



# OpenShift Container Platform 4.11

## マシン管理

クラスターマシンの追加および保守



クラスターマシンの追加および保守

## 法律上の通知

Copyright © 2023 Red Hat, Inc.

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

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

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

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

## 概要

本書では、OpenShift Container Platform クラスターを設定するマシンを管理する方法を説明します。一部のタスクでは、OpenShift Container Platform クラスターの強化されたマシン管理機能を利用し、一部のタスクを手動で行うこともできます。本書で説明するすべてのタスクが必ずしもすべてのインストールタイプで利用可能である訳ではありません。

## 目次

<b>第1章 マシン管理の概要</b>	<b>4</b>
1.1. マシン API の概要	4
1.2. コンピューティングマシンの管理	5
1.3. OPENSIFT CONTAINER PLATFORM クラスターへの自動スケーリングの適用	6
1.4. ユーザーがプロビジョニングしたインフラストラクチャーへのコンピューティングマシンの追加	6
1.5. RHEL コンピュータマシンのクラスターへの追加	6
<b>第2章 MACHINE API を使用したコンピュータマシンの管理</b>	<b>7</b>
2.1. ALIBABA CLOUD でマシンセットを作成する	7
2.2. AWS でのマシンセットの作成	13
2.3. AZURE でのマシンセットの作成	19
2.4. AZURE STACK HUB でのマシンセットの作成	35
2.5. GCP でのマシンセットの作成	41
2.6. IBM CLOUD でのマシンセットの作成	51
2.7. NUTANIX でマシンセットを作成する	55
2.8. OPENSTACK でのマシンセットの作成	59
2.9. RHV でのマシンセットの作成	68
2.10. VSPHERE でのマシンセットの作成	73
2.11. ベアメタル上でのコンピュータマシンセットの作成	82
<b>第3章 マシンセットの手動によるスケーリング</b>	<b>87</b>
3.1. 前提条件	87
3.2. マシンセットの手動によるスケーリング	87
3.3. マシンセットの削除ポリシー	89
3.4. 関連情報	89
<b>第4章 マシンセットの変更</b>	<b>90</b>
4.1. マシンセットの変更	90
4.2. RHV 上の別のストレージドメインへのノードの移行	91
<b>第5章 マシンの削除</b>	<b>94</b>
5.1. 特定マシンの削除	94
5.2. マシン削除フェーズのライフサイクルフック	94
5.3. 関連情報	101
<b>第6章 OPENSIFT CONTAINER PLATFORM クラスターへの自動スケーリングの適用</b>	<b>102</b>
6.1. CLUSTER AUTOSCALER について	102
6.2. CLUSTER AUTOSCALER の設定	104
6.3. 次のステップ	106
6.4. MACHINE AUTOSCALER	106
6.5. MACHINE AUTOSCALER の設定	106
6.6. 関連情報	108
<b>第7章 インフラストラクチャーマシンセットの作成</b>	<b>109</b>
7.1. OPENSIFT CONTAINER PLATFORM インフラストラクチャーコンポーネント	109
7.2. 実稼働環境用のインフラストラクチャーマシンセットの作成	110
7.3. マシンセットリソースのインフラストラクチャーノードへの割り当て	140
7.4. リソースのインフラストラクチャーマシンセットへの移行	142
<b>第8章 RHEL コンピュータマシンの OPENSIFT CONTAINER PLATFORM クラスターへの追加</b>	<b>153</b>
8.1. RHEL コンピュータノードのクラスターへの追加について	153
8.2. RHEL コンピュータノードのシステム要件	153
8.3. クラウド用イメージの準備	155

8.4. PLAYBOOK 実行のためのマシンの準備	156
8.5. RHEL コンピュートノードの準備	157
8.6. AWS での RHEL インスタンスへのロールパーミッションの割り当て	159
8.7. 所有または共有されている RHEL ワーカーノードへのタグ付け	159
8.8. RHEL コンピュータマシンのクラスターへの追加	159
8.9. マシンの証明書署名要求の承認	160
8.10. ANSIBLE ホストファイルの必須パラメーター	163
<b>第9章 RHEL コンピュータマシンの OPENSIFT CONTAINER PLATFORM クラスターへのさらなる追加 ...</b>	<b>166</b>
9.1. RHEL コンピュートノードのクラスターへの追加について	166
9.2. RHEL コンピュートノードのシステム要件	166
9.3. クラウド用イメージの準備	168
9.4. RHEL コンピュートノードの準備	169
9.5. AWS での RHEL インスタンスへのロールパーミッションの割り当て	170
9.6. 所有または共有されている RHEL ワーカーノードへのタグ付け	171
9.7. RHEL コンピュータマシンのクラスターへのさらなる追加	171
9.8. マシンの証明書署名要求の承認	172
9.9. ANSIBLE ホストファイルの必須パラメーター	175
<b>第10章 ユーザーがプロビジョニングしたインフラストラクチャーを手動で管理する .....</b>	<b>177</b>
10.1. ユーザーがプロビジョニングしたインフラストラクチャーを使用してクラスターに計算マシンを手動で追加する	177
10.2. CLOUDFORMATION テンプレートの使用によるコンピュータマシンの AWS への追加	178
10.3. コンピューティングマシンを VSPHERE に手動で追加する	182
10.4. RHV 上のクラスターへのコンピュータマシンの追加	186
10.5. コンピュータマシンのベアメタルへの追加	187
<b>第11章 CLUSTER API によるマシンの管理 .....</b>	<b>194</b>
利点	194
制限事項	194
11.1. CLUSTER API アーキテクチャー	195
11.2. サンプル YAML ファイル	196
11.3. CLUSTER API マシンセットの作成	200
11.4. CLUSTER API を使用するクラスターのトラブルシューティング	204
<b>第12章 マシンヘルスチェックのデプロイ .....</b>	<b>205</b>
12.1. マシンのヘルスチェック	205
12.2. サンプル MACHINEHEALTHCHECK リソース	206
12.3. MACHINEHEALTHCHECK リソースの作成	208
12.4. ベアメタルの電源ベースの修復について	208



## 第1章 マシン管理の概要

マシン管理を使用して、Amazon Web Services (AWS)、Azure、Google Cloud Platform (GCP)、OpenStack、Red Hat Virtualization (RHV)、および vSphere などの基礎となるインフラストラクチャーを柔軟に使用して OpenShift Container Platform クラスターを管理できます。クラスターを制御し、特定のワークロードポリシーに基づいてクラスターをスケールアップやスケールダウンするなどの自動スケーリングを実行できます。

OpenShift Container Platform クラスターは、負荷の増減時に水平にスケールアップおよびスケールダウンできます。ワークロードの変更に適応するクラスターがあることが重要になります。

マシン管理は [カスタムリソース定義](#) (CRD) として実装されます。CRD オブジェクトは、クラスター内に新規の固有オブジェクト **Kind** を定義し、Kubernetes API サーバーはオブジェクトのライフサイクル全体を処理できます。

Machine API Operator は以下のリソースをプロビジョニングします。

- MachineSet
- マシン
- Cluster Autoscaler
- Machine Autoscaler
- Machine Health Checks

### 1.1. マシン API の概要

マシン API は、アップストリームのクラスター API プロジェクトおよびカスタム OpenShift Container Platform リソースに基づく重要なリソースの組み合わせです。

OpenShift Container Platform 4.11 クラスターの場合、マシン API はクラスターインストールの終了後にすべてのノードホストのプロビジョニングの管理アクションを実行します。このシステムにより、OpenShift Container Platform 4.11 はパブリックまたはプライベートのクラウドインフラストラクチャーに加えて弾力性があり、動的なプロビジョニング方法を提供します。

以下の 2 つのリソースは重要なリソースになります。

#### Machines

ノードのホストを記述する基本的なユニットです。マシンには、複数の異なるクラウドプラットフォーム用に提供されるコンピュートノードのタイプを記述する **providerSpec** 仕様があります。たとえば、Amazon Web Services (AWS) 上のワーカーノードのマシンタイプは特定のマシンタイプおよび必要なメタデータを定義する場合があります。

#### マシンセット

**MachineSet** リソースはマシンのグループです。マシンセットとマシンの関係は、レプリカセットと Pod の関係と同様です。マシンを追加する必要がある場合や、マシンの数を縮小したりする必要がある場合、コンピューティングのニーズに応じてマシンセットの **replicas** フィールドを変更します。

**警告**

コントロールプレーンマシンは、マシンセットで管理することはできません。

以下のカスタムリソースは、クラスターに機能を追加します。

**Machine Autoscaler**

**MachineAutoscaler** リソースは、クラウド内のコンピューティングマシンを自動的にスケーリングします。指定したコンピューティングマシンセット内のノードの最小および最大スケーリング境界を設定でき Machine Autoscaler はそのノード範囲を維持します。

**MachineAutoscaler** オブジェクトは **ClusterAutoscaler** オブジェクトの設定後に有効になります。**ClusterAutoscaler** および **MachineAutoscaler** リソースは、どちらも **ClusterAutoscalerOperator** オブジェクトによって利用可能にされます。

**Cluster Autoscaler**

このリソースはアップストリームの Cluster Autoscaler プロジェクトに基づいています。OpenShift Container Platform の実装では、これはマシンセット API を拡張することによってクラスター API に統合されます。コア、ノード、メモリー、および GPU などのリソースのクラスター全体でのスケーリング制限を設定できます。優先順位を設定することにより、重要度の低い Pod のために新規ノードがオンラインにならないようにクラスターで Pod の優先順位付けを実行できます。また、スケーリングポリシーを設定してノードをスケールダウンせずにスケールアップできるようにすることもできます。

**マシンのヘルスチェック**

**MachineHealthCheck** リソースはマシンの正常でない状態を検知し、マシンを削除し、サポートされているプラットフォームでは新規マシンを作成します。

OpenShift Container Platform バージョン 3.11 では、クラスターでマシンのプロビジョニングが管理されないためにマルチゾーンアーキテクチャーを容易に展開することができませんでした。しかし、OpenShift Container Platform バージョン 4.1 以降、このプロセスはより簡単になりました。それぞれのマシンセットのスコープが単一ゾーンに設定されるため、インストールプログラムはユーザーに代わって、アベイラビリティゾーン全体にマシンセットを送信します。さらに、コンピューティングは動的に展開されるため、ゾーンに障害が発生した場合の、マシンのリバランスが必要な場合に使用するゾーンを常に確保できます。複数のアベイラビリティゾーンを持たないグローバル Azure リージョンでは、アベイラビリティセットを使用して高可用性を確保できます。Autoscaler はクラスターの有効期間中にベストエフォートでバランシングを提供します。

## 1.2. コンピューティングマシンの管理

クラスター管理者として、以下を実行できます。

- 以下でマシンセットを作成する。
  - [AWS](#)
  - [Azure](#)
  - [GCP](#)
  - [OpenStack](#)

- [RHV](#)
- [vSphere](#)
- [ベアメタルのデプロイメント用マシンセットの作成: ベアメタルでのコンピュートマシンセットの作成](#)
- マシンセットからマシンを追加または削除して、[マシンセットを手動でスケーリング](#)する。
- **MachineSet** YAML 設定ファイルを使用して [マシンセットを変更](#) します。
- マシンを [削除](#)する。
- [インフラストラクチャーマシンセットを作成](#)する。
- [マシンヘルスチェック](#) を設定してデプロイし、マシンプール内の破損したマシンを自動的に修正する。

### 1.3. OPENSIFT CONTAINER PLATFORM クラスターへの自動スケーリングの適用

ワークロードの変化に対する柔軟性を確保するために、OpenShift Container Platform クラスターを自動的にスケーリングできます。クラスターを [自動スケーリング](#) するには、Cluster Autoscaler をデプロイしてから、各コンピュートマシンセットに Machine Autoscaler をデプロイする必要があります。

- **Cluster Autoscaler** は、デプロイメントのニーズに応じてクラスターのサイズを拡大し、縮小します。
- **Machine Autoscaler** は、OpenShift Container Platform クラスターにデプロイするマシンセットのコンピュートマシン数を調整します。

### 1.4. ユーザーがプロビジョニングしたインフラストラクチャーへのコンピューティングマシンの追加

ユーザーによってプロビジョニングされるインフラストラクチャーとは、OpenShift Container Platform をホストするコンピュート、ネットワーク、ストレージリソースなどのインフラストラクチャーをデプロイできる環境です。インストールプロセス中またはインストールプロセス後に、ユーザーがプロビジョニングしたインフラストラクチャー上のクラスターに [コンピューティングマシンを追加](#) できます。

### 1.5. RHEL コンピュートマシンのクラスターへの追加

クラスター管理者は、次のアクションを実行できます。

- ユーザーによってプロビジョニングされるインフラストラクチャークラスターまたはインストールでプロビジョニングされるクラスターに、[Red Hat Enterprise Linux \(RHEL\) コンピュートマシン \(ワーカーマシンとしても知られる\)](#) を追加する。
- 既存のクラスターに [Red Hat Enterprise Linux \(RHEL\) コンピュートマシン](#) をさらに追加する。

## 第2章 MACHINE API を使用したコンピュータマシンの管理

### 2.1. ALIBABA CLOUD でマシンセットを作成する

Alibaba Cloud 上の Open Shift Container Platform クラスターで特定の目的を果たすために別のマシンセットを作成できます。たとえば、インフラストラクチャーマシンセットおよび関連マシンを作成して、サポートするワークロードを新しいマシンに移動できます。

#### 重要

高度なマシン管理およびスケーリング機能は、マシン API が機能しているクラスターでのみ使用することができます。ユーザーがプロビジョニングしたインフラストラクチャーを持つクラスターでは、マシン API を使用するために追加の検証と設定が必要です。

インフラストラクチャープラットフォームタイプが **none** のクラスターは、マシン API を使用できません。この制限は、クラスターに接続されている計算マシンが、この機能をサポートするプラットフォームにインストールされている場合でも適用されます。このパラメーターは、インストール後に変更することはできません。

クラスターのプラットフォームタイプを表示するには、以下のコマンドを実行します。

```
$ oc get infrastructure cluster -o jsonpath='{.status.platform}'
```

#### 2.1.1. Alibaba Cloud 上のマシンセットカスタムリソースのサンプル YAML

このサンプル YAML は、リージョン内の指定された Alibaba Cloud ゾーンで実行され、**node-role.kubernetes.io/<role>: ""** というラベルの付いたノードを作成するマシンセットを定義します。

このサンプルでは、**<infrastructure\_id>** はクラスターのプロビジョニング時に設定したクラスター ID に基づくインフラストラクチャー ID であり、**<role>** は追加するノードラベルです。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
    machine.openshift.io/cluster-api-machine-role: <role> 2
    machine.openshift.io/cluster-api-machine-type: <role> 3
  name: <infrastructure_id>-<role>-<zone> 4
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<zone> 6
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 7
        machine.openshift.io/cluster-api-machine-role: <role> 8
```

```

machine.openshift.io/cluster-api-machine-type: <role> 9
machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<zone> 10
spec:
  metadata:
    labels:
      node-role.kubernetes.io/<role>: ""
  providerSpec:
    value:
      apiVersion: machine.openshift.io/v1
      credentialsSecret:
        name: alibabacloud-credentials
      imageId: <image_id> 11
      instanceType: <instance_type> 12
      kind: AlibabaCloudMachineProviderConfig
      ramRoleName: <infrastructure_id>-role-worker 13
      regionId: <region> 14
      resourceGroup: 15
        id: <resource_group_id>
        type: ID
      securityGroups:
        - tags: 16
            - Key: Name
              Value: <infrastructure_id>-sg-<role>
            type: Tags
      systemDisk: 17
        category: cloud_essd
        size: <disk_size>
      tag: 18
        - Key: kubernetes.io/cluster/<infrastructure_id>
          Value: owned
      userDataSecret:
        name: <user_data_secret> 19
      vSwitch:
        tags: 20
        - Key: Name
          Value: <infrastructure_id>-vswitch-<zone>
        type: Tags
      vpId: ""
      zoneId: <zone> 21

```

- 1 5 7 クラスターのプロビジョニング時に設定したクラスター ID を基にするインフラストラクチャー ID を指定します。OpenShift CLI (**oc**) がインストールされている場合は、以下のコマンドを実行してインフラストラクチャー ID を取得できます。

```
$ oc get -o jsonpath='{.status.infrastructureName}' infrastructure cluster
```

- 2 3 8 9 追加するノードラベルを指定します。

- 4 6 10 インフラストラクチャー ID、ノードラベル、およびゾーンを指定します。

- 11 使用するイメージを指定します。クラスターに設定されている既存のデフォルトマシンのイメージを使用します。

- 12 マシンセットに使用するインスタンスタイプを指定します。

- 13 マシンセットに使用する RAM ロールの名前を指定します。インストーラーがデフォルトのマシンセットに入力する値を使用します。
- 14 マシンを配置するリージョンを指定します。
- 15 クラスターのリソースグループとタイプを指定します。インストーラーがデフォルトのマシンセットに入力する値を使用するか、別の値を指定できます。
- 16 18 20 マシンセットに使用するタグを指定します。少なくとも、この例に示されているタグを、クラスターに適切な値とともに含める必要があります。必要に応じて、インストーラーが作成するデフォルトのマシンセットに設定するタグなど、追加のタグを含めることができます。
- 17 ルートディスクのタイプとサイズを指定します。インストーラーが作成するデフォルトのマシンセットに入力する **category** 値を使用します。必要に応じて、**size** にギガバイト単位の別の値を指定します。
- 19 **openshift-machine-api** 名前空間にあるユーザーデータ YAML ファイルでシークレットの名前を指定します。インストーラーがデフォルトのマシンセットに入力する値を使用します。
- 21 マシンを配置するリージョン内のゾーンを指定します。リージョンがゾーンをサポートすることを確認してください。

### 2.1.1.1. Alibaba Cloud 使用統計のマシンセットパラメーター

インストーラーが Alibaba Cloud クラスター用に作成するデフォルトのマシンセットには、Alibaba Cloud が使用統計を追跡するために内部的に使用する不要なタグ値が含まれています。これらのタグは、**spec.template.spec.providerSpec.value** リストの **securityGroups**、**tag**、および **vSwitch** パラメーターに入力されます。

追加のマシンをデプロイするマシンセットを作成するときは、必要な Kubernetes タグを含める必要があります。使用統計タグは、作成するマシンセットで指定されていない場合でも、デフォルトで適用されます。必要に応じて、追加のタグを含めることもできます。

次の YAML スニペットは、デフォルトのマシンセットのどのタグがオプションでどれが必須かを示しています。

#### **spec.template.spec.providerSpec.value.securityGroups** のタグ

```
spec:
  template:
    spec:
      providerSpec:
        value:
          securityGroups:
            - tags:
              - Key: kubernetes.io/cluster/<infrastructure_id> 1
                Value: owned
              - Key: GISV
                Value: ocp
              - Key: sigs.k8s.io/cloud-provider-alibaba/origin 2
                Value: ocp
              - Key: Name
                Value: <infrastructure_id>-sg-<role> 3
            type: Tags
```

**1 2** オプション: このタグは、マシンセットで指定されていない場合でも適用されます。

**3** 必須。

ここでは、以下のようになります。

- **<infrastructure\_id>** は、クラスターのプロビジョニング時に設定したクラスター ID に基づくインフラストラクチャー ID です。
- **<role>** は、追加するノードラベルです。

### spec.template.spec.providerSpec.value.tag のタグ

```
spec:
  template:
    spec:
      providerSpec:
        value:
          tag:
            - Key: kubernetes.io/cluster/<infrastructure_id> 1
              Value: owned
            - Key: GISV 2
              Value: ocp
            - Key: sigs.k8s.io/cloud-provider-alibaba/origin 3
              Value: ocp
```

**2 3** オプション: このタグは、マシンセットで指定されていない場合でも適用されます。

**1** 必須。

**<infrastructure\_id>** は、クラスターのプロビジョニング時に設定したクラスター ID に基づくインフラストラクチャー ID です。

### spec.template.spec.providerSpec.value.vSwitch のタグ

```
spec:
  template:
    spec:
      providerSpec:
        value:
          vSwitch:
            tags:
              - Key: kubernetes.io/cluster/<infrastructure_id> 1
                Value: owned
              - Key: GISV 2
                Value: ocp
              - Key: sigs.k8s.io/cloud-provider-alibaba/origin 3
                Value: ocp
              - Key: Name
                Value: <infrastructure_id>-vswitch-<zone> 4
            type: Tags
```

**1 2 3** オプション: このタグは、マシンセットで指定されていない場合でも適用されます。

**4** 必須。

ここでは、以下のようになります。

- **<infrastructure\_id>** は、クラスターのプロビジョニング時に設定したクラスター ID に基づくインフラストラクチャー ID です。
- **<zone>** は、マシンを配置するリージョン内のゾーンです。

## 2.1.2. マシンセットの作成

インストールプログラムによって作成されるコンピュータセットセットに加えて、独自のマシンセットを作成して、選択した特定のワークロードのマシンコンピューティングリソースを動的に管理できます。

### 前提条件

- OpenShift Container Platform クラスターをデプロイすること。
- OpenShift CLI (**oc**) がインストールされている。
- **cluster-admin** パーミッションを持つユーザーとして、**oc** にログインする。

### 手順

1. 説明されているようにマシンセット カスタムリソース (CR) サンプルを含む新規 YAML ファイルを作成し、そのファイルに **<file\_name>.yaml** という名前を付けます。  
**<clusterID>** および **<role>** パラメーターの値を設定していることを確認します。
2. オプション: 特定のフィールドに設定する値がわからない場合は、クラスターから既存のコンピュータマシンセットを確認できます。
  - a. クラスター内のコンピュータマシンセットをリスト表示するには、次のコマンドを実行します。

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

### 出力例

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

- b. 特定のコンピュータマシンセットカスタムリソース(CR)の値を表示するには、以下のコマンドを実行します。

```
$ oc get machineset <machineset_name> \
  -n openshift-machine-api -o yaml
```

## 出力例

```

apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> ❶
    name: <infrastructure_id>-<role> ❷
    namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>
        machine.openshift.io/cluster-api-machine-role: <role>
        machine.openshift.io/cluster-api-machine-type: <role>
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
    spec:
      providerSpec: ❸
      ...

```

- ❶ クラスターインフラストラクチャー ID。
- ❷ デフォルトのノードラベル。



## 注記

ユーザーがプロビジョニングしたインフラストラクチャーを持つクラスターの場合、コンピュートマシンセットは **worker** および **infra** タイプのマシンのみを作成できます。

- ❸ コンピュートマシンセット CR の **<providerSpec>** セクションの値は、プラットフォーム固有です。CR の **<providerSpec>** パラメーターの詳細については、プロバイダーのサンプルコンピュートマシンセット CR 設定を参照してください。

3. 次のコマンドを実行して **MachineSet** CR を作成します。

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

## 検証

- 次のコマンドを実行して、コンピュートマシンセットのリストを表示します。

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

## 出力例

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
agl030519-vplxk-infra-us-east-1a	1	1	1	1	11m
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

新規のマシンセットが利用可能な場合、**DESIRED** および **CURRENT** の値は一致します。マシンセットが利用可能でない場合、数分待機してからコマンドを再度実行します。

## 2.2. AWS でのマシンセットの作成

Amazon Web Services (AWS) で OpenShift Container Platform クラスターの特定の目的を果たすように異なるマシンセットを作成することができます。たとえば、インフラストラクチャーマシンセットおよび関連マシンを作成して、サポートするワークロードを新しいマシンに移動できます。

### 重要

高度なマシン管理およびスケーリング機能は、マシン API が機能しているクラスターでのみ使用することができます。ユーザーがプロビジョニングしたインフラストラクチャーを持つクラスターでは、マシン API を使用するために追加の検証と設定が必要です。

インフラストラクチャープラットフォームタイプが **none** のクラスターは、マシン API を使用できません。この制限は、クラスターに接続されている計算マシンが、この機能をサポートするプラットフォームにインストールされている場合でも適用されます。このパラメーターは、インストール後に変更することはできません。

クラスターのプラットフォームタイプを表示するには、以下のコマンドを実行します。

```
$ oc get infrastructure cluster -o jsonpath='{.status.platform}'
```

### 2.2.1. AWS 上のマシンセットカスタムリソースのサンプル YAML

このサンプル YAML は **us-east-1a** Amazon Web Services (AWS) ゾーンで実行され、**node-role.kubernetes.io/<role>:""** というラベルが付けられたノードを作成するマシンセットを定義します。

このサンプルでは、**<infrastructure\_id>** はクラスターのプロビジョニング時に設定したクラスター ID に基づくインフラストラクチャー ID であり、**<role>** は追加するノードラベルです。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> ❶
  name: <infrastructure_id>-<role>-<zone> ❷
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
```

```

matchLabels:
  machine.openshift.io/cluster-api-cluster: <infrastructure_id> 3
  machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<zone> 4
template:
  metadata:
    labels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
      machine.openshift.io/cluster-api-machine-role: <role> 6
      machine.openshift.io/cluster-api-machine-type: <role> 7
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<zone> 8
  spec:
    metadata:
      labels:
        node-role.kubernetes.io/<role>: "" 9
    providerSpec:
      value:
        ami:
          id: ami-046fe691f52a953f9 10
        apiVersion: awsproviderconfig.openshift.io/v1beta1
        blockDevices:
          - ebs:
              iops: 0
              volumeSize: 120
              volumeType: gp2
        credentialsSecret:
          name: aws-cloud-credentials
        deviceIndex: 0
        iamInstanceProfile:
          id: <infrastructure_id>-worker-profile 11
        instanceType: m6i.large
        kind: AWSMachineProviderConfig
        placement:
          availabilityZone: <zone> 12
          region: <region> 13
        securityGroups:
          - filters:
              - name: tag:Name
                values:
                  - <infrastructure_id>-worker-sg 14
        subnet:
          filters:
            - name: tag:Name
              values:
                - <infrastructure_id>-private-<zone> 15
        tags:
          - name: kubernetes.io/cluster/<infrastructure_id> 16
            value: owned
        userDataSecret:
          name: worker-user-data

```

1 3 5 11 14 16 クラスターのプロビジョニング時に設定したクラスター ID を基にするインフラストラクチャー ID を指定します。OpenShift CLI がインストールされている場合は、以下のコマンドを実行してインフラストラクチャー ID を取得できます。

```
$ oc get -o jsonpath='{.status.infrastructureName}' infrastructure cluster
```

**2 4 8** インフラストラクチャー ID、ノードラベル、およびゾーンを指定します。

**6 7 9** 追加するノードラベルを指定します。

**10** OpenShift Container Platform ノードの AWS ゾーンに有効な Red Hat Enterprise Linux CoreOS (RHCOS) AMI を指定します。AWS Marketplace イメージを使用する場合は、[AWS Marketplace](#) から OpenShift Container Platform サブスクリプションを完了して、リージョンの AMI ID を取得する必要があります。

**12** ゾーン (例: **us-east-1a**) を指定します。

**13** リージョン (例: **us-east-1**) を指定します。

**15** インフラストラクチャー ID とゾーンを指定します。

## 2.2.2. マシンセットの作成

インストールプログラムによって作成されるコンピュートセットに加えて、独自のマシンセットを作成して、選択した特定のワークロードのマシンコンピューティングリソースを動的に管理できます。

### 前提条件

- OpenShift Container Platform クラスターをデプロイすること。
- OpenShift CLI (**oc**) がインストールされている。
- **cluster-admin** パーミッションを持つユーザーとして、**oc** にログインする。

### 手順

1. 説明されているようにマシンセット カスタムリソース (CR) サンプルを含む新規 YAML ファイルを作成し、そのファイルに **<file\_name>.yaml** という名前を付けます。  
**<clusterID>** および **<role>** パラメーターの値を設定していることを確認します。
2. オプション: 特定のフィールドに設定する値がわからない場合は、クラスターから既存のコンピュートマシンセットを確認できます。
  - a. クラスター内のコンピュートマシンセットをリスト表示するには、次のコマンドを実行します。

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

### 出力例

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

- b. 特定のコンピュートマシンセットカスタムリソース(CR)の値を表示するには、以下のコマンドを実行します。

```
$ oc get machineset <machineset_name> \
-n openshift-machine-api -o yaml
```

### 出力例

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> ❶
  name: <infrastructure_id>-<role> ❷
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>
        machine.openshift.io/cluster-api-machine-role: <role>
        machine.openshift.io/cluster-api-machine-type: <role>
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
    spec:
      providerSpec: ❸
      ...
```

- ❶ クラスターインフラストラクチャー ID。
- ❷ デフォルトのノードラベル。



### 注記

ユーザーがプロビジョニングしたインフラストラクチャーを持つクラスターの場合、コンピュートマシンセットは **worker** および **infra** タイプのマシンのみを作成できます。

- ❸ コンピュートマシンセット CR の **<providerSpec>** セクションの値は、プラットフォーム固有です。CR の **<providerSpec>** パラメーターの詳細については、プロバイダーのサンプルコンピュートマシンセット CR 設定を参照してください。

3. 次のコマンドを実行して **MachineSet** CR を作成します。

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

4. 他のアベイラビリティゾーンでコンピュートマシンセットが必要な場合、このプロセスを繰り返して追加のコンピュートマシンセットを作成します。

## 検証

- 次のコマンドを実行して、コンピュートマシンセットのリストを表示します。

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

## 出力例

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
agl030519-vplxk-infra-us-east-1a	1	1	1	1	11m
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

新規のマシンセットが利用可能な場合、**DESIRED** および **CURRENT** の値は一致します。マシンセットが利用可能でない場合、数分待機してからコマンドを再度実行します。

## 2.2.3. Amazon EC2 インスタンスメタデータサービスのマシンセットオプション

マシンセットを使用して、Amazon EC2 インスタンスメタデータサービス (IMDS) の特定のバージョンを使用するコンピュートマシンを作成できます。マシンセットは、IMDSv1 と [IMDSv2](#) の両方の使用を許可するコンピュートマシン、または IMDSv2 の使用を必要とするコンピュートマシンを作成できます。



## 注記

IMDSv2 の使用は、OpenShift Container Platform バージョン 4.7 以降で作成された AWS クラスターでのみサポートされます。

既存のコンピュートマシンの IMDS 設定を変更するには、それらのマシンを管理するマシンセット YAML ファイルを編集します。好みの IMDS 設定で新しいコンピュートマシンを展開するには、適切な値を使用してマシンセット YAML ファイルを作成します。

コントロールプレーンマシンの IMDS 設定は、クラスターのインストール時に設定されます。コントロールプレーンマシンの IMDS 設定を変更するには、AWS CLI を使用する必要があります。詳細は、[既存のインスタンスのインスタンスメタデータオプション](#) を変更する方法に関する AWS のドキュメントを参照してください。



## 重要

IMDSv2 を必要とするコンピュートマシンを作成するようにマシンセットを設定する前に、AWS メタデータサービスと相互作用するすべてのワークロードが IMDSv2 をサポートしていることを確認してください。

## 2.2.3.1. マシンセットを使用した IMDS の設定

コンピュートマシンのマシンセット YAML ファイルで **metadataServiceOptions.authentication** の値を追加または編集することで、IMDSv2 の使用を要求するかどうかを指定できます。

## 前提条件

- IMDSv2 を使用するには、AWS クラスターが OpenShift Container Platform バージョン 4.7 以降で作成されている必要があります。

## 手順

- **providerSpec** フィールドの下に次の行を追加または編集します。

```
providerSpec:
  value:
    metadataServiceOptions:
      authentication: Required ❶
```

- ❶ IMDSv2 を要求するには、パラメーター値を **Required** に設定します。IMDSv1 と IMDSv2 の両方の使用を許可するには、パラメーター値を **Optional** に設定します。値が指定されていない場合、IMDSv1 と IMDSv2 の両方が許可されます。

### 2.2.4. マシンを専有インスタンス (Dedicated Instance) としてデプロイするマシンセット

マシンを専有インスタンス (Dedicated Instance) としてデプロイする AWS で実行されるマシンセットを作成できます。専有インスタンス (Dedicated Instance) は、単一のお客様専用のハードウェア上の仮想プライベートクラウド (VPC) で実行されます。これらの Amazon EC2 インスタンスは、ホストのハードウェアレベルで物理的に分離されます。インスタンスが単一の有料アカウントにリンクされている別の AWS アカウントに属する場合でも、専有インスタンス (Dedicated Instance) の分離が生じます。ただし、専用ではない他のインスタンスは、それらが同じ AWS アカウントに属する場合は、ハードウェアを専有インスタンス (Dedicated Instance) と共有できます。

パブリックまたは専用テナンシーのいずれかを持つインスタンスは、マシン API によってサポートされます。パブリックテナンシーを持つインスタンスは、共有ハードウェア上で実行されます。パブリックテナンシーはデフォルトのテナンシーです。専用のテナンシーを持つインスタンスは、単一テナントのハードウェアで実行されます。

#### 2.2.4.1. マシンセットの使用による専有インスタンス (Dedicated Instance) の作成

マシン API 統合を使用して、専有インスタンス (Dedicated Instance) によってサポートされるマシンを実行できます。マシンセット YAML ファイルの **tenancy** フィールドを設定し、AWS で専有インスタンス (Dedicated Instance) を起動します。

## 手順

- **providerSpec** フィールドに専用テナンシーを指定します。

```
providerSpec:
  placement:
    tenancy: dedicated
```

### 2.2.5. マシンを Spot インスタンスとしてデプロイするマシンセット

マシンを保証されていない Spot インスタンスとしてデプロイする AWS で実行されるマシンセットを作成して、コストを節約できます。Spot インスタンスは未使用の AWS EC2 容量を使用し、On-Demand インスタンスよりもコストが低くなります。Spot インスタンスは、バッチやステートレス、水平的に拡張可能なワークロードなどの割り込みを許容できるワークロードに使用することができます。

AWS EC2 は Spot インスタンスをいつでも終了できます。AWS は、中断の発生時にユーザーに警告を 2 分間表示します。OpenShift Container Platform は、AWS が終了についての警告を発行する際に影響を受けるインスタンスからワークロードを削除し始めます。

以下の理由により、Spot インスタンスを使用すると中断が生じる可能性があります。

- インスタンス価格は最大価格を超えます。
- Spot インスタンスの需要は増大します。
- Spot インスタンスの供給は減少します。

AWS がインスタンスを終了すると、Spot インスタンスノードで実行される終了ハンドラーによりマシンリソースが削除されます。マシンセットの **replicas** の量を満たすために、マシンセットは Spot インスタンスを要求するマシンを作成します。

### 2.2.5.1. マシンセットの使用による Spot インスタンスの作成

**spotMarketOptions** をマシンセットの YAML ファイルに追加して、AWS で Spot インスタンスを起動できます。

#### 手順

- **providerSpec** フィールドの下に以下の行を追加します。

```
providerSpec:
  value:
    spotMarketOptions: {}
```

オプションで、Spot インスタンスのコストを制限するために、**spotMarketOptions.maxPrice** フィールドを設定できます。たとえば、**maxPrice: '2.50'** を設定できます。

**maxPrice** が設定されている場合、この値は毎時の最大 Spot 価格として使用されます。これを設定しないと、デフォルトで最大価格として On-Demand インスタンス価格までチャージされます。



#### 注記

デフォルトの On-Demand 価格を **maxPrice** 値として使用し、Spot インスタンスの最大価格を設定しないことが強く推奨されます。

## 2.3. AZURE でのマシンセットの作成

Microsoft Azure 上の OpenShift Container Platform クラスターで特定の目的を果たすように異なるマシンセットを作成することができます。たとえば、インフラストラクチャマシンセットおよび関連マシンを作成して、サポートするワークロードを新しいマシンに移動できます。

## 重要

高度なマシン管理およびスケーリング機能は、マシン API が機能しているクラスターでのみ使用することができます。ユーザーがプロビジョニングしたインフラストラクチャーを持つクラスターでは、マシン API を使用するために追加の検証と設定が必要です。

インフラストラクチャープラットフォームタイプが **none** のクラスターは、マシン API を使用できません。この制限は、クラスターに接続されている計算マシンが、この機能をサポートするプラットフォームにインストールされている場合でも適用されます。このパラメーターは、インストール後に変更することはできません。

クラスターのプラットフォームタイプを表示するには、以下のコマンドを実行します。

```
$ oc get infrastructure cluster -o jsonpath='{.status.platform}'
```

### 2.3.1. Azure 上のマシンセットのカスタムリソースのサンプル YAML

このサンプル YAML は、リージョンの **1** Microsoft Azure ゾーンで実行され、**node-role.kubernetes.io/<role>: ""** というラベルの付けられたノードを作成するマシンセットを定義します。

このサンプルでは、**<infrastructure\_id>** はクラスターのプロビジョニング時に設定したクラスター ID に基づくインフラストラクチャー ID であり、**<role>** は追加するノードラベルです。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
    machine.openshift.io/cluster-api-machine-role: <role> 2
    machine.openshift.io/cluster-api-machine-type: <role> 3
  name: <infrastructure_id>-<role>-<region> 4
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<region> 6
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 7
        machine.openshift.io/cluster-api-machine-role: <role> 8
        machine.openshift.io/cluster-api-machine-type: <role> 9
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<region> 10
    spec:
      metadata:
        creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-machineset: <machineset_name> 11
        node-role.kubernetes.io/<role>: "" 12
```

```

providerSpec:
  value:
    apiVersion: azureproviderconfig.openshift.io/v1beta1
    credentialsSecret:
      name: azure-cloud-credentials
      namespace: openshift-machine-api
    image: 13
    offer: ""
    publisher: ""
    resourceID: /resourceGroups/<infrastructure_id>-
rg/providers/Microsoft.Compute/images/<infrastructure_id> 14
    sku: ""
    version: ""
    internalLoadBalancer: ""
    kind: AzureMachineProviderSpec
    location: <region> 15
    managedIdentity: <infrastructure_id>-identity 16
    metadata:
      creationTimestamp: null
    natRule: null
    networkResourceGroup: ""
    osDisk:
      diskSizeGB: 128
      managedDisk:
        storageAccountType: Premium_LRS
      osType: Linux
    publicIP: false
    publicLoadBalancer: ""
    resourceGroup: <infrastructure_id>-rg 17
    sshPrivateKey: ""
    sshPublicKey: ""
    tags:
      - name: <custom_tag_name> 18
        value: <custom_tag_value> 19
    subnet: <infrastructure_id>-<role>-subnet 20 21
    userDataSecret:
      name: worker-user-data 22
    vmSize: Standard_D4s_v3
    vnet: <infrastructure_id>-vnet 23
    zone: "1" 24

```

1 5 7 16 17 20 23 クラスターのプロビジョニング時に設定したクラスター ID を基にするインフラストラクチャー ID を指定します。OpenShift CLI がインストールされている場合は、以下のコマンドを実行してインフラストラクチャー ID を取得できます。

```
$ oc get -o jsonpath='{.status.infrastructureName}{"\n"}' infrastructure cluster
```

以下のコマンドを実行してサブネットを取得できます。

```
$ oc -n openshift-machine-api \
  -o jsonpath='{.spec.template.spec.providerSpec.value.subnet}{"\n"}' \
  get machineset/<infrastructure_id>-worker-centralus1
```

以下のコマンドを実行して vnet を取得できます。

```
$ oc -n openshift-machine-api \
  -o jsonpath='{.spec.template.spec.providerSpec.value.vnet}' \
  get machineset/<infrastructure_id>-worker-centralus1
```

**2 3 8 9 12 21 22** 追加するノードラベルを指定します。

**4 6 10** インフラストラクチャー ID、ノードラベル、およびリージョンを指定します。

**11** オプション: 可用性セットの使用を有効にするためにマシンセットの名前を指定します。この設定は、新規コンピュートマシンにのみ適用されます。

**13** マシンセットのイメージの詳細を指定します。Azure Marketplace イメージを使用する場合は、Azure Marketplace イメージの選択を参照してください。

**14** インスタンスタイプと互換性のあるイメージを指定します。インストールプログラムによって作成された Hyper-V 世代の V2 イメージには接尾辞 **-gen2** が付いていますが、V1 イメージには接尾辞のない同じ名前が付いています。

**15** マシンを配置するリージョンを指定します。

**24** マシンを配置するリージョン内のゾーンを指定します。リージョンがゾーンをサポートすることを確認してください。

**18 19** オプション: マシンセットでカスタムタグを指定します。<custom\_tag\_name> フィールドにタグ名を指定し、対応するタグ値を <custom\_tag\_value> フィールドに指定します。

### 2.3.2. マシンセットの作成

インストールプログラムによって作成されるコンピュートセットセットに加えて、独自のマシンセットを作成して、選択した特定のワークロードのマシンコンピューティングリソースを動的に管理できます。

#### 前提条件

- OpenShift Container Platform クラスターをデプロイすること。
- OpenShift CLI (**oc**) がインストールされている。
- **cluster-admin** パーミッションを持つユーザーとして、**oc** にログインする。

#### 手順

1. 説明されているようにマシンセット カスタムリソース (CR) サンプルを含む新規 YAML ファイルを作成し、そのファイルに <file\_name>.yaml という名前を付けます。  
<clusterID> および <role> パラメーターの値を設定していることを確認します。
2. オプション: 特定のフィールドに設定する値がわからない場合は、クラスターから既存のコンピュートマシンセットを確認できます。
  - a. クラスター内のコンピュートマシンセットをリスト表示するには、次のコマンドを実行します。

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

## 出力例

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

- b. 特定のコンピュータマシンセットカスタムリソース(CR)の値を表示するには、以下のコマンドを実行します。

```
$ oc get machineset <machineset_name> \
-n openshift-machine-api -o yaml
```

## 出力例

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> ❶
    name: <infrastructure_id>-<role> ❷
    namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>
        machine.openshift.io/cluster-api-machine-role: <role>
        machine.openshift.io/cluster-api-machine-type: <role>
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
    spec:
      providerSpec: ❸
      ...
```

❶ クラスターインフラストラクチャー ID。

❷ デフォルトのノードラベル。



## 注記

ユーザーがプロビジョニングしたインフラストラクチャーを持つクラスターの場合、コンピュータマシンセットは **worker** および **infra** タイプのマシンのみを作成できます。

❸

コンピュートマシンセット CR の **<providerSpec>** セクションの値は、プラットフォーム固有です。CR の **<providerSpec>** パラメーターの詳細については、プロバ

3. 次のコマンドを実行して **MachineSet** CR を作成します。

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

## 検証

- 次のコマンドを実行して、コンピュートマシンセットのリストを表示します。

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

## 出力例

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
agl030519-vplxk-infra-us-east-1a	1	1	1	1	11m
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

新規のマシンセットが利用可能な場合、**DESIRED** および **CURRENT** の値は一致します。マシンセットが利用可能でない場合、数分待機してからコマンドを再度実行します。

## 2.3.3. Azure Marketplace イメージの選択

Azure Marketplace サービスを使用するマシンをデプロイする、Azure で実行するマシンセットを作成できます。このサービスを使用するには、まず Azure Marketplace イメージを取得する必要があります。イメージを取得するときは、次の点を考慮してください。

- イメージは同じですが、Azure Marketplace のパブリシャーは地域によって異なります。北米にお住まいの場合は、**redhat** をパブリッシャーとして指定してください。EMEA にお住まいの場合は、**redhat-limited** をパブリッシャーとして指定してください。
- このオファーには、**rh-ocp-worker** SKU と **rh-ocp-worker-gen1** SKU が含まれています。**rh-ocp-worker** SKU は、Hyper-V 世代のバージョン 2 VM イメージを表します。OpenShift Container Platform で使用されるデフォルトのインスタンスタイプは、バージョン 2 と互換性があります。バージョン 1 のみと互換性のあるインスタンスタイプを使用する場合は、**rh-ocp-worker-gen1** SKU に関連付けられたイメージを使用します。**rh-ocp-worker-gen1** SKU は、Hyper-V バージョン 1 VM イメージを表します。

## 前提条件

- Azure CLI クライアント (**az**) をインストールしている。
- お客様の Azure アカウントにはオファーのエンタイトルメントがあり、Azure CLI クライアントを使用してこのアカウントにログインしている。

## 手順

- 以下のいずれかのコマンドを実行して、利用可能なすべての OpenShift Container Platform イメージを表示します。

- 北米:

```
$ az vm image list --all --offer rh-ocp-worker --publisher redhat -o table
```

#### 出力例

Offer	Publisher	Skus	Urn	Version
rh-ocp-worker	RedHat	rh-ocp-worker	RedHat:rh-ocp-worker:rh-ocpworker:4.8.2021122100	4.8.2021122100
rh-ocp-worker-gen1	RedHat	rh-ocp-worker-gen1	RedHat:rh-ocp-worker:rh-ocp-worker-gen1:4.8.2021122100	4.8.2021122100

- EMEA:

```
$ az vm image list --all --offer rh-ocp-worker --publisher redhat-limited -o table
```

#### 出力例

Offer	Publisher	Skus	Urn	Version
rh-ocp-worker	redhat-limited	rh-ocp-worker	redhat-limited:rh-ocp-worker:rh-ocp-worker:4.8.2021122100	4.8.2021122100
rh-ocp-worker-gen1	redhat-limited	rh-ocp-worker-gen1	redhat-limited:rh-ocp-worker:rh-ocp-worker-gen1:4.8.2021122100	4.8.2021122100



#### 注記

インストールする OpenShift Container Platform のバージョンに関係なく、使用する Azure Marketplace イメージの正しいバージョンは 4.8.x です。必要に応じて、インストールプロセスの一環として、VM が自動的にアップグレードされます。

- 次のいずれかのコマンドを実行して、オファターのイメージを調べます。

- 北米:

```
$ az vm image show --urn redhat:rh-ocp-worker:rh-ocp-worker:<version>
```

- EMEA:

```
$ az vm image show --urn redhat-limited:rh-ocp-worker:rh-ocp-worker:<version>
```

- 次のコマンドのいずれかを実行して、オファターの条件を確認します。

- 北米:

```
$ az vm image terms show --urn redhat:rh-ocp-worker:rh-ocp-worker:<version>
```

- EMEA:

```
$ az vm image terms show --urn redhat-limited:rh-ocp-worker:rh-ocp-worker:<version>
```

4. 次のコマンドのいずれかを実行して、オフリングの条件に同意します。

- 北米:

```
$ az vm image terms accept --urn redhat:rh-ocp-worker:rh-ocp-worker:<version>
```

- EMEA:

