Ready      master   15h   v1.24.0+3882f8f
```

2. **etcd-operator** がクラスターを正常ではないと検出していることを確認します。

```
$ oc logs -n openshift-etcd-operator etcd-operator-8668df65d-lvpjf
```

出力例

```
E0927 08:24:23.983733       1 base_controller.go:272] DefragController reconciliation failed:
cluster is unhealthy: 2 of 3 members are available, worker-2 is unhealthy
```

3. **etcdctl** メンバーを確認します。

```
$ oc rsh -n openshift-etcd etcd-worker-3
etcdctl member list -w table
```

出力例

```
+--------+---------+--------+-------------+-------------+---------+
|   ID   | STATUS  |  NAME  |  PEER ADDRS  | CLIENT ADDRS | LEARNER |
+--------+---------+--------+-------------+-------------+---------+
|2c18942f| started |worker-3|192.168.111.26|192.168.111.26|  false  |
|61e2a860| started |worker-2|192.168.111.25|192.168.111.25|  false  |
|ead4f280| started |worker-5|192.168.111.28|192.168.111.28|  false  |
+--------+---------+--------+-------------+-------------+---------+
```

4. **etcdctl** がクラスターの正常でないメンバーを報告していることを確認します。

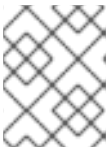
```
$ etcdctl endpoint health
```

出力例

```
{"level":"warn","ts":"2022-09-
27T08:25:35.953Z","logger":"client","caller":"v3/retry_interceptor.go:62","msg":"retrying of
unary invoker failed","target":"etcd-
endpoints://0xc000680380/192.168.111.25","attempt":0,"error":"rpc error: code =
DeadlineExceeded desc = latest balancer error: last connection error: connection error: desc
= \"transport: Error while dialing dial tcp 192.168.111.25: connect: no route to host\""}
192.168.111.28 is healthy: committed proposal: took = 12.465641ms
192.168.111.26 is healthy: committed proposal: took = 12.297059ms
192.168.111.25 is unhealthy: failed to commit proposal: context deadline exceeded
Error: unhealthy cluster
```

5. **Machine** カスタムリソースを削除して、正常でないコントロールプレーンを削除します。

```
$ oc delete machine -n openshift-machine-api test-day2-1-6qv96-master-2
```



注記

正常でないクラスターを正常に実行できない場合、**Machine** および **Node** のカスタムリソース (CR) は削除されません。

6. **etcd-operator** が正常でないマシンを削除していないことを確認します。

```
$ oc logs -n openshift-etcd-operator etcd-operator-8668df65d-lvpjf -f
```

出力例

```
I0927 08:58:41.249222       1 machinedeletionhooks.go:135] skip removing the deletion hook
from machine test-day2-1-6qv96-master-2 since its member is still present with any of:
[{InternalIP } {InternalIP 192.168.111.26}]
```

7. 正常でない **etcdctl** メンバーを手動で削除します。

```
$ oc rsh -n openshift-etcd etcd-worker-3\
etcdctl member list -w table
```

出力例

```
+--------+---------+--------+-------------+-------------+---------+
|   ID   | STATUS |  NAME  |  PEER ADDRS  | CLIENT ADDRS | LEARNER |
+--------+---------+--------+-------------+-------------+---------+
|2c18942f| started |worker-3|192.168.111.26|192.168.111.26|  false  |
|61e2a860| started |worker-2|192.168.111.25|192.168.111.25|  false  |
|ead4f280| started |worker-5|192.168.111.28|192.168.111.28|  false  |
+--------+---------+--------+-------------+-------------+---------+
```

8. **etcdctl** がクラスターの正常でないメンバーを報告していることを確認します。

```
$ etcdctl endpoint health
```

出力例

```
{"level":"warn","ts":"2022-09-
27T10:31:07.227Z","logger":"client","caller":"v3/retry_interceptor.go:62","msg":"retrying of
unary invoker failed","target":"etcd-
endpoints://0xc0000d6e00/192.168.111.25","attempt":0,"error":"rpc error: code =
DeadlineExceeded desc = latest balancer error: last connection error: connection error: desc
= \"transport: Error while dialing dial tcp 192.168.111.25: connect: no route to host\""}
192.168.111.28 is healthy: committed proposal: took = 13.038278ms
192.168.111.26 is healthy: committed proposal: took = 12.950355ms
192.168.111.25 is unhealthy: failed to commit proposal: context deadline exceeded
Error: unhealthy cluster
```

9. **etcdctl** メンバーのカスタムリソースを削除して正常でないクラスターを削除します。

```
$ etcdctl member remove 61e2a86084aafa62
```

出力例

```
Member 61e2a86084aafa62 removed from cluster 6881c977b97990d7
```

10. 次のコマンドを実行して **etcdctl** のメンバーを確認します。

```
$ etcdctl member list -w table
```

出力例

```
+----------+---------+--------+-------------+-------------+-------+
|    ID    | STATUS  |  NAME  |  PEER ADDRS  | CLIENT ADDRS |LEARNER|
+----------+---------+--------+-------------+-------------+-------+
| 2c18942f | started |worker-3|192.168.111.26|192.168.111.26| false |
| ead4f280 | started |worker-5|192.168.111.28|192.168.111.28| false |
+----------+---------+--------+-------------+-------------+-------+
```

11. 証明書署名要求を確認して承認します。

    a. 証明書署名要求 (CSR) を確認します。

```
$ oc get csr | grep Pending
```

出力例

```
csr-5sd59   8m19s   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   <none>
Pending
csr-xzqts   10s     kubernetes.io/kubelet-serving                 system:node:worker-6
<none>              Pending
```

b. 保留中の 全 CSR を承認します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

注記

インストールを完了するには、CSR を承認する必要があります。

12. コントロールプレーンノードが Ready ステータスであることを確認します。

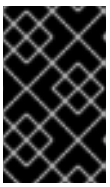
```
$ oc get nodes
```

出力例

```
NAME       STATUS   ROLES    AGE     VERSION
worker-1   Ready    worker   22h     v1.24.0+3882f8f
master-3   Ready    master   22h     v1.24.0+3882f8f
worker-4   Ready    worker   22h     v1.24.0+3882f8f
master-5   Ready    master   17h     v1.24.0+3882f8f
master-6   Ready    master   2m52s   v1.24.0+3882f8f
```

13. **Machine**、**Node**、および **BareMetalHost** カスタムリソースを検証します。
機能する Machine API を使用してクラスターを実行する場合、**etcd-operator** に **Machine** CR が存在する必要があります。**Machine** CR は、存在する場合は **Running** フェーズ中に表示されます。

14. **BareMetalHost** および **Node** にリンクされた **Machine** カスタムリソースを作成します。
新しく追加されたノードを参照する **Machine** CR があることを確認します。



重要

Boot-it-yourself では **BareMetalHost** および **Machine** CR は作成されないため、これらを作成する必要があります。**BareMetalHost** および **Machine** CR の作成に失敗すると、**etcd-operator** の実行時にエラーが生成されます。

15. **BareMetalHost** カスタムリソースを追加します。

```
$ oc create bmh -n openshift-machine-api custom-master3
```

16. **Machine** カスタムリソースを追加します。

```
$ oc create machine -n openshift-machine-api custom-master3
```

17. **link-machine-and-node.sh** スクリプトを実行して、**BareMetalHost**、**Machine**、および **Node** をリンクします。

```
#!/bin/bash

# Credit goes to https://bugzilla.redhat.com/show_bug.cgi?id=1801238.
# This script will link Machine object and Node object. This is needed
# in order to have IP address of the Node present in the status of the Machine.

# set -x
set -e

machine="$1"
node="$2"

if [ -z "$machine" ] || [ -z "$node" ]; then
    echo "Usage: $0 MACHINE NODE"
    exit 1
fi

# uid=$(echo "${node}" | cut -f1 -d':')
node_name=$(echo "${node}" | cut -f2 -d':')

oc proxy &
proxy_pid=$!
function kill_proxy {
    kill $proxy_pid
}
trap kill_proxy EXIT SIGINT

HOST_PROXY_API_PATH="http://localhost:8001/apis/metal3.io/v1alpha1/namespaces/opensh
ift-machine-api/baremetalhosts"

function print_nics() {
    local ips
    local eob
    declare -a ips

    readarray -t ips < <(echo "${1}" \
                        | jq '.[] | select(. | .type == "InternalIP") | .address' \
                        | sed 's/"//g')

    eob=','
    for (( i=0; i<${#ips[@]}; i++ )); do
        if [ $((i+1)) -eq ${#ips[@]} ]; then
            eob=""
        fi
        cat <<- EOF
          {
            "ip": "${ips[$i]}",
            "mac": "00:00:00:00:00:00",
            "model": "unknown",
            "speedGbps": 10,
            "vlanId": 0,
            "pxe": true,
            "name": "eth1"
```

```
        }${eob}
EOF
    done
}

function wait_for_json() {
    local name
    local url
    local curl_opts
    local timeout

    local start_time
    local curr_time
    local time_diff

    name="$1"
    url="$2"
    timeout="$3"
    shift 3
    curl_opts="$@"
    echo -n "Waiting for $name to respond"
    start_time=$(date +%s)
    until curl -g -X GET "$url" "${curl_opts[@]}" 2> /dev/null | jq '.' 2> /dev/null > /dev/null; do
        echo -n "."
        curr_time=$(date +%s)
        time_diff=$((curr_time - start_time))
        if [[ $time_diff -gt $timeout ]]; then
            printf '\nTimed out waiting for %s' "${name}"
            return 1
        fi
        sleep 5
    done
    echo " Success!"
    return 0
}
wait_for_json oc_proxy "${HOST_PROXY_API_PATH}" 10 -H "Accept: application/json" -H
"Content-Type: application/json"

addresses=$(oc get node -n openshift-machine-api "${node_name}" -o json | jq -c
'.status.addresses')

machine_data=$(oc get machines.machine.openshift.io -n openshift-machine-api -o json
"${machine}")
host=$(echo "$machine_data" | jq '.metadata.annotations["metal3.io/BareMetalHost"]' | cut -
f2 -d/ | sed 's/"//g')

if [ -z "$host" ]; then
    echo "Machine $machine is not linked to a host yet." 1>&2
    exit 1
fi

