OpenShift Container Platform 4.16

マシン管理

クラスターマシンの追加および保守

Last Updated: 2024-07-06

OpenShift Container Platform 4.16 マシン管理

クラスターマシンの追加および保守

法律上の通知

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

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）、Microsoft Azure、Google Cloud Platform (GCP）、Red Hat OpenStack Platform (RHOSP）、VMware vSphere などの基礎インフラストラクチャーと柔軟に連携し、OpenShift Container Platform クラスターの管理を行うことができます。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。

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。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。

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**
- **Machine**
- **ClusterAutoscaler**
- **MachineAutoscaler**
- **MachineHealthCheck**

1.1. MACHINE API の概要

Machine API は、プライマリーリソースの組み合わせたものであり、アップストリーム Cluster API プロジェクトとカスタム OpenShift Container Platform リソースをベースとしています。

OpenShift Container Platform 4.16 クラスターの場合、Machine 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に対する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 4.16 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、弾力的かつ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、コンピュートノードのマシンタイプは、特定のマシンタイプと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、計算マシンのグループです。コンピューティングマシンセットはコンピューティングマシン用であり、レプリカセットは Pod 用です。より多くのコンピューティングマシンが必要な場合、またはそれらを縮小する必要がある場合は、コンピューティングのニーズを満たすように **MachineSet** リソースの **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、コンピューティングマシンセットでは管理できません。

コントロールプレーンマシンセットは、サポートされているコントロールプレーンマシンに対して、コンピュートマシンセットがコンピュートマシンに提供するものと同様の管理機能を提供します。

詳細は、「コントロールプレーンマシンの管理」を参照してください。

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、これはコンピュートマシンセット 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。クラスターオートスケーラーを使用して、次の方法でクラスターを管理できます。

- コア、ノード、メモリー、GPU などのリソースに対してクラスター全体のスケーリング制限を設定
- クラスターが Pod に優先順位を付け、重要度の低い Pod のために新しいノードがオンラインにならないように、優先順位を設定します。
- ノードをスケールアップできるがスケールダウンできないようにスケーリングポリシーを設定

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デプロイメント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。各コンピュートマシンセットのスコープは 1 つのゾーンに限定されるため、インストールプログラムはユーザーに代わって複数のアベイラビリティゾーンにコンピューティング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ーゾーンを持たないグローバル Azure リージョンでは、アベイラビリティー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。

関連情報

- マシンのフェーズとライフサイクル

1.2. コンピューティングマシンの管理

クラスター管理者は、次のアクションを実行できます。

- 次のクラウドプロバイダー用のコンピューティングマシンセットを作成します。
  - Alibaba Cloud
  - AWS
  - Azure
  - Azure Stack Hub
  - GCP
  - IBM Cloud
  - IBM Power Virtual Server
  - Nutanix
  - RHOSP
  - 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
- コンピュートマシンセットにマシンを追加または削除して、コンピュートマシンセットを手動でスケーリング します。
- **MachineSet** YAML 設定ファイルを介して、コンピュート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コンピューティングマシンセットを作成します。
- 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. コントロールプレーンマシンの管理

クラスター管理者は、次のアクションを実行できます。

- 次のクラウドプロバイダー用に設定されたコントロールプレーンマシンを使用して、コントロールプレーンの設定を更新します。
  - Amazon Web Services
  - Google Cloud Platform
  - Microsoft Azure

    - Nutanix
    - Red Hat OpenStack Platform (RHOSP)
    - VMware vSphere

- マシンのヘルスチェック を設定してデプロイメントし、異常なコントロールプレーンマシンを自動的に回復します。

1.4. OPENSHIFT 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。

1.5. 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。

1.6. RHEL コンピュートマシンのクラスターへの追加

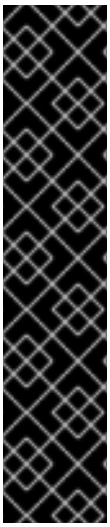
クラスター管理者は、次のアクションを実行できます。

- 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トマシンの管理

2.1. ALIBABA CLOUD で計算機セットを作成する

Alibaba Cloud 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、Machine API が動作しているクラスターでのみ使用できます。user-provisioned infrastructure を持つクラスターでは、Machine 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** のクラスターでは、Machine 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 のコンピューティング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コンピュート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
      vpcId: ""
      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}{"\n"}' 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コンピュートデフォルトマシンのイメージを使用します。

12 コンピュート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 コンピュート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コンピュート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コンピュート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 コンピュート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コンピュートマシンセットに入力するタグなど、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コンピューティング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** namespace 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コンピュート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コンピュートマシンセットには、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** パラメーターに設定されます。

追加のマシンをデプロイするコンピュート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、作成するコンピュート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、デフォルトのコンピュート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、コンピュート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、コンピュート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、コンピュート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. コンピュート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. コンピュートマシンセットのカスタムリソース (CR) サンプルを含む新しい YAML ファイルを作成し、**<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) 値を表示するには、以下のコマンドを実行します。

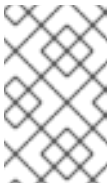
```
$ oc get machineset <machineset_name> \
  -n openshift-machine-api -o yaml
```

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

user-provisioned infrastructure 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。

3 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
```

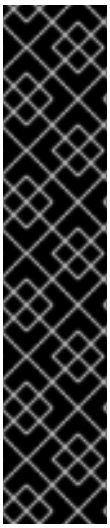
出力例

```
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
```

新しいコンピュートマシンセットが利用可能になると、**DESIRED** と **CURRENT** の値が一致します。コンピュートマシンセットが使用できない場合は、数分待ってからコマンドを再実行してください。

2.2. AWS でコンピュートマシンセットを作成する

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コンピュート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、Machine API が動作しているクラスターでのみ使用できます。user-provisioned infrastructure を持つクラスターでは、Machine 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** のクラスターでは、Machine 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 上のコンピュート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、Amazon Web Services (AWS) Local Zone の **us-east-1a** 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ト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
  name: <infrastructure_id>-<role>-<zone> 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>-<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: machine.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
            - name: <custom_tag_name> 17
```

```
        value: <custom_tag_value> 18
    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}{"\n"}' infrastructure cluster
```

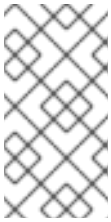
2 4 8 インフラストラクチャー ID、ロールノードラベル、およびゾーンを指定します。

6 7 9 追加するロール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) Amazon Machine Image (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}{"\n"}' \
    get machineset/<infrastructure_id>-<role>-<zone>
```

17 18 オプション: クラスターのカスタムタグデータを指定します。たとえば、**name:value** のペアである **Email:admin-email@example.com** を指定して、管理者の連絡先電子メールアドレスを追加できます。



注記

カスタムタグは、インストール中に **install-config.yml** ファイルで指定することもできます。**install-config.yml** ファイルとマシンセットに同じ **名前** のデータを持つタグが含まれている場合、マシンセットのタグの値が **install-config.yml** ファイルのタグの値よりも優先されます。

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. コンピュート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. コンピュートマシンセットのカスタムリソース (CR) サンプルを含む新しい YAML ファイルを作成し、**<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) 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

user-provisioned infrastructure 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。

3 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ー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
```

新しいコンピュートマシンセットが利用可能になると、**DESIRED** と **CURRENT** の値が一致します。コンピュートマシンセットが使用できない場合は、数分待ってからコマンドを再実行してください。

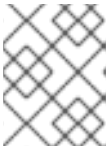
2.2.3. マシンセットを使用した Elastic Fabric Adapter インスタンスの配置グループへのマシンの割り当て

既存の AWS 配置グループ内の Elastic Fabric Adaptor (EFA) インスタンスにマシンをデプロイするようにマシンセットを設定できます。

EFA インスタンスには配置グループは必要なく、EFA の設定以外の目的にも配置グループを使用できます。この例では、両方を使用して、指定された配置グループ内のマシンのネットワークパフォーマンスを向上できる設定を示します。

前提条件

- AWS コンソールで配置グループを作成しました。



注記

作成する配置グループのタイプの ルールと制限が、意図した使用例と互換性があることを確認してください。

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. **providerSpec** フィールドの下に次の行を編集します。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
# ...
spec:
  template:
    spec:
      providerSpec:
        value:
          instanceType: <supported_instance_type> (1)
          networkInterfaceType: EFA (2)
          placement:
            availabilityZone: <zone> (3)
            region: <region> (4)
          placementGroupName: <placement_group> (5)
# ...
```

(1) EFA をサポートする インスタンスタイプを指定します。

(2) **EFA** ネットワークインターフェイスのタイプを指定します。

(3) ゾーン (例: **us-east-1a**) を指定します。

(4) リージョン (例: **us-east-1**) を指定します。

(5) マシンをデプロイする既存の AWS 配置グループの名前を指定します。

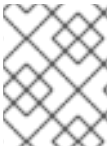
検証

- AWS コンソールで、マシンセットが作成したマシンを見つけて、マシンのプロパティーで次のことを確認します。
  - 配置グループフィールドには、マシンセットの **placeGroupName** パラメーターに指定した値が含まれます。

- インターフェイスタイプフィールドは、EFA を使用することを示します。

2.2.4. 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

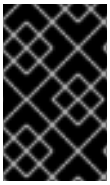
マシンセットを使用して、Amazon EC2 インスタンスメタデータサービス (IMDS) の特定のバージョンを使用するマシンを作成できます。マシンセットは、IMDSv1 と IMDSv2 の両方を使用できるマシン、または IMDSv2 の使用を必要とするマシンを作成できます。



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。

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コンピュートシンセット YAML ファイルを作成します。マシンセットの拡張時に、既存のマシンセットを編集して、希望する IMDS 設定で新しいマシンを作成することもできます。



重要

IMDSv2 を必要とする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。

2.2.4.1. マシンセットを使用した IMDS の設定

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 (1)
```

(1) 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。

2.2.5. 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つ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。

パブリックテナンシーまたは専用テナンシーのいずれかを持つインスタンスが、Machine 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。

2.2.5.1. 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

Machine 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
```

2.2.6. 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 で実行されるコンピュート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。コンピュートマシンセットの **replicas** の量を満たすために、コンピュートマシンセットは Spot インスタンスを要求するマシンを作成します。

2.2.6.1. コンピュートマシンセットの使用による Spot インスタンスの作成

spotMarketOptions をコンピュートマシンセットの 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。

2.2.7. 既存の OpenShift Container Platform クラスターへの GPU ノードの追加

デフォルトのコンピュートマシンセット設定をコピーおよび変更して、AWS EC2 クラウドプロバイダー用の GPU 対応マシンセットとマシンを作成できます。

サポートされているインスタンスタイプの詳細は、以下の NVIDIA ドキュメントを参照してください。

- NVIDIA GPU Operator Community support matrix
- NVIDIA AI Enterprise support matrix

手順

1. 次のコマンドを実行して、既存のノード、マシン、およびマシンセットを表示します。各ノードは、特定の AWS リージョンと OpenShift Container Platform ロールを持つマシン定義のインスタンスであることに注意してください。

```
$ oc get nodes
```

出力例

```
NAME                                         STATUS   ROLES                  AGE     VERSION
ip-10-0-52-50.us-east-2.compute.internal     Ready    worker                 3d17h   v1.29.4
ip-10-0-58-24.us-east-2.compute.internal     Ready    control-plane,master   3d17h   v1.29.4
ip-10-0-68-148.us-east-2.compute.internal    Ready    worker                 3d17h   v1.29.4
ip-10-0-68-68.us-east-2.compute.internal     Ready    control-plane,master   3d17h   v1.29.4
ip-10-0-72-170.us-east-2.compute.internal    Ready    control-plane,master   3d17h   v1.29.4
ip-10-0-74-50.us-east-2.compute.internal     Ready    worker                 3d17h   v1.29.4
```

2. 次のコマンドを実行して、**openshift-machine-api** namespace に存在するマシンとマシンセットを表示します。各コンピュートマシンセットは、AWS リージョン内の異なるアベイラビリティーゾーンに関連付けられています。インストーラーは、アベイラビリティゾーン全体でコンピュートマシンの負荷を自動的に分散します。

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

出力例

```
NAME                                DESIRED   CURRENT   READY   AVAILABLE   AGE
preserve-dsoc12r4-ktjfc-worker-us-east-2a   1         1         1       1           3d11h
preserve-dsoc12r4-ktjfc-worker-us-east-2b   2         2         2       2           3d11h
```

3. 次のコマンドを実行して、**openshift-machine-api** namespace に存在するマシンを表示します。現時点では、マシンセットごとに1つのコンピュートマシンしかありませんが、特定のリージョンとゾーンにノードを追加するようにコンピュートマシンセットをスケーリングすることができます。

```
$ oc get machines -n openshift-machine-api | grep worker
```

出力例

```
preserve-dsoc12r4-ktjfc-worker-us-east-2a-dts8r      Running   m5.xlarge   us-east-2   us-east-2a   3d11h
preserve-dsoc12r4-ktjfc-worker-us-east-2b-dkv7w      Running   m5.xlarge   us-east-2   us-east-2b   3d11h
preserve-dsoc12r4-ktjfc-worker-us-east-2b-k58cw      Running   m5.xlarge   us-east-2   us-east-2b   3d11h
```

4. 次のコマンドを実行して、既存のコンピュート **MachineSet** 定義のいずれかのコピーを作成し、結果を JSON ファイルに出力します。これは、GPU 対応のコンピュートマシンセット定義の基礎となります。

```
$ oc get machineset preserve-dsoc12r4-ktjfc-worker-us-east-2a -n openshift-machine-api -o json > <output_file.json>
```

5. JSON ファイルを編集し、新しい **MachineSet** 定義に次の変更を加えます。

   - **worker** を **gpu** に置き換えます。これが新しいマシンセットの名前になります。
   - 新しい **MachineSet** 定義のインスタンスタイプを、NVIDIA Tesla T4 GPU を含む **g4dn** に変更します。AWS **g4dn** インスタンスタイプの詳細については、Accelerated Computing を参照してください。

     ```
     $ jq .spec.template.spec.providerSpec.value.instanceType preserve-dsoc12r4-ktjfc-worker-gpu-us-east-2a.json

     "g4dn.xlarge"
     ```

     <output_file.json> ファイルは **preserve-dsoc12r4-ktjfc-worker-gpu-us-east-2a.json** として保存されます。

6. **preserve-dsoc12r4-ktjfc-worker-gpu-us-east-2a.json** の次のフィールドを更新します。

   - **.metadata.name** を **gpu** を含む名前に変更します。
   - **.spec.selector.matchLabels["machine.openshift.io/cluster-api-machineset"]** を新しい **.metadata.name** に一致させます。
   - **.spec.template.metadata.labels["machine.openshift.io/cluster-api-machineset"]** を新しい **.metadata.name** に一致させます。

- **.spec.template.spec.providerSpec.value.instanceType** to **g4dn.xlarge**.

7. 変更を確認するには、次のコマンドを実行して、元のコンピュート定義と新しい GPU 対応ノード定義の **diff** を実行します。

```
$ oc -n openshift-machine-api get preserve-dsoc12r4-ktjfc-worker-us-east-2a -o json | diff
preserve-dsoc12r4-ktjfc-worker-gpu-us-east-2a.json -
```

出力例

```
10c10

< "name": "preserve-dsoc12r4-ktjfc-worker-gpu-us-east-2a",
---
> "name": "preserve-dsoc12r4-ktjfc-worker-us-east-2a",

21c21

< "machine.openshift.io/cluster-api-machineset": "preserve-dsoc12r4-ktjfc-worker-gpu-us-
east-2a"
---
> "machine.openshift.io/cluster-api-machineset": "preserve-dsoc12r4-ktjfc-worker-us-east-2a"

31c31

< "machine.openshift.io/cluster-api-machineset": "preserve-dsoc12r4-ktjfc-worker-gpu-us-
east-2a"
---
> "machine.openshift.io/cluster-api-machineset": "preserve-dsoc12r4-ktjfc-worker-us-east-2a"

60c60

< "instanceType": "g4dn.xlarge",
---
> "instanceType": "m5.xlarge",
```

8. 次のコマンドを実行して、定義から GPU 対応のコンピュートマシンセットを作成します。

```
$ oc create -f preserve-dsoc12r4-ktjfc-worker-gpu-us-east-2a.json
```

出力例

```
machineset.machine.openshift.io/preserve-dsoc12r4-ktjfc-worker-gpu-us-east-2a created
```

検証

1. 次のコマンドを実行して、作成したマシンセットを表示します。

```
$ oc -n openshift-machine-api get machinesets | grep gpu
```

MachineSet レプリカ数は **1** に設定されているため、新しい **Machine** オブジェクトが自動的に作成されます。

出力例

```
preserve-dsoc12r4-ktjfc-worker-gpu-us-east-2a   1        1        1      1          4m21s
```

2. 次のコマンドを実行して、マシンセットが作成した **Machine** オブジェクトを表示します。

```
$ oc -n openshift-machine-api get machines | grep gpu
```

出力例

```
preserve-dsoc12r4-ktjfc-worker-gpu-us-east-2a    running    g4dn.xlarge    us-east-2    us-east-2a  4m36s
```

ノードの namespace を指定する必要がないことに注意してください。ノード定義はクラスタースコープ指定されています。

2.2.8. Node Feature Discovery Operator のデプロイ

GPU 対応ノードを作成したら、スケジュールできるように GPU 対応ノードを検出する必要があります。これを行うには、Node Feature Discovery (NFD) Operator をインストールします。NFD Operator は、ノード内のハードウェアデバイス機能を識別します。OpenShift Container Platform で使用できるようにインフラストラクチャーノードのハードウェアリソースを識別してカタログ化するという一般的な問題を解決します。

手順

1. OpenShift Container Platform コンソールの **OperatorHub** から Node Feature Discovery Operator をインストールします。

2. NFD Operator を **OperatorHub** にインストールした後、インストールされた Operator リストから **Node Feature Discovery** を選択し、**Create instance** を選択します。これにより、**openshift-nfd** namespace に、**nfd-master** Pod と **nfd-worker** Pod (各コンピュートノードに 1 つの **nfd-worker** Pod) がインストールされます。

3. 次のコマンドを実行して、Operator がインストールされ、実行されていることを確認します。

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

出力例

```
NAME                                  READY   STATUS    RESTARTS   AGE

nfd-controller-manager-8646fcbb65-x5qgk   2/2     Running 7  (8h ago)   1d
```

4. コンソールでインストール済みの Operator へ移動し、**Create Node Feature Discovery** を選択します。

5. **Create** を選択して、NFD カスタムリソースをビルドします。これにより、OpenShift Container Platform ノードのハードウェアリソースをポーリングしてカタログ化する NFD Pod が **openshift-nfd** namespace に作成されます。

検証

1. ビルドが成功したら、次のコマンドを実行して、各ノードで NFD Pod が実行されていることを確認します。

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

出力例

```
NAME                                  READY   STATUS      RESTARTS        AGE
nfd-controller-manager-8646fcbb65-x5qgk    2/2     Running     7 (8h ago)      12d
nfd-master-769656c4cb-w9vrv                1/1     Running     0               12d
nfd-worker-qjxb2                      1/1     Running     3 (3d14h ago)   12d
nfd-worker-xtz9b                      1/1     Running     5 (3d14h ago)   12d
```

NFD Operator は、ベンダー PCI ID を使用してノード内のハードウェアを識別します。NVIDIA は PCI ID **10de** を使用します。

2. 次のコマンドを実行して、NFD Operator によって検出された NVIDIA GPU を表示します。

```
$ oc describe node ip-10-0-132-138.us-east-2.compute.internal | egrep 'Roles|pci'
```

出力例

```
Roles: worker

feature.node.kubernetes.io/pci-1013.present=true

feature.node.kubernetes.io/pci-10de.present=true

feature.node.kubernetes.io/pci-1d0f.present=true
```

GPU 対応ノードのノード機能リストに **10de** が表示されます。これは、NFD Operator が GPU 対応の MachineSet からノードを正しく識別したことを意味します。

2.3. AZURE でコンピュートマシンセットを作成

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コンピュート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、Machine API が動作しているクラスターでのみ使用できます。user-provisioned infrastructure を持つクラスターでは、Machine 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** のクラスターでは、Machine 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 上のコンピュート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コンピュート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>
  name: <infrastructure_id>-<role>-<region> 3
  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>-<region>
  template:
    metadata:
      creationTimestamp: null
      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>-<region>
    spec:
      metadata:
        creationTimestamp: null
        labels:
          machine.openshift.io/cluster-api-machineset: <machineset_name>
          node-role.kubernetes.io/<role>: ""
      providerSpec:
        value:
          apiVersion: azureproviderconfig.openshift.io/v1beta1
          credentialsSecret:
            name: azure-cloud-credentials
            namespace: openshift-machine-api
          image: 4
            offer: ""
            publisher: ""
            resourceID: /resourceGroups/<infrastructure_id>-
rg/providers/Microsoft.Compute/galleries/gallery_<infrastructure_id>/images/<infrastructure_id>-
gen2/versions/latest 5
            sku: ""
            version: ""
          internalLoadBalancer: ""
          kind: AzureMachineProviderSpec
          location: <region> 6
```

```
        managedIdentity: <infrastructure_id>-identity
        metadata:
          creationTimestamp: null
        natRule: null
        networkResourceGroup: ""
        osDisk:
          diskSizeGB: 128
          managedDisk:
            storageAccountType: Premium_LRS
          osType: Linux
        publicIP: false
        publicLoadBalancer: ""
        resourceGroup: <infrastructure_id>-rg
        sshPrivateKey: ""
        sshPublicKey: ""
        tags:
          - name: <custom_tag_name> 7
            value: <custom_tag_value>
        subnet: <infrastructure_id>-<role>-subnet
        userDataSecret:
          name: worker-user-data
        vmSize: Standard_D4s_v3
        vnet: <infrastructure_id>-vnet
        zone: "1" 8
```

1. 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. 追加するノードラベルを指定します。
3. 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。
4. コンピュート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。
5. 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。
6. マシンを配置するリージョンを指定します。

7 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。

8 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。

2.3.2. コンピュート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. コンピュートマシンセットのカスタムリソース (CR) サンプルを含む新しい YAML ファイルを作成し、**<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) 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

user-provisioned infrastructure 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。

3. 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
```

新しいコンピュートマシンセットが利用可能になると、**DESIRED** と **CURRENT** の値が一致します。コンピュートマシンセットが使用できない場合は、数分待ってからコマンドを再実行してください。

2.3.3. Azure Marketplace オファリングの使用

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。

重要

Azure マーケットプレイスを使用したイメージのインストールは、64 ビット ARM インスタンスを備えたクラスターではサポートされていません。

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

1. 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       Sku                 Urn                                                          Version
-------------  --------------  ------------------  -----------------------------------------------------------  -----------------
rh-ocp-worker  RedHat          rh-ocp-worker       RedHat:rh-ocp-worker:rh-ocp-
```

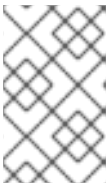
```
worker:413.92.2023101700            413.92.2023101700
rh-ocp-worker  RedHat          rh-ocp-worker-gen1  RedHat:rh-ocp-worker:rh-ocp-worker-
gen1:413.92.2023101700       413.92.2023101700
```

- EMEA:

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

出力例

```
Offer          Publisher       Sku                 Urn
Version
-------------  --------------  ------------------  ----------------------------------------------------------
----           -----------------
rh-ocp-worker  redhat-limited  rh-ocp-worker       redhat-limited:rh-ocp-worker:rh-ocp-
worker:413.92.2023101700            413.92.2023101700
rh-ocp-worker  redhat-limited  rh-ocp-worker-gen1  redhat-limited:rh-ocp-worker:rh-ocp-
worker-gen1:413.92.2023101700       413.92.2023101700
```