```
$ az vm image terms accept --urn redhat-limited:rh-ocp-worker:rh-ocp-worker:<version>
```

5. オファのイメージの詳細 (具体的には **publisher**、**offer**、**sku**、および **version** の値) を記録します。
6. オファのイメージの詳細を使用して、マシンセット YAML ファイルの **providerSpec** セクションに次のパラメーターを追加します。

#### Azure Marketplace コンピュータマシンのサンプルの **providerSpec** イメージ値

```
providerSpec:
  value:
    image:
      offer: rh-ocp-worker
      publisher: redhat
      resourceID: ""
      sku: rh-ocp-worker
      type: MarketplaceWithPlan
      version: 4.8.2021122100
```

### 2.3.4. マシンを **Spot** 仮想マシンとしてデプロイするマシンセット

マシンを保証されていない Spot 仮想マシンとしてデプロイする Azure で実行されるマシンセットを作成して、コストを節約できます。Spot 仮想マシンは未使用の Azure 容量を使用し、標準の仮想マシンよりもコストが低くなります。Spot 仮想マシンは、バッチやステートレス、水平的に拡張可能なワークロードなどの割り込みを許容できるワークロードに使用することができます。

Azure は Spot 仮想マシンをいつでも終了できます。Azure は、中断の発生時にユーザーに警告を 30 秒間表示します。OpenShift Container Platform は、Azure が終了についての警告を発行する際に影響を受けるインスタンスからワークロードを削除し始めます。

以下の理由により、Spot 仮想マシンを使用すると中断が生じる可能性があります。

- インスタンス価格は最大価格を超えます。
- Spot 仮想マシンの供給は減少します。
- Azure は容量を戻す必要があります。

Azure がインスタンスを終了すると、Spot 仮想マシンノードで実行される終了ハンドラーによりマシンリソースが削除されます。マシンセットの **replicas** の量を満たすために、マシンセットは Spot 仮想マシンを要求するマシンを作成します。

### 2.3.4.1. マシンセットの使用による Spot 仮想マシンの作成

**spotVMOptions** をマシンセットの YAML ファイルに追加して、Azure で Spot 仮想マシンを起動できます。

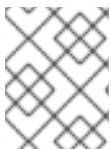
#### 手順

- **providerSpec** フィールドの下に以下の行を追加します。

```
providerSpec:
  value:
    spotVMOptions: {}
```

オプションで、Spot 仮想マシンのコストを制限するために、**spotVMOptions.maxPrice** フィールドを設定できます。たとえば、**maxPrice: '0.98765'** を設定できます。**maxPrice** が設定されている場合、この値は毎時の最大 Spot 価格として使用されます。設定されていない場合、最大価格はデフォルトの **-1** に設定され、標準の仮想マシン価格までチャージされます。

Azure は標準価格で Spot 仮想マシン価格を制限します。インスタンスがデフォルトの **maxPrice** で設定されている場合、Azure は価格設定によりインスタンスをエビクトしません。ただし、インスタンスは容量の制限によって依然としてエビクトできます。



#### 注記

デフォルトの仮想マシンの標準価格を **maxPrice** 値として使用し、Spot 仮想マシンの最大価格を設定しないことが強く推奨されます。

### 2.3.5. マシンを一時 OS ディスクにデプロイするマシンセット

マシンを Ephemeral OS ディスクにデプロイする Azure で実行されるマシンセットを作成できます。Azure Ephemeral OS ディスクは、リモートの Azure Storage ではなく、ローカルの VM 容量を使用します。したがって、この設定により、追加コストがなく、読み取り、書き込み、および再イメージ化のレイテンシーが短くなります。

#### 関連情報

- 詳細は、[Ephemeral OS disks for Azure VMs](#) についての Microsoft Azure ドキュメントを参照してください。

#### 2.3.5.1. マシンセットの使用による一時 OS ディスクでのマシンの作成

マシンセット YAML ファイルを編集して、Azure で Ephemeral OS ディスクでマシンを起動できます。

#### 前提条件

- 既存の Microsoft Azure クラスタがある。

#### 手順

1. 以下のコマンドを実行してカスタムリソース (CR) を編集します。

```
$ oc edit machineset <machine-set-name>
```

ここで、<machine-set-name> は、エフェメラル OS ディスクにマシンをプロビジョニングするマシンセットです。

2. 以下を **providerSpec** フィールドに追加します。

```
providerSpec:
  value:
    ...
    osDisk:
      ...
      diskSettings: ❶
        ephemeralStorageLocation: Local ❷
      cachingType: ReadOnly ❸
      managedDisk:
        storageAccountType: Standard_LRS ❹
      ...
```

❶ ❷ ❸ これらの行では、Ephemeral OS ディスクを使用できます。

❹ 一時 OS ディスクは、標準の LRS ストレージのアカウントタイプを使用する仮想マシンまたはスケールセットインスタンスでのみサポートされます。



### 重要

OpenShift Container Platform での Ephemeral OS ディスクのサポートの実装は、**CacheDisk** 配置タイプのみをサポートします。**placement** 設定は変更しないでください。

3. 更新された設定を使用してマシンセットを作成します。

```
$ oc create -f <machine-set-config>.yaml
```

### 検証

- Microsoft Azure ポータルで、マシンセットによってデプロイされたマシンの **Overview** ページを確認し、**Ephemeral OS** ディスク フィールドが **OS** キャッシュ配置に設定されていることを確認します。

## 2.3.6. Machine sets that deploy machines with ultra disks as data disks

Ultra ディスクと共にマシンをデプロイする Azure で実行されるマシンセットを作成できます。Ultra ディスクは、最も要求の厳しいデータワークロードでの使用を目的とした高性能ストレージです。

Azure ウルトラディスクに支えられたストレージクラスに動的にバインドし、それらを Pod にマウントする永続ボリューム要求 (PVC) を作成することもできます。



### 注記

データディスクは、ディスクスループットまたはディスク IOPS を指定する機能をサポートしていません。これらのプロパティは、PVC を使用して設定できます。

### 関連情報

- [Microsoft Azure Ultra ディスクのドキュメント](#)
- [CSI PVC を使用してウルトラディスクにマシンを展開するマシンセット](#)
- [in-tree\(インツリー\)PVC を使用して Ultra ディスクにマシンをデプロイするマシンセット](#)

### 2.3.6.1. マシンセットを使用した Ultra ディスクを持つマシンの作成

マシンセットの YAML ファイルを編集することで、Azure 上に Ultra ディスクと共にマシンをデプロイできます。

#### 前提条件

- 既存の Microsoft Azure クラスタがある。

#### 手順

1. 次のコマンドを実行して、ワーカーデータシークレットを使用して **openshift-machine-api** namespace にカスタムシークレットを作成します。

```
$ oc -n openshift-machine-api \
  get secret worker-user-data \
  --template='{{index .data.userData | base64decode}}' | jq > userData.txt
```

ここで、**userData.txt** は新しいカスタムシークレットの名前です。

2. テキストエディターで、**userData.txt** ファイルを開き、ファイル内の最後の **}** 文字を見つけます。
  - a. 直前の行に、**,** を追加します。
  - b. **,** の後に新しい行を作成し、以下の設定内容を追加します。

```
"storage": {
  "disks": [ 1
    {
      "device": "/dev/disk/azure/scsi1/lun0", 2
      "partitions": [ 3
        {
          "label": "lun0p1", 4
          "sizeMiB": 1024, 5
          "startMiB": 0
        }
      ]
    }
  ],
  "filesystems": [ 6
    {
      "device": "/dev/disk/by-partlabel/lun0p1",
      "format": "xfs",
      "path": "/var/lib/lun0p1"
    }
  ]
},
"systemd": {
```

```

"units": [ 7
{
  "contents": "[Unit]\nBefore=local-
fs.target\n[Mount]\nWhere=/var/lib/lun0p1\nWhat=/dev/disk/by-
partlabel/lun0p1\nOptions=defaults,pquota\n[Install]\nWantedBy=local-fs.target\n", 8
  "enabled": true,
  "name": "var-lib-lun0p1.mount"
}
]
}

```

- 1 ウルトラディスクとしてノードに接続するディスクの設定の詳細。
- 2 使用しているマシンセットの **dataDisks** スタンザで定義されている **lun** 値を指定します。たとえば、マシンセットに **lun:0** が含まれている場合は、**lun0** を指定します。この設定ファイルで複数の **"disks"** エントリーを指定することにより、複数のデータディスクを初期化できます。複数の **"disks"** エントリーを指定する場合は、それぞれの **lun** 値がマシンセットの値と一致することを確認してください。
- 3 ディスク上の新しいパーティションの設定の詳細。
- 4 パーティションのラベルを指定します。**lun0** の最初のパーティションに **lun0p1** などの階層名を使用すると便利な場合があります。
- 5 パーティションの合計サイズを MiB で指定します。
- 6 パーティションをフォーマットするときに使用するファイルシステムを指定します。パーティションラベルを使用して、パーティションを指定します。
- 7 起動時にパーティションをマウントする **systemd** ユニットを指定します。パーティションラベルを使用して、パーティションを指定します。この設定ファイルで複数の **"partitions"** エントリーを指定することにより、複数のパーティションを作成できます。複数の **"partitions"** エントリーを指定する場合は、それぞれに **systemd** ユニットを指定する必要があります。
- 8 **Where** には、**storage.filesystems.path** の値を指定します。**What** には、**storage.filesystems.device** の値を指定します。

3. 次のコマンドを実行して、無効化テンプレート値を **disableTemplating.txt** というファイルに抽出します。

```

$ oc -n openshift-machine-api get secret worker-user-data \
--template='{{index .data.disableTemplating | base64decode}}' | jq > disableTemplating.txt

```

4. 次のコマンドを実行して、**userData.txt** ファイルと **disableTemplating.txt** ファイルを組み合わせてデータシークレットファイルを作成します。

```

$ oc -n openshift-machine-api create secret generic worker-user-data-x5 \
--from-file=userData=userData.txt \
--from-file=disableTemplating=disableTemplating.txt

```

ここで、**worker-user-data-x5** はシークレットの名前です。

5. 既存の Azure **MachineSet** カスタムリソース (CR) をコピーし、次のコマンドを実行して編集します。

```
$ oc edit machineset <machine-set-name>
```

ここで、<**machine-set-name**> は、Ultra ディスクと共にマシンをプロビジョニングするマシンセットです。

6. 示された位置に次の行を追加します。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
...
spec:
...
template:
...
spec:
  metadata:
...
  labels:
...
    disk: ultrasdd ❶
...
  providerSpec:
    value:
      ...
      ultraSSDCapability: Enabled ❷
      dataDisks: ❸
      - nameSuffix: ultrasdd
        lun: 0
        diskSizeGB: 4
        deletionPolicy: Delete
        cachingType: None
        managedDisk:
          storageAccountType: UltraSSD_LRS
      userDataSecret:
        name: worker-user-data-x5 ❹
      ...
```

❶ このマシンセットによって作成されるノードを選択するために使用するラベルを指定します。この手順では、この値に **disk.ultrasdd** を使用します。

❷ ❸ これらのラインにより、ウルトラディスクの使用が可能になります。**dataDisk** の場合、スタンザ全体を含めます。

❹ 以前に作成したユーザーデータシークレットを指定します。

7. 次のコマンドを実行して、更新された設定を使用してマシンセットを作成します。

```
$ oc create -f <machine-set-name>.yaml
```

## 検証

1. 次のコマンドを実行して、マシンが作成されていることを確認します。

```
$ oc get machines
```

マシンは **Running** 状態になっているはずです。

2. 実行中でノードが接続されているマシンの場合、次のコマンドを実行してパーティションを検証します。

```
$ oc debug node/<node-name> -- chroot /host lsblk
```

このコマンドでは、**oc debug node/<node-name>** がノード **<node-name>** でデバッグシェルを開始し、**--** を付けてコマンドを渡します。渡されたコマンド **chroot /host** は、基盤となるホスト OS バイナリーへのアクセスを提供し、**lsblk** は、ホスト OS マシンに接続されているブロックデバイスを表示します。

## 次のステップ

- Pod 内から Ultra ディスクを使用するには、マウントポイントを使用するワークロードを作成します。次の例のような YAML ファイルを作成します。

```
apiVersion: v1
kind: Pod
metadata:
  name: ssd-benchmark1
spec:
  containers:
  - name: ssd-benchmark1
    image: nginx
    ports:
    - containerPort: 80
      name: "http-server"
    volumeMounts:
    - name: lun0p1
      mountPath: "/tmp"
  volumes:
  - name: lun0p1
    hostPath:
      path: /var/lib/lun0p1
      type: DirectoryOrCreate
  nodeSelector:
    disktype: ultrassd
```

### 2.3.6.2. Ultra ディスクを有効にするマシンセットのリソースに関するトラブルシューティング

このセクションの情報をを使用して、発生する可能性のある問題を理解し、回復してください。

#### 2.3.6.2.1. ウルトラディスク設定が正しくありません

マシンセットで **ultraSSDCapability** パラメーターの誤った設定が指定されている場合、マシンのプロビジョニングは失敗します。

たとえば、**ultraSSDCapability** パラメーターが **Disabled** に設定されているが、**dataDisks** パラメーターでウルトラディスクが指定されている場合、次のエラーメッセージが表示されます。

StorageAccountType UltraSSD\_LRS can be used only when additionalCapabilities.ultraSSDEnabled is set.

- この問題を解決するには、マシンセットの設定が正しいことを確認してください。

#### 2.3.6.2.2. サポートされていないディスクパラメーター

ウルトラディスクと互換性のないリージョン、アベイラビリティゾーン、またはインスタンスサイズがマシンセットで指定されている場合、マシンのプロビジョニングは失敗します。ログで次のエラーメッセージを確認してください。

```
failed to create vm <machine_name>: failure sending request for machine <machine_name>: cannot
create vm: compute.VirtualMachinesClient#CreateOrUpdate: Failure sending request:
StatusCode=400 -- Original Error: Code="BadRequest" Message="Storage Account type
'UltraSSD_LRS' is not supported <more_information_about_why>."
```

- この問題を解決するには、サポートされている環境でこの機能を使用していること、およびマシンセットの設定が正しいことを確認してください。

#### 2.3.6.2.3. ディスクを削除できません

データディスクとしてのウルトラディスクの削除が期待どおりに機能しない場合、マシンが削除され、データディスクが孤立します。必要に応じて、孤立したディスクを手動で削除する必要があります。

### 2.3.7. マシンセットの顧客管理の暗号鍵の有効化

Azure に暗号化キーを指定して、停止中に管理ディスクのデータを暗号化できます。マシン API を使用して、顧客管理の鍵でサーバー側の暗号化を有効にすることができます。

お客様が管理する鍵を使用するために、Azure Key Vault、ディスク暗号化セット、および暗号化キーが必要です。ディスク暗号化セットは、Cloud Credential Operator (CCO) にパーミッションが付与されたりソースグループに事前に存在する必要があります。これがない場合は、ディスク暗号化セットで追加のリーダーロールを指定する必要があります。

#### 前提条件

- [Azure Key Vault インスタンスを作成](#) します。
- [ディスク暗号化セットのインスタンスを作成](#) します。
- [ディスク暗号化セットに Key Vault へのアクセスを付与](#) します。

#### 手順

- マシンセット YAML ファイルの **providerSpec** フィールドでディスクの暗号化キーを設定します。以下に例を示します。

```
...
providerSpec:
  value:
    ...
    osDisk:
      diskSizeGB: 128
      managedDisk:
```

```

diskEncryptionSet:
  id:
    /subscriptions/<subscription_id>/resourceGroups/<resource_group_name>/providers/Microsoft.
    Compute/diskEncryptionSets/<disk_encryption_set_name>
  storageAccountType: Premium_LRS
...

```

## 関連情報

- [お客様が管理する鍵](#) についての詳細は、Azure ドキュメントを参照してください。

### 2.3.8. Microsoft Azure 仮想マシンのネットワークアクセラレート

アクセラレートネットワークは、Single Root I/O Virtualization (SR-IOV) を使用して、スイッチへのより直接的なパスを持つ Microsoft Azure 仮想マシンを提供します。これにより、ネットワークパフォーマンスが向上します。この機能は、インストール時またはインストール後に有効にできます。

#### 2.3.8.1. 制限事項

Accelerated Networking を使用するかどうかを決定する際には、以下の制限を考慮してください。

- ネットワークのアクセラレートは、マシン API が機能しているクラスターでのみサポートされます。
- Azure ワーカーノードの最小要件は 2 つの vCPU ですが、Accelerated Networking には 4 つ以上の vCPU を含む Azure 仮想マシンのサイズが必要です。この要件を満たすには、マシンセットの **vmSize** の値を変更します。Azure VM サイズの詳細は、[Microsoft Azure のドキュメント](#) を参照してください。
- この機能が既存の Azure クラスターで有効にされている場合、新たにプロビジョニングされたノードのみが影響を受けます。現在実行中のノードは調整されていません。全ノードで機能を有効にするには、それぞれの既存マシンを置き換える必要があります。これは、各マシンに対して個別に行うか、レプリカをゼロにスケールダウンしてから、必要なレプリカ数にスケールアップして実行できます。

## 関連情報

- [インストール中の高速ネットワークの有効化](#)

#### 2.3.8.2. 既存の Microsoft Azure クラスターでの Accelerated Networking の有効化

Azure で Accelerated Networking を有効にするには、**Networking** をマシンセットの YAML ファイルに追加します。

## 前提条件

- マシン API が機能している既存の Microsoft Azure クラスターがあること。

## 手順

1. 以下のコマンドを実行して、クラスター内のマシンセットを一覧表示します。

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

マシンセットは **<cluster-id>-worker-<region>** の形式で一覧表示されます。

## 出力例

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
jmywbfb-8zqpx-worker-centralus1	1	1	1	1	15m
jmywbfb-8zqpx-worker-centralus2	1	1	1	1	15m
jmywbfb-8zqpx-worker-centralus3	1	1	1	1	15m

2. それぞれのマシンセットについて以下を実行します。

a. 以下のコマンドを実行してカスタムリソース (CR) を編集します。

```
$ oc edit machineset <machine-set-name>
```

b. 以下を **providerSpec** フィールドに追加します。

```
providerSpec:
  value:
    ...
    acceleratedNetworking: true ❶
    ...
    vmSize: <azure-vm-size> ❷
    ...
```

❶ この行は Accelerated Networking を有効にします。

❷ 4 つ以上の vCPU を含む Azure 仮想マシンのサイズを指定します。仮想マシンのサイズに関する情報は、[Microsoft Azure のドキュメント](#) を参照してください。

3. 現在実行中のノードで機能を有効にするには、それぞれの既存マシンを置き換える必要があります。これは、各マシンに対して個別に行うか、レプリカをゼロにスケールダウンしてから、必要なレプリカ数にスケールアップして実行できます。

## 検証

- Microsoft Azure ポータルで、マシンセットによってプロビジョニングされるマシンの **Networking** 設定ページを確認し、**Accelerated networking** フィールドが **Enabled** に設定されていることを確認します。

## 関連情報

- [マシンセットの手動によるスケーリング](#)

## 2.4. AZURE STACK HUB でのマシンセットの作成

Microsoft Azure Stack Hub の OpenShift Container Platform クラスタで特定の目的を果たすように異なるマシンセットを作成することができます。たとえば、インフラストラクチャマシンセットおよび関連マシンを作成して、サポートするワークロードを新しいマシンに移動できます。

## 重要

高度なマシン管理およびスケーリング機能は、マシン API が機能しているクラスターでのみ使用することができます。ユーザーがプロビジョニングしたインフラストラクチャーを持つクラスターでは、マシン API を使用するために追加の検証と設定が必要です。

インフラストラクチャープラットフォームタイプが **none** のクラスターは、マシン API を使用できません。この制限は、クラスターに接続されている計算マシンが、この機能をサポートするプラットフォームにインストールされている場合でも適用されます。このパラメーターは、インストール後に変更することはできません。

クラスターのプラットフォームタイプを表示するには、以下のコマンドを実行します。

```
$ oc get infrastructure cluster -o jsonpath='{.status.platform}'
```

### 2.4.1. Azure Stack Hub 上のマシンセットのカスタムリソースのサンプル YAML

このサンプル YAML は、リージョンの **1** Microsoft Azure ゾーンで実行され、**node-role.kubernetes.io/<role>: ""** というラベルの付けられたノードを作成するマシンセットを定義します。

このサンプルでは、**<infrastructure\_id>** はクラスターのプロビジョニング時に設定したクラスター ID に基づくインフラストラクチャー ID であり、**<role>** は追加するノードラベルです。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
    machine.openshift.io/cluster-api-machine-role: <role> 2
    machine.openshift.io/cluster-api-machine-type: <role> 3
  name: <infrastructure_id>-<role>-<region> 4
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<region> 6
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 7
        machine.openshift.io/cluster-api-machine-role: <role> 8
        machine.openshift.io/cluster-api-machine-type: <role> 9
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<region> 10
    spec:
      metadata:
        creationTimestamp: null
      labels:
        node-role.kubernetes.io/<role>: "" 11
      providerSpec:
```

```

value:
  apiVersion: machine.openshift.io/v1beta1
  availabilitySet: <availability_set> 12
  credentialsSecret:
    name: azure-cloud-credentials
    namespace: openshift-machine-api
  image:
    offer: ""
    publisher: ""
    resourceID: /resourceGroups/<infrastructure_id>-
rg/providers/Microsoft.Compute/images/<infrastructure_id> 13
    sku: ""
    version: ""
  internalLoadBalancer: ""
  kind: AzureMachineProviderSpec
  location: <region> 14
  managedIdentity: <infrastructure_id>-identity 15
  metadata:
    creationTimestamp: null
  natRule: null
  networkResourceGroup: ""
  osDisk:
    diskSizeGB: 128
    managedDisk:
      storageAccountType: Premium_LRS
    osType: Linux
  publicIP: false
  publicLoadBalancer: ""
  resourceGroup: <infrastructure_id>-rg 16
  sshPrivateKey: ""
  sshPublicKey: ""
  subnet: <infrastructure_id>-<role>-subnet 17 18
  userDataSecret:
    name: worker-user-data 19
  vmSize: Standard_DS4_v2
  vnet: <infrastructure_id>-vnet 20
  zone: "1" 21

```

1 5 7 13 15 16 17 20 クラスターのプロビジョニング時に設定したクラスター ID を基にするインフラストラクチャー ID を指定します。OpenShift CLI がインストールされている場合は、以下のコマンドを実行してインフラストラクチャー ID を取得できます。

```
$ oc get -o jsonpath='{.status.infrastructureName}' infrastructure cluster
```

以下のコマンドを実行してサブネットを取得できます。

```
$ oc -n openshift-machine-api \
  -o jsonpath='{.spec.template.spec.providerSpec.value.subnet}' \
  get machineset/<infrastructure_id>-worker-centralus1
```

以下のコマンドを実行して vnet を取得できます。

```
$ oc -n openshift-machine-api \
  -o jsonpath='{.spec.template.spec.providerSpec.value.vnet}' \
  get machineset/<infrastructure_id>-worker-centralus1
```

**2 3 8 9 11 18 19** 追加するノードラベルを指定します。

**4 6 10** インフラストラクチャー ID、ノードラベル、およびリージョンを指定します。

**14** マシンを配置するリージョンを指定します。

**21** マシンを配置するリージョン内のゾーンを指定します。リージョンがゾーンをサポートすることを確認してください。

**12** クラスターの可用性セットを指定します。

## 2.4.2. マシンセットの作成

インストールプログラムによって作成されるコンピュートセットに加えて、独自のマシンセットを作成して、選択した特定のワークロードのマシンコンピューティングリソースを動的に管理できます。

### 前提条件

- OpenShift Container Platform クラスターをデプロイすること。
- OpenShift CLI (**oc**) がインストールされている。
- **cluster-admin** パーミッションを持つユーザーとして、**oc** にログインする。
- Azure Stack Hub マシンをデプロイする可用性セットを作成します。

### 手順

1. 説明されているようにマシンセット カスタムリソース (CR) サンプルを含む新規 YAML ファイルを作成し、そのファイルに **<file\_name>.yaml** という名前を付けます。  
**<availability Set>**、**<cluster ID>**、および **<role>** パラメーター値を必ず設定してください。
2. オプション: 特定のフィールドに設定する値がわからない場合は、クラスターから既存のコンピュートマシンセットを確認できます。
  - a. クラスター内のコンピュートマシンセットをリスト表示するには、次のコマンドを実行します。

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

### 出力例

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

- b. 特定のコンピュータマシンセットカスタムリソース(CR)の値を表示するには、以下のコマンドを実行します。

```
$ oc get machineset <machineset_name> \
-n openshift-machine-api -o yaml
```

### 出力例

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> ❶
  name: <infrastructure_id>-<role> ❷
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>
        machine.openshift.io/cluster-api-machine-role: <role>
        machine.openshift.io/cluster-api-machine-type: <role>
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
    spec:
      providerSpec: ❸
      ...
```

- ❶ クラスターインフラストラクチャー ID。
- ❷ デフォルトのノードラベル。



### 注記

ユーザーがプロビジョニングしたインフラストラクチャーを持つクラスターの場合、コンピュータマシンセットは **worker** および **infra** タイプのマシンのみを作成できます。

- ❸ コンピュータマシンセット CR の **<providerSpec>** セクションの値は、プラットフォーム固有です。CR の **<providerSpec>** パラメーターの詳細については、プロバイダーのサンプルコンピュータマシンセット CR 設定を参照してください。

3. 次のコマンドを実行して **MachineSet** CR を作成します。

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

### 検証

- 次のコマンドを実行して、コンピューマシンセットのリストを表示します。

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

### 出力例

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
agl030519-vplxk-infra-us-east-1a	1	1	1	1	11m
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

新規のマシンセットが利用可能な場合、**DESIRED** および **CURRENT** の値は一致します。マシンセットが利用可能でない場合、数分待機してからコマンドを再度実行します。

### 2.4.3. マシンセットの顧客管理の暗号鍵の有効化

Azure に暗号化キーを指定して、停止中に管理ディスクのデータを暗号化できます。マシン API を使用して、顧客管理の鍵でサーバー側の暗号化を有効にすることができます。

お客様が管理する鍵を使用するために、Azure Key Vault、ディスク暗号化セット、および暗号化キーが必要です。ディスク暗号化セットは、Cloud Credential Operator (CCO) にパーミッションが付与されたりソースグループに事前に存在する必要があります。これがない場合は、ディスク暗号化セットで追加のリーダーロールを指定する必要があります。

#### 前提条件

- [Azure Key Vault インスタンスを作成](#) します。
- [ディスク暗号化セットのインスタンスを作成](#) します。
- [ディスク暗号化セットに Key Vault へのアクセスを付与](#) します。

#### 手順

- マシンセット YAML ファイルの **providerSpec** フィールドでディスクの暗号化キーを設定します。以下に例を示します。

```
...
providerSpec:
  value:
    ...
    osDisk:
      diskSizeGB: 128
      managedDisk:
        diskEncryptionSet:
          id:
            /subscriptions/<subscription_id>/resourceGroups/<resource_group_name>/providers/Microsoft.
            Compute/diskEncryptionSets/<disk_encryption_set_name>
          storageAccountType: Premium_LRS
    ...
```

## 関連情報

- [お客様が管理する鍵](#) についての詳細は、Azure ドキュメントを参照してください。

## 2.5. GCP でのマシンセットの作成

異なるマシンセットを作成して、Google Cloud Platform (GCP) 上の OpenShift Container Platform クラスタで特定の目的で使用できます。たとえば、インフラストラクチャーマシンセットおよび関連マシンを作成して、サポートするワークロードを新しいマシンに移動できます。

### 重要

高度なマシン管理およびスケーリング機能は、マシン API が機能しているクラスタでのみ使用することができます。ユーザーがプロビジョニングしたインフラストラクチャーを持つクラスタでは、マシン API を使用するために追加の検証と設定が必要です。

インフラストラクチャープラットフォームタイプが **none** のクラスタは、マシン API を使用できません。この制限は、クラスタに接続されている計算マシンが、この機能をサポートするプラットフォームにインストールされている場合でも適用されます。このパラメーターは、インストール後に変更することはできません。

クラスタのプラットフォームタイプを表示するには、以下のコマンドを実行します。

```
$ oc get infrastructure cluster -o jsonpath='{.status.platform}'
```

### 2.5.1. GCP 上のマシンセットのカスタムリソースのサンプル YAML

このサンプル YAML は、Google Cloud Platform (GCP) で実行され、**node-role.kubernetes.io/<role>:** "" というラベルが付けられたノードを作成するマシンセットを定義します。

このサンプルでは、**<infrastructure\_id>** はクラスタのプロビジョニング時に設定したクラスタ ID に基づくインフラストラクチャー ID であり、**<role>** は追加するノードラベルです。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
  name: <infrastructure_id>-w-a
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-w-a
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>
        machine.openshift.io/cluster-api-machine-role: <role> 2
        machine.openshift.io/cluster-api-machine-type: <role>
```

```

machine.openshift.io/cluster-api-machineset: <infrastructure_id>-w-a
spec:
  metadata:
    labels:
      node-role.kubernetes.io/<role>: ""
  providerSpec:
    value:
      apiVersion: gcpprovider.openshift.io/v1beta1
      canIPForward: false
      credentialsSecret:
        name: gcp-cloud-credentials
      deletionProtection: false
      disks:
        - autoDelete: true
          boot: true
          image: <path_to_image> ❸
          labels: null
          sizeGb: 128
          type: pd-ssd
      gcpMetadata: ❹
        - key: <custom_metadata_key>
          value: <custom_metadata_value>
      kind: GCPMachineProviderSpec
      machineType: n1-standard-4
      metadata:
        creationTimestamp: null
      networkInterfaces:
        - network: <infrastructure_id>-network
          subnetwork: <infrastructure_id>-worker-subnet
      projectId: <project_name> ❺
      region: us-central1
      serviceAccounts:
        - email: <infrastructure_id>-w@<project_name>.iam.gserviceaccount.com
          scopes:
            - https://www.googleapis.com/auth/cloud-platform
      tags:
        - <infrastructure_id>-worker
      userDataSecret:
        name: worker-user-data
      zone: us-central1-a

```

- ❶ **<infrastructure\_id>** は、クラスターのプロビジョニング時に設定したクラスター ID に基づくインフラストラクチャー ID を指定します。OpenShift CLI がインストールされている場合は、以下のコマンドを実行してインフラストラクチャー ID を取得できます。

```
$ oc get -o jsonpath='{.status.infrastructureName}' infrastructure cluster
```

- ❷ **<node>** には、追加するノードラベルを指定します。

- ❸ 現在のマシンセットで使用するイメージへのパスを指定します。OpenShift CLI がインストールされている場合は、以下のコマンドを実行してイメージへのパスを取得できます。

```
$ oc -n openshift-machine-api \
  -o jsonpath='{.spec.template.spec.providerSpec.value.disks[0].image}' \
  get machineset/<infrastructure_id>-worker-a
```

GCP Marketplace イメージを使用するには、使用するオファーを指定します。

- OpenShift Container Platform:  
<https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-ocp-48-x86-64-202210040145>
- OpenShift Platform Plus: <https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-opp-48-x86-64-202206140145>
- OpenShift Kubernetes Engine:  
<https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-oke-48-x86-64-202206140145>

4 オプション: **key:value** のペアの形式でカスタムメタデータを指定します。ユースケースの例については、[カスタムメタデータの設定](#) について GCP のドキュメントを参照してください。

5 **<project\_name>** には、クラスターに使用する GCP プロジェクトの名前を指定します。

## 2.5.2. マシンセットの作成

インストールプログラムによって作成されるコンピュートセットセットに加えて、独自のマシンセットを作成して、選択した特定のワークロードのマシンコンピューティングリソースを動的に管理できます。

### 前提条件

- OpenShift Container Platform クラスターをデプロイすること。
- OpenShift CLI (**oc**) がインストールされている。
- **cluster-admin** パーミッションを持つユーザーとして、**oc** にログインする。

### 手順

1. 説明されているようにマシンセット カスタムリソース (CR) サンプルを含む新規 YAML ファイルを作成し、そのファイルに **<file\_name>.yaml** という名前を付けます。  
**<clusterID>** および **<role>** パラメーターの値を設定していることを確認します。
2. オプション: 特定のフィールドに設定する値がわからない場合は、クラスターから既存のコンピュートマシンセットを確認できます。
  - a. クラスター内のコンピュートマシンセットをリスト表示するには、次のコマンドを実行します。

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

### 出力例

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
agl030519-vplxx-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxx-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxx-worker-us-east-1c	1	1	1	1	55m

```

agl030519-vplxk-worker-us-east-1d 0    0    55m
agl030519-vplxk-worker-us-east-1e 0    0    55m
agl030519-vplxk-worker-us-east-1f 0    0    55m

```

- b. 特定のコンピューマシンセットカスタムリソース(CR)の値を表示するには、以下のコマンドを実行します。

```

$ oc get machineset <machineset_name> \
  -n openshift-machine-api -o yaml

```

## 出力例

```

apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
    name: <infrastructure_id>-<role> 2
    namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>
        machine.openshift.io/cluster-api-machine-role: <role>
        machine.openshift.io/cluster-api-machine-type: <role>
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
    spec:
      providerSpec: 3
      ...

```

**1** クラスターインフラストラクチャー ID。

**2** デフォルトのノードラベル。



### 注記

ユーザーがプロビジョニングしたインフラストラクチャーを持つクラスターの場合、コンピューマシンセットは **worker** および **infra** タイプのマシンのみを作成できます。

**3** コンピューマシンセット CR の **<providerSpec>** セクションの値は、プラットフォーム固有です。CR の **<providerSpec>** パラメーターの詳細については、プロバイダーのサンプルコンピューマシンセット CR 設定を参照してください。

3. 次のコマンドを実行して **MachineSet** CR を作成します。

```

$ oc create -f <file_name>.yaml

```

## 検証

- 次のコマンドを実行して、コンピュータマシンセットのリストを表示します。

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

## 出力例

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
agl030519-vplxk-infra-us-east-1a	1	1	1	1	11m
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

新規のマシンセットが利用可能な場合、**DESIRED** および **CURRENT** の値は一致します。マシンセットが利用可能でない場合、数分待機してからコマンドを再度実行します。

### 2.5.3. マシンセットを使用した永続ディスクタイプの設定

マシンセットの YAML ファイルを編集することで、マシンセットがマシンをデプロイする永続ディスクのタイプを設定できます。

永続ディスクの種類、互換性、地域の可用性、制限の詳細については、[永続ディスク](#) に関する GCPComputeEngine のドキュメントを参照してください。

## 手順

- テキストエディターで、既存のマシンセットの YAML ファイルを開くか、新しいマシンセットを作成します。
- providerSpec** フィールドの下で以下の行を編集します。

```
providerSpec:
  value:
    disks:
      type: <pd-disk-type> ❶
```

- ❶ ディスク永続タイプを指定します。有効な値は、**pd-ssd**、**pd-standard**、および **pd-balanced** です。デフォルト値は **pd-standard** です。

## 検証

- Google Cloud コンソールで、マシンセットによってデプロイされたマシンの詳細を確認し、**Type** フィールドが設定済みのディスクタイプと一致することを確認します。

### 2.5.4. マシンをプリエンプション可能な仮想マシンインスタンスとしてデプロイするマシンセット

マシンを保証されていないプリエンブション可能な仮想マシン インスタンスとしてデプロイする GCP で実行されるマシンセットを作成して、コストを節約できます。プリエンブション可能な仮想マシン インスタンスは、追加の Compute Engine 容量を使用し、通常のインスタンスよりもコストが低くなります。プリエンブション可能な仮想マシンインスタンスは、バッチやステートレス、水平的に拡張可能な ワークロードなどの割り込みを許容できるワークロードに使用することができます。

GCP Compute Engine は、プリエンブション可能な仮想マシンインスタンスをいつでも終了することができます。Compute Engine は、中断が 30 秒後に発生することを示すプリエンブションの通知をユーザーに送信します。OpenShift Container Platform は、Compute Engine がプリエンブションについての通知を発行する際に影響を受けるインスタンスからワークロードを削除し始めます。インスタンスが停止していない場合は、ACPI G3 Mechanical Off シグナルが 30 秒後にオペレーティングシステムに送信されます。プリエンブション可能な仮想マシンインスタンスは、Compute Engine によって **TERMINATED** 状態に移行されます。

以下の理由により、プリエンブション可能な仮想マシンインスタンスを使用すると中断が生じる可能性があります。

- システムまたはメンテナンスイベントがある
- プリエンブション可能な仮想マシンインスタンスの供給が減少する
- インスタンスは、プリエンブション可能な仮想マシンインスタンスについて割り当てられている 24 時間後に終了します。

GCP がインスタンスを終了すると、プリエンブション可能な仮想マシンインスタンスで実行される終了ハンドラーによりマシンリソースが削除されます。マシンセットの **replicas** の量を満たすために、マシンセットはプリエンブション可能な仮想マシンインスタンスを要求するマシンを作成します。

#### 2.5.4.1. マシンセットの使用によるプリエンブション可能な仮想マシンインスタンスの作成

**preemptible** をマシンセットの YAML ファイルに追加し、GCP でプリエンブション可能な仮想マシン インスタンスを起動できます。

##### 手順

- **providerSpec** フィールドの下に以下の行を追加します。

```
providerSpec:
  value:
    preemptible: true
```

**preemptible** が **true** に設定される場合、インスタンスの起動後に、マシンに **interruptable-instance** というラベルが付けられます。

#### 2.5.5. マシンセットの顧客管理の暗号鍵の有効化

Google Cloud Platform (GCP) Compute Engine を使用すると、ユーザーは暗号鍵を指定してディスク上の停止状態のデータを暗号化することができます。この鍵は、顧客のデータの暗号化に使用されず、データ暗号化キーの暗号化に使用されます。デフォルトでは、Compute Engine は Compute Engine キーを使用してこのデータを暗号化します。

マシン API を使用して、顧客管理の鍵で暗号化を有効にすることができます。まず [KMS キーを作成](#) し、適切なパーミッションをサービスアカウントに割り当てる必要があります。サービスアカウントが鍵を使用できるようにするには、KMS キー名、キーリング名、および場所が必要です。



## 注記

KMS の暗号化に専用のサービスアカウントを使用しない場合は、代わりに Compute Engine のデフォルトのサービスアカウントが使用されます。専用のサービスアカウントを使用しない場合、デフォルトのサービスアカウントに、キーにアクセスするためのパーミッションを付与する必要があります。Compute Engine のデフォルトのサービスアカウント名は、**service-`<project_number>`@compute-system.iam.gserviceaccount.com** パターンをベースにしています。

## 手順

1. KMS キー名、キーリング名、および場所を指定して以下のコマンドを実行し、特定のサービスアカウントが KMS キーを使用し、サービスアカウントに正しい IAM ロールを付与できるようにします。

```
gcloud kms keys add-iam-policy-binding <key_name> \
  --keyring <key_ring_name> \
  --location <key_ring_location> \
  --member "serviceAccount:service-<project_number>@compute-
system.iam.gserviceaccount.com" \
  --role roles/cloudkms.cryptoKeyEncrypterDecrypter
```

2. マシンセット YAML ファイルの **providerSpec** フィールドで暗号化キーを設定します。以下に例を示します。

```
providerSpec:
  value:
    # ...
    disks:
      - type:
        # ...
        encryptionKey:
          kmsKey:
            name: machine-encryption-key ❶
            keyRing: openshift-encrpytion-ring ❷
            location: global ❸
            projectID: openshift-gcp-project ❹
            kmsKeyServiceAccount: openshift-service-account@openshift-gcp-
project.iam.gserviceaccount.com ❺
```

- ❶ ディスク暗号化に使用される顧客管理の暗号鍵の名前。
- ❷ KMS キーが属する KMS キーリングの名前。
- ❸ KMS キーリングが存在する GCP の場所。
- ❹ オプション: KMS キーリングが存在するプロジェクトの ID。プロジェクト ID が設定されていない場合、マシンセットが作成されたマシンセットの **projectID** が使用されます。
- ❺ オプション: 指定の KMS キーの暗号化要求に使用されるサービスアカウント。サービスアカウントが設定されていない場合、Compute Engine のデフォルトのサービスアカウントが使用されます。

更新された **providerSpec** オブジェクト設定を使用して新規マシンが作成された後に、ディスクの暗号化キーは KMS キーを使用して暗号化されます。

## 2.5.6. マシンセットの GPU サポートを有効にする

Google Cloud Platform (GCP) Compute Engine を使用すると、ユーザーは仮想マシンインスタンスに GPU を追加できます。GPU リソースにアクセスできるワークロードは、この機能を有効にしてコンピュートマシンでより優れたパフォーマンスが得られます。GCP 上の Open Shift Container Platform は、A2 および N1 マシンシリーズの NVIDIA GPU モデルをサポートします。

表2.1 サポートされている GPU 設定

モデル名	GPU タイプ	マシンタイプ [1]
NVIDIA A100	<b>nvidia-tesla-a100</b>	<ul style="list-style-type: none"><li>● <b>a2-highgpu-1g</b></li><li>● <b>a2-highgpu-2g</b></li><li>● <b>a2-highgpu-4g</b></li><li>● <b>a2-highgpu-8g</b></li><li>● <b>a2-megagpu-16g</b></li></ul>

モデル名	GPU タイプ	マシンタイプ [1]
NVIDIA K80	nvidia-tesla-k80	<ul style="list-style-type: none"> <li>● n1-standard-1</li> <li>● n1-standard-2</li> <li>● n1-standard-4</li> <li>● n1-standard-8</li> <li>● n1-standard-16</li> </ul>
NVIDIA P100	nvidia-tesla-p100	
NVIDIA P4	nvidia-tesla-p4	
NVIDIA T4	nvidia-tesla-t4	
NVIDIA V100	nvidia-tesla-v100	
		<ul style="list-style-type: none"> <li>● n1-standard-32</li> <li>● n1-standard-64</li> <li>● n1-standard-96</li> <li>● n1-highmem-2</li> <li>● n1-highmem-4</li> <li>● n1-highmem-8</li> <li>● n1-highmem-16</li> <li>● n1-highmem-32</li> <li>● n1-highmem-64</li> <li>● n1-highmem-96</li> <li>● n1-highcpu-2</li> <li>● n1-highcpu-4</li> <li>● n1-highcpu-8</li> <li>● n1-highcpu-16</li> <li>● n1-highcpu-32</li> <li>● n1-highcpu-64</li> <li>● n1-highcpu-96</li> </ul>

1. 仕様、互換性、地域の可用性、制限など、マシンタイプの詳細については、[N1 マシンシリーズ](#)、[A2 マシンシリーズ](#)、[GPU リージョンとゾーンの可用性](#) に関する GCP Compute Engine のドキュメントをご覧ください。

Machine API を使用して、インスタンスに使用するサポートされている GPU を定義できます。

N1 マシンシリーズのマシンを、サポートされている GPU タイプの1つでデプロイするように設定できます。A2 マシンシリーズのマシンには GPU が関連付けられており、ゲストアクセラレータを使用することはできません。



## 注記

グラフィックワークロード用の GPU はサポートされていません。

## 手順

1. テキストエディターで、既存のマシンセットの YAML ファイルを開くか、新しいマシンセットを作成します。
2. マシンセットの YAML ファイルの **provider Spec** フィールドで GPU 設定を指定します。有効な設定の次の例を参照してください。

### A2 マシンシリーズの設定例:

```
providerSpec:
  value:
    machineType: a2-highgpu-1g ❶
    onHostMaintenance: Terminate ❷
    restartPolicy: Always ❸
```

- ❶ マシンタイプを指定します。マシンタイプが A2 マシンシリーズに含まれていることを確認してください。
- ❷ GPU サポートを使用する場合は、**onHostMaintenance** を **Terminate** に設定する必要があります。
- ❸ マシンセットによってデプロイされたマシンの再起動ポリシーを指定します。許可される値は、**Always** または **Never** です。

### N1 マシンシリーズの設定例:

```
providerSpec:
  value:
    gpus:
      - count: 1 ❶
        type: nvidia-tesla-p100 ❷
    machineType: n1-standard-1 ❸
    onHostMaintenance: Terminate ❹
    restartPolicy: Always ❺
```

- ❶ マシンに接続する GPU の数を指定します。
- ❷ マシンに接続する GPU のタイプを指定します。マシンタイプと GPU タイプに互換性があることを確認してください。
- ❸ マシンタイプを指定します。マシンタイプと GPU タイプに互換性があることを確認してください。
- ❹ GPU サポートを使用する場合は、**onHostMaintenance** を **Terminate** に設定する必要があります。
- ❺ マシンセットによってデプロイされたマシンの再起動ポリシーを指定します。許可される値は、**Always** または **Never** です。

## 2.6. IBMCLLOUD でのマシンセットの作成

IBMCloud 上の Open Shift Container Platform クラスターで特定の目的を果たすために別のマシンセットを作成できます。たとえば、インフラストラクチャーマシンセットおよび関連マシンを作成して、サポートするワークロードを新しいマシンに移動できます。

### 重要

高度なマシン管理およびスケーリング機能は、マシン API が機能しているクラスターでのみ使用することができます。ユーザーがプロビジョニングしたインフラストラクチャーを持つクラスターでは、マシン API を使用するために追加の検証と設定が必要です。

インフラストラクチャープラットフォームタイプが **none** のクラスターは、マシン API を使用できません。この制限は、クラスターに接続されている計算マシンが、この機能をサポートするプラットフォームにインストールされている場合でも適用されます。このパラメーターは、インストール後に変更することはできません。

クラスターのプラットフォームタイプを表示するには、以下のコマンドを実行します。