# The address structure on the host doesn't match the node, so extract
# the values we want into separate variables so we can build the patch
# we need.
hostname=$(echo "${addresses}" | jq '.[] | select(. | .type == "Hostname") | .address' | sed
's/"//g')
```

```
set +e
read -r -d '' host_patch << EOF
{
  "status": {
    "hardware": {
      "hostname": "${hostname}",
      "nics": [
$(print_nics "${addresses}")
      ],
      "systemVendor": {
        "manufacturer": "Red Hat",
        "productName": "product name",
        "serialNumber": ""
      },
      "firmware": {
        "bios": {
          "date": "04/01/2014",
          "vendor": "SeaBIOS",
          "version": "1.11.0-2.el7"
        }
      },
      "ramMebibytes": 0,
      "storage": [],
      "cpu": {
        "arch": "x86_64",
        "model": "Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz",
        "clockMegahertz": 2199.998,
        "count": 4,
        "flags": []
      }
    }
  }
}
EOF
set -e

echo "PATCHING HOST"
echo "${host_patch}" | jq .

curl -s \
    -X PATCH \
    "${HOST_PROXY_API_PATH}/${host}/status" \
    -H "Content-type: application/merge-patch+json" \
    -d "${host_patch}"

oc get baremetalhost -n openshift-machine-api -o yaml "${host}"
```

```
$ bash link-machine-and-node.sh custom-master3 worker-3
```

18. 次のコマンドを実行して **etcdctl** のメンバーを確認します。

```
$ oc rsh -n openshift-etcd etcd-worker-3
etcdctl member list -w table
```

出力例

```
+---------+-------+--------+-------------+-------------+-------+
|   ID    | STATUS|  NAME  |   PEER ADDRS | CLIENT ADDRS |LEARNER|
+---------+-------+--------+-------------+-------------+-------+
| 2c18942f|started|worker-3|192.168.111.26|192.168.111.26| false |
| ead4f280|started|worker-5|192.168.111.28|192.168.111.28| false |
| 79153c5a|started|worker-6|192.168.111.29|192.168.111.29| false |
+---------+-------+--------+-------------+-------------+-------+
```

19. **etcd** Operator がすべてのノードを設定したことを確認します。

```
$ oc get clusteroperator etcd
```

出力例

```
NAME   VERSION   AVAILABLE   PROGRESSING   DEGRADED   SINCE
etcd   4.11.5    True        False         False      22h
```

20. **etcdctl** の正常性を確認します。

```
$ oc rsh -n openshift-etcd etcd-worker-3
etcdctl endpoint health
```

出力例

```
192.168.111.26 is healthy: committed proposal: took = 9.105375ms
192.168.111.28 is healthy: committed proposal: took = 9.15205ms
192.168.111.29 is healthy: committed proposal: took = 10.277577ms
```

21. ノードの正常性を確認します。

```
$ oc get Nodes
```

出力例

```
NAME       STATUS   ROLES    AGE   VERSION
worker-1   Ready    worker   22h   v1.24.0+3882f8f
master-3   Ready    master   22h   v1.24.0+3882f8f
worker-4   Ready    worker   22h   v1.24.0+3882f8f
master-5   Ready    master   18h   v1.24.0+3882f8f
master-6   Ready    master   40m   v1.24.0+3882f8f
```

22. **ClusterOperators** の正常性を確認します。

```
$ oc get ClusterOperators
```

出力例

```
NAME                       VERSION   AVAILABLE   PROGRESSING   DEGRADED   SINCE
authentication              4.11.5    True        False         False      150m
```

```
baremetal                  4.11.5    True        False        False      22h
cloud-controller-manager   4.11.5    True        False        False      22h
cloud-credential           4.11.5    True        False        False      22h
cluster-autoscaler         4.11.5    True        False        False      22h
config-operator            4.11.5    True        False        False      22h
console                    4.11.5    True        False        False      145m
csi-snapshot-controller    4.11.5    True        False        False      22h
dns                        4.11.5    True        False        False      22h
etcd                       4.11.5    True        False        False      22h
image-registry             4.11.5    True        False        False      22h
ingress                    4.11.5    True        False        False      22h
insights                   4.11.5    True        False        False      22h
kube-apiserver             4.11.5    True        False        False      22h
kube-controller-manager    4.11.5    True        False        False      22h
kube-scheduler             4.11.5    True        False        False      22h
kube-storage-version-migrator  4.11.5    True        False        False      148m
machine-api                4.11.5    True        False        False      22h
machine-approver           4.11.5    True        False        False      22h
machine-config             4.11.5    True        False        False      110m
marketplace                4.11.5    True        False        False      22h
monitoring                 4.11.5    True        False        False      22h
network                    4.11.5    True        False        False      22h
node-tuning                4.11.5    True        False        False      22h
openshift-apiserver        4.11.5    True        False        False      163m
openshift-controller-manager  4.11.5    True        False        False      22h
openshift-samples          4.11.5    True        False        False      22h
operator-lifecycle-manager  4.11.5    True        False        False      22h
operator-lifecycle-manager-catalog 4.11.5    True        False        False      22h
operator-lifecycle-manager-pkgsvr  4.11.5    True        False        False      22h
service-ca                 4.11.5    True        False        False      22h
storage                    4.11.5    True        False        False      22h
```

23. **ClusterVersion** を確認します。

```
$ oc get ClusterVersion
```

出力例

```
NAME      VERSION   AVAILABLE   PROGRESSING   SINCE   STATUS
version   4.11.5    True        False         22h     Cluster version is 4.11.5
```

12.7. 関連情報

- 正常なクラスターへのプライマリーコントロールプレーンノードのインストール
- REST API を使用した認証

第13章 オプション: NUTANIX へのインストール

OpenShift Container Platform を Nutanix にインストールする場合、Assisted Installer は OpenShift Container Platform クラスターを Nutanix プラットフォームと統合できます。これにより、Machine API が Nutanix に公開され、Nutanix Container Storage Interface (CSI) を使用したストレージコンテナーの自動スケーリングと動的プロビジョニングが可能になります。

重要

OpenShift Container Platform クラスターをデプロイし、日常的な運用を維持するには、必要な環境要件を備えた Nutanix アカウントにアクセスする必要があります。詳細は、環境要件 を参照してください。

13.1. UI を使用した NUTANIX へのホストの追加

ユーザーインターフェイス (UI) を使用して Nutanix にホストを追加するには、Assisted Installer から検出イメージ ISO を生成します。最小限の検出イメージ ISO を使用します。これはデフォルト設定です。このイメージには、ネットワークを使用してホストを起動するために必要なものだけが含まれています。コンテンツの大部分は、起動時にダウンロードされます。ISO イメージのサイズは約 100MB です。

これが完了したら、Nutanix プラットフォームのイメージを作成し、Nutanix 仮想マシンを作成する必要があります。

前提条件

- Assisted Installer の UI でクラスタープロファイルを作成している。
- Nutanix クラスター環境をセットアップし、クラスター名とサブネット名を書き留めた。

手順

1. **Cluster details** の **Integrate with external partner platforms** ドロップダウンリストから Nutanix を選択します。**Include custom manifest**チェックボックスはオプションです。
2. ホストの検出で、ホストの追加ボタンをクリックします。
3. オプション: **core** ユーザーとして Nutanix 仮想マシンに接続できるように、SSH 公開鍵を追加します。クラスターホストにログインすると、インストール中にデバッグ情報を入手できます。
   a. ローカルマシンに既存の SSH キーペアがない場合は、クラスターノード SSH アクセス用のキーペアの生成 の手順に従います。
   b. **SSH public key** フィールドで **Browse** をクリックして、SSH 公開鍵を含む **id_rsa.pub** ファイルをアップロードします。あるいは、ファイルマネージャーからファイルをフィールドにドラッグアンドドロップします。ファイルマネージャーでファイルを表示するには、メニューで **Show hidden files** を選択します。
4. 目的のプロビジョニングタイプを選択します。

注記

Minimal image file: Provision with virtual mediaでは、起動に必要なデータをフェッチする小さなイメージがダウンロードされます。

5. **Networking** で、**Cluster-managed networking** を選択します。Nutanix は **ユーザーが管理するネットワーク** をサポートしていません。

   a. オプション: クラスターホストがプロキシーの使用を必要とするファイアウォールの内側にある場合は、**Configure cluster-wide proxy settings** を選択します。プロキシーサーバーの HTTP および HTTPS URL のユーザー名、パスワード、IP アドレス、およびポートを入力します。

   b. オプション: Ignition ファイルを使用して起動する場合は、検出イメージを設定します。詳しくは、検出イメージの設定 を参照してください。

6. **Generate Discovery ISO** をクリックします。

7. **Discovery ISO URL** をコピーします。

8. Nutanix Prism UI で、指示に従って Assisted Installer から検出イメージをアップロード します。

9. Nutanix Prism UI で、Prism Central を介して コントロールプレーン (マスター) 仮想マシンを作成します。

   a. **Name** を入力します。たとえば、**control-plane** または **master** と入力します。

   b. **Number of VMs** を入力します。これは、コントロールプレーンの場合は 3 である必要があります。

   c. 残りの設定がコントロールプレーンホストの最小要件を満たしていることを確認します。

10. Nutanix Prism UI で、Prism Central を介して ワーカー仮想マシンを作成します。

    a. **Name** を入力します。たとえば、**worker** と入力します。

    b. **Number of VMs** を入力します。少なくとも 2 つのワーカーノードを作成する必要があります。

    c. 残りの設定がワーカーホストの最小要件を満たしていることを確認します。

11. Assisted Installer のユーザーインターフェイスに戻り、Assisted Installer がホストを検出し、それぞれが **Ready** ステータスになるまで待ちます。

12. インストール手順を続行します。

13.2. API を使用した NUTANIX へのホストの追加

API を使用して Nutanix にホストを追加するには、Assisted Installer から検出イメージ ISO を生成します。最小限の検出イメージ ISO を使用します。これはデフォルト設定です。このイメージには、ネットワークを使用してホストを起動するために必要なものだけが含まれています。コンテンツの大部分は、起動時にダウンロードされます。ISO イメージのサイズは約 100MB です。

これが完了したら、Nutanix プラットフォームのイメージを作成し、Nutanix 仮想マシンを作成する必要があります。

前提条件

- Assisted Installer API の認証を設定している。
- Assisted Installer のクラスタープロファイルを作成している。

- Assisted Installer のインフラストラクチャー環境を作成している。
- インフラストラクチャー環境 ID がシェルに **$INFRA_ENV_ID** としてエクスポートされている。
- Assisted Installer のクラスター設定を完了している。
- Nutanix クラスター環境をセットアップし、クラスター名とサブネット名を書き留めた。

手順

1. Ignition ファイルを使用して起動する場合は、検出イメージを設定します。
2. 環境変数を保持する Nutanix クラスター設定ファイルを作成します。