注記

コンピュートおよびコントロールプレーンノードで利用可能な最新のイメージを使用します。必要に応じて、VM はインストールプロセスの一部として自動的にアップグレードされます。

2. 次のいずれかのコマンドを実行して、オファ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>
```

3. 次のコマンドのいずれかを実行して、オファ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、オファ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. オファーのイメージの詳細 (具体的には **publisher**、**offer**、**sku**、および **version** の値) を記録します。

6. オファー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 マシンのサンプル providerSpec イメージ値

```
providerSpec:
  value:
    image:
      offer: rh-ocp-worker
      publisher: redhat
      resourceID: ""
      sku: rh-ocp-worker
      type: MarketplaceWithPlan
      version: 413.92.2023101700
```

2.3.4. Azure ブート診断の有効化

マシンセットが作成する Azure マシンで起動診断を有効にできます。

前提条件

- 既存の Microsoft Azure クラスターがある。

手順

- ストレージタイプに適用可能な **diagnostics** 設定を、マシンセット YAML ファイルの **providerSpec** フィールドに追加します。
  - Azure Managed ストレージアカウントの場合:

```
providerSpec:
  diagnostics:
    boot:
      storageAccountType: AzureManaged 1
```

1 Azure Managed ストレージアカウントを指定します。

  - Azure Unmanaged ストレージアカウントの場合:

```
providerSpec:
  diagnostics:
    boot:
      storageAccountType: CustomerManaged 1
      customerManaged:
        storageAccountURI: https://<storage-account>.blob.core.windows.net 2
```

1 Azure Unmanaged ストレージアカウントを指定します。

2 **<storage-account>** をストレージアカウントの名前に置き換えます。

注記

Azure Blob Storage データサービスのみサポートされています。

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 **起動診断** ページを確認し、マシンのシリアルログが表示されることを確認します。

2.3.5. 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 で実行されるコンピュート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。コンピュートマシンセットの **replicas** の量を満たすために、コンピュートマシンセットは Spot VM を要求するマシンを作成します。

2.3.5.1. コンピュートマシンセットの使用による Spot VM の作成

spotVMOptions をコンピュータマシンセットの 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。

2.3.6. 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 で実行されるコンピュート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。

2.3.6.1. コンピュートマシンセットを使用してエフェメラル OS ディスク上にマシンを作成する

コンピュートマシンセットの YAML ファイルを編集して、Azure の一時 OS ディスクでコンピュートマシンを起動できます。

前提条件

- 既存の Microsoft Azure クラスターがある。

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>
```

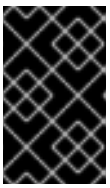
ここで、**<machine-set-name>** は、エフェメラル OS ディスクにマシンをプロビジョニングするコンピュートマシンセットです。

2. 以下を **providerSpec** フィールドに追加します。

```
providerSpec:
  value:
    ...
    osDisk:
      ...
      diskSettings: 1
        ephemeralStorageLocation: Local 2
      cachingType: ReadOnly 3
      managedDisk:
        storageAccountType: Standard_LRS 4
      ...
```

1 2 3 これらの行では、Ephemeral OS ディスクを使用できます。

4 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コンピュート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、コンピュート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。

2.3.7. 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ーは、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

2.3.7.1. 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 クラスターがある。

手順

1. 次のコマンドを実行して、**worker** 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 <role>-user-data \ 1
--template='{{index .data.userData | base64decode}}' | jq > userData.txt 2
```

1. **<role>** を **worker** に置き換えます。
2. 新しいカスタムシークレットの名前として **userData.txt** を指定します。

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 <role>-user-data \ 1
--template='{{index .data.disableTemplating | base64decode}}' | jq > disableTemplating.txt
```

1 **<role>** を **worker** に置き換えます。

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 <role>-user-data-x5 \ 1
--from-file=userData=userData.txt \
--from-file=disableTemplating=disableTemplating.txt
```

1 **<role>-user-data-x5** には、シークレットの名前を指定します。**<role>** を **worker** に置き換えます。

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: ultrassd (1)
      providerSpec:
        value:
          ultraSSDCapability: Enabled (2)
          dataDisks: (3)
          - nameSuffix: ultrassd
            lun: 0
            diskSizeGB: 4
            deletionPolicy: Delete
            cachingType: None
            managedDisk:
              storageAccountType: UltraSSD_LRS
          userDataSecret:
            name: <role>-user-data-x5 (4)
```

(1) 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.ultrassd** を使用します。

(2) (3) 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。

(4) 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。**<role>** を **worker** に置き換えます。

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
```

2.3.7.2. 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。

2.3.7.2.1. 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。

2.3.7.2.2. 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、アベイラビリティーゾーン、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。

2.3.7.2.3. 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。

2.3.8. 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。Machine API を使用すると、顧客管理の鍵によるサーバー側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) がアクセス許可を付与したリソースグループ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

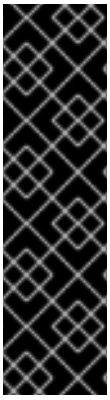
- マシンセット YAML ファイルの **providerSpec** フィールドでディスクの暗号化キーを設定します。以下に例を示します。

```
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

- カスタマーマネージドキーに関する Azure ドキュメント

2.3.9. マシンセットを使用した Azure 仮想マシンの信頼された起動の設定



重要

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

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

OpenShift Container Platform 4.16 は、Azure 仮想マシンの信頼された起動をサポートします。マシンセットの YAML ファイルを編集することで、マシンセットがデプロイメントするマシンに使用する信頼できる起動オプションを設定できます。たとえば、セキュアブートや専用の仮想 Trusted Platform Module (vTPM) インスタンスなどの UEFI セキュリティー機能を使用するようにこれらのマシンを設定できます。

注記

一部の機能の組み合わせでは、無効な設定が発生します。

表2.1 UEFI 機能の組み合わせの互換性

Secure Boot[1]	vTPM[2]	有効な設定
有効	有効	はい
有効	無効	はい
有効	省略	はい
無効	有効	はい
省略	有効	はい
無効	無効	いいえ
省略	無効	いいえ
省略	省略	いいえ

1. **secureBoot** フィールドの使用。
2. **virtualizedTrustedPlatformModule** フィールドの使用。

関連する機能の詳細は、Azure 仮想マシンの信頼された起動 に関する Microsoft Azure のドキュメントを参照してください。

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. **providerSpec** フィールドの下の次のセクションを編集して、有効な設定を指定します。

UEFI セキュアブートと vTPM が有効になっている有効な設定のサンプル

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
# ...
spec:
  template:
    spec:
      providerSpec:
        value:
          securityProfile:
            settings:
              securityType: TrustedLaunch (1)
              trustedLaunch:
                uefiSettings: (2)
                  secureBoot: Enabled (3)
                  virtualizedTrustedPlatformModule: Enabled (4)
# ...
```

(1) Azure 仮想マシンの信頼された起動の使用を有効にします。この値は、すべての有効な設定に必要です。

(2) 使用する UEFI セキュリティー機能を指定します。このセクションは、すべての有効な設定に必要です。

(3) UEFI セキュアブートを有効にします。

(4) vTPM の使用を有効にします。

検証

- 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詳細を確認し、信頼された起動オプションが設定した値と一致することを確認します。

2.3.10. マシンセットを使用した Azure 機密仮想マシンの設定

重要

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

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

OpenShift Container Platform 4.16 は、Azure 機密仮想マシンをサポートします。

注記

現在、機密仮想マシンは 64 ビット ARM アーキテクチャーではサポートされていません。

マシンセットの YAML ファイルを編集することにより、マシンセットがデプロイするマシンに使用する Confidential VM オプションを設定できます。たとえば、セキュアブートや専用の仮想 Trusted Platform Module (vTPM) インスタンスなどの UEFI セキュリティー機能を使用するようにこれらのマシンを設定できます。

関連する機能の詳細は、機密仮想マシン に関する Microsoft Azure のドキュメントを参照してください。

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. **providerSpec** フィールドの下の次のセクションを編集します。

設定サンプル

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
# ...
spec:
  template:
    spec:
      providerSpec:
        value:
          osDisk:
            # ...
            managedDisk:
              securityProfile: (1)
                securityEncryptionType: VMGuestStateOnly (2)
            # ...
          securityProfile: (3)
            settings:
              securityType: ConfidentialVM (4)
              confidentialVM:
                uefiSettings: (5)
                  secureBoot: Disabled (6)
                  virtualizedTrustedPlatformModule: Enabled (7)
          vmSize: Standard_DC16ads_v5 (8)
# ...
```

(1) 機密仮想マシンを使用する場合のマネージドディスクのセキュリティープロファイル設定を指定します。

(2) Azure 仮想マシンゲスト状態 (VMGS) ブロブの暗号化を有効にします。この設定には vTPM の使用が必要です。

3 機密仮想マシンのセキュリティープロファイル設定を指定します。

4 機密仮想マシンの使用を有効にします。この値は、すべての有効な設定に必要です。

5 使用する UEFI セキュリティー機能を指定します。このセクションは、すべての有効な設定に必要です。

6 UEFI セキュアブートを無効にします。

7 vTPM の使用を有効にします。

8 機密仮想マシンをサポートするインスタンスタイプを指定します。

検証

- 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詳細を確認し、Confidential VM オプションが設定した値に一致していることを確認します。

2.3.11. 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。

2.3.11.1. 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。

- Accelerated Networking は、Machine API が動作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。

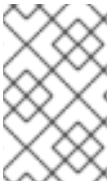
2.3.12. 既存の OpenShift Container Platform クラスターへの GPU ノードの追加

デフォルトのコンピュートマシンセット設定をコピーおよび変更して、Azure クラウドプロバイダー用の GPU 対応マシンセットとマシンを作成できます。

次の表は、検証済みのインスタンスタイプを示しています。

vmSize	NVIDIA GPU アクセラレーター	GPU の最大数	アーキテクチャー
Standard_NC24s_v3	V100	4	x86

vmSize	NVIDIA GPU アクセラレーター	GPU の最大数	アーキテクチャー
Standard_NC4as_T4_v3	T4	1	x86
ND A100 v4	A100	8	x86



注記

デフォルトでは、Azure サブスクリプションには、GPU を使用する Azure インスタンスタイプのクォータがありません。お客様は、上記の Azure インスタンスファミリーのクォータの引き上げを要求する必要があります。

手順

1. 次のコマンドを実行して、**openshift-machine-api** namespace に存在するマシンとマシンセットを表示します。各コンピュートマシンセットは、Azure リージョン内の異なるアベイラビリティーゾーンに関連付けられています。インストーラーは、アベイラビリティゾーン全体でコンピュートマシンの負荷を自動的に分散します。

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

出力例

```
NAME                           DESIRED   CURRENT   READY   AVAILABLE   AGE
myclustername-worker-centralus1   1         1         1       1          6h9m
myclustername-worker-centralus2   1         1         1       1          6h9m
myclustername-worker-centralus3   1         1         1       1          6h9m
```

2. 次のコマンドを実行して、既存のコンピュート **MachineSet** 定義のいずれかのコピーを作成し、結果を YAML ファイルに出力します。これは、GPU 対応のコンピュートマシンセット定義の基礎となります。

```
$ oc get machineset -n openshift-machine-api myclustername-worker-centralus1 -o yaml > machineset-azure.yaml
```

3. マシンセットの内容を表示します。

```
$ cat machineset-azure.yaml
```

machineset-azure.yaml ファイルの例

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  annotations:
    machine.openshift.io/GPU: "0"
    machine.openshift.io/memoryMb: "16384"
    machine.openshift.io/vCPU: "4"
  creationTimestamp: "2023-02-06T14:08:19Z"
```

```
  generation: 1
  labels:
    machine.openshift.io/cluster-api-cluster: myclustername
    machine.openshift.io/cluster-api-machine-role: worker
    machine.openshift.io/cluster-api-machine-type: worker
  name: myclustername-worker-centralus1
  namespace: openshift-machine-api
  resourceVersion: "23601"
  uid: acd56e0c-7612-473a-ae37-8704f34b80de
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: myclustername
      machine.openshift.io/cluster-api-machineset: myclustername-worker-centralus1
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: myclustername
        machine.openshift.io/cluster-api-machine-role: worker
        machine.openshift.io/cluster-api-machine-type: worker
        machine.openshift.io/cluster-api-machineset: myclustername-worker-centralus1
    spec:
      lifecycleHooks: {}
      metadata: {}
      providerSpec:
        value:
          acceleratedNetworking: true
          apiVersion: machine.openshift.io/v1beta1
          credentialsSecret:
            name: azure-cloud-credentials
            namespace: openshift-machine-api
          diagnostics: {}
          image:
            offer: ""
            publisher: ""
            resourceID: /resourceGroups/myclustername-
rg/providers/Microsoft.Compute/galleries/gallery_myclustername_n6n4r/images/myclustername
-gen2/versions/latest
            sku: ""
            version: ""
          kind: AzureMachineProviderSpec
          location: centralus
          managedIdentity: myclustername-identity
          metadata:
            creationTimestamp: null
          networkResourceGroup: myclustername-rg
          osDisk:
            diskSettings: {}
            diskSizeGB: 128
            managedDisk:
              storageAccountType: Premium_LRS
            osType: Linux
          publicIP: false
          publicLoadBalancer: myclustername
          resourceGroup: myclustername-rg
```

```
          spotVMOptions: {}
          subnet: myclustername-worker-subnet
          userDataSecret:
            name: worker-user-data
          vmSize: Standard_D4s_v3
          vnet: myclustername-vnet
          zone: "1"
status:
  availableReplicas: 1
  fullyLabeledReplicas: 1
  observedGeneration: 1
  readyReplicas: 1
  replicas: 1
```

4. 次のコマンドを実行して、**machineset-azure.yaml** ファイルのコピーを作成します。

```
$ cp machineset-azure.yaml machineset-azure-gpu.yaml
```

5. **machineset-azure-gpu.yaml** の次のフィールドを更新します。

- **.metadata.name** を **gpu** を含む名前に変更します。
- **.spec.selector.matchLabels["machine.openshift.io/cluster-api-machineset"]** を変更して新しい .metadata.name に一致させます。
- **.spec.template.metadata.labels["machine.openshift.io/cluster-api-machineset"]** を変更して新しい **.metadata.name** に一致させます。
- **.spec.template.spec.providerSpec.value.vmSize** を **Standard_NC4as_T4_v3** に変更します。

machineset-azure-gpu.yaml ファイルの例

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  annotations:
    machine.openshift.io/GPU: "1"
    machine.openshift.io/memoryMb: "28672"
    machine.openshift.io/vCPU: "4"
  creationTimestamp: "2023-02-06T20:27:12Z"
  generation: 1
  labels:
    machine.openshift.io/cluster-api-cluster: myclustername
    machine.openshift.io/cluster-api-machine-role: worker
    machine.openshift.io/cluster-api-machine-type: worker
  name: myclustername-nc4ast4-gpu-worker-centralus1
  namespace: openshift-machine-api
  resourceVersion: "166285"
  uid: 4eedce7f-6a57-4abe-b529-031140f02ffa
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: myclustername
      machine.openshift.io/cluster-api-machineset: myclustername-nc4ast4-gpu-worker-
```

```
centralus1
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: myclustername
        machine.openshift.io/cluster-api-machine-role: worker
        machine.openshift.io/cluster-api-machine-type: worker
        machine.openshift.io/cluster-api-machineset: myclustername-nc4ast4-gpu-worker-
centralus1
    spec:
      lifecycleHooks: {}
      metadata: {}
      providerSpec:
        value:
          acceleratedNetworking: true
          apiVersion: machine.openshift.io/v1beta1
          credentialsSecret:
            name: azure-cloud-credentials
            namespace: openshift-machine-api
          diagnostics: {}
          image:
            offer: ""
            publisher: ""
            resourceID: /resourceGroups/myclustername-
rg/providers/Microsoft.Compute/galleries/gallery_myclustername_n6n4r/images/myclustern
ame-gen2/versions/latest
            sku: ""
            version: ""
          kind: AzureMachineProviderSpec
          location: centralus
          managedIdentity: myclustername-identity
          metadata:
            creationTimestamp: null
          networkResourceGroup: myclustername-rg
          osDisk:
            diskSettings: {}
            diskSizeGB: 128
            managedDisk:
              storageAccountType: Premium_LRS
            osType: Linux
          publicIP: false
          publicLoadBalancer: myclustername
          resourceGroup: myclustername-rg
          spotVMOptions: {}
          subnet: myclustername-worker-subnet
          userDataSecret:
            name: worker-user-data
          vmSize: Standard_NC4as_T4_v3
          vnet: myclustername-vnet
          zone: "1"
status:
  availableReplicas: 1
  fullyLabeledReplicas: 1
  observedGeneration: 1
  readyReplicas: 1
  replicas: 1
```

6. 変更を確認するには、次のコマンドを実行して、元のコンピュート定義と新しい GPU 対応ノード定義の **diff** を実行します。

```
$ diff machineset-azure.yaml machineset-azure-gpu.yaml
```

出力例

```
14c14
<   name: myclustername-worker-centralus1
---
>   name: myclustername-nc4ast4-gpu-worker-centralus1
23c23
<       machine.openshift.io/cluster-api-machineset: myclustername-worker-centralus1
---
>       machine.openshift.io/cluster-api-machineset: myclustername-nc4ast4-gpu-worker-
centralus1
30c30
<         machine.openshift.io/cluster-api-machineset: myclustername-worker-centralus1
---
>         machine.openshift.io/cluster-api-machineset: myclustername-nc4ast4-gpu-worker-
centralus1
67c67
<           vmSize: Standard_D4s_v3
---
>           vmSize: Standard_NC4as_T4_v3
```

7. 次のコマンドを実行して、定義ファイルから GPU 対応のコンピュートマシンセットを作成します。

```
$ oc create -f machineset-azure-gpu.yaml
```

出力例

```
machineset.machine.openshift.io/myclustername-nc4ast4-gpu-worker-centralus1 created
```

8. 次のコマンドを実行して、**openshift-machine-api** namespace に存在するマシンとマシンセットを表示します。各コンピュートマシンセットは、Azure リージョン内の異なるアベイラビリティーゾーンに関連付けられています。インストーラーは、アベイラビリティゾーン全体でコンピュートマシンの負荷を自動的に分散します。

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

出力例

```
NAME                                DESIRED   CURRENT   READY   AVAILABLE   AGE
clustername-n6n4r-nc4ast4-gpu-worker-centralus1   1         1         1       1           122m
clustername-n6n4r-worker-centralus1               1         1         1       1           8h
clustername-n6n4r-worker-centralus2               1         1         1       1           8h
clustername-n6n4r-worker-centralus3               1         1         1       1           8h
```

9. 次のコマンドを実行して、**openshift-machine-api** namespace に存在するマシンを表示します。セットごとに設定できるコンピュートマシンは 1 つだけですが、コンピュートマシンセットをスケーリングして、特定のリージョンとゾーンにノードを追加することはできます。

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

出力例

```
NAME                                 PHASE     TYPE              REGION      ZONE   AGE
myclustername-master-0               Running   Standard_D8s_v3   centralus   2
6h40m
myclustername-master-1               Running   Standard_D8s_v3   centralus   1
6h40m
myclustername-master-2               Running   Standard_D8s_v3   centralus   3
6h40m
myclustername-nc4ast4-gpu-worker-centralus1-w9bqn   Running   centralus   1      21m
myclustername-worker-centralus1-rbh6b               Running   Standard_D4s_v3   centralus
1      6h38m
myclustername-worker-centralus2-dbz7w               Running   Standard_D4s_v3   centralus
2      6h38m
myclustername-worker-centralus3-p9b8c               Running   Standard_D4s_v3   centralus
3      6h38m
```

10. 次のコマンドを実行して、既存のノード、マシン、およびマシンセットを表示します。各ノードは、特定の Azure リージョンと OpenShift Container Platform ロールを持つマシン定義のインスタンスであることに注意してください。

```
$ oc get nodes
```

出力例

```
NAME                                 STATUS   ROLES                  AGE     VERSION
myclustername-master-0               Ready    control-plane,master   6h39m   v1.29.4
myclustername-master-1               Ready    control-plane,master   6h41m   v1.29.4
myclustername-master-2               Ready    control-plane,master   6h39m   v1.29.4
myclustername-nc4ast4-gpu-worker-centralus1-w9bqn   Ready    worker          14m
v1.29.4
myclustername-worker-centralus1-rbh6b               Ready    worker          6h29m   v1.29.4
myclustername-worker-centralus2-dbz7w               Ready    worker          6h29m   v1.29.4
myclustername-worker-centralus3-p9b8c               Ready    worker          6h31m   v1.29.4
```

11. 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
myclustername-worker-centralus1      1         1         1       1           8h
myclustername-worker-centralus2      1         1         1       1           8h
myclustername-worker-centralus3      1         1         1       1           8h
```

12. 次のコマンドを実行して、定義ファイルから GPU 対応のコンピュートマシンセットを作成します。

```
$ oc create -f machineset-azure-gpu.yaml
```

13. 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
myclustername-nc4ast4-gpu-worker-centralus1   1         1         1       1           121m
myclustername-worker-centralus1       1         1         1       1           8h
myclustername-worker-centralus2       1         1         1       1           8h
myclustername-worker-centralus3       1         1         1       1           8h
```

検証

1. 次のコマンドを実行して、作成したマシンセットを表示します。

```
$ oc get machineset -n openshift-machine-api | grep gpu
```

MachineSet レプリカ数は **1** に設定されているため、新しい **Machine** オブジェクトが自動的に作成されます。

出力例

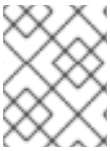
```
myclustername-nc4ast4-gpu-worker-centralus1   1         1         1       1           121m
```

2. 次のコマンドを実行して、マシンセットが作成した **Machine** オブジェクトを表示します。

```
$ oc -n openshift-machine-api get machines | grep gpu
```

出力例

```
myclustername-nc4ast4-gpu-worker-centralus1-w9bqn   Running   Standard_NC4as_T4_v3
centralus   1      21m
```



注記

ノードの namespace を指定する必要はありません。ノード定義はクラススコープ指定されています。

2.3.13. Node Feature Discovery Operator のデプロイ

GPU 対応ノードを作成したら、スケジュールできるように GPU 対応ノードを検出する必要があります。これを行うには、Node Feature Discovery (NFD) Operator をインストールします。NFD Operator は、ノード内のハードウェアデバイス機能を識別します。OpenShift Container Platform で使用できるようにインフラストラクチャーノードのハードウェアリソースを識別してカタログ化するという一般的な問題を解決します。

手順

1. OpenShift Container Platform コンソールの **OperatorHub** から Node Feature Discovery Operator をインストールします。

2. NFD Operator を **OperatorHub** にインストールした後、インストールされた Operator リストから **Node Feature Discovery** を選択し、**Create instance** を選択します。これにより、**openshift-nfd** namespace に、**nfd-master** Pod と **nfd-worker** Pod (各コンピュートノードに 1 つの **nfd-worker** Pod) がインストールされます。

3. 次のコマンドを実行して、Operator がインストールされ、実行されていることを確認します。

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

出力例

```
NAME                                  READY   STATUS    RESTARTS   AGE

nfd-controller-manager-8646fcbb65-x5qgk   2/2     Running 7  (8h ago)   1d
```