```
$ oc get infrastructure cluster -o jsonpath='{.status.platform}'
```

### 2.6.1. IBMCloud 上のマシンセットカスタムリソースのサンプル YAML

このサンプル YAML は、リージョン内の指定された IBM Cloud ゾーンで実行され、**node-role.kubernetes.io/<role>: ""** というラベルの付いたノードを作成するマシンセットを定義します。

このサンプルでは、**<infrastructure\_id>** はクラスターのプロビジョニング時に設定したクラスター ID に基づくインフラストラクチャー ID であり、**<role>** は追加するノードラベルです。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> ❶
    machine.openshift.io/cluster-api-machine-role: <role> ❷
    machine.openshift.io/cluster-api-machine-type: <role> ❸
  name: <infrastructure_id>-<role>-<region> ❹
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> ❺
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<region> ❻
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> ❼
        machine.openshift.io/cluster-api-machine-role: <role> ❽
        machine.openshift.io/cluster-api-machine-type: <role> ❾
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<region> ❿
    spec:
```

```

metadata:
  labels:
    node-role.kubernetes.io/<role>: ""
providerSpec:
  value:
    apiVersion: ibmcloudproviderconfig.openshift.io/v1beta1
    credentialsSecret:
      name: ibmcloud-credentials
    image: <infrastructure_id>-rhcos 11
    kind: IBMCloudMachineProviderSpec
    primaryNetworkInterface:
      securityGroups:
        - <infrastructure_id>-sg-cluster-wide
        - <infrastructure_id>-sg-openshift-net
      subnet: <infrastructure_id>-subnet-compute-<zone> 12
    profile: <instance_profile> 13
    region: <region> 14
    resourceGroup: <resource_group> 15
    userDataSecret:
      name: <role>-user-data 16
    vpc: <vpc_name> 17
    zone: <zone> 18

```

- 1 5 7** クラスターのプロビジョニング時に設定したクラスター ID に基づくインフラストラクチャー ID。OpenShift CLI がインストールされている場合は、以下のコマンドを実行してインフラストラクチャー ID を取得できます。

```
$ oc get -o jsonpath='{.status.infrastructureName}' infrastructure cluster
```

- 2 3 8 9 16** 追加するノードラベル。

- 4 6 10** インフラストラクチャー ID、ノードラベル、およびリージョン。

- 11** クラスターのインストールに使用されたカスタム Red Hat Enterprise Linux CoreOS (RHCOS) イメージ。

- 12** マシンを配置するためのリージョン内のインフラストラクチャー ID とゾーン。リージョンがゾーンをサポートすることを確認してください。

- 13** [IBM Cloud インスタンスプロファイル](#) を指定します。

- 14** マシンを配置するリージョンを指定します。

- 15** マシンリソースが配置されるリソースグループ。これは、インストール時に指定された既存のリソースグループ、またはインフラストラクチャー ID に基づいて名前が付けられたインストーラーによって作成されたリソースグループのいずれかです。

- 17** VPC 名。

- 18** マシンを配置するリージョン内のゾーンを指定します。リージョンがゾーンをサポートすることを確認してください。

## 2.6.2. マシンセットの作成

インストールプログラムによって作成されるコンピュータセットセットに加えて、独自のマシンセットを作成して、選択した特定のワークロードのマシンコンピューティングリソースを動的に管理できます。

## 前提条件

- OpenShift Container Platform クラスターをデプロイすること。
- OpenShift CLI (**oc**) がインストールされている。
- **cluster-admin** パーミッションを持つユーザーとして、**oc** にログインする。

## 手順

1. 説明されているようにマシンセット カスタムリソース (CR) サンプルを含む新規 YAML ファイルを作成し、そのファイルに **<file\_name>.yaml** という名前を付けます。  
**<clusterID>** および **<role>** パラメーターの値を設定していることを確認します。
2. オプション: 特定のフィールドに設定する値がわからない場合は、クラスターから既存のコンピュータマシンセットを確認できます。
  - a. クラスター内のコンピュータマシンセットをリスト表示するには、次のコマンドを実行します。

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

### 出力例

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

- b. 特定のコンピュータマシンセットカスタムリソース(CR)の値を表示するには、以下のコマンドを実行します。

```
$ oc get machineset <machineset_name> \
-n openshift-machine-api -o yaml
```

### 出力例

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> ❶
    name: <infrastructure_id>-<role> ❷
    namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
```

```

matchLabels:
  machine.openshift.io/cluster-api-cluster: <infrastructure_id>
  machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
template:
  metadata:
    labels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machine-role: <role>
      machine.openshift.io/cluster-api-machine-type: <role>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
  spec:
    providerSpec: ❸
    ...

```

❶ クラスターインフラストラクチャー ID。

❷ デフォルトのノードラベル。



### 注記

ユーザーがプロビジョニングしたインフラストラクチャーを持つクラスターの場合、コンピュートマシンセットは **worker** および **infra** タイプのマシンのみを作成できます。

❸ コンピュートマシンセット CR の **<providerSpec>** セクションの値は、プラットフォーム固有です。CR の **<providerSpec>** パラメーターの詳細については、プロバイダーのサンプルコンピュートマシンセット CR 設定を参照してください。

3. 次のコマンドを実行して **MachineSet** CR を作成します。

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

### 検証

- 次のコマンドを実行して、コンピュートマシンセットのリストを表示します。

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

### 出力例

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
agl030519-vplxk-infra-us-east-1a	1	1	1	1	11m
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

新規のマシンセットが利用可能な場合、**DESIRED** および **CURRENT** の値は一致します。マシンセットが利用可能でない場合、数分待機してからコマンドを再度実行します。

## 2.7. NUTANIX でマシンセットを作成する

Nutanix 上の Open Shift Container Platform クラスターで特定の目的を果たすために別のマシンセットを作成できます。たとえば、インフラストラクチャーマシンセットおよび関連マシンを作成して、サポートするワークロードを新しいマシンに移動できます。

### 重要

高度なマシン管理およびスケーリング機能は、マシン API が機能しているクラスターでのみ使用することができます。ユーザーがプロビジョニングしたインフラストラクチャーを持つクラスターでは、マシン API を使用するために追加の検証と設定が必要です。

インフラストラクチャープラットフォームタイプが **none** のクラスターは、マシン API を使用できません。この制限は、クラスターに接続されている計算マシンが、この機能をサポートするプラットフォームにインストールされている場合でも適用されます。このパラメーターは、インストール後に変更することはできません。

クラスターのプラットフォームタイプを表示するには、以下のコマンドを実行します。

```
$ oc get infrastructure cluster -o jsonpath='{.status.platform}'
```

### 2.7.1. Nutanix 上のマシンセットのカスタムリソースのサンプル YAML

このサンプル YAML は、**node-role.kubernetes.io/<role>: ""** でラベル付けされたノードを作成する Nutanix マシンセットを定義します。

このサンプルでは、**<infrastructure\_id>** はクラスターのプロビジョニング時に設定したクラスター ID に基づくインフラストラクチャー ID であり、**<role>** は追加するノードラベルです。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
    machine.openshift.io/cluster-api-machine-role: <role> 2
    machine.openshift.io/cluster-api-machine-type: <role> 3
  name: <infrastructure_id>-<role>-<zone> 4
  namespace: openshift-machine-api
  annotations: 5
    machine.openshift.io/memoryMb: "16384"
    machine.openshift.io/vCPU: "4"
spec:
  replicas: 3
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 6
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<zone> 7
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 8
        machine.openshift.io/cluster-api-machine-role: <role> 9
```

```

machine.openshift.io/cluster-api-machine-type: <role> 10
machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<zone> 11
spec:
  metadata:
    labels:
      node-role.kubernetes.io/<role>: ""
  providerSpec:
    value:
      apiVersion: machine.openshift.io/v1
      cluster:
        type: uuid
        uuid: <cluster_uuid>
      credentialsSecret:
        name: nutanix-creds-secret
      image:
        name: <infrastructure_id>-rhcos 12
        type: name
      kind: NutanixMachineProviderConfig
      memorySize: 16Gi 13
      subnets:
        - type: uuid
          uuid: <subnet_uuid>
      systemDiskSize: 120Gi 14
      userDataSecret:
        name: <user_data_secret> 15
      vcpuSockets: 4 16
      vcpusPerSocket: 1 17

```

- 1 6 8 クラスターのプロビジョニング時に設定したクラスター ID を基にするインフラストラクチャー ID を指定します。OpenShift CLI (**oc**) がインストールされている場合は、以下のコマンドを実行してインフラストラクチャー ID を取得できます。

```
$ oc get -o jsonpath='{.status.infrastructureName}{"\n"}' infrastructure cluster
```

- 2 3 9 10 追加するノードラベルを指定します。
- 4 7 11 インフラストラクチャー ID、ノードラベル、およびゾーンを指定します。
- 5 クラスターオートスケーラーのアノテーション。
- 12 使用するイメージを指定します。クラスターに設定されている既存のデフォルトマシンのイメージを使用します。
- 13 クラスターのメモリー量を Gi で指定します。
- 14 システムディスクのサイズを Gi で指定します。
- 15 **openshift-machine-api** 名前空間にあるユーザーデータ YAML ファイルでシークレットの名前を指定します。インストーラーがデフォルトのマシンセットに入力する値を使用します。
- 16 vCPU ソケットの数を指定します。
- 17 ソケットあたりの vCPU の数を指定します。

## 2.7.2. マシンセットの作成

インストールプログラムによって作成されるコンピュータセットセットに加えて、独自のマシンセットを作成して、選択した特定のワークロードのマシンコンピューティングリソースを動的に管理できます。

### 前提条件

- OpenShift Container Platform クラスターをデプロイすること。
- OpenShift CLI (**oc**) がインストールされている。
- **cluster-admin** パーミッションを持つユーザーとして、**oc** にログインする。

### 手順

1. 説明されているようにマシンセット カスタムリソース (CR) サンプルを含む新規 YAML ファイルを作成し、そのファイルに **<file\_name>.yaml** という名前を付けます。  
**<clusterID>** および **<role>** パラメーターの値を設定していることを確認します。
2. オプション: 特定のフィールドに設定する値がわからない場合は、クラスターから既存のコンピュータマシンセットを確認できます。
  - a. クラスター内のコンピュータマシンセットをリスト表示するには、次のコマンドを実行します。

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

### 出力例

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

- b. 特定のコンピュータマシンセットカスタムリソース(CR)の値を表示するには、以下のコマンドを実行します。

```
$ oc get machineset <machineset_name> \
-n openshift-machine-api -o yaml
```

### 出力例

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> ❶
    name: <infrastructure_id>-<role> ❷
    namespace: openshift-machine-api
spec:
```

```

replicas: 1
selector:
  matchLabels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id>
    machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
template:
  metadata:
    labels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machine-role: <role>
      machine.openshift.io/cluster-api-machine-type: <role>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
  spec:
    providerSpec: ❸
    ...

```

❶ クラスターインフラストラクチャー ID。

❷ デフォルトのノードラベル。



### 注記

ユーザーがプロビジョニングしたインフラストラクチャーを持つクラスターの場合、コンピュートマシンセットは **worker** および **infra** タイプのマシンのみを作成できます。

❸ コンピュートマシンセット CR の **<providerSpec>** セクションの値は、プラットフォーム固有です。CR の **<providerSpec>** パラメーターの詳細については、プロバイダーのサンプルコンピュートマシンセット CR 設定を参照してください。

3. 次のコマンドを実行して **MachineSet** CR を作成します。

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

### 検証

- 次のコマンドを実行して、コンピュートマシンセットのリストを表示します。

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

### 出力例

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
agl030519-vplxk-infra-us-east-1a	1	1	1	1	11m
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

新規のマシンセットが利用可能な場合、**DESIRED** および **CURRENT** の値は一致します。マシンセットが利用可能でない場合、数分待機してからコマンドを再度実行します。

## 2.8. OPENSTACK でのマシンセットの作成

異なるマシンセットを作成して、Red Hat OpenStack Platform (RHOSP) 上の OpenShift Container Platform クラスターで特定の目的で使用できます。たとえば、インフラストラクチャーマシンセットおよび関連マシンを作成して、サポートするワークロードを新しいマシンに移動できます。

### 重要

高度なマシン管理およびスケーリング機能は、マシン API が機能しているクラスターでのみ使用することができます。ユーザーがプロビジョニングしたインフラストラクチャーを持つクラスターでは、マシン API を使用するために追加の検証と設定が必要です。

インフラストラクチャープラットフォームタイプが **none** のクラスターは、マシン API を使用できません。この制限は、クラスターに接続されている計算マシンが、この機能をサポートするプラットフォームにインストールされている場合でも適用されます。このパラメーターは、インストール後に変更することはできません。

クラスターのプラットフォームタイプを表示するには、以下のコマンドを実行します。

```
$ oc get infrastructure cluster -o jsonpath='{.status.platform}'
```

### 2.8.1. RHOSP 上のマシンセットのカスタムリソースのサンプル YAML

このサンプル YAML は、Red Hat OpenStack Platform (RHOSP) で実行され、**node-role.kubernetes.io/<role>: ""** というラベルが付けられたノードを作成するマシンセットを定義します。

このサンプルでは、**<infrastructure\_id>** はクラスターのプロビジョニング時に設定したクラスター ID に基づくインフラストラクチャー ID であり、**<role>** は追加するノードラベルです。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
    machine.openshift.io/cluster-api-machine-role: <role> 2
    machine.openshift.io/cluster-api-machine-type: <role> 3
  name: <infrastructure_id>-<role> 4
  namespace: openshift-machine-api
spec:
  replicas: <number_of_replicas>
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role> 6
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 7
```

```

machine.openshift.io/cluster-api-machine-role: <role> 8
machine.openshift.io/cluster-api-machine-type: <role> 9
machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role> 10
spec:
  providerSpec:
    value:
      apiVersion: openstackproviderconfig.openshift.io/v1alpha1
      cloudName: openstack
      cloudsSecret:
        name: openstack-cloud-credentials
        namespace: openshift-machine-api
      flavor: <nova_flavor>
      image: <glance_image_name_or_location>
      serverGroupID: <optional_UUID_of_server_group> 11
      kind: OpenstackProviderSpec
      networks: 12
      - filter: {}
        subnets:
          - filter:
              name: <subnet_name>
              tags: openshiftClusterID=<infrastructure_id> 13
        primarySubnet: <rhosp_subnet_UUID> 14
      securityGroups:
        - filter: {}
          name: <infrastructure_id>-worker 15
      serverMetadata:
        Name: <infrastructure_id>-worker 16
        openshiftClusterID: <infrastructure_id> 17
      tags:
        - openshiftClusterID=<infrastructure_id> 18
      trunk: true
      userDataSecret:
        name: worker-user-data 19
      availabilityZone: <optional_openstack_availability_zone>

```

1 5 7 13 15 16 17 18 クラスターのプロビジョニング時に設定したクラスター ID を基にするインフラストラクチャー ID を指定します。OpenShift CLI がインストールされている場合は、以下のコマンドを実行してインフラストラクチャー ID を取得できます。

```
$ oc get -o jsonpath='{.status.infrastructureName}' infrastructure cluster
```

2 3 8 9 19 追加するノードラベルを指定します。

4 6 10 インフラストラクチャー ID およびノードラベルを指定します。

11 MachineSet のサーバーグループポリシーを設定するには、[サーバーグループの作成](#) から返された値を入力します。ほとんどのデプロイメントでは、**anti-affinity** または **soft-anti-affinity** が推奨されます。

12 複数ネットワークへのデプロイメントに必要です。複数のネットワークを指定するには、ネットワークアレイに別のエントリーを追加します。また、**primarySubnet** の値として使用されるネットワークが含まれる必要があります。

14

ノードのエンドポイントを公開する RHOSP サブネットを指定します。通常、これは **install-config.yaml** ファイルの **machinesSubnet** の値として使用される同じサブネットです。

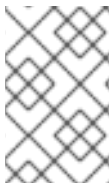
## 2.8.2. RHOSP 上の SR-IOV を使用するマシンセットのカスタムリソースのサンプル YAML

クラスターを SR-IOV (Single-root I/O Virtualization) 用に設定している場合に、その技術を使用するマシンセットを作成できます。

このサンプル YAML は SR-IOV ネットワークを使用するマシンセットを定義します。作成するノードには **node-role.openshift.io/<node\_role>: ""** というラベルが付けられます。

このサンプルでは、**infrastructure\_id** はクラスターのプロビジョニング時に設定したクラスター ID に基づくインフラストラクチャー ID ラベルであり、**node\_role** は追加するノードラベルです。

この例では、radio と uplink という名前の 2 つの SR-IOV ネットワークを想定しています。これらのネットワークは、**spec.template.spec.providerSpec.value.ports** 一覧のポート定義で使用されます。



### 注記

この例では、SR-IOV デプロイメント固有のパラメーターのみを説明します。より一般的なサンプルを確認するには、RHOSP 上のマシンセットのカスタムリソースのサンプル YAML を参照してください。

### SR-IOV ネットワークを使用するマシンセットの例

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id>
    machine.openshift.io/cluster-api-machine-role: <node_role>
    machine.openshift.io/cluster-api-machine-type: <node_role>
  name: <infrastructure_id>-<node_role>
  namespace: openshift-machine-api
spec:
  replicas: <number_of_replicas>
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<node_role>
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>
        machine.openshift.io/cluster-api-machine-role: <node_role>
        machine.openshift.io/cluster-api-machine-type: <node_role>
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<node_role>
    spec:
      metadata:
      providerSpec:
        value:
          apiVersion: openstackproviderconfig.openshift.io/v1alpha1
          cloudName: openstack
```

```

cloudsSecret:
  name: openstack-cloud-credentials
  namespace: openshift-machine-api
flavor: <nova_flavor>
image: <glance_image_name_or_location>
serverGroupID: <optional_UUID_of_server_group>
kind: OpenstackProviderSpec
networks:
  - subnets:
    - UUID: <machines_subnet_UUID>
ports:
  - networkID: <radio_network_UUID> ❶
    nameSuffix: radio
    fixedIPs:
      - subnetID: <radio_subnet_UUID> ❷
    tags:
      - sriov
      - radio
    vnicType: direct ❸
    portSecurity: false ❹
  - networkID: <uplink_network_UUID> ❺
    nameSuffix: uplink
    fixedIPs:
      - subnetID: <uplink_subnet_UUID> ❻
    tags:
      - sriov
      - uplink
    vnicType: direct ❼
    portSecurity: false ❽
primarySubnet: <machines_subnet_UUID>
securityGroups:
  - filter: {}
    name: <infrastructure_id>-<node_role>
serverMetadata:
  Name: <infrastructure_id>-<node_role>
  openshiftClusterID: <infrastructure_id>
tags:
  - openshiftClusterID=<infrastructure_id>
trunk: true
userDataSecret:
  name: <node_role>-user-data
availabilityZone: <optional_openstack_availability_zone>

```

❶ ❺ 各ポートにネットワークの UUID を入力します。

❷ ❻ 各ポートのサブネット UUID を入力します。

❸ ❼ **vnicType** パラメーターの値は、各ポートに **直接** 指定する必要があります。

❹ ❽ **portSecurity** パラメーターの値は、各ポートで **false** である必要があります。

ポートセキュリティが無効な場合は、ポートにセキュリティグループと使用可能なアドレスペアを設定できません。インスタンスにセキュリティグループを設定すると、グループが割り当てられているすべてのポートに適用されます。



## 重要

SR-IOV 対応のコンピュータマシンをデプロイしたら、そのようにラベルを付ける必要があります。たとえば、コマンドラインから次のように入力します。

```
$ oc label node <NODE_NAME> feature.node.kubernetes.io/network-sriov.capable="true"
```



## 注記

トランクは、ネットワークおよびサブネットの一覧のエントリで作成されるポート向けに有効にされます。これらの一覧から作成されたポートの名前は、**<machine\_name>-<nameSuffix>** パターンを使用します。**nameSuffix** フィールドは、ポート定義に必要です。

それぞれのポートにトランキングを有効にすることができます。

オプションで、タグを **タグ** 一覧の一部としてポートに追加できます。

## 関連情報

- [Preparing to install a cluster that uses SR-IOV or OVS-DPDK on OpenStack](#)

### 2.8.3. ポートセキュリティが無効にされている SR-IOV デプロイメントのサンプル YAML

ポートセキュリティが無効にされたネットワークに single-root I/O Virtualization (SR-IOV) ポートを作成するには、**spec.template.spec.providerSpec.value.ports** 一覧の項目としてポートを含めてマシンセットを定義します。標準の SR-IOV マシンセットとのこの相違点は、ネットワークとサブネットインターフェイスを使用して作成されたポートに対して発生する自動セキュリティグループと使用可能なアドレスペア設定によるものです。

マシンのサブネット用に定義するポートには、以下が必要です。

- API および Ingress 仮想 IP ポート用に許可されるアドレスペア
- コンピュータセキュリティグループ
- マシンネットワークおよびサブネットへの割り当て



## 注記

以下の例のように、ポートセキュリティが無効になっている SR-IOV デプロイメント固有のパラメーターのみを説明します。より一般的なサンプルを確認するには、RHOSP 上の SR-IOV を使用するマシンセットカスタムリソースのサンプル YAML について参照してください。

### SR-IOV ネットワークを使用し、ポートセキュリティが無効にされているマシンセットの例

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id>
```

```

machine.openshift.io/cluster-api-machine-role: <node_role>
machine.openshift.io/cluster-api-machine-type: <node_role>
name: <infrastructure_id>-<node_role>
namespace: openshift-machine-api
spec:
  replicas: <number_of_replicas>
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<node_role>
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>
        machine.openshift.io/cluster-api-machine-role: <node_role>
        machine.openshift.io/cluster-api-machine-type: <node_role>
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<node_role>
    spec:
      metadata: {}
      providerSpec:
        value:
          apiVersion: openstackproviderconfig.openshift.io/v1alpha1
          cloudName: openstack
          cloudsSecret:
            name: openstack-cloud-credentials
            namespace: openshift-machine-api
          flavor: <nova_flavor>
          image: <glance_image_name_or_location>
          kind: OpenstackProviderSpec
          ports:
            - allowedAddressPairs: ❶
              - ipAddress: <API_VIP_port_IP>
              - ipAddress: <ingress_VIP_port_IP>
              fixedIPs:
                - subnetID: <machines_subnet_UUID> ❷
              nameSuffix: nodes
              networkID: <machines_network_UUID> ❸
              securityGroups:
                - <compute_security_group_UUID> ❹
            - networkID: <SRIOV_network_UUID>
              nameSuffix: sriov
              fixedIPs:
                - subnetID: <SRIOV_subnet_UUID>
              tags:
                - sriov
              vnicType: direct
              portSecurity: false
          primarySubnet: <machines_subnet_UUID>
          serverMetadata:
            Name: <infrastructure_ID>-<node_role>
            openshiftClusterID: <infrastructure_id>
          tags:
            - openshiftClusterID=<infrastructure_id>
          trunk: false
          userDataSecret:
            name: worker-user-data

```

- 
- 1 API および Ingress ポート用に許可されるアドレスペアを指定します。
- 2 3 マシンネットワークおよびサブネットを指定します。
- 4 コンピュータマシンのセキュリティーグループを指定します。

### 注記

トランクは、ネットワークおよびサブネットの一覧のエントリで作成されるポート向けに有効にされます。これらの一覧から作成されたポートの名前は、**<machine\_name>-<nameSuffix>** パターンを使用します。**nameSuffix** フィールドは、ポート定義に必要です。

それぞれのポートにトランキングを有効にすることができます。

オプションで、タグを **タグ** 一覧の一部としてポートに追加できます。

クラスターで Kuryr を使用し、RHOSP SR-IOV ネットワークでポートセキュリティーが無効にされている場合に、コンピュータマシンのプライマリーポートには以下が必要になります。

- **spec.template.spec.providerSpec.value.networks.portSecurityEnabled** パラメーターの値を **false** に設定します。
- 各サブネットについて、**spec.template.spec.providerSpec.value.networks.subnets.portSecurityEnabled** パラメーターの値を **false** に設定します。
- **spec.template.spec.providerSpec.value.securityGroups** の値は、空: [] に指定します。

SR-IOV を使用し、ポートセキュリティーが無効な Kuryr にあるクラスターのマシンセットのセクション例

```
...
  networks:
    - subnets:
      - uuid: <machines_subnet_UUID>
        portSecurityEnabled: false
        portSecurityEnabled: false
      securityGroups: []
...

```

今回の場合は、仮想マシンの作成後にコンピュータセキュリティーグループをプライマリー仮想マシンインターフェイスに適用できます。たとえば、コマンドラインでは、以下を実行します。

```
$ openstack port set --enable-port-security --security-group <infrastructure_id>-<node_role>
<main_port_ID>
```



## 重要

SR-IOV 対応のコンピュータマシンをデプロイしたら、そのようにラベルを付ける必要があります。たとえば、コマンドラインから次のように入力します。

```
$ oc label node <NODE_NAME> feature.node.kubernetes.io/network-sriov.capable="true"
```

## 2.8.4. マシンセットの作成

インストールプログラムによって作成されるコンピュータセットセットに加えて、独自のマシンセットを作成して、選択した特定のワークロードのマシンコンピューティングリソースを動的に管理できます。

### 前提条件

- OpenShift Container Platform クラスターをデプロイすること。
- OpenShift CLI (**oc**) がインストールされている。
- **cluster-admin** パーミッションを持つユーザーとして、**oc** にログインする。

### 手順

1. 説明されているようにマシンセット カスタムリソース (CR) サンプルを含む新規 YAML ファイルを作成し、そのファイルに **<file\_name>.yaml** という名前を付けます。  
**<clusterID>** および **<role>** パラメーターの値を設定していることを確認します。
2. オプション: 特定のフィールドに設定する値がわからない場合は、クラスターから既存のコンピュータマシンセットを確認できます。
  - a. クラスター内のコンピュータマシンセットをリスト表示するには、次のコマンドを実行します。

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

### 出力例

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

- b. 特定のコンピュータマシンセットカスタムリソース(CR)の値を表示するには、以下のコマンドを実行します。

```
$ oc get machineset <machineset_name> \
-n openshift-machine-api -o yaml
```

### 出力例

■

```

apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> ❶
  name: <infrastructure_id>-<role> ❷
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>
        machine.openshift.io/cluster-api-machine-role: <role>
        machine.openshift.io/cluster-api-machine-type: <role>
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
    spec:
      providerSpec: ❸
      ...

```

❶ クラスターインフラストラクチャー ID。

❷ デフォルトのノードラベル。



#### 注記

ユーザーがプロビジョニングしたインフラストラクチャーを持つクラスターの場合、コンピュータマシンセットは **worker** および **infra** タイプのマシンのみを作成できます。

❸ コンピュータマシンセット CR の **<providerSpec>** セクションの値は、プラットフォーム固有です。CR の **<providerSpec>** パラメーターの詳細については、プロバイダーのサンプルコンピュータマシンセット CR 設定を参照してください。

3. 次のコマンドを実行して **MachineSet** CR を作成します。

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

#### 検証

- 次のコマンドを実行して、コンピュータマシンセットのリストを表示します。

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

#### 出力例

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
agl030519-vplxk-infra-us-east-1a	1	1	1	1	11m

agl030519-vplxx-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxx-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxx-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxx-worker-us-east-1d	0	0			55m
agl030519-vplxx-worker-us-east-1e	0	0			55m
agl030519-vplxx-worker-us-east-1f	0	0			55m

新規のマシンセットが利用可能な場合、**DESIRED** および **CURRENT** の値は一致します。マシンセットが利用可能でない場合、数分待機してからコマンドを再度実行します。

## 2.9. RHV でのマシンセットの作成

異なるマシンセットを作成して、Red Hat Virtualization (RHV) 上の OpenShift Container Platform クラスタで特定の目的で使用できます。たとえば、インフラストラクチャーマシンセットおよび関連マシンを作成して、サポートするワークロードを新しいマシンに移動できます。

### 重要

高度なマシン管理およびスケーリング機能は、マシン API が機能しているクラスターでのみ使用することができます。ユーザーがプロビジョニングしたインフラストラクチャーを持つクラスターでは、マシン API を使用するために追加の検証と設定が必要です。

インフラストラクチャープラットフォームタイプが **none** のクラスターは、マシン API を使用できません。この制限は、クラスターに接続されている計算マシンが、この機能をサポートするプラットフォームにインストールされている場合でも適用されます。このパラメーターは、インストール後に変更することはできません。

クラスターのプラットフォームタイプを表示するには、以下のコマンドを実行します。

```
$ oc get infrastructure cluster -o jsonpath='{.status.platform}'
```

### 2.9.1. RHV 上のマシンセットのカスタムリソースのサンプル YAML

このサンプル YAML は、RHV で実行され、**node-role.kubernetes.io/<node\_role>: ""** というラベルが付けられたノードを作成するマシンセットを定義します。

このサンプルでは、**<infrastructure\_id>** はクラスターのプロビジョニング時に設定したクラスター ID に基づくインフラストラクチャー ID であり、**<role>** は追加するノードラベルです。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> ❶
    machine.openshift.io/cluster-api-machine-role: <role> ❷
    machine.openshift.io/cluster-api-machine-type: <role> ❸
  name: <infrastructure_id>-<role> ❹
  namespace: openshift-machine-api
spec:
  replicas: <number_of_replicas> ❺
  Selector: ❻
    matchLabels:
```

```

machine.openshift.io/cluster-api-cluster: <infrastructure_id> 7
machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role> 8
template:
  metadata:
    labels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 9
      machine.openshift.io/cluster-api-machine-role: <role> 10
      machine.openshift.io/cluster-api-machine-type: <role> 11
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role> 12
  spec:
    metadata:
      labels:
        node-role.kubernetes.io/<role>: "" 13
    providerSpec:
      value:
        apiVersion: ovirtproviderconfig.machine.openshift.io/v1beta1
        cluster_id: <ovirt_cluster_id> 14
        template_name: <ovirt_template_name> 15
        sparse: <boolean_value> 16
        format: <raw_or_cow> 17
        cpu: 18
          sockets: <number_of_sockets> 19
          cores: <number_of_cores> 20
          threads: <number_of_threads> 21
        memory_mb: <memory_size> 22
        guaranteed_memory_mb: <memory_size> 23
        os_disk: 24
          size_gb: <disk_size> 25
          storage_domain_id: <storage_domain_UUID> 26
        network_interfaces: 27
          vnic_profile_id: <vnic_profile_id> 28
        credentialsSecret:
          name: ovirt-credentials 29
        kind: OvirtMachineProviderSpec
        type: <workload_type> 30
        auto_pinning_policy: <auto_pinning_policy> 31
        hugepages: <hugepages> 32
        affinityGroupsNames:
          - compute 33
        userDataSecret:
          name: worker-user-data

```

1 7 9 クラスターのプロビジョニング時に設定したクラスター ID を基にするインフラストラクチャー ID を指定します。OpenShift CLI (**oc**) がインストールされている場合は、以下のコマンドを実行してインフラストラクチャー ID を取得できます。

```
$ oc get -o jsonpath='{.status.infrastructureName}' infrastructure cluster
```

2 3 10 11 13 追加するノードラベルを指定します。

4 8 12 インフラストラクチャー ID およびノードラベルを指定します。これら 2 つの文字列は 35 文字を超えることができません。

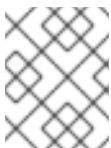
- 5 作成するマシンの数を指定します。
- 6 マシンのセクター。
- 14 この仮想マシンインスタンスが属する RHV クラスターの UUID を指定します。
- 15 マシンの作成に使用する RHV 仮想マシンテンプレートを指定します。
- 16 このオプションを **false** に設定すると、ディスクの事前割り当てが有効になります。デフォルトは **true** です。**format** を **raw** に設定して **sparse** を **true** に設定することは、ブロックストレージドメインでは使用できません。**raw** 形式は、仮想ディスク全体を基盤となる物理ディスクに書き込みます。
- 17 **cow** または **raw** に設定できます。デフォルトは **cow** です。**cow** のフォーマットは仮想マシン用に最適化されています。



#### 注記

ファイルストレージドメインにディスクを事前に割り当てると、ファイルにゼロが書き込まれます。基盤となるストレージによっては、実際にはディスクが事前に割り当てられない場合があります。

- 18 オプション: CPU フィールドには、ソケット、コア、スレッドを含む CPU の設定が含まれます。
- 19 オプション: 仮想マシンのソケット数を指定します。
- 20 オプション: ソケットあたりのコア数を指定します。
- 21 オプション: コアあたりのスレッド数を指定します。
- 22 オプション: 仮想マシンのメモリーサイズを MiB 単位で指定します。
- 23 オプション: 仮想マシンの保証されたメモリーのサイズを MiB で指定します。これは、バルーニングメカニズムによって排出されないことが保証されているメモリーの量です。詳細は、[Memory Ballooning](#) と [Optimization Settings Explained](#) を参照してください。



#### 注記

RHV 4.4.8 より前のバージョンを使用している場合は、[Red Hat Virtualization クラスターでの OpenShift の保証されたメモリー要件](#)を参照してください。

- 24 オプション: ノードのルートディスク。
- 25 オプション: ブート可能なディスクのサイズを GiB 単位で指定します。
- 26 オプション: コンピュートノードのディスクのストレージドメインの UUID を指定します。何も指定されていない場合、コンピュートノードはコントロールノードと同じストレージドメインに作成されます。(デフォルト)
- 27 オプション: 仮想マシンのネットワークインターフェイスの一覧。このパラメーターを含めると、OpenShift Container Platform はテンプレートからすべてのネットワークインターフェイスを破棄し、新規ネットワークインターフェイスを作成します。
- 28 オプション: vNIC プロファイル ID を指定します。

- 29 RHV クレデンシャルを保持するシークレットオブジェクトの名前を指定します。
- 30 オプション: インスタンスが最適化されるワークロードタイプを指定します。この値は **RHV VM** パラメーターに影響します。サポートされる値: **desktop**、**server** (デフォルト)、**high\_performance** です。**high\_performance** は、VM のパフォーマンスを向上させます。制限があります。たとえば、グラフィカルコンソールで VM にアクセスすることはできません。詳細は、**Virtual Machine Management Guide**の [ハイパフォーマンス仮想マシン、テンプレート、およびプールの設定](#) を参照してください。
- 31 オプション: AutoPinningPolicy は、このインスタンスのホストへのピニングを含む、CPU と NUMA 設定を自動的に設定するポリシーを定義します。サポートされる値は、**none**、**resize\_and\_pin** です。詳細は、**Virtual Machine Management Guide**の [Setting NUMA Nodes](#) を参照してください。
- 32 オプション: hugepages は、仮想マシンで hugepage を定義するためのサイズ (KiB) です。対応している値は **2048** および **1048576** です。詳細は、**Virtual Machine Management Guide**の [Configuring Huge Pages](#) を参照してください。
- 33 オプション: 仮想マシンに適用されるアフィニティグループ名のリスト。アフィニティグループは oVirt に存在している必要があります。



#### 注記

RHV は仮想マシンの作成時にテンプレートを使用するため、任意のパラメーターの値を指定しない場合、RHV はテンプレートに指定されるパラメーターの値を使用します。

### 2.9.2. マシンセットの作成

インストールプログラムによって作成されるコンピュートセットセットに加えて、独自のマシンセットを作成して、選択した特定のワークロードのマシンコンピューティングリソースを動的に管理できます。

#### 前提条件

- OpenShift Container Platform クラスターをデプロイすること。
- OpenShift CLI (**oc**) がインストールされている。
- **cluster-admin** パーミッションを持つユーザーとして、**oc** にログインする。

#### 手順

1. 説明されているようにマシンセット カスタムリソース (CR) サンプルを含む新規 YAML ファイルを作成し、そのファイルに **<file\_name>.yaml** という名前を付けます。  
**<clusterID>** および **<role>** パラメーターの値を設定していることを確認します。
2. オプション: 特定のフィールドに設定する値がわからない場合は、クラスターから既存のコンピュートマシンセットを確認できます。
  - a. クラスター内のコンピュートマシンセットをリスト表示するには、次のコマンドを実行します。

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

## 出力例

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

- b. 特定のコンピュータマシンセットカスタムリソース(CR)の値を表示するには、以下のコマンドを実行します。

```
$ oc get machineset <machineset_name> \
-n openshift-machine-api -o yaml
```

## 出力例

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> ❶
    name: <infrastructure_id>-<role> ❷
    namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>
        machine.openshift.io/cluster-api-machine-role: <role>
        machine.openshift.io/cluster-api-machine-type: <role>
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
    spec:
      providerSpec: ❸
      ...
```

❶ クラスターインフラストラクチャー ID。

❷ デフォルトのノードラベル。



## 注記

ユーザーがプロビジョニングしたインフラストラクチャーを持つクラスターの場合、コンピュータマシンセットは **worker** および **infra** タイプのマシンのみを作成できます。

❸ コンピュータマシンセット CR の **<providerSpec>** セクションの値は、プラットフォーム固有です。CR の **<providerSpec>** パラメーターの詳細については、プロバ

イダーのサンプルコンピュータマシンセット CR 設定を参照してください。

3. 次のコマンドを実行して **MachineSet** CR を作成します。

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

## 検証

- 次のコマンドを実行して、コンピュータマシンセットのリストを表示します。

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

## 出力例

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
agl030519-vplxk-infra-us-east-1a	1	1	1	1	11m
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

新規のマシンセットが利用可能な場合、**DESIRED** および **CURRENT** の値は一致します。マシンセットが利用可能でない場合、数分待機してからコマンドを再度実行します。

## 2.10. VSPHERE でのマシンセットの作成

VMware vSphere 上の OpenShift Container Platform クラスタで特定の目的を果たすように異なるマシンセットを作成することができます。たとえば、インフラストラクチャマシンセットおよび関連マシンを作成して、サポートするワークロードを新しいマシンに移動できます。

### 重要

高度なマシン管理およびスケーリング機能は、マシン API が機能しているクラスタでのみ使用することができます。ユーザーがプロビジョニングしたインフラストラクチャを持つクラスタでは、マシン API を使用するために追加の検証と設定が必要です。

インフラストラクチャプラットフォームタイプが **none** のクラスタは、マシン API を使用できません。この制限は、クラスタに接続されている計算マシンが、この機能をサポートするプラットフォームにインストールされている場合でも適用されます。このパラメーターは、インストール後に変更することはできません。

クラスタのプラットフォームタイプを表示するには、以下のコマンドを実行します。

```
$ oc get infrastructure cluster -o jsonpath='{.status.platform}'
```

### 2.10.1. vSphere 上のマシンセットのカスタムリソースのサンプル YAML

このサンプル YAML は、VMware vSphere で実行され、**node-role.kubernetes.io/<role>: ""** というラベルが付けられたノードを作成するマシンセットを定義します。

このサンプルでは、**<infrastructure\_id>** はクラスターのプロビジョニング時に設定したクラスター ID に基づくインフラストラクチャー ID であり、**<role>** は追加するノードラベルです。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  creationTimestamp: null
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> ❶
  name: <infrastructure_id>-<role> ❷
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> ❸
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role> ❹
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> ❺
        machine.openshift.io/cluster-api-machine-role: <role> ❻
        machine.openshift.io/cluster-api-machine-type: <role> ❼
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role> ❽
    spec:
      metadata:
        creationTimestamp: null
        labels:
          node-role.kubernetes.io/<role>: "" ❾
      providerSpec:
        value:
          apiVersion: vsphereprovider.openshift.io/v1beta1
          credentialsSecret:
            name: vsphere-cloud-credentials
          diskGiB: 120
          kind: VSphereMachineProviderSpec
          memoryMiB: 8192
          metadata:
            creationTimestamp: null
          network:
            devices:
              - networkName: "<vm_network_name>" ❿
          numCPUs: 4
          numCoresPerSocket: 1
          snapshot: ""
          template: <vm_template_name> ⓫
          userDataSecret:
            name: worker-user-data
          workspace:
            datacenter: <vcenter_datacenter_name> ⓬
```

```

datastore: <vcenter_datastore_name> 13
folder: <vcenter_vm_folder_path> 14
resourcepool: <vsphere_resource_pool> 15
server: <vcenter_server_ip> 16

```

- 1 3 5 クラスターのプロビジョニング時に設定したクラスター ID を基にするインフラストラクチャー ID を指定します。OpenShift CLI (**oc**) がインストールされている場合は、以下のコマンドを実行してインフラストラクチャー ID を取得できます。

```
$ oc get -o jsonpath='{.status.infrastructureName}' infrastructure cluster
```

- 2 4 8 インフラストラクチャー ID およびノードラベルを指定します。

- 6 7 9 追加するノードラベルを指定します。

- 10 コンピュータマシンセットをデプロイする vSphere 仮想マシンネットワークを指定します。この仮想マシンネットワークは、他のコンピューティングマシンがクラスター内に存在する場所である必要があります。

- 11 **user-5ddjd-rhcos** などの使用する vSphere 仮想マシンテンプレートを指定します。

- 12 コンピュータマシンセットをデプロイする vCenter Datacenter を指定します。

- 13 コンピュータマシンセットをデプロイする vCenter Datastore を指定します。

- 14 **/dc1/vm/user-inst-5ddjd** などの vCenter の vSphere 仮想マシンフォルダーへのパスを指定します。

- 15 仮想マシンの vSphere リソースプールを指定します。

- 16 vCenter サーバーの IP または完全修飾ドメイン名を指定します。

## 2.10.2. マシンセット管理に最低限必要な vCenter 権限

vCenter 上の OpenShift Container Platform クラスターでマシンセットを管理するには、必要なリソースの読み取り、作成、および削除を行う権限を持つアカウントを使用する必要があります。グローバル管理者権限のあるアカウントを使用すること方法が、必要なすべてのパーミッションにアクセスするための最も簡単な方法です。

グローバル管理者権限を持つアカウントを使用できない場合は、最低限必要な権限を付与するロールを作成する必要があります。次の表に、マシンセットの作成、スケーリング、削除、および OpenShift Container Platform クラスター内のマシンの削除に必要な vCenter の最小のロールと特権を示します。

### 例2.1 マシンセットの管理に必要な最小限の vCenter のロールと権限

ロールの vSphere オブジェクト	必要になる場合	必要な特権
---------------------	---------	-------

ロールの vSphere オブジェクト	必要になる場合	必要な特権
vSphere vCenter	Always	<b>InventoryService.Tagging.AttachTag</b> <b>InventoryService.Tagging.CreateCategory</b> <b>InventoryService.Tagging.CreateTag</b> <b>InventoryService.Tagging.DeleteCategory</b> <b>InventoryService.Tagging.DeleteTag</b> <b>InventoryService.Tagging.EditCategory</b> <b>InventoryService.Tagging.EditTag</b> <b>Sessions.ValidateSession</b> <b>StorageProfile.Update<sup>1</sup></b> <b>StorageProfile.View<sup>1</sup></b>
vSphere vCenter Cluster	Always	<b>Resource.AssignVMToPool</b>
vSphere Datastore	Always	<b>Datastore.AllocateSpace</b> <b>Datastore.Browse</b>
vSphere ポートグループ	Always	<b>Network.Assign</b>
仮想マシンフォルダー	Always	<b>VirtualMachine.Config.AddRemoveDevice</b> <b>VirtualMachine.Config.AdvancedConfig</b> <b>VirtualMachine.Config.Annotation</b> <b>VirtualMachine.Config.CPUCount</b> <b>VirtualMachine.Config.DiskExtend</b> <b>VirtualMachine.Config.Memory</b> <b>VirtualMachine.Config.Settings</b> <b>VirtualMachine.Interact.PowerOff</b> <b>VirtualMachine.Interact.PowerOn</b> <b>VirtualMachine.Inventory.CreateFromExisting</b> <b>VirtualMachine.Inventory.Delete</b> <b>VirtualMachine.Provisioning.Clone</b>

ロールの vSphere オブジェクト	必要になる場合	必要な特権
vSphere vCenter Datacenter	インストールプログラムが仮想マシンフォルダーを作成する場合	<b>Resource.AssignVMToPool</b> <b>VirtualMachine.Provisioning.DeployTemplate</b>
<sup>1</sup> <b>StorageProfile.Update</b> および <b>StorageProfile.View</b> 権限は、Container Storage Interface (CSI) を使用するストレージバックエンドにのみ必要です。		

次の表に、マシンセットの管理に必要なパーミッションと伝播設定の詳細を示します。

### 例2.2 必要なパーミッションおよび伝播の設定

vSphere オブジェクト	フォルダータイプ	子への伝播	パーミッションが必要
vSphere vCenter	Always	必須ではありません。	必要な特権が一覧表示
vSphere vCenter Datacenter	既存のフォルダー	必須ではありません。	<b>ReadOnly</b> パーミッション
	インストールプログラムがフォルダーを作成する	必須	必要な特権が一覧表示
vSphere vCenter Cluster	Always	必須	必要な特権が一覧表示
vSphere vCenter Datastore	Always	必須ではありません。	必要な特権が一覧表示
vSphere Switch	Always	必須ではありません。	<b>ReadOnly</b> パーミッション
vSphere ポートグループ	Always	必須ではありません。	必要な特権が一覧表示
vSphere vCenter 仮想マシンフォルダー	既存のフォルダー	必須	必要な特権が一覧表示

必要な権限のみを持つアカウントの作成に関する詳細は、vSphere ドキュメントの [vSphere Permissions and User Management Tasks](#) を参照してください。

### 2.10.3. コンピュータマシンセットを使用するための、ユーザーがプロビジョニングしたインフラストラクチャーを持つクラスターの要件

ユーザーがプロビジョニングしたインフラストラクチャーを持つクラスターでコンピュートマシンセットを使用するには、クラスター設定が Machine API の使用をサポートしていることを確認する必要があります。

### インフラストラクチャー ID の取得

コンピュートマシンセットを作成するには、クラスターのインフラストラクチャー ID を指定できる必要があります。

#### 手順

- クラスターのインフラストラクチャー ID を取得するには、次のコマンドを実行します。

```
$ oc get infrastructure cluster -o jsonpath='{.status.infrastructureName}'
```

### vSphere 認証情報の要件を満たす

コンピュートマシンセットを使用するには、マシン API が vCenter と対話できる必要があります。マシン API コンポーネントが vCenter と対話することを許可する認証情報は、**openshift-machine-api** 名前空間のシークレットに存在する必要があります。

#### 手順

- 必要な認証情報が存在するかどうかを確認するには、次のコマンドを実行します。

```
$ oc get secret \
  -n openshift-machine-api vsphere-cloud-credentials \
  -o go-template='{{range $k,$v := .data}}{{printf "%s: " $k}}{{if not $v}}{{$v}}{{else}}{{$v |
  base64decode}}{{end}}{{"\n"}}{{end}}'
```

#### 出力例

```
<vcenter-server>.password=<openshift-user-password>
<vcenter-server>.username=<openshift-user>
```

ここで、**<vcenter-server>** は vCenter サーバーの IP アドレスまたは完全修飾ドメイン名 (FQDN) であり、**<openshift-user>** および **<openshift-user-password>** は使用する OpenShift Container Platform 管理者の認証情報です。

- シークレットが存在しない場合は、次のコマンドを実行して作成します。

```
$ oc create secret generic vsphere-cloud-credentials \
  -n openshift-machine-api \
  --from-literal=<vcenter-server>.username=<openshift-user> --from-literal=<vcenter-
  server>.password=<openshift-user-password>
```

### Ignition 設定要件を満たす

仮想マシン (VM) のプロビジョニングには、有効な Ignition 設定が必要です。Ignition 設定には、**machine-config-server** アドレスと、Machine Config Operator からさらに Ignition 設定を取得するためのシステム信頼バンドルが含まれています。

デフォルトでは、この設定は **machine-api-operator** namespace の **worker-user-data** シークレットに保存されます。コンピュートマシンセットは、マシンの作成プロセス中にシークレットを参照します。

#### 手順

1. 必要なシークレットが存在するかどうかを判断するには、次のコマンドを実行します。

```
$ oc get secret \
  -n openshift-machine-api worker-user-data \
  -o go-template='{{range $k,$v := .data}}{{printf "%s: " $k}}{{if not $v}}{{$v}}{{else}}{{$v |
  base64decode}}{{end}}{{"\n"}}{{end}}'
```

### 出力例

```
disableTemplating: false
userData: ❶
{
  "ignition": {
    ...
  },
  ...
}
```

- ❶ ここでは完全な出力は省略しますが、この形式にする必要があります。

2. シークレットが存在しない場合は、次のコマンドを実行して作成します。

```
$ oc create secret generic worker-user-data \
  -n openshift-machine-api \
  --from-file=<installation_directory>/worker.ign
```

ここで **<installation\_directory>**、クラスターのインストール中にインストール資産を保管するために使用されたディレクトリーです。

### 関連情報

- [Machine Config Operator について](#)
- [RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始](#)

### 2.10.4. マシンセットの作成

インストールプログラムによって作成されるコンピュータセットセットに加えて、独自のマシンセットを作成して、選択した特定のワークロードのマシンコンピューティングリソースを動的に管理できます。



#### 注記

ユーザーがプロビジョニングしたインフラストラクチャーを使用してインストールされたクラスターには、インストールプログラムによってプロビジョニングされたインフラストラクチャーを使用したクラスターとは異なるネットワークスタックがあります。この違いの結果、自動ロードバランサー管理は、ユーザーがプロビジョニングしたインフラストラクチャーを持つクラスターではサポートされません。これらのクラスターの場合、コンピュータマシンセットは **worker** および **infra** タイプのマシンのみを作成できます。

## 前提条件

- OpenShift Container Platform クラスターをデプロイすること。
- OpenShift CLI (**oc**) がインストールされている。
- **cluster-admin** パーミッションを持つユーザーとして、**oc** にログインする。
- vCenter インスタンスに仮想マシンをデプロイするのに必要なパーミッションがあり、指定されたデータストアへのアクセス権限が必要です。
- クラスターがユーザーによってプロビジョニングされたインフラストラクチャーを使用している場合、その設定の特定のマシン API 要件を満たしています。

## 手順

1. 説明されているようにマシンセット カスタムリソース (CR) サンプルを含む新規 YAML ファイルを作成し、そのファイルに **<file\_name>.yaml** という名前を付けます。  
**<clusterID>** および **<role>** パラメーターの値を設定していることを確認します。
2. オプション: 特定のフィールドに設定する値がわからない場合は、クラスターから既存のコンピュートマシンセットを確認できます。
  - a. クラスター内のコンピュートマシンセットをリスト表示するには、次のコマンドを実行します。

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

### 出力例

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

- b. 特定のコンピュートマシンセットカスタムリソース(CR)の値を表示するには、以下のコマンドを実行します。

```
$ oc get machineset <machineset_name> \
-n openshift-machine-api -o yaml
```

### 出力例

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
  name: <infrastructure_id>-<role> 2
  namespace: openshift-machine-api
spec:
```

```

replicas: 1
selector:
  matchLabels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id>
    machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
template:
  metadata:
    labels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machine-role: <role>
      machine.openshift.io/cluster-api-machine-type: <role>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
  spec:
    providerSpec: ❸
    ...