```
$ touch ~/nutanix-cluster-env.sh
```

```
$ chmod +x ~/nutanix-cluster-env.sh
```

新しいターミナルセッションを開始する必要がある場合は、環境変数を簡単に再読み込みできます。以下に例を示します。

```
$ source ~/nutanix-cluster-env.sh
```

3. Nutanix クラスターの名前を設定ファイルの **NTX_CLUSTER_NAME** 環境変数に割り当てます。

```
$ cat << EOF >> ~/nutanix-cluster-env.sh
export NTX_CLUSTER_NAME=<cluster_name>
EOF
```

<cluster_name> を Nutanix クラスターの名前に置き換えます。

4. Nutanix クラスターのサブネット名を設定ファイルの **NTX_SUBNET_NAME** 環境変数に割り当てます。

```
$ cat << EOF >> ~/nutanix-cluster-env.sh
export NTX_SUBNET_NAME=<subnet_name>
EOF
```

<subnet_name> を Nutanix クラスターのサブネットの名前に置き換えます。

5. API トークンを更新します。

```
$ source refresh-token
```

6. ダウンロード URL を取得します。

```
$ curl -H "Authorization: Bearer ${API_TOKEN}" \
https://api.openshift.com/api/assisted-install/v2/infra-
envs/${INFRA_ENV_ID}/downloads/image-url
```

7. Nutanix イメージ設定ファイルを作成します。

```
$ cat << EOF > create-image.json
{
  "spec": {
    "name": "ocp_ai_discovery_image.iso",
    "description": "ocp_ai_discovery_image.iso",
    "resources": {
      "architecture": "X86_64",
      "image_type": "ISO_IMAGE",
      "source_uri": "<image_url>",
      "source_options": {
        "allow_insecure_connection": true
      }
    }
  },
  "metadata": {
    "spec_version": 3,
    "kind": "image"
  }
}
EOF
```

<image_url> を、前の手順でダウンロードしたイメージの URL に置き換えます。

8. Nutanix イメージを作成します。

```
$ curl  -k -u <user>:'<password>' -X 'POST' \
'https://<domain-or-ip>:<port>/api/nutanix/v3/images \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d @./create-image.json | jq '.metadata.uuid'
```

<user> を Nutanix ユーザー名に置き換えます。**'<password>'** を Nutanix パスワードに置き換えます。**<domain-or-ip>** を Nutanix プラットフォームのドメイン名または IP アドレスに置き換えます。**<port>** を Nutanix サーバーのポートに置き換えます。ポートのデフォルトは **9440** です。

9. 返された UUID を設定ファイルの **NTX_IMAGE_UUID** 環境変数に割り当てます。

```
$ cat << EOF >> ~/nutanix-cluster-env.sh
export NTX_IMAGE_UUID=<uuid>
EOF
```

10. Nutanix クラスターの UUID を取得します。

```
$ curl -k -u <user>:'<password>' -X 'POST' \
  'https://<domain-or-ip>:<port>/api/nutanix/v3/clusters/list' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "kind": "cluster"
}'  | jq '.entities[] | select(.spec.name=="<nutanix_cluster_name>") | .metadata.uuid'
```

<user> を Nutanix ユーザー名に置き換えます。**'<password>'** を Nutanix パスワードに置き換えます。**<domain-or-ip>** を Nutanix プラットフォームのドメイン名または IP アドレスに置き換えます。**<port>** を Nutanix サーバーのポートに置き換えます。ポートのデフォルトは **9440**

です。**<nutanix_cluster_name>** を Nutanix クラスターの名前に置き換えます。

11. 返された Nutanix クラスター UUID を設定ファイルの **NTX_CLUSTER_UUID** 環境変数に割り当てます。

```
$ cat << EOF >> ~/nutanix-cluster-env.sh
export NTX_CLUSTER_UUID=<uuid>
EOF
```

<uuid> を Nutanix クラスターの返された UUID に置き換えます。

12. Nutanix クラスターのサブネット UUID を取得します。

```
$ curl -k -u <user>:'<password>' -X 'POST' \
  'https://<domain-or-ip>:<port>/api/nutanix/v3/subnets/list' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "kind": "subnet",
  "filter": "name==<subnet_name>"
}' | jq '.entities[].metadata.uuid'
```

<user> を Nutanix ユーザー名に置き換えます。**'<password>'** を Nutanix パスワードに置き換えます。**<domain-or-ip>** を Nutanix プラットフォームのドメイン名または IP アドレスに置き換えます。**<port>** を Nutanix サーバーのポートに置き換えます。ポートのデフォルトは **9440** です。**<subnet_name>** をクラスターのサブネットの名前に置き換えます。

13. 返された Nutanix サブネット UUID を設定ファイルの **NTX_CLUSTER_UUID** 環境変数に割り当てます。

```
$ cat << EOF >> ~/nutanix-cluster-env.sh
export NTX_SUBNET_UUID=<uuid>
EOF
```

<uuid> をクラスターサブネットの返された UUID に置き換えます。

14. Nutanix 環境変数が設定されていることを確認します。

```
$ source ~/nutanix-cluster-env.sh
```

15. Nutanix ホストごとに仮想マシン設定ファイルを作成します。3 つのコントロールプレーン (マスター) 仮想マシンと少なくとも 2 つのワーカー仮想マシンを作成します。以下に例を示します。

```
$ touch create-master-0.json
```

```
$ cat << EOF > create-master-0.json
{
  "spec": {
    "name": "<host_name>",
    "resources": {
      "power_state": "ON",
      "num_vcpus_per_socket": 1,
      "num_sockets": 16,
```

```
        "memory_size_mib": 32768,
        "disk_list": [
          {
            "disk_size_mib": 122880,
            "device_properties": {
              "device_type": "DISK"
            }
          },
          {
            "device_properties": {
              "device_type": "CDROM"
            },
            "data_source_reference": {
              "kind": "image",
              "uuid": "$NTX_IMAGE_UUID"
            }
          }
        ],
        "nic_list": [
          {
            "nic_type": "NORMAL_NIC",
            "is_connected": true,
            "ip_endpoint_list": [
              {
                "ip_type": "DHCP"
              }
            ],
            "subnet_reference": {
              "kind": "subnet",
              "name": "$NTX_SUBNET_NAME",
              "uuid": "$NTX_SUBNET_UUID"
            }
          }
        ],
        "guest_tools": {
          "nutanix_guest_tools": {
            "state": "ENABLED",
            "iso_mount_state": "MOUNTED"
          }
        }
      },
      "cluster_reference": {
        "kind": "cluster",
        "name": "$NTX_CLUSTER_NAME",
        "uuid": "$NTX_CLUSTER_UUID"
      }
    },
    "api_version": "3.1.0",
    "metadata": {
      "kind": "vm"
    }
  }
  EOF
```

<host_name> をホストの名前に置き換えます。

16. 各 Nutanix 仮想マシンを起動します。

```
$ curl -k -u <user>:'<password>' -X 'POST' \
  'https://<domain-or-ip>:<port>/api/nutanix/v3/vms' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d @./<vm_config_file_name> | jq '.metadata.uuid'
```

<user> を Nutanix ユーザー名に置き換えます。**'<password>'** を Nutanix パスワードに置き換えます。**<domain-or-ip>** を Nutanix プラットフォームのドメイン名または IP アドレスに置き換えます。**<port>** を Nutanix サーバーのポートに置き換えます。ポートのデフォルトは **9440** です。**<vm_config_file_name>** を仮想マシン設定ファイルの名前に置き換えます。

17. 返された仮想マシン UUID を設定ファイル内の一意の環境変数に割り当てます。

```
$ cat << EOF >> ~/nutanix-cluster-env.sh
export NTX_MASTER_0_UUID=<uuid>
EOF
```

<uuid> を仮想マシンの返された UUID に置き換えます。

注記

環境変数には、仮想マシンごとの一意の名前が必要です。

18. Assisted Installer が各仮想マシンを検出し、検証にパスするまで待ちます。

```
$ curl -s -X GET "https://api.openshift.com/api/assisted-install/v2/clusters/$CLUSTER_ID"
--header "Content-Type: application/json"
-H "Authorization: Bearer $API_TOKEN"
| jq '.enabled_host_count'
```

19. クラスター定義を変更して、Nutanix との統合を有効にします。

```
$ curl https://api.openshift.com/api/assisted-install/v2/clusters/${CLUSTER_ID} \
-X PATCH \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d '
{
    "platform_type":"nutanix"
}
' | jq
```

20. インストール手順を続行します。

13.3. NUTANIX のインストール後の設定

以下の手順に従って、OpenShift Container Platform と Nutanix クラウドプロバイダーの統合を完了し、検証します。

前提条件

- Assisted Installer によってクラスターが正常にインストールされている。
- クラスターが console.redhat.com に接続されている。
- Red Hat OpenShift Container Platform コマンドラインインターフェイスにアクセスできる。

13.3.1. Nutanix 設定の更新

Assisted Installer を使用して OpenShift Container Platform を Nutanix プラットフォームにインストールした後、以下の Nutanix 設定を手動で更新する必要があります。

- **<prismcentral_username>**: Nutanix Prism Central ユーザー名。
- **<prismcentral_password>**: Nutanix Prism Central のパスワード。
- **<prismcentral_address>**: Nutanix Prism Central のアドレス。
- **<prismcentral_port>**: Nutanix Prism Central のポート。
- **<prismelement_username>**: Nutanix Prism Element ユーザー名。
- **<prismelement_password>**: Nutanix Prism Element のパスワード。
- **<prismelement_address>**: Nutanix Prism Element のアドレス。
- **<prismelement_port>**: Nutanix Prism Element のポート。
- **<prismelement_clustername>**: Nutanix Prism Element のクラスター名。
- **<nutanix_storage_container>**: Nutanix Prism ストレージコンテナー。

手順

1. OpenShift Container Platform コマンドラインインターフェイスで、Nutanix クラスター設定を更新します。