4. コンソールでインストール済みの Operator へ移動し、**Create Node Feature Discovery** を選択します。

5. **Create** を選択して、NFD カスタムリソースをビルドします。これにより、OpenShift Container Platform ノードのハードウェアリソースをポーリングしてカタログ化する NFD Pod が **openshift-nfd** namespace に作成されます。

検証

1. ビルドが成功したら、次のコマンドを実行して、各ノードで NFD Pod が実行されていることを確認します。

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

出力例

```
NAME                                  READY   STATUS      RESTARTS       AGE
nfd-controller-manager-8646fcbb65-x5qgk   2/2     Running     7 (8h ago)     12d
nfd-master-769656c4cb-w9vrv               1/1     Running     0              12d
nfd-worker-qjxb2                      1/1     Running     3 (3d14h ago)  12d
nfd-worker-xtz9b                      1/1     Running     5 (3d14h ago)  12d
```

NFD Operator は、ベンダー PCI ID を使用してノード内のハードウェアを識別します。NVIDIA は PCI ID **10de** を使用します。

2. 次のコマンドを実行して、NFD Operator によって検出された NVIDIA GPU を表示します。

```
$ oc describe node ip-10-0-132-138.us-east-2.compute.internal | egrep 'Roles|pci'
```

出力例

```
Roles: worker

feature.node.kubernetes.io/pci-1013.present=true

feature.node.kubernetes.io/pci-10de.present=true

feature.node.kubernetes.io/pci-1d0f.present=true
```

GPU 対応ノードのノード機能リストに **10de** が表示されます。これは、NFD Operator が GPU 対応の MachineSet からノードを正しく識別したことを意味します。

関連情報

- インストール中の高速ネットワークの有効化

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

マシンセット YAML ファイルに **acceleratedNetworking** を追加することで、Azure で Accelerated Networking を有効にすることができます。

前提条件

- Machine API が動作している既存の Microsoft Azure クラスターがある。

手順

- 以下を **providerSpec** フィールドに追加します。

```
providerSpec:
  value:
    acceleratedNetworking: true (1)
    vmSize: <azure-vm-size> (2)
```

(1) この行は Accelerated Networking を有効にします。

(2) 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。

次のステップ

- 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

- コンピュートマシンセットの手動スケーリング

2.4. AZURE STACK HUB でコンピュートマシンセットを作成

Microsoft Azure Stack Hub 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コンピュート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、Machine API が動作しているクラスターでのみ使用できます。user-provisioned infrastructure を持つクラスターでは、Machine 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** のクラスターでは、Machine 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 上のコンピュート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コンピュート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}{"\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 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. コンピュート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。
- Azure Stack Hub コンピュート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. コンピュートマシンセットのカスタムリソース (CR) サンプルを含む新しい YAML ファイルを作成し、**<file_name>.yaml** という名前を付けます。
   <availabilitySet>、**<clusterID>**、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. 特定のコンピュートマシンセットカスタムリソース (CR) 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

user-provisioned infrastructure 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。

3 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
```

新しいコンピュートマシンセットが利用可能になると、**DESIRED** と **CURRENT** の値が一致します。コンピュートマシンセットが使用できない場合は、数分待ってからコマンドを再実行してください。

2.4.3. Azure ブート診断の有効化

マシンセットが作成する Azure マシンで起動診断を有効にできます。

前提条件

- 既存の Microsoft Azure Stack Hub クラスターがある。

手順

- ストレージタイプに適用可能な **diagnostics** 設定を、マシンセット YAML ファイルの **providerSpec** フィールドに追加します。
  - Azure Managed ストレージアカウントの場合:

```
providerSpec:
  diagnostics:
    boot:
      storageAccountType: AzureManaged 1
```

1 Azure Managed ストレージアカウントを指定します。

  - Azure Unmanaged ストレージアカウントの場合:

```
providerSpec:
  diagnostics:
    boot:
      storageAccountType: CustomerManaged 1
      customerManaged:
        storageAccountURI: https://<storage-account>.blob.core.windows.net 2
```

1 Azure Unmanaged ストレージアカウントを指定します。

2 **<storage-account>** をストレージアカウントの名前に置き換えます。

注記

Azure Blob Storage データサービスのみサポートされています。

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 **起動診断** ページを確認し、マシンのシリアルログが表示されることを確認します。

2.4.4. 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。Machine API を使用すると、顧客管理の鍵によるサーバー側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) がアクセス許可を付与したリソースグループ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。

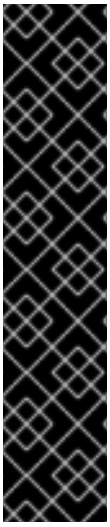
```
providerSpec:
  value:
    osDisk:
      diskSizeGB: 128
      managedDisk:
        diskEncryptionSet:
          id:
/subscriptions/<subscription_id>/resourceGroups/<resource_group_name>/providers/Microsoft.
Compute/diskEncryptionSets/<disk_encryption_set_name>
        storageAccountType: Premium_LRS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

関連情報

- カスタマーマネージドキーに関する Azure ドキュメント

2.5. GCP でコンピュートマシンセットを作成する

異なるコンピュートマシンセットを作成して、Google Cloud Platform (GCP) 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、Machine API が動作しているクラスターでのみ使用できます。user-provisioned infrastructure を持つクラスターでは、Machine 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** のクラスターでは、Machine 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}'
```

2.5.1. GCP 上のコンピュート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、Google Cloud Platform (GCP) で実行されるコンピューティングマシンセットを定義し、**node-role.kubernetes.io/<role>: ""** でラベル付けされたノードを作成します。**<role>** は追加するノードラベルです。

OpenShift CLI を使用して取得した値
以下の例では、OpenShift CLI を使用してクラスターの値の一部を取得できます。

インフラストラクチャー ID

<infrastructure_id> 文字列は、クラスターをプロビジョニングしたとき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。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
```

イメージパス

<path_to_image> 文字列は、ディスクの作成に使用されたイメージへのパスです。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}{"\n"}' \
  get machineset/<infrastructure_id>-worker-a
```

サンプル GCP MachineSet 値

```
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> 3
            labels: null
            sizeGb: 128
            type: pd-ssd
          gcpMetadata: 4
          - 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> 5
          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
```

1. **<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。
2. **<node>** には、追加するノードラベルを指定します。
3. 現在のコンピュートマシンセットで使用されるイメージへのパスを指定します。

   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-413-x86-64-202305021736**
   - OpenShift Platform Plus: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-opp-413-x86-64-202305021736**
   - OpenShift Kubernetes Engine: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-oke-413-x86-64-202305021736**

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. コンピュート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. コンピュートマシンセットのカスタムリソース (CR) サンプルを含む新しい YAML ファイルを作成し、**<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) 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

user-provisioned infrastructure 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。

3 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
```

新しいコンピュートマシンセットが利用可能になると、**DESIRED** と **CURRENT** の値が一致します。コンピュートマシンセットが使用できない場合は、数分待ってからコマンドを再実行してください。

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。

永続ディスクのタイプ、互換性、リージョン別の可用性、制限事項の詳細は、永続ディスク に関する GCP Compute Engine のドキュメントを参照してください。

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. **providerSpec** フィールドの下で以下の行を編集します。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
...
spec:
  template:
    spec:
      providerSpec:
        value:
          disks:
            type: <pd-disk-type> (1)
```

(1) 永続ディスクの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 コンソールを使用して、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に一致していることを確認します。

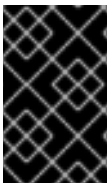
2.5.4. マシンセットを使用した Confidential VM の設定

マシンセットの YAML ファイルを編集することにより、マシンセットがデプロイするマシンに使用する Confidential VM オプションを設定できます。

機密仮想マシンの機能、互換性の詳細は、Confidential VM に関する GCP Compute Engine のドキュメントを参照してください。

注記

現在、機密仮想マシンは 64 ビット ARM アーキテクチャーではサポートされていません。



重要

OpenShift Container Platform 4.16 は、AMD Secure Encrypted Virtualization Secure Nested Paging (SEV-SNP) を備えた機密仮想マシンなど、一部の Confidential Compute 機能をサポートしていません。

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. **providerSpec** フィールドの下の次のセクションを編集します。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
...
spec:
  template:
    spec:
      providerSpec:
        value:
          confidentialCompute: Enabled (1)
          onHostMaintenance: Terminate (2)
          machineType: n2d-standard-8 (3)
...
```

(1) Confidential VM を有効にするかどうかを指定します。有効な値は **Disabled** または **Enabled** です。

(2) ハードウェアやソフトウェアの更新など、ホストのメンテナンスイベント中の VM の動作を指定します。Confidential VM を使用するマシンの場合は、この値を **Terminate** に設定する必要があります。これにより、VM が停止します。Confidential VM はライブ VM 移行をサポートしていません。

(3) Confidential VM をサポートするマシンタイプを指定します。Confidential VM は、N2D および C2D シリーズのマシンタイプをサポートしています。

検証

- GCP 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、Confidential VM オプションが設定した値に一致していることを確認します。

2.5.5. 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 GCP で実行されるコンピュートマシンセットを作成して、コストを節約できます。プリエンプ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。プリエンプ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、プリエンプ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プションの通知をユーザーに送信します。OpenShift Container Platform は、Compute Engine がプリエンプ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。プリエンプ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、プリエンプ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
- プリエンプ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、プリエンプ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、プリエンプ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。コンピュートマシンセットの **レプリカ** 数を満たすために、ココンピュートマシンセットは、プリエンプティブル VM インスタンスを要求するマシンを作成します。

2.5.5.1. コンピュートマシンセットの使用によるプリエンプション可能な仮想マシンインスタンスの作成

preemptible をコンピュートマシンセットの YAML ファイルに追加し、GCP でプリエンプ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。

2.5.6. マシンセットを使用した Shielded VM オプションの設定

マシンセットの YAML ファイルを編集することで、マシンセットがデプロイメントするマシンに使用する Shielded VM オプションを設定できます。

Shielded VM の特徴と機能の詳細については、Shielded VM に関する GCP Compute Engine のドキュメントを参照してください。

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. **providerSpec** フィールドの下の次のセクションを編集します。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
# ...
spec:
  template:
    spec:
      providerSpec:
        value:
          shieldedInstanceConfig: 1
            integrityMonitoring: Enabled 2
            secureBoot: Disabled 3
            virtualizedTrustedPlatformModule: Enabled 4
# ...
```

1. このセクションでは、必要な Shielded VM オプションを指定します。
2. 整合性監視を有効にするかどうかを指定します。有効な値は **Disabled** または **Enabled** です。

注記

整合性監視が有効になっている場合、仮想トラステッドプラットフォームモジュール (vTPM) を無効にしないでください。

3. UEFI セキュアブートを有効にするかどうかを指定します。有効な値は **Disabled** または **Enabled** です。
4. vTPM を有効にするかどうかを指定します。有効な値は **Disabled** または **Enabled** です。

検証

- Google Cloud コンソールを使用して、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、Shielded VM オプションが設定した値に一致していることを確認します。

関連情報

- Shielded VM とは
  - セキュアブート

- 仮想トラステッドプラットフォームモジュール (vTPM)
- 整合性監視

2.5.7. 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。

Machine API を使用するクラスターでは、顧客管理の鍵による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. 特定のサービスアカウントが KMS キーを使用できるようにし、サービスアカウントに正しい IAM ロールを付与するには、KMS キー名、キーリング名、場所を指定して次のコマンドを実行します。

```
$ 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。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
...
spec:
  template:
    spec:
      providerSpec:
        value:
          disks:
          - type:
            encryptionKey:
              kmsKey:
                name: machine-encryption-key 1
                keyRing: openshift-encrpytion-ring 2
```

```
            location: global 3
            projectID: openshift-gcp-project 4
          kmsKeyServiceAccount: openshift-service-account@openshift-gcp-
project.iam.gserviceaccount.com 5
```

1. ディスク暗号化に使用される顧客管理の暗号鍵の名前。
2. KMS キーが属する KMS キーリングの名前。
3. KMS キーリングが存在する GCP の場所。
4. 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。
5. 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しいマシンが作成されると、ディスク暗号化キーが KMS キーで暗号化されます。

2.5.8. コンピュートマシンセットの GPU サポートの有効化

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 OpenShift 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ています。

表2.2 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	**nvidia-tesla-a100**	• **a2-highgpu-1g** • **a2-highgpu-2g** • **a2-highgpu-4g** • **a2-highgpu-8g** • **a2-megagpu-16g**

モデル名	GPU タイプ	マシンタイプ [1]
NVIDIA K80	**nvidia-tesla-k80**	• **n1-standard-1** • **n1-standard-2** • **n1-standard-4** • **n1-standard-8** • **n1-standard-16** • **n1-standard-32** • **n1-standard-64** • **n1-standard-96** • **n1-highmem-2** • **n1-highmem-4** • **n1-highmem-8** • **n1-highmem-16** • **n1-highmem-32** • **n1-highmem-64** • **n1-highmem-96** • **n1-highcpu-2** • **n1-highcpu-4** • **n1-highcpu-8** • **n1-highcpu-16** • **n1-highcpu-32** • **n1-highcpu-64** • **n1-highcpu-96**
NVIDIA P100	**nvidia-tesla-p100**	
NVIDIA P4	**nvidia-tesla-p4**	
NVIDIA T4	**nvidia-tesla-t4**	
NVIDIA V100	**nvidia-tesla-v100**	

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、既存のコンピュートマシンセットの 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. コンピュートマシンセットの YAML ファイルの **providerSpec** 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 (1)
    onHostMaintenance: Terminate (2)
    restartPolicy: Always (3)
```

(1) 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。

(2) 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。

(3) コンピュート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 (1)
      type: nvidia-tesla-p100 (2)
    machineType: n1-standard-1 (3)
    onHostMaintenance: Terminate (4)
    restartPolicy: Always (5)
```

(1) マシンに接続する GPU の数を指定します。

(2) 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。

(3) 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。

(4) 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。

(5) コンピュート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。

2.5.9. 既存の OpenShift Container Platform クラスターへの GPU ノードの追加

デフォルトのコンピュートマシンセット設定をコピーおよび変更して、GCP クラウドプロバイダー用の GPU 対応マシンセットとマシンを作成できます。

次の表は、検証済みのインスタンスタイプを示しています。

インスタンスタイプ	NVIDIA GPU アクセラレーター	GPU の最大数	アーキテクチャー
a2-highgpu-1g	A100	1	x86
n1-standard-4	T4	1	x86

手順

1. 既存の **MachineSet** のコピーを作成します。
2. 新しいコピーで、**metadata.name** と **machine.openshift.io/cluster-api-machineset** の両方のインスタンスで、マシンセットの **name** を変更します。
3. インスタンスタイプを変更して、新しくコピーした **MachineSet** に次の 2 行を追加します。

```
machineType: a2-highgpu-1g
onHostMaintenance: Terminate
```

a2-highgpu-1g.json ファイルの例

```
{
    "apiVersion": "machine.openshift.io/v1beta1",
    "kind": "MachineSet",
    "metadata": {
        "annotations": {
            "machine.openshift.io/GPU": "0",
            "machine.openshift.io/memoryMb": "16384",
            "machine.openshift.io/vCPU": "4"
        },
        "creationTimestamp": "2023-01-13T17:11:02Z",
        "generation": 1,
        "labels": {
            "machine.openshift.io/cluster-api-cluster": "myclustername-2pt9p"
        },
        "name": "myclustername-2pt9p-worker-gpu-a",
        "namespace": "openshift-machine-api",
        "resourceVersion": "20185",
        "uid": "2daf4712-733e-4399-b4b4-d43cb1ed32bd"
    },
    "spec": {
        "replicas": 1,
        "selector": {
            "matchLabels": {
                "machine.openshift.io/cluster-api-cluster": "myclustername-2pt9p",
                "machine.openshift.io/cluster-api-machineset": "myclustername-2pt9p-worker-gpu-
```

```
a"
          }
       },
       "template": {
          "metadata": {
             "labels": {
                "machine.openshift.io/cluster-api-cluster": "myclustername-2pt9p",
                "machine.openshift.io/cluster-api-machine-role": "worker",
                "machine.openshift.io/cluster-api-machine-type": "worker",
                "machine.openshift.io/cluster-api-machineset": "myclustername-2pt9p-worker-
gpu-a"
             }
          },
          "spec": {
             "lifecycleHooks": {},
             "metadata": {},
             "providerSpec": {
                "value": {
                   "apiVersion": "machine.openshift.io/v1beta1",
                   "canIPForward": false,
                   "credentialsSecret": {
                      "name": "gcp-cloud-credentials"
                   },
                   "deletionProtection": false,
                   "disks": [
                      {
                         "autoDelete": true,
                         "boot": true,
                         "image": "projects/rhcos-cloud/global/images/rhcos-412-86-
202212081411-0-gcp-x86-64",
                         "labels": null,
                         "sizeGb": 128,
                         "type": "pd-ssd"
                      }
                   ],
                   "kind": "GCPMachineProviderSpec",
                   "machineType": "a2-highgpu-1g",
                   "onHostMaintenance": "Terminate",
                   "metadata": {
                      "creationTimestamp": null
                   },
                   "networkInterfaces": [
                      {
                         "network": "myclustername-2pt9p-network",
                         "subnetwork": "myclustername-2pt9p-worker-subnet"
                      }
                   ],
                   "preemptible": true,
                   "projectID": "myteam",
                   "region": "us-central1",
                   "serviceAccounts": [
                      {
                         "email": "myclustername-2pt9p-w@myteam.iam.gserviceaccount.com",
                         "scopes": [
                            "https://www.googleapis.com/auth/cloud-platform"
                         ]
```

```
                    }
                ],
                "tags": [
                    "myclustername-2pt9p-worker"
                ],
                "userDataSecret": {
                    "name": "worker-user-data"
                },
                "zone": "us-central1-a"
            }
        }
    }
  }
},
"status": {
    "availableReplicas": 1,
    "fullyLabeledReplicas": 1,
    "observedGeneration": 1,
    "readyReplicas": 1,
    "replicas": 1
}
}
```


4. 次のコマンドを実行して、既存のノード、マシン、およびマシンセットを表示します。各ノードは、特定の GCP リージョンと OpenShift Container Platform ロールを持つマシン定義のインスタンスであることに注意してください。

```
$ oc get nodes
```

出力例

```
NAME                                                STATUS   ROLES                  AGE   VERSION
myclustername-2pt9p-master-0.c.openshift-qe.internal        Ready    control-plane,master   8h    v1.29.4
myclustername-2pt9p-master-1.c.openshift-qe.internal        Ready    control-plane,master   8h    v1.29.4
myclustername-2pt9p-master-2.c.openshift-qe.internal        Ready    control-plane,master   8h    v1.29.4
myclustername-2pt9p-worker-a-mxtnz.c.openshift-qe.internal   Ready    worker                 8h    v1.29.4
myclustername-2pt9p-worker-b-9pzzn.c.openshift-qe.internal   Ready    worker                 8h    v1.29.4
myclustername-2pt9p-worker-c-6pbg6.c.openshift-qe.internal   Ready    worker                 8h    v1.29.4
myclustername-2pt9p-worker-gpu-a-wxcr6.c.openshift-qe.internal   Ready    worker             4h35m   v1.29.4
```

5. 次のコマンドを実行して、**openshift-machine-api** namespace に存在するマシンとマシンセットを表示します。各コンピュートマシンセットは、GCP リージョン内の異なるアベイラビリティーゾーンに関連付けられています。インストーラーは、アベイラビリティゾーン全体でコンピュートマシンの負荷を自動的に分散します。

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

出力例

```
NAME                          DESIRED   CURRENT   READY   AVAILABLE   AGE
myclustername-2pt9p-worker-a     1         1       1       1          8h
myclustername-2pt9p-worker-b     1         1       1       1          8h
myclustername-2pt9p-worker-c     1         1                          8h
myclustername-2pt9p-worker-f     0         0                          8h
```

6. 次のコマンドを実行して、**openshift-machine-api** namespace に存在するマシンを表示します。セットごとに設定できるコンピュートマシンは 1 つだけですが、コンピュートマシンセットをスケーリングして、特定のリージョンとゾーンにノードを追加することはできます。

```
$ oc get machines -n openshift-machine-api | grep worker
```

出力例

```
myclustername-2pt9p-worker-a-mxtnz       Running   n2-standard-4   us-central1   us-
central1-a   8h
myclustername-2pt9p-worker-b-9pzzn       Running   n2-standard-4   us-central1   us-
central1-b   8h
myclustername-2pt9p-worker-c-6pbg6       Running   n2-standard-4   us-central1   us-
central1-c   8h
```

7. 次のコマンドを実行して、既存のコンピュート **MachineSet** 定義のいずれかのコピーを作成し、結果を JSON ファイルに出力します。これは、GPU 対応のコンピュートマシンセット定義の基礎となります。

```
$ oc get machineset myclustername-2pt9p-worker-a -n openshift-machine-api -o json  >
<output_file.json>
```

8. JSON ファイルを編集し、新しい **MachineSet** 定義に次の変更を加えます。

   - サブストリング **gpu** を **metadata.name** と **machine.openshift.io/cluster-api-machineset** の両方のインスタンスに挿入し、マシンセット **name** を変更します。
   - 新しい **MachineSet** 定義の **machineType** を、NVIDIA A100 GPU を含む **a2-highgpu-1g** に変更します。

     ```
     jq .spec.template.spec.providerSpec.value.machineType ocp_4.16_machineset-a2-
     highgpu-1g.json

     "a2-highgpu-1g"
     ```

     <output_file.json> ファイルは **ocp_4.16_machineset-a2-highgpu-1g.json** として保存されます。

9. **ocp_4.16_machineset-a2-highgpu-1g.json** の次のフィールドを更新します。

   - **.metadata.name** を **gpu** を含む名前に変更します。
   - **.spec.selector.matchLabels["machine.openshift.io/cluster-api-machineset"]** を変更して新しい **.metadata.name** に一致させます。

- **.spec.template.metadata.labels["machine.openshift.io/cluster-api-machineset"]** を変更して新しい **.metadata.name** に一致させます。
- **.spec.template.spec.providerSpec.value.MachineType** を **a2-highgpu-1g** に変更します。
- **machineType** の下に次の行を追加します: `"onHostMaintenance": "Terminate"。以下に例を示します。

```
"machineType": "a2-highgpu-1g",
"onHostMaintenance": "Terminate",
```

10. 変更を確認するには、次のコマンドを実行して、元のコンピュート定義と新しい GPU 対応ノード定義の **diff** を実行します。

```
$ oc get machineset/myclustername-2pt9p-worker-a -n openshift-machine-api -o json | diff
ocp_4.16_machineset-a2-highgpu-1g.json -
```

出力例

```
15c15
<         "name": "myclustername-2pt9p-worker-gpu-a",
---
>         "name": "myclustername-2pt9p-worker-a",
25c25
<                 "machine.openshift.io/cluster-api-machineset": "myclustername-2pt9p-worker-
gpu-a"
---
>                 "machine.openshift.io/cluster-api-machineset": "myclustername-2pt9p-worker-a"
34c34
<                   "machine.openshift.io/cluster-api-machineset": "myclustername-2pt9p-worker-
gpu-a"
---
>                   "machine.openshift.io/cluster-api-machineset": "myclustername-2pt9p-worker-
a"
59,60c59
<                       "machineType": "a2-highgpu-1g",
<                       "onHostMaintenance": "Terminate",
---
>                       "machineType": "n2-standard-4",
```

11. 次のコマンドを実行して、定義ファイルから GPU 対応のコンピュートマシンセットを作成します。