```

- ❶ クラスターインフラストラクチャー ID。
- ❷ デフォルトのノードラベル。



### 注記

ユーザーがプロビジョニングしたインフラストラクチャーを持つクラスターの場合、コンピュータマシンセットは **worker** および **infra** タイプのマシンのみを作成できます。

- ❸ コンピュータマシンセット CR の **<providerSpec>** セクションの値は、プラットフォーム固有です。CR の **<providerSpec>** パラメーターの詳細については、プロバイダーのサンプルコンピュータマシンセット CR 設定を参照してください。

- c. ユーザーがプロビジョニングしたインフラストラクチャーを持つクラスター用のコンピュータマシンセットを作成する場合は、次の重要な値に注意してください。

### 例: vSphere providerSpec 値

```

apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
...
template:
  ...
  spec:
    providerSpec:
      value:
        apiVersion: machine.openshift.io/v1beta1
        credentialsSecret:
          name: vsphere-cloud-credentials ❶
        diskGiB: 120
        kind: VSphereMachineProviderSpec
        memoryMiB: 16384
        network:
          devices:
            - networkName: "<vm_network_name>"
        numCPUs: 4
        numCoresPerSocket: 4

```

```

snapshot: ""
template: <vm_template_name> ❷
userDataSecret:
  name: worker-user-data ❸
workspace:
  datacenter: <vcenter_datacenter_name>
  datastore: <vcenter_datastore_name>
  folder: <vcenter_vm_folder_path>
  resourcepool: <vsphere_resource_pool>
  server: <vcenter_server_address> ❹

```

- ❶ 必要な vCenter 認証情報を含む **openshift-machine-api** 名前空間のシークレットの名前。
- ❷ インストール中に作成されたクラスターの RHCOS VM テンプレートの名前。
- ❸ 必要な Ignition 設定認証情報を含む **openshift-machine-api** namespace のシークレットの名前。
- ❹ vCenter サーバーの IP アドレスまたは完全修飾ドメイン名 (FQDN)。

3. 次のコマンドを実行して **MachineSet** CR を作成します。

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

## 検証

- 次のコマンドを実行して、コンピュートマシンセットのリストを表示します。

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

## 出力例

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
agl030519-vplxk-infra-us-east-1a	1	1	1	1	11m
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

新規のマシンセットが利用可能な場合、**DESIRED** および **CURRENT** の値は一致します。マシンセットが利用可能でない場合、数分待機してからコマンドを再度実行します。

## 2.11. ベアメタル上でのコンピュートマシンセットの作成

ベアメタル上の OpenShift Container Platform クラスターで、特定の目的を果たす別のコンピューティングマシンセットを作成できます。たとえば、インフラストラクチャーマシンセットおよび関連マシンを作成して、サポートするワークロードを新しいマシンに移動できます。

## 重要

高度なマシン管理およびスケーリング機能は、マシン API が機能しているクラスターでのみ使用することができます。ユーザーがプロビジョニングしたインフラストラクチャーを持つクラスターでは、マシン API を使用するために追加の検証と設定が必要です。

インフラストラクチャープラットフォームタイプが **none** のクラスターは、マシン API を使用できません。この制限は、クラスターに接続されている計算マシンが、この機能をサポートするプラットフォームにインストールされている場合でも適用されます。このパラメーターは、インストール後に変更することはできません。

クラスターのプラットフォームタイプを表示するには、以下のコマンドを実行します。

```
$ oc get infrastructure cluster -o jsonpath='{.status.platform}'
```

### 2.11.1. ベアメタル上のコンピュータマシンセットカスタムリソースのサンプル YAML

このサンプル YAML は、ベアメタル上で実行され、**node-role.kubernetes.io/<role>: ""** というラベルが付けられたノードを作成するコンピュータマシンセットを定義します。

このサンプルでは、**<infrastructure\_id>** はクラスターのプロビジョニング時に設定したクラスター ID に基づくインフラストラクチャー ID であり、**<role>** は追加するノードラベルです。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  creationTimestamp: null
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
  name: <infrastructure_id>-<role> 2
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 3
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role> 4
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
        machine.openshift.io/cluster-api-machine-role: <role> 6
        machine.openshift.io/cluster-api-machine-type: <role> 7
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role> 8
    spec:
      metadata:
        creationTimestamp: null
        labels:
          node-role.kubernetes.io/<role>: "" 9
      providerSpec:
        value:
          apiVersion: baremetal.cluster.k8s.io/v1alpha1
```

```

hostSelector: {}
image:
  checksum: http://172.22.0.3:6181/images/rhcos-<version>.<architecture>.qcow2.<md5sum>
  url: http://172.22.0.3:6181/images/rhcos-<version>.<architecture>.qcow2
kind: BareMetalMachineProviderSpec
metadata:
  creationTimestamp: null
userData:
  name: worker-user-data

```

10

11

- 1 3 5 クラスターのプロビジョニング時に設定したクラスター ID を基にするインフラストラクチャー ID を指定します。OpenShift CLI (**oc**) がインストールされている場合は、以下のコマンドを実行してインフラストラクチャー ID を取得できます。

```
$ oc get -o jsonpath='{.status.infrastructureName}' infrastructure cluster
```

- 2 4 8 インフラストラクチャー ID およびノードラベルを指定します。

- 6 7 9 追加するノードラベルを指定します。

- 10 API VIP アドレスを使用するように **checksum** URL を編集します。

- 11 **url** URL を編集して API VIP アドレスを使用します。

## 2.11.2. マシンセットの作成

インストールプログラムによって作成されるコンピュートセットセットに加えて、独自のマシンセットを作成して、選択した特定のワークロードのマシンコンピューティングリソースを動的に管理できます。

### 前提条件

- OpenShift Container Platform クラスターをデプロイすること。
- OpenShift CLI (**oc**) がインストールされている。
- **cluster-admin** パーミッションを持つユーザーとして、**oc** にログインする。

### 手順

1. 説明されているようにマシンセット カスタムリソース (CR) サンプルを含む新規 YAML ファイルを作成し、そのファイルに **<file\_name>.yaml** という名前を付けます。  
**<clusterID>** および **<role>** パラメーターの値を設定していることを確認します。
2. オプション: 特定のフィールドに設定する値がわからない場合は、クラスターから既存のコンピュートマシンセットを確認できます。
  - a. クラスター内のコンピュートマシンセットをリスト表示するには、次のコマンドを実行します。

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

### 出力例

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

- b. 特定のコンピュータマシンセットカスタムリソース(CR)の値を表示するには、以下のコマンドを実行します。

```
$ oc get machineset <machineset_name> \
-n openshift-machine-api -o yaml
```

## 出力例

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> ❶
    name: <infrastructure_id>-<role> ❷
    namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>
        machine.openshift.io/cluster-api-machine-role: <role>
        machine.openshift.io/cluster-api-machine-type: <role>
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
    spec:
      providerSpec: ❸
      ...
```

❶ クラスターインフラストラクチャー ID。

❷ デフォルトのノードラベル。



### 注記

ユーザーがプロビジョニングしたインフラストラクチャーを持つクラスターの場合、コンピュータマシンセットは **worker** および **infra** タイプのマシンのみを作成できます。

❸ コンピュータマシンセット CR の **<providerSpec>** セクションの値は、プラットフォーム固有です。CR の **<providerSpec>** パラメーターの詳細については、プロバイダーのサンプルコンピュータマシンセット CR 設定を参照してください。

3. 次のコマンドを実行して **MachineSet** CR を作成します。

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

## 検証

- 次のコマンドを実行して、コンピュータマシンセットのリストを表示します。

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

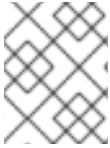
## 出力例

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
agl030519-vplxk-infra-us-east-1a	1	1	1	1	11m
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

新規のマシンセットが利用可能な場合、**DESIRED** および **CURRENT** の値は一致します。マシンセットが利用可能でない場合、数分待機してからコマンドを再度実行します。

## 第3章 マシンセットの手動によるスケーリング

マシンセットのマシンのインスタンスを追加または削除できます。



### 注記

スケーリング以外のマシンセットの要素を変更する必要がある場合は、[マシンセットの変更](#)を参照してください。

### 3.1. 前提条件

- クラスター全体のプロキシを有効にし、インストール設定から `networking.machineNetwork[].cidr` に含まれていないワーカーをスケールアップする場合、[ワーカーをプロキシオブジェクトの `noProxy` フィールドに追加](#) し、接続の問題を防ぐ必要があります。



### 重要

高度なマシン管理およびスケーリング機能は、マシン API が機能しているクラスターでのみ使用することができます。ユーザーがプロビジョニングしたインフラストラクチャーを持つクラスターでは、マシン API を使用するために追加の検証と設定が必要です。

インフラストラクチャープラットフォームタイプが **none** のクラスターは、マシン API を使用できません。この制限は、クラスターに接続されている計算マシンが、この機能をサポートするプラットフォームにインストールされている場合でも適用されます。このパラメーターは、インストール後に変更することはできません。

クラスターのプラットフォームタイプを表示するには、以下のコマンドを実行します。

```
$ oc get infrastructure cluster -o jsonpath='{.status.platform}'
```

### 3.2. マシンセットの手動によるスケーリング

マシンセットのマシンのインスタンスを追加したり、削除したりする必要がある場合、マシンセットを手動でスケーリングできます。

本書のガイダンスは、完全に自動化されたインストーラーでプロビジョニングされるインフラストラクチャーのインストールに関連します。ユーザーによってプロビジョニングされるインフラストラクチャーのカスタマイズされたインストールにはマシンセットがありません。

#### 前提条件

- OpenShift Container Platform クラスターおよび **oc** コマンドラインをインストールすること。
- **cluster-admin** パーミッションを持つユーザーとして、**oc** にログインする。

#### 手順

1. クラスターにあるマシンセットを表示します。

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

マシンセットは **<clusterid>-worker-<aws-region-az>** の形式で一覧表示されます。

2. クラスタ内にあるマシンを表示します。

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

3. 削除するマシンに注釈を設定します。

```
$ oc annotate machine/<machine_name> -n openshift-machine-api
machine.openshift.io/cluster-api-delete-machine="true"
```

4. 次のコマンドのいずれかを実行して、マシンセットをスケールします。

```
$ oc scale --replicas=2 machineset <machineset> -n openshift-machine-api
```

または、以下を実行します。

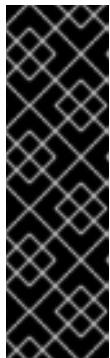
```
$ oc edit machineset <machineset> -n openshift-machine-api
```

## ヒント

または、以下の YAML を適用してマシンセットをスケーリングすることもできます。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  name: <machineset>
  namespace: openshift-machine-api
spec:
  replicas: 2
```

マシンセットをスケールアップまたはスケールダウンできます。新規マシンが利用可能になるまで数分の時間がかかります。



## 重要

デフォルトでは、マシンコントローラーは、成功するまでマシンによってサポートされるノードをドレイン (解放) しようとします。Pod 中断バジェットの設定が間違っているなど、状況によっては、ドレイン操作が成功しない可能性があります。排水操作が失敗した場合、マシンコントローラーはマシンの取り外しを続行できません。

特定のマシンの **machine.openshift.io/exclude-node-draining** にアノテーションを付けると、ノードのドレイン (解放) を省略できます。

## 検証

- 目的のマシンの削除を確認します。

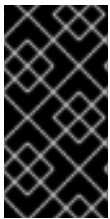
```
$ oc get machines
```

### 3.3. マシンセットの削除ポリシー

**Random**、**Newest**、および **Oldest** は3つのサポートされる削除オプションです。デフォルトは **Random** であり、これはマシンセットのスケールダウン時にランダムマシンが選択され、削除されることを意味します。削除ポリシーは、特定のマシンセットを変更し、ユースケースに基づいて設定できます。

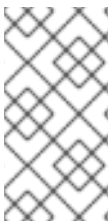
```
spec:
  deletePolicy: <delete_policy>
  replicas: <desired_replica_count>
```

削除についての特定のマシンの優先順位は、削除ポリシーに関係なく、関連するマシンにアノテーション **machine.openshift.io/cluster-api-delete-machine=true** を追加して設定できます。



#### 重要

デフォルトで、OpenShift Container Platform ルーター Pod はワーカーにデプロイされます。ルーターは Web コンソールなどの一部のクラスターリソースにアクセスすることが必要であるため、ルーター Pod をまず再配置しない限り、ワーカーのマシンセットを **0** にスケーリングできません。



#### 注記

カスタムマシンセットは、サービスを特定のノードサービスで実行し、それらのサービスがワーカーのマシンセットのスケールダウン時にコントローラーによって無視されるようにする必要があるユースケースで使用できます。これにより、サービスの中断が回避されます。

### 3.4. 関連情報

- [マシン削除フェーズのライフサイクルフック](#)

## 第4章 マシンセットの変更

ラベルの追加、インスタンスタイプの変更、ブロックストレージの変更など、マシンセットに変更を加えることができます。

Red Hat Virtualization (RHV) では、マシンセットを変更して新規ノードを別のストレージドメインにプロビジョニングすることもできます。



### 注記

他の変更なしにマシンセットをスケーリングする必要がある場合は、[マシンセットの手動によるスケーリング](#)を参照してください。

### 4.1. マシンセットの変更

マシンセットを変更するには、**MachineSet** YAML を編集します。次に、各マシンを削除するか、またはマシンセットを **0** レプリカにスケールダウンしてマシンセットに関連付けられたすべてのマシンを削除します。レプリカは必要な数にスケーリングします。マシンセットへの変更は既存のマシンに影響を与えません。

他の変更を加えずに、マシンセットをスケーリングする必要がある場合、マシンを削除する必要はありません。



### 注記

デフォルトで、OpenShift Container Platform ルーター Pod はワーカーにデプロイされます。ルーターは Web コンソールなどの一部のクラスターリソースにアクセスすることが必要であるため、ルーター Pod をまず再配置しない限り、ワーカーのマシンセットを **0** にスケーリングできません。

#### 前提条件

- OpenShift Container Platform クラスターおよび **oc** コマンドラインをインストールすること。
- **cluster-admin** パーミッションを持つユーザーとして、**oc** にログインする。

#### 手順

1. マシンセットを編集します。

```
$ oc edit machineset <machineset> -n openshift-machine-api
```

2. マシンセットを **0** にスケールダウンします。

```
$ oc scale --replicas=0 machineset <machineset> -n openshift-machine-api
```

または、以下を実行します。

```
$ oc edit machineset <machineset> -n openshift-machine-api
```

## ヒント

または、以下の YAML を適用してマシンセットをスケーリングすることもできます。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  name: <machineset>
  namespace: openshift-machine-api
spec:
  replicas: 0
```

マシンが削除されるまで待機します。

3. マシンセットを随時スケールアップします。

```
$ oc scale --replicas=2 machineset <machineset> -n openshift-machine-api
```

または、以下を実行します。

```
$ oc edit machineset <machineset> -n openshift-machine-api
```

## ヒント

または、以下の YAML を適用してマシンセットをスケーリングすることもできます。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  name: <machineset>
  namespace: openshift-machine-api
spec:
  replicas: 2
```

マシンが起動するまで待ちます。新規マシンにはマシンセットに加えられた変更が含まれます。

## 関連情報

- [マシン削除フェーズのライフサイクルフック](#)

## 4.2. RHV 上の別のストレージドメインへのノードの移行

OpenShift Container Platform コントロールプレーンおよびコンピュートノードを Red Hat Virtualization (RHV) クラスターの別のストレージドメインに移行できます。

### 4.2.1. RHV 上の別のストレージドメインへのコンピュートノードの移行

#### 前提条件

- Manager にログインしている。

- ターゲットとなるストレージドメインの名前を把握している。

## 手順

1. 仮想マシンテンプレートを特定します。

```
$ oc get -o jsonpath='{.items[0].spec.template.spec.providerSpec.value.template_name}' machineset -A
```

2. 指定したテンプレートに基づいて、Manager で新規の仮想マシンを作成します。その他の設定はすべて変更しません。詳細は、Red Hat Virtualization **Virtual Machine Management Guide**の [Creating a Virtual Machine Based on a Template](#) を参照してください。

## ヒント

新しい仮想マシンを起動する必要はありません。

3. 新規仮想マシンから新規テンプレートを作成します。**Target** にターゲットストレージドメインを指定します。詳細は、Red Hat Virtualization **Virtual Machine Management Guide**の [Creating a Template](#) を参照してください。
4. 新規テンプレートを使用して、新規マシンセットを OpenShift Container Platform クラスターに追加します。

- a. 現在のマシンセットの詳細を取得します。

```
$ oc get machineset -o yaml
```

- b. これらの詳細を使用してマシンセットを作成します。詳細は、[マシンセットの作成](#)を参照してください。

**template\_name** フィールドに新規仮想マシンテンプレート名を入力します。Manager の **New template** ダイアログで使用したものと同一テンプレート名を使用します。

- c. 古いマシンセットと新しいマシンセットの名前の両方をメモします。後続の手順でこれらを参照する必要があります。

5. ワークロードを移行します。

- a. 新規のマシンセットをスケールアップします。マシンセットの手動によるスケールアップについての詳細は、[マシンセットの手動によるスケールアップ](#)を参照してください。

OpenShift Container Platform は、古いマシンが削除されると Pod を利用可能なワーカーに移動します。

- b. 古いマシンセットをスケールダウンします。

6. 古いマシンセットを削除します。

```
$ oc delete machineset <machineset-name>
```

## 関連情報

- [マシンセットの作成](#)
- [マシンセットの手動によるスケールアップ](#)

- スケジューラーによる Pod 配置の制御

#### 4.2.2. RHV 上の別のストレージドメインへのコントロールプレーンノードの移行


OpenShift Container Platform はコントロールプレーンノードを管理しないため、コンピュータードよりも移行が容易になります。Red Hat Virtualization (RHV) 上の他の仮想マシンと同様に移行することができます。

ノードごとに個別にこの手順を実行します。

##### 前提条件

- Manager にログインしている。
- コントロールプレーンノードを特定している。Manager で **master** というラベルが付けられています。

##### 手順

1. **master** というラベルが付けられた仮想マシンを選択します。
2. 仮想マシンをシャットダウンします。
3. **Disks** タブをクリックします。
4. 仮想マシンのディスクをクリックします。
5. **More Actions**  をクリックし、**Move** を選択します。
6. ターゲットストレージドメインを選択し、移行プロセスが完了するまで待ちます。
7. 仮想マシンを起動します。
8. OpenShift Container Platform クラスタが安定していることを確認します。

```
$ oc get nodes
```

出力には、ステータスが **Ready** のノードが表示されます。

9. コントロールプレーンノードごとに、この手順を繰り返します。

## 第5章 マシンの削除

特定のマシンを削除できます。

### 5.1. 特定マシンの削除

特定のマシンを削除できます。



#### 注記

コントロールプレーンマシンは削除できません。

#### 前提条件

- OpenShift Container Platform クラスターをインストールします。
- OpenShift CLI (**oc**) がインストールされている。
- **cluster-admin** パーミッションを持つユーザーとして、**oc** にログインする。

#### 手順

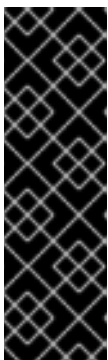
1. 次のコマンドを実行して、クラスター内のマシンを表示します。

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

コマンド出力には、**<clusterid>-<role>-<cloud\_region>** 形式のマシンのリストが含まれます。

2. 削除するマシンを特定します。
3. 次のコマンドを実行してマシンを削除します。

```
$ oc delete machine <machine> -n openshift-machine-api
```



#### 重要

デフォルトでは、マシンコントローラーは、成功するまでマシンによってサポートされるノードをドレイン (解放) しようとします。Pod 中断バジェットの設定が間違っているなど、状況によっては、ドレイン操作が成功しない可能性があります。排水操作が失敗した場合、マシンコントローラーはマシンの取り外しを続行できません。

特定のマシンの **machine.openshift.io/exclude-node-draining** にアノテーションを付けると、ノードのドレイン (解放) を省略できます。

削除するマシンがマシンセットに属している場合は、指定された数のレプリカを満たす新しいマシンがすぐに作成されます。

### 5.2. マシン削除フェーズのライフサイクルフック

マシンのライフサイクルフックは、通常のライフサイクルプロセスが中断できる、マシンの調整ライフサイクル内のポイントです。マシンの **Deleting** フェーズでは、これらの中断により、コンポーネントがマシンの削除プロセスを変更する機会が提供されます。

### 5.2.1. 用語と定義

マシンの削除フェーズのライフサイクルフックの動作を理解するには、次の概念を理解する必要があります。

#### 調整

調整は、コントローラーがクラスターの実際の状態とクラスターを設定するオブジェクトをオブジェクト仕様の要件と一致させようとするプロセスです。

#### マシンコントローラー

マシンコントローラーは、マシンの調整ライフサイクルを管理します。クラウドプラットフォーム上のマシンの場合、マシンコントローラーは OpenShift Container Platform コントローラーとクラウドプロバイダーのプラットフォーム固有のアクチュエーターを組み合わせたものです。マシンの削除のコンテキストでは、マシンコントローラーは次のアクションを実行します。

- マシンによってバックアップされているノードをドレインします。
- クラウドプロバイダーからマシンインスタンスを削除します。
- **Node** オブジェクトを削除します。

#### ライフサイクルフック

ライフサイクルフックは、通常のライフサイクルプロセスを中断できる、オブジェクトの調整ライフサイクル内の定義されたポイントです。コンポーネントはライフサイクルフックを使用してプロセスに変更を注入し、望ましい結果を達成できます。マシンの **Deleting** フェーズには2つのライフサイクルフックがあります。

- **preDrain** ライフサイクルフックは、マシンによってサポートされているノードをドレインする前に解決する必要があります。
- **preTerminate** ライフサイクルフックは、インスタンスをインフラストラクチャプロバイダーから削除する前に解決する必要があります。

#### フック実装コントローラー

フック実装コントローラーは、ライフサイクルフックと対話できる、マシンコントローラー以外のコントローラーです。フック実装コントローラーは、次の1つ以上のアクションを実行できます。

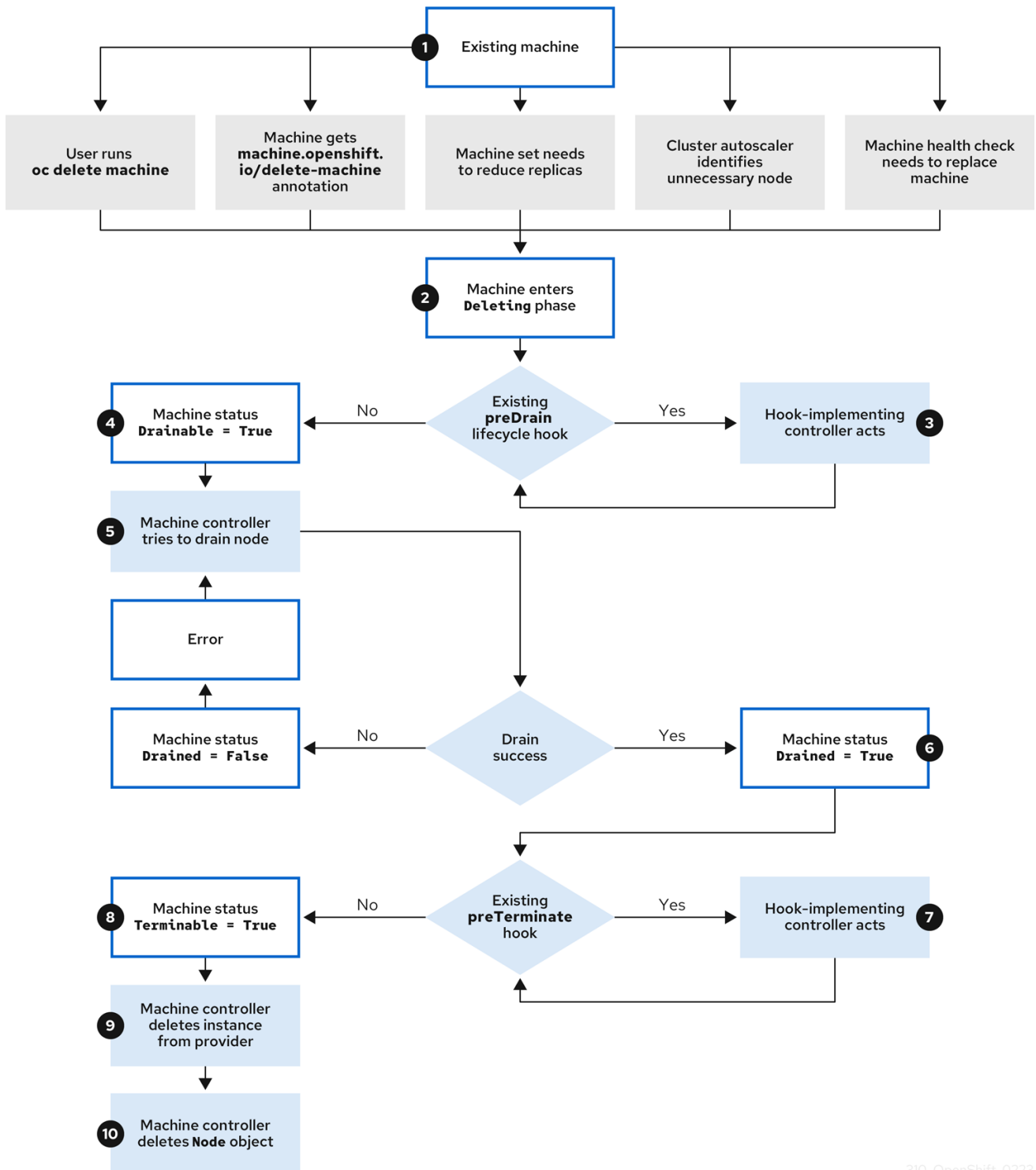
- ライフサイクルフックを追加します。
- ライフサイクルフックに応答します。
- ライフサイクルフックを削除します。

各ライフサイクルフックには1つのフック実装コントローラーがありますが、フック実装コントローラーは1つ以上のフックを管理できます。

### 5.2.2. マシン削除処理順序

OpenShift Container Platform 4.11 には、マシン削除フェーズ用の 2 つのライフサイクルフック (**preDrain** と **preTerminate**) があります。特定のライフサイクルポイントのすべてのフックが削除されると、調整は通常どおり続行されます。

図5.1 マシン削除のフロー



310\_OpenShift\_0223

マシンの **Deleting** フェーズは次の順序で続行されます。

1. 既存のマシンは、次のいずれかの理由により削除される予定です。
  - **cluster-admin** 権限を持つユーザーは、**oc delete machine** コマンドを使用します。
  - マシンは **machine.openshift.io/delete-machine** アノテーションを取得します。

- マシンを管理するマシンセットは、調整の一環としてレプリカ数を減らすために、そのマシンに削除のマークを付けます。
  - クラスターオートスケーラーは、クラスターのデプロイメントニーズを満たすために必要なノードを特定します。
  - マシンの健全性チェックは、異常なマシンを置き換えるように設定されています。
2. マシンは **Deleting** フェーズに入ります。このフェーズでは、マシンは削除対象としてマークされていますが、API にはまだ存在しています。
  3. **preDrain** ライフサイクルフックが存在する場合、それを管理するフック実装コントローラーは指定されたアクションを実行します。  
すべての **preDrain** ライフサイクルフックが満たされるまで、マシンのステータス条件 **Drainable** は **False** に設定されます。
  4. 未解決の **preDrain** ライフサイクルフックはなく、マシンのステータス条件 **Drainable** が **True** に設定されています。
  5. マシンコントローラーは、マシンによってサポートされているノードをドレインしようとしません。
    - ドレインが失敗した場合、**Drained** は、**False** に設定され、マシンコントローラーはノードのドレインを再度試行します。
    - ドレインに成功すると、**Drained** は **True** に設定されます。
  6. マシンのステータス条件 **Drained** は **True** に設定されます。
  7. **preTerminate** ライフサイクルフックが存在する場合、それを管理するフック実装コントローラーは指定されたアクションを実行します。  
すべての **preTerminate** ライフサイクルフックが満たされるまで、マシンのステータス条件 **Terminable** は **False** に設定されます。
  8. 未解決の **preTerminate** ライフサイクルフックはなく、マシンのステータス条件 **Terminable** が **True** に設定されています。
  9. マシンコントローラーは、インフラストラクチャプロバイダーからインスタンスを削除します。
  10. マシンコントローラーは **Node** オブジェクトを削除します。

### 5.2.3. 削除ライフサイクルフック設定

次の YAML スニペットは、マシンセット内の削除ライフサイクルフック設定の形式と配置を示しています。

#### preDrain ライフサイクルフックを示す YAML スニペット

```
apiVersion: machine.openshift.io/v1beta1
kind: Machine
metadata:
  ...
spec:
  lifecycleHooks:
    preDrain:
```

```
- name: <hook_name> ❶
  owner: <hook_owner> ❷
...
```

❶ **preDrain** ライフサイクルフックの名前。

❷ **preDrain** ライフサイクルフックを管理するフック実装コントローラー。

### preTerminate ライフサイクルフックを示す YAML スニペット

```
apiVersion: machine.openshift.io/v1beta1
kind: Machine
metadata:
  ...
spec:
  lifecycleHooks:
    preTerminate:
      - name: <hook_name> ❶
        owner: <hook_owner> ❷
  ...
```

❶ **preTerminate** ライフサイクルフックの名前。

❷ **preTerminate** ライフサイクルフックを管理するフック実装コントローラー。

### ライフサイクルフックの設定例

次の例は、マシンの削除プロセスを中断する複数の架空のライフサイクルフックの実装を示しています。

### ライフサイクルフックの設定例

```
apiVersion: machine.openshift.io/v1beta1
kind: Machine
metadata:
  ...
spec:
  lifecycleHooks:
    preDrain: ❶
      - name: MigrateImportantApp
        owner: my-app-migration-controller
    preTerminate: ❷
      - name: BackupFileSystem
        owner: my-backup-controller
      - name: CloudProviderSpecialCase
        owner: my-custom-storage-detach-controller ❸
      - name: WaitForStorageDetach
        owner: my-custom-storage-detach-controller
  ...
```

❶ 単一のライフサイクルフックを含む **preDrain** ライフサイクルフックスタンザ。

❷ 3つのライフサイクルフックを含む **preTerminate** ライフサイクルフックスタンザ。

- 3 2つの **preTerminate** ライフサイクルフック **CloudProviderSpecialCase** と **WaitForStorageDetach** を管理するフック実装コントローラー。

#### 5.2.4. Operator 開発者向けのマシン削除ライフサイクルフックの例

Operator は、マシン削除フェーズのライフサイクルフックを使用して、マシン削除プロセスを変更できます。次の例は、Operator がこの機能を使用できる方法を示しています。

##### preDrain ライフサイクルフックの使用例

###### 積極的にマシンを入れ替える

Operator は、削除されたマシンのインスタンスを削除する前に、**preDrain** ライフサイクルフックを使用して、代替マシンが正常に作成され、クラスターに参加していることを確認できます。これにより、マシンの交換中の中断や、すぐに初期化されない交換用インスタンスの影響を軽減できます。

###### カスタムドレインロジックの実装

Operator は、**preDrain** ライフサイクルフックを使用して、マシンコントローラーのドレインロジックを別のドレインコントローラーに置き換えることができます。ドレインロジックを置き換えることにより、Operator は各ノードのワークロードのライフサイクルをより柔軟に制御できるようになります。

たとえば、マシンコントローラーのドレインライブラリーは順序付けをサポートしていませんが、カスタムドレインプロバイダーはこの機能を提供できます。カスタムドレインプロバイダーを使用することで、Operator はノードをドレインする前にミッションクリティカルなアプリケーションの移動を優先して、クラスターの容量が制限されている場合にサービスの中断を最小限に抑えることができます。

##### preTerminate ライフサイクルフックの使用例

###### ストレージの切り離しを確認する

Operator は、**preTerminate** ライフサイクルフックを使用して、マシンがインフラストラクチャプロバイダーから削除される前に、マシンに接続されているストレージが確実に切り離されるようにすることができます。

###### ログの信頼性の向上

ノードがドレインされた後、ログエクスポートデーモンがログを集中ログシステムに同期するのに時間がかかります。

ロギング Operator は、**preTerminate** ライフサイクルフックを使用して、ノードがドレインするとき、マシンがインフラストラクチャプロバイダーから削除されるときとの間に遅延を追加できます。この遅延により、Operator は主要なワークロードが削除され、ログバックログに追加されないようにする時間が確保されます。ログバックログに新しいデータが追加されていない場合、ログエクスポートは同期プロセスに追いつくことができるため、すべてのアプリケーションログが確実にキャプチャーされます。

#### 5.2.5. マシンライフサイクルフックによるクォーラム保護

Machine API Operator を使用する OpenShift Container Platform クラスターの場合、etcd Operator はマシン削除フェーズのライフサイクルフックを使用して、クォーラム保護メカニズムを実装します。

**preDrain** ライフサイクルフックを使用することにより、etcd Operator は、コントロールプレーンマシン上の Pod がいつドレインされ、削除されるかを制御できます。etcd クォーラムを保護するために、etcd Operator は、etcd メンバーをクラスター内の新しいノードに移行するまで、そのメンバーの削除を防ぎます。

このメカニズムにより、etcd Operator は etcd クォーラムのメンバーを正確に制御できるようになり、マシン API Operator は etcd クラスターの特別な操作知識がなくても、コントロールプレーンマシンを安全に作成および削除できるようになります。

### 5.2.5.1. クォーラム保護処理順序によるコントロールプレーンの削除

OpenShift Container Platform 4.11 以降のクラスター上でコントロールプレーンマシンが置き換えられると、クラスターには一時的に 4 つのコントロールプレーンマシンが存在します。4 番目のコントロールプレーンノードがクラスターに参加すると、etcd Operator は代替ノードで新しい etcd メンバーを開始します。etcd Operator は、古いコントロールプレーンマシンが削除対象としてマークされていることを確認すると、古いノード上の etcd メンバーを停止し、代替の etcd メンバーをクラスターのクォーラムに参加するように昇格させます。

コントロールプレーンマシンの **Deleting** フェーズは、以下の順序で続行されます。

1. コントロールプレーンマシンは削除される予定です。
2. コントロールプレーンマシンは **Deleting** フェーズに入ります。
3. **preDrain** ライフサイクルフックを満たすために、etcd Operator は次のアクションを実行します。
  - a. etcd Operator は、4 番目のコントロールプレーンマシンが etcd メンバーとしてクラスターに追加されるまで待機します。この新しい etcd メンバーの状態は **Running** ですが、etcd リーダーから完全なデータベース更新を受信するまでは **ready** ができていません。
  - b. 新しい etcd メンバーが完全なデータベース更新を受け取ると、etcd Operator は新しい etcd メンバーを投票メンバーに昇格させ、古い etcd メンバーをクラスターから削除します。

この移行が完了すると、古い etcd Pod とそのデータは安全に削除されるため、**preDrain** ライフサイクルフックが削除されます。

4. コントロールプレーンマシンのステータス条件 **Drainable** が **True** に設定されます。
5. マシンコントローラーは、コントロールプレーンマシンによってサポートされているノードをドレインしようとします。
  - ドレインが失敗した場合、**Drained** は、**False** に設定され、マシンコントローラーはノードのドレインを再度試行します。
  - ドレインに成功すると、**Drained** は **True** に設定されます。
6. コントロールプレーンマシンのステータス条件 **Drained** が **True** に設定されます。
7. 他の Operator が **preTerminate** ライフサイクルフックを追加していない場合、コントロールプレーンのマシンステータス条件 **Terminable** は **True** に設定されます。
8. マシンコントローラーは、インフラストラクチャプロバイダーからインスタンスを削除します。
9. マシンコントローラーは **Node** オブジェクトを削除します。

etcd クォーラム保護の **preDrain** ライフサイクルフックを示す YAML スニペット

```
apiVersion: machine.openshift.io/v1beta1
kind: Machine
```

```
metadata:
...
spec:
  lifecycleHooks:
    preDrain:
      - name: EtcdQuorumOperator ❶
        owner: clusteroperator/etcd ❷
...

```

- ❶ **preDrain** ライフサイクルフックの名前。
- ❷ **preDrain** ライフサイクルフックを管理するフック実装コントローラー。

## 5.3. 関連情報

- [正常でない etcd メンバーの置き換え](#)

## 第6章 OPENSIFT CONTAINER PLATFORM クラスターへの自動スケーリングの適用

自動スケーリングの OpenShift Container Platform クラスターへの適用には、クラスターへの Cluster Autoscaler のデプロイと各マシンタイプの Machine Autoscaler のデプロイが必要です。



### 重要

Cluster Autoscaler は、マシン API が機能しているクラスターでのみ設定できます。

### 6.1. CLUSTER AUTOSCALER について

Cluster Autoscaler は、現行のデプロイメントのニーズに合わせて OpenShift Container Platform クラスターのサイズを調整します。これは、Kubernetes 形式の宣言引数を使用して、特定のクラウドプロバイダーのオブジェクトに依存しないインフラストラクチャー管理を提供します。Cluster Autoscaler には cluster スコープがあり、特定の namespace には関連付けられていません。

Cluster Autoscaler は、リソース不足のために現在のワーカーノードのいずれにもスケジュールできない Pod がある場合や、デプロイメントのニーズを満たすために別のノードが必要な場合に、クラスターのサイズを拡大します。Cluster Autoscaler は、指定される制限を超えてクラスターリソースを拡大することはありません。

Cluster Autoscaler は、コントロールプレーンノードを管理しない場合でも、クラスター内のすべてのノードのメモリ、CPU、および GPU の合計を計算します。これらの値は、単一マシン指向ではありません。これらは、クラスター全体での全リソースの集約です。たとえば、最大メモリリソースの制限を設定する場合、Cluster Autoscaler は現在のメモリ使用量を計算する際にクラスター内のすべてのノードを含めます。この計算は、Cluster Autoscaler にワーカーリソースを追加する容量があるかどうかを判断するために使用されます。



### 重要

作成する **ClusterAutoscaler** リソース定義の **maxNodesTotal** 値が、クラスター内のマシンの想定される合計数に対応するのに十分な大きさの値であることを確認します。この値は、コントロールプレーンマシンの数とスケーリングする可能性のあるコンピュータマシンの数に対応できる値である必要があります。

Cluster Autoscaler は 10 秒ごとに、クラスターで不要なノードをチェックし、それらを削除します。Cluster Autoscaler は、以下の条件が適用される場合に、ノードを削除すべきと考えます。

- ノードの使用率はクラスターの **ノード使用率レベル** のしきい値よりも低い場合。ノード使用率レベルとは、要求されたリソースの合計をノードに割り当てられたリソースで除算したものです。**ClusterAutoscaler** カスタムリソースで値を指定しない場合、Cluster Autoscaler は 50% の使用率に対応するデフォルト値 **0.5** を使用します。
- Cluster Autoscaler がノードで実行されているすべての Pod を他のノードに移動できる。Kubernetes スケジューラーは、ノード上の Pod のスケジュールを担当します。
- Cluster Autoscaler で、スケールダウンが無効にされたアノテーションがない。

以下のタイプの Pod がノードにある場合、Cluster Autoscaler はそのノードを削除しません。

- 制限のある Pod の Disruption Budget (停止状態の予算、PDB) を持つ Pod。
- デフォルトでノードで実行されない Kube システム Pod。

- PDB を持たないか、または制限が厳しい PDB を持つ Kuber システム Pod。
- デプロイメント、レプリカセット、またはステートフルセットなどのコントローラーオブジェクトによってサポートされない Pod。
- ローカルストレージを持つ Pod。
- リソース不足、互換性のないノードセクターまたはアフィニティー、一致する非アフィニティーなどにより他の場所に移動できない Pod。
- それらに **"cluster-autoscaler.kubernetes.io/safe-to-evict": "true"** アノテーションがない場合、**"cluster-autoscaler.kubernetes.io/safe-to-evict": "false"** アノテーションを持つ Pod。

たとえば、CPU の上限を 64 コアに設定し、それぞれ 8 コアを持つマシンのみを作成するように Cluster Autoscaler を設定したとします。クラスターが 30 コアで起動する場合、Cluster Autoscaler は最大で 4 つのノード (合計 32 コア) を追加できます。この場合、総計は 62 コアになります。

Cluster Autoscaler を設定する場合、使用に関する追加の制限が適用されます。

- 自動スケーリングされたノードグループにあるノードを直接変更しないようにしてください。同じノードグループ内のすべてのノードには同じ容量およびラベルがあり、同じシステム Pod を実行します。
- Pod の要求を指定します。
- Pod がすぐに削除されるのを防ぐ必要がある場合、適切な PDB を設定します。
- クラウドプロバイダーのクォータが、設定する最大のノードプールに対応できる十分な大きさであることを確認します。
- クラウドプロバイダーで提供されるものなどの、追加のノードグループの Autoscaler を実行しないようにしてください。

Horizontal Pod Autoscaler (HPA) および Cluster Autoscaler は複数の異なる方法でクラスターリソースを変更します。HPA は、現在の CPU 負荷に基づいてデプロイメント、またはレプリカセットのレプリカ数を変更します。負荷が増大すると、HPA はクラスターで利用できるリソース量に関係なく、新規レプリカを作成します。十分なリソースがない場合、Cluster Autoscaler はリソースを追加し、HPA で作成された Pod が実行できるようにします。負荷が減少する場合、HPA は一部のレプリカを停止します。この動作によって一部のノードの使用率が低くなるか、または完全に空になる場合、Cluster Autoscaler は不必要なノードを削除します。

Cluster Autoscaler は Pod の優先順位を考慮に入れます。Pod の優先順位とプリエンプション機能により、クラスターに十分なリソースがない場合に優先順位に基づいて Pod のスケジューリングを有効にできますが、Cluster Autoscaler はクラスターがすべての Pod を実行するのに必要なリソースを確保できます。これら両方の機能の意図を反映するべく、Cluster Autoscaler には優先順位のカットオフ機能が含まれています。このカットオフを使用して Best Effort の Pod をスケジューリングできますが、これにより Cluster Autoscaler がリソースを増やすことはなく、余分なリソースがある場合にのみ実行されます。

カットオフ値よりも低い優先順位を持つ Pod は、クラスターをスケールアップせず、クラスターのスケールダウンを防ぐこともありません。これらの Pod を実行するために新規ノードは追加されず、これらの Pod を実行しているノードはリソースを解放するために削除される可能性があります。

クラスターの自動スケーリングは、マシン API が利用可能なプラットフォームでサポートされています。

## 6.2. CLUSTER AUTOSCALER の設定

まず Cluster Autoscaler をデプロイし、リソースの自動スケーリングを OpenShift Container Platform クラスターで管理します。



### 注記

Cluster Autoscaler のスコープはクラスター全体に設定されるため、クラスター用に1つの Cluster Autoscaler のみを作成できます。

### 6.2.1. ClusterAutoscaler リソース定義

この **ClusterAutoscaler** リソース定義は、Cluster Autoscaler のパラメーターおよびサンプル値を表示します。