```
$ oc patch infrastructure/cluster --type=merge --patch-file=/dev/stdin <<-EOF
{
  "spec": {
    "platformSpec": {
      "nutanix": {
        "prismCentral": {
          "address": "<prismcentral_address>",
          "port": <prismcentral_port>
        },
        "prismElements": [
          {
            "endpoint": {
              "address": "<prismelement_address>",
              "port": <prismelement_port>
            },
            "name": "<prismelement_clustername>"
          }
        ]
      },
      "type": "Nutanix"
```

```
    }
  }
}
EOF
```

出力例

```
infrastructure.config.openshift.io/cluster patched
```

詳細は、Nutanix でのマシンセットの作成 を参照してください。

2. Nutanix シークレットを作成します。

```
$ cat <<EOF | oc create -f -
apiVersion: v1
kind: Secret
metadata:
  name: nutanix-credentials
  namespace: openshift-machine-api
type: Opaque
stringData:
 credentials: |
[{"type":"basic_auth","data":{"prismCentral":
{"username":"${<prismcentral_username>}","password":"${<prismcentral_password>}"},"prism
Elements":null}}]
EOF
```

出力例

```
secret/nutanix-credentials created
```

3. OpenShift Container Platform バージョン 4.13 以降をインストールする場合は、Nutanix クラウドプロバイダー設定を更新します。

   a. Nutanix クラウドプロバイダー設定 YAML ファイルを取得します。

   ```
   $ oc get cm cloud-provider-config -o yaml -n openshift-config > cloud-provider-config-
   backup.yaml
   ```

   b. 設定ファイルのバックアップを作成します。

   ```
   $ cp cloud-provider-config_backup.yaml cloud-provider-config.yaml
   ```

   c. 設定 YAML ファイルを開きます。

   ```
   $ vi cloud-provider-config.yaml
   ```

   d. 以下のように設定 YAML ファイルを編集します。

   ```
   kind: ConfigMap
   apiVersion: v1
   metadata:
     name: cloud-provider-config
   ```

```
  namespace: openshift-config
data:
  config: |
    {
     "prismCentral": {
      "address": "<prismcentral_address>",
      "port":<prismcentral_port>,
      "credentialRef": {
         "kind": "Secret",
         "name": "nutanix-credentials",
         "namespace": "openshift-cloud-controller-manager"
      }
     },
     "topologyDiscovery": {
      "type": "Prism",
      "topologyCategories": null
     },
     "enableCustomLabeling": true
    }
```

e. 設定の更新を適用します。

```
$ oc apply -f cloud-provider-config.yaml
```

出力例

```
Warning: resource configmaps/cloud-provider-config is missing the
kubectl.kubernetes.io/last-applied-configuration annotation which is required by oc apply.
oc apply should only be used on resources created declaratively by either oc create --
save-config or oc apply. The missing annotation will be patched automatically.

configmap/cloud-provider-config configured
```

13.3.2. Nutanix CSI Operator グループの作成

Nutanix CSI Operator の Operator グループを作成します。

注記

Operator グループと関連概念の説明は、**関連情報** の **Operator Framework の一般的な用語** を参照してください。

手順

1. Nutanix CSI Operator Group YAML ファイルを開きます。

```
$ vi openshift-cluster-csi-drivers-operator-group.yaml
```

2. YAML ファイルを次のように編集します。

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
```

```
  generateName: openshift-cluster-csi-drivers
  namespace: openshift-cluster-csi-drivers
spec:
  targetNamespaces:
  - openshift-cluster-csi-drivers
  upgradeStrategy: Default
```

3. Operator グループを作成します。

```
$ oc create -f openshift-cluster-csi-drivers-operator-group.yaml
```

出力例

```
operatorgroup.operators.coreos.com/openshift-cluster-csi-driversjw9cd created
```

13.3.3. Nutanix CSI Operator のインストール

Nutanix Container Storage Interface (CSI) Operator for Kubernetes は、Nutanix CSI ドライバーをデプロイおよび管理します。

注記

Red Hat OpenShift Container Platform の Web コンソールを通じてこのステップを実行する手順については、**関連情報** の **Nutanix CSI Operator** ドキュメントの **Operator のインストール** セクションを参照してください。

手順

1. Nutanix CSI Operator YAML ファイルのパラメーター値を取得します。
   a. Nutanix CSI Operator が存在することを確認します。

```
$ oc get packagemanifests | grep nutanix
```

   出力例

```
nutanixcsioperator   Certified Operators   129m
```

   b. Operator のデフォルトチャネルを BASH 変数に割り当てます。

```
$ DEFAULT_CHANNEL=$(oc get packagemanifests nutanixcsioperator -o jsonpath=
{.status.defaultChannel})
```

   c. Operator の開始クラスターサービスバージョン (CSV) を BASH 変数に割り当てます。

```
$ STARTING_CSV=$(oc get packagemanifests nutanixcsioperator -o jsonpath=\
{.status.channels[*].currentCSV\})
```

   d. サブスクリプションのカタログソースを BASH 変数に割り当てます。

```
$ CATALOG_SOURCE=$(oc get packagemanifests nutanixcsioperator -o jsonpath=\
{.status.catalogSource\})
```

e. Nutanix CSI Operator ソース namespace を BASH 変数に割り当てます。

```
$ SOURCE_NAMESPACE=$(oc get packagemanifests nutanixcsioperator -o jsonpath=\
{.status.catalogSourceNamespace\})
```

2. BASH 変数を使用して Nutanix CSI Operator YAML ファイルを作成します。

```
$ cat << EOF > nutanixcsioperator.yaml
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: nutanixcsioperator
  namespace: openshift-cluster-csi-drivers
spec:
  channel: $DEFAULT_CHANNEL
  installPlanApproval: Automatic
  name: nutanixcsioperator
  source: $CATALOG_SOURCE
  sourceNamespace: $SOURCE_NAMESPACE
  startingCSV: $STARTING_CSV
EOF
```

3. CSI Nutanix Operator を作成します。

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

出力例

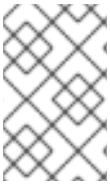
```
subscription.operators.coreos.com/nutanixcsioperator created
```

4. Operator サブスクリプションの状態が **AtLatestKnown** に変わるまで、以下のコマンドを実行します。これは Operator サブスクリプションが作成されたことを示しており、しばらく時間がかかる場合があります。

```
$ oc get subscription nutanixcsioperator -n openshift-cluster-csi-drivers -o 'jsonpath=
{..status.state}'
```

13.3.4. Nutanix CSI ストレージドライバーのデプロイ

Nutanix Container Storage Interface (CSI) Driver for Kubernetes は、ステートフルアプリケーションにスケーラブルで永続的なストレージを提供します。



注記

OpenShift Container Platform の Web コンソールを通じてこのステップを実行する手順については、**関連情報** の **Nutanix CSI Operator** ドキュメントの **Operator を使用した CSI ドライバーのインストール** セクションを参照してください。

手順

1. **NutanixCsiStorage** リソースを作成して、ドライバーをデプロイします。

```
$ cat <<EOF | oc create -f -
```

```
apiVersion: crd.nutanix.com/v1alpha1
kind: NutanixCsiStorage
metadata:
  name: nutanixcsistorage
  namespace: openshift-cluster-csi-drivers
spec: {}
EOF
```

出力例

```
snutanixcsistorage.crd.nutanix.com/nutanixcsistorage created
```

2. CSI ストレージドライバーの Nutanix シークレット YAML ファイルを作成します。

```
$ cat <<EOF | oc create -f -
apiVersion: v1
kind: Secret
metadata:
  name: ntnx-secret
  namespace: openshift-cluster-csi-drivers
stringData:
  # prism-element-ip:prism-port:admin:password
  key:
<prismelement_address:prismelement_port:prismcentral_username:prismcentral_password>
(1)
EOF
```

注記

(1) 同じ形式を維持したまま、これらのパラメーターを実際の値に置き換えます。

出力例

```
secret/nutanix-secret created
```

13.3.5. インストール後の設定の検証

以下のコマンドを実行して設定を検証します。

手順

1. ストレージクラスを作成できることを確認します。

```
$ cat <<EOF | oc create -f -
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: nutanix-volume
  annotations:
    storageclass.kubernetes.io/is-default-class: 'true'
provisioner: csi.nutanix.com
```

```
parameters:
  csi.storage.k8s.io/fstype: ext4
  csi.storage.k8s.io/provisioner-secret-namespace: openshift-cluster-csi-drivers
  csi.storage.k8s.io/provisioner-secret-name: ntnx-secret
  storageContainer: <nutanix_storage_container> 1
  csi.storage.k8s.io/controller-expand-secret-name: ntnx-secret
  csi.storage.k8s.io/node-publish-secret-namespace: openshift-cluster-csi-drivers
  storageType: NutanixVolumes
  csi.storage.k8s.io/node-publish-secret-name: ntnx-secret
  csi.storage.k8s.io/controller-expand-secret-namespace: openshift-cluster-csi-drivers
reclaimPolicy: Delete
allowVolumeExpansion: true
volumeBindingMode: Immediate
EOF
```

注記

1 Nutanix 設定から <nutanix_storage_container> を取得します (例: SelfServiceContainer)。

出力例

```
storageclass.storage.k8s.io/nutanix-volume created
```

2. Nutanix 永続ボリューム要求 (PVC) を作成できることを確認します。

   a. 永続ボリューム要求 (PVC) を作成します。

```
$ cat <<EOF | oc create -f -
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: nutanix-volume-pvc
  namespace: openshift-cluster-csi-drivers
  annotations:
    volume.beta.kubernetes.io/storage-provisioner: csi.nutanix.com
    volume.kubernetes.io/storage-provisioner: csi.nutanix.com
  finalizers:
    - kubernetes.io/pvc-protection
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: nutanix-volume
  volumeMode: Filesystem
EOF
```

出力例

```
persistentvolumeclaim/nutanix-volume-pvc created
```

b. Persistent Volume Claim (PVC) ステータスが Bound であることを確認します。

```
$ oc get pvc -n openshift-cluster-csi-drivers
```

出力例

```
NAME                 STATUS   VOLUME   CAPACITY   ACCESS MODES
STORAGECLASS     AGE
nutanix-volume-pvc   Bound                       nutanix-volume   52s
```