```
$ oc create -f ocp_4.16_machineset-a2-highgpu-1g.json
```

出力例

```
machineset.machine.openshift.io/myclustername-2pt9p-worker-gpu-a created
```

検証

1. 次のコマンドを実行して、作成したマシンセットを表示します。

```
$ oc -n openshift-machine-api get machinesets | grep gpu
```

MachineSet レプリカ数は **1** に設定されているため、新しい **Machine** オブジェクトが自動的に作成されます。

出力例

```
myclustername-2pt9p-worker-gpu-a   1         1         1       1           5h24m
```

2. 次のコマンドを実行して、マシンセットが作成した **Machine** オブジェクトを表示します。

```
$ oc -n openshift-machine-api get machines | grep gpu
```

出力例

```
myclustername-2pt9p-worker-gpu-a-wxcr6   Running   a2-highgpu-1g   us-central1   us-central1-a   5h25m
```

注記

ノードの namespace を指定する必要がないことに注意してください。ノード定義はクラスタースコープ指定されています。

2.5.10. Node Feature Discovery Operator のデプロイ

GPU 対応ノードを作成したら、スケジュールできるように GPU 対応ノードを検出する必要があります。これを行うには、Node Feature Discovery (NFD) Operator をインストールします。NFD Operator は、ノード内のハードウェアデバイス機能を識別します。OpenShift Container Platform で使用できるようにインフラストラクチャーノードのハードウェアリソースを識別してカタログ化するという一般的な問題を解決します。

手順

1. OpenShift Container Platform コンソールの **OperatorHub** から Node Feature Discovery Operator をインストールします。

2. NFD Operator を **OperatorHub** にインストールした後、インストールされた Operator リストから **Node Feature Discovery** を選択し、**Create instance** を選択します。これにより、**openshift-nfd** namespace に、**nfd-master** Pod と **nfd-worker** Pod (各コンピュートノードに 1 つの **nfd-worker** Pod) がインストールされます。

3. 次のコマンドを実行して、Operator がインストールされ、実行されていることを確認します。

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

出力例

```
NAME                                  READY   STATUS    RESTARTS   AGE

nfd-controller-manager-8646fcbb65-x5qgk   2/2     Running 7  (8h ago)   1d
```

4. コンソールでインストール済みの Operator へ移動し、**Create Node Feature Discovery** を選択します。

5. **Create** を選択して、NFD カスタムリソースをビルドします。これにより、OpenShift Container Platform ノードのハードウェアリソースをポーリングしてカタログ化する NFD Pod が **openshift-nfd** namespace に作成されます。

検証

1. ビルドが成功したら、次のコマンドを実行して、各ノードで NFD Pod が実行されていることを確認します。

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

出力例

```
NAME                                  READY   STATUS      RESTARTS        AGE
nfd-controller-manager-8646fcbb65-x5qgk    2/2     Running     7 (8h ago)      12d
nfd-master-769656c4cb-w9vrv                1/1     Running     0               12d
nfd-worker-qjxb2                      1/1     Running     3 (3d14h ago)   12d
nfd-worker-xtz9b                      1/1     Running     5 (3d14h ago)   12d
```

NFD Operator は、ベンダー PCI ID を使用してノード内のハードウェアを識別します。NVIDIA は PCI ID **10de** を使用します。

2. 次のコマンドを実行して、NFD Operator によって検出された NVIDIA GPU を表示します。

```
$ oc describe node ip-10-0-132-138.us-east-2.compute.internal | egrep 'Roles|pci'
```

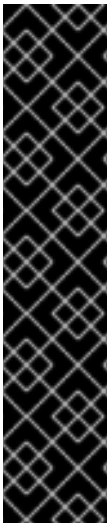
出力例

```
Roles: worker

feature.node.kubernetes.io/pci-1013.present=true

feature.node.kubernetes.io/pci-10de.present=true

feature.node.kubernetes.io/pci-1d0f.present=true
```

GPU 対応ノードのノード機能リストに **10de** が表示されます。これは、NFD Operator が GPU 対応の MachineSet からノードを正しく識別したことを意味します。

2.6. IBM CLOUD でコンピュートマシンセットを作成する

IBM Cloud® 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、Machine API が動作しているクラスターでのみ使用できます。user-provisioned infrastructure を持つクラスターでは、Machine 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** のクラスターでは、Machine 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. IBM Cloud 上のコンピュート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>: ""** というラベルが付いたノードを作成するコンピュート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:
      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:
        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}{"\n"}' 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® instance profile 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. コンピュート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. コンピュートマシンセットのカスタムリソース (CR) サンプルを含む新しい YAML ファイルを作成し、**<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) 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

user-provisioned infrastructure 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。

3. 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
```

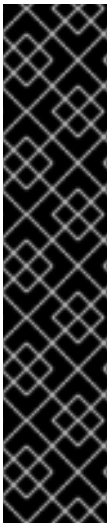
出力例

```
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
```

新しいコンピュートマシンセットが利用可能になると、**DESIRED** と **CURRENT** の値が一致します。コンピュートマシンセットが使用できない場合は、数分待ってからコマンドを再実行してください。

2.7. IBM POWER VIRTUAL SERVER 上でのコンピュートマシンセットの作成

IBM Power® Virtual Server 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、Machine API が動作しているクラスターでのみ使用できます。user-provisioned infrastructure を持つクラスターでは、Machine 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** のクラスターでは、Machine 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. IBM Power Virtual Server 上のコンピュート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 ファイルは、リージョン内の指定された IBM Power® Virtual Server ゾーンで実行され、**node-role.kubernetes.io/<role>: ""** というラベルが付いたノードを作成するコンピュート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:
      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:
        labels:
          node-role.kubernetes.io/<role>: ""
      providerSpec:
        value:
          apiVersion: machine.openshift.io/v1
```

```
      credentialsSecret:
        name: powervs-credentials
      image:
        name: rhcos-<infrastructure_id> 11
        type: Name
      keyPairName: <infrastructure_id>-key
      kind: PowerVSMachineProviderConfig
      memoryGiB: 32
      network:
        regex: ^DHCPSERVER[0-9a-z]{32}_Private$
        type: RegEx
      processorType: Shared
      processors: "0.5"
      serviceInstance:
        id: <ibm_power_vs_service_instance_id>
        type: ID 12
      systemType: s922
      userDataSecret:
        name: <role>-user-data
```

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}{"\n"}' infrastructure cluster
```

2 3 8 9 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。

2.7.2. コンピュート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. コンピュートマシンセットのカスタムリソース (CR) サンプルを含む新しい YAML ファイルを作成し、**<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) 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

user-provisioned infrastructure 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。

3 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
```

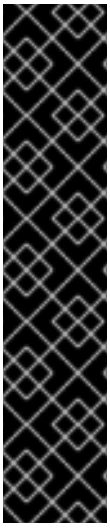
出力例

```
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
```

新しいコンピュートマシンセットが利用可能になると、**DESIRED** と **CURRENT** の値が一致します。コンピュートマシンセットが使用できない場合は、数分待ってからコマンドを再実行してください。

2.8. NUTANIX でコンピュートマシンセットを作成する

Nutanix 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、Machine API が動作しているクラスターでのみ使用できます。user-provisioned infrastructure を持つクラスターでは、Machine 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** のクラスターでは、Machine 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. Nutanix 上のコンピュート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、**node-role.kubernetes.io/<role>: ""** でラベル付けされたノードを作成する Nutanix コンピュート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。

OpenShift CLI を使用して取得した値
以下の例では、OpenShift CLI (**oc**) を使用してクラスターの値の一部を取得できます。

インフラストラクチャー ID

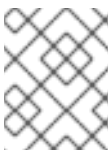
<infrastructure_id> 文字列は、クラスターをプロビジョニングしたときに設定したクラスター ID に基づくインフラストラクチャー ID です。OpenShift CLI がインストールされている場合は、以下のコマンドを実行してインフラストラクチャー ID を取得できます。

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

```
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>
  name: <infrastructure_id>-<role>-<zone> 3
  namespace: openshift-machine-api
  annotations: 4
    machine.openshift.io/memoryMb: "16384"
    machine.openshift.io/vCPU: "4"
spec:
  replicas: 3
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<zone>
  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>-<zone>
  spec:
    metadata:
      labels:
        node-role.kubernetes.io/<role>: ""
    providerSpec:
      value:
        apiVersion: machine.openshift.io/v1
        bootType: "" 5
        categories: 6
        - key: <category_name>
          value: <category_value>
        cluster: 7
          type: uuid
          uuid: <cluster_uuid>
        credentialsSecret:
          name: nutanix-credentials
        image:
          name: <infrastructure_id>-rhcos 8
          type: name
        kind: NutanixMachineProviderConfig
        memorySize: 16Gi 9
        project: 10
          type: name
          name: <project_name>
        subnets:
        - type: uuid
          uuid: <subnet_uuid>
        systemDiskSize: 120Gi 11
        userDataSecret:
          name: <user_data_secret> 12
        vcpuSockets: 4 13
        vcpusPerSocket: 1 14
```

1. **<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。
2. 追加するノードラベルを指定します。
3. 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。
4. クラスターオートスケーラーのアノテーション。
5. コンピュートマシンが使用するブートタイプを指定します。ブートタイプの詳細については、仮想化環境内の UEFI、セキュアブート、および TPM について を参照してください。有効な値は、**Legacy**、**SecureBoot**、または **UEFI** です。デフォルトは、**Legacy** です。



注記

OpenShift Container Platform 4.16 では、**Legacy** ブートタイプを使用する必要があります。

6 コンピュートマシンに適用する Nutanix Prism カテゴリーを 1 つ以上指定します。このスタンザには、Prism Central に存在するカテゴリーのキーと値のペアの **key** および **value** パラメーターが必

7 Nutanix Prism Element のクラスター設定を指定します。この例のクラスタータイプは **uuid** であるため、**uuid** スタンザがあります。

8 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コンピュートデフォルトマシンのイメージを使用します。

9 クラスターのメモリー量を Gi で指定します。

10 クラスターに使用する Nutanix プロジェクトを指定します。この例のプロジェクトタイプは **name** であるため、**name** スタンザがあります。

11 システムディスクのサイズを Gi で指定します。

12 **openshift-machine-api** namespace 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。インストールプログラムがデフォルトのコンピュートマシンセットに入力する値を使用します。

13 vCPU ソケットの数を指定します。

14 ソケットあたりの vCPU の数を指定します。

2.8.2. コンピュート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. コンピュートマシンセットのカスタムリソース (CR) サンプルを含む新しい YAML ファイルを作成し、**<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) 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

user-provisioned infrastructure 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。

3 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
```

新しいコンピュートマシンセットが利用可能になると、**DESIRED** と **CURRENT** の値が一致します。コンピュートマシンセットが使用できない場合は、数分待ってからコマンドを再実行してください。

2.8.3. Nutanix クラスターの障害ドメイン

Nutanix クラスターの障害ドメイン設定を追加または更新するには、整合性のある変更を複数のリソースに加える必要があります。次の操作が必要です。

1. クラスターインフラストラクチャーのカスタムリソース (CR) を変更します。
2. クラスターコントロールプレーンマシンセットの CR を変更します。
3. コンピュートマシンセットの CR を変更または置き換えます。

詳細は、**インストール後の設定** コンテンツの「既存の Nutanix クラスターへの障害ドメインの追加」を参照してください。

関連情報

- 既存の Nutanix クラスターへの障害ドメインの追加

2.9. OPENSTACK でコンピュートマシンセットを作成する

異なるコンピュート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、Machine API が動作しているクラスターでのみ使用できます。user-provisioned infrastructure を持つクラスターでは、Machine 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** のクラスターでは、Machine 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. RHOSP 上のコンピュート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、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コンピュート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: machine.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}{"\n"}' 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。

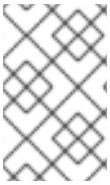
2.9.2. RHOSP 上の SR-IOV を使用するコンピュートマシンセットのカスタムリソースのサンプル YAML

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コンピュート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コンピュート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** リスト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 上のコンピュート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コンピュートマシンセットの例

```
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: machine.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> (1)
```

```
      nameSuffix: radio
      fixedIPs:
        - subnetID: <radio_subnet_UUID> 2
      tags:
        - sriov
        - radio
      vnicType: direct 3
      portSecurity: false 4
    - networkID: <uplink_network_UUID> 5
      nameSuffix: uplink
      fixedIPs:
        - subnetID: <uplink_subnet_UUID> 6
      tags:
        - sriov
        - uplink
      vnicType: direct 7
      portSecurity: false 8
  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>
```

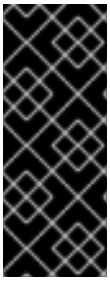
1 5 各ポートにネットワークの UUID を入力します。

2 6 各ポートのサブネット UUID を入力します。

3 7 **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。

4 8 **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ーが無効な場合は、ポートにセキュリティー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ー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 対応のコンピュート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ーで作成されるポート向けに有効にされます。これらのリスト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

2.9.3. ポートセキュリティー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ー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コンピュートマシンセットを定義します。標準の SR-IOV コンピュート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ー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
- コンピュートセキュリティーグループ
- 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ー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 を使用するコンピュートマシンセット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し、ポートセキュリティーが無効にされているコンピュートマシンセットの例

```
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: machine.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: (1)
            - ipAddress: <API_VIP_port_IP>
            - ipAddress: <ingress_VIP_port_IP>
            fixedIPs:
              - subnetID: <machines_subnet_UUID> (2)
            nameSuffix: nodes
            networkID: <machines_network_UUID> (3)
            securityGroups:
                - <compute_security_group_UUID> (4)
          - 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) コンピュート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ーで作成されるポート向けに有効にされます。これらのリスト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。

2.9.4. コンピュート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. コンピュートマシンセットのカスタムリソース (CR) サンプルを含む新しい YAML ファイルを作成し、**<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) 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

user-provisioned infrastructure 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。

3. 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
```

新しいコンピュートマシンセットが利用可能になると、**DESIRED** と **CURRENT** の値が一致します。コンピュートマシンセットが使用できない場合は、数分待ってからコマンドを再実行してください。

2.10. VSPHERE でコンピュートマシンセットを作成する

VMware vSphere 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コンピュート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、Machine API が動作しているクラスターでのみ使用できます。user-provisioned infrastructure を持つクラスターでは、Machine 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** のクラスターでは、Machine 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}'
```

2.10.1. vSphere 上のコンピュート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、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コンピュート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: vsphereprovider.openshift.io/v1beta1
          credentialsSecret:
            name: vsphere-cloud-credentials
          diskGiB: 120
          kind: VSphereMachineProviderSpec
          memoryMiB: 8192
          metadata:
            creationTimestamp: null
          network:
            devices:
            - networkName: "<vm_network_name>" 10
          numCPUs: 4
          numCoresPerSocket: 1
          snapshot: ""
          template: <vm_template_name> 11
          userDataSecret:
            name: worker-user-data
          workspace:
            datacenter: <vcenter_data_center_name> 12
            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}{"\n"}' 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

コンピュート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

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

13 コンピュート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. コンピュートマシンセット管理に最低限必要な vCenter 権限

vCenter 上の OpenShift Container Platform クラスターでコンピュート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、コンピュート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 コンピュート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
vSphere vCenter	常時	**InventoryService.Tagging.AttachTag** **InventoryService.Tagging.CreateCategory** **InventoryService.Tagging.CreateTag** **InventoryService.Tagging.DeleteCategory** **InventoryService.Tagging.DeleteTag** **InventoryService.Tagging.EditCategory** **InventoryService.Tagging.EditTag** **Sessions.ValidateSession** **StorageProfile.Update**[1] **StorageProfile.View**[1]
vSphere vCenter Cluster	常時	**Resource.AssignVMToPool**

ロールの vSphere オブジェクト	必要になる場合	必要な特権
vSphere データストア	常時	**Datastore.AllocateSpace** **Datastore.Browse**
vSphere ポートグループ	常時	**Network.Assign**
仮想マシンフォルダー	常時	**VirtualMachine.Config.AddRemoveDevice** **VirtualMachine.Config.AdvancedConfig** **VirtualMachine.Config.Annotation** **VirtualMachine.Config.CPUCount** **VirtualMachine.Config.DiskExtend** **VirtualMachine.Config.Memory** **VirtualMachine.Config.Settings** **VirtualMachine.Interact.PowerOff** **VirtualMachine.Interact.PowerOn** **VirtualMachine.Inventory.CreateFromExisting** **VirtualMachine.Inventory.Delete** **VirtualMachine.Provisioning.Clone**
vSphere vCenter データセンター	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	**Resource.AssignVMToPool** **VirtualMachine.Provisioning.DeployTemplate**

[1] **StorageProfile.Update** および **StorageProfile.View** 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、コンピュート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	常時	必須ではありません。	必要な特権が一覧表示
vSphere vCenter データセンター	既存のフォルダー	必須ではありません。	**ReadOnly** パーミッション

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
	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	常時	必須	必要な特権が一覧表示
vSphere vCenter Datastore	常時	必須ではありません。	必要な特権が一覧表示
vSphere Switch	常時	必須ではありません。	**ReadOnly** パーミッション
vSphere ポートグループ	常時	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. コンピュート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、Machine API が vCenter と対話できる必要があります。Machine 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が、**openshift-machine-api** namespace のシークレット内に存在している必要があります。

手順

1. 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。

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 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: 1
  {
    "ignition": {
      ...
      },
    ...
  }
```

1 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. コンピュート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。

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。

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 user-provisioned infrastructure を使用している場合は、その設定に応じた特定の Machine API 要件を満たしている。

手順

1. コンピュートマシンセットのカスタムリソース (CR) サンプルを含む新しい YAML ファイルを作成し、**<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) 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

user-provisioned infrastructure 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。

3 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

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ト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 1
        diskGiB: 120
        kind: VSphereMachineProviderSpec
        memoryMiB: 16384
        network:
          devices:
            - networkName: "<vm_network_name>"
        numCPUs: 4
        numCoresPerSocket: 4
        snapshot: ""
        template: <vm_template_name> 2
        userDataSecret:
          name: worker-user-data 3
        workspace:
          datacenter: <vcenter_data_center_name>
          datastore: <vcenter_datastore_name>
          folder: <vcenter_vm_folder_path>
          resourcepool: <vsphere_resource_pool>
          server: <vcenter_server_address> 4
```

1 必要な vCenter 認証情報を含む **openshift-machine-api** namespace のシークレットの名前。

2 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。

3 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。

4 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
```

新しいコンピュートマシンセットが利用可能になると、**DESIRED** と **CURRENT** の値が一致します。コンピュートマシンセットが使用できない場合は、数分待ってからコマンドを再実行してください。

2.10.5. マシンセットの使用によるマシンへのタグの追加

OpenShift Container Platform は、作成する各仮想マシンにクラスター固有のタグを追加します。インストールプログラムはこれらのタグを使用して、クラスターをアンインストールするときに削除する仮想マシンを選択します。

仮想マシンに割り当てられたクラスター固有のタグのほかに、プロビジョニングする仮想マシンに最大 10 個の vSphere タグを追加するようにマシンセットを設定できます。

前提条件

- **cluster-admin** 権限を持つアカウントを使用して、vSphere にインストールされた OpenShift Container Platform クラスターにアクセスできる。
- クラスターに関連付けられた VMware vCenter コンソールにアクセスできる。
- vCenter コンソールにタグを作成している。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. vCenter コンソールを使用して、マシンに追加するタグのタグ ID を検索します。
   a. vCenter コンソールにログインします。
   b. **Home** メニューから、**Tags & Custom Attributes**をクリックします。
   c. マシンに追加するタグを選択します。
   d. 選択したタグのブラウザー URL を使用して、タグ ID を特定します。

   タグ URL の例

   ```
   https://vcenter.example.com/ui/app/tags/tag/urn:vmomi:InventoryServiceTag:208e713c-cae3-4b7f-918e-4051ca7d1f97:GLOBAL/permissions
   ```

-

タグ ID の例

```
urn:vmomi:InventoryServiceTag:208e713c-cae3-4b7f-918e-4051ca7d1f97:GLOBAL
```

2. 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。

3. **providerSpec** フィールドの下に次の行を編集します。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
# ...
spec:
  template:
    spec:
      providerSpec:
        value:
          tagIDs: 1
          - <tag_id_value> 2
# ...
```

1 このマシンセットがプロビジョニングするマシンに追加する最大 10 個のタグのリストを指定します。

2 マシンに追加するタグの値を指定します。たとえば、**urn:vmomi:InventoryServiceTag:208e713c-cae3-4b7f-918e-4051ca7d1f97:GLOBAL** です。

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、Machine API が動作しているクラスターでのみ使用できます。user-provisioned infrastructure を持つクラスターでは、Machine 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** のクラスターでは、Machine 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ト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ト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>
10
            url: http://172.22.0.3:6181/images/rhcos-<version>.<architecture>.qcow2 11
          kind: BareMetalMachineProviderSpec
          metadata:
            creationTimestamp: null
          userData:
            name: worker-user-data
```

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}{"\n"}' 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. コンピュート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. コンピュートマシンセットのカスタムリソース (CR) サンプルを含む新しい YAML ファイルを作成し、**<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) 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

user-provisioned infrastructure 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。

3. 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
```

新しいコンピュートマシンセットが利用可能になると、**DESIRED** と **CURRENT** の値が一致します。コンピュートマシンセットが使用できない場合は、数分待ってからコマンドを再実行してください。

第3章 コンピュートマシンセットの手動スケーリング

コンピュート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コンピュートマシンセットの要素を変更する必要がある場合は、コンピュート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ーを有効にし、インストール設定からの **networking.machineNetwork[].cidr** に含まれていないコンピュートマシンをスケールアップする場合、コンピュートマシンをプロキシーオブジェクトの **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、Machine API が動作しているクラスターでのみ使用できます。user-provisioned infrastructure を持つクラスターでは、Machine 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** のクラスターでは、Machine 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. コンピュートマシンセットの手動スケーリング

コンピュート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、コンピュート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 installer-provisioned infrastructure のインストールに関連します。user-provisioned infrastructure のカスタマイズされたインストールにはコンピュート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. 次のコマンドを実行して、クラスター内のコンピュートマシンセットを表示します。

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

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

2. 次のコマンドを実行して、クラスター内のコンピュートマシンを表示します。

```
$ oc get machines.machine.openshift.io -n openshift-machine-api
```

3. 次のコマンドを実行して、削除するコンピュートマシンに注釈を設定します。

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

4. 次のいずれかのコマンドを実行して、コンピュートマシンセットをスケーリングします。

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

または、以下を実行します。

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

ヒント

または、以下の YAML を適用してコンピュート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
```

コンピュート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

- 次のコマンドを実行して、目的のマシンが削除されたことを確認します。

```
$ oc get machines.machine.openshift.io
```