```
apiVersion: "autoscaling.openshift.io/v1"
kind: "ClusterAutoscaler"
metadata:
  name: "default"
spec:
  podPriorityThreshold: -10 1
  resourceLimits:
    maxNodesTotal: 24 2
  cores:
    min: 8 3
    max: 128 4
  memory:
    min: 4 5
    max: 256 6
  gpus:
    - type: nvidia.com/gpu 7
      min: 0 8
      max: 16 9
    - type: amd.com/gpu
      min: 0
      max: 4
  scaleDown: 10
    enabled: true 11
    delayAfterAdd: 10m 12
    delayAfterDelete: 5m 13
    delayAfterFailure: 30s 14
    unneededTime: 5m 15
    utilizationThreshold: "0.4" 16
```

- 1 Cluster Autoscaler に追加のノードをデプロイさせるために Pod が超えている必要のある優先順位を指定します。32 ビットの整数値を入力します。**podPriorityThreshold** 値は、各 Pod に割り当てた **PriorityClass** の値と比較されます。
- 2 デプロイするノードの最大数を指定します。この値は、Autoscaler が制御するマシンだけでなく、クラスターにデプロイされるマシンの合計数です。この値は、すべてのコントロールプレーンおよびコンピュータマシン、および **MachineAutoscaler** リソースに指定するレプリカの合計数に対応するのに十分な大きさの値であることを確認します。

- 3 クラスターにデプロイするコアの最小数を指定します。
- 4 クラスターにデプロイするコアの最大数を指定します。
- 5 クラスターのメモリーの最小量 (GiB 単位) を指定します。
- 6 クラスターのメモリーの最大量 (GiB 単位) を指定します。
- 7 オプション: デプロイする GPU ノードのタイプを指定します。 [nvidia.com/gpu](https://nvidia.com/gpu) および [amd.com/gpu](https://amd.com/gpu) のみが有効なタイプです。
- 8 クラスターにデプロイする GPU の最小数を指定します。
- 9 クラスターにデプロイする GPU の最大数を指定します。
- 10 このセクションでは、有効な [ParseDuration](#) 期間 ( **ns**、**us**、**ms**、**s**、**m**、および **h** を含む) を使用して各アクションについて待機する期間を指定できます。
- 11 Cluster Autoscaler が不必要なノードを削除できるかどうかを指定します。
- 12 オプション: ノードが最後に **追加** されてからノードを削除するまで待機する期間を指定します。値を指定しない場合、デフォルト値の **10m** が使用されます。
- 13 オプション: ノードが最後に **削除** されてからノードを削除するまで待機する期間を指定します。値を指定しない場合、デフォルト値の **0s** が使用されます。
- 14 オプション: スケールダウンが失敗してからノードを削除するまで待機する期間を指定します。値を指定しない場合、デフォルト値の **3m** が使用されます。
- 15 オプション: 不要なノードが削除の対象となるまでの期間を指定します。値を指定しない場合、デフォルト値の **10m** が使用されます。
- 16 オプション: **ノード使用率レベル** を指定します。ノードの使用率がこの値を下回ると、不要なノードが削除の対象となります。ノード使用率は、要求されたリソースをそのノードに割り当てられたリソースで割ったもので、**"0"** より大きく **"1"** より小さい値でなければなりません。値を指定しない場合、Cluster Autoscaler は 50% の使用率に対応するデフォルト値 **"0.5"** を使用します。この値は文字列として表現する必要があります。

### 注記

スケーリング操作の実行時に、Cluster Autoscaler は、デプロイするコアの最小および最大数、またはクラスター内のメモリー量などの **ClusterAutoscaler** リソース定義に設定された範囲内に残ります。ただし、Cluster Autoscaler はそれらの範囲内に留まるようクラスターの現在の値を修正しません。

Cluster Autoscaler がノードを管理しない場合でも、最小および最大の CPU、メモリー、および GPU の値は、クラスター内のすべてのノードのこれらのリソースを計算することによって決定されます。たとえば、Cluster Autoscaler がコントロールプレーンノードを管理しない場合でも、コントロールプレーンノードはクラスターのメモリーの合計に考慮されます。

## 6.2.2. Cluster Autoscaler のデプロイ

Cluster Autoscaler をデプロイするには、**ClusterAutoscaler** リソースのインスタンスを作成します。

## 手順

1. カスタマイズされたリソース定義を含む **ClusterAutoscaler** リソースの YAML ファイルを作成します。
2. クラスターにリソースを作成します。

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

❶ **<filename>** は、カスタマイズしたリソースファイルの名前です。

## 6.3. 次のステップ

- Cluster Autoscaler の設定後に、1つ以上の Machine Autoscaler を設定する必要があります。

## 6.4. MACHINE AUTOSCALER

Machine Autoscaler は、マシンセットで OpenShift Container Platform クラスターにデプロイするマシン数を調整します。デフォルトの **worker** マシンセットおよび作成する他のマシンセットの両方をスケールリングできます。Machine Autoscaler は、追加のデプロイメントをサポートするのに十分なリソースがクラスターにない場合に追加のマシンを作成します。**MachineAutoscaler** リソースの値への変更 (例: インスタンスの最小または最大数) は、それらがターゲットとするマシンセットに即時に適用されます。



### 重要

マシンをスケールリングするには、Cluster Autoscaler の Machine Autoscaler をデプロイする必要があります。Cluster Autoscaler は、スケールリングできるリソースを判別するために、Machine Autoscaler が設定するアノテーションをマシンセットで使用します。Machine Autoscaler を定義せずにクラスター Autoscaler を定義する場合、クラスター Autoscaler はクラスターをスケールリングできません。

## 6.5. MACHINE AUTOSCALER の設定

Cluster Autoscaler の設定後に、クラスターのスケールリングに使用されるマシンセットを参照する **MachineAutoscaler** リソースをデプロイします。



### 重要

**ClusterAutoscaler** リソースのデプロイ後に、1つ以上の **MachineAutoscaler** リソースをデプロイする必要があります。



### 注記

各マシンセットに対して別々のリソースを設定する必要があります。マシンセットはそれぞれのリージョンごとに異なるため、複数のリージョンでマシンのスケールリングを有効にする必要があるかどうかを考慮してください。スケールリングするマシンセットには1つ以上のマシンが必要です。

### 6.5.1. MachineAutoscaler リソース定義

この **MachineAutoscaler** リソース定義は、Machine Autoscaler のパラメーターおよびサンプル値を表示します。

```
apiVersion: "autoscaling.openshift.io/v1beta1"
kind: "MachineAutoscaler"
metadata:
  name: "worker-us-east-1a" ❶
  namespace: "openshift-machine-api"
spec:
  minReplicas: 1 ❷
  maxReplicas: 12 ❸
  scaleTargetRef: ❹
    apiVersion: machine.openshift.io/v1beta1
    kind: MachineSet ❺
    name: worker-us-east-1a ❻
```

- ❶ Machine Autoscaler の名前を指定します。この Machine Autoscaler がスケーリングするマシンセットを簡単に特定できるようにするには、スケーリングするマシンセットの名前を指定するか、またはこれを組み込みます。マシンセットの名前は、**<clusterid>-<machineset>-<region>** の形式を使用します。
- ❷ Cluster Autoscaler がクラスターのスケーリングを開始した後に、指定されたゾーンに残っている必要のある指定されたタイプのマシンの最小数を指定します。AWS、GCP、Azure、RHOSP または vSphere で実行している場合は、この値は **0** に設定できます。他のプロバイダーの場合は、この値は **0** に設定しないでください。

特殊なワークロードに使用されるコストがかかり、用途が限られたハードウェアを稼働する場合などのユースケースにはこの値を **0** に設定するか、若干大きいマシンを使用してマシンセットをスケーリングすることで、コストを節約できます。Cluster Autoscaler は、マシンが使用されていない場合にマシンセットをゼロにスケールダウンします。



### 重要

インストーラーでプロビジョニングされるインフラストラクチャーの OpenShift Container Platform インストールプロセス時に作成される 3 つのコンピュートマシンセットについては、**spec.minReplicas** の値を **0** に設定しないでください。

- ❸ Cluster Autoscaler がクラスタースケリングの開始後に指定されたゾーンにデプロイできる指定されたタイプのマシンの最大数を指定します。**ClusterAutoscaler** リソース定義の **maxNodesTotal** 値が、Machine AutoScaler がこの数のマシンをデプロイするのに十分な大きさの値であることを確認します。
- ❹ このセクションでは、スケーリングする既存のマシンセットを記述する値を指定します。
- ❺ **kind** パラメーターの値は常に **MachineSet** です。
- ❻ **name** の値は、**metadata.name** パラメーター値に示されるように既存のマシンセットの名前に一致する必要があります。

## 6.5.2. Machine Autoscaler のデプロイ

Machine Autoscaler をデプロイするには、**MachineAutoscaler** リソースのインスタンスを作成します。

## 手順

1. カスタマイズされたリソース定義を含む **MachineAutoscaler** リソースの YAML ファイルを作成します。
2. クラスターにリソースを作成します。

```
$ oc create -f <filename>.yaml 1
```

1 <filename> は、カスタマイズしたリソースファイルの名前です。

## 6.6. 関連情報

- Pod の優先順位についての詳細は、OpenShift Container Platform の [Including pod priority in pod scheduling decisions](#) を参照してください。

## 第7章 インフラストラクチャーマシンセットの作成

### 重要

高度なマシン管理およびスケーリング機能は、マシン API が機能しているクラスターでのみ使用することができます。ユーザーがプロビジョニングしたインフラストラクチャーを持つクラスターでは、マシン API を使用するために追加の検証と設定が必要です。

インフラストラクチャープラットフォームタイプが **none** のクラスターは、マシン API を使用できません。この制限は、クラスターに接続されている計算マシンが、この機能をサポートするプラットフォームにインストールされている場合でも適用されます。このパラメーターは、インストール後に変更することはできません。

クラスターのプラットフォームタイプを表示するには、以下のコマンドを実行します。

```
$ oc get infrastructure cluster -o jsonpath='{.status.platform}'
```

インフラストラクチャーマシンセットを使用して、デフォルトのルーター、統合コンテナイメージレジストリー、およびクラスターメトリクスおよびモニタリングのコンポーネントなどのインフラストラクチャーコンポーネントのみをホストするマシンを作成できます。これらのインフラストラクチャーマシンは、環境の実行に必要なサブスクリプションの合計数にカウントされません。

実稼働デプロイメントでは、インフラストラクチャーコンポーネントを保持するために3つ以上のマシンセットをデプロイすることが推奨されます。OpenShift Logging と Red Hat OpenShift Service Mesh の両方が Elasticsearch をデプロイします。これには、3つのインスタンスを異なるノードにインストールする必要があります。これらの各ノードは、高可用性のために異なるアベイラビリティゾーンにデプロイできます。この設定には、可用性ゾーンごとに1つずつ、合計3つの異なるマシンセットが必要です。複数のアベイラビリティゾーンを持たないグローバル Azure リージョンでは、アベイラビリティセットを使用して高可用性を確保できます。

### 7.1. OPENSIFT CONTAINER PLATFORM インフラストラクチャーコンポーネント

以下のインフラストラクチャーワークロードでは、OpenShift Container Platform ワーカーのサブスクリプションは不要です。

- マスターで実行される Kubernetes および OpenShift Container Platform コントロールプレーンサービス
- デフォルトルーター
- 統合コンテナイメージレジストリー
- HAProxy ベースの Ingress Controller
- ユーザー定義プロジェクトのモニタリング用のコンポーネントを含む、クラスターメトリクスの収集またはモニタリングサービス
- クラスター集計ロギング
- サービスブローカー
- Red Hat Quay

- Red Hat OpenShift Data Foundation
- Red Hat Advanced Cluster Manager
- Kubernetes 用 Red Hat Advanced Cluster Security
- Red Hat OpenShift GitOps
- Red Hat OpenShift Pipelines

他のコンテナ、Pod またはコンポーネントを実行するノードは、サブスクリプションが適用される必要のあるワーカーノードです。

インフラストラクチャーノードおよびインフラストラクチャーノードで実行できるコンポーネントの詳細は、[OpenShift sizing and subscription guide for enterprise Kubernetes](#) の "Red Hat OpenShift control plane and infrastructure nodes" セクションを参照してください。

インフラストラクチャーノードを作成するには、[マシンセットを使用する](#) か、[ノードにラベルを付ける](#) か、[マシン設定プールを使用します](#)。

## 7.2. 実稼働環境用のインフラストラクチャーマシンセットの作成

実稼働デプロイメントでは、インフラストラクチャーコンポーネントを保持するために 3 つ以上のマシンセットをデプロイすることが推奨されます。OpenShift Logging と Red Hat OpenShift Service Mesh の両方が Elasticsearch をデプロイします。これには、3 つのインスタンスを異なるノードにインストールする必要があります。これらの各ノードは、高可用性のために異なるアベイラビリティゾーンにデプロイできます。このような設定では、各アベイラビリティゾーンに 1 つずつ、3 つの異なるマシンセットが必要です。複数のアベイラビリティゾーンを持たないグローバル Azure リージョンでは、アベイラビリティセットを使用して高可用性を確保できます。

### 7.2.1. 異なるクラウドのマシンセットの作成

クラウドのサンプルマシンセットを使用します。

#### 7.2.1.1. Alibaba Cloud 上のマシンセットカスタムリソースのサンプル YAML

このサンプル YAML は、リージョン内の指定された Alibaba Cloud ゾーンで実行され、**node-role.kubernetes.io/infra: ""** というラベルの付いたノードを作成するマシンセットを定義します。

このサンプルでは、**infrastructure\_id** はクラスターのプロビジョニング時に設定したクラスター ID に基づくインフラストラクチャー ID であり、**<infra>** は追加するノードラベルです。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> ❶
    machine.openshift.io/cluster-api-machine-role: <infra> ❷
    machine.openshift.io/cluster-api-machine-type: <infra> ❸
  name: <infrastructure_id>-<infra>-<zone> ❹
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
```

```

machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<infra>-<zone> 6
template:
  metadata:
    labels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 7
      machine.openshift.io/cluster-api-machine-role: <infra> 8
      machine.openshift.io/cluster-api-machine-type: <infra> 9
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<infra>-<zone> 10
  spec:
    metadata:
      labels:
        node-role.kubernetes.io/infra: ""
    providerSpec:
      value:
        apiVersion: machine.openshift.io/v1
        credentialsSecret:
          name: alibabacloud-credentials
        imageId: <image_id> 11
        instanceType: <instance_type> 12
        kind: AlibabaCloudMachineProviderConfig
        ramRoleName: <infrastructure_id>-role-worker 13
        regionId: <region> 14
        resourceGroup: 15
          id: <resource_group_id>
          type: ID
        securityGroups:
          - tags: 16
              - Key: Name
                Value: <infrastructure_id>-sg-<role>
              type: Tags
        systemDisk: 17
          category: cloud_essd
          size: <disk_size>
        tag: 18
          - Key: kubernetes.io/cluster/<infrastructure_id>
            Value: owned
        userDataSecret:
          name: <user_data_secret> 19
        vSwitch:
          tags: 20
            - Key: Name
              Value: <infrastructure_id>-vswitch-<zone>
            type: Tags
        vpcId: ""
        zoneId: <zone> 21
      taints: 22
        - key: node-role.kubernetes.io/infra
          effect: NoSchedule

```

**1 5 7** クラスターのプロビジョニング時に設定したクラスター ID を基にするインフラストラクチャー ID を指定します。OpenShift CLI (**oc**) がインストールされている場合は、以下のコマンドを実行してインフラストラクチャー ID を取得できます。

-

```
$ oc get -o jsonpath='{.status.infrastructureName}' infrastructure cluster
```

- 2 3 8 9 **<infra>** ノードラベルを指定します。
- 4 6 10 インフラストラクチャー ID、**<infra>** ノードラベル、およびゾーンを指定します。
- 11 使用するイメージを指定します。クラスターに設定されている既存のデフォルトマシンのイメージを使用します。
- 12 マシンセットに使用するインスタンスタイプを指定します。
- 13 マシンセットに使用する RAM ロールの名前を指定します。インストーラーがデフォルトのマシンセットに入力する値を使用します。
- 14 マシンを配置するリージョンを指定します。
- 15 クラスターのリソースグループとタイプを指定します。インストーラーがデフォルトのマシンセットに入力する値を使用するか、別の値を指定できます。
- 16 18 20 マシンセットに使用するタグを指定します。少なくとも、この例に示されているタグを、クラスターに適切な値とともに含める必要があります。必要に応じて、インストーラーが作成するデフォルトのマシンセットに設定するタグなど、追加のタグを含めることができます。
- 17 ルートディスクのタイプとサイズを指定します。インストーラーが作成するデフォルトのマシンセットに入力する **category** 値を使用します。必要に応じて、**size** にギガバイト単位の別の値を指定します。
- 19 **openshift-machine-api** 名前空間にあるユーザーデータ YAML ファイルでシークレットの名前を指定します。インストーラーがデフォルトのマシンセットに入力する値を使用します。
- 21 マシンを配置するリージョン内のゾーンを指定します。リージョンがゾーンをサポートすることを確認してください。
- 22 ユーザーのワークロードが **infra** ノードにスケジュールされないようにテイントを指定します。



## 注記

インフラストラクチャーノードに **NoSchedule** テイントを追加すると、そのノードで実行されている既存の DNS Pod は **misscheduled** としてマークされます。 [misscheduled DNS Pod に対する容認の追加](#) または削除を行う必要があります。

## Alibaba Cloud 使用統計のマシンセットパラメーター

インストーラーが Alibaba Cloud クラスター用に作成するデフォルトのマシンセットには、Alibaba Cloud が使用統計を追跡するために内部的に使用する不要なタグ値が含まれています。これらのタグは、**spec.template.spec.provider Spec.value** リストの **securityGroups**、**tag**、および **vSwitch** パラメーターに入力されます。

追加のマシンをデプロイするマシンセットを作成するときは、必要な Kubernetes タグを含める必要があります。使用統計タグは、作成するマシンセットで指定されていない場合でも、デフォルトで適用されます。必要に応じて、追加のタグを含めることもできます。

次の YAML スニペットは、デフォルトのマシンセットのどのタグがオプションでどれが必須かを示しています。

**spec.template.spec.providerSpec.value.securityGroups のタグ**

```
spec:
  template:
    spec:
      providerSpec:
        value:
          securityGroups:
            - tags:
                - Key: kubernetes.io/cluster/<infrastructure_id> ❶
                  Value: owned
                - Key: GISV
                  Value: ocp
                - Key: sigs.k8s.io/cloud-provider-alibaba/origin ❷
                  Value: ocp
                - Key: Name
                  Value: <infrastructure_id>-sg-<role> ❸
            type: Tags
```

❶ ❷ オプション: このタグは、マシンセットで指定されていない場合でも適用されます。

❸ 必須。

ここでは、以下のようになります。

- **<infrastructure\_id>** は、クラスターのプロビジョニング時に設定したクラスター ID に基づくインフラストラクチャー ID です。
- **<role>** は、追加するノードラベルです。

**spec.template.spec.providerSpec.value.tag のタグ**

```
spec:
  template:
    spec:
      providerSpec:
        value:
          tag:
            - Key: kubernetes.io/cluster/<infrastructure_id> ❶
              Value: owned
            - Key: GISV ❷
              Value: ocp
            - Key: sigs.k8s.io/cloud-provider-alibaba/origin ❸
              Value: ocp
```

❷ ❸ オプション: このタグは、マシンセットで指定されていない場合でも適用されます。

❶ 必須。

**<infrastructure\_id>** は、クラスターのプロビジョニング時に設定したクラスター ID に基づくインフラストラクチャー ID です。

**spec.template.spec.providerSpec.value.vSwitch のタグ**

```
spec:
  template:
    spec:
      providerSpec:
        value:
          vSwitch:
            tags:
              - Key: kubernetes.io/cluster/<infrastructure_id> ❶
                Value: owned
              - Key: GISV ❷
                Value: ocp
              - Key: sigs.k8s.io/cloud-provider-alibaba/origin ❸
                Value: ocp
              - Key: Name
                Value: <infrastructure_id>-vswitch-<zone> ❹
            type: Tags
```

❶ ❷ ❸ オプション: このタグは、マシンセットで指定されていない場合でも適用されます。

❹ 必須。

ここでは、以下のようになります。

- **<infrastructure\_id>** は、クラスターのプロビジョニング時に設定したクラスター ID に基づくインフラストラクチャー ID です。
- **<zone>** は、マシンを配置するリージョン内のゾーンです。

#### 7.2.1.2. AWS 上のマシンセットカスタムリソースのサンプル YAML

このサンプル YAML は **us-east-1a** Amazon Web Services (AWS) ゾーンで実行され、**node-role.kubernetes.io/infra:""** というラベルが付けられたノードを作成するマシンセットを定義します。

このサンプルでは、**infrastructure\_id** はクラスターのプロビジョニング時に設定したクラスター ID に基づくインフラストラクチャー ID であり、**<infra>** は追加するノードラベルです。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> ❶
  name: <infrastructure_id>-infra-<zone> ❷
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> ❸
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra-<zone> ❹
  template:
    metadata:
      labels:
```

```

machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
machine.openshift.io/cluster-api-machine-role: <infra> 6
machine.openshift.io/cluster-api-machine-type: <infra> 7
machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra-<zone> 8
spec:
  metadata:
    labels:
      node-role.kubernetes.io/infra: "" 9
  taints: 10
  - key: node-role.kubernetes.io/infra
    effect: NoSchedule
  providerSpec:
    value:
      ami:
        id: ami-046fe691f52a953f9 11
      apiVersion: awsproviderconfig.openshift.io/v1beta1
      blockDevices:
        - ebs:
            iops: 0
            volumeSize: 120
            volumeType: gp2
      credentialsSecret:
        name: aws-cloud-credentials
      deviceIndex: 0
      iamInstanceProfile:
        id: <infrastructure_id>-worker-profile 12
      instanceType: m6i.large
      kind: AWSMachineProviderConfig
      placement:
        availabilityZone: <zone> 13
        region: <region> 14
      securityGroups:
        - filters:
            - name: tag:Name
              values:
                - <infrastructure_id>-worker-sg 15
      subnet:
        filters:
          - name: tag:Name
            values:
              - <infrastructure_id>-private-<zone> 16
      tags:
        - name: kubernetes.io/cluster/<infrastructure_id> 17
          value: owned
      userDataSecret:
        name: worker-user-data

```

1 3 5 12 15 17 クラスターのプロビジョニング時に設定したクラスター ID を基にするインフラストラクチャー ID を指定します。OpenShift CLI がインストールされている場合は、以下のコマンドを実行してインフラストラクチャー ID を取得できます。

```
$ oc get -o jsonpath='{.status.infrastructureName}' infrastructure cluster
```

2 4 8 インフラストラクチャー ID、<infra> ノードラベル、およびゾーンを指定します。

6 7 9 <infra> ノードラベルを指定します。

10 ユーザーのワークロードが infra ノードにスケジュールされないようにテイントを指定します。



#### 注記

インフラストラクチャーノードに **NoSchedule** テイントを追加すると、そのノードで実行されている既存の DNS Pod は **misscheduled** としてマークされます。[misscheduled DNS Pod に対する容認の追加](#) または削除を行う必要があります。

11 OpenShift Container Platform ノードの AWS ゾーンに有効な Red Hat Enterprise Linux CoreOS (RHCOS) AMI を指定します。AWS Marketplace イメージを使用する場合は、[AWS Marketplace](#) から OpenShift Container Platform サブスクリプションを完了して、リージョンの AMI ID を取得する必要があります。

```
$ oc -n openshift-machine-api \
  -o jsonpath='{.spec.template.spec.providerSpec.value.ami.id}' \
  get machineset/<infrastructure_id>-worker-<zone>
```

13 ゾーン (例: **us-east-1a**) を指定します。

14 リージョン (例: **us-east-1**) を指定します。

16 インフラストラクチャー ID とゾーンを指定します。

AWS で実行されるマシンセットは保証されていない [Spot インスタンス](#) をサポートします。AWS の On-Demand インスタンスと比較すると、Spot インスタンスをより低い価格で使用することでコストを節約できます。**MachineSet** YAML ファイルに **spotMarketOptions** を追加して [Spot Instances](#) を設定します。

### 7.2.1.3. Azure 上のマシンセットのカスタムリソースのサンプル YAML

このサンプル YAML は、リージョンの **1** Microsoft Azure ゾーンで実行され、**node-role.kubernetes.io/infra: ""** というラベルの付けられたノードを作成するマシンセットを定義します。

このサンプルでは、**infrastructure\_id** はクラスターのプロビジョニング時に設定したクラスター ID に基づくインフラストラクチャー ID であり、<infra> は追加するノードラベルです。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
    machine.openshift.io/cluster-api-machine-role: <infra> 2
    machine.openshift.io/cluster-api-machine-type: <infra> 3
  name: <infrastructure_id>-infra-<region> 4
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
```

```

machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra-<region> 6
template:
  metadata:
    creationTimestamp: null
    labels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 7
      machine.openshift.io/cluster-api-machine-role: <infra> 8
      machine.openshift.io/cluster-api-machine-type: <infra> 9
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra-<region> 10
  spec:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-machineset: <machineset_name> 11
        node-role.kubernetes.io/infra: "" 12
    providerSpec:
      value:
        apiVersion: azureproviderconfig.openshift.io/v1beta1
        credentialsSecret:
          name: azure-cloud-credentials
          namespace: openshift-machine-api
        image: 13
        offer: ""
        publisher: ""
        resourceID: /resourceGroups/<infrastructure_id>-
rg/providers/Microsoft.Compute/images/<infrastructure_id> 14
        sku: ""
        version: ""
        internalLoadBalancer: ""
        kind: AzureMachineProviderSpec
        location: <region> 15
        managedIdentity: <infrastructure_id>-identity 16
        metadata:
          creationTimestamp: null
        natRule: null
        networkResourceGroup: ""
        osDisk:
          diskSizeGB: 128
          managedDisk:
            storageAccountType: Premium_LRS
          osType: Linux
        publicIP: false
        publicLoadBalancer: ""
        resourceGroup: <infrastructure_id>-rg 17
        sshPrivateKey: ""
        sshPublicKey: ""
        tags:
          - name: <custom_tag_name> 18
            value: <custom_tag_value> 19
        subnet: <infrastructure_id>-<role>-subnet 20 21
        userDataSecret:
          name: worker-user-data 22
        vmSize: Standard_D4s_v3
        vnet: <infrastructure_id>-vnet 23

```

```

zone: "1" 24
taints: 25
- key: node-role.kubernetes.io/infra
  effect: NoSchedule

```

1 5 7 16 17 20 23 クラスターのプロビジョニング時に設定したクラスター ID を基にするインフラストラクチャー ID を指定します。OpenShift CLI がインストールされている場合は、以下のコマンドを実行してインフラストラクチャー ID を取得できます。

```
$ oc get -o jsonpath='{.status.infrastructureName}' infrastructure cluster
```

以下のコマンドを実行してサブネットを取得できます。

```
$ oc -n openshift-machine-api \
-o jsonpath='{.spec.template.spec.providerSpec.value.subnet}' \
get machineset/<infrastructure_id>-worker-centralus1
```

以下のコマンドを実行して vnet を取得できます。

```
$ oc -n openshift-machine-api \
-o jsonpath='{.spec.template.spec.providerSpec.value.vnet}' \
get machineset/<infrastructure_id>-worker-centralus1
```

2 3 8 9 12 21 22 <infra> ノードラベルを指定します。

4 6 10 インフラストラクチャー ID、<infra> ノードラベル、およびリージョンを指定します。

11 オプション: 可用性セットの使用を有効にするためにマシンセットの名前を指定します。この設定は、新規コンピュートマシンにのみ適用されます。

13 マシンセットのイメージの詳細を指定します。Azure Marketplace イメージを使用する場合は、Azure Marketplace イメージの選択を参照してください。

14 インスタンスタイプと互換性のあるイメージを指定します。インストールプログラムによって作成された Hyper-V 世代の V2 イメージには接尾辞 **-gen2** が付いていますが、V1 イメージには接尾辞のない同じ名前が付いています。

15 マシンを配置するリージョンを指定します。

24 マシンを配置するリージョン内のゾーンを指定します。リージョンがゾーンをサポートすることを確認してください。

18 19 オプション: マシンセットでカスタムタグを指定します。<custom\_tag\_name> フィールドにタグ名を指定し、対応するタグ値を <custom\_tag\_value> フィールドに指定します。

25 ユーザーのワークロードが infra ノードにスケジュールされないようにテイントを指定します。



## 注記

インフラストラクチャーノードに **NoSchedule** テイントを追加すると、そのノードで実行されている既存の DNS Pod は **misscheduled** としてマークされます。[misscheduled DNS Pod に対する容認の追加](#) または削除を行う必要があります。

Azure で実行されるマシンセットは、保証されていない [Spot 仮想マシン](#) をサポートします。Azure の標準仮想マシンと比較すると、Spot 仮想マシンをより低い価格で使用することでコストを節約できます。**MachineSet** YAML ファイルに **spotVMOptions** を追加することで、[Spot VM を設定](#) することができます。

## 関連情報

- [Azure Marketplace イメージの選択](#)

### 7.2.1.4. Azure Stack Hub 上のマシンセットのカスタムリソースのサンプル YAML

このサンプル YAML は、リージョンの **1** Microsoft Azure ゾーンで実行され、**node-role.kubernetes.io/infra: ""** というラベルの付けられたノードを作成するマシンセットを定義します。

このサンプルでは、**infrastructure\_id** はクラスターのプロビジョニング時に設定したクラスター ID に基づくインフラストラクチャー ID であり、**<infra>** は追加するノードラベルです。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> ❶
    machine.openshift.io/cluster-api-machine-role: <infra> ❷
    machine.openshift.io/cluster-api-machine-type: <infra> ❸
  name: <infrastructure_id>-infra-<region> ❹
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> ❺
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra-<region> ❻
  template:
    metadata:
      creationTimestamp: null
    labels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> ❼
      machine.openshift.io/cluster-api-machine-role: <infra> ❽
      machine.openshift.io/cluster-api-machine-type: <infra> ❾
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra-<region> ❿
    spec:
      metadata:
        creationTimestamp: null
      labels:
        node-role.kubernetes.io/infra: "" ㉑
      taints: ㉒
      - key: node-role.kubernetes.io/infra
        effect: NoSchedule
      providerSpec:
        value:
          apiVersion: machine.openshift.io/v1beta1
          availabilitySet: <availability_set> ㉓
          credentialsSecret:
            name: azure-cloud-credentials
```

```

namespace: openshift-machine-api
image:
  offer: ""
  publisher: ""
  resourceID: /resourceGroups/<infrastructure_id>-
rg/providers/Microsoft.Compute/images/<infrastructure_id> 14
  sku: ""
  version: ""
internalLoadBalancer: ""
kind: AzureMachineProviderSpec
location: <region> 15
managedIdentity: <infrastructure_id>-identity 16
metadata:
  creationTimestamp: null
natRule: null
networkResourceGroup: ""
osDisk:
  diskSizeGB: 128
  managedDisk:
    storageAccountType: Premium_LRS
  osType: Linux
publicIP: false
publicLoadBalancer: ""
resourceGroup: <infrastructure_id>-rg 17
sshPrivateKey: ""
sshPublicKey: ""
subnet: <infrastructure_id>-<role>-subnet 18 19
userDataSecret:
  name: worker-user-data 20
vmSize: Standard_DS4_v2
vnet: <infrastructure_id>-vnet 21
zone: "1" 22

```

1 5 7 14 16 17 18 21 クラスターのプロビジョニング時に設定したクラスター ID を基にするインフラストラクチャー ID を指定します。OpenShift CLI がインストールされている場合は、以下のコマンドを実行してインフラストラクチャー ID を取得できます。

```
$ oc get -o jsonpath='{.status.infrastructureName}{"\n"}' infrastructure cluster
```

以下のコマンドを実行してサブネットを取得できます。

```
$ oc -n openshift-machine-api \
  -o jsonpath='{.spec.template.spec.providerSpec.value.subnet}{"\n"}' \
  get machineset/<infrastructure_id>-worker-centralus1
```

以下のコマンドを実行して vnet を取得できます。

```
$ oc -n openshift-machine-api \
  -o jsonpath='{.spec.template.spec.providerSpec.value.vnet}{"\n"}' \
  get machineset/<infrastructure_id>-worker-centralus1
```

2 3 8 9 11 19 20 **<infra>** ノードラベルを指定します。

**4 6 10** インフラストラクチャー ID、**<infra>** ノードラベル、およびリージョンを指定します。

**12** ユーザーのワークロードが infra ノードにスケジュールされないようにテイントを指定します。



#### 注記

インフラストラクチャーノードに **NoSchedule** テイントを追加すると、そのノードで実行されている既存の DNS Pod は **misscheduled** としてマークされます。[misscheduled DNS Pod に対する容認の追加](#) または削除を行う必要があります。

**15** マシンを配置するリージョンを指定します。

**13** クラスターの可用性セットを指定します。

**22** マシンを配置するリージョン内のゾーンを指定します。リージョンがゾーンをサポートすることを確認してください。



#### 注記

Azure Stack Hub で実行されるマシンセットは、保証されていない Spot 仮想マシンをサポートしません。

### 7.2.1.5. IBMCloud 上のマシンセットカスタムリソースのサンプル YAML

このサンプル YAML は、リージョン内の指定された IBM Cloud ゾーンで実行され、**node-role.kubernetes.io/infra: ""** というラベルの付いたノードを作成するマシンセットを定義します。

このサンプルでは、**infrastructure\_id** はクラスターのプロビジョニング時に設定したクラスター ID に基づくインフラストラクチャー ID であり、**<infra>** は追加するノードラベルです。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
    machine.openshift.io/cluster-api-machine-role: <infra> 2
    machine.openshift.io/cluster-api-machine-type: <infra> 3
  name: <infrastructure_id>-<infra>-<region> 4
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<infra>-<region> 6
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 7
        machine.openshift.io/cluster-api-machine-role: <infra> 8
        machine.openshift.io/cluster-api-machine-type: <infra> 9
```

```

machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<infra>-<region> 10
spec:
  metadata:
    labels:
      node-role.kubernetes.io/infra: ""
  providerSpec:
    value:
      apiVersion: ibmcloudproviderconfig.openshift.io/v1beta1
      credentialsSecret:
        name: ibmcloud-credentials
      image: <infrastructure_id>-rhcos 11
      kind: IBMCloudMachineProviderSpec
      primaryNetworkInterface:
        securityGroups:
          - <infrastructure_id>-sg-cluster-wide
          - <infrastructure_id>-sg-openshift-net
        subnet: <infrastructure_id>-subnet-compute-<zone> 12
      profile: <instance_profile> 13
      region: <region> 14
      resourceGroup: <resource_group> 15
      userDataSecret:
        name: <role>-user-data 16
      vpc: <vpc_name> 17
      zone: <zone> 18
    taints: 19
      - key: node-role.kubernetes.io/infra
        effect: NoSchedule

```

- 1 5 7 クラスターのプロビジョニング時に設定したクラスター ID に基づくインフラストラクチャー ID。OpenShift CLI がインストールされている場合は、以下のコマンドを実行してインフラストラクチャー ID を取得できます。

```
$ oc get -o jsonpath='{.status.infrastructureName}' infrastructure cluster
```

- 2 3 8 9 16 **<infra>** ノードラベル。

- 4 6 10 インフラストラクチャー ID、 **<infra>** ノードラベル、およびリージョン。

- 11 クラスターのインストールに使用されたカスタム Red Hat Enterprise Linux CoreOS (RHCOS) イメージ。

- 12 マシンを配置するためのリージョン内のインフラストラクチャー ID とゾーン。リージョンがゾーンをサポートすることを確認してください。

- 13 [IBM Cloud インスタンスプロファイル](#) を指定します。

- 14 マシンを配置するリージョンを指定します。

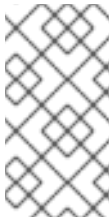
- 15 マシンリソースが配置されるリソースグループ。これは、インストール時に指定された既存のリソースグループ、またはインフラストラクチャー ID に基づいて名前が付けられたインストーラーによって作成されたリソースグループのいずれかです。

- 17 VPC 名。

- 18 マシンを配置するリージョン内のゾーンを指定します。リージョンがゾーンをサポートすることを確認してください。

確認してください。

- 19 ユーザーのワークロードがインフラノードでスケジュールされないようにするための汚染。



### 注記

インフラストラクチャーノードに **NoSchedule** テイントを追加すると、そのノードで実行されている既存の DNS Pod は **misscheduled** としてマークされます。 **misscheduled DNS Pod に対する容認の追加** または削除を行う必要があります。

#### 7.2.1.6. GCP 上のマシンセットのカスタムリソースのサンプル YAML

このサンプル YAML は、Google Cloud Platform (GCP) で実行され、**node-role.kubernetes.io/infra:** "" というラベルが付けられたノードを作成するマシンセットを定義します。

このサンプルでは、**infrastructure\_id** はクラスターのプロビジョニング時に設定したクラスター ID に基づくインフラストラクチャー ID であり、**<infra>** は追加するノードラベルです。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
  name: <infrastructure_id>-w-a
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-w-a
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>
        machine.openshift.io/cluster-api-machine-role: <infra> 2
        machine.openshift.io/cluster-api-machine-type: <infra>
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-w-a
    spec:
      metadata:
        labels:
          node-role.kubernetes.io/infra: ""
      providerSpec:
        value:
          apiVersion: gcpprovider.openshift.io/v1beta1
          canIPForward: false
          credentialsSecret:
            name: gcp-cloud-credentials
          deletionProtection: false
          disks:
            - autoDelete: true
              boot: true
```

```

    image: <path_to_image> ❸
    labels: null
    sizeGb: 128
    type: pd-ssd
  gcpMetadata: ❹
  - key: <custom_metadata_key>
    value: <custom_metadata_value>
  kind: GCPMachineProviderSpec
  machineType: n1-standard-4
  metadata:
    creationTimestamp: null
  networkInterfaces:
  - network: <infrastructure_id>-network
    subnetwork: <infrastructure_id>-worker-subnet
  projectID: <project_name> ❺
  region: us-central1
  serviceAccounts:
  - email: <infrastructure_id>-w@<project_name>.iam.gserviceaccount.com
    scopes:
    - https://www.googleapis.com/auth/cloud-platform
  tags:
  - <infrastructure_id>-worker
  userDataSecret:
    name: worker-user-data
  zone: us-central1-a
  taints: ❻
  - key: node-role.kubernetes.io/infra
    effect: NoSchedule

```

- ❶ **<infrastructure\_id>** は、クラスターのプロビジョニング時に設定したクラスター ID に基づくインフラストラクチャー ID を指定します。OpenShift CLI がインストールされている場合は、以下のコマンドを実行してインフラストラクチャー ID を取得できます。

```
$ oc get -o jsonpath='{.status.infrastructureName}' infrastructure cluster
```

- ❷ **<infra>** には、**<infra>** ノードラベルを指定します。

- ❸ 現在のマシンセットで使用するイメージへのパスを指定します。OpenShift CLI がインストールされている場合は、以下のコマンドを実行してイメージへのパスを取得できます。

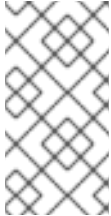
```
$ oc -n openshift-machine-api \
  -o jsonpath='{.spec.template.spec.providerSpec.value.disks[0].image}' \
  get machineset/<infrastructure_id>-worker-a
```

GCP Marketplace イメージを使用するには、使用するオファ―を指定します。

- OpenShift Container Platform: <https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-ocp-48-x86-64-202210040145>
- OpenShift Platform Plus: <https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-opp-48-x86-64-202206140145>

- OpenShift Kubernetes Engine:  
<https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-oke-48-x86-64-202206140145>

- 4 オプション: **key:value** のペアの形式でカスタムメタデータを指定します。ユースケースの例については、[カスタムメタデータの設定](#) について GCP のドキュメントを参照してください。
- 5 **<project\_name>** には、クラスターに使用する GCP プロジェクトの名前を指定します。
- 6 ユーザーのワークロードが infra ノードにスケジュールされないようにテイントを指定します。



### 注記

インフラストラクチャノードに **NoSchedule** テイントを追加すると、そのノードで実行されている既存の DNS Pod は **misscheduled** としてマークされます。[misscheduled DNS Pod に対する容認の追加](#) または削除を行う必要があります。

GCP で実行されるマシンセットは、保証されていない [プリエンプション可能な仮想マシンインスタンス](#) をサポートします。GCP の通常のインスタンスと比較して、プリエンプション可能な仮想マシンインスタンスをより低い価格で使用することでコストを節約できます。**MachineSet** YAML ファイルに **preemptible** を追加することで、[プリエンプション可能な仮想マシンインスタンスを設定](#) することができます。

#### 7.2.1.7. Nutanix 上のマシンセットのカスタムリソースのサンプル YAML

このサンプル YAML は、node-role.kubernetes.io **node-role.kubernetes.io/infra: ""** でラベル付けされたノードを作成する Nutanix マシンセットを定義します。

このサンプルでは、**infrastructure\_id** はクラスターのプロビジョニング時に設定したクラスター ID に基づくインフラストラクチャー ID であり、**<infra>** は追加するノードラベルです。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
    machine.openshift.io/cluster-api-machine-role: <infra> 2
    machine.openshift.io/cluster-api-machine-type: <infra> 3
  name: <infrastructure_id>-<infra>-<zone> 4
  namespace: openshift-machine-api
  annotations: 5
    machine.openshift.io/memoryMb: "16384"
    machine.openshift.io/vCPU: "4"
spec:
  replicas: 3
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 6
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<infra>-<zone> 7
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 8
```

```

machine.openshift.io/cluster-api-machine-role: <infra> 9
machine.openshift.io/cluster-api-machine-type: <infra> 10
machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<infra>-<zone> 11
spec:
  metadata:
    labels:
      node-role.kubernetes.io/infra: ""
  providerSpec:
    value:
      apiVersion: machine.openshift.io/v1
      cluster:
        type: uuid
        uuid: <cluster_uuid>
      credentialsSecret:
        name: nutanix-creds-secret
      image:
        name: <infrastructure_id>-rhcos 12
        type: name
      kind: NutanixMachineProviderConfig
      memorySize: 16Gi 13
      subnets:
        - type: uuid
          uuid: <subnet_uuid>
      systemDiskSize: 120Gi 14
      userDataSecret:
        name: <user_data_secret> 15
      vcpuSockets: 4 16
      vcpusPerSocket: 1 17
    taints: 18
      - key: node-role.kubernetes.io/infra
        effect: NoSchedule

```

- 1 6 8 クラスターのプロビジョニング時に設定したクラスター ID を基にするインフラストラクチャー ID を指定します。OpenShift CLI (**oc**) がインストールされている場合は、以下のコマンドを実行してインフラストラクチャー ID を取得できます。

```
$ oc get -o jsonpath='{.status.infrastructureName}' infrastructure cluster
```

- 2 3 9 10 **<infra>** ノードラベルを指定します。

- 4 7 11 インフラストラクチャー ID、**<infra>** ノードラベル、およびゾーンを指定します。

- 5 クラスターオートスケーラーのアノテーション。

- 12 使用するイメージを指定します。クラスターに設定されている既存のデフォルトマシンのイメージを使用します。

- 13 クラスターのメモリー量を Gi で指定します。

- 14 システムディスクのサイズを Gi で指定します。

- 15 **openshift-machine-api** 名前空間にあるユーザーデータ YAML ファイルでシークレットの名前を指定します。インストーラーがデフォルトのマシンセットに入力する値を使用します。

- 16 vCPU ソケットの数を指定します。
- 17 ソケットあたりの vCPU の数を指定します。
- 18 ユーザーのワークロードが infra ノードにスケジュールされないようにテイントを指定します。



### 注記

インフラストラクチャノードに **NoSchedule** テイントを追加すると、そのノードで実行されている既存の DNS Pod は **misscheduled** としてマークされます。 **misscheduled DNS Pod に対する容認の追加** または削除を行う必要があります。

#### 7.2.1.8. RHOSP 上のマシンセットのカスタムリソースのサンプル YAML

このサンプル YAML は、Red Hat OpenStack Platform (RHOSP) で実行され、**node-role.kubernetes.io/infra: ""** というラベルが付けられたノードを作成するマシンセットを定義します。

このサンプルでは、**infrastructure\_id** はクラスターのプロビジョニング時に設定したクラスター ID に基づくインフラストラクチャ ID であり、**<infra>** は追加するノードラベルです。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
    machine.openshift.io/cluster-api-machine-role: <infra> 2
    machine.openshift.io/cluster-api-machine-type: <infra> 3
  name: <infrastructure_id>-infra 4
  namespace: openshift-machine-api
spec:
  replicas: <number_of_replicas>
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra 6
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 7
        machine.openshift.io/cluster-api-machine-role: <infra> 8
        machine.openshift.io/cluster-api-machine-type: <infra> 9
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra 10
    spec:
      metadata:
        creationTimestamp: null
      labels:
        node-role.kubernetes.io/infra: ""
      taints: 11
      - key: node-role.kubernetes.io/infra
        effect: NoSchedule
      providerSpec:
```

```

value:
  apiVersion: openstackproviderconfig.openshift.io/v1alpha1
  cloudName: openstack
  cloudsSecret:
    name: openstack-cloud-credentials
    namespace: openshift-machine-api
  flavor: <nova_flavor>
  image: <glance_image_name_or_location>
  serverGroupID: <optional_UUID_of_server_group> 12
  kind: OpenstackProviderSpec
  networks: 13
  - filter: {}
    subnets:
      - filter:
          name: <subnet_name>
          tags: openshiftClusterID=<infrastructure_id> 14
  primarySubnet: <rhosp_subnet_UUID> 15
  securityGroups:
  - filter: {}
    name: <infrastructure_id>-worker 16
  serverMetadata:
    Name: <infrastructure_id>-worker 17
    openshiftClusterID: <infrastructure_id> 18
  tags:
  - openshiftClusterID=<infrastructure_id> 19
  trunk: true
  userDataSecret:
    name: worker-user-data 20
  availabilityZone: <optional_openstack_availability_zone>

```

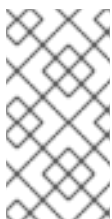
1 5 7 14 16 17 18 19 クラスターのプロビジョニング時に設定したクラスター ID を基にするインフラストラクチャー ID を指定します。OpenShift CLI がインストールされている場合は、以下のコマンドを実行してインフラストラクチャー ID を取得できます。