関連情報

- Nutanix でのマシンセットの作成
- Nutanix CSI Operator
- Storage Management
- Operator Framework の一般的な用語

第14章 オプション: VSPHERE へのインストール

Assisted Installer は、OpenShift Container Platform クラスターを vSphere プラットフォームと統合します。これにより、Machine API が vSphere に公開され、自動スケーリングが有効になります。

14.1. VSPHERE へのホストの追加

オンラインの vSphere クライアントまたは **govc** vSphere CLI ツールを使用して、Assisted Installer クラスターにホストを追加できます。次の手順は、**govc** CLI ツールを使用してホストを追加する方法を示しています。オンラインの vSphere Client を使用するには、vSphere のドキュメントを参照してください。

vSphere **govc** CLI を使用して vSphere にホストを追加するには、Assisted Installer から検出イメージ ISO を生成します。最小限の検出イメージ ISO がデフォルト設定です。このイメージには、ネットワークを使用してホストを起動するために必要なものだけが含まれています。コンテンツの大部分は、起動時にダウンロードされます。ISO イメージのサイズは約 100MB です。

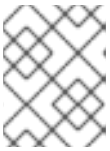
これが完了したら、vSphere プラットフォームのイメージを作成し、vSphere 仮想マシンを作成する必要があります。

前提条件

- vSphere 7.0.2 以降を使用している。
- vSphere **govc** CLI ツールがインストールされ、設定されている。
- vSphere で、**clusterSet disk.enableUUID** を **true** に設定している。
- Assisted Installer Web コンソールでクラスターを作成している。または
- API を使用して、Assisted Installer のクラスタープロファイルとインフラストラクチャー環境を作成した。
- シェルのインフラストラクチャー環境 ID を **$INFRA_ENV_ID** としてエクスポートした。

手順

1. Ignition ファイルを使用して起動する場合は、検出イメージを設定します。
2. **Cluster details** の **Integrate with external partner platforms** ドロップダウンリストから vSphere を選択します。**Include custom manifest** チェックボックスはオプションです。
3. **Host discovery** で、**Add hosts** ボタンをクリックし、プロビジョニングタイプを選択します。
4. **core** ユーザーとして vSphere 仮想マシンに接続できるように、SSH 公開鍵を追加します。クラスターホストにログインすると、インストール中にデバッグ情報を入手できます。
    a. ローカルマシンに既存の SSH キーペアがない場合は、クラスターノード SSH アクセス用のキーペアの生成 の手順に従います。
    b. **SSH public key** フィールドで **Browse** をクリックして、SSH 公開鍵を含む **id_rsa.pub** ファイルをアップロードします。あるいは、ファイルマネージャーからファイルをフィールドにドラッグアンドドロップします。ファイルマネージャーでファイルを表示するには、メニューで **Show hidden files** を選択します。
5. 目的の検出イメージ ISO を選択します。



注記

Minimal image file: Provision with virtual mediaでは、起動に必要なデータをフェッチする小さなイメージがダウンロードされます。

6. **Networking** で、**Cluster-managed networking** または **User-managed networking** を選択します。

   a. オプション: クラスターホストがプロキシーの使用を必要とするファイアウォールの内側にある場合は、**Configure cluster-wide proxy settings** を選択します。プロキシーサーバーの HTTP および HTTPS URL のユーザー名、パスワード、IP アドレス、およびポートを入力します。

   b. オプション: クラスターホストが再暗号化中間者 (MITM) プロキシーを使用するネットワーク内にある場合、またはクラスターがコンテナーイメージレジストリーなどの他の目的で証明書を信頼する必要がある場合は、**Configure cluster-wide trusted certificates** を選択し、追加の証明書を追加します。

   c. オプション: Ignition ファイルを使用して起動する場合は、検出イメージを設定します。詳しくは、検出イメージの設定 を参照してください。

7. **Generate Discovery ISO** をクリックします。

8. **Discovery ISO URL** をコピーします。

9. 検出 ISO をダウンロードします。

```
$ wget - O vsphere-discovery-image.iso <discovery_url>
```

   <discovery_url> は、前の手順の **Discovery ISO URL** に置き換えます。

10. コマンドラインで、電源を落とし、既存の仮想マシンをすべて破棄します。

```
$ for VM in $(/usr/local/bin/govc ls /<datacenter>/vm/<folder_name>)
do
  /usr/local/bin/govc vm.power -off $VM
  /usr/local/bin/govc vm.destroy $VM
done
```

    <datacenter> をデータセンターの名前に置き換えます。**<folder_name>** を仮想マシンのインベントリーフォルダーの名前に置き換えます。

11. 既存の ISO イメージがある場合は、データストアから削除します。

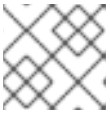
```
$ govc datastore.rm -ds <iso_datastore> <image>
```

    <iso_datastore> をデータストアの名前に置き換えます。**image** を ISO イメージの名前に置き換えます。

12. Assisted Installer の検出 ISO をアップロードします。

```
$ govc datastore.upload -ds <iso_datastore>  vsphere-discovery-image.iso
```

    <iso_datastore> をデータストアの名前に置き換えます。



注記

クラスター内のすべてのノードは、検出イメージから起動する必要があります。

13. 3 つのコントロールプレーン (マスター) ノードを起動します。

```
$ govc vm.create -net.adapter <network_adapter_type> \
        -disk.controller <disk_controller_type> \
        -pool=<resource_pool> \
        -c=16 \
        -m=32768 \
        -disk=120GB \
        -disk-datastore=<datastore_file> \
        -net.address="<nic_mac_address>" \
        -iso-datastore=<iso_datastore> \
        -iso="vsphere-discovery-image.iso" \
        -folder="<inventory_folder>" \
        <hostname>.<cluster_name>.example.com
```

詳細は、vm.create を参照してください。

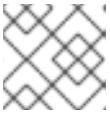
注記

前述の例は、コントロールプレーンノードに必要な最小限のリソースを示しています。

14. 少なくとも 2 つのワーカーノードを起動します。

```
$ govc vm.create -net.adapter <network_adapter_type> \
        -disk.controller <disk_controller_type> \
        -pool=<resource_pool> \
        -c=4 \
        -m=8192 \
        -disk=120GB \
        -disk-datastore=<datastore_file> \
        -net.address="<nic_mac_address>" \
        -iso-datastore=<iso_datastore> \
        -iso="vsphere-discovery-image.iso" \
        -folder="<inventory_folder>" \
        <hostname>.<cluster_name>.example.com
```

詳細は、vm.create を参照してください。



注記

上記の例は、ワーカーノードに必要な最小限のリソースを示しています。

15. 仮想マシンが実行されていることを確認します。

```
$  govc ls /<datacenter>/vm/<folder_name>
```

<datacenter> をデータセンターの名前に置き換えます。**<folder_name>** を仮想マシンのインベントリーフォルダーの名前に置き換えます。

16. 2 分後、仮想マシンをシャットダウンします。

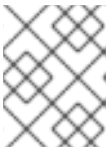
```
$ for VM in $(govc ls /<datacenter>/vm/<folder_name>)
do
    govc vm.power -s=true $VM
done
```

<datacenter> をデータセンターの名前に置き換えます。**<folder_name>** を仮想マシンのインベントリーフォルダーの名前に置き換えます。

17. **disk.enableUUID** 設定を **TRUE** に設定します。

```
$ for VM in $(govc ls /<datacenter>/vm/<folder_name>)
do
    govc vm.change -vm $VM -e disk.enableUUID=TRUE
done
```

<datacenter> をデータセンターの名前に置き換えます。**<folder_name>** を仮想マシンのインベントリーフォルダーの名前に置き換えます。



注記

vSphere で自動スケーリングを有効にするには、すべてのノードで **disk.enableUUID** を **TRUE** に設定する必要があります。

18. 仮想マシンを再起動します。

```
$  for VM in $(govc ls /<datacenter>/vm/<folder_name>)
do
    govc vm.power -on=true $VM
done
```

<datacenter> をデータセンターの名前に置き換えます。**<folder_name>** を仮想マシンのインベントリーフォルダーの名前に置き換えます。

19. Assisted Installer のユーザーインターフェイスに戻り、Assisted Installer がホストを検出し、それぞれが **Ready** ステータスになるまで待ちます。

20. 必要に応じてロールを選択します。

21. **Networking** で、**Allocate IPs via DHCP server**のチェックを外します。

22. API VIP アドレスを設定します。

23. Ingress VIP アドレスを設定します。

24. インストール手順を続行します。

14.2. CLI を使用した VSPHERE のインストール後の設定

プラットフォーム統合機能を有効にして、vSphere で Assisted Installer を使用して OpenShift Container Platform クラスターをインストールした後、以下の vSphere 設定を手動で更新する必要があります。

- vCenter ユーザー名
- vCenter パスワード
- vCenter アドレス
- vCenter クラスター
- datacenter
- datastore
- folder

前提条件

- Assisted Installer によってクラスターが正常にインストールされている。
- クラスターが console.redhat.com に接続されている。

手順

1. vCenter 用の base64 でエンコードされたユーザー名とパスワードを生成します。

```
$ echo -n "<vcenter_username>" | base64 -w0
```

<vcenter_username> を vCenter ユーザー名に置き換えます。

```
$ echo -n "<vcenter_password>" | base64 -w0
```

<vcenter_password> を vCenter パスワードに置き換えます。

2. vSphere 認証情報をバックアップします。

```
$ oc get secret vsphere-creds -o yaml -n kube-system > creds_backup.yaml
```

3. vSphere 認証情報を編集します。

```
$ cp creds_backup.yaml vsphere-creds.yaml
```

```
$ vi vsphere-creds.yaml
```

```
apiVersion: v1
data:
  <vcenter_address>.username: <vcenter_username_encoded>
  <vcenter_address>.password: <vcenter_password_encoded>
kind: Secret
metadata:
  annotations:
    cloudcredential.openshift.io/mode: passthrough
  creationTimestamp: "2022-01-25T17:39:50Z"
  name: vsphere-creds
  namespace: kube-system
```

```
  resourceVersion: "2437"
  uid: 06971978-e3a5-4741-87f9-2ca3602f2658
type: Opaque
```