3.3. コンピュートマシンセットの削除ポリシー

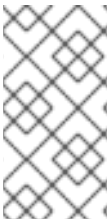
Random、**Newest**、および **Oldest** は 3 つのサポートされる削除オプションです。デフォルトは **Random** です。これは、コンピュートマシンセットのスケールダウン時にランダムなマシンが選択され、削除されることを意味します。削除ポリシーは、特定のコンピュートマシンセットを変更し、ユースケースに基づいて設定できます。

```
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/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、ワーカーのコンピュートマシンセットを **0** にスケーリングできません。



注記

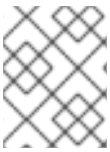
カスタムのコンピュートマシンセットは、サービスを特定のノードサービスで実行し、それらのサービスがワーカーのコンピュートマシンセットのスケールダウン時にコントローラーによって無視されるようにする必要があるユースケースで使用できます。これにより、サービスの中断が回避されます。

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 コンピュート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、コンピュートマシンセットに変更を加えることができます。



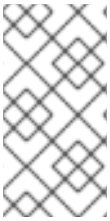
注記

他の変更なしにコンピュートマシンセットをスケーリングする必要がある場合は、コンピュート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. CLI を使用してコンピュートマシンセットを変更する

コンピュートマシンセットを変更すると、その変更は、更新された **MachineSet** カスタムリソース (CR) を保存した後に作成されたコンピュートマシンにのみ適用されます。この変更は既存のマシンには影響しません。コンピュートマシンセットをスケーリングすることで、既存のマシンを、更新された設定を反映した新しいマシンに置き換えることができます。

他の変更を加えずに、コンピュート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

デフォルトでは、OpenShift Container Platform ルーター Pod はコンピュートマシン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、コンピュートマシンセットを **0** にスケーリングできません。

この手順の出力例では、AWS クラスターの値を使用します。

前提条件

- OpenShift Container Platform クラスターは、Machine API を使用する。
- OpenShift CLI (**oc**) を使用して、管理者としてクラスターにログインしている。

手順

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

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

出力例

```
NAME                        DESIRED   CURRENT   READY   AVAILABLE   AGE
<compute_machine_set_name_1>  1       1         1       1           55m
<compute_machine_set_name_2>  1       1         1       1           55m
```

2. 次のコマンドを実行して、コンピュートマシンセットを編集します。

```
$ oc edit machinesets.machine.openshift.io <machine_set_name> \
  -n openshift-machine-api
```

3. **spec.replicas** フィールドの値をメモします。この値は、変更を適用するためにコンピュートマシンセットをスケーリングする際に必要になるためです。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  name: <machine_set_name>
  namespace: openshift-machine-api
spec:
  replicas: 2 (1)
# ...
```

(1) この手順例では、**replicas** 値が **2** のコンピュートマシンセットを示しています。

4. 必要な設定オプションを使用してコンピュートマシンセット CR を更新し、変更を保存します。

5. 次のコマンドを実行して、更新されたコンピュートマシンセットによって管理されているマシンをリスト表示します。

```
$ oc get machines.machine.openshift.io \
  -n openshift-machine-api \
  -l machine.openshift.io/cluster-api-machineset=<machine_set_name>
```

AWS クラスターの出力例

```
NAME                      PHASE     TYPE        REGION      ZONE         AGE
<machine_name_original_1>  Running   m6i.xlarge  us-west-1   us-west-1a   4h
<machine_name_original_2>  Running   m6i.xlarge  us-west-1   us-west-1a   4h
```

6. 次のコマンドを実行して、更新されたコンピュートマシンセットで管理されるマシンごとに **delete** アノテーションを設定します。

```
$ oc annotate machine.machine.openshift.io/<machine_name_original_1> \
  -n openshift-machine-api \
  machine.openshift.io/delete-machine="true"
```

7. 代わりとなるマシンを新しい設定で作成するために、次のコマンドを実行して、コンピュートマシンセットをレプリカ数の 2 倍にスケーリングします。

```
$ oc scale --replicas=4 \ (1)
  machineset.machine.openshift.io <machine_set_name> \
  -n openshift-machine-api
```

(1) 元の例の値 **2** は 2 倍の **4** になります。

8. 次のコマンドを実行して、更新されたコンピュートマシンセットによって管理されているマシンをリスト表示します。

```
$ oc get machines.machine.openshift.io \
  -n openshift-machine-api \
  -l machine.openshift.io/cluster-api-machineset=<machine_set_name>
```

AWS クラスターの出力例

```
NAME                      PHASE         TYPE        REGION      ZONE         AGE
<machine_name_original_1>  Running       m6i.xlarge  us-west-1   us-west-1a   4h
<machine_name_original_2>  Running       m6i.xlarge  us-west-1   us-west-1a   4h
<machine_name_updated_1>   Provisioned   m6i.xlarge  us-west-1   us-west-1a   55s
<machine_name_updated_2>   Provisioning  m6i.xlarge  us-west-1   us-west-1a   55s
```

新しいマシンが **Running** フェーズにある場合、コンピュートマシンセットを元のレプリカ数にスケーリングできます。

9. 古い設定で作成されたマシンを削除するために、次のコマンドを実行して、コンピュートマシンセットを元のレプリカ数にスケーリングします。

```
$ oc scale --replicas=2 \ 1
  machineset.machine.openshift.io <machine_set_name> \
  -n openshift-machine-api
```

1 元の例の値は **2** です。

検証

- 更新されたマシンセットによって作成されたマシンの設定が正しいことを確認するには、次のコマンドを実行して、新しいマシンの 1 つで CR の関連フィールドを調べます。

```
$ oc describe machine.machine.openshift.io <machine_name_updated_1> \
  -n openshift-machine-api
```

- 設定が更新されていないコンピュートマシンが削除されたことを確認するには、次のコマンドを実行して、更新されたコンピュートマシンセットによって管理されているマシンをリスト表示します。

```
$ oc get machines.machine.openshift.io \
  -n openshift-machine-api \
  -l machine.openshift.io/cluster-api-machineset=<machine_set_name>
```

AWS クラスターの削除進行中の出力例

```
NAME                      PHASE         TYPE        REGION      ZONE         AGE
<machine_name_original_1>  Deleting      m6i.xlarge  us-west-1   us-west-1a   4h
<machine_name_original_2>  Deleting      m6i.xlarge  us-west-1   us-west-1a   4h
<machine_name_updated_1>   Running       m6i.xlarge  us-west-1   us-west-1a   5m41s
<machine_name_updated_2>   Running       m6i.xlarge  us-west-1   us-west-1a   5m41s
```

AWS クラスターの削除完了時の出力例

```
NAME                      PHASE         TYPE        REGION      ZONE         AGE
<machine_name_updated_1>   Running       m6i.xlarge  us-west-1   us-west-1a   6m30s
<machine_name_updated_2>   Running       m6i.xlarge  us-west-1   us-west-1a   6m30s
```

関連情報

- マシン削除フェーズのライフサイクルフック
- コンピュートマシンセットの手動スケーリング
- スケジューラーによる Pod 配置の制御

第5章 マシンのフェーズとライフサイクル

マシンには **ライフサイクル** があり、ライフサイクルにはいくつかの定義されたフェーズがあります。マシンのライフサイクルとそのフェーズを理解すると、手順が完了したかどうかを確認したり、望ましくない動作をトラブルシューティングしたりするのに役立ちます。OpenShift Container Platform では、マシンのライフサイクルがサポート対象の全クラウドプロバイダーで一貫しています。

5.1. マシンのフェーズ

マシンのライフサイクルが進むにつれ、フェーズが変化します。各フェーズは、マシンの状態を表すための基本です。

Provisioning

新しいマシンのプロビジョニング要求があります。マシンはまだ存在せず、インスタンス、プロバイダー ID、アドレスはありません。

Provisioned

マシンが存在し、プロバイダー ID かアドレスがあります。クラウドプロバイダーがマシンのインスタンスを作成しました。マシンはまだノードになっておらず、マシンオブジェクトの **status.nodeRef** セクションにデータはありません。

Running

マシンが存在し、プロバイダー ID またはアドレスがあります。Ignition が正常に実行され、クラスターマシンの承認者は証明書署名要求 (CSR) を承認しました。マシンはノードになり、マシンオブジェクトの **status.nodeRef** セクションにノードの詳細が格納されました。

Deleting

マシンの削除要求があります。マシンオブジェクトには、削除要求の時刻を示す **DeletionTimestamp** フィールドがあります。

Failed

マシンに回復不可能な問題があります。これは、クラウドプロバイダーがマシンのインスタンスを削除した場合などに発生する可能性があります。

5.2. マシンのライフサイクル

ライフサイクルは、マシンのプロビジョニング要求から始まり、マシンが存在しなくなるまで継続します。

マシンのライフサイクルは次の順序で進行します。エラーやライフサイクルフックによる中断は、この概要には含まれていません。

1. 次のいずれかの理由で、新しいマシンをプロビジョニング要求が発生します。
   - クラスター管理者がマシンセットをスケーリングするため、追加のマシンが必要になる。
   - 自動スケーリングポリシーによりマシンセットがスケーリングされるため、追加のマシンが必要になる。
   - マシンセットが管理するマシンで障害が発生した、またはマシンセットが管理するマシンが削除され、必要なマシン数を満たすためにマシンセットが代替マシンを作成する。
2. マシンは **Provisioning** フェーズに入ります。
3. インフラストラクチャープロバイダーは、マシンのインスタンスを作成します。

4. マシンにはプロバイダー ID またはアドレスがあり、**Provisioned** フェーズに入ります。
5. Ignition 設定ファイルが処理されます。
6. kubelet は証明書署名要求 (CSR) を発行します。
7. クラスターマシンの承認者が CSR を承認します。
8. マシンはノードになり、**Running** フェーズに入ります。
9. 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不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。
10. 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。
11. マシンコントローラーは、インフラストラクチャープロバイダーからインスタンスを削除します。
12. マシンコントローラーは **Node** オブジェクトを削除します。

5.3. マシンのフェーズを確認する

マシンのフェーズは、OpenShift CLI (**oc**) または Web コンソールを使用して確認できます。この情報を使用して、手順が完了したかどうかを確認したり、望ましくない動作のトラブルシューティングを行うことができます。

5.3.1. CLI を使用してマシンのフェーズを確認する

マシンのフェーズは、OpenShift CLI (**oc**) を使用して確認できます。

前提条件

- **cluster-admin** パーミッションを持つアカウントを使用して OpenShift Container Platform クラスターにアクセスできる。
- **oc** CLI がインストールされている。

手順

- 次のコマンドを実行して、クラスター上のマシンをリスト表示します。

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

出力例

```
NAME                                 PHASE     TYPE        REGION      ZONE         AGE
mycluster-5kbsp-master-0             Running   m6i.xlarge  us-west-1   us-west-1a   4h55m
mycluster-5kbsp-master-1             Running   m6i.xlarge  us-west-1   us-west-1b   4h55m
mycluster-5kbsp-master-2             Running   m6i.xlarge  us-west-1   us-west-1a   4h55m
mycluster-5kbsp-worker-us-west-1a-fmx8t   Running   m6i.xlarge  us-west-1   us-west-1a   4h51m
mycluster-5kbsp-worker-us-west-1a-m889l   Running   m6i.xlarge  us-west-1   us-west-1a   4h51m
mycluster-5kbsp-worker-us-west-1b-c8qzm   Running   m6i.xlarge  us-west-1   us-west-1b   4h51m
```

出力の **PHASE** 列には、各マシンのフェーズが含まれます。

5.3.2. Web コンソールを使用してマシンのフェーズを確認する

OpenShift Container Platform Web コンソールを使用して、マシンのフェーズを確認できます。

前提条件

- **cluster-admin** 権限を持つアカウントを使用して OpenShift Container Platform クラスターにアクセスできる。

手順

1. **cluster-admin** ロールを持つユーザーとして、Web コンソールにログインします。
2. **Compute → Machines** に移動します。
3. **Machine** ページで、フェーズを確認するマシンの名前を選択します。
4. **Machine details** ページで **YAML** タブを選択します。
5. YAML ブロックで、**status.phase** フィールドの値を確認します。

YAML スニペットの例

```
apiVersion: machine.openshift.io/v1beta1
kind: Machine
metadata:
  name: mycluster-5kbsp-worker-us-west-1a-fmx8t
# ...
status:
  phase: Running 1
```

1 この例のフェーズは **Running** です。

5.4. 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6章 マシンの削除

特定のマシンを削除できます。

6.1. 特定マシンの削除

特定のマシンを削除できます。

重要

クラスターがコントロールプレーンマシンセットを使用していない限り、コントロールプレーンマシンを削除しないでください。

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

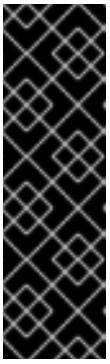
1. 次のコマンドを実行して、クラスター内のマシンを表示します。

```
$ 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。

6.2. 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。

6.2.1. 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ー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。

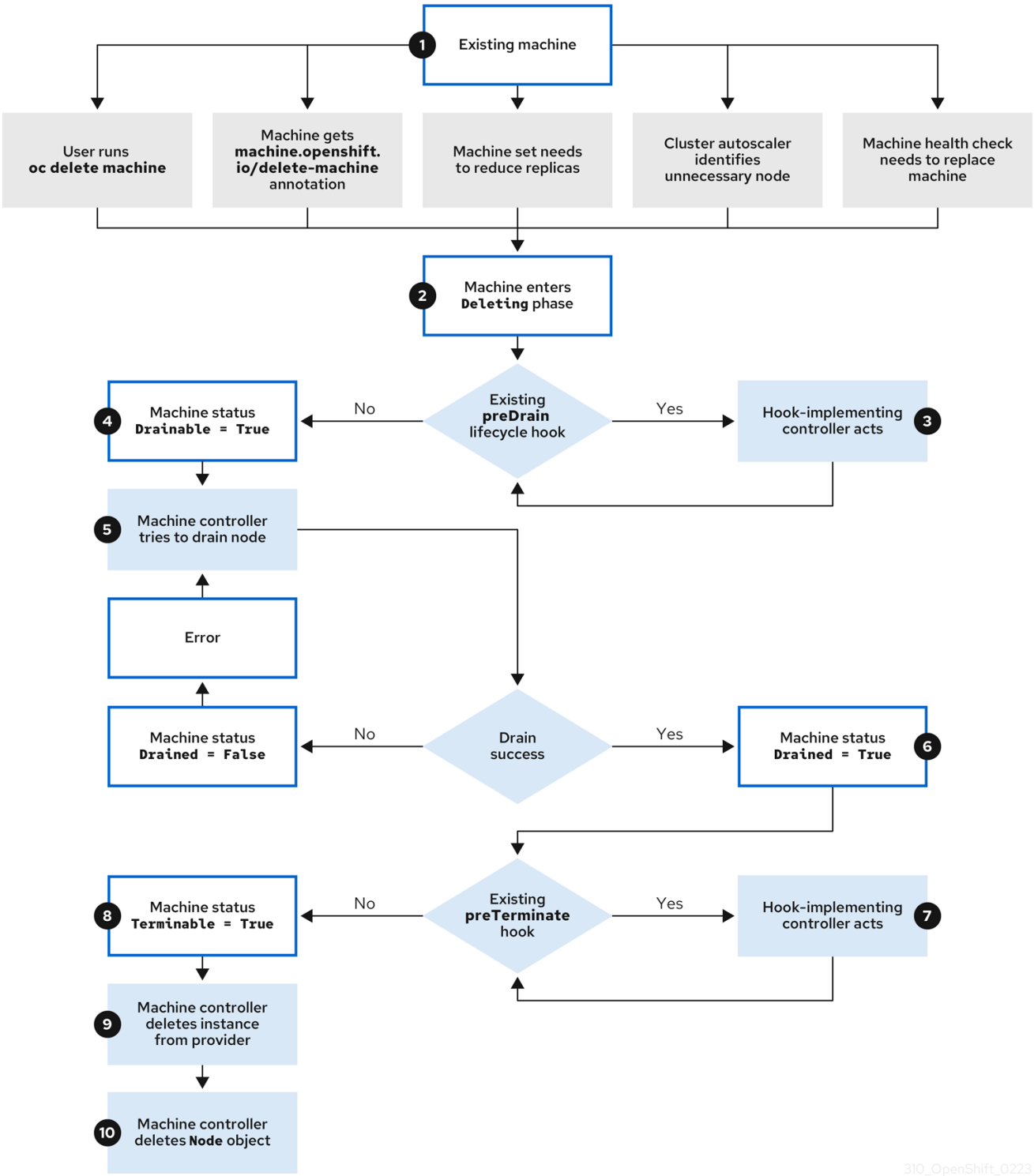
- ライフサイクルフックを追加します。
- ライフサイクルフックに応答します。
- ライフサイクルフックを削除します。

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。

6.2.2. マシン削除処理順序

OpenShift Container Platform 4.16 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6.1 マシン削除のフロー



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不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. 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ー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。

6.2.3. 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> 1
      owner: <hook_owner> 2
  ...
```

1 **preDrain** ライフサイクルフックの名前。

2 **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> 1
      owner: <hook_owner> 2
  ...
```

1 **preTerminate** ライフサイクルフックの名前。

2 **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: 1
    - name: MigrateImportantApp
      owner: my-app-migration-controller
    preTerminate: 2
    - name: BackupFileSystem
      owner: my-backup-controller
    - name: CloudProviderSpecialCase
      owner: my-custom-storage-detach-controller 3
    - name: WaitForStorageDetach
      owner: my-custom-storage-detach-controller
  ...
```

1 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。

2 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。

6.2.4. 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ー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、ログエクスポータ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と、マシンがインフラストラクチャー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、ログエクスポータ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。

6.2.5. 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 クラスターの具体的な運用上の知識がなくても、etcd Operator によって 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、Machine API Operator によって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。

6.2.5.1. 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

コントロールプレーンマシンセットを使用する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ー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 1
      owner: clusteroperator/etcd 2
  ...
```

1 **preDrain** ライフサイクルフックの名前。

2 **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。

6.3. 関連情報

- マシンのフェーズとライフサイクル
- 正常でない etcd メンバーの置き換え
- コントロールプレーンマシンセットを使用したコントロールプレーンマシンの管理

第7章 OPENSHIFT 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

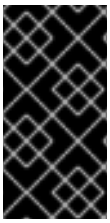
Cluster Autoscaler は、Machine API Operator が動作しているクラスターでのみ設定できます。

7.1. 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ー、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、最大メモリーリソースの制限を設定する場合、Cluster Autoscaler は現在のメモリー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別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ト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、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レ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、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 をスケジュールできますが、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。

7.1.1. 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。

7.1.1.1. Cluster Autoscaler 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
  logVerbosity: 4 (10)
  scaleDown: (11)
    enabled: true (12)
    delayAfterAdd: 10m (13)
    delayAfterDelete: 5m (14)
    delayAfterFailure: 30s (15)
    unneededTime: 5m (16)
    utilizationThreshold: "0.4" (17)
  expanders: ["Random"] (18)
```

(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 に割り当てる **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トマシン、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** および **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 ロギングの詳細レベルを **0** から **10** の間で指定します。次のログレベルのしきい値は、ガイダンスとして提供されています。

- **1**: (デフォルト) 変更に関する基本情報。
- **4**: 一般的な問題をトラブルシューティングするためのデバッグレベルの詳細度。
- **9**: 広範なプロトコルレベルのデバッグ情報。

値を指定しない場合は、デフォルト値の **1** が使用されます。

11 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。

12 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。

13 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。

14 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。

15 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。

16 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。

17 オプション: **node utilization level** を指定します。この使用率レベルを下回るノードは、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。

18 オプション: クラスターオートスケーラーで使用するエクスパンダーを指定します。次の値が有効です。

- **LeastWaste** : スケーリング後にアイドル CPU を最小限に抑えるマシンセットを選択します。複数のマシンセットで同じ量のアイドル CPU が生成される場合、選択によって未使用のメモリーが最小限に抑えられます。
- **Priority**: ユーザーが割り当てた優先度が最も高いマシンセットを選択します。このエクスパンダーを使用するには、マシンセットの優先順位を定義する config map を作成する必要があります。詳細は、「クラスターオートスケーラーの優先度エクスパンダーの設定」を参照してください。
- **Random**: (デフォルト) マシンセットをランダムに選択します。

値を指定しない場合は、デフォルト値 **Random** が使用されます。

[LeastWaste, Priority] 形式を使用して複数のエクスパンダーを指定できます。クラスターオートスケーラーは、指定された順序に従って各エクスパンダーを適用します。

[LeastWaste, Priority] の例では、クラスターオートスケーラーは最初に **LeastWaste** 基準に従って評価します。複数のマシンセットが **LeastWaste** 基準を同等に満たしている場合、クラスターオートスケーラーは **Priority** 基準に従って評価します。複数のマシンセットが指定されたエクスパンダーのすべてを同等に満たす場合、クラスターオートスケーラーはランダムに 1 つを選択して使用します。

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。

7.1.1.2. クラスターオートスケーラーの優先度エクスパンダーの設定

クラスターオートスケーラーが優先度エクスパンダーを使用する場合、ユーザーが割り当てた優先度が最も高いマシンセットを使用してスケールアップします。このエクスパンダーを使用するには、マシンセットの優先順位を定義する config map を作成する必要があります。

指定された優先レベルごとに、マシンセットの選択の優先順位を付けるときに使用するマシンセットを識別するための正規表現を作成する必要があります。正規表現は、クラスターオートスケーラーが選択対象として考慮するコンピュートマシンセットの名前と一致する必要があります。

前提条件

- Machine API を使用する OpenShift Container Platform クラスターをデプロイしている。
- **cluster-admin** 権限を持つアカウントを使用してクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. 以下のコマンドを実行して、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します。

```
$ oc get machinesets.machine.openshift.io
```

出力例

```
NAME                                DESIRED   CURRENT   READY   AVAILABLE   AGE
archive-agl030519-vplxk-worker-us-east-1c   1         1         1       1           25m
fast-01-agl030519-vplxk-worker-us-east-1a   1         1         1       1           55m
fast-02-agl030519-vplxk-worker-us-east-1a   1         1         1       1           55m
fast-03-agl030519-vplxk-worker-us-east-1b   1         1         1       1           55m
fast-04-agl030519-vplxk-worker-us-east-1b   1         1         1       1           55m
prod-01-agl030519-vplxk-worker-us-east-1a   1         1         1       1           33m
prod-02-agl030519-vplxk-worker-us-east-1c   1         1         1       1           33m
```

2. 正規表現を使用して、優先レベルを設定するコンピュートマシンセットの名前に一致する 1 つ以上のパターンを作成します。
たとえば、名前に文字列 **fast** が含まれるすべてのコンピュートマシンセットに一致させるには、正規表現パターン ***fast*** を使用します。

3. 次のように config map を定義する **cluster-autoscaler-priority-expander.yml** YAML ファイルを作成します。

優先度エクスパンダー config map の例

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-autoscaler-priority-expander (1)
  namespace: openshift-machine-api (2)
data:
  priorities: |- (3)
    10:
      - *fast*
      - *archive*
    40:
      - *prod*
```

(1) config map に **cluster-autoscaler-priority-expander** という名前を付ける必要があります。

(2) クラスターオートスケーラー Pod と同じ namespace (**openshift-machine-api** namespace) に config map を作成する必要があります。

(3) マシンセットの優先順位を定義します。

priorities の値は正の整数である必要があります。クラスターオートスケーラーは、値の低い優先度の前に値の大きい優先度を使用します。

優先度レベルごとに、使用するマシンセットに対応する正規表現を指定します。

4. 以下のコマンドを実行して config map を作成します。

```
$ oc create configmap cluster-autoscaler-priority-expander \
  --from-file=<location_of_config_map_file>/cluster-autoscaler-priority-expander.yml
```

検証

- 以下のコマンドを実行して config map を確認します。

```
$ oc get configmaps cluster-autoscaler-priority-expander -o yaml
```

次のステップ

- 優先度エクスパンダーを使用するには、**ClusterAutoscaler** リソース定義が **expanders: ["Priority"]** パラメーターを使用するように設定されていることを確認します。

7.1.2. 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. カスタム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. 以下のコマンドを実行して、クラスター内にカスタムリソースを作成します。