```
$ oc get -o jsonpath='{.status.infrastructureName}' infrastructure cluster
```

2 3 8 9 20 <infra> ノードラベルを指定します。

4 6 10 インフラストラクチャー ID および <infra> ノードラベルを指定します。

11 ユーザーのワークロードが infra ノードにスケジュールされないようにティントを指定します。



### 注記

インフラストラクチャーノードに **NoSchedule** テイントを追加すると、そのノードで実行されている既存の DNS Pod は **misscheduled** としてマークされます。 **misscheduled DNS Pod に対する容認の追加** または削除を行う必要があります。

12 MachineSet のサーバーグループポリシーを設定するには、[サーバーグループの作成](#) から返された値を入力します。ほとんどのデプロイメントでは、**anti-affinity** または **soft-anti-affinity** が推奨されます。

13 複数ネットワークへのデプロイメントに必要です。複数ネットワークにデプロイする場合、この一

覧には、**primarySubnet** が の値として使用されるネットワークが含まれる必要があります。

- 15 ノードのエンドポイントを公開する RHOSP サブネットを指定します。通常、これは **install-config.yaml** ファイルの **machinesSubnet** の値として使用される同じサブネットです。

### 7.2.1.9. RHV 上のマシンセットのカスタムリソースのサンプル YAML

このサンプル YAML は、RHV で実行され、**node-role.kubernetes.io/<node\_role>: ""** というラベルが付けられたノードを作成するマシンセットを定義します。

このサンプルでは、**<infrastructure\_id>** はクラスターのプロビジョニング時に設定したクラスター ID に基づくインフラストラクチャー ID であり、**<role>** は追加するノードラベルです。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
    machine.openshift.io/cluster-api-machine-role: <role> 2
    machine.openshift.io/cluster-api-machine-type: <role> 3
  name: <infrastructure_id>-<role> 4
  namespace: openshift-machine-api
spec:
  replicas: <number_of_replicas> 5
  Selector: 6
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 7
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role> 8
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 9
        machine.openshift.io/cluster-api-machine-role: <role> 10
        machine.openshift.io/cluster-api-machine-type: <role> 11
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role> 12
    spec:
      metadata:
        labels:
          node-role.kubernetes.io/<role>: "" 13
      providerSpec:
        value:
          apiVersion: ovirtproviderconfig.machine.openshift.io/v1beta1
          cluster_id: <ovirt_cluster_id> 14
          template_name: <ovirt_template_name> 15
          sparse: <boolean_value> 16
          format: <raw_or_cow> 17
          cpu: 18
            sockets: <number_of_sockets> 19
            cores: <number_of_cores> 20
            threads: <number_of_threads> 21
          memory_mb: <memory_size> 22
          guaranteed_memory_mb: <memory_size> 23
```

```

os_disk: 24
size_gb: <disk_size> 25
storage_domain_id: <storage_domain_UUID> 26
network_interfaces: 27
  vnic_profile_id: <vnic_profile_id> 28
credentialsSecret:
  name: ovirt-credentials 29
kind: OvirtMachineProviderSpec
type: <workload_type> 30
auto_pinning_policy: <auto_pinning_policy> 31
hugepages: <hugepages> 32
affinityGroupsNames:
  - compute 33
userDataSecret:
  name: worker-user-data

```

- 1 7 9 クラスターのプロビジョニング時に設定したクラスター ID を基にするインフラストラクチャー ID を指定します。OpenShift CLI (**oc**) がインストールされている場合は、以下のコマンドを実行してインフラストラクチャー ID を取得できます。

```
$ oc get -o jsonpath='{.status.infrastructureName}{"\n"}' infrastructure cluster
```

- 2 3 10 11 13 追加するノードラベルを指定します。

- 4 8 12 インフラストラクチャー ID およびノードラベルを指定します。これら 2 つの文字列は 35 文字を超えることができません。

- 5 作成するマシンの数を指定します。

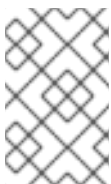
- 6 マシンのセクター。

- 14 この仮想マシンインスタンスが属する RHV クラスターの UUID を指定します。

- 15 マシンの作成に使用する RHV 仮想マシンテンプレートを指定します。

- 16 このオプションを **false** に設定すると、ディスクの事前割り当てが有効になります。デフォルトは **true** です。**format** を **raw** に設定して **sparse** を **true** に設定することは、ブロックストレージドメインでは使用できません。**raw** 形式は、仮想ディスク全体を基盤となる物理ディスクに書き込みます。

- 17 **cow** または **raw** に設定できます。デフォルトは **cow** です。**cow** のフォーマットは仮想マシン用に最適化されています。



#### 注記

ファイルストレージドメインにディスクを事前に割り当てると、ファイルにゼロが書き込まれます。基盤となるストレージによっては、実際にはディスクが事前に割り当てられない場合があります。

- 18 オプション: CPU フィールドには、ソケット、コア、スレッドを含む CPU の設定が含まれます。

- 19 オプション: 仮想マシンのソケット数を指定します。

- 20 オプション: ソケットあたりのコア数を指定します。

- 21 オプション: コアあたりのスレッド数を指定します。
- 22 オプション: 仮想マシンのメモリーサイズを MiB 単位で指定します。
- 23 オプション: 仮想マシンの保証されたメモリーのサイズを MiB で指定します。これは、バルーニングメカニズムによって排出されないことが保証されているメモリーの量です。詳細は、[Memory Ballooning](#) と [Optimization Settings Explained](#) を参照してください。



#### 注記

RHV 4.4.8 より前のバージョンを使用している場合は、[Red Hat Virtualization クラスタでの OpenShift の保証されたメモリー要件](#)を参照してください。

- 24 オプション: ノードのルートディスク。
- 25 オプション: ブート可能なディスクのサイズを GiB 単位で指定します。
- 26 オプション: コンピュートノードのディスクのストレージドメインの UUID を指定します。何も指定されていない場合、コンピュートノードはコントロールノードと同じストレージドメインに作成されます。(デフォルト)
- 27 オプション: 仮想マシンのネットワークインターフェイスの一覧。このパラメーターを含めると、OpenShift Container Platform はテンプレートからすべてのネットワークインターフェイスを破棄し、新規ネットワークインターフェイスを作成します。
- 28 オプション: vNIC プロファイル ID を指定します。
- 29 RHV クレデンシャルを保持するシークレットオブジェクトの名前を指定します。
- 30 オプション: インスタンスが最適化されるワークロードタイプを指定します。この値は **RHV VM** パラメーターに影響します。サポートされる値: **desktop**、**server** (デフォルト)、**high\_performance** です。**high\_performance** は、VM のパフォーマンスを向上させます。制限があります。たとえば、グラフィカルコンソールで VM にアクセスすることはできません。詳細は、[Virtual Machine Management Guide](#)の [ハイパフォーマンス仮想マシン、テンプレート、およびプールの設定](#)を参照してください。
- 31 オプション: AutoPinningPolicy は、このインスタンスのホストへのピンニングを含む、CPU と NUMA 設定を自動的に設定するポリシーを定義します。サポートされる値は、**none**、**resize\_and\_pin** です。詳細は、[Virtual Machine Management Guide](#)の [Setting NUMA Nodes](#)を参照してください。
- 32 オプション: hugepages は、仮想マシンで hugepage を定義するためのサイズ (KiB) です。対応している値は **2048** および **1048576** です。詳細は、[Virtual Machine Management Guide](#)の [Configuring Huge Pages](#)を参照してください。
- 33 オプション: 仮想マシンに適用されるアフィニティグループ名のリスト。アフィニティグループは oVirt に存在する必要があります。



#### 注記

RHV は仮想マシンの作成時にテンプレートを使用するため、任意のパラメーターの値を指定しない場合、RHV はテンプレートに指定されるパラメーターの値を使用します。

### 7.2.1.10. vSphere 上のマシンセットのカスタムリソースのサンプル YAML

このサンプル YAML は、VMware vSphere で実行され、**node-role.kubernetes.io/infra: ""** というラベルが付けられたノードを作成するマシンセットを定義します。

このサンプルでは、**infrastructure\_id** はクラスターのプロビジョニング時に設定したクラスター ID に基づくインフラストラクチャー ID であり、**<infra>** は追加するノードラベルです。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  creationTimestamp: null
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> ❶
  name: <infrastructure_id>-infra ❷
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> ❸
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra ❹
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> ❺
        machine.openshift.io/cluster-api-machine-role: <infra> ❻
        machine.openshift.io/cluster-api-machine-type: <infra> ❼
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra ❽
    spec:
      metadata:
        creationTimestamp: null
        labels:
          node-role.kubernetes.io/infra: "" ❾
      taints: ❿
      - key: node-role.kubernetes.io/infra
        effect: NoSchedule
      providerSpec:
        value:
          apiVersion: vsphereprovider.openshift.io/v1beta1
          credentialsSecret:
            name: vsphere-cloud-credentials
          diskGiB: 120
          kind: VSphereMachineProviderSpec
          memoryMiB: 8192
          metadata:
            creationTimestamp: null
          network:
            devices:
              - networkName: "<vm_network_name>" ⓫
          numCPUs: 4
          numCoresPerSocket: 1
          snapshot: ""
          template: <vm_template_name> ⓬
          userDataSecret:
```

```

name: worker-user-data
workspace:
  datacenter: <vcenter_datacenter_name> 13
  datastore: <vcenter_datastore_name> 14
  folder: <vcenter_vm_folder_path> 15
  resourcepool: <vsphere_resource_pool> 16
  server: <vcenter_server_ip> 17

```

- 1 3 5 クラスターのプロビジョニング時に設定したクラスター ID を基にするインフラストラクチャー ID を指定します。OpenShift CLI (**oc**) がインストールされている場合は、以下のコマンドを実行してインフラストラクチャー ID を取得できます。

```
$ oc get -o jsonpath='{.status.infrastructureName}' infrastructure cluster
```

- 2 4 8 インフラストラクチャー ID および **<infra>** ノードラベルを指定します。

- 6 7 9 **<infra>** ノードラベルを指定します。

- 10 ユーザーのワークロードが infra ノードにスケジュールされないようにテイントを指定します。



#### 注記

インフラストラクチャーノードに **NoSchedule** テイントを追加すると、そのノードで実行されている既存の DNS Pod は **misscheduled** としてマークされます。[misscheduled DNS Pod に対する容認の追加](#) または削除を行う必要があります。

- 11 マシンセットをデプロイする vSphere 仮想マシンネットワークを指定します。この仮想マシンネットワークは、他のコンピューティングマシンがクラスター内に存在する場所である必要があります。
- 12 **user-5ddjd-rhcos** などの使用する vSphere 仮想マシンテンプレートを指定します。
- 13 マシンセットをデプロイする vCenter Datacenter を指定します。
- 14 マシンセットをデプロイする vCenter Datastore を指定します。
- 15 **/dc1/vm/user-inst-5ddjd** などの vCenter の vSphere 仮想マシンフォルダーへのパスを指定します。
- 16 仮想マシンの vSphere リソースプールを指定します。
- 17 vCenter サーバーの IP または完全修飾ドメイン名を指定します。

### 7.2.2. マシンセットの作成

インストールプログラムによって作成されるコンピュートセットセットに加えて、独自のマシンセットを作成して、選択した特定のワークロードのマシンコンピューティングリソースを動的に管理できます。

#### 前提条件

- OpenShift Container Platform クラスターをデプロイすること。

- OpenShift CLI (**oc**) がインストールされている。
- **cluster-admin** パーミッションを持つユーザーとして、**oc** にログインする。

## 手順

1. 説明されているようにマシンセット カスタムリソース (CR) サンプルを含む新規 YAML ファイルを作成し、そのファイルに **<file\_name>.yaml** という名前を付けます。  
**<clusterID>** および **<role>** パラメーターの値を設定していることを確認します。
2. オプション: 特定のフィールドに設定する値がわからない場合は、クラスターから既存のコンピュートマシンセットを確認できます。
  - a. クラスター内のコンピュートマシンセットをリスト表示するには、次のコマンドを実行します。

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

## 出力例

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

- b. 特定のコンピュートマシンセットカスタムリソース(CR)の値を表示するには、以下のコマンドを実行します。

```
$ oc get machineset <machineset_name> \
-n openshift-machine-api -o yaml
```

## 出力例

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
    name: <infrastructure_id>-<role> 2
    namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>
        machine.openshift.io/cluster-api-machine-role: <role>
```

```
machine.openshift.io/cluster-api-machine-type: <role>
machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
spec:
  providerSpec: ❸
  ...
```

❶ クラスターインフラストラクチャー ID。

❷ デフォルトのノードラベル。



#### 注記

ユーザーがプロビジョニングしたインフラストラクチャーを持つクラスターの場合、コンピュートマシンセットは **worker** および **infra** タイプのマシンのみを作成できます。

❸ コンピュートマシンセット CR の **<providerSpec>** セクションの値は、プラットフォーム固有です。CR の **<providerSpec>** パラメーターの詳細については、プロバイダーのサンプルコンピュートマシンセット CR 設定を参照してください。

3. 次のコマンドを実行して **MachineSet** CR を作成します。

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

#### 検証

- 次のコマンドを実行して、コンピュートマシンセットのリストを表示します。

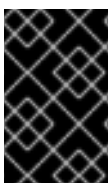
```
$ oc get machineset -n openshift-machine-api
```

#### 出力例

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
agl030519-vplxk-infra-us-east-1a	1	1	1	1	11m
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

新規のマシンセットが利用可能な場合、**DESIRED** および **CURRENT** の値は一致します。マシンセットが利用可能でない場合、数分待機してからコマンドを再度実行します。

### 7.2.3. 専用インフラストラクチャーノードの作成



#### 重要

インストーラーでプロビジョニングされるインフラストラクチャー環境またはコントロールプレーンノードがマシン API によって管理されているクラスターについて、**Creating infrastructure machine set** を参照してください。

クラスターの要件により、インフラストラクチャー (**infra** ノードとも呼ばれる) がプロビジョニングされます。インストーラーは、コントロールプレーンノードとワーカーノードのプロビジョニングのみを提供します。ワーカーノードは、ラベル付けによって、インフラストラクチャーノードまたはアプリケーション (**app** と呼ばれる) として指定できます。

## 手順

1. アプリケーションノードとして機能させるワーカーノードにラベルを追加します。

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

2. インフラストラクチャーノードとして機能する必要があるワーカーノードにラベルを追加します。

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

3. 該当するノードに **infra** ロールおよび **app** ロールがあるかどうかを確認します。

```
$ oc get nodes
```

4. デフォルトのクラスタースコープのセレクトラを作成するには、以下を実行します。デフォルトのノードセレクトラはすべての namespace で作成された Pod に適用されます。これにより、Pod の既存のノードセレクトラとの交差が作成され、Pod のセレクトラをさらに制限します。

### 重要

デフォルトのノードセレクトラのキーが Pod のラベルのキーと競合する場合、デフォルトのノードセレクトラは適用されません。

ただし、Pod がスケジューリング対象外になる可能性のあるデフォルトノードセレクトラを設定しないでください。たとえば、Pod のラベルが **node-role.kubernetes.io/master=""** などの別のノードロールに設定されている場合、デフォルトのノードセレクトラを **node-role.kubernetes.io/infra=""** などの特定のノードロールに設定すると、Pod がスケジューリング不能になる可能性があります。このため、デフォルトのノードセレクトラを特定のノードロールに設定する際には注意が必要です。

または、プロジェクトノードセレクトラを使用して、クラスター全体でのノードセレクトラの競合を避けることができます。

- a. **Scheduler** オブジェクトを編集します。

```
$ oc edit scheduler cluster
```

- b. 適切なノードセレクトラと共に **defaultNodeSelector** フィールドを追加します。

```
apiVersion: config.openshift.io/v1
kind: Scheduler
metadata:
  name: cluster
```

```
spec:
  defaultNodeSelector: topology.kubernetes.io/region=us-east-1 ❶
# ...
```

- ❶ このサンプルノードセクターは、デフォルトで **us-east-1** リージョンのノードに Pod をデプロイします。

c. 変更を適用するためにファイルを保存します。

これで、インフラストラクチャリソースを新しくラベル付けされた **infra** ノードに移動できます。

## 関連情報

- [リソースのインフラストラクチャマシンセットへの移行](#)

### 7.2.4. インフラストラクチャマシンのマシン設定プール作成

インフラストラクチャマシンに専用の設定が必要な場合は、infra プールを作成する必要があります。

## 手順

1. 特定のラベルを持つ infra ノードとして割り当てるノードに、ラベルを追加します。

```
$ oc label node <node_name> <label>
```

```
$ oc label node ci-ln-n8mqwr2-f76d1-xscn2-worker-c-6fmtx node-role.kubernetes.io/infra=
```

2. ワーカーロールとカスタムロールの両方をマシン設定セクターとして含まれるマシン設定プールを作成します。

```
$ cat infra.mcp.yaml
```

## 出力例

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

- ❶ ワーカーロールおよびカスタムロールを追加します。

- ❷ ノードに追加したラベルを **nodeSelector** として追加します。



## 注記

カスタムマシン設定プールは、ワーカープールからマシン設定を継承します。カスタムプールは、ワーカープールのターゲット設定を使用しますが、カスタムプールのみをターゲットに設定する変更をデプロイする機能を追加します。カスタムプールはワーカープールから設定を継承するため、ワーカープールへの変更もカスタムプールに適用されます。

3. YAML ファイルを用意した後に、マシン設定プールを作成できます。

```
$ oc create -f infra.mcp.yaml
```

4. マシン設定をチェックして、インフラストラクチャー設定が正常にレンダリングされていることを確認します。

```
$ oc get machineconfig
```

## 出力例

NAME	GENERATEDBYCONTROLLER
IGNITIONVERSION CREATED	
00-master	365c1cfd14de5b0e3b85e0fc815b0060f36ab955
3.2.0 31d	
00-worker	365c1cfd14de5b0e3b85e0fc815b0060f36ab955
3.2.0 31d	
01-master-container-runtime	
365c1cfd14de5b0e3b85e0fc815b0060f36ab955 3.2.0 31d	
01-master-kubelet	365c1cfd14de5b0e3b85e0fc815b0060f36ab955
3.2.0 31d	
01-worker-container-runtime	
365c1cfd14de5b0e3b85e0fc815b0060f36ab955 3.2.0 31d	
01-worker-kubelet	365c1cfd14de5b0e3b85e0fc815b0060f36ab955
3.2.0 31d	
99-master-1ae2a1e0-a115-11e9-8f14-005056899d54-registries	
365c1cfd14de5b0e3b85e0fc815b0060f36ab955 3.2.0 31d	
99-master-ssh	3.2.0 31d
99-worker-1ae64748-a115-11e9-8f14-005056899d54-registries	
365c1cfd14de5b0e3b85e0fc815b0060f36ab955 3.2.0 31d	
99-worker-ssh	3.2.0 31d
rendered-infra-4e48906dca84ee702959c71a53ee80e7	
365c1cfd14de5b0e3b85e0fc815b0060f36ab955 3.2.0 23m	
rendered-master-072d4b2da7f88162636902b074e9e28e	
5b6fb8349a29735e48446d435962dec4547d3090 3.2.0 31d	
rendered-master-3e88ec72aed3886dec061df60d16d1af	
02c07496ba0417b3e12b78fb32baf6293d314f79 3.2.0 31d	
rendered-master-419bee7de96134963a15fdf9dd473b25	
365c1cfd14de5b0e3b85e0fc815b0060f36ab955 3.2.0 17d	
rendered-master-53f5c91c7661708adce18739cc0f40fb	
365c1cfd14de5b0e3b85e0fc815b0060f36ab955 3.2.0 13d	
rendered-master-a6a357ec18e5bce7f5ac426fc7c5ffcd	
365c1cfd14de5b0e3b85e0fc815b0060f36ab955 3.2.0 7d3h	
rendered-master-dc7f874ec77fc4b969674204332da037	
5b6fb8349a29735e48446d435962dec4547d3090 3.2.0 31d	
rendered-worker-1a75960c52ad18ff5dfa6674eb7e533d	
5b6fb8349a29735e48446d435962dec4547d3090 3.2.0 31d	

```

rendered-worker-2640531be11ba43c61d72e82dc634ce6
5b6fb8349a29735e48446d435962dec4547d3090 3.2.0 31d
rendered-worker-4e48906dca84ee702959c71a53ee80e7
365c1cfd14de5b0e3b85e0fc815b0060f36ab955 3.2.0 7d3h
rendered-worker-4f110718fe88e5f349987854a1147755
365c1cfd14de5b0e3b85e0fc815b0060f36ab955 3.2.0 17d
rendered-worker-afc758e194d6188677eb837842d3b379
02c07496ba0417b3e12b78fb32baf6293d314f79 3.2.0 31d
rendered-worker-daa08cc1e8f5fcdeba24de60cd955cc3
365c1cfd14de5b0e3b85e0fc815b0060f36ab955 3.2.0 13d

```

新規のマシン設定には、接頭辞 **rendered-infra-\*** が表示されるはずです。

5. オプション: カスタムプールへの変更をデプロイするには、**infra** などのラベルとしてカスタムプール名を使用するマシン設定を作成します。これは必須ではありませんが、説明の目的でのみ表示されていることに注意してください。これにより、インフラストラクチャーノードのみに固有のカスタム設定を適用できます。



### 注記

新規マシン設定プールの作成後に、MCO はそのプールに新たにレンダリングされた設定を生成し、そのプールに関連付けられたノードは再起動して、新規設定を適用します。

- a. マシン設定を作成します。

```
$ cat infra.mc.yaml
```

### 出力例

```

apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: 51-infra
  labels:
    machineconfiguration.openshift.io/role: infra ❶
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
        - path: /etc/infratest
          mode: 0644
          contents:
            source: data:,infra

```

- ❶ ノードに追加したラベルを **nodeSelector** として追加します。

- b. マシン設定を infra のラベルが付いたノードに適用します。

```
$ oc create -f infra.mc.yaml
```

6. 新規のマシン設定プールが利用可能であることを確認します。

```
$ oc get mcp
```

### 出力例

```
NAME          CONFIG                                UPDATED  UPDATING  DEGRADED
MACHINECOUNT READYMACHINECOUNT UPDATEDMACHINECOUNT
DEGRADEDMACHINECOUNT AGE
infra  rendered-infra-60e35c2e99f42d976e084fa94da4d0fc  True    False    False    1
1          1          0          4m20s
master  rendered-master-9360fdb895d4c131c7c4bebbae099c90  True    False    False    3
3          3          3          0          91m
worker  rendered-worker-60e35c2e99f42d976e084fa94da4d0fc  True    False    False    2
2          2          2          0          91m
```

この例では、ワーカーノードが `infra` ノードに変更されました。

### 関連情報

- カスタムプールでインフラマシンをグループ化する方法に関する詳細は、[Node configuration management with machine config pools](#) を参照してください。

## 7.3. マシンセットリソースのインフラストラクチャーノードへの割り当て

インフラストラクチャーマシンセットの作成後、**worker** および **infra** ロールが新規の `infra` ノードに適用されます。**infra** ロールが適用されるノードは、**worker** ロールも適用されている場合でも、環境を実行するために必要なサブスクリプションの合計数にはカウントされません。

ただし、`infra` ノードがワーカーとして割り当てられると、ユーザーのワークロードが誤って `infra` ノードに割り当てられる可能性があります。これを回避するには、ティントを、制御する必要のある Pod の `infra` ノードおよび容認に適用できます。

### 7.3.1. テイントおよび容認を使用したインフラストラクチャーノードのワークロードのバインディング

**infra** および **worker** ロールが割り当てられている `infra` ノードがある場合、ユーザーのワークロードがこれに割り当てられないようにノードを設定する必要があります。

#### 重要

`infra` ノード用に作成されたデュアル **infra,worker** ラベルを保持し、ティントおよび容認 (Toleration) を使用してユーザーのワークロードがスケジュールされているノードを管理することを推奨します。ノードから **worker** ラベルを削除する場合には、カスタムプールを作成して管理する必要があります。**master** または **worker** 以外のラベルが割り当てられたノードは、カスタムプールなしには MCO で認識されません。**worker** ラベルを維持すると、カスタムラベルを選択するカスタムプールが存在しない場合に、ノードをデフォルトのワーカーマシン設定プールで管理できます。**infra** ラベルは、サブスクリプションの合計数にカウントされないクラスターと通信します。

### 前提条件

- 追加の **MachineSet** を OpenShift Container Platform クラスターに設定します。

## 手順

1. テイントを infra ノードに追加し、ユーザーのワークロードをこれにスケジュールできないようにします。
  - a. ノードにテイントがあるかどうかを判別します。

```
$ oc describe nodes <node_name>
```

### 出力例

```
oc describe node ci-ln-iyhx092-f76d1-nvdfm-worker-b-wln2l
Name:          ci-ln-iyhx092-f76d1-nvdfm-worker-b-wln2l
Roles:         worker
...
Taints:        node-role.kubernetes.io/infra:NoSchedule
...
```

この例では、ノードにテイントがあることを示しています。次の手順に進み、容認を Pod に追加してください。

- b. ユーザーワークロードをスケジューリングできないように、テイントを設定していない場合は、以下を実行します。

```
$ oc adm taint nodes <node_name> <key>=<value>:<effect>
```

以下に例を示します。

```
$ oc adm taint nodes node1 node-role.kubernetes.io/infra=reserved:NoExecute
```

### ヒント

または、以下の YAML を適用してテイントを追加できます。

```
kind: Node
apiVersion: v1
metadata:
  name: <node_name>
  labels:
    ...
spec:
  taints:
    - key: node-role.kubernetes.io/infra
      effect: NoExecute
      value: reserved
    ...
```

この例では、テイントを、キー **node-role.kubernetes.io/infra** およびテイントの effect **NoSchedule** を持つ **node1** に配置します。effect が **NoSchedule** のノードは、テイントを容認する Pod のみをスケジューリングしますが、既存の Pod はノードにスケジューリングされたままになります。



### 注記

Descheduler が使用されると、ノードのテイントに違反する Pod はクラスターからエビクトされる可能性があります。

2. ルーター、レジストリーおよびモニタリングのワークロードなどの、infra ノードにスケジュールする必要のある Pod 設定の容認を追加します。以下のコードを **Pod** オブジェクトの仕様に追加します。

```
tolerations:
  - effect: NoExecute ❶
    key: node-role.kubernetes.io/infra ❷
    operator: Exists ❸
    value: reserved ❹
```

- ❶ ノードに追加した effect を指定します。
- ❷ ノードに追加したキーを指定します。
- ❸ **Exists** Operator を、キー **node-role.kubernetes.io/infra** のあるテイントがノードに存在するように指定します。
- ❹ ノードに追加したキーと値のペア Taint の値を指定します。

この容認は、**oc adm taint** コマンドで作成されたテイントと一致します。この容認のある Pod は infra ノードにスケジュールできます。



### 注記

OLM でインストールされた Operator の Pod を infra ノードに常に移動できる訳ではありません。Operator Pod を移動する機能は、各 Operator の設定によって異なります。

3. スケジューラーを使用して Pod を infra ノードにスケジュールします。詳細は、**Pod のノードへの配置の制御** についてのドキュメントを参照してください。

### 関連情報

- Pod のノードへのスケジュールの一般的な情報については、[スケジューラーを使用した Pod の配置の制御](#) について参照してください。
- Pod を infra ノードにスケジュールする方法については、[リソースのインフラストラクチャマシンセットへの移動](#) について参照してください。

## 7.4. リソースのインフラストラクチャマシンセットへの移行

インフラストラクチャーリソースの一部はデフォルトでクラスターにデプロイされます。次のように、インフラストラクチャーノードセクターを追加して、作成したインフラストラクチャマシンセットにそれらを移動できます。

```
spec:
  nodePlacement: ❶
  nodeSelector:
```

```

matchLabels:
  node-role.kubernetes.io/infra: ""
tolerations:
- effect: NoSchedule
  key: node-role.kubernetes.io/infra
  value: reserved
- effect: NoExecute
  key: node-role.kubernetes.io/infra
  value: reserved

```

- 1 適切な値が設定された **nodeSelector** パラメーターを、移動する必要があるコンポーネントに追加します。表示されている形式の **nodeSelector** を使用することも、ノードに指定された値に基づいて **<key>: <value>** ペアを使用することもできます。インフラストラクチャーノードにテイントを追加した場合は、一致する容認も追加します。

特定のノードセレクターをすべてのインフラストラクチャーコンポーネントに適用すると、OpenShift Container Platform は [そのラベルを持つノードでそれらのワークロードをスケジュール](#) します。

### 7.4.1. ルーターの移動

ルーター Pod を異なるマシンセットにデプロイできます。デフォルトで、この Pod はワーカーノードにデプロイされます。

#### 前提条件

- 追加のマシンセットを OpenShift Container Platform クラスターに設定します。

#### 手順

1. ルーター Operator の **IngressController** カスタムリソースを表示します。

```
$ oc get ingresscontroller default -n openshift-ingress-operator -o yaml
```

コマンド出力は以下のテキストのようになります。

```

apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  creationTimestamp: 2019-04-18T12:35:39Z
  finalizers:
  - ingresscontroller.operator.openshift.io/finalizer-ingresscontroller
  generation: 1
  name: default
  namespace: openshift-ingress-operator
  resourceVersion: "11341"
  selfLink: /apis/operator.openshift.io/v1/namespaces/openshift-ingress-operator/ingresscontrollers/default
  uid: 79509e05-61d6-11e9-bc55-02ce4781844a
spec: {}
status:
  availableReplicas: 2
  conditions:
  - lastTransitionTime: 2019-04-18T12:36:15Z
    status: "True"

```

```

type: Available
domain: apps.<cluster>.example.com
endpointPublishingStrategy:
  type: LoadBalancerService
selector: ingresscontroller.operator.openshift.io/deployment-ingresscontroller=default

```

2. **ingresscontroller** リソースを編集し、 **nodeSelector** を **infra** ラベルを使用するように変更します。

```
$ oc edit ingresscontroller default -n openshift-ingress-operator
```

```

spec:
  nodePlacement:
    nodeSelector: ❶
    matchLabels:
      node-role.kubernetes.io/infra: ""
  tolerations:
    - effect: NoSchedule
      key: node-role.kubernetes.io/infra
      value: reserved
    - effect: NoExecute
      key: node-role.kubernetes.io/infra
      value: reserved

```

- ❶ 適切な値が設定された **nodeSelector** パラメーターを、移動する必要があるコンポーネントに追加します。表示されている形式の **nodeSelector** を使用することも、ノードに指定された値に基づいて **<key>: <value>** ペアを使用することもできます。インフラストラクチャーノードにテイントを追加した場合は、一致する容認も追加します。

3. ルーター Pod が **infra** ノードで実行されていることを確認します。

- a. ルーター Pod の一覧を表示し、実行中の Pod のノード名をメモします。

```
$ oc get pod -n openshift-ingress -o wide
```

### 出力例

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
NOMINATED NODE READINESS GATES						
router-default-86798b4b5d-bdlvd	1/1	Running	0	28s	10.130.2.4	ip-10-0-217-226.ec2.internal
	<none>	<none>				
router-default-955d875f4-255g8	0/1	Terminating	0	19h	10.129.2.4	ip-10-0-148-172.ec2.internal
	<none>	<none>				

この例では、実行中の Pod は **ip-10-0-217-226.ec2.internal** ノードにあります。

- b. 実行中の Pod のノードのステータスを表示します。

```
$ oc get node <node_name> ❶
```

- ❶ Pod の一覧より取得した **<node\_name>** を指定します。

## 出力例

NAME	STATUS	ROLES	AGE	VERSION
ip-10-0-217-226.ec2.internal	Ready	infra,worker	17h	v1.24.0

ロールの一覧に **infra** が含まれているため、Pod は正しいノードで実行されます。

## 7.4.2. デフォルトレジストリーの移行

レジストリー Operator を、その Pod を複数の異なるノードにデプロイするように設定します。

## 前提条件

- 追加のマシンセットを OpenShift Container Platform クラスターに設定します。

## 手順

1. **config/instance** オブジェクトを表示します。

```
$ oc get configs.imageregistry.operator.openshift.io/cluster -o yaml
```

## 出力例

```
apiVersion: imageregistry.operator.openshift.io/v1
kind: Config
metadata:
  creationTimestamp: 2019-02-05T13:52:05Z
  finalizers:
  - imageregistry.operator.openshift.io/finalizer
  generation: 1
  name: cluster
  resourceVersion: "56174"
  selfLink: /apis/imageregistry.operator.openshift.io/v1/configs/cluster
  uid: 36fd3724-294d-11e9-a524-12fdee2931b
spec:
  httpSecret: d9a012ccd117b1e6616ceccb2c3bb66a5fed1b5e481623
  logging: 2
  managementState: Managed
  proxy: {}
  replicas: 1
  requests:
    read: {}
    write: {}
  storage:
    s3:
      bucket: image-registry-us-east-1-c92e88cad85b48ec8b312344dff03c82-392c
      region: us-east-1
status:
...
```

2. **config/instance** オブジェクトを編集します。

```
$ oc edit configs.imageregistry.operator.openshift.io/cluster
```

```
spec:
  affinity:
    podAntiAffinity:
      preferredDuringSchedulingIgnoredDuringExecution:
        - podAffinityTerm:
            namespaces:
              - openshift-image-registry
            topologyKey: kubernetes.io/hostname
            weight: 100
  logLevel: Normal
  managementState: Managed
  nodeSelector: ❶
    node-role.kubernetes.io/infra: ""
  tolerations:
    - effect: NoSchedule
      key: node-role.kubernetes.io/infra
      value: reserved
    - effect: NoExecute
      key: node-role.kubernetes.io/infra
      value: reserved
```

- ❶ 適切な値が設定された **nodeSelector** パラメーターを、移動する必要があるコンポーネントに追加します。表示されている形式の **nodeSelector** を使用することも、ノードに指定された値に基づいて **<key>: <value>** ペアを使用することもできます。インフラストラクチャーノードにテイントを追加した場合は、一致する容認も追加します。

3. レジストリー Pod がインフラストラクチャーノードに移動していることを確認します。
  - a. 以下のコマンドを実行して、レジストリー Pod が置かれているノードを特定します。

```
$ oc get pods -o wide -n openshift-image-registry
```

- b. ノードに指定したラベルがあることを確認します。

```
$ oc describe node <node_name>
```

コマンド出力を確認し、**node-role.kubernetes.io/infra** が **LABELS** 一覧にあることを確認します。

### 7.4.3. モニタリングソリューションの移動

監視スタックには、Prometheus、Thanos Querier、Alertmanager などの複数のコンポーネントが含まれています。Cluster Monitoring Operator は、このスタックを管理します。モニタリングスタックをインフラストラクチャーノードに再デプロイするために、カスタム config map を作成して適用できます。

#### 手順

1. **cluster-monitoring-config** 設定マップを編集し、**nodeSelector** を変更して **infra** ラベルを使用します。

```
$ oc edit configmap cluster-monitoring-config -n openshift-monitoring
```

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |+
    alertmanagerMain:
      nodeSelector: ❶
        node-role.kubernetes.io/infra: ""
      tolerations:
        - key: node-role.kubernetes.io/infra
          value: reserved
          effect: NoSchedule
        - key: node-role.kubernetes.io/infra
          value: reserved
          effect: NoExecute
    prometheusK8s:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      tolerations:
        - key: node-role.kubernetes.io/infra
          value: reserved
          effect: NoSchedule
        - key: node-role.kubernetes.io/infra
          value: reserved
          effect: NoExecute
    prometheusOperator:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      tolerations:
        - key: node-role.kubernetes.io/infra
          value: reserved
          effect: NoSchedule
        - key: node-role.kubernetes.io/infra
          value: reserved
          effect: NoExecute
    k8sPrometheusAdapter:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      tolerations:
        - key: node-role.kubernetes.io/infra
          value: reserved
          effect: NoSchedule
        - key: node-role.kubernetes.io/infra
          value: reserved
          effect: NoExecute
    kubeStateMetrics:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      tolerations:
        - key: node-role.kubernetes.io/infra
          value: reserved
          effect: NoSchedule
        - key: node-role.kubernetes.io/infra
          value: reserved
```

```

    effect: NoExecute
  telemetryClient:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
  tolerations:
    - key: node-role.kubernetes.io/infra
      value: reserved
      effect: NoSchedule
    - key: node-role.kubernetes.io/infra
      value: reserved
      effect: NoExecute
  openshiftStateMetrics:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
  tolerations:
    - key: node-role.kubernetes.io/infra
      value: reserved
      effect: NoSchedule
    - key: node-role.kubernetes.io/infra
      value: reserved
      effect: NoExecute
  thanosQuerier:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
  tolerations:
    - key: node-role.kubernetes.io/infra
      value: reserved
      effect: NoSchedule
    - key: node-role.kubernetes.io/infra
      value: reserved
      effect: NoExecute

```

- 1 適切な値が設定された **nodeSelector** パラメーターを、移動する必要があるコンポーネントに追加します。表示されている形式の **nodeSelector** を使用することも、ノードに指定された値に基づいて **<key>: <value>** ペアを使用することもできます。インフラストラクチャーノードにテイントを追加した場合は、一致する容認も追加します。

2. モニタリング Pod が新規マシンに移行することを確認します。

```
$ watch 'oc get pod -n openshift-monitoring -o wide'
```

3. コンポーネントが **infra** ノードに移動していない場合は、このコンポーネントを持つ Pod を削除します。

```
$ oc delete pod -n openshift-monitoring <pod>
```

削除された Pod からのコンポーネントが **infra** ノードに再作成されます。

#### 7.4.4. OpenShift Logging リソースの移動

Elasticsearch および Kibana などのロギングシステムコンポーネントの pod を異なるノードにデプロイするように Cluster Logging Operator を設定できます。Cluster Logging Operator Pod をインストールした場所から移動することはできません。

たとえば、Elasticsearch Pod の CPU、メモリーおよびディスクの要件が高いために、この Pod を別のノードに移動できます。

## 前提条件

- Red Hat OpenShift Logging および Elasticsearch Operators がインストールされている必要があります。これらの機能はデフォルトでインストールされません。

## 手順

- openshift-logging** プロジェクトで **ClusterLogging** カスタムリソース (CR) を編集します。

```
$ oc edit ClusterLogging instance
```

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging

...

spec:
  collection:
    logs:
      fluentd:
        resources: null
        type: fluentd
  logStore:
    elasticsearch:
      nodeCount: 3
      nodeSelector: ❶
        node-role.kubernetes.io/infra: "
    tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/infra
        value: reserved
      - effect: NoExecute
        key: node-role.kubernetes.io/infra
        value: reserved
    redundancyPolicy: SingleRedundancy
    resources:
      limits:
        cpu: 500m
        memory: 16Gi
      requests:
        cpu: 500m
        memory: 16Gi
    storage: {}
    type: elasticsearch
  managementState: Managed
  visualization:
    kibana:
      nodeSelector: ❷
        node-role.kubernetes.io/infra: "
      tolerations:
        - effect: NoSchedule
          key: node-role.kubernetes.io/infra
```

```

    value: reserved
  - effect: NoExecute
    key: node-role.kubernetes.io/infra
    value: reserved
  proxy:
    resources: null
  replicas: 1
  resources: null
  type: kibana
...