<vcenter_address> を vCenter アドレスに置き換えます。**<vcenter_username_encoded>** を base64 でエンコードされたバージョンの vSphere ユーザー名に置き換えます。**<vcenter_password_encoded>** を base64 でエンコードされたバージョンの vSphere パスワードに置き換えます。

4. vSphere 認証情報を置き換えます。

```
$ oc replace -f vsphere-creds.yaml
```

5. kube-controller-manager Pod を再デプロイします。

```
$ oc patch kubecontrollermanager cluster -p='{"spec": {"forceRedeploymentReason":
"recovery-'"$( date --rfc-3339=ns )"'"}}' --type=merge
```

6. vSphere クラウドプロバイダー設定をバックアップします。

```
$ oc get cm cloud-provider-config -o yaml -n openshift-config > cloud-provider-
config_backup.yaml
```

7. クラウドプロバイダーの設定を編集します。

```
$ cloud-provider-config_backup.yaml cloud-provider-config.yaml
```

```
$ vi cloud-provider-config.yaml
```

```
apiVersion: v1
data:
  config: |
    [Global]
    secret-name = "vsphere-creds"
    secret-namespace = "kube-system"
    insecure-flag = "1"

    [Workspace]
    server = "<vcenter_address>"
    datacenter = "<datacenter>"
    default-datastore = "<datastore>"
    folder = "/<datacenter>/vm/<folder>"

    [VirtualCenter "<vcenter_address>"]
    datacenters = "<datacenter>"
kind: ConfigMap
metadata:
  creationTimestamp: "2022-01-25T17:40:49Z"
  name: cloud-provider-config
  namespace: openshift-config
  resourceVersion: "2070"
  uid: 80bb8618-bf25-442b-b023-b31311918507
```

<vcenter_address> を vCenter アドレスに置き換えます。**<datacenter>** をデータセンターの名前に置き換えます。**<datastore>** をデータストアの名前に置き換えます。**<folder>** をクラスターの仮想マシンを含むフォルダーに置き換えます。

8. クラウドプロバイダーの設定を適用します。

```
$ oc apply -f cloud-provider-config.yaml
```

9. **uninitialized** テイントでノードをテイントします。

重要

OpenShift Container Platform 4.13 以降をインストールする場合は、ステップ 9 から 12 に従います。

a. テイントするノードを特定します。

```
$ oc get nodes
```

b. ノードごとに以下のコマンドを実行します。

```
$ oc adm taint node <node_name>
node.cloudprovider.kubernetes.io/uninitialized=true:NoSchedule
```

<node_name> はノード名に置き換えてください。

例

```
$ oc get nodes
NAME              STATUS   ROLES                AGE   VERSION
master-0   Ready    control-plane,master   45h   v1.26.3+379cd9f
master-1   Ready    control-plane,master   45h   v1.26.3+379cd9f
worker-0   Ready    worker                 45h   v1.26.3+379cd9f
worker-1   Ready    worker                 45h   v1.26.3+379cd9f
master-2   Ready    control-plane,master   45h   v1.26.3+379cd9f

$ oc adm taint node master-0
node.cloudprovider.kubernetes.io/uninitialized=true:NoSchedule
$ oc adm taint node master-1
node.cloudprovider.kubernetes.io/uninitialized=true:NoSchedule
$ oc adm taint node master-2
node.cloudprovider.kubernetes.io/uninitialized=true:NoSchedule
$ oc adm taint node worker-0
node.cloudprovider.kubernetes.io/uninitialized=true:NoSchedule
$ oc adm taint node worker-1
node.cloudprovider.kubernetes.io/uninitialized=true:NoSchedule
```

10. インフラストラクチャー設定をバックアップします。

```
$ oc get infrastructures.config.openshift.io -o yaml >
infrastructures.config.openshift.io.yaml.backup
```

11. インフラストラクチャー設定を編集します。

-

```
$ cp infrastructures.config.openshift.io.yaml.backup infrastructures.config.openshift.io.yaml
```

```
$ vi infrastructures.config.openshift.io.yaml
```

```
apiVersion: v1
items:
- apiVersion: config.openshift.io/v1
  kind: Infrastructure
  metadata:
    creationTimestamp: "2022-05-07T10:19:55Z"
    generation: 1
    name: cluster
    resourceVersion: "536"
    uid: e8a5742c-6d15-44e6-8a9e-064b26ab347d
  spec:
    cloudConfig:
      key: config
      name: cloud-provider-config
    platformSpec:
      type: VSphere
      vsphere:
        failureDomains:
        - name: assisted-generated-failure-domain
          region: assisted-generated-region
          server: <vcenter_address>
          topology:
            computeCluster: /<data_center>/host/<vcenter_cluster>
            datacenter: <data_center>
            datastore: /<data_center>/datastore/<datastore>
            folder: "/<data_center>/path/to/folder"
            networks:
            - "VM Network"
            resourcePool: /<data_center>/host/<vcenter_cluster>/Resources
          zone: assisted-generated-zone
        nodeNetworking:
          external: {}
          internal: {}
        vcenters:
        - datacenters:
          - <data_center>
          server: <vcenter_address>

kind: List
metadata:
  resourceVersion: ""
```

<vcenter_address> を vCenter アドレスに置き換えます。**<datacenter>** を vCenter データセンターの名前に置き換えます。**<datastore>** を vCenter データストアの名前に置き換えます。**<folder>** をクラスターの仮想マシンを含むフォルダーに置き換えます。**<vcenter_cluster>** を、OpenShift Container Platform がインストールされている vSphere vCenter クラスターに置き換えます。

12. インフラストラクチャー設定を適用します。

```
$ oc apply -f infrastructures.config.openshift.io.yaml --overwrite=true
```

-

14.3. WEB コンソールを使用した VSPHERE のインストール後の設定

プラットフォーム統合機能を有効にして、vSphere で Assisted Installer を使用して OpenShift Container Platform クラスターをインストールした後、以下の vSphere 設定を手動で更新する必要があります。

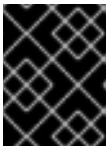
- vCenter アドレス
- vCenter クラスター
- vCenter ユーザー名
- vCenter パスワード
- データセンター
- デフォルトのデータストア
- 仮想マシンフォルダー

前提条件

- Assisted Installer によってクラスターが正常にインストールされている。
- クラスターが console.redhat.com に接続されている。

手順

1. Administrator パースペクティブで、**Home → Overview** に移動します。
2. **Status** で **vSphere connection** をクリックし、**vSphere connection configuration** ウィザードを開きます。
3. **vCenter** フィールドに、vSphere vCenter サーバーのネットワークアドレスを入力します。ドメイン名または IP アドレスのいずれかを入力できます。これは vSphere Web クライアント URL に表示されます (例: **https://[your_vCenter_address]/ui**)。
4. **vCenter クラスター** フィールドには、OpenShift Container Platform がインストールされている vSphere vCenter クラスターの名前を入力します。



重要

この手順は、OpenShift Container Platform 4.13 以降をインストールしている場合は必須となります。

5. **Username** フィールドに、vSphere vCenter のユーザー名を入力します。
6. **Password** フィールドに、vSphere vCenter のパスワードを入力します。

警告

システムは、クラスターの **kube-system** namespace の **vsphere-creds** シークレットにユーザー名とパスワードを保存します。vCenter のユーザー名またはパスワードが間違っていると、クラスターノードをスケジュールできなくなります。

7. **Datacenter** フィールドに、クラスターのホストに使用する仮想マシンが含まれる vSphere データセンターの名前を入力します (例: **SDDC-Datacenter**)。

8. **Default data store** フィールドに、永続データボリュームを保存する vSphere データストアを入力します (例: **/SDDC-Datacenter/datastore/datastorename**)。

警告

設定の保存後に vSphere データセンターまたはデフォルトのデータストアを更新すると、アクティブな vSphere **PersistentVolumes** がデタッチされます。

9. **Virtual Machine Folder** フィールドに、クラスターの仮想マシンが含まれるデータセンターフォルダーを入力します (例: **/SDDC-Datacenter/vm/ci-ln-hjg4vg2-c61657-t2gzr**)。正常に OpenShift Container Platform をインストールするには、クラスターを構成するすべての仮想マシンを単一のデータセンターフォルダーに配置する必要があります。

10. **Save Configuration** をクリックします。これにより、**openshift-config** namespace の **cloud-provider-config** ファイルが更新され、設定プロセスが開始されます。

11. **vSphere connection configuration** ウィザードを再度開き、**Monitored operators** パネルを展開します。Operator のステータスが **Progressing** または **Healthy** であることを確認します。

検証

接続設定プロセスは、Operator ステータスとコントロールプレーンノードを更新します。完了するまでに約 1 時間かかります。設定プロセスの中でノードが再起動します。これまでは、バインドされた **PersistentVolumeClaims** オブジェクトの接続が切断される可能性がありました。

設定プロセスを監視するには、以下の手順に従ってください。

1. 設定プロセスが正常に完了したことを確認します。

   a. OpenShift Container Platform の Administrator パースペクティブで、**Home → Overview** に移動します。

   b. **Status** で **Operators** をクリックします。すべての Operator ステータスが **Progressing** から **All succeeded** に変わるまで待機します。**Failed** ステータスは、設定が失敗したことを示します。

c. **Status** で **Control Plane** をクリックします。すべての Control Pane コンポーネントの応答レートが 100% に戻るまで待機します。**Failed** コントロールプレーンコンポーネントは、設定が失敗したことを示します。

失敗は、少なくとも 1 つの接続設定が間違っていることを示します。**vSphere connection configuration** ウィザードで設定を変更し、その設定を再度保存します。

2. 以下の手順を実行して、**PersistentVolumeClaims** オブジェクトをバインドできることを確認します。

a. 以下の YAML を使用して **StorageClass** オブジェクトを作成します。

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
 name: vsphere-sc
provisioner: kubernetes.io/vsphere-volume
parameters:
 datastore: YOURVCENTERDATASTORE
 diskformat: thin
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