```
$ oc create -f <filename>.yaml 1
```

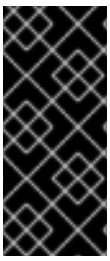
1 **<filename>** はカスタムリソースファイルの名前です。

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。

7.2. MACHINE AUTOSCALER について

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。デフォルトの **worker** コンピュートマシンセットおよび作成する他のコンピュートマシンセットの両方をスケーリングできます。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コンピュートマシンセットに即時に適用されます。

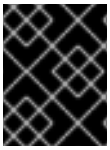


重要

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コンピュート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リングできません。

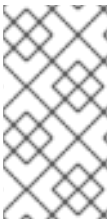
7.2.1. Machine Autoscaler の設定

Cluster Autoscaler の設定後に、クラスターのスケーリングに使用されるコンピュート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

各コンピュート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。コンピュート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。スケーリングするコンピュートマシンセットには 1 つ以上のマシンが必要です。

7.2.1.1. Machine Autoscaler 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" (1)
  namespace: "openshift-machine-api"
spec:
  minReplicas: 1 (2)
  maxReplicas: 12 (3)
  scaleTargetRef: (4)
    apiVersion: machine.openshift.io/v1beta1
    kind: MachineSet (5)
    name: worker-us-east-1a (6)
```

(1) Machine Autoscaler の名前を指定します。この Machine Autoscaler がスケーリングするコンピュートマシンセットを簡単に特定できるようにするには、スケーリングするコンピュートマシンセットの名前を指定するか、これを組み込みます。コンピュートマシンセットの名前は、**<clusterid>-<machineset>-<region>** の形式を使用します。

(2) 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コンピュート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コンピュート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。

3. Cluster Autoscaler がクラスタースケー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。Machine Autoscaler がこの数のマシンをデプロイ
4. このセクションでは、スケーリングする既存のコンピュートマシンセットを記述する値を指定します。
5. **kind** パラメーターの値は常に **MachineSet** です。
6. **name** の値は、**metadata.name** パラメーター値に示されるように既存のコンピュートマシンセットの名前に一致する必要があります。

7.2.2. 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. カスタム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. 以下のコマンドを実行して、クラスター内にカスタムリソースを作成します。

```
$ oc create -f <filename>.yaml 1
```

1. **<filename>** はカスタムリソースファイルの名前です。

7.3. 自動スケーリングの無効化

クラスター内の個々の Machine Autoscaler を無効にすることも、クラスター全体で自動スケーリングを無効にすることもできます。

7.3.1. Machine Autoscaler の無効化

Machine Autoscaler を無効にするには、対応する **MachineAutoscaler** カスタムリソース (CR) を削除します。

注記

Machine Autoscaler を無効にしても、Cluster Autoscaler は無効になりません。Cluster Autoscaler を無効にするには、「Cluster Autoscaler の無効化」に記載されている手順に従ってください。

手順

1. 次のコマンドを実行して、クラスターの **MachineAutoscaler** CR をリスト表示します。

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

出力例

```
NAME                REF KIND     REF NAME            MIN   MAX   AGE
compute-us-east-1a  MachineSet   compute-us-east-1a  1     12    39m
compute-us-west-1a  MachineSet   compute-us-west-1a  2     4     37m
```

2. オプション: 次のコマンドを実行して、**MachineAutoscaler** CR の YAML ファイルバックアップを作成します。

```
$ oc get MachineAutoscaler/<machine_autoscaler_name> \ 1
  -n openshift-machine-api \
  -o yaml> <machine_autoscaler_name_backup>.yaml 2
```

1 **<machine_autoscaler_name>** は、削除する CR の名前です。

2 **<machine_autoscaler_name_backup>** は、CR のバックアップの名前です。

3. 次のコマンドを実行して、**MachineAutoscaler** CR を削除します。

```
$ oc delete MachineAutoscaler/<machine_autoscaler_name> -n openshift-machine-api
```

出力例

```
machineautoscaler.autoscaling.openshift.io "compute-us-east-1a" deleted
```

検証

- Machine Autoscaler が無効になっていることを確認するには、次のコマンドを実行します。

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

無効化された Machine Autoscaler は、Machine Autoscaler リストに表示されません。

次のステップ

- Machine Autoscaler を再度有効にする必要がある場合は、**<machine_autoscaler_name_backup>.yaml** バックアップファイルを使用し、「Machine Autoscaler のデプロイ」に記載されている手順に従います。

関連情報

- Cluster Autoscaler の無効化
- Machine Autoscaler のデプロイ

7.3.2. Cluster Autoscaler の無効化

Cluster Autoscaler を無効にするには、対応する **ClusterAutoscaler** リソースを削除します。

注記

クラスターに既存の Machine Autoscaler がある場合も、Cluster Autoscaler を無効にするとクラスター上の自動スケーリングが無効になります。

手順

1. 次のコマンドを実行して、クラスターの **ClusterAutoscaler** リソースを一覧表示します。

```
$ oc get ClusterAutoscaler
```

出力例

```
NAME      AGE
default   42m
```

2. オプション: 次のコマンドを実行して、**ClusterAutoscaler** CR の YAML ファイルバックアップを作成します。

```
$ oc get ClusterAutoscaler/default \ (1)
  -o yaml> <cluster_autoscaler_backup_name>.yaml (2)
```

(1) **default** は、**ClusterAutoscaler** CR の名前です。

(2) **<cluster_autoscaler_backup_name>** は、CR のバックアップの名前です。

3. 次のコマンドを実行して、**ClusterAutoscaler** CR を削除します。

```
$ oc delete ClusterAutoscaler/default
```

出力例

```
clusterautoscaler.autoscaling.openshift.io "default" deleted
```

検証

- Cluster Autoscaler が無効になっていることを確認するには、次のコマンドを実行します。

```
$ oc get ClusterAutoscaler
```

予想される出力

```
No resources found
```

次のステップ

- **ClusterAutoscaler** CR を削除して Cluster Autoscaler を無効にすると、クラスターは自動スケーリングできなくなりますが、クラスター上の既存の Machine Autoscaler は削除されません。不要な Machine Autoscaler をクリーンアップするには、「Machine Autoscaler の無効化」を参照してください。
- Cluster Autoscaler を再度有効にする必要がある場合は、**<cluster_autoscaler_name_backup>.yaml** バックアップファイルを使用し、「Cluster Autoscaler のデプロイ」に記載された手順に従います。

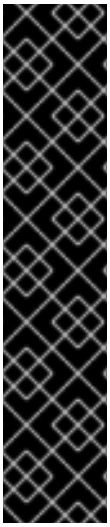
関連情報

- Machine Autoscaler の無効化
- Cluster Autoscaler のデプロイ

7.4. 関連情報

- OpenShift Container Platform における Pod スケジューリングに関する決定に Pod の優先順位を含める

第8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、Machine API が動作しているクラスターでのみ使用できます。user-provisioned infrastructure を持つクラスターでは、Machine 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** のクラスターでは、Machine 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、統合コンテナー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。Red Hat OpenShift Service Mesh は Elasticsearch をデプロイしますが、そのためには 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ー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ーゾーンを持たないグローバル Azure リージョンでは、アベイラビリティーセットを使用して高可用性を確保できます。

8.1. OPENSHIFT CONTAINER PLATFORM インフラストラクチャーコンポーネント

セルフマネージド Red Hat OpenShift の各サブスクリプションには、OpenShift Container Platform とその他の OpenShift 関連コンポーネントのエンタイトルメントが含まれています。これらのエンタイトルメントは、OpenShift Container Platform のコントロールプレーンおよびインフラストラクチャーのワークロードを実行するために含まれています。サイジング時にこれらのエンタイトルメントを考慮する必要はありません。

インフラストラクチャーノードとしての要件を満たし、含まれるエンタイトルメントを使用するには、(エンドユーザーのアプリケーションに含まれない) クラスターをサポートするコンポーネントだけを、それらのインスタンス上で実行します。たとえば、次のコンポーネントがあります。

- 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
- 統合コンテナー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
- Red Hat Quay
- Red Hat OpenShift Data Foundation
- Red Hat Advanced Cluster Management for Kubernetes
- Kubernetes 用 Red Hat Advanced Cluster Security
- Red Hat OpenShift GitOps
- Red Hat OpenShift Pipelines
- Red Hat OpenShift Service Mesh

他のコンテナー、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。

8.2. 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 つ以上のコンピュートマシンセットをデプロイすることが推奨されます。Red Hat OpenShift Service Mesh は Elasticsearch をデプロイしますが、そのためには 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ー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ーゾーンに 1 つずつ、3 つの異なるコンピュートマシンセットが必要です。複数のアベイラビリティーゾーンを持たないグローバル Azure リージョンでは、アベイラビリティーセットを使用して高可用性を確保できます。

8.2.1. さまざまなクラウドのインフラストラクチャーマシンセットの作成

クラウド用のサンプルコンピュートマシンセットを使用します。

8.2.1.1. Alibaba Cloud のコンピューティング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コンピュート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
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}{"\n"}' 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コンピュートデフォルトマシンのイメージを使用します。

(12) コンピュート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) コンピュート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コンピュート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コンピュート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。

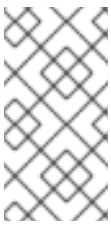
(16)(18)(20) コンピュートマシンセットに使用するタグを指定します。少なくとも、この例に示されているタグを、クラスターに適切な値とともに含める必要があります。必要に応じて、インストーラーが作成するデフォルトのコンピュートマシンセットに入力するタグなど、追加のタグを含めることができます。

(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コンピューティング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** namespace 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コンピュート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コンピュートマシンセットには、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

のタグは、**spec.template.spec.providerSpec.value** リストの **securityGroups**、**tag**、および **vSwitch** パラメーターに設定されます。

追加のマシンをデプロイするコンピュート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、作成するコンピュート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、デフォルトのコンピュート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、コンピュート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、コンピュート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、コンピュート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。

8.2.1.2. AWS 上のコンピュート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、Amazon Web Services (AWS) Local Zone の **us-east-1a** 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コンピュート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>-infra-<zone> 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>-infra-<zone> 4
  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
      providerSpec:
        value:
          ami:
            id: ami-046fe691f52a953f9 10
          apiVersion: machine.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
            - name: <custom_tag_name> 17
              value: <custom_tag_value> 18
```

```
      userDataSecret:
        name: worker-user-data
    taints: 19
      - key: node-role.kubernetes.io/infra
        effect: NoSchedule
```

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}{"\n"}' infrastructure cluster
```

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

6 7 9 **infra** ロール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) Amazon Machine Image (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}{"\n"}' \
    get machineset/<infrastructure_id>-<role>-<zone>
```

17 18 オプション: クラスターのカスタムタグデータを指定します。たとえば、**name:value** のペアである **Email:admin-email@example.com** を指定して、管理者の連絡先電子メールアドレスを追加できます。

注記

カスタムタグは、インストール中に **install-config.yml** ファイルで指定することもできます。**install-config.yml** ファイルとマシンセットに同じ **名前** のデータを持つタグが含まれている場合、マシンセットのタグの値が **install-config.yml** ファイルのタグの値よりも優先されます。

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。

19 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。

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。

8.2.1.3. Azure 上のコンピュート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コンピュート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
  name: <infrastructure_id>-infra-<region> 3
  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>-infra-<region>
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>
        machine.openshift.io/cluster-api-machine-role: infra
        machine.openshift.io/cluster-api-machine-type: infra
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra-<region>
    spec:
      metadata:
        creationTimestamp: null
        labels:
          machine.openshift.io/cluster-api-machineset: <machineset_name>
          node-role.kubernetes.io/infra: ""
      providerSpec:
        value:
          apiVersion: azureproviderconfig.openshift.io/v1beta1
          credentialsSecret:
            name: azure-cloud-credentials
            namespace: openshift-machine-api
          image: 4
            offer: ""
            publisher: ""
            resourceID: /resourceGroups/<infrastructure_id>-
rg/providers/Microsoft.Compute/galleries/gallery_<infrastructure_id>/images/<infrastructure_id>-
gen2/versions/latest 5
            sku: ""
            version: ""
```

```
      internalLoadBalancer: ""
      kind: AzureMachineProviderSpec
      location: <region> 6
      managedIdentity: <infrastructure_id>-identity
      metadata:
        creationTimestamp: null
      natRule: null
      networkResourceGroup: ""
      osDisk:
        diskSizeGB: 128
        managedDisk:
          storageAccountType: Premium_LRS
        osType: Linux
      publicIP: false
      publicLoadBalancer: ""
      resourceGroup: <infrastructure_id>-rg
      sshPrivateKey: ""
      sshPublicKey: ""
      tags:
        - name: <custom_tag_name> 7
          value: <custom_tag_value>
      subnet: <infrastructure_id>-<role>-subnet
      userDataSecret:
        name: worker-user-data
      vmSize: Standard_D4s_v3
      vnet: <infrastructure_id>-vnet
      zone: "1" 8
    taints: 9
    - key: node-role.kubernetes.io/infra
      effect: NoSchedule
```

1 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 **infra** ノードラベルを指定します。

3 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。

4 コンピュート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

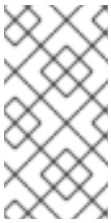
5 インスタンスタイプと互換性のあるイメージを指定します。インストールプログラムによって作成された Hyper-V 世代の V2 イメージには接尾辞 **-gen2** が付いていますが、V1 イメージには接尾辞

6 マシンを配置するリージョンを指定します。

7 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。

8 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。

9 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 できます。

関連情報

- Azure Marketplace イメージの選択

8.2.1.4. Azure Stack Hub 上のコンピュート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コンピュート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:
          node-role.kubernetes.io/infra: "" 11
      taints: 12
      - key: node-role.kubernetes.io/infra
        effect: NoSchedule
      providerSpec:
        value:
          apiVersion: machine.openshift.io/v1beta1
          availabilitySet: <availability_set> 13
          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

```
$ 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
```

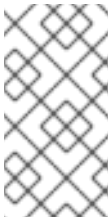
以下のコマンドを実行して vnet を取得できます。

```
$ 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。



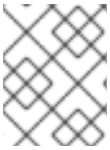
注記

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。

8.2.1.5. IBM Cloud 上のコンピュート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コンピュート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}{"\n"}' 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® instance profile を指定します。

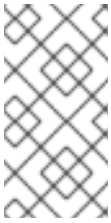
14 マシンを配置するリージョンを指定します。

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。

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テイント。



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。

8.2.1.6. GCP 上のコンピュート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、Google Cloud Platform (GCP) で実行されるコンピューティングマシンセットを定義し、**node-role.kubernetes.io/infra: ""** でラベル付けされたノードを作成します。ここで、**infra** は追加するノードラベルです。

OpenShift CLI を使用して取得した値
以下の例では、OpenShift CLI を使用してクラスターの値の一部を取得できます。

インフラストラクチャー ID

<infrastructure_id> 文字列は、クラスターをプロビジョニングしたとき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。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
```

イメージパス

<path_to_image> 文字列は、ディスクの作成に使用されたイメージへのパスです。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}{"\n"}' \
  get machineset/<infrastructure_id>-worker-a
```

サンプル GCP MachineSet 値

```
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> 3
            labels: null
            sizeGb: 128
            type: pd-ssd
          gcpMetadata: 4
          - 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> 5
        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: 6
      - key: node-role.kubernetes.io/infra
        effect: NoSchedule
```

1. **<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。

2. **<infra>** には、**<infra>** ノードラベルを指定します。

3. 現在のコンピュートマシンセットで使用されるイメージへのパスを指定します。

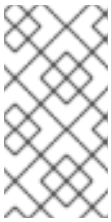
   GCP Marketplace イメージを使用するには、使用するオファーを指定します。

   - OpenShift Container Platform: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-ocp-413-x86-64-202305021736**
   - OpenShift Platform Plus: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-opp-413-x86-64-202305021736**
   - OpenShift Kubernetes Engine: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-oke-413-x86-64-202305021736**

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。

GCP で実行しているマシンセットは、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。

8.2.1.7. Nutanix 上のコンピュート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、**node-role.kubernetes.io/infra: ""** でラベル付けされたノードを作成する Nutanix コンピュート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。

OpenShift CLI を使用して取得した値
以下の例では、OpenShift CLI (**oc**) を使用してクラスターの値の一部を取得できます。

インフラストラクチャー ID

<infrastructure_id> 文字列は、クラスターをプロビジョニングしたとき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。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
```

```
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>
  name: <infrastructure_id>-<infra>-<zone> 3
  namespace: openshift-machine-api
  annotations: 4
    machine.openshift.io/memoryMb: "16384"
    machine.openshift.io/vCPU: "4"
spec:
  replicas: 3
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<infra>-<zone>
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>
        machine.openshift.io/cluster-api-machine-role: <infra>
        machine.openshift.io/cluster-api-machine-type: <infra>
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<infra>-<zone>
    spec:
      metadata:
        labels:
          node-role.kubernetes.io/infra: ""
      providerSpec:
        value:
          apiVersion: machine.openshift.io/v1
          bootType: "" 5
          categories: 6
          - key: <category_name>
            value: <category_value>
```

```
        cluster: 7
          type: uuid
          uuid: <cluster_uuid>
        credentialsSecret:
          name: nutanix-credentials
        image:
          name: <infrastructure_id>-rhcos 8
          type: name
        kind: NutanixMachineProviderConfig
        memorySize: 16Gi 9
        project: 10
          type: name
          name: <project_name>
        subnets:
        - type: uuid
          uuid: <subnet_uuid>
        systemDiskSize: 120Gi 11
        userDataSecret:
          name: <user_data_secret> 12
        vcpuSockets: 4 13
        vcpusPerSocket: 1 14
      taints: 15
      - key: node-role.kubernetes.io/infra
        effect: NoSchedule
```

1. **<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。
2. **<infra>** ノードラベルを指定します。
3. 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。
4. クラスターオートスケーラーのアノテーション。
5. コンピュートマシンが使用するブートタイプを指定します。ブートタイプの詳細については、仮想化環境内の UEFI、セキュアブート、および TPM について を参照してください。有効な値は、**Legacy**、**SecureBoot**、または **UEFI** です。デフォルトは、**Legacy** です。

   注記

   OpenShift Container Platform 4.16 では、**Legacy** ブートタイプを使用する必要があります。

6. コンピュートマシンに適用する Nutanix Prism カテゴリーを 1 つ以上指定します。このスタンザには、Prism Central に存在するカテゴリーのキーと値のペアの **key** および **value** パラメーターが必要です。カテゴリーの詳細は、カテゴリー管理 を参照してください。
7. Nutanix Prism Element のクラスター設定を指定します。この例のクラスタータイプは **uuid** であるため、**uuid** スタンザがあります。
8. 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コンピュートデフォルトマシンのイメージを使用します。
9. クラスターのメモリー量を Gi で指定します。

10 クラスターに使用する Nutanix プロジェクトを指定します。この例のプロジェクトタイプは **name** であるため、**name** スタンザがあります。

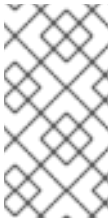
11 システムディスクのサイズを Gi で指定します。

12 **openshift-machine-api** namespace 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。インストールプログラムがデフォルトのコンピュートマシンセットに入力する値を使用します。

13 vCPU ソケットの数を指定します。

14 ソケットあたりの vCPU の数を指定します。

15 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。

8.2.1.8. RHOSP 上のコンピュート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、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コンピュート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 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: machine.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>
```

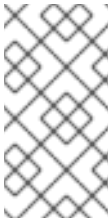
(1) (5) (7) (14) (16) (17) (18) (19) クラスターのプロビジョニング時に設定したクラスター ID を基にするインフラストラクチャー ID を指定します。OpenShift CLI がインストールされている場合は、以下のコマンドを実行してインフラストラクチャー ID を取得できます。

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

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、このリストには、**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。

8.2.1.9. vSphere 上のコンピュート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、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コンピュート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> 1
  name: <infrastructure_id>-infra 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>-infra 4
  template:
    metadata:
      creationTimestamp: null
      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 8
    spec:
      metadata:
        creationTimestamp: null
        labels:
          node-role.kubernetes.io/infra: "" 9
      taints: 10
```

```
      - 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>" (11)
          numCPUs: 4
          numCoresPerSocket: 1
          snapshot: ""
          template: <vm_template_name> (12)
          userDataSecret:
            name: worker-user-data
          workspace:
            datacenter: <vcenter_data_center_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}{"\n"}' 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) コンピュート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) コンピュートマシンセットをデプロイする vCenter Datacenter を指定します。

14 コンピュート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。

8.2.2. コンピュート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. コンピュートマシンセットのカスタムリソース (CR) サンプルを含む新しい YAML ファイルを作成し、**<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) 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

user-provisioned infrastructure 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。

3. 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
```

新しいコンピュートマシンセットが利用可能になると、**DESIRED** と **CURRENT** の値が一致します。コンピュートマシンセットが使用できない場合は、数分待ってからコマンドを再実行してください。

8.2.3. 専用インフラストラクチャーノードの作成

重要

installer-provisioned infrastructure 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** とも呼ばれる) 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=""
```

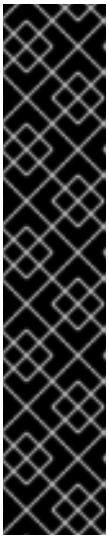
2. インフラストラクチャーノードとして機能する必要のあるワーカーノードにラベルを追加します。

```
$ 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スコープのセレクターを作成するには、以下を実行します。デフォルトのノードセレクターはすべての 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ターとの交差が作成され、Pod のセレクターをさらに制限します。



重要

デフォルトのノードセレクターのキーが 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ターは適用されません。

ただし、Pod がスケジュール対象外になる可能性のあるデフォルトノードセレクター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ターを **node-role.kubernetes.io/infra=""** などの特定のノードロールに設定すると、Pod がスケジュール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ター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ターを使用して、クラスター全体でのノードセレクター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ターと共に **defaultNodeSelector** フィールドを追加します。

```
apiVersion: config.openshift.io/v1
kind: Scheduler
metadata:
  name: cluster
spec:
  defaultNodeSelector: node-role.kubernetes.io/infra="" 1
# ...
```

1 この例のノードセレクターは、デフォルトでインフラストラクチャーノードに 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。

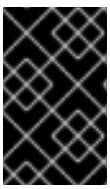
これで、インフラストラクチャーリソースを新しくラベル付けされた **infra** ノードに移動できます。

関連情報

- リソースのインフラストラクチャーマシンセットへの移行

8.2.4. インフラストラクチャーマシンのマシン設定プール作成

インフラストラクチャー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。



重要

カスタムマシン設定プールを作成すると、デフォルトのワーカープール設定がオーバーライドされます (デフォルトのワーカープール設定が同じファイルまたはユニットを参照する場合)。

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レクターとして含まれるマシン設定プールを作成します。

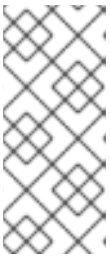
```
$ cat infra.mcp.yaml
```

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]} (1)
  nodeSelector:
    matchLabels:
      node-role.kubernetes.io/infra: "" (2)
```

(1) ワーカーロールおよびカスタムロールを追加します。

(2) 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
```

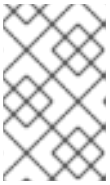
出力例

```
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 (1)
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
      - path: /etc/infratest
        mode: 0644
        contents:
          source: data:,infra
```

(1) 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                    0                    91m
worker   rendered-worker-60e35c2e99f42d976e084fa94da4d0fc   True      False      False
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。

8.3. 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、テイントを、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。

8.3.1. 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、テイントおよび容認 (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:NoSchedule
```

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: NoSchedule
      value: reserved
  ...
```