```

- 1 2 適切な値が設定された **nodeSelector** パラメーターを、移動する必要があるコンポーネントに追加します。表示されている形式の **nodeSelector** を使用することも、ノードに指定された値に基づいて **<key>: <value>** ペアを使用することもできます。インフラストラクチャーノードにテイントを追加した場合は、一致する容認も追加します。

## 検証

コンポーネントが移動したことを確認するには、**oc get pod -o wide** コマンドを使用できます。

以下に例を示します。

- Kibana Pod を **ip-10-0-147-79.us-east-2.compute.internal** ノードから移動する必要がある場合は、以下を実行します。

```
$ oc get pod kibana-5b8bdf44f9-ccpq9 -o wide
```

### 出力例

```

NAME                                READY STATUS RESTARTS AGE IP          NODE
NOMINATED NODE READINESS GATES
kibana-5b8bdf44f9-ccpq9 2/2   Running 0      27s 10.129.2.18 ip-10-0-147-79.us-
east-2.compute.internal <none>      <none>

```

- Kibana Pod を、専用インフラストラクチャーノードである **ip-10-0-139-48.us-east-2.compute.internal** ノードに移動する必要がある場合は、以下を実行します。

```
$ oc get nodes
```

### 出力例

```

NAME                                STATUS ROLES    AGE  VERSION
ip-10-0-133-216.us-east-2.compute.internal Ready master    60m  v1.24.0
ip-10-0-139-146.us-east-2.compute.internal Ready master    60m  v1.24.0
ip-10-0-139-192.us-east-2.compute.internal Ready worker    51m  v1.24.0
ip-10-0-139-241.us-east-2.compute.internal Ready worker    51m  v1.24.0
ip-10-0-147-79.us-east-2.compute.internal Ready worker    51m  v1.24.0
ip-10-0-152-241.us-east-2.compute.internal Ready master    60m  v1.24.0
ip-10-0-139-48.us-east-2.compute.internal Ready infra     51m  v1.24.0

```

ノードには **node-role.kubernetes.io/infra: "** ラベルがあることに注意してください。

```
$ oc get node ip-10-0-139-48.us-east-2.compute.internal -o yaml
```

## 出力例

```
kind: Node
apiVersion: v1
metadata:
  name: ip-10-0-139-48.us-east-2.compute.internal
  selfLink: /api/v1/nodes/ip-10-0-139-48.us-east-2.compute.internal
  uid: 62038aa9-661f-41d7-ba93-b5f1b6ef8751
  resourceVersion: '39083'
  creationTimestamp: '2020-04-13T19:07:55Z'
  labels:
    node-role.kubernetes.io/infra: "
  ...
```

- Kibana Pod を移動するには、**ClusterLogging** CR を編集してノードセクターを追加します。

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging
...

spec:
...

visualization:
  kibana:
    nodeSelector: ❶
    node-role.kubernetes.io/infra: "
    proxy:
      resources: null
    replicas: 1
    resources: null
  type: kibana
```

- ❶ ノード仕様のラベルに一致するノードセクターを追加します。

- CR を保存した後に、現在の Kibana Pod は終了し、新規 Pod がデプロイされます。

```
$ oc get pods
```

## 出力例

NAME	READY	STATUS	RESTARTS	AGE
cluster-logging-operator-84d98649c4-zb9g7	1/1	Running	0	29m
elasticsearch-cdm-hwv01pf7-1-56588f554f-kpmlg	2/2	Running	0	28m
elasticsearch-cdm-hwv01pf7-2-84c877d75d-75wqj	2/2	Running	0	28m
elasticsearch-cdm-hwv01pf7-3-f5d95b87b-4nx78	2/2	Running	0	28m
fluentd-42dzz	1/1	Running	0	28m
fluentd-d74rq	1/1	Running	0	28m
fluentd-m5vr9	1/1	Running	0	28m

```

fluentd-nkx17          1/1   Running    0      28m
fluentd-pdvqb          1/1   Running    0      28m
fluentd-tflh6          1/1   Running    0      28m
kibana-5b8bdf44f9-ccpq9 2/2   Terminating 0      4m11s
kibana-7d85dcffc8-bfpfp 2/2   Running    0      33s

```

- 新規 Pod が **ip-10-0-139-48.us-east-2.compute.internal** ノードに置かれます。

```
$ oc get pod kibana-7d85dcffc8-bfpfp -o wide
```

## 出力例

```

NAME                READY  STATUS   RESTARTS  AGE  IP            NODE
NOMINATED NODE READINESS GATES
kibana-7d85dcffc8-bfpfp 2/2   Running    0         43s  10.131.0.22  ip-10-0-139-48.us-east-2.compute.internal <none> <none>

```

- しばらくすると、元の Kibana Pod が削除されます。

```
$ oc get pods
```

## 出力例

```

NAME                READY  STATUS   RESTARTS  AGE
cluster-logging-operator-84d98649c4-zb9g7 1/1   Running    0         30m
elasticsearch-cdm-hwv01pf7-1-56588f554f-kpmlg 2/2   Running    0         29m
elasticsearch-cdm-hwv01pf7-2-84c877d75d-75wqj 2/2   Running    0         29m
elasticsearch-cdm-hwv01pf7-3-f5d95b87b-4nx78 2/2   Running    0         29m
fluentd-42dzz        1/1   Running    0         29m
fluentd-d74rq        1/1   Running    0         29m
fluentd-m5vr9        1/1   Running    0         29m
fluentd-nkx17        1/1   Running    0         29m
fluentd-pdvqb        1/1   Running    0         29m
fluentd-tflh6        1/1   Running    0         29m
kibana-7d85dcffc8-bfpfp 2/2   Running    0         62s

```

## 関連情報

- OpenShift Container Platform コンポーネントの移動についての一般的な情報は、[モニタリングについてのドキュメント](#) を参照してください。

## 第8章 RHEL コンピュータマシンの OPENSIFT CONTAINER PLATFORM クラスターへの追加

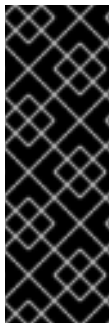
OpenShift Container Platform では、Red Hat Enterprise Linux (RHEL) コンピュータマシンを、**x86\_64** アーキテクチャー上のユーザープロビジョニングされたインフラストラクチャークラスターまたはインストールプロビジョニングされたインフラストラクチャークラスターに追加できます。RHEL は、コンピュータマシンでのみのオペレーティングシステムとして使用できます。

### 8.1. RHEL コンピュータノードのクラスターへの追加について

OpenShift Container Platform 4.11 では、**x86\_64** アーキテクチャー上でユーザープロビジョニングまたはインストーラープロビジョニングのインフラストラクチャーインストールを使用する場合、クラスター内のコンピューティングマシンとして Red Hat Enterprise Linux (RHEL) マシンを使用するオプションがあります。クラスター内のコントロールプレーンマシンには Red Hat Enterprise Linux CoreOS (RHCOS) マシンを使用する必要があります。

クラスターで RHEL コンピュータマシンを使用することを選択した場合は、すべてのオペレーティングシステムのライフサイクル管理とメンテナンスを担当します。システムの更新を実行し、パッチを適用し、その他すべての必要なタスクを完了する必要があります。

インストーラーがプロビジョニングしたインフラストラクチャークラスターの場合、インストーラーがプロビジョニングしたインフラストラクチャークラスターの自動スケーリングにより Red Hat Enterprise Linux CoreOS (RHCOS) コンピューティングマシンがデフォルトで追加されるため、RHEL コンピューティングマシンを手動で追加する必要があります。



#### 重要

- OpenShift Container Platform をクラスター内のマシンから削除するには、オペレーティングシステムを破棄する必要があるため、クラスターに追加する RHEL マシンについては専用のハードウェアを使用する必要があります。
- swap メモリーは、OpenShift Container Platform クラスターに追加されるすべての RHEL マシンで無効にされます。これらのマシンで swap メモリーを有効にすることはできません。

RHEL コンピュータマシンは、コントロールプレーンを初期化してからクラスターに追加する必要があります。

### 8.2. RHEL コンピュータノードのシステム要件

OpenShift Container Platform 環境の Red Hat Enterprise Linux (RHEL) コンピュータマシンは以下の最低のハードウェア仕様およびシステムレベルの要件を満たしている必要があります。

- まず、お使いの Red Hat アカウントに有効な OpenShift Container Platform サブスクリプションがなければなりません。これがない場合は、営業担当者にお問い合わせください。
- 実稼働環境では予想されるワークロードに対応するコンピューターノードを提供する必要があります。クラスター管理者は、予想されるワークロードを計算し、オーバーヘッドの約 10 % を追加する必要があります。実稼働環境の場合、ノードホストの障害が最大容量に影響を与えることがないように、十分なリソースを割り当てるようにします。
- 各システムは、以下のハードウェア要件を満たしている必要があります。
  - 物理または仮想システム、またはパブリックまたはプライベート IaaS で実行されるイン

マシン。

- ベース OS: [RHEL 8.5 または 8.7](#) (最小のインストールオプション)。



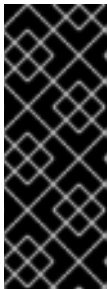
### 重要

OpenShift Container Platform クラスターへの RHEL 7 コンピュータマシンの追加はサポートされません。

以前の OpenShift Container Platform のバージョンで以前にサポートされていた RHEL 7 コンピュータマシンがある場合、RHEL 8 にアップグレードすることはできません。新しい RHEL 8 ホストをデプロイする必要があり、古い RHEL 7 ホストを削除する必要があります。詳細は、ノードの管理セクションを参照してください。

OpenShift Container Platform で非推奨となったか、または削除された主な機能の最新の一覧については、OpenShift Container Platform リリースノート [の 非推奨および削除された機能セクション](#)を参照してください。

- FIPS モードで OpenShift Container Platform をデプロイしている場合、起動する前に FIPS を RHEL マシン上で有効にする必要があります。RHEL 8 ドキュメントの [Installing a RHEL 8 system with FIPS mode enabled](#) を参照してください。



### 重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピュータからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、[FIPS モードでのシステムのインストール](#) を参照してください。プロセス暗号化ライブラリーでの FIPS 検証済みまたはモジュールの使用は、**x86\_64** アーキテクチャーでの OpenShift Container Platform デプロイメントでのみサポートされます。

- NetworkManager 1.0 以降。
- 1vCPU。
- 最小 8 GB の RAM。
- `/var/` を含むファイルシステムの最小 15 GB のハードディスク領域。
- `/usr/local/bin/` を含むファイルシステムの最小 1 GB のハードディスク領域。
- 一時ディレクトリーを含むファイルシステムの最小 1 GB のハードディスク領域。システムの一時的ディレクトリーは、Python の標準ライブラリーの `tempfile` モジュールで定義されるルールに基づいて決定されます。
  - 各システムは、システムプロバイダーの追加の要件を満たす必要があります。たとえば、クラスターを VMware vSphere にインストールしている場合、ディスクはその [ストレージガイドライン](#) に応じて設定され、**disk.enableUUID=true** 属性が設定される必要があります。
  - 各システムは、DNS で解決可能なホスト名を使用してクラスターの API エンドポイントにアクセスする必要があります。配置されているネットワークセキュリティアクセス制御は、クラスターの API サービスエンドポイントへのシステムアクセスを許可する必要があります。

## 関連情報

- [ノードの削除](#)

### 8.2.1. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

## 8.3. クラウド用イメージの準備

各種のイメージ形式は AWS で直接使用できないので、Amazon Machine Images (AMI) が必要です。Red Hat が提供している AMI を使用するか、または独自のイメージを手動でインポートできます。EC2 インスタンスをプロビジョニングする前に AMI が存在している必要があります。コンピュータマシンに必要な正しい RHEL バージョンを選択するには、有効な AMI ID が必要です。

### 8.3.1. AWS で利用可能な最新の RHEL イメージの一覧表示

AMI ID は、AWS のネイティブブートイメージに対応します。EC2 インスタンスがプロビジョニングされる前に AMI が存在している必要があるため、設定前に AMI ID を把握しておく必要があります。[AWS コマンドラインインターフェイス \(CLI\)](#) は、利用可能な Red Hat Enterprise Linux (RHEL) イメージ ID の一覧を表示するために使用されます。

#### 前提条件

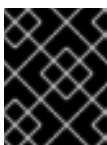
- AWS CLI をインストールしている。

#### 手順

- このコマンドを使用して、RHEL 8.4 Amazon Machine Images (AMI) の一覧を表示します。

```
$ aws ec2 describe-images --owners 309956199498 \ ❶
--query 'sort_by(Images, &CreationDate)[*].[CreationDate,Name,ImageId]' \ ❷
--filters "Name=name,Values=RHEL-8.4*" \ ❸
--region us-east-1 \ ❹
--output table ❺
```

- ❶ **--owners** コマンドオプションは、アカウント ID **309956199498** に基づいて Red Hat イメージを表示します。



#### 重要

Red Hat が提供するイメージの AMI ID を表示するには、このアカウント ID が必要です。

- ❷ **--query** コマンドオプションは、イメージが **'sort\_by(Images, &CreationDate)[\*].[CreationDate,Name,ImageId]'** のパラメーターでソートされる方法を設定します。この場合、イメージは作成日でソートされ、テーブルが作成日、イメージ名、および AMI ID を表示するように設定されます。

- 3 **--filter** コマンドオプションは、表示される RHEL のバージョンを設定します。この例では、フィルターが **"Name=name,Values=RHEL-8.4\*"** で設定されているため、RHEL 8.4
- 4 **--region** コマンドオプションは、AMI が保存されるリージョンを設定します。
- 5 **--output** コマンドオプションは、結果の表示方法を設定します。



#### 注記

AWS 用の RHEL コンピュートマシンを作成する場合、AMI が RHEL 8.4 または 8.5 であることを確認します。

#### 出力例

```
-----
|                               DescribelImages                               |
+-----+-----+-----+-----+-----+-----+
| 2021-03-18T14:23:11.000Z | RHEL-8.4.0_HVM_BETA-20210309-x86_64-1-Hourly2-GP2 | ami-07eeb4db5f7e5a8fb |
| 2021-03-18T14:38:28.000Z | RHEL-8.4.0_HVM_BETA-20210309-arm64-1-Hourly2-GP2 | ami-069d22ec49577d4bf |
| 2021-05-18T19:06:34.000Z | RHEL-8.4.0_HVM-20210504-arm64-2-Hourly2-GP2      | ami-01fc429821bf1f4b4 |
| 2021-05-18T20:09:47.000Z | RHEL-8.4.0_HVM-20210504-x86_64-2-Hourly2-GP2      | ami-0b0af3577fe5e3532 |
+-----+-----+-----+-----+-----+-----+

```

#### 関連情報

- [RHEL イメージを AWS に手動でインポートする](#) こともできます。

## 8.4. PLAYBOOK 実行のためのマシンの準備

Red Hat Enterprise Linux (RHEL) をオペレーティングシステムとして使用するコンピュートマシンを OpenShift Container Platform 4.11 クラスターに追加する前に、新たなノードをクラスターに追加する Ansible Playbook を実行する RHEL 8 マシンを準備する必要があります。このマシンはクラスターの一部にはなりませんが、クラスターにアクセスする必要があります。

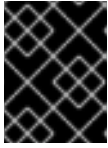
#### 前提条件

- Playbook を実行するマシンに OpenShift CLI (**oc**) をインストールします。
- **cluster-admin** 権限を持つユーザーとしてログインしている。

#### 手順

1. クラスターの **kubeconfig** ファイルおよびクラスターのインストールに使用したインストールプログラムが RHEL 8 マシン上にあることを確認します。これを実行する 1 つの方法として、クラスターのインストールに使用したマシンと同じマシンを使用することができます。
2. マシンを、コンピュートマシンとして使用する予定のすべての RHEL ホストにアクセスできるように設定します。Bastion と SSH プロキシまたは VPN の使用など、所属する会社で許可されるすべての方法を利用できます。

3. すべての RHEL ホストへの SSH アクセスを持つユーザーを Playbook を実行するマシンで設定します。



### 重要

SSH キーベースの認証を使用する場合、キーを SSH エージェントで管理する必要があります。

4. これを実行していない場合には、マシンを RHSM に登録し、**OpenShift** サブスクリプションのプールをこれにアタッチします。

- a. マシンを RHSM に登録します。

```
# subscription-manager register --username=<user_name> --password=<password>
```

- b. RHSM から最新のサブスクリプションデータをプルします。

```
# subscription-manager refresh
```

- c. 利用可能なサブスクリプションを一覧表示します。

```
# subscription-manager list --available --matches '*OpenShift*'
```

- d. 直前のコマンドの出力で、OpenShift Container Platform サブスクリプションのプール ID を見つけ、これをアタッチします。

```
# subscription-manager attach --pool=<pool_id>
```

5. OpenShift Container Platform 4.11 で必要なりポジトリを有効にします。

```
# subscription-manager repos \
  --enable="rhel-8-for-x86_64-baseos-rpms" \
  --enable="rhel-8-for-x86_64-appstream-rpms" \
  --enable="rhocp-4.11-for-rhel-8-x86_64-rpms"
```

6. **openshift-ansible** を含む必要なパッケージをインストールします。

```
# yum install openshift-ansible openshift-clients jq
```

**openshift-ansible** パッケージはインストールプログラムユーティリティを提供し、Ansible Playbook などのクラスターに RHEL コンピュータノードを追加するために必要な他のパッケージおよび関連する設定ファイルをプルします。**openshift-clients** は **oc** CLI を提供し、**jq** パッケージはコマンドライン上での JSON 出力の表示方法を向上させます。

## 8.5. RHEL コンピュータノードの準備

Red Hat Enterprise Linux (RHEL) マシンを OpenShift Container Platform クラスターに追加する前に、各ホストを Red Hat Subscription Manager (RHSM) に登録し、有効な OpenShift Container Platform サブスクリプションをアタッチし、必要なりポジトリを有効にする必要があります。

1. 各ホストで RHSM に登録します。

```
# subscription-manager register --username=<user_name> --password=<password>
```

2. RHSM から最新のサブスクリプションデータをプルします。

```
# subscription-manager refresh
```

3. 利用可能なサブスクリプションを一覧表示します。

```
# subscription-manager list --available --matches '*OpenShift*'
```

4. 直前のコマンドの出力で、OpenShift Container Platform サブスクリプションのプール ID を見つけ、これをアタッチします。

```
# subscription-manager attach --pool=<pool_id>
```

5. yum リポジトリをすべて無効にします。

- a. 有効にされている RHSM リポジトリをすべて無効にします。

```
# subscription-manager repos --disable="**"
```

- b. 残りの yum リポジトリを一覧表示し、**repo id** にあるそれらの名前をメモします (ある場合)。

```
# yum repolist
```

- c. **yum-config-manager** を使用して、残りの yum リポジトリを無効にします。

```
# yum-config-manager --disable <repo_id>
```

または、すべてのリポジトリを無効にします。

```
# yum-config-manager --disable \*
```

利用可能なリポジトリが多い場合には、数分の時間がかかることがあります。

6. OpenShift Container Platform 4.11 で必要なリポジトリのみを有効にします。

```
# subscription-manager repos \
  --enable="rhel-8-for-x86_64-baseos-rpms" \
  --enable="rhel-8-for-x86_64-appstream-rpms" \
  --enable="rhocp-4.11-for-rhel-8-x86_64-rpms" \
  --enable="fast-datapath-for-rhel-8-x86_64-rpms"
```

7. ホストで firewalld を停止し、無効にします。

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



### 注記

firewalld は、後で有効にすることはできません。これを実行する場合、ワーカー上の OpenShift Container Platform ログにはアクセスできません。

## 8.6. AWS での RHEL インスタンスへのロールパーミッションの割り当て

ブラウザーで Amazon IAM コンソールを使用して、必要なロールを選択し、ワーカーノードに割り当てることができます。

### 手順

1. AWS IAM コンソールから、[任意の IAM ロール](#) を作成します。
2. [IAM ロール](#) を必要なワーカーノードに割り当てます。

### 関連情報

- [Required AWS permissions for IAM roles](#) を参照してください。

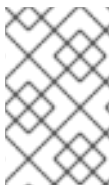
## 8.7. 所有または共有されている RHEL ワーカーノードへのタグ付け

クラスターは `kubernetes.io/cluster/<clusterid>,Value=(owned|shared)` タグの値を使用して、AWS クラスターに関するリソースの有効期間を判別します。

- リソースをクラスターの破棄の一環として破棄する必要がある場合は、**owned** タグの値を追加する必要があります。
- クラスターが破棄された後にリソースが引き続いて存在する場合、**shared** タグの値を追加する必要があります。このタグ付けは、クラスターがこのリソースを使用することを示しますが、リソースには別の所有者が存在します。

### 手順

- RHEL コンピュータマシンの場合、RHEL ワーカーマシンでは、`kubernetes.io/cluster/<clusterid>=owned` または `kubernetes.io/cluster/<cluster-id>=shared` でタグ付けする必要があります。



### 注記

すべての既存セキュリティグループに `kubernetes.io/cluster/<name>,Value=<clusterid>` のタグを付けないでください。その場合、Elastic Load Balancing (ELB) がロードバランサーを作成できなくなります。

## 8.8. RHEL コンピュータマシンのクラスターへの追加

Red Hat Enterprise Linux をオペレーティングシステムとして使用するコンピュータマシンを OpenShift Container Platform 4.11 クラスターに追加することができます。

### 前提条件

- Playbook を実行するマシンに必要なパッケージをインストールし、必要な設定が行われています。
- インストール用の RHEL ホストを準備しています。

### 手順

Playbook を実行するために準備しているマシンで以下の手順を実行します。

1. コンピュータマシンホストおよび必要な変数を定義する `/<path>/inventory/hosts` という名前の Ansible インベントリーファイルを作成します。

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

openshift_kubeconfig_path=~/.kube/config" ❸

[new_workers] ❹
mycluster-rhel8-0.example.com
mycluster-rhel8-1.example.com
```

- ❶ Ansible タスクをリモートコンピュータマシンで実行するユーザー名を指定します。
- ❷ **ansible\_user** の **root** を指定しない場合、**ansible\_become** を **True** に設定し、ユーザーに `sudo` パーミッションを割り当てる必要があります。
- ❸ クラスターの **kubeconfig** ファイルへのパスを指定します。
- ❹ クラスターに追加する各 RHEL マシンを一覧表示します。各ホストについて完全修飾ドメイン名を指定する必要があります。この名前は、クラスターがマシンにアクセスするために使用するホスト名であるため、マシンにアクセスできるように正しいパブリックまたはプライベートの名前を設定します。

2. Ansible Playbook ディレクトリーに移動します。

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

3. Playbook を実行します。

```
$ ansible-playbook -i /<path>/inventory/hosts playbooks/scaleup.yml ❶
```

- ❶ **<path>** については、作成した Ansible インベントリーファイルへのパスを指定します。

## 8.9. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

### 前提条件

- マシンがクラスターに追加されています。

### 手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

## 出力例

NAME	STATUS	ROLES	AGE	VERSION
master-0	Ready	master	63m	v1.24.0
master-1	Ready	master	63m	v1.24.0
master-2	Ready	master	64m	v1.24.0

出力には作成したすべてのマシンが一覧表示されます。



## 注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

## 出力例

NAME	AGE	REQUESTOR	CONDITION
csr-8b2br	15m	system:serviceaccount:openshift-machine-config-operator:node-bootstrapper	Pending
csr-8vnps	15m	system:serviceaccount:openshift-machine-config-operator:node-bootstrapper	Pending
...			

この例では、2つのマシンがクラスターに参加しています。この一覧にはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。



## 注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後1時間以内に CSR を承認してください。1時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに3つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。



## 注記

ベアメタルおよび他のユーザーによってプロビジョニングされるインフラストラクチャーなどのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、**oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrap** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティを確認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

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

- 1** **<csr\_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ 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

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

## 出力例

```
NAME      AGE   REQUESTOR                                     CONDITION
csr-bfd72 5m26s system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv 5m26s system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

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

1 **<csr\_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

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

- すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

### 出力例

```
NAME      STATUS  ROLES  AGE  VERSION
master-0  Ready   master 73m  v1.24.0
master-1  Ready   master 73m  v1.24.0
master-2  Ready   master 74m  v1.24.0
worker-0  Ready   worker 11m  v1.24.0
worker-1  Ready   worker 11m  v1.24.0
```



### 注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

### 関連情報

- CSR の詳細は、[Certificate Signing Requests](#) を参照してください。

## 8.10. ANSIBLE ホストファイルの必須パラメーター

Red Hat Enterprise Linux (RHEL) コンピュータマシンをクラスターに追加する前に、以下のパラメーターを Ansible ホストファイルに定義する必要があります。

パラメーター	説明	値
<b>ansible_user</b>	パスワードなしの SSH ベースの認証を許可する SSH ユーザー。SSH キーベースの認証を使用する場合、キーを SSH エージェントで管理する必要があります。	システム上のユーザー名。デフォルト値は <b>root</b> です。
<b>ansible_become</b>	<b>ansible_user</b> の値が root ではない場合、 <b>ansible_become</b> を <b>True</b> に設定する必要があり、 <b>ansible_user</b> として指定するユーザーはパスワードなしの sudo アクセスが可能になるように設定される必要があります。	<b>True</b> 。値が <b>True</b> ではない場合、このパラメーターを指定したり、定義したりしないでください。

パラメーター	説明	値
<b>openshift_kubeconfig_path</b>	クラスターの <b>kubeconfig</b> ファイルが含まれるローカルディレクトリーへのパスおよびファイル名を指定します。	設定ファイルのパスと名前。

### 8.10.1. オプション: RHCOS コンピュータマシンのクラスターからの削除

Red Hat Enterprise Linux (RHEL) コンピュータマシンをクラスターに追加した後に、オプションで Red Hat Enterprise Linux CoreOS (RHCOS) コンピュータマシンを削除し、リソースを解放できます。

#### 前提条件

- RHEL コンピュータマシンをクラスターに追加済みです。

#### 手順

1. マシンの一覧を表示し、RHCOS コンピュータマシンのノード名を記録します。

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

2. それぞれの RHCOS コンピュータマシンについて、ノードを削除します。
  - a. **oc adm cordon** コマンドを実行して、ノードにスケジュール対象外 (unschedulable) のマークを付けます。

```
$ oc adm cordon <node_name> ❶
```

- ❶ RHCOS コンピュータマシンのノード名を指定します。

- b. ノードからすべての Pod をドレイン (解放) します。

```
$ oc adm drain <node_name> --force --delete-emptydir-data --ignore-daemonsets ❶
```

- ❶ 分離した RHCOS コンピュータマシンのノード名を指定します。

- c. ノードを削除します。

```
$ oc delete nodes <node_name> ❶
```

- ❶ ドレイン (解放) した RHCOS コンピュータマシンのノード名を指定します。

3. コンピュータマシンの一覧を確認し、RHEL ノードのみが残っていることを確認します。

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

4. RHCOS マシンをクラスターのコンピュータマシンのロードバランサーから削除します。仮想マシンを削除したり、RHCOS コンピュータマシンの物理ハードウェアを再イメージ化したりできます。

## 第9章 RHEL コンピュータマシンの OPENSIFT CONTAINER PLATFORM クラスターへのさらなる追加

OpenShift Container Platform クラスターに Red Hat Enterprise Linux (RHEL) コンピュータマシン (またはワーカーマシンとしても知られる) がすでに含まれる場合、RHEL コンピュータマシンをさらに追加することができます。

### 9.1. RHEL コンピュータノードのクラスターへの追加について

OpenShift Container Platform 4.11 では、**x86\_64** アーキテクチャー上でユーザープロビジョニングまたはインストーラープロビジョニングのインフラストラクチャーインストールを使用する場合、クラスター内のコンピューティングマシンとして Red Hat Enterprise Linux (RHEL) マシンを使用するオプションがあります。クラスター内のコントロールプレーンマシンには Red Hat Enterprise Linux CoreOS (RHCOS) マシンを使用する必要があります。

クラスターで RHEL コンピュータマシンを使用することを選択した場合は、すべてのオペレーティングシステムのライフサイクル管理とメンテナンスを担当します。システムの更新を実行し、パッチを適用し、その他すべての必要なタスクを完了する必要があります。

インストーラーがプロビジョニングしたインフラストラクチャークラスターの場合、インストーラーがプロビジョニングしたインフラストラクチャークラスターの自動スケーリングにより Red Hat Enterprise Linux CoreOS (RHCOS) コンピューティングマシンがデフォルトで追加されるため、RHEL コンピューティングマシンを手動で追加する必要があります。



#### 重要

- OpenShift Container Platform をクラスター内のマシンから削除するには、オペレーティングシステムを破棄する必要があるため、クラスターに追加する RHEL マシンについては専用のハードウェアを使用する必要があります。
- swap メモリーは、OpenShift Container Platform クラスターに追加されるすべての RHEL マシンで無効にされます。これらのマシンで swap メモリーを有効にすることはできません。

RHEL コンピュータマシンは、コントロールプレーンを初期化してからクラスターに追加する必要があります。

### 9.2. RHEL コンピュータノードのシステム要件

OpenShift Container Platform 環境の Red Hat Enterprise Linux (RHEL) コンピュータマシンは以下の最低のハードウェア仕様およびシステムレベルの要件を満たしている必要があります。

- まず、お使いの Red Hat アカウントに有効な OpenShift Container Platform サブスクリプションがなければなりません。これがない場合は、営業担当者にお問い合わせください。
- 実稼働環境では予想されるワークロードに対応するコンピューターノードを提供する必要があります。クラスター管理者は、予想されるワークロードを計算し、オーバーヘッドの約 10 % を追加する必要があります。実稼働環境の場合、ノードホストの障害が最大容量に影響を与えることがないように、十分なりソースを割り当てるようにします。
- 各システムは、以下のハードウェア要件を満たしている必要があります。
  - 物理または仮想システム、またはパブリックまたはプライベート IaaS で実行されるインスタンス。

- ベース OS: [RHEL 8.5 または 8.7](#) (最小のインストールオプション)。



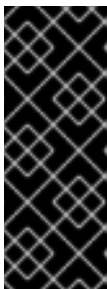
### 重要

OpenShift Container Platform クラスターへの RHEL 7 コンピュータマシンの追加はサポートされません。

以前の OpenShift Container Platform のバージョンで以前にサポートされていた RHEL 7 コンピュータマシンがある場合、RHEL 8 にアップグレードすることはできません。新しい RHEL 8 ホストをデプロイする必要がある、古い RHEL 7 ホストを削除する必要があります。詳細は、ノードの管理セクションを参照してください。

OpenShift Container Platform で非推奨となったか、または削除された主な機能の最新の一覧については、OpenShift Container Platform リリースノートの [非推奨および削除された機能セクション](#)を参照してください。

- FIPS モードで OpenShift Container Platform をデプロイしている場合、起動する前に FIPS を RHEL マシン上で有効にする必要があります。RHEL 8 ドキュメントの[Installing a RHEL 8 system with FIPS mode enabled](#)を参照してください。



### 重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、[FIPS モードでのシステムのインストール](#)を参照してください。プロセス暗号化ライブラリーでの FIPS 検証済みまたはモジュールの使用は、**x86\_64** アーキテクチャーでの OpenShift Container Platform デプロイメントでのみサポートされます。

- NetworkManager 1.0 以降。
- 1vCPU。
- 最小 8 GB の RAM。
- **/var/** を含むファイルシステムの最小 15 GB のハードディスク領域。
- **/usr/local/bin/** を含むファイルシステムの最小 1GB のハードディスク領域。
- 一時ディレクトリーを含むファイルシステムの最小 1GB のハードディスク領域。システムの一時的ディレクトリーは、Python の標準ライブラリーの tempfile モジュールで定義されるルールに基づいて決定されます。
  - 各システムは、システムプロバイダーの追加の要件を満たす必要があります。たとえば、クラスターを VMware vSphere にインストールしている場合、ディスクはその [ストレージガイドライン](#) に応じて設定され、**disk.enableUUID=true** 属性が設定される必要があります。
  - 各システムは、DNS で解決可能なホスト名を使用してクラスターの API エンドポイントにアクセスする必要があります。配置されているネットワークセキュリティアクセス制御は、クラスターの API サービスエンドポイントへのシステムアクセスを許可する必要があります。

- ノードの削除

### 9.2.1. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

## 9.3. クラウド用イメージの準備

各種のイメージ形式は AWS で直接使用できないので、Amazon Machine Images (AMI) が必要です。Red Hat が提供している AMI を使用するか、または独自のイメージを手動でインポートできます。EC2 インスタンスをプロビジョニングする前に AMI が存在している必要があります。コンピュータマシンに必要な正しい RHEL バージョンを選択するには、AMI ID を一覧表示する必要があります。

### 9.3.1. AWS で利用可能な最新の RHEL イメージの一覧表示

AMI ID は、AWS のネイティブブートイメージに対応します。EC2 インスタンスがプロビジョニングされる前に AMI が存在している必要があるため、設定前に AMI ID を把握しておく必要があります。[AWS コマンドラインインターフェイス \(CLI\)](#) は、利用可能な Red Hat Enterprise Linux (RHEL) イメージ ID の一覧を表示するために使用されます。

#### 前提条件

- AWS CLI をインストールしている。

#### 手順

- このコマンドを使用して、RHEL 8.4 Amazon Machine Images (AMI) の一覧を表示します。

```
$ aws ec2 describe-images --owners 309956199498 \ ❶
--query 'sort_by(Images, &CreationDate)[*].[CreationDate,Name,ImageId]' \ ❷
--filters "Name=name,Values=RHEL-8.4*" \ ❸
--region us-east-1 \ ❹
--output table ❺
```

- ❶ **--owners** コマンドオプションは、アカウント ID **309956199498** に基づいて Red Hat イメージを表示します。



#### 重要

Red Hat が提供するイメージの AMI ID を表示するには、このアカウント ID が必要です。

- ❷ **--query** コマンドオプションは、イメージが '**sort\_by(Images, &CreationDate)[\*].[CreationDate,Name,ImageId]**' のパラメーターでソートされる方法を設定します。この場合、イメージは作成日でソートされ、テーブルが作成日、イメージ名、および AMI ID を表示するように設定されます。

- 3 **--filter** コマンドオプションは、表示される RHEL のバージョンを設定します。この例では、フィルターが **"Name=name,Values=RHEL-8.4"** で設定されているため、RHEL 8.4 AMI が表示されます。
- 4 **--region** コマンドオプションは、AMI が保存されるリージョンを設定します。
- 5 **--output** コマンドオプションは、結果の表示方法を設定します。



### 注記

AWS 用の RHEL コンピュータマシンを作成する場合、AMI が RHEL 8.4 または 8.5 であることを確認します。

### 出力例

```
-----
|                               DescribelImages                               |
+-----+-----+-----+-----+-----+-----+
| 2021-03-18T14:23:11.000Z | RHEL-8.4.0_HVM_BETA-20210309-x86_64-1-Hourly2-GP2 | ami-07eeb4db5f7e5a8fb |
| 2021-03-18T14:38:28.000Z | RHEL-8.4.0_HVM_BETA-20210309-arm64-1-Hourly2-GP2 | ami-069d22ec49577d4bf |
| 2021-05-18T19:06:34.000Z | RHEL-8.4.0_HVM-20210504-arm64-2-Hourly2-GP2      | ami-01fc429821bf1f4b4 |
| 2021-05-18T20:09:47.000Z | RHEL-8.4.0_HVM-20210504-x86_64-2-Hourly2-GP2      | ami-0b0af3577fe5e3532 |
+-----+-----+-----+-----+-----+-----+

```

### 関連情報

- [RHEL イメージを AWS に手動でインポートする](#) こともできます。

## 9.4. RHEL コンピュータノードの準備

Red Hat Enterprise Linux (RHEL) マシンを OpenShift Container Platform クラスターに追加する前に、各ホストを Red Hat Subscription Manager (RHSM) に登録し、有効な OpenShift Container Platform サブスクリプションをアタッチし、必要なリポジトリを有効にする必要があります。

1. 各ホストで RHSM に登録します。

```
# subscription-manager register --username=<user_name> --password=<password>
```

2. RHSM から最新のサブスクリプションデータをプルします。

```
# subscription-manager refresh
```

3. 利用可能なサブスクリプションを一覧表示します。

```
# subscription-manager list --available --matches '*OpenShift*'
```

4. 直前のコマンドの出力で、OpenShift Container Platform サブスクリプションのプール ID を見つけ、これをアタッチします。

```
# subscription-manager attach --pool=<pool_id>
```

5. yum リポジトリをすべて無効にします。

- a. 有効にされている RHSM リポジトリをすべて無効にします。

```
# subscription-manager repos --disable="*"
```

- b. 残りの yum リポジトリを一覧表示し、**repo id** にあるそれらの名前をメモします (ある場合)。

```
# yum repolist
```

- c. **yum-config-manager** を使用して、残りの yum リポジトリを無効にします。

```
# yum-config-manager --disable <repo_id>
```

または、すべてのリポジトリを無効にします。

```
# yum-config-manager --disable *
```

利用可能なリポジトリが多い場合には、数分の時間がかかることがあります。

6. OpenShift Container Platform 4.11 で必要なリポジトリのみを有効にします。

```
# subscription-manager repos \
  --enable="rhel-8-for-x86_64-baseos-rpms" \
  --enable="rhel-8-for-x86_64-appstream-rpms" \
  --enable="rhocp-4.11-for-rhel-8-x86_64-rpms" \
  --enable="fast-datapath-for-rhel-8-x86_64-rpms"
```

7. ホストで firewalld を停止し、無効にします。

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



#### 注記

firewalld は、後で有効にすることはできません。これを実行する場合、ワーカー上の OpenShift Container Platform ログにはアクセスできません。

## 9.5. AWS での RHEL インスタンスへのロールパーミッションの割り当て

ブラウザーで Amazon IAM コンソールを使用して、必要なロールを選択し、ワーカーノードに割り当てることができます。

### 手順

1. AWS IAM コンソールから、[任意の IAM ロール](#) を作成します。
2. [IAM ロール](#) を必要なワーカーノードに割り当てます。

### 関連情報

- [Required AWS permissions for IAM roles](#) を参照してください。

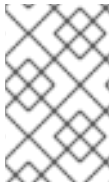
## 9.6. 所有または共有されている RHEL ワーカーノードへのタグ付け

クラスターは **kubernetes.io/cluster/<clusterid>,Value=(owned|shared)** タグの値を使用して、AWS クラスターに関するリソースの有効期間を判別します。

- リソースをクラスターの破棄の一環として破棄する必要がある場合は、**owned** タグの値を追加する必要があります。
- クラスターが破棄された後にリソースが引き続いて存在する場合、**shared** タグの値を追加する必要があります。このタグ付けは、クラスターがこのリソースを使用することを示しますが、リソースには別の所有者が存在します。

### 手順

- RHEL コンピュータマシンの場合、RHEL ワーカーマシンでは、**kubernetes.io/cluster/<clusterid>=owned** または **kubernetes.io/cluster/<cluster-id>=shared** でタグ付けする必要があります。



### 注記

すべての既存セキュリティグループに **kubernetes.io/cluster/<name>,Value=<clusterid>** のタグを付けないでください。その場合、Elastic Load Balancing (ELB) がロードバランサーを作成できなくなります。

## 9.7. RHEL コンピュータマシンのクラスターへのさらなる追加

Red Hat Enterprise Linux (RHEL) をオペレーティングシステムとして使用するコンピュータマシンを OpenShift Container Platform 4.11 クラスターにさらに追加することができます。

### 前提条件

- OpenShift Container Platform クラスターに RHEL コンピュータノードがすでに含まれています。
- 最初の RHEL コンピュータマシンをクラスターに追加するために使用した **hosts** ファイルは、Playbook を実行するマシン上にあります。
- Playbook を実行するマシンは RHEL ホストにアクセスする必要があります。Bastion と SSH プロキシまたは VPN の使用など、所属する会社で許可されるすべての方法を利用できます。
- クラスターの **kubeconfig** ファイルおよびクラスターのインストールに使用したインストールプログラムが Playbook の実行に使用するマシン上にあります。
- インストール用の RHEL ホストを準備する必要があります。
- すべての RHEL ホストへの SSH アクセスを持つユーザーを Playbook を実行するマシンで設定します。
- SSH キーベースの認証を使用する場合、キーを SSH エージェントで管理する必要があります。
- Playbook を実行するマシンに OpenShift CLI (**oc**) をインストールします。

## 手順

1. コンピュータマシンホストおよび必要な変数を定義する `/<path>/inventory/hosts` にある Ansible インベントリーファイルを開きます。
2. ファイルの `[new_workers]` セクションの名前を `[workers]` に変更します。
3. `[new_workers]` セクションをファイルに追加し、それぞれの新規ホストの完全修飾ドメイン名を定義します。ファイルは以下の例のようになります。

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

openshift_kubeconfig_path=~/.kube/config"

[workers]
mycluster-rhel8-0.example.com
mycluster-rhel8-1.example.com

[new_workers]
mycluster-rhel8-2.example.com
mycluster-rhel8-3.example.com
```

この例では、**mycluster-rhel8-0.example.com** および **mycluster-rhel8-1.example.com** マシンがクラスターにあり、**mycluster-rhel8-2.example.com** および **mycluster-rhel8-3.example.com** マシンを追加します。

4. Ansible Playbook ディレクトリーに移動します。

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

5. スケールアップ Playbook を実行します。

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

**1** `<path>` については、作成した Ansible インベントリーファイルへのパスを指定します。

## 9.8. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

### 前提条件

- マシンがクラスターに追加されています。

### 手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

## 出力例

NAME	STATUS	ROLES	AGE	VERSION
master-0	Ready	master	63m	v1.24.0
master-1	Ready	master	63m	v1.24.0
master-2	Ready	master	64m	v1.24.0

出力には作成したすべてのマシンが一覧表示されます。



## 注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

## 出力例

NAME	AGE	REQUESTOR	CONDITION
csr-8b2br	15m	system:serviceaccount:openshift-machine-config-operator:node-bootstrap	Pending
csr-8vnps	15m	system:serviceaccount:openshift-machine-config-operator:node-bootstrap	Pending
...			

この例では、2つのマシンがクラスターに参加しています。この一覧にはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。



## 注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後1時間以内に CSR を承認してください。1時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに3つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。



## 注記

ベアメタルおよび他のユーザーによってプロビジョニングされるインフラストラクチャーなどのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、**oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrap** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティを確認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

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

- 1** **<csr\_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ 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

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

## 出力例

```
NAME      AGE   REQUESTOR                                     CONDITION
csr-bfd72 5m26s system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv 5m26s system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

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

1 **<csr\_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

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

- すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

### 出力例

```
NAME      STATUS  ROLES  AGE  VERSION
master-0  Ready   master 73m  v1.24.0
master-1  Ready   master 73m  v1.24.0
master-2  Ready   master 74m  v1.24.0
worker-0  Ready   worker 11m  v1.24.0
worker-1  Ready   worker 11m  v1.24.0
```



### 注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

### 関連情報

- CSR の詳細は、[Certificate Signing Requests](#) を参照してください。

## 9.9. ANSIBLE ホストファイルの必須パラメーター

Red Hat Enterprise Linux (RHEL) コンピュータマシンをクラスターに追加する前に、以下のパラメーターを Ansible ホストファイルに定義する必要があります。

パラメーター	説明	値
<b>ansible_user</b>	パスワードなしの SSH ベースの認証を許可する SSH ユーザー。SSH キーベースの認証を使用する場合、キーを SSH エージェントで管理する必要があります。	システム上のユーザー名。デフォルト値は <b>root</b> です。
<b>ansible_become</b>	<b>ansible_user</b> の値が root ではない場合、 <b>ansible_become</b> を <b>True</b> に設定する必要があり、 <b>ansible_user</b> として指定するユーザーはパスワードなしの sudo アクセスが可能になるように設定される必要があります。	<b>True</b> 。値が <b>True</b> ではない場合、このパラメーターを指定したり、定義したりしないでください。

パラメーター	説明	値
<b>openshift_kubeconfig_path</b>	クラスターの <b>kubeconfig</b> ファイルが含まれるローカルディレクトリーへのパスおよびファイル名を指定します。	設定ファイルのパスと名前。

## 第10章 ユーザーがプロビジョニングしたインフラストラクチャーを手動で管理する

### 10.1. ユーザーがプロビジョニングしたインフラストラクチャーを使用してクラスターに計算マシンを手動で追加する

インストールプロセスの一環として、あるいはインストール後に、ユーザーによってプロビジョニングされるインフラストラクチャーのクラスターにコンピュータマシンを追加できます。インストール後のプロセスでは、インストール時に使用されたものと同じ設定ファイルおよびパラメーターの一部が必要です。

#### 10.1.1. コンピュータマシンの Amazon Web Services への追加

Amazon Web Services (AWS) 上の OpenShift Container Platform クラスターにコンピュータマシンを追加するには、[CloudFormation テンプレートの使用によるコンピュータマシンの AWS への追加](#) を参照してください。

#### 10.1.2. コンピュータマシンの Microsoft Azure への追加

Microsoft Azure 上の OpenShift Container Platform クラスターにコンピュータマシンを追加するには、[Creating additional worker machines in Azure](#) を参照してください。

#### 10.1.3. コンピュータマシンの Azure Stack Hub への追加

Azure Stack Hub 上の OpenShift Container Platform クラスターにコンピュータマシンを追加するには、[Creating additional worker machines in Azure Stack Hub](#) を参照してください。

#### 10.1.4. コンピュータマシンの Google Cloud Platform への追加

Google Cloud Platform (GCP) 上の OpenShift Container Platform クラスターにコンピュータマシンを追加するには、[Creating additional worker machines in GCP](#) を参照してください。

#### 10.1.5. コンピュータマシンの vSphere への追加

[コンピューティングマシンセットを使用して](#)、vSphere 上の OpenShift Container Platform クラスター用の追加のコンピューティングマシンの作成を自動化できます。

クラスターにコンピューティングマシンを手動で追加するには、[コンピューティングマシンを vSphere に手動で追加する](#) を参照してください。

#### 10.1.6. RHV へのコンピュータマシンの追加

RHV 上の OpenShift Container Platform クラスターにコンピュータマシンをさらに追加するには、[Adding compute machines to RHV](#) を参照してください。

#### 10.1.7. コンピュータマシンのベアメタルへの追加

ベアメタル上の OpenShift Container Platform クラスターにコンピュータマシンを追加するには、[コンピュータマシンのベアメタルへの追加](#) を参照してください。

## 10.2. CLOUDFORMATION テンプレートの使用によるコンピュータマシンの AWS への追加

サンプルの CloudFormation テンプレートを使用して作成した Amazon Web Services (AWS) の OpenShift Container Platform クラスターにコンピュータマシンを追加することができます。

### 10.2.1. 前提条件

- 提供される [AWS CloudFormation テンプレート](#) を使用して AWS にクラスターをインストールしている。
- クラスターのインストール時にコンピュータマシンを作成するために使用した JSON ファイルおよび CloudFormation テンプレートがある。これらのファイルがない場合は、[インストール手順](#) に従ってこれらを作成する必要があります。

### 10.2.2. CloudFormation テンプレートの使用によるコンピュータマシンの AWS クラスターへの追加

サンプルの CloudFormation テンプレートを使用して作成した Amazon Web Services (AWS) の OpenShift Container Platform クラスターにコンピュータマシンを追加することができます。



#### 重要

CloudFormation テンプレートは、1つのコンピュータマシンを表すスタックを作成します。それぞれのコンピュータマシンにスタックを作成する必要があります。



#### 注記

提供される CloudFormation テンプレートを使用してコンピュータノードを作成しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせる必要がある可能性があります。

### 前提条件

- CloudFormation テンプレートを使用して OpenShift Container Platform クラスターをインストールし、クラスターのインストール時にコンピュータマシンの作成に使用した JSON ファイルおよび CloudFormation テンプレートにアクセスできる。
- AWS CLI をインストールしている。

### 手順

1. 別のコンピュータスタックを作成します。
  - a. テンプレートを起動します。

```
$ aws cloudformation create-stack --stack-name <name> \ ❶
--template-body file://<template>.yaml \ ❷
--parameters file://<parameters>.json ❸
```

- ❶ **<name>** は **cluster-workers** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前を指定する必要があります。

- 2 **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。
- 3 **<parameters>** は、CloudFormation パラメーター JSON ファイルへの相対パスまたは名前です。

b. テンプレートのコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

2. クラスターに作成するコンピューティングマシンが十分な数に達するまでコンピューティングスタックの作成を続けます。

### 10.2.3. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて2つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

#### 前提条件

- マシンがクラスターに追加されています。

#### 手順

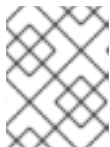
1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

#### 出力例

NAME	STATUS	ROLES	AGE	VERSION
master-0	Ready	master	63m	v1.24.0
master-1	Ready	master	63m	v1.24.0
master-2	Ready	master	64m	v1.24.0

出力には作成したすべてのマシンが一覧表示されます。



#### 注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

#### 出力例

NAME	AGE	REQUESTOR	CONDITION
csr-8b2br	15m	system:serviceaccount:openshift-machine-config-operator:node-bootstrapper	Pending
csr-8vnps	15m	system:serviceaccount:openshift-machine-config-operator:node-bootstrapper	Pending
...			

この例では、2つのマシンがクラスターに参加しています。この一覧にはさらに多くの承認された CSR が表示される可能性があります。

- 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

### 注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

### 注記

ベアメタルおよび他のユーザーによってプロビジョニングされるインフラストラクチャーなどのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、**oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

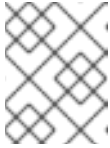
- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

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

❶ **<csr\_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ 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**

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

- クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

**出力例**

```
NAME      AGE  REQUESTOR                                CONDITION
csr-bfd72 5m26s system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv 5m26s system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

- 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。
  - それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

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

**1** **<csr\_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

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

- すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

**出力例**

```
NAME      STATUS  ROLES  AGE  VERSION
master-0  Ready   master  73m  v1.24.0
master-1  Ready   master  73m  v1.24.0
master-2  Ready   master  74m  v1.24.0
worker-0  Ready   worker  11m  v1.24.0
worker-1  Ready   worker  11m  v1.24.0
```

**注記**

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

## 関連情報

- CSR の詳細は、[Certificate Signing Requests](#) を参照してください。

## 10.3. コンピューティングマシンを VSPHERE に手動で追加する

コンピュータマシンを VMware vSphere の OpenShift Container Platform クラスターに追加することができます。



### 注記

また、[コンピューティングマシンセット](#)を使用して クラスター用の追加の VMware vSphere コンピュータマシンの作成を自動化することもできます。

### 10.3.1. 前提条件

- [クラスターを vSphere にインストールしている](#)。
- クラスターの作成に使用したインストールメディアおよび Red Hat Enterprise Linux CoreOS (RHCOS) イメージがある。これらのファイルがない場合は、[インストール手順](#) に従ってこれらを取得する必要があります。



### 重要

クラスターの作成に使用された Red Hat Enterprise Linux CoreOS (RHCOS) イメージへのアクセスがない場合、より新しいバージョンの Red Hat Enterprise Linux CoreOS (RHCOS) イメージと共にコンピュータマシンを OpenShift Container Platform クラスターに追加できます。手順については、[OpenShift 4.6+ へのアップグレード後の新規ノードの UPI クラスターへの追加の失敗](#) について参照してください。

### 10.3.2. vSphere でのコンピュータマシンのクラスターへの追加

コンピュータマシンを VMware vSphere のユーザーがプロビジョニングした OpenShift Container Platform クラスターに追加することができます。

#### 前提条件

- コンピュータマシンの base64 でエンコードされた Ignition ファイルを取得します。
- クラスター用に作成した vSphere テンプレートにアクセスできる必要があります。

#### 手順

1. テンプレートがデプロイされた後に、マシンの仮想マシンをクラスターにデプロイします。
  - a. テンプレートの名前を右クリックし、**Clone → Clone to Virtual Machine**をクリックします。
  - b. **Select a name and folder**タブで、仮想マシンの名前を指定します。**compute-1** などのように、マシンタイプを名前に含めることができるかもしれません。



### 注記

vSphere インストール全体のすべての仮想マシン名が一意であることを確認してください。

- c. **Select a name and folder** タブで、クラスターに作成したフォルダーの名前を選択します。
  - d. **Select a compute resource** タブで、データセンター内のホストの名前を選択します。
  - e. **Select clone options** で、**Customize this virtual machine's hardware** を選択します。
  - f. **Customize hardware** タブで、**VM Options** → **Advanced** をクリックします。
    - **Edit Configuration** をクリックし、**Configuration Parameters** ウィンドウで **Add Configuration Params** をクリックします。以下のパラメーター名および値を定義します。
      - **guestinfo.ignition.config.data**: このマシンファイルの base64 でエンコードしたコンピュート Ignition 設定ファイルの内容を貼り付けます。
      - **guestinfo.ignition.config.data.encoding**: **base64** を指定します。
      - **disk.EnableUUID**: **TRUE** を指定します。
  - g. **Customize hardware** タブの **Virtual Hardware** パネルで、必要に応じて指定した値を変更します。RAM、CPU、およびディスクストレージの量がマシンタイプの最小要件を満たすことを確認してください。また、ネットワークが複数利用可能な場合は、必ず **Add network adapter** に正しいネットワークを選択してください。
  - h. 設定を完了し、仮想マシンの電源をオンにします。
2. 継続してクラスター用の追加のコンピュートマシンを作成します。