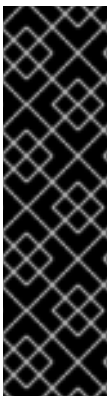
b. 以下の YAML を使用して **PersistentVolumeClaims** オブジェクトを作成します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
 name: test-pvc
 namespace: openshift-config
 annotations:
   volume.beta.kubernetes.io/storage-provisioner: kubernetes.io/vsphere-volume
 finalizers:
   - kubernetes.io/pvc-protection
spec:
 accessModes:
   - ReadWriteOnce
 resources:
   requests:
    storage: 10Gi
 storageClassName: vsphere-sc
 volumeMode: Filesystem
```

手順については、OpenShift Container Platform ドキュメントの 動的プロビジョニング を参照してください。PersistentVolumeClaims オブジェクトのトラブルシューティングを行うには、OpenShift Container Platform UI の **Administrator** パースペクティブで、**Storage → Persistent VolumeClaims** に移動します。

第15章 オプション: ORACLE CLOUD INFRASTRUCTURE (OCI) へのインストール

OpenShift Container Platform 4.14 以降のバージョンでは、提供するインフラストラクチャーを使用して、Assisted Installer で Oracle Cloud Infrastructure にクラスターをインストールできます。Oracle Cloud Infrastructure は、規制遵守、パフォーマンス、費用対効果の要件を満たすサービスを提供します。OCI Resource Manager 設定にアクセスして、OCI リソースをプロビジョニングおよび設定できます。



重要

OpenShift Container Platform 4.14 および 4.15 の OCI 統合は、テクノロジープレビュー機能としてのみ使用できます。テクノロジープレビュー機能は、Red Hat 製品サポートのサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではない場合があります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビューの機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

このセクションは、Oracle Cloud Infrastructure との統合をサポートするために Assisted Installer UI で必要な手順の概要です。Oracle Cloud Infrastructureで実行する手順や、2 つのプラットフォーム間の統合も説明されていません。完全かつ包括的な手順は、Assisted Installer を使用した OCI へのクラスターのインストール を参照してください。

15.1. OCI 互換の検出 ISO イメージの生成

必要な手順を実行して、Assisted Installer で検出 ISO イメージを生成します。Oracle Cloud Infrastructure に OpenShift Container Platform をインストールする前に、イメージを Oracle Cloud Infrastructure にアップロードする必要があります。

前提条件

- Oracle Cloud Infrastructure 上に子コンパートメントとオブジェクトストレージバケットを作成した。OpenShift Container Platform ドキュメントの OCI リソースとサービスの作成 を参照してください。
- クラスターのインストールに必要な要件を満たしている。詳細は、前提条件 を参照してください。

手順

1. Red Hat Hybrid Cloud コンソール にログインします。
2. **Create cluster** をクリックします。
3. **Cluster type** ページで **Datacenter** タブをクリックします。
4. **Assisted Installer** セクションで、**Create cluster** をクリックします。
5. **Cluster Details** ページで、次のフィールドに入力します。
    a. **Cluster name** フィールドに、クラスターの名前 (**ocidemo** など) を指定します。

b. **ベースドメイン** フィールドに、クラスターのベースドメイン (**splat-oci.devcluster.openshift.com** など) を指定します。コンパートメントとゾーンを作成した後、OCI からベースドメインを取得します。

c. **OpenShift version** フィールドに、OpenShift 4.15 以降のバージョンを指定します。

d. **[CPU アーキテクチャー]** フィールドで、**x86_64** または **Arm64** を指定します。

e. **Integrate with external partner platforms** リストから、**Oracle Cloud Infrastructure** を選択します。**Include custom manifests** チェックボックスが自動的に選択されます。

6. **Next** をクリックします。

7. **Operators** ページで、**Next** をクリックします。

8. **Host Discovery** ページで、以下の操作を実行します。

a. **Add host** をクリックしてダイアログボックスを表示します。

b. **SSH public key** フィールドには、ローカルシステムから SSH 公開鍵をアップロードします。**ssh-keygen** を使用して、SSH 鍵ペアを生成できます。

c. **Generate Discovery ISO** をクリックして、検出イメージの ISO ファイルを生成します。

d. ローカルシステムにファイルをダウンロードします。次に、ファイルを Oracle Cloud Infrastructure のバケットにオブジェクトとしてアップロードします。

15.2. ノードのロールとカスタムマニフェストの割り当て

Oracle Cloud Infrastructure (OCI) リソースをプロビジョニングし、OpenShift Container Platform カスタムマニフェスト設定ファイルを OCI にアップロードした後、インスタンス OCI を作成する前に、Assisted Installer で残りのクラスターのインストール手順を完了する必要があります。

前提条件

- OCI 上にリソーススタックを作成しており、スタックにはカスタムマニフェスト設定ファイルと OCI Resource Manager 設定リソースが含まれている。詳細は、OpenShift Container Platform ドキュメントの マニフェストファイルおよびデプロイメントリソースのダウンロード を参照してください。

手順

1. Red Hat Hybrid Cloud Console から、**Host discovery** ページに移動します。

2. **Role** 列で、ターゲットのホスト名ごとにノードロール (**Control plane node** or **Worker**) を割り当てます。**Next** をクリックします。

3. **Storage** ページおよび **Networking** ページのデフォルト設定を受け入れます。

4. **Next** をクリックして **Custom manifests** ページに移動します。

5. **Folder** フィールドで **manifests** を選択します。

6. **File name** フィールドに、**oci-ccm.yml** などの値を入力します。

7. **Content** セクションで、**Browse** をクリックします。**custom_ manifest/manifests/oci-ccm.yml** にある CCM マニフェストを選択します。

8. **Add another manifest** をクリックします。Oracle が提供する次のマニフェストに対して同じ手順を繰り返します。

   - CSI ドライバーマニフェスト: **custom_manifest/manifests/oci-csi.yml**。
   - CCM マシン設定: **custom_manifest/openshift/machineconfig-ccm.yml**。
   - CSI ドライバーのマシン設定: **custom_manifest/openshift/machineconfig-csi.yml**。

9. **レビューと作成** ステップを完了して、OCI 上に OpenShift Container Platform クラスターを作成します。

10. **Install cluster** をクリックして、クラスターインストールを終了します。

第16章 トラブルシューティング

Assisted Installer がインストールを開始できない場合や、クラスターが正しくインストールされない場合があります。これらのイベントでは、可能性のある障害モードと、障害のトラブルシューティング方法を理解しておくと役に立ちます。

16.1. 検出 ISO の問題のトラブルシューティング

Assisted Installer は、ISO イメージを使用して、ホストをクラスターに登録し、OpenShift のインストールを試行する前にハードウェアとネットワークの検証を実行するエージェントを実行します。以下の手順に従って、ホスト検出に関連する問題をトラブルシューティングできます。

検出 ISO イメージを使用してホストを起動すると、Assisted Installer がホストを検出し、Assisted Service UI に表示します。詳しくは、検出イメージの設定 を参照してください。

16.1.1. 検出エージェントが実行されていることを確認する

前提条件

- API を使用してインフラストラクチャー環境を作成したか、UI を使用してクラスターを作成している。
- インフラストラクチャー環境の検出 ISO を使用してホストを起動しましたが、ホストは登録に失敗した。
- ホストに ssh アクセスできる。
- パスワードなしでマシンに SSH 接続できるように、Discovery ISO を生成する前にホストの追加 ダイアログで SSH 公開キーを指定した。

手順

1. ホストマシンの電源が入っていることを確認します。
2. **DHCP ネットワーク** を選択した場合は、DHCP サーバーが有効になっていることを確認します。
3. **静的 IP、ブリッジ、および結合** ネットワーキングを選択した場合は、設定が正しいことを確認してください。
4. SSH、BMC などのコンソール、または仮想マシンコンソールを使用してホストマシンにアクセスできることを確認します。

   ```
   $ ssh core@<host_ip_address>
   ```

   デフォルトディレクトリーに格納されていない場合は、**-i** パラメーターを使用して秘密鍵ファイルを指定できます。

   ```
   $ ssh -i <ssh_private_key_file> core@<host_ip_address>
   ```

   ホストへの ssh に失敗した場合、ホストは起動中に失敗したか、ネットワークの設定に失敗しました。

   ログインすると、次のメッセージが表示されます。

ログイン例

```
**  **  **  **  **  **  **  **  **  **  **  **  **  **  **  **  ** **  **  **  **  **
This is a host being installed by the OpenShift Assisted Installer.
It will be installed from scratch during the installation.
The primary service is agent.service. To watch its status, run:
sudo journalctl -u agent.service
To view the agent log, run:
sudo journalctl TAG=agent
**  **  **  **  **  **  **  **  **  **  **  **  **  **  **  **  ** **  **  **  **  **
```

このメッセージが表示されない場合は、ホストが assisted-installer ISO で起動されなかったことを意味します。起動順序を正しく設定したことを確認してください (ホストは live-ISO から 1 回起動する必要があります)。

5. エージェントサービスログを確認します。

```
$ sudo journalctl -u agent.service
```

次の例のエラーは、ネットワークに問題があることを示しています。

エージェントサービスログの例 エージェントサービスログのスクリーンショット

```
Oct 15 11:26:35 localhost systemd[1]: agent. service: Service RestartSec=3s expired, scheduling restart.
Oct 15 11:26:35 localhost systemd[1]: agent. service: Scheduled restart job, restart counter is at 9.
Oct 15 11:26:35 localhost systemd[1]: Stopped agent. service.
Oct 15 11:26:35 localhost systemd[1]: Starting agent. service...
Oct 15 11:26:35 localhost podman[1834]: Trying to pull quay.io/ocpmetal/assisted-installer-agent: lates
Oct 15 11:26:35 localhost podman[1834]:
Get "https://quay.io/v2/": dial top: lookup quay.io on [::1]:53: read udp [::1]:582
Oct 15 11:26:35 localhost podman[1834]: Error: unable to pull quay. io/ocpmetal/assisted-installer-agent:latest: unable to pull
Oct 15 11:26:35 localhost systemd[1]: agent. service: Control process exited, code=exited status=125
Oct 15 11:26:35 localhost systemd[1]: agent. service: Failed with result 'exit-code'.
Oct 15 11:26:35 localhost systemd[1]: Failed to start agent. service.
```

エージェントイメージのプルでエラーが発生した場合は、プロキシー設定を確認してください。ホストがネットワークに接続されていることを確認します。**nmcli** を使用して、ネットワーク設定に関する追加情報を取得できます。

16.1.2. エージェントが assisted-service にアクセスできることを確認する

前提条件

- API を使用してインフラストラクチャー環境を作成したか、UI を使用してクラスターを作成している。
- インフラストラクチャー環境の検出 ISO を使用してホストを起動しましたが、ホストは登録に失敗した。
- 検出エージェントが実行されていることを確認した。

手順

- エージェントログをチェックして、エージェントが Assisted Service にアクセスできることを確認します。