この例では、テイントを、**node-role.kubernetes.io/infra** キーおよび **NoSchedule** effect のテイントを持つ **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 のみをスケジュールしますが、既存の Pod はノードにスケジュール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。

c. 上記の NoSchedule Effect のテイントとともに、NoExecute Effect のテイントを追加します。

```
$ 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、**node-role.kubernetes.io/infra** キーおよび **NoExecute** effect のテイントを持つ **node1** に配置します。**NoExecute** effect を持つノードは、テイントを容認する Pod のみをスケジュールします。effect は、一致する容認を持たないノードから既存の Pod を削除します。

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: NoSchedule (1)
    key: node-role.kubernetes.io/infra (2)
    value: reserved (3)
  - effect: NoExecute (4)
    key: node-role.kubernetes.io/infra (5)
    operator: Exists (6)
    value: reserved (7)
```

(1) ノードに追加した effect を指定します。

(2) ノードに追加したキーを指定します。

(3) ノードに追加したキーと値のペア Taint の値を指定します。

(4) ノードに追加した effect を指定します。

(5) ノードに追加したキーを指定します。

(6) **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。

(7) 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

- ノードへの Pod のスケジューリングに関する一般的な情報については、Controlling pod placement using the scheduler を参照してください。
- 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ーマシンセットへの移動 について参照してください。
- テイントのさまざまな影響の詳細は、テイントと容認範囲について を参照してください。

8.4. 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

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レクターを追加して、作成したインフラストラクチャーマシンセットにそれらを移動できます。

```
spec:
  nodePlacement: 1
    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、移動する必要の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。

8.4.1. ルーターの移動

ルーター Pod を異なるコンピュート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

- 追加のコンピュートマシンセットを 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: 1
        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、移動する必要の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 のリストを表示し、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> 1
   ```

   1 Pod のリストより取得した **<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   v1.29.4
   ```

   ロールのリストに **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。

8.4.2. 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

- 追加のコンピュートマシンセットを 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-a524-12ffeee2931b
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: (1)
    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、移動する必要の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** リストにあることを確認します。

8.4.3. 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** config map 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: 1
        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
  metricsServer:
    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
  telemeterClient:
    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
  monitoringPlugin:
    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、移動する必要の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。

8.4.4. Vertical Pod Autoscaler Operator コンポーネントの移動

Vertical Pod Autoscaler Operator (VPA) は、レコメンダー、アップデーター、アドミッションコントローラーの 3 つのコンポーネントで設定されます。Operator と各コンポーネントには、コントロールプレーンノードの VPA namespace に独自の Pod があります。VPA Operator およびコンポーネント Pod をインフラストラクチャーノードに移動するには、ノードセレクターを VPA サブスクリプションおよび **VerticalPodAutoscalerController** CR に追加します。

以下の例は、VPA Pod のコントロールプレーンノードへのデフォルトのデプロイメントを示しています。

出力例

```
NAME                                        READY   STATUS    RESTARTS   AGE     IP          NODE
NOMINATED NODE   READINESS GATES
vertical-pod-autoscaler-operator-6c75fcc9cd-5pb6z   1/1     Running   0          7m59s   10.128.2.24
c416-tfsbj-master-1   <none>           <none>
vpa-admission-plugin-default-6cb78d6f8b-rpcrj       1/1     Running   0          5m37s   10.129.2.22
c416-tfsbj-master-1   <none>           <none>
vpa-recommender-default-66846bd94c-dsmpp            1/1     Running   0          5m37s   10.129.2.20
```

```
c416-tfsbj-master-0   <none>          <none>
vpa-updater-default-db8b58df-2nkvf                1/1     Running   0          5m37s   10.129.2.21   c416-
tfsbj-master-1   <none>          <none>
```

手順

1. VPA Operator の **サブスクリプション** カスタムリソース (CR) にノードセレクターを追加して、VPA Operator Pod を移動します。

   a. CR を編集します。

   ```
   $ oc edit Subscription vertical-pod-autoscaler -n openshift-vertical-pod-autoscaler
   ```

   b. infra ノードのノードロールラベルに一致するノードセレクターを追加します。

   ```
   apiVersion: operators.coreos.com/v1alpha1
   kind: Subscription
   metadata:
     labels:
       operators.coreos.com/vertical-pod-autoscaler.openshift-vertical-pod-autoscaler: ""
     name: vertical-pod-autoscaler
   # ...
   spec:
     config:
       nodeSelector:
         node-role.kubernetes.io/infra: "" (1)
   ```

   (1) infra ノードのノードロールを指定します。

注記

infra ノードがテイントを使用する場合は、容認を **Subscription** CR に追加する必要があります。

以下に例を示します。

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  labels:
    operators.coreos.com/vertical-pod-autoscaler.openshift-vertical-pod-
autoscaler: ""
  name: vertical-pod-autoscaler
# ...
spec:
  config:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
    tolerations: (1)
    - key: "node-role.kubernetes.io/infra"
      operator: "Exists"
      effect: "NoSchedule"
```

1 infra ノードのテイントの容認を指定します。

2. **VerticalPodAutoScaler** カスタムリソース (CR) にノードセレクターを追加して、各 VPA コンポーネントを移動します。

   a. CR を編集します。

```
$ oc edit VerticalPodAutoscalerController default -n openshift-vertical-pod-autoscaler
```

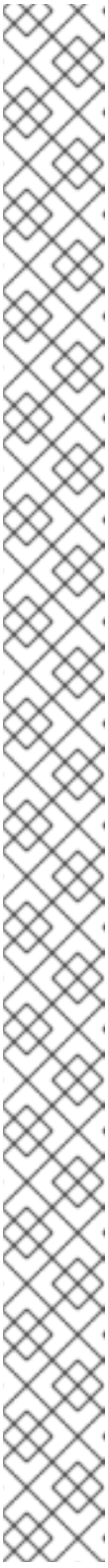
   b. infra ノードのノードロールラベルに一致するようにノードセレクターを追加します。

```
apiVersion: autoscaling.openshift.io/v1
kind: VerticalPodAutoscalerController
metadata:
 name: default
  namespace: openshift-vertical-pod-autoscaler
# ...
spec:
  deploymentOverrides:
    admission:
      container:
        resources: {}
      nodeSelector:
        node-role.kubernetes.io/infra: "" (1)
    recommender:
      container:
        resources: {}
      nodeSelector:
        node-role.kubernetes.io/infra: "" (2)
    updater:
      container:
        resources: {}
      nodeSelector:
        node-role.kubernetes.io/infra: "" (3)
```

1 オプション: VPA アドミッション Pod のノードロールを指定します。

2 オプション: VPA レコメンダー Pod のノードロールを指定します。

3 オプション: VPA アップデータ Pod のノードロールを指定します。



注記

ターゲットノードがテイントを使用する場合は、**VerticalPodAutoscalerController** CR に許容範囲を追加する必要があります。

以下に例を示します。

```
apiVersion: autoscaling.openshift.io/v1
kind: VerticalPodAutoscalerController
metadata:
 name: default
  namespace: openshift-vertical-pod-autoscaler
# ...
spec:
  deploymentOverrides:
    admission:
      container:
        resources: {}
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      tolerations: 1
      - key: "my-example-node-taint-key"
        operator: "Exists"
        effect: "NoSchedule"
    recommender:
      container:
        resources: {}
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      tolerations: 2
      - key: "my-example-node-taint-key"
        operator: "Exists"
        effect: "NoSchedule"
    updater:
      container:
        resources: {}
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      tolerations: 3
      - key: "my-example-node-taint-key"
        operator: "Exists"
        effect: "NoSchedule"
```

1. infra ノード上のテイントに対するアドミッションコントローラー Pod の許容範囲を指定します。
2. infra ノード上のテイントに対する推奨 Pod の許容を指定します。
3. infra ノードのテイントのアップデーター Pod の容認を指定します。

検証

- 次のコマンドを使用して、Pod が移動したことを確認できます。

```
$ oc get pods -n openshift-vertical-pod-autoscaler -o wide
```

Pod はコントロールプレーンノードにデプロイされなくなりました。

出力例

```
NAME                                        READY   STATUS    RESTARTS   AGE     IP
NODE                          NOMINATED NODE   READINESS GATES
vertical-pod-autoscaler-operator-6c75fcc9cd-5pb6z   1/1     Running   0          7m59s
10.128.2.24   c416-tfsbj-infra-eastus3-2bndt   <none>           <none>
vpa-admission-plugin-default-6cb78d6f8b-rpcrj       1/1     Running   0          5m37s
10.129.2.22   c416-tfsbj-infra-eastus1-lrgj8   <none>           <none>
vpa-recommender-default-66846bd94c-dsmpp            1/1     Running   0          5m37s
10.129.2.20   c416-tfsbj-infra-eastus1-lrgj8   <none>           <none>
vpa-updater-default-db8b58df-2nkvf                  1/1     Running   0          5m37s   10.129.2.21
c416-tfsbj-infra-eastus1-lrgj8   <none>           <none>
```

関連情報

- モニタリングコンポーネントの異なるノードへの移動
- ノードセレクターを使用したロギングリソースの移動
- テイントと容認を使用したロギング Pod の配置制御

第9章 RHEL コンピュートマシンの OPENSHIFT CONTAINER PLATFORM クラスターへの追加

OpenShift Container Platform では、Red Hat Enterprise Linux (RHEL) コンピュートマシンを、**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、コンピュート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。

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

OpenShift Container Platform 4.16 では、**x86_64** アーキテクチャー上で user-provisioned infrastructure インストールまたは installer-provisioned infrastructure 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トマシンとして 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 コンピュート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。

installer-provisioned infrastructure クラスターの場合、installer-provisioned infrastructure 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 コンピュートマシンは、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 コンピュートノードのシステム要件

OpenShift Container Platform 環境の Red Hat Enterprise Linux (RHEL) コンピュート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ト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、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 で実行されるインスタンス。

- ベース OS: "最小" インストールオプションを備えた RHEL 8.6 以降。

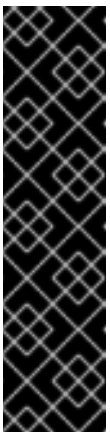
重要

OpenShift Container Platform クラスターへの RHEL 7 コンピュート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 コンピュートマシンがある場合、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、「ノードの削除」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、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。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

- NetworkManager 1.0 以降。
- 1 vCPU。
- 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できる必要があります。配置されているネットワークセキュリティーアクセス制御は、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

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、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。コンピュート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。

9.3.1. AWS で利用可能な最新の RHEL イメージのリスト表示

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 のリスト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) のリストを表示します。

```
$ aws ec2 describe-images --owners 309956199498 \ 1
--query 'sort_by(Images, &CreationDate)[*].[CreationDate,Name,ImageId]' \ 2
--filters "Name=name,Values=RHEL-8.4*" \ 3
--region us-east-1 \ 4
--output table 5
```

1. **--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。

2. **--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

```
------------------------------------------------------------------------------------------------------
|                                          DescribeImages                                            |
+--------------------------+-----------------------------------------------------+-----------------------+
| 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. 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 4.16 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できる必要があります。

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 プロキシーまたは 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リスト表示します。

   ```
   # 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 4.16 で必要なリポジトリーを有効にします。

```
# subscription-manager repos \
    --enable="rhel-8-for-x86_64-baseos-rpms" \
    --enable="rhel-8-for-x86_64-appstream-rpms" \
    --enable="rhocp-4.16-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ーを提供し、Ansible Playbook などのクラスターに RHEL コンピュート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。

9.5. RHEL コンピュートノードの準備

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、必要なリポジトリー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リスト表示します。

```
# 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 リポジトリーをすべて無効にします。

   a. 有効にされている RHSM リポジトリーをすべて無効にします。

   ```
   # subscription-manager repos --disable="*"
   ```

   b. 残りの yum リポジトリーをリスト表示し、**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 リポジトリーを無効にします。

   ```
   # yum-config-manager --disable <repo_id>
   ```

   または、すべてのリポジトリーを無効にします。

   ```
   # yum-config-manager --disable \*
   ```

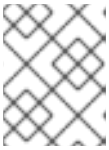
   利用可能なリポジトリーが多い場合には、数分の時間がかかることがあります。

6. OpenShift Container Platform 4.16 で必要なリポジトリー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-4.16-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。

9.6. 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

- IAM ロールに必要な AWS パーミッション を参照してください。

9.7. 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 コンピュートマシンの場合、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ーグループに **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。

9.8. RHEL コンピュート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トマシンを、OpenShift Container Platform 4.16 クラスターに追加できます。

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. コンピュート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 (1)
#ansible_become=True (2)

openshift_kubeconfig_path="~/.kube/config" (3)

[new_workers] (4)
mycluster-rhel8-0.example.com
mycluster-rhel8-1.example.com
```

(1) Ansible タスクをリモートコンピュートマシンで実行するユーザー名を指定します。

(2) **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。

(3) クラスターの **kubeconfig** ファイルへのパスを指定します。

(4) クラスターに追加する各 RHEL マシンをリスト表示します。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 (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。

9.9. 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、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

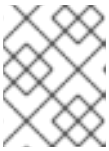
1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.29.4
master-1  Ready     master  63m  v1.29.4
master-2  Ready     master  64m  v1.29.4
```

出力には作成したすべてのマシンがリスト表示されます。



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。このリスト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 user-provisioned infrastructure 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> 1
```

1 **<csr_name>** は、現行の CSR のリストからの 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 のリストからの 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
```

6. 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  v1.29.4
master-1  Ready     master  73m  v1.29.4
master-2  Ready     master  74m  v1.29.4
worker-0  Ready     worker  11m  v1.29.4
worker-1  Ready     worker  11m  v1.29.4
```

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。

9.10. ANSIBLE ホストファイルの必須パラメーター

Red Hat Enterprise Linux (RHEL) コンピュート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
ansible_user	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 **root** です。
ansible_become	**ansible_user** の値が root ではない場合、**ansible_become** を **True** に設定する必要があり、**ansible_user** 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。	**True**。値が **True** 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。
openshift_kubeconfig_path	クラスターの **kubeconfig** 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。

9.10.1. オプション: RHCOS コンピュートマシンのクラスターからの削除

Red Hat Enterprise Linux (RHEL) コンピュート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) コンピュートマシンを削除し、リソースを解放できます。

前提条件

- RHEL コンピュートマシンをクラスターに追加済みです。

手順

1. マシンのリストを表示し、RHCOS コンピュー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 コンピュートマシンについて、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> 1
```

   1 RHCOS コンピュート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 1
```

   1 分離した RHCOS コンピュート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> 1
```

   1 ドレイン (解放) した RHCOS コンピュートマシンのノード名を指定します。

3. コンピュートマシンのリストを確認し、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ト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 コンピュート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10章 RHEL コンピュートマシンの OPENSHIFT CONTAINER PLATFORM クラスターへのさらなる追加

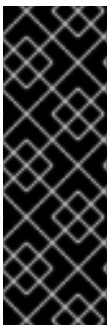
OpenShift Container Platform クラスターに Red Hat Enterprise Linux (RHEL) コンピュートマシン (またはワーカーマシンとしても知られる) がすでに含まれる場合、RHEL コンピュートマシンをさらに追加することができます。

10.1. RHEL コンピュートノードのクラスターへの追加について

OpenShift Container Platform 4.16 では、**x86_64** アーキテクチャー上で user-provisioned infrastructure インストールまたは installer-provisioned infrastructure 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トマシンとして 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 コンピュート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。

installer-provisioned infrastructure クラスターの場合、installer-provisioned infrastructure 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 コンピュートマシンは、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。

10.2. RHEL コンピュートノードのシステム要件

OpenShift Container Platform 環境の Red Hat Enterprise Linux (RHEL) コンピュート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ト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、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 で実行されるインスタンス。

- ベース OS: "最小" インストールオプションを備えた RHEL 8.6 以降。

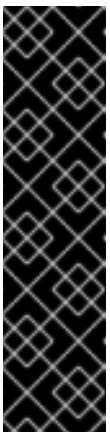
重要

OpenShift Container Platform クラスターへの RHEL 7 コンピュート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 コンピュートマシンがある場合、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、「ノードの削除」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、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。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

- NetworkManager 1.0 以降。
- 1 vCPU。
- 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できる必要があります。配置されているネットワークセキュリティーアクセス制御は、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

10.2.1. 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。

10.3. 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、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。コンピュート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 をリスト表示する必要があります。

10.3.1. AWS で利用可能な最新の RHEL イメージのリスト表示

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 のリスト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) のリストを表示します。

```
$ aws ec2 describe-images --owners 309956199498 \ 1
--query 'sort_by(Images, &CreationDate)[*].[CreationDate,Name,ImageId]' \ 2
--filters "Name=name,Values=RHEL-8.4*" \ 3
--region us-east-1 \ 4
--output table 5
```

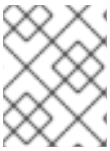
1 **--owners** コマンドオプションは、アカウント ID **309956199498** に基づいて Red Hat イメージを表示します。

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。

2 **--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

```
----------------------------------------------------------------------------------------------------
|                                          DescribeImages                                          |
+--------------------------+-----------------------------------------------------+-----------------------+
| 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。

10.4. RHEL コンピュートノードの準備

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、必要なリポジトリー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リスト表示します。

```
# 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 リポジトリーをすべて無効にします。

   a. 有効にされている RHSM リポジトリーをすべて無効にします。

   ```
   # subscription-manager repos --disable="*"
   ```

   b. 残りの yum リポジトリーをリスト表示し、**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 リポジトリーを無効にします。

   ```
   # yum-config-manager --disable <repo_id>
   ```

   または、すべてのリポジトリーを無効にします。

   ```
   # yum-config-manager --disable \*
   ```

   利用可能なリポジトリーが多い場合には、数分の時間がかかることがあります。

6. OpenShift Container Platform 4.16 で必要なリポジトリー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-4.16-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。

10.5. 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

- IAM ロールに必要な AWS パーミッション を参照してください。

10.6. 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 コンピュートマシンの場合、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ーグループに **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。

10.7. RHEL コンピュート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トマシンを、OpenShift Container Platform 4.16 クラスターにさらに追加できます。

前提条件

- OpenShift Container Platform クラスターに RHEL コンピュートノードがすでに含まれています。
- 最初の RHEL コンピュート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 ホストにアクセスできる必要があります。Bastion と SSH プロキシーまたは 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. コンピュート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。

10.8. 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、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

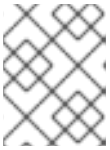
1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.29.4
master-1  Ready     master  63m  v1.29.4
master-2  Ready     master  64m  v1.29.4
```

出力には作成したすべてのマシンがリスト表示されます。



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。

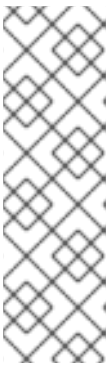
```
$ oc get csr
```

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。このリスト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 user-provisioned infrastructure 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。

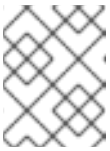
- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

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

1 **<csr_name>** は、現行の CSR のリストからの 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 のリストからの 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
```

6. 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  v1.29.4
master-1  Ready     master  73m  v1.29.4
master-2  Ready     master  74m  v1.29.4
worker-0  Ready     worker  11m  v1.29.4
worker-1  Ready     worker  11m  v1.29.4
```

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。

10.9. ANSIBLE ホストファイルの必須パラメーター

Red Hat Enterprise Linux (RHEL) コンピュート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
ansible_user	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 **root** です。
ansible_become	**ansible_user** の値が root ではない場合、**ansible_become** を **True** に設定する必要があり、**ansible_user** 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。	**True**。値が **True** 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。
openshift_kubeconfig_path	クラスターの **kubeconfig** 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11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

11.1. 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ト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。

11.1.1. コンピュートマシンの 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ト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トマシンの AWS への追加 を参照してください。

11.1.2. コンピュートマシンの Microsoft Azure への追加

Microsoft Azure 上の OpenShift Container Platform クラスターにコンピュートマシンをさらに追加するには、Azure で追加のワーカーマシンを作成する を参照してください。

11.1.3. コンピュートマシンの 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トマシンをさらに追加するには、Azure Stack Hub で追加のワーカーマシンを作成する を参照してください。

11.1.4. コンピュートマシンの 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トマシンをさらに追加するには、GCP で追加のワーカーマシンを作成する を参照してください。

11.1.5. コンピュートマシンの 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。

11.1.6. コンピュート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トマシンを追加するには、コンピュートマシンのベアメタルへの追加 を参照してください。

11.2. CLOUDFORMATION テンプレートの使用によるコンピュートマシンの AWS への追加

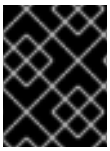
サンプルの CloudFormation テンプレートを使用して作成した Amazon Web Services (AWS) の OpenShift Container Platform クラスターにコンピュートマシンを追加することができます。

11.2.1. 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ト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、再作成する必要があります。

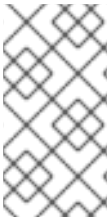
11.2.2. CloudFormation テンプレートの使用によるコンピュートマシンの AWS クラスターへの追加

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ト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 つのコンピュート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ト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ト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す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ト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. 別のコンピュート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> \ 1
    --template-body file://<template>.yaml \ 2
    --parameters file://<parameters>.json 3
```

1. **<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トマシンが十分な数に達するまでコンピュートスタックの作成を継続します。

11.2.3. 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、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

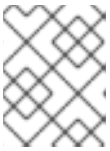
1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.29.4
master-1  Ready     master  63m  v1.29.4
master-2  Ready     master  64m  v1.29.4
```

出力には作成したすべてのマシンがリスト表示されます。



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。このリスト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 user-provisioned infrastructure 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。

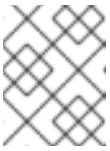
- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

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

1 **<csr_name>** は、現行の CSR のリストからの 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 のリストからの 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
     ```

6. 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  v1.29.4
   master-1  Ready     master  73m  v1.29.4
   master-2  Ready     master  74m  v1.29.4
   worker-0  Ready     worker  11m  v1.29.4
   worker-1  Ready     worker  11m  v1.29.4
   ```

注記

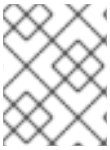
サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

11.3. コンピューティングマシンを VSPHERE に手動で追加する

コンピュートマシンを 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 コンピュートマシンの作成を自動化することもできます。

11.3.1. 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トマシンを 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。

11.3.2. vSphere でのコンピュートマシンのクラスターへの追加

コンピュートマシンを 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。

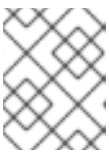
vSphere テンプレートを OpenShift Container Platform クラスターにデプロイした後に、そのクラスター内のマシンの仮想マシン (VM) をデプロイできます。

前提条件

- コンピュートマシンの 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. テンプレートの名前を右クリックし、**Clone → Clone to Virtual Machine**をクリックします。
2. **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。

3. **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。
4. **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。
5. **Select storage** タブで、設定ファイルとディスクファイル用のストレージを選択します。
6. **Select clone options** で、**Customize this virtual machine's hardware**を選択します。

7. **Customize hardware** タブで、**Advanced Parameters** をクリックします。
   - **Attribute** フィールドおよび **Values** フィールドにデータを指定して、以下の設定パラメーター名と値を追加します。作成するパラメーターごとに **Add** ボタンを選択してください。
     - **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。
8. **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。多くのネットワークが存在する場合は、**Add New Device** > **Network Adapter** を選択し、**New Network** メニュー項目に表示されるフィールドにネットワーク情報を入力します。
9. 残りの設定手順を完了します。**Finish** ボタンをクリックして、クローン作成操作を完了します。
10. **Virtual Machines** タブで仮想マシンを右クリックし、**Power → Power On** を選択します。

次のステップ

- 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。

11.3.3. 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、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