### 10.3.3. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて2つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

#### 前提条件

- マシンがクラスターに追加されています。

#### 手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

#### 出力例

```
NAME      STATUS    ROLES    AGE   VERSION
master-0  Ready     master   63m   v1.24.0
```

```
master-1 Ready   master 63m v1.24.0
master-2 Ready   master 64m v1.24.0
```

出力には作成したすべてのマシンが一覧表示されます。



### 注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

### 出力例

```
NAME      AGE   REQUESTOR                                     CONDITION
csr-8b2br 15m   system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper Pending
csr-8vnps 15m   system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper Pending
...
```

この例では、2つのマシンがクラスターに参加しています。この一覧にはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。



### 注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。



## 注記

ベアメタルおよび他のユーザーによってプロビジョニングされるインフラストラクチャーなどのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、**oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrap** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティを確認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

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

❶ **<csr\_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ 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

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

## 出力例

```
NAME      AGE   REQUESTOR                                     CONDITION
csr-bfd72 5m26s system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv 5m26s system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

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

1 **<csr\_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

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

- すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

### 出力例

NAME	STATUS	ROLES	AGE	VERSION
master-0	Ready	master	73m	v1.24.0
master-1	Ready	master	73m	v1.24.0
master-2	Ready	master	74m	v1.24.0
worker-0	Ready	worker	11m	v1.24.0
worker-1	Ready	worker	11m	v1.24.0



### 注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

### 関連情報

- CSR の詳細は、[Certificate Signing Requests](#) を参照してください。

## 10.4. RHV 上のクラスターへのコンピュータマシンの追加

OpenShift Container Platform バージョン 4.11 では、RHV 上でユーザーがプロビジョニングした OpenShift Container Platform クラスターにコンピュータマシンをさらに追加できます。

### 前提条件

- ユーザーによってプロビジョニングされたインフラストラクチャーを使用して、RHV 上にクラスターをインストールしている。

#### 10.4.1. RHV 上のクラスターへのコンピュータマシンの追加

### 手順

- inventory.yml** ファイルを変更して、新規ワーカーを含めます。
- create-templates-and-vms** Ansible Playbook を実行して、ディスクと仮想マシンを作成します:

```
$ ansible-playbook -i inventory.yml create-templates-and-vms.yml
```

- worker.yml** Ansible Playbook を実行して、仮想マシンを起動します:

```
$ ansible-playbook -i inventory.yml workers.yml
```

4. クラスターに結合する新規ワーカーの CSR は、管理者によって承認される必要があります。次のコマンドは、保留中のすべてのリクエストを承認するのに役立ちます。

```
$ oc get csr -ojson | jq -r '.items[] | select(.status == {} ) | .metadata.name' | xargs oc adm certificate approve
```

## 10.5. コンピュータマシンのベアメタルへの追加

ベアメタルの OpenShift Container Platform クラスターにコンピュータマシンを追加することができます。

### 10.5.1. 前提条件

- [クラスターをベアメタルにインストールしている](#)。
- クラスターの作成に使用したインストールメディアおよび Red Hat Enterprise Linux CoreOS (RHCOS) イメージがある。これらのファイルがない場合は、[インストール手順](#)に従ってこれらを取得する必要があります。
- ユーザーがプロビジョニングするインフラストラクチャーに DHCP サーバーを利用できる場合には追加のコンピュータマシンの詳細を DHCP サーバー設定に追加している。これには、永続的な IP アドレス、DNS サーバー情報、および各マシンのホスト名が含まれます。
- 追加する各コンピュータマシンのレコード名と IP アドレスを追加するように DNS 設定を更新している。DNS ルックアップおよび逆引き DNS ルックアップが正しく解決されていることを検証している。



#### 重要

クラスターの作成に使用された Red Hat Enterprise Linux CoreOS (RHCOS) イメージへのアクセスがない場合、より新しいバージョンの Red Hat Enterprise Linux CoreOS (RHCOS) イメージと共にコンピュータマシンを OpenShift Container Platform クラスターに追加できます。手順については、[OpenShift 4.6+ へのアップグレード後の新規ノードの UPI クラスターへの追加の失敗](#)について参照してください。

### 10.5.2. Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成

ベアメタルインフラストラクチャーにインストールされているクラスターにコンピュータマシンを追加する前に、それが使用する RHCOS マシンを作成する必要があります。ISO イメージまたはネットワーク PXE ブートを使用してマシンを作成できます。



#### 注記

クラスターに新しいノードをすべてデプロイするには、クラスターのインストールに使用した ISO イメージと同じ ISO イメージを使用する必要があります。同じ Ignition 設定ファイルを使用することが推奨されます。ノードは、ワークロードを実行する前に初回起動時に自動的にアップグレードされます。アップグレードの前後にノードを追加することができます。

#### 10.5.2.1. ISO イメージを使用した追加の RHCOS マシンの作成

ISO イメージを使用して、ベアメタルクラスターの追加の Red Hat Enterprise Linux CoreOS (RHCOS) コンピュータマシンを作成できます。

## 前提条件

- クラスターのコンピュータマシンの Ignition 設定ファイルの URL を取得します。このファイルがインストール時に HTTP サーバーにアップロードされている必要があります。

## 手順

1. ISO ファイルを使用して、追加のコンピュータマシンに RHCOS をインストールします。クラスターのインストール前にマシンを作成する際に使用したのと同じ方法を使用します。
  - ディスクに ISO イメージを書き込み、これを直接起動します。
  - LOM インターフェイスで ISO リダイレクトを使用します。
2. オプションを指定したり、ライブ起動シーケンスを中断したりせずに、RHCOS ISO イメージを起動します。インストーラーが RHCOS ライブ環境でシェルプロンプトを起動するのを待ちます。



### 注記

RHCOS インストールの起動プロセスを中断して、カーネル引数を追加できます。ただし、この ISO 手順では、カーネル引数を追加する代わりに、次の手順で概説するように **coreos-installer** コマンドを使用する必要があります。

3. **coreos-installer** コマンドを実行し、インストール要件を満たすオプションを指定します。少なくとも、ノードタイプの Ignition 設定ファイルを参照する URL と、インストール先のデバイスを指定する必要があります。

```
$ sudo coreos-installer install --ignition-url=http://<HTTP_server>/<node_type>.ign <device>
--ignition-hash=sha512-<digest> ① ②
```

- ① コア ユーザーにはインストールを実行するために必要な root 権限がないため、**sudo** を使用して **coreos-installer** コマンドを実行する必要があります。

- ② **--ignition-hash** オプションは、Ignition 設定ファイルを HTTP URL を使用して取得し、クラスターノードの Ignition 設定ファイルの信頼性を検証するために必要です。**<digest>** は、先の手順で取得した Ignition 設定ファイル SHA512 ダイジェストです。



### 注記

TLS を使用する HTTPS サーバーを使用して Ignition 設定ファイルを提供する必要がある場合は、**coreos-installer** を実行する前に内部認証局 (CA) をシステム信頼ストアに追加できます。

以下の例では、**/dev/sda** デバイスへのブートストラップノードのインストールを初期化します。ブートストラップノードの Ignition 設定ファイルは、IP アドレス 192.168.1.2 で HTTP Web サーバーから取得されます。

```
$ sudo coreos-installer install --ignition-
url=http://192.168.1.2:80/installation_directory/bootstrap.ign /dev/sda --ignition-hash=sha512-
```

```
a5a2d43879223273c9b60af66b44202a1d1248fc01cf156c46d4a79f552b6bad47bc8cc78ddf0116e80c59d2ea9e32ba53bc807afbca581aa059311def2c3e3b
```

4. マシンのコンソールで RHCOS インストールの進捗を監視します。



### 重要

OpenShift Container Platform のインストールを開始する前に、各ノードでインストールが成功していることを確認します。インストールプロセスを監視すると、発生する可能性のある RHCOS インストールの問題の原因を特定する上でも役立ちます。

5. 継続してクラスター用の追加のコンピュータマシンを作成します。

## 10.5.2.2. PXE または iPXE ブートによる追加の RHCOS マシンの作成

PXE または iPXE ブートを使用して、ベアメタルクラスターの追加の Red Hat Enterprise Linux CoreOS (RHCOS) コンピュータマシンを作成できます。

### 前提条件

- クラスターのコンピュータマシンの Ignition 設定ファイルの URL を取得します。このファイルがインストール時に HTTP サーバーにアップロードされている必要があります。
- クラスターのインストール時に HTTP サーバーにアップロードした RHCOS ISO イメージ、圧縮されたメタル BIOS、**kernel**、および **initramfs** ファイルの URL を取得します。
- インストール時に OpenShift Container Platform クラスターのマシンを作成するために使用した PXE ブートインフラストラクチャーにアクセスできる必要があります。RHCOS のインストール後にマシンはローカルディスクから起動する必要があります。
- UEFI を使用する場合、OpenShift Container Platform のインストール時に変更した **grub.conf** ファイルにアクセスできます。

### 手順

1. RHCOS イメージの PXE または iPXE インストールが正常に行われていることを確認します。

- PXE の場合:

```
DEFAULT pxeboot
TIMEOUT 20
PROMPT 0
LABEL pxeboot
  KERNEL http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> ❶
  APPEND initrd=http://<HTTP_server>/rhcos-<version>-live-initramfs.
<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/worker.ign
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
<architecture>.img ❷
```

❶ HTTP サーバーにアップロードしたライブ **kernel** ファイルの場所を指定します。

❷ HTTP サーバーにアップロードした RHCOS ファイルの場所を指定します。 **initrd** パ

この設定では、グラフィカルコンソールを使用するマシンでシリアルコンソールアクセスを有効にしません。別のコンソールを設定するには、**APPEND** 行に1つ以上の **console=** 引数を追加します。たとえば、**console=tty0 console=ttyS0** を追加して、最初の PC シリアルポートをプライマリーコンソールとして、グラフィカルコンソールをセカンダリーコンソールとして設定します。詳細は、[How does one set up a serial terminal and/or console in Red Hat Enterprise Linux?](#) を参照してください。

- iPXE の場合:

```
kernel http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> initrd=main
coreos.inst.install_dev=/dev/sda coreos.inst.ignition_url=http://<HTTP_server>/worker.ign
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.<architecture>.img
```

1

```
initrd --name main http://<HTTP_server>/rhcos-<version>-live-initramfs.<architecture>.img
```

2

- 1 HTTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**kernel** パラメーター値は **kernel** ファイルの場所であり、**initrd=main** 引数は UEFI システムでの起動に必要であり、**coreos.inst.ignition\_url** パラメーター値はワーカー Ignition 設定ファイルの場所であり、**coreos.live.rootfs\_url** パラメーター値は **rootfs** のライブファイルの場所です。**coreos.inst.ignition\_url** および **coreos.live.rootfs\_url** パラメーターは HTTP および HTTPS のみをサポートします。

- 2 HTTP サーバーにアップロードした **initramfs** ファイルの場所を指定します。

この設定では、グラフィカルコンソールを使用するマシンでシリアルコンソールアクセスを有効にしません。別のコンソールを設定するには、**kernel** 行に **console=** 引数を1つ以上追加します。たとえば、**console=tty0 console=ttyS0** を追加して、最初の PC シリアルポートをプライマリーコンソールとして、グラフィカルコンソールをセカンダリーコンソールとして設定します。詳細は、[How does one set up a serial terminal and/or console in Red Hat Enterprise Linux?](#) を参照してください。

1. PXE または iPXE インフラストラクチャーを使用して、クラスターに必要なコンピュートマシンを作成します。

### 10.5.3. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて2つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

#### 前提条件

- マシンがクラスターに追加されています。

#### 手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

#### 出力例

```
NAME      STATUS    ROLES    AGE  VERSION
```

```
master-0 Ready    master 63m v1.24.0
master-1 Ready    master 63m v1.24.0
master-2 Ready    master 64m v1.24.0
```

出力には作成したすべてのマシンが一覧表示されます。



### 注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

### 出力例

```
NAME      AGE  REQUESTOR                                     CONDITION
csr-8b2br  15m  system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper  Pending
csr-8vnps  15m  system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper  Pending
...
```

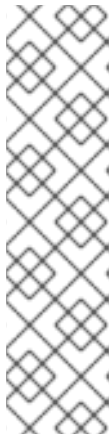
この例では、2つのマシンがクラスターに参加しています。この一覧にはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。



### 注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後1時間以内に CSR を承認してください。1時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに3つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。



## 注記

ベアメタルおよび他のユーザーによってプロビジョニングされるインフラストラクチャーなどのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、**oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrap** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティを確認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

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

- ❶ **<csr\_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ 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

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

- クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

## 出力例

```
NAME      AGE   REQUESTOR                                     CONDITION
csr-bfd72 5m26s system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv 5m26s system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

- 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

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

1 **<csr\_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

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

- すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

### 出力例

NAME	STATUS	ROLES	AGE	VERSION
master-0	Ready	master	73m	v1.24.0
master-1	Ready	master	73m	v1.24.0
master-2	Ready	master	74m	v1.24.0
worker-0	Ready	worker	11m	v1.24.0
worker-1	Ready	worker	11m	v1.24.0



### 注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

### 関連情報

- CSR の詳細は、[Certificate Signing Requests](#) を参照してください。

## 第11章 CLUSTER API によるマシンの管理



### 重要

Cluster API を使用したマシン管理は、テクノロジープレビュー機能のみです。テクノロジープレビュー機能は、Red Hat 製品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ことがあります。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。

[Cluster API](#) は、Amazon Web Services (AWS) および Google Cloud Platform (GCP) クラスターのテクノロジープレビューとして OpenShift Container Platform に統合されるアップストリームプロジェクトです。クラスター API を使用して、OpenShift Container Platform クラスターでマシンセットとマシンを作成および管理できます。この機能は、Machine API を使用してマシンを管理するための追加または代替の機能になります。

OpenShift Container Platform 4.11 クラスターの場合、クラスター API を使用して、クラスターのインストールが完了した後にノードホストのプロビジョニング管理アクションを実行できます。このシステムにより、パブリックまたはプライベートクラウドインフラストラクチャー上で柔軟かつ動的な方法でプロビジョニングできます。

Cluster API テクノロジープレビューを使用すると、サポートされているプロバイダーの OpenShift Container Platform クラスター上にコンピュータマシンおよびマシンセットを作成できます。Machine API では利用できない可能性がある、この実装によって有効になる機能も確認できます。

### 利点

Cluster API を使用することで、OpenShift Container Platform ユーザーおよび開発者には以下の利点がもたらされます。

- Machine API でサポートされていない可能性があるアップストリームコミュニティの Cluster API インフラストラクチャープロバイダーを使用するオプション。
- インフラストラクチャープロバイダーのマシンコントローラーを保守するサードパーティーと協力する機会。
- OpenShift Container Platform でのインフラストラクチャー管理に一連の同じ Kubernetes ツールを使用する機能。
- Machine API では利用できない機能をサポートする Cluster API を使用してマシンセットを作成する機能。

### 制限事項

Cluster API を使用したマシン管理はテクノロジープレビュー機能であり、次の制限があります。

- サポート対象は AWS および GCP クラスターのみです。
- この機能を使用するには、**TechPreviewNoUpgrade 機能セット** を有効にする必要があります。この機能セットを有効にすると元に戻すことができなくなり、マイナーバージョン更新ができなくなります。
- クラスター API が必要とするプライマリーリソースを手動で作成する必要があります。

- コントロールプレーンマシンは、Cluster API では管理できません。
- Machine API によって作成された既存のマシンセットの、Cluster API マシンセットへの移行はサポートされていません。
- Machine API との完全な機能パリティは利用できません。

## 11.1. CLUSTER API アーキテクチャー

アップストリーム Cluster API の OpenShift Container Platform 統合は、Cluster CAPI Operator によって実装および管理されます。Cluster CAPI Operator とそのオペランドは、**openshift-machine-api** namespace を使用する Machine API とは対照的に、**openshift-cluster-api** namespace でプロビジョニングされます。

### 11.1.1. Cluster CAPI Operator

Cluster CAPI Operator は、Cluster API リソースのライフサイクルを維持する OpenShift Container Platform Operator です。この Operator は、OpenShift Container Platform クラスター内での Cluster API プロジェクトのデプロイに関連するすべての管理タスクを行います。

Cluster API の使用を許可するようにクラスターが正しく設定されている場合、Cluster CAPI Operator は Cluster API Operator をクラスターにインストールします。



#### 注記

Cluster CAPI Operator は、アップストリームの Cluster API Operator とは異なります。

詳細については、**Cluster Operators** リファレンス コンテンツの Cluster CAPI Operator のエントリーを参照してください。

### 11.1.2. プライマリーリソース

Cluster API は、次のプライマリーリソースで設定されています。この機能のテクノロジープレビューでは、**openshift-cluster-api** namespace でこれらのリソースを手動で作成する必要があります。

#### クラスター

Cluster API によって管理されるクラスターを表す基本単位。

#### インフラストラクチャー

リージョンやサブネットなど、クラスター内のすべてのマシンセットで共有されるプロパティを定義するプロバイダー固有のリソース。

#### マシンテンプレート

マシンセットが作成するマシンのプロパティを定義するプロバイダー固有のテンプレート。

#### マシンセット

マシンのグループ。

マシンセットとマシンの関係は、レプリカセットと Pod の関係と同様です。マシンを追加する必要がある場合や、マシンの数を縮小したりする必要がある場合、コンピューティングのニーズに応じてマシンセットの **replicas** フィールドを変更します。

Cluster API を使用すると、マシンセットは **Cluster** オブジェクトとプロバイダー固有のマシンテンプレートを参照します。

#### マシン

ノードのホストを記述する基本的なユニットです。  
Cluster API は、マシンテンプレートの設定に基づいてマシンを作成します。

## 関連情報

- [Cluster CAPI Operator](#)

## 11.2. サンプル YAML ファイル

Cluster API テクノロジープレビューの場合、Cluster API が必要とするプライマリーリソースを手動で作成する必要があります。このセクションのサンプル YAML ファイルは、これらのリソースを連携させて、それらが作成するマシンを環境に応じて設定する方法を示しています。

### 11.2.1. Cluster API クラスターリソースのサンプル YAML

クラスターリソースは、クラスターの名前とインフラストラクチャプロバイダーを定義し、Cluster API によって管理されます。このリソースは、すべてのプロバイダーで同じ構造を持っています。

```
apiVersion: cluster.x-k8s.io/v1beta1
kind: Cluster
metadata:
  name: <cluster_name> ❶
  namespace: openshift-cluster-api
spec:
  infrastructureRef:
    apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
    kind: <infrastructure_kind> ❷
    name: <cluster_name> ❸
    namespace: openshift-cluster-api
```

❶❸ クラスターの名前を指定します。

❷ クラスターのインフラストラクチャーの種類を指定します。有効な値は以下のとおりです。

- **AWSCluster**: クラスターは Amazon Web Services (AWS) で実行されています。
- **GCPCluster**: クラスターは Google Cloud Platform (GCP) で実行されています。

残りはプロバイダー固有の Cluster API リソースです。クラスターのサンプル YAML ファイルを参照してください。

- [Amazon Web Services クラスターを設定するサンプル YAML ファイル](#)
- [Google Cloud Platform クラスターを設定するサンプル YAML ファイル](#)

### 11.2.2. Amazon Web Services クラスターを設定するサンプル YAML ファイル

一部の Cluster API リソースはプロバイダー固有です。このセクションのサンプル YAML ファイル、Amazon Web Services (AWS) クラスターの設定を示しています。

#### 11.2.2.1. Amazon Web Services 上の Cluster API インフラストラクチャーリソースのサンプル YAML

インフラストラクチャリソースはプロバイダー固有であり、リージョンやサブネットなど、クラスター内のすべてのマシンセットで共有されるプロパティを定義します。マシンセットは、マシン作成時にこのリソースを参照します。

```
apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: AWSCluster 1
metadata:
  name: <cluster_name> 2
  namespace: openshift-cluster-api
spec:
  region: <region> 3
```

- 1** クラスターのインフラストラクチャーの種類を指定します。この値は、プラットフォームの値と一致する必要があります。
- 2** クラスターの名前を指定します。
- 3** AWS リージョンを指定します。

#### 11.2.2.2. Amazon Web Services の Cluster API マシンテンプレートリソースのサンプル YAML

マシンテンプレートリソースはプロバイダー固有であり、マシンセットが作成するマシンの基本的なプロパティを定義します。マシンセットは、マシン作成時にこのテンプレートを参照します。

```
apiVersion: infrastructure.cluster.x-k8s.io/v1alpha4
kind: AWSMachineTemplate 1
metadata:
  name: <template_name> 2
  namespace: openshift-cluster-api
spec:
  template:
    spec: 3
      uncompressedUserData: true
      iamInstanceProfile: ....
      instanceType: m5.large
      cloudInit:
        insecureSkipSecretsManager: true
      ami:
        id: ....
      subnet:
        filters:
          - name: tag:Name
            values:
              - ...
      additionalSecurityGroups:
        - filters:
            - name: tag:Name
              values:
                - ...
```

- 1** マシンテンプレートの種類を指定します。この値は、プラットフォームの値と一致する必要があります。

- 2 マシンテンプレートの名前を指定します。
- 3 環境の詳細を指定します。ここに示す値はサンプルです。

### 11.2.2.3. Amazon Web Services の Cluster API マシンセットリソースのサンプル YAML

マシンセットリソースは、作成するマシンの追加プロパティを定義します。マシンセットは、マシン作成時にインフラストラクチャーリソースとマシンテンプレートも参照します。

```
apiVersion: cluster.x-k8s.io/v1alpha4
kind: MachineSet
metadata:
  name: <machine_set_name> 1
  namespace: openshift-cluster-api
spec:
  clusterName: <cluster_name> 2
  replicas: 1
  selector:
    matchLabels:
      test: example
  template:
    metadata:
      labels:
        test: example
    spec:
      bootstrap:
        dataSecretName: worker-user-data 3
        clusterName: <cluster_name> 4
      infrastructureRef:
        apiVersion: infrastructure.cluster.x-k8s.io/v1alpha4
        kind: AWSMachineTemplate 5
        name: <cluster_name> 6
```

- 1 マシンセットの名前を指定します。
- 2 4 6 クラスターの名前を指定します。
- 3 Cluster API テクノロジープレビューの場合、Operator は **openshift-machine-api** namespace のワーカーユーザーデータシークレットを使用できます。
- 5 マシンテンプレートの種類を指定します。この値は、プラットフォームの値と一致する必要があります。

### 11.2.3. Google Cloud Platform クラスターを設定するサンプル YAML ファイル

一部の Cluster API リソースはプロバイダー固有です。このセクションのサンプル YAML ファイルは、Google Cloud Platform (GCP) クラスターの設定を示しています。

#### 11.2.3.1. Google Cloud Platform 上の Cluster API インフラストラクチャーリソースのサンプル YAML

このセクションのサンプル YAML ファイルは、Google Cloud Platform (GCP) クラスターを設定するための Cluster API インフラストラクチャーリソースのサンプル YAML ファイルを示しています。

インフラストラクチャーリソースはプロバイダー固有であり、リージョンやサブネットなど、クラスター内のすべてのマシンセットで共有されるプロパティを定義します。マシンセットは、マシン作成時にこのリソースを参照します。

```
apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: GCPCluster ❶
metadata:
  name: <cluster_name> ❷
spec:
  network:
    name: <cluster_name>-network ❸
  project: <project> ❹
  region: <region> ❺
```

- ❶ クラスターのインフラストラクチャーの種類を指定します。この値は、プラットフォームの値と一致する必要があります。
- ❷ ❸ クラスターの名前を指定します。
- ❹ GCP プロジェクト名を指定します。
- ❺ GCP リージョンを指定します。

### 11.2.3.2. Google Cloud Platform 上の Cluster API マシンテンプレートリソースのサンプル YAML

マシンテンプレートリソースはプロバイダー固有であり、マシンセットが作成するマシンの基本的なプロパティを定義します。マシンセットは、マシン作成時にこのテンプレートを参照します。

```
apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: GCPMachineTemplate ❶
metadata:
  name: <template_name> ❷
  namespace: openshift-cluster-api
spec:
  template:
    spec: ❸
      rootDeviceType: pd-ssd
      rootDeviceSize: 128
      instanceType: n1-standard-4
      image: projects/rhcos-cloud/global/images/rhcos-411-85-202203181601-0-gcp-x86-64
      subnet: <cluster_name>-worker-subnet
      serviceAccounts:
        email: <service_account_email_address>
        scopes:
          - https://www.googleapis.com/auth/cloud-platform
      additionalLabels:
        kubernetes-io-cluster-<cluster_name>: owned
      additionalNetworkTags:
        - <cluster_name>-worker
      ipForwarding: Disabled
```

- 1 マシンテンプレートの種類を指定します。この値は、プラットフォームの値と一致する必要があります。
- 2 マシンテンプレートの名前を指定します。
- 3 環境の詳細を指定します。ここに示す値はサンプルです。

### 11.2.3.3. Google Cloud Platform 上の Cluster API マシンセットリソースのサンプル YAML

マシンセットリソースは、作成するマシンの追加プロパティを定義します。マシンセットは、マシン作成時にインフラストラクチャーリソースとマシンテンプレートも参照します。

```
apiVersion: cluster.x-k8s.io/v1beta1
kind: MachineSet
metadata:
  name: <machine_set_name> 1
  namespace: openshift-cluster-api
spec:
  clusterName: <cluster_name> 2
  replicas: 1
  selector:
    matchLabels:
      test: test
  template:
    metadata:
      labels:
        test: test
    spec:
      bootstrap:
        dataSecretName: worker-user-data 3
        clusterName: <cluster_name> 4
      infrastructureRef:
        apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
        kind: GCPMachineTemplate 5
        name: <machine_set_name> 6
        failureDomain: <failure_domain> 7
```

- 1 6 マシンセットの名前を指定します。
- 2 4 クラスターの名前を指定します。
- 3 Cluster API テクノロジープレビューの場合、Operator は **openshift-machine-api** namespace のワーカーユーザーデータシークレットを使用できます。
- 5 マシンテンプレートの種類を指定します。この値は、プラットフォームの値と一致する必要があります。
- 7 GCP リージョン内の障害ドメインを指定します。

## 11.3. CLUSTER API マシンセットの作成

Cluster API を使用して、選択した特定のワークロードのマシンコンピューティングリソースを動的に管理するマシンセットを作成できます。

## 前提条件

- OpenShift Container Platform クラスターをデプロイすること。
- Cluster API の使用を有効にします。
- OpenShift CLI (**oc**) がインストールされている。
- **cluster-admin** パーミッションを持つユーザーとして、**oc** にログインする。

## 手順

1. クラスターカスタムリソース (CR) を含む、**<cluster\_resource\_file>.yaml** という名前の YAML ファイルを作成します。  
**<cluster\_name>** パラメーターに設定する値がわからない場合は、クラスターに設定されている既存の Machine API マシンの値を確認してください。

- a. Machine API マシンセットを一覧表示するには、次のコマンドを実行します。

```
$ oc get machinesets -n openshift-machine-api ❶
```

- ❶ **openshift-machine-api** namespace を指定します。

## 出力例

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

- b. 特定のマシンセット CR の内容を表示するには、次のコマンドを実行します。

```
$ oc get machineset <machineset_name> \
-n openshift-machine-api \
-o yaml
```

## 出力例

```
...
template:
  metadata:
    labels:
      machine.openshift.io/cluster-api-cluster: agl030519-vplxk ❶
      machine.openshift.io/cluster-api-machine-role: worker
      machine.openshift.io/cluster-api-machine-type: worker
      machine.openshift.io/cluster-api-machineset: agl030519-vplxk-worker-us-east-1a
...
```

1 **<cluster\_name>** パラメーターに使用するクラスター ID。

2. 次のコマンドを実行して、クラスターを作成します。

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

### 検証

クラスター CR が作成されたことを確認するには、次のコマンドを実行します。

```
$ oc get cluster
```

### 出力例

```
NAME          PHASE    AGE  VERSION
<cluster_name> Provisioning 4h6m
```

3. インフラストラクチャー CR を含む、**<infrastructure\_resource\_file>.yaml** という名前の YAML ファイルを作成します。
4. 次のコマンドを実行して、インフラストラクチャー CR を作成します。

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

### 検証

インフラストラクチャー CR が作成されたことを確認するには、次のコマンドを実行します。

```
$ oc get <infrastructure_kind>
```

**<infrastructure\_kind>** は、プラットフォームに対応する値です。

### 出力例

```
NAME          CLUSTER    READY VPC BASTION IP
<cluster_name> <cluster_name> true
```

5. マシンテンプレート CR を含む、**<machine\_template\_resource\_file>.yaml** という名前の YAML ファイルを作成します。
6. 次のコマンドを実行して、マシンテンプレート CR を作成します。

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

### 検証

マシンテンプレート CR が作成されたことを確認するには、次のコマンドを実行します。

```
$ oc get <machine_template_kind>
```

**<machine\_template\_kind>** は、プラットフォームに対応する値です。

### 出力例

```
NAME          AGE
<template_name> 77m
```

7. マシンセット CR を含む、**<machine\_set\_resource\_file>.yaml** という名前の YAML ファイルを作成します。
8. 次のコマンドを実行して、マシンセット CR を作成します。

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

## 検証

マシンセット CR が作成されたことを確認するには、次のコマンドを実行します。

```
$ oc get machineset -n openshift-cluster-api ❶
```

- ❶ **openshift-cluster-api** namespace を指定します。

## 出力例

```
NAME          CLUSTER    REPLICAS READY AVAILABLE AGE VERSION
<machine_set_name> <cluster_name> 1      1      1      17m
```

新しいマシンセットが利用可能な場合、**REPLICAS** と **AVAILABLE** の値が一致します。マシンセットが利用可能でない場合、数分待機してからコマンドを再度実行します。

## 検証

- マシンセットが指定した設定に従ってマシンを作成していることを確認するには、クラスター内のマシンとノードのリストを確認します。
  - Cluster API マシンのリストを表示するには、次のコマンドを実行します。

```
$ oc get machine -n openshift-cluster-api ❶
```

- ❶ **openshift-cluster-api** namespace を指定します。

## 出力例

```
NAME          CLUSTER    NODENAME          PROVIDERID
PHASE AGE VERSION
<machine_set_name>-<string_id> <cluster_name> <ip_address>.
<region>.compute.internal <provider_id> Running 8m23s
```

- ノードのリストを表示するには、次のコマンドを実行します。

```
$ oc get node
```

## 出力例

```
NAME          STATUS ROLES AGE VERSION
```

```
<ip_address_1>.<region>.compute.internal Ready worker 5h14m v1.24.0+284d62a  
<ip_address_2>.<region>.compute.internal Ready master 5h19m v1.24.0+284d62a  
<ip_address_3>.<region>.compute.internal Ready worker 7m v1.24.0+284d62a
```

## 11.4. CLUSTER API を使用するクラスターのトラブルシューティング

このセクションの情報を使用して、発生する可能性のある問題を理解し、回復してください。通常、Cluster API に関する問題のトラブルシューティング手順は、マシン API に関する問題の手順と似ています。

Cluster CAPI Operator とそのオペランドは **openshift-cluster-api** namespace でプロビジョニングされますが、マシン API は **openshift-machine-api** namespace を使用します。namespace を参照する **oc** コマンドを使用する場合は、必ず正しい namespace を参照してください。

### 11.4.1. CLI コマンドで返される Cluster API マシン

Cluster API を使用するクラスターの場合、**oc get machine** などの **oc** コマンドは、Cluster API マシンの結果を返します。**c** の文字はアルファベット順で **m** の前にあるため、Cluster API マシンは Machine API マシンより前に返されます。

- Machine API マシンのみを一覧表示するには、**oc get machine** コマンドを実行する際に、完全修飾名 **machines.machine.openshift.io** を使用します。

```
$ oc get machines.machine.openshift.io
```

- Cluster API マシンのみを一覧表示するには、**oc get machine** コマンドを実行する際に、完全修飾名 **machines.cluster.x-k8s.io** を使用します。

```
$ oc get machines.cluster.x-k8s.io
```

## 第12章 マシンヘルスチェックのデプロイ

マシンヘルスチェックを設定し、デプロイして、マシンプールにある破損したマシンを自動的に修復します。

### 重要

高度なマシン管理およびスケーリング機能は、マシン API が機能しているクラスターでのみ使用することができます。ユーザーがプロビジョニングしたインフラストラクチャーを持つクラスターでは、マシン API を使用するために追加の検証と設定が必要です。

インフラストラクチャープラットフォームタイプが **none** のクラスターは、マシン API を使用できません。この制限は、クラスターに接続されている計算マシンが、この機能をサポートするプラットフォームにインストールされている場合でも適用されます。このパラメーターは、インストール後に変更することはできません。

クラスターのプラットフォームタイプを表示するには、以下のコマンドを実行します。

```
$ oc get infrastructure cluster -o jsonpath='{.status.platform}'
```

### 12.1. マシンのヘルスチェック

マシンのヘルスチェックは特定のマシンプールの正常ではないマシンを自動的に修復します。

マシンの正常性を監視するには、リソースを作成し、コントローラーの設定を定義します。5 分間 **NotReady** ステータスにすることや、node-problem-detector に永続的な条件を表示すること、および監視する一連のマシンのラベルなど、チェックする条件を設定します。

### 注記

マスターロールのあるマシンにマシンヘルスチェックを適用することはできません。

**MachineHealthCheck** リソースを監視するコントローラーは定義済みのステータスをチェックします。マシンがヘルスチェックに失敗した場合、このマシンは自動的に検出され、その代わりとなるマシンが作成されます。マシンが削除されると、**machine deleted** イベントが表示されます。

マシンの削除による破壊的な影響を制限するために、コントローラーは1度に1つのノードのみをドレイン (解放) し、これを削除します。マシンのターゲットプールで許可される **maxUnhealthy** しきい値を上回る数の正常でないマシンがある場合、修復が停止するため、手動による介入が可能になります。

### 注記

タイムアウトについて注意深い検討が必要であり、ワークロードと要件を考慮してください。

- タイムアウトの時間が長くなると、正常でないマシンのワークロードのダウンタイムが長くなる可能性があります。
- タイムアウトが短すぎると、修復ループが生じる可能性があります。たとえば、**NotReady** ステータスを確認するためのタイムアウトについては、マシンが起動プロセスを完了できるように十分な時間を設定する必要があります。

チェックを停止するには、リソースを削除します。

### 12.1.1. マシンヘルスチェックのデプロイ時の制限

マシンヘルスチェックをデプロイする前に考慮すべき制限事項があります。

- マシンセットが所有するマシンのみがマシンヘルスチェックによって修復されます。
- コントロールプレーンマシンは現在サポートされておらず、それらが正常でない場合にも修正されません。
- マシンのノードがクラスターから削除される場合、マシンヘルスチェックはマシンが正常ではないとみなし、すぐにこれを修復します。
- **nodeStartupTimeout** の後にマシンの対応するノードがクラスターに加わらない場合、マシンは修復されます。
- **Machine** リソースフェーズが **Failed** の場合、マシンはすぐに修復されます。

#### 関連情報

- **MachineHealthCheck** CR で定義できるノードの状態についての詳細は、[About listing all the nodes in a cluster](#) を参照してください。
- 一時停止 (short-circuiting) についての詳細は、[マシンヘルスチェックによる修復の一時停止 \(short-circuiting\)](#) を参照してください。

## 12.2. サンプル MACHINEHEALTHCHECK リソース

ベアメタルを除くすべてのクラウドベースのインストールタイプの **MachineHealthCheck** リソースは、以下の YAML ファイルのようになります。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineHealthCheck
metadata:
  name: example ❶
  namespace: openshift-machine-api
spec:
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-machine-role: <role> ❷
      machine.openshift.io/cluster-api-machine-type: <role> ❸
      machine.openshift.io/cluster-api-machineset: <cluster_name>-<label>-<zone> ❹
  unhealthyConditions:
    - type: "Ready"
      timeout: "300s" ❺
      status: "False"
    - type: "Ready"
      timeout: "300s" ❻
      status: "Unknown"
  maxUnhealthy: "40%" ❼
  nodeStartupTimeout: "10m" ❽
```

- ❶ デプロイするマシンヘルスチェックの名前を指定します。

- 2 3 チェックする必要のあるマシンプールのラベルを指定します。
- 4 追跡するマシンセットを `<cluster_name>-<label>-<zone>` 形式で指定します。たとえば、`prod-node-us-east-1a` とします。
- 5 6 ノードの状態のタイムアウト期間を指定します。タイムアウト期間の条件が満たされると、マシンは修正されます。タイムアウトの時間が長くなると、正常でないマシンのワークロードのダウンタイムが長くなる可能性があります。
- 7 ターゲットプールで同時に修復できるマシンの数を指定します。これはパーセンテージまたは整数として設定できます。正常でないマシンの数が `maxUnhealthy` で設定された制限を超える場合、修復は実行されません。
- 8 マシンが正常でないと判別される前に、ノードがクラスターに参加するまでマシンヘルスチェックが待機する必要のあるタイムアウト期間を指定します。



#### 注記

`matchLabels` はあくまでもサンプルであるため、特定のニーズに応じてマシングループをマッピングする必要があります。

### 12.2.1. マシンヘルスチェックによる修復の一時停止 (short-circuiting)

一時停止 (short-circuiting) が実行されることにより、マシンのヘルスチェックはクラスターが正常な場合にのみマシンを修復するようになります。一時停止 (short-circuiting) は、`MachineHealthCheck` リソースの `maxUnhealthy` フィールドで設定されます。

ユーザーがマシンの修復前に `maxUnhealthy` フィールドの値を定義する場合、`MachineHealthCheck` は `maxUnhealthy` の値を、正常でないと判別するターゲットプール内のマシン数と比較します。正常でないマシンの数が `maxUnhealthy` の制限を超える場合、修復は実行されません。



#### 重要

`maxUnhealthy` が設定されていない場合、値は **100%** にデフォルト設定され、マシンはクラスターの状態に関係なく修復されます。

適切な `maxUnhealthy` 値は、デプロイするクラスターの規模や、`MachineHealthCheck` が対応するマシンの数によって異なります。たとえば、`maxUnhealthy` 値を使用して複数のアベイラビリティゾーン間で複数のマシンセットに対応でき、ゾーン全体が失われると、`maxUnhealthy` の設定によりクラスター内で追加の修復を防ぐことができます。複数のアベイラビリティゾーンを持たないグローバル Azure リージョンでは、アベイラビリティセットを使用して高可用性を確保できます。

`maxUnhealthy` フィールドは整数またはパーセンテージのいずれかに設定できます。`maxUnhealthy` の値によって、修復の実装が異なります。

#### 12.2.1.1. 絶対値を使用した `maxUnhealthy` の設定

`maxUnhealthy` が **2** に設定される場合:

- 2 つ以下のノードが正常でない場合に、修復が実行されます。
- 3 つ以上のノードが正常でない場合は、修復は実行されません。

これらの値は、マシンヘルスチェックによってチェックされるマシン数と別個の値です。

### 12.2.1.2. パーセンテージを使用した maxUnhealthy の設定

**maxUnhealthy** が 40% に設定され、25 のマシンがチェックされる場合:

- 10 以下のノードが正常でない場合に、修復が実行されます。
- 11 以上のノードが正常でない場合は、修復は実行されません。

**maxUnhealthy** が 40% に設定され、6 マシンがチェックされる場合:

- 2 つ以下のノードが正常でない場合に、修復が実行されます。
- 3 つ以上のノードが正常でない場合は、修復は実行されません。



#### 注記

チェックされる **maxUnhealthy** マシンの割合が整数ではない場合、マシンの許可される数は切り捨てられます。

## 12.3. MACHINEHEALTHCHECK リソースの作成

クラスターに、すべての **MachineSets** の **MachineHealthCheck** リソースを作成できます。コントロールプレーンマシンをターゲットとする **MachineHealthCheck** リソースを作成することはできません。

#### 前提条件

- **oc** コマンドラインインターフェイスをインストールします。

#### 手順

1. マシンヘルスチェックの定義を含む **healthcheck.yml** ファイルを作成します。
2. **healthcheck.yml** ファイルをクラスターに適用します。

```
$ oc apply -f healthcheck.yml
```

マシンヘルスチェックを設定し、デプロイして、正常でないベアメタルノードを検出し、修復することができます。

## 12.4. ベアメタルの電源ベースの修復について

ベアメタルクラスターでは、クラスター全体の正常性を確保するためにノードの修復は重要になります。クラスターの物理的な修復には難題が伴う場合があります。マシンを安全な状態または動作可能な状態にするまでの遅延が原因で、クラスターが動作が低下した状態のままに置かれる時間が長くなり、その後の障害の発生によりクラスターがオフラインになるリスクが生じます。電源ベースの修復は、このような課題への対応に役立ちます。

ノードの再プロビジョニングを行う代わりに、電源ベースの修復は電源コントローラーを使用して、動作不能なノードの電源をオフにします。この種の修復は、電源フェンシングとも呼ばれます。

OpenShift Container Platform は **MachineHealthCheck** コントローラーを使用して障害のあるベアメタルノードを検出します。電源ベースの修復は高速であり、障害のあるノードをクラスターから削除する代わりにこれを再起動します。

電源ベースの修復は以下の機能を提供します。

- コントロールプレーンノードのリカバリーの許可
- ハイパーコンバージド環境でのデータ損失リスクの軽減
- 物理マシンのリカバリーに関連するダウンタイムの削減

#### 12.4.1. ベアメタル上の MachineHealthCheck

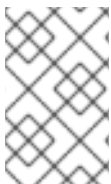
ベアメタルクラスターでのマシンの削除により、ベアメタルホストの再プロビジョニングがトリガーされます。通常、ベアメタルの再プロビジョニングは長いプロセスで、クラスターにコンピュートリソースがなくなり、アプリケーションが中断される可能性があります。デフォルトの修復プロセスをマシンの削除からホストの電源サイクルに切り換えるには、**MachineHealthCheck** リソースに **machine.openshift.io/remediation-strategy: external-baremetal** アノテーションを付けます。

アノテーションの設定後に、BMC 認証情報を使用して正常でないマシンの電源が入れ直されます。

#### 12.4.2. 修復プロセスについて

修復プロセスは以下のように機能します。

1. MachineHealthCheck (MHC) コントローラーは、ノードが正常ではないことを検知します。
2. MHC は、正常でないノードの電源オフを要求するベアメタルマシンコントローラーに通知します。
3. 電源がオフになった後にノードが削除され、クラスターは影響を受けたワークロードを他のノードで再スケジューリングできます。
4. ベアメタルマシンコントローラーはノードの電源をオンにするよう要求します。
5. ノードの起動後、ノードはクラスターに自らを再登録し、これにより新規ノードが作成されます。
6. ノードが再作成されると、ベアメタルマシンコントローラーは、削除前に正常でないノードに存在したアノテーションとラベルを復元します。



#### 注記

電源操作が完了していない場合、ベアメタルマシンコントローラーは、外部でプロビジョニングされたコントロールプレーンノードやノードでない場合に正常でないノードの再プロビジョニングをトリガーします。

#### 12.4.3. ベアメタルの MachineHealthCheck リソースの作成

##### 前提条件

- OpenShift Container Platform は、インストーラーでプロビジョニングされるインフラストラクチャー (IPI) を使用してインストールされます。
- ベースボード管理コントローラー (BMC) 認証情報へのアクセス (または 各ノードへの BMC アクセス)。
- 正常でないノードの BMC インターフェイスへのネットワークアクセス。

##### 手順

1. マシンヘルスチェックの定義を含む **healthcheck.yaml** ファイルを作成します。
2. 以下のコマンドを使用して、**healthcheck.yaml** ファイルをクラスターに適用します。

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

### ベアメタルのサンプル **MachineHealthCheck** リソース

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineHealthCheck
metadata:
  name: example ❶
  namespace: openshift-machine-api
  annotations:
    machine.openshift.io/remediation-strategy: external-baremetal ❷
spec:
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-machine-role: <role> ❸
      machine.openshift.io/cluster-api-machine-type: <role> ❹
      machine.openshift.io/cluster-api-machineset: <cluster_name>-<label>-<zone> ❺
  unhealthyConditions:
    - type: "Ready"
      timeout: "300s" ❻
      status: "False"
    - type: "Ready"
      timeout: "300s" ❼
      status: "Unknown"
  maxUnhealthy: "40%" ❽
  nodeStartupTimeout: "10m" ❾
```

- ❶ デプロイするマシンヘルスチェックの名前を指定します。
- ❷ ベアメタルクラスターの場合、電源サイクルの修復を有効にするために **machine.openshift.io/remediation-strategy: external-baremetal** アノテーションを **annotations** セクションに含める必要があります。この修復ストラテジーにより、正常でないホストはクラスターから削除される代わりに、再起動されます。
- ❸ ❹ チェックする必要があるマシンプールのラベルを指定します。
- ❺ 追跡するマシンセットを **<cluster\_name>-<label>-<zone>** 形式で指定します。たとえば、**prod-node-us-east-1a** とします。
- ❻ ❼ ノード状態のタイムアウト期間を指定します。タイムアウト期間の条件が満たされると、マシンは修正されます。タイムアウトの時間が長くなると、正常でないマシンのワークロードのダウンタイムが長くなる可能性があります。
- ❽ ターゲットプールで同時に修復できるマシンの数を指定します。これはパーセンテージまたは整数として設定できます。正常でないマシンの数が **maxUnhealthy** で設定された制限を超える場合、修復は実行されません。
- ❾ マシンが正常でないと判別される前に、ノードがクラスターに参加するまでマシンヘルスチェックが待機する必要があるタイムアウト期間を指定します。



## 注記

**matchLabels** はあくまでもサンプルであるため、特定のニーズに応じてマシングループをマッピングする必要があります。

<mgmt-troubleshooting-issue-power-remediation\_deploying-machine-health-checks><title>電源ベースの修復に関する問題のトラブルシューティング</title>

電源ベースの修復についての問題のトラブルシューティングを行うには、以下を確認します。

- BMC にアクセスできる。
- BMC は修復タスクを実行するコントロールプレーンノードに接続されている。

</mgmt-troubleshooting-issue-power-remediation\_deploying-machine-health-checks>