```
$ sudo journalctl TAG=agent
```

次の例のエラーは、エージェントが Assisted Service へのアクセスに失敗したことを示しています。

エージェントログの例

```
Jul 21 19:39:44 test-infra-cluster-7c35a054-master-1 next_step_runne[1909]: time="21-07-2022 19:39:44" leve
l=warning msg="Could not query next steps: Get \"https://api.stage.openshift.com/api/assisted-install/v2/in
fra-envs/ba747803-f85d-40f4-8af4-01d7f0d8914f/hosts/8daa0b33-d10a-46aa-ab59-ea9be2e0c4d9/instructions?times
tamp=1658432367\": dial tcp: lookup api.stage.openshift.com on 192.168.131.1:53: read udp 192.168.131.11:58
016->192.168.131.1:53: i/o timeout" file="step_processor.go:238" request_id=00a041ba-0314-4d00-83f1-486b36b
d02bb
Jul 21 19:40:44 test-infra-cluster-7c35a054-master-1 next_step_runne[1909]: time="21-07-2022 19:40:44" leve
l=info msg="Query for next steps" file="step_processor.go:223" request_id=6cb39274-7f5b-4099-9894-912702c77
b09
Jul 21 19:40:54 test-infra-cluster-7c35a054-master-1 next_step_runne[1909]: time="21-07-2022 19:40:54" leve
l=warning msg="Could not query next steps: Get \"https://api.stage.openshift.com/api/assisted-install/v2/in
fra-envs/ba747803-f85d-40f4-8af4-01d7f0d8914f/hosts/8daa0b33-d10a-46aa-ab59-ea9be2e0c4d9/instructions?times
tamp=1658432444\": net/http: TLS handshake timeout" file="step_processor.go:238" request_id=6cb39274-7f5b-4
099-9894-912702c77b09
Jul 21 19:41:54 test-infra-cluster-7c35a054-master-1 next_step_runne[1909]: time="21-07-2022 19:41:54" leve
l=info msg="Query for next steps" file="step_processor.go:223" request_id=8ca23a1c-4d5c-494b-8d59-9846fcdff
b9e
```

クラスター用に設定したプロキシー設定を確認します。設定されている場合、プロキシーは Assisted Service URL へのアクセスを許可する必要があります。

16.2. 最小限の検出 ISO の問題をトラブルシューティングする

仮想メディア接続の帯域幅が制限されている場合は、最小限の ISO イメージを使用する必要があります。これには、ネットワークを使用してホストを起動するために必要なものだけが含まれています。コンテンツの大部分は、起動時にダウンロードされます。結果の ISO イメージのサイズは、フル ISO イメージの 1GB と比較して約 100MB です。

16.2.1. ブートプロセスを中断して最小限の ISO のブート失敗をトラブルシューティングする

お使いの環境で Assisted Installer サービスにアクセスするために静的ネットワーク設定が必要な場合、その設定に問題があると、最小限の ISO が適切にブートしない可能性があります。ホストがルートファイルシステムイメージのダウンロードに失敗したことがブート画面に表示される場合、ネットワークが正しく設定されていない可能性があります。

ブートストラッププロセスの早い段階で、ルートファイルシステムイメージがダウンロードされる前に、カーネルのブートを中断できます。これにより、ルートコンソールにアクセスしてネットワーク設定を確認できます。

rootfs ダウンロードの失敗例

```
[**    ] A start job is running for Acquire ▮rootfs image (1min 41s / no limit)[
  104.578592] coreos-livepxe-rootfs[922]: curl: (6) Could not resolve host: api.
openshift.com
[  104.579201] coreos-livepxe-rootfs[849]: Couldn't establish connectivity with
the server specified by:
[  104.579600] coreos-livepxe-rootfs[849]: coreos.live.rootfs_url=https://api.op
enshift.com/api/assisted-images/boot-artifacts/rootfs?arch=x86_64&version=4.11
[  104.580107] coreos-livepxe-rootfs[849]: Retrying in 5s...
[   **] A start job is running for Acquire ▮rootfs image (1min 46s / no limit)[
  109.619825] coreos-livepxe-rootfs[925]: curl: (6) Could not resolve host: api.
openshift.com
[  109.620608] coreos-livepxe-rootfs[849]: Couldn't establish connectivity with
the server specified by:
[  109.621053] coreos-livepxe-rootfs[849]: coreos.live.rootfs_url=https://api.op
enshift.com/api/assisted-images/boot-artifacts/rootfs?arch=x86_64&version=4.11
[  109.621564] coreos-livepxe-rootfs[849]: Retrying in 5s...
[   **] A start job is running for Acquire ▮rootfs image (1min 51s / no limit)[
  114.647843] coreos-livepxe-rootfs[928]: curl: (6) Could not resolve host: api.
openshift.com
[  114.648464] coreos-livepxe-rootfs[849]: Couldn't establish connectivity with
the server specified by:
[  114.648821] coreos-livepxe-rootfs[849]: coreos.live.rootfs_url=https://api.op
enshift.com/api/assisted-images/boot-artifacts/rootfs?arch=x86_64&version=4.11
[  114.649323] coreos-livepxe-rootfs[849]: Retrying in 5s...
[   **] A start job is running for Acquire ▮rootfs image (1min 53s / no limit)
```

手順

1. デプロイするクラスターの **infraEnv** オブジェクトに **.spec.kernelArguments** スタンザを追加します。

注記

インフラストラクチャー環境の変更の詳細は、**関連情報** を参照してください。

```
# ...
spec:
  clusterRef:
    name: sno1
    namespace: sno1
  cpuArchitecture: x86_64
  ipxeScriptType: DiscoveryImageAlways
  kernelArguments:
  - operation: append
    value: rd.break=initqueue (1)
  nmStateConfigLabelSelector:
    matchLabels:
      nmstate-label: sno1
  pullSecretRef:
    name: assisted-deployment-pull-secret
```

(1) **rd.break=initqueue** は、**dracut** メインループでブートを中断します。詳細は、rd.break options for debugging kernel boot を参照してください。

2. **rootfs** がダウンロードされる前に、関連するノードが自動的に再起動し、**inqueue** 段階でブートが中断するのを待ちます。ルートコンソールにリダイレクトされます。

3. 正しくないネットワーク設定を特定して変更します。以下に、役立つ診断コマンドをいくつか示します。

   a. **journalctl** を使用してシステムログを表示します。以下に例を示します。

   ```
   # journalctl -p err //Sorts logs by errors
   # journalctl -p crit //Sorts logs by critical errors
   # journalctl -p warning //Sorts logs by warnings
   ```

   b. 次のように、**nmcli** を使用してネットワーク接続情報を表示します。

   ```
   # nmcli conn show
   ```

   c. 設定ファイルを確認して正しくないネットワーク接続を探します。以下に例を示します。

   ```
   # cat /etc/assisted/network/host0/eno3.nmconnection
   ```

4. ブートストラッププロセスを再開するには、**Ctrl+D** を押します。サーバーが **rootfs** をダウンロードし、プロセスを完了します。

5. **infraEnv** オブジェクトを再度開き、**.spec.kernelArguments** スタンザを削除します。

関連情報

- インフラストラクチャー環境の変更

16.3. ホストの起動順序の修正

Discovery Image の一部として実行されるインストールが完了すると、Assisted Installer がホストを再起動します。 クラスターの形成を続行するには、ホストをインストールディスクから起動する必要があります。 ホストの起動順序を正しく設定していない場合、代わりに別のディスクから起動し、インストールが中断されます。

ホストが検出イメージを再度起動すると、Assisted Installer はこのイベントをすぐに検出し、ホストのステータスを **Installing Pending User Action** に設定します。 または、ホストが割り当てられた時間内に正しいディスクを起動したことを Assisted Installer が検出しない場合、このホストステータスも設定されます。

手順

- ホストを再起動し、その起動順序をインストールディスクから起動するように設定します。インストールディスクを選択しなかった場合は、Assisted Installer がディスクを選択します。選択したインストールディスクを表示するには、ホストインベントリーでホストの情報をクリックしてデプロイメントし、どのディスクにインストールディスクのロールがあるかを確認します。

16.4. 部分的に成功したインストールの修正

エラーが発生したにもかかわらず、Assisted Installer がインストールの成功を宣言する場合があります。

- OLM Operator のインストールを要求し、1 つ以上のインストールに失敗した場合は、クラスターのコンソールにログインして問題を修復します。

- 3 つ以上のワーカーノードのインストールをリクエストし、少なくとも 1 つがインストールに失敗したが、少なくとも 2 つが成功した場合は、インストールされたクラスターに失敗したワーカーを追加します。

16.5. クラスターにノード追加時の API 接続の失敗

Day 2 操作の一部としてノードを既存のクラスターに追加する場合、ノードは Day 1 クラスターから Ignition 設定ファイルをダウンロードします。ダウンロードが失敗し、ノードがクラスターに接続できない場合、**Host discovery** ステップのホストのステータスは **Insufficient** に変わります。このステータスをクリックすると、次のエラーメッセージが表示されます。

> The host failed to download the ignition file from <URL>. You must ensure the host can reach the URL. Check your DNS and network configuration or update the IP address or domain used to reach the cluster.
>
> error: ignition file download failed.... no route to host

接続の失敗にはさまざまな理由が考えられます。ここでは、推奨されるアクションをいくつか紹介します。

手順

1. クラスターの IP アドレスとドメイン名を確認します。
   a. **set the IP or domain used to reach the cluster**のハイパーリンクをクリックします。
   b. **Update cluster hostname**ウィンドウで、クラスターの正しい IP アドレスまたはドメイン名を入力します。
2. DNS 設定をチェックして、DNS が指定したドメインを解決できることを確認します。
3. すべてのファイアウォールでポート **22624** が開いていることを確認します。
4. ホストのエージェントログをチェックして、エージェントが SSH 経由で Assisted Service にアクセスできることを確認します。

```
$ sudo journalctl TAG=agent
```

注記

詳細は、エージェントが Assisted Service にアクセスできることを確認する を参照してください。