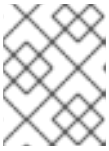
1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME     STATUS   ROLES   AGE  VERSION
master-0 Ready    master  63m  v1.29.4
master-1 Ready    master  63m  v1.29.4
master-2 Ready    master  64m  v1.29.4
```

出力には作成したすべてのマシンがリスト表示されます。



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

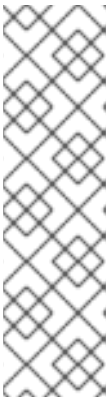
```
NAME        AGE     REQUESTOR                                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   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。このリスト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 user-provisioned infrastructure 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> 1
```

1 **<csr_name>** は、現行の CSR のリストからの 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 のリストからの 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
```

6. 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  v1.29.4
master-1  Ready     master  73m  v1.29.4
```

```
master-2  Ready    master  74m  v1.29.4
worker-0  Ready    worker  11m  v1.29.4
worker-1  Ready    worker  11m  v1.29.4
```

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。

11.4. コンピュート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トマシンを追加することができます。

11.4.1. 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トマシンの詳細を 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ト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トマシンを 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。

11.4.2. 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ト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。

11.4.2.1. ISO イメージを使用した 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) コンピュート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トマシンの 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。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. 次のコマンドを実行して、クラスターから Ignition 設定ファイルを抽出します。

```
$ oc extract -n openshift-machine-api secret/worker-user-data-managed --keys=userData --to=- > worker.ign
```

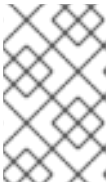
2. クラスターからエクスポートした **worker.ign** Ignition 設定ファイルを HTTP サーバーにアップロードします。これらのファイルの URL をメモします。
3. Ignition ファイルが URL で利用可能であることを検証できます。次の例では、コンピュートノードの Ignition 設定ファイルを取得します。

```
$ curl -k http://<HTTP_server>/worker.ign
```

4. 次のコマンドを実行すると、新しいマシンを起動するための ISO イメージにアクセスできます。

```
RHCOS_VHD_ORIGIN_URL=$(oc -n openshift-machine-config-operator get configmap/coreos-bootimages -o jsonpath='{.data.stream}' | jq -r '.architectures.<architecture>.artifacts.metal.formats.iso.disk.location')
```

5. ISO ファイルを使用して、追加のコンピュートマシンに 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。
6. 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。

7. **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。

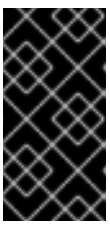
注記

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場合は、**coreos-installer** を実行する前に、内部認証局 (CA) をシステムのトラスト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-
a5a2d43879223273c9b60af66b44202a1d1248fc01cf156c46d4a79f552b6bad47bc8cc78ddf011
6e80c59d2ea9e32ba53bc807afbca581aa059311def2c3e3b
```

8. 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。

9. 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。

11.4.2.2. PXE または iPXE ブートによる 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) コンピュート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トマシンの 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。

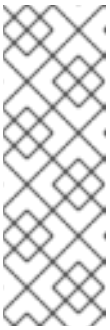
手順

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> (1)
    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 (2)
```

(1) 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。

(2) 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** パラメーターはライブ **initramfs** ファイルの場所であり、**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** パラメーター値はライブ **rootfs** ファイルの場所になります。**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。



注記

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 (**x86_**64 + **aarch64**):

```
kernel http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> initrd=main
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
<architecture>.img coreos.inst.install_dev=/dev/sda
```

```
coreos.inst.ignition_url=http://<HTTP_server>/worker.ign 1 2
initrd --name main http://<HTTP_server>/rhcos-<version>-live-initramfs.
<architecture>.img 3
boot
```

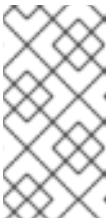
1 HTTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**kernel** パラメーター値は **kernel** ファイルの場所であり、**initrd=main** 引数は UEFI システムでの起動に必要であり、**coreos.live.rootfs_url** パラメーター値はワーカー Ignition 設定ファイルの場所であり、**coreos.inst.ignition_url** パラメーター値は **rootfs** のライブファイルの場所です。

2 複数の NIC を使用する場合、**ip** オプションに単一インターフェイスを指定します。たとえば、**eno1** という名前の NIC で DHCP を使用するには、**ip=eno1:dhcp** を設定します。

3 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。

注記

この設定では、グラフィカルコンソールを備えたマシンでのシリアルコンソールアクセスは有効になりません。別のコンソールを設定するには、**kernel** 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? と、「高度な RHCOS インストール設定」セクションの「PXE および ISO インストール用シリアルコンソールの有効化」を参照してください。



注記

aarch64 アーキテクチャーで CoreOS **kernel** をネットワークブートするには、**IMAGE_GZIP** オプションが有効になっているバージョンの iPXE ビルドを使用する必要があります。iPXE の **IMAGE_GZIP** オプション を参照してください。

- **aarch64** 上の PXE (第 2 段階として UEFI および GRUB を使用) の場合:

```
menuentry 'Install CoreOS' {
    linux rhcos-<version>-live-kernel-<architecture>
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/worker.ign 1 2
    initrd rhcos-<version>-live-initramfs.<architecture>.img 3
}
```

1 HTTP/TFTP 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、TFTP サーバー上の **kernel** ファイルの場所になります。**coreos.live.rootfs_url** パラメーター値は **rootfs** ファイルの場所であり、**coreos.inst.ignition_url** パラメーター値は HTTP サーバー上のブートストラップ Ignition 設定ファイルの場所になります。

2

複数の NIC を使用する場合、**ip** オプションに単一インターフェイスを指定します。たとえば、**eno1** という名前の NIC で DHCP を使用するには、 **ip=eno1:dhcp** を設定し

3 TFTP 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。

2. 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。

11.4.3. 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、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

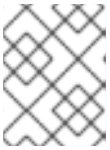
1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME     STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.29.4
master-1  Ready     master  63m  v1.29.4
master-2  Ready     master  64m  v1.29.4
```

出力には作成したすべてのマシンがリスト表示されます。



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
```

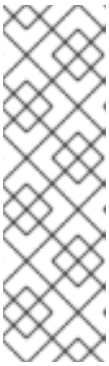
出力例

```
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。このリスト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 user-provisioned infrastructure 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> 1
```

1 **<csr_name>** は、現行の CSR のリストからの 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 のリストからの 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
     ```

6. 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  v1.29.4
   master-1  Ready     master  73m  v1.29.4
   master-2  Ready     master  74m  v1.29.4
   worker-0  Ready     worker  11m  v1.29.4
   worker-1  Ready     worker  11m  v1.29.4
   ```

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12章 コントロールプレーンマシンの管理

12.1. コントロールプレーンマシンセットについて

コントロールプレーンマシンセットを使用すると、OpenShift Container Platform クラスター内のコントロールプレーンマシンリソースの管理を自動化できます。

重要

コントロールプレーンマシンセットはコンピューティングマシンを管理できず、コンピューティングマシンセットはコントロールプレーンマシンを管理できません。

コントロールプレーンマシンセットは、コンピューティングマシンセットがコンピューティングマシンに提供するのと同様の管理機能をコントロールプレーンマシンに提供します。ただし、これら 2 種類のマシンセットは、Machine API 内で定義された別々のカスタムリソースであり、アーキテクチャーと機能にいくつかの基本的な違いがあります。

12.1.1. Control Plane Machine Set Operator の概要

Control Plane Machine Set Operator は、**ControlPlaneMachineSet** カスタムリソース (CR) を使用して、OpenShift Container Platform クラスター内のコントロールプレーンマシンリソースの管理を自動化します。

クラスターコントロールプレーンマシンセットの状態が **Active** に設定されている場合、Operator は、指定された設定を持つ正しい数のコントロールプレーンマシンがクラスターにあることを確認します。これにより、劣化したコントロールプレーンマシンの自動交換と、コントロールプレーンへの変更のロールアウトが可能になります。

クラスターにあるコントロールプレーンマシンセットは 1 つだけであり、Operator は **openshift-machine-api** namespace のオブジェクトのみを管理します。

12.1.1.1. Control Plane Machine Set Operator の制限事項

Control Plane Machine Set Operator には、次の制限があります。

- Amazon Web Services (AWS)、Google Cloud Platform (GCP)、IBM Power® Virtual Server、Microsoft Azure、Nutanix、VMware vSphere、および Red Hat OpenStack Platform (RHOSP) クラスターのみがサポートされています。

- コントロールプレーンノードを表す既存のマシンを持たないクラスターは、コントロールプレーンマシンセットを使用することも、インストール後にコントロールプレーンマシンセットの使用を有効にすることもできません。通常、既存のコントロールプレーンマシンは、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てクラスターがインストールされた場合にのみ存在します。
  クラスターに必要な既存のコントロールプレーンマシンがあるかどうかを確認するには、管理者権限を持つユーザーとして次のコマンドを実行します。

  ```
  $ oc get machine \
    -n openshift-machine-api \
    -l machine.openshift.io/cluster-api-machine-role=master
  ```

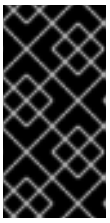
  既存のコントロールプレーンマシンを示す出力例

```
NAME                 PHASE     TYPE        REGION      ZONE         AGE
<infrastructure_id>-master-0   Running   m6i.xlarge   us-west-1   us-west-1a   5h19m
<infrastructure_id>-master-1   Running   m6i.xlarge   us-west-1   us-west-1b   5h19m
<infrastructure_id>-master-2   Running   m6i.xlarge   us-west-1   us-west-1a   5h19m
```

既存のコントロールプレーンマシンが欠落している出力例

```
No resources found in openshift-machine-api namespace.
```

- Operator は Machine API Operator が動作している必要があるため、手動でプロビジョニングされたマシンを含むクラスターではサポートされていません。アクティブに生成された **ControlPlaneMachineSet** カスタムリソース (CR) を作成するプラットフォーム用に手動でプロビジョニングされたマシンを使用して、OpenShift Container Platform クラスターをインストールする場合は、インストールプロセスの指示に従って、コントロールプレーンマシンセットを定義する Kubernetes マニフェストファイルを削除する必要があります。
- 3 つのコントロールプレーンマシンを持つクラスターのみがサポートされます。
- コントロールプレーンの水平スケーリングはサポートされていません。
- Azure コントロールプレーンマシンをエフェメラル OS ディスクにデプロイすると、データ損失のリスクが高まるため、サポートされていません。
- コントロールプレーンマシンを AWS スポットインスタンス、GCP プリエンプティブル VM、または Azure スポット VM としてデプロイすることはサポートされていません。



重要

コントロールプレーンマシンを AWS スポットインスタンス、GCP プリエンプティブル VM、または Azure スポット VM としてデプロイしようとすると、クラスターが etcd クォーラムを失う可能性があります。すべてのコントロールプレーンマシンが同時に失われたクラスターは回復できません。

- インストール中またはインストール前にコントロールプレーンマシンセットを変更することはサポートされていません。インストール後にのみ、コントロールプレーンマシンセットに変更を加える必要があります。

12.1.2. 関連情報

- Control Plane Machine Set Operator リファレンス
- **ControlPlaneMachineSet** カスタムリソース

12.2. コントロールプレーンマシンセットの概要

コントロールプレーンマシンセットを使い始めるプロセスは、クラスター内の **ControlPlaneMachineSet** カスタムリソース(CR)の状態によって異なります。

アクティブに生成された CR を持つクラスター

アクティブな状態で生成された CR を持つクラスターは、デフォルトで設定されたコントロールプレーンマシンを使用します。管理者の操作は必要ありません。

非アクティブな CR が生成されたクラスター

生成された非アクティブな CR を含むクラスターの場合、CR 設定を確認して CR を アクティブ化する 必要があります。

CR が生成されていないクラスター

生成された CR が含まれていないクラスターの場合、クラスターに適した設定で CR を作成してアクティブ化する 必要があります。

クラスター内の **ControlPlaneMachineSet** CR の状態が不明な場合は、CR の状態を確認 できます。

12.2.1. サポートされているクラウドプロバイダー

OpenShift Container Platform 4.16 では、Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、Nutanix、および VMware vSphere クラスターのコントロールプレーンマシンセットがサポートされています。

インストール後のコントロールプレーンマシンセットのステータスは、クラウドプロバイダーと、クラスターにインストールした OpenShift Container Platform のバージョンによって異なります。

表12.1 OpenShift Container Platform 4.16 のコントロールプレーンマシンセットの実装

クラウドプロバイダー	デフォルトでアクティブ	生成された CR	手動の CR が必要
Amazon Web Services (AWS)	X [1]	X	
Google Cloud Platform (GCP)	X [2]	X	
Microsoft Azure	X [2]	X	
Nutanix	X [3]	X	
Red Hat OpenStack Platform (RHOSP)	X [3]	X	
VMware vSphere	X [4]	X	

1. バージョン 4.11 以前からアップグレードされた AWS クラスターには、CR アクティベーション が必要です。
2. バージョン 4.12 以前からアップグレードされた GCP および Azure クラスターでは、CR アクティベーション が必要です。
3. バージョン 4.13 以前からアップグレードされた Nutanix クラスターおよび RHOSP クラスターには CR アクティベーション が必要です。
4. バージョン 4.15 以前からアップグレードされた vSphere クラスターには、CR アクティベーション が必要です。

12.2.2. コントロールプレーンマシンセットのカスタムリソースの状態を確認する

ControlPlaneMachineSet カスタムリソース (CR) の存在と状態を確認できます。

手順

- 次のコマンドを実行して、CR の状態を確認します。

```
$ oc get controlplanemachineset.machine.openshift.io cluster \
  --namespace openshift-machine-api
```

  - **Active** の結果は、**ControlPlaneMachineSet** CR が存在し、アクティブ化されていることを示します。管理者の操作は必要ありません。
  - **Inactive** の結果は、**ControlPlaneMachineSet** CR が存在するがアクティブ化されていないことを示します。
  - **NotFound** の結果は、既存の **ControlPlaneMachineSet** CR がないことを示します。

次のステップ

コントロールプレーンマシンセットを使用するには、クラスターの正しい設定を持つ **ControlPlaneMachineSet** CR が存在することを確認する必要があります。

- クラスターに既存の CR がある場合は、CR の設定がクラスターに対して正しいことを確認する必要があります。
- クラスターに既存の CR がない場合は、クラスターの正しい設定で CR を作成する必要があります。

12.2.3. コントロールプレーンマシンセットカスタムリソースの有効化

コントロールプレーンマシンセットを使用するには、クラスターの正しい設定を持つ **ControlPlaneMachineSet** カスタムリソース (CR) が存在することを確認する必要があります。CR が生成されたクラスターでは、CR の設定がクラスターに対して正しいことを確認し、アクティブ化する必要があります。

注記

CR のパラメーターの詳細については、コントロールプレーンマシンセットの設定を参照してください。

手順

1. 次のコマンドを実行して、CR の設定を表示します。

```
$ oc --namespace openshift-machine-api edit controlplanemachineset.machine.openshift.io
cluster
```

2. クラスター設定に不適切なフィールドの値を変更します。
3. 設定が正しい場合は、**.spec.state** フィールドを **Active** に設定し、変更を保存して CR をアクティブにします。

重要

CR を有効にするには、CR 設定の更新に使用するのと同じ **oc edit** セッションで **.spec.state** フィールドを **Active** に変更する必要があります。CR が **Inactive** のままの状態で保存された場合、コントロールプレーンマシンセットジェネレーターは CR を元の設定にリセットします。

関連情報

- コントロールプレーンマシンセットの設定

12.2.4. コントロールプレーンマシンセットのカスタムリソースの作成

コントロールプレーンマシンセットを使用するには、クラスターの正しい設定を持つ **ControlPlaneMachineSet** カスタムリソース (CR) が存在することを確認する必要があります。CR が生成されていないクラスターでは、CR を手動で作成してアクティブ化する必要があります。

注記

CR の構造とパラメーターの詳細については、コントロールプレーンマシンセットの設定を参照してください。

手順

1. 次のテンプレートを使用して YAML ファイルを作成します。

コントロールプレーンマシンセットの CR YAML ファイルテンプレート

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
  name: cluster
  namespace: openshift-machine-api
spec:
  replicas: 3
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <cluster_id> 1
      machine.openshift.io/cluster-api-machine-role: master
      machine.openshift.io/cluster-api-machine-type: master
  state: Active 2
  strategy:
    type: RollingUpdate 3
  template:
    machineType: machines_v1beta1_machine_openshift_io
    machines_v1beta1_machine_openshift_io:
      failureDomains:
        platform: <platform> 4
        <platform_failure_domains> 5
      metadata:
        labels:
          machine.openshift.io/cluster-api-cluster: <cluster_id> 6
          machine.openshift.io/cluster-api-machine-role: master
          machine.openshift.io/cluster-api-machine-type: master
```

```
    spec:
      providerSpec:
        value:
          <platform_provider_spec> 7
```

1 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。**ControlPlaneMachineSet** CR を作成するときに、この値を指定する必要があります。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 Operator の状態を指定します。状態が **Inactive** の場合、Operator は操作できません。値を **Active** に設定することで、Operator をアクティブ化できます。

重要

CR をアクティブ化する前に、その設定がクラスター要件に対して正しいことを確認する必要があります。

3 クラスターの更新戦略を指定します。有効な値は **OnDelete** および **RollingUpdate** です。デフォルト値は **RollingUpdate** です。更新戦略の詳細については、コントロールプレーン設定の更新を参照してください。

4 クラウドプロバイダーのプラットフォーム名を指定します。有効な値は、**AWS**、**Azure**、**GCP**、**Nutanix**、**VSphere**、および **OpenStack** です。

5 クラスターの **<platform_failure_domains>** 設定を追加します。このセクションのフォーマットと値はプロバイダー固有です。詳細については、クラウドプロバイダーの障害ドメイン設定サンプルを参照してください。

6 インフラストラクチャー ID を指定します。

7 クラスターの **<platform_provider_spec>** 設定を追加します。このセクションのフォーマットと値はプロバイダー固有です。詳細については、クラウドプロバイダーのサンプルプロバイダー仕様を参照してください。

2. コントロールプレーンマシンセット CR のサンプル YAML を参照し、クラスター設定に適した値をファイルに入力します。
3. クラウドプロバイダーのサンプル障害ドメイン設定とサンプルプロバイダー仕様を参照し、ファイルのこれらのセクションを適切な値で更新します。
4. 設定が正しい場合は、**.spec.state** フィールドを **Active** に設定し、変更を保存して CR をアクティブにします。
5. 次のコマンドを実行して、YAML ファイルから CR を作成します。

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

<control_plane_machine_set> は、CR 設定を含む YAML ファイルの名前です。

関連情報

- コントロールプレーン設定の更新
- コントロールプレーンマシンセットの設定
- プロバイダー固有の設定オプション

12.3. コントロールプレーンマシンセットを使用したコントロールプレーンマシンの管理

コントロールプレーンマシンセットは、コントロールプレーン管理のいくつかの重要な側面を自動化します。

12.3.1. コントロールプレーン設定の更新

コントロールプレーンマシンセットのカスタムリソース (CR) の仕様を更新することで、コントロールプレーンのマシンの設定を変更できます。

Control Plane Machine Set Operator は、コントロールプレーンマシンを監視し、それらの設定をコントロールプレーンマシンセット CR の仕様と比較します。CR の仕様とコントロールプレーンマシンの設定との間に不一致がある場合、オペレータはそのコントロールプレーンマシンに交換用のマークを付けます。

注記

CR のパラメーターの詳細については、コントロールプレーンマシンセットの設定を参照してください。

前提条件

- クラスターには、アクティブ化され機能している Control Plane Machine Set Operator があります。

手順

1. 次のコマンドを実行して、コントロールプレーンマシンセットの CR を編集します。

```
$ oc edit controlplanemachineset.machine.openshift.io cluster \
  -n openshift-machine-api
```

2. クラスター設定で更新するフィールドの値を変更します。
3. 変更を保存します。

次のステップ

- デフォルトの **RollingUpdate** 更新ストラテジーを使用するクラスターの場合、コントロールプレーンマシンセットは、変更をコントロールプレーン設定に自動的に伝達します。
- **OnDelete** 更新戦略を使用するように設定されているクラスターの場合、コントロールプレーンマシンを手動で置き換える必要があります。

12.3.1.1. コントロールプレーン設定の自動更新

RollingUpdate 更新戦略により、変更がコントロールプレーン設定に自動的に反映されます。この更新戦略は、コントロールプレーンマシンセットのデフォルト設定です。

RollingUpdate 更新戦略を使用するクラスターの場合、Operator は、CR で指定された設定を使用して、代替のコントロールプレーンマシンを作成します。交換用のコントロールプレーンマシンの準備ができたら、Operator は、交換用にマークされたコントロールプレーンマシンを削除します。次に、交換用のマシンがコントロールプレーンに参加します。

複数のコントロールプレーンマシンが交換対象としてマークされている場合、Operator は、各マシンを交換するまでこの交換プロセスを一度に 1 台のマシンずつ繰り返すことで、交換中の etcd の健全性を保護します。

12.3.1.2. コントロールプレーン設定の手動更新

OnDelete 更新ストラテジーを使用して、マシンを手動で交換することで、コントロールプレーン設定に変更を反映できます。マシンを手動で置き換えると、変更をより広範囲に適用する前に、単一のマシンで設定への変更をテストできます。

OnDelete 更新ストラテジーを使用するように設定されているクラスターの場合、既存のマシンを削除すると、Operator は代替のコントロールプレーンマシンを作成します。代替のコントロールプレーンマシンの準備ができたら、etcd Operator を使用して既存のマシンを削除できます。次に、交換用のマシンがコントロールプレーンに参加します。

複数のコントロールプレーンマシンが削除された場合、Operator は必要なすべての代替マシンを同時に作成します。Operator は、複数のマシンがコントロールプレーンから同時に削除されるのを防ぐことで etcd の健全性を維持します。

12.3.2. コントロールプレーンマシンの置き換え

コントロールプレーンマシンが設定されているクラスター内のコントロールプレーンマシンを置き換えるには、マシンを手動で削除します。コントロールプレーンマシンセットは、コントロールプレーンマシンセットのカスタムリソース (CR) の仕様を使用するマシンに削除されたマシンを置き換えます。

前提条件

- クラスターが Red Hat OpenStack Platform (RHOSP) 上で実行されており、アップグレードなどのためにコンピューティングサーバーを退避する必要がある場合は、次のコマンドを実行して、マシンが実行している RHOSP コンピューティングノードを無効にする必要があります。

  ```
  $ openstack compute service set <target_node_host_name> nova-compute --disable
  ```

  詳細は、RHOSP ドキュメントの 移行の準備 を参照してください。

手順

1. 次のコマンドを実行して、クラスター内のコントロールプレーンマシンを一覧表示します。

   ```
   $ oc get machines \
     -l machine.openshift.io/cluster-api-machine-role==master \
     -n openshift-machine-api
   ```

2. 次のコマンドを実行して、コントロールプレーンマシンを削除します。

```
$ oc delete machine \
  -n openshift-machine-api \
  <control_plane_machine_name> 1
```

1 削除するコントロールプレーンマシンの名前を指定します。

注記

複数のコントロールプレーンマシンを削除すると、設定された更新戦略に従ってコントロールプレーンマシンセットがそれらを置き換えます。

- デフォルトの **RollingUpdate** 更新戦略を使用するクラスターの場合、Operator は、各マシンが交換されるまで、一度に 1 台のマシンを交換します。
- **OnDelete** 更新戦略を使用するように設定されたクラスターの場合、Operator は必要なすべての代替マシンを同時に作成します。

どちらの戦略も、コントロールプレーンマシンの交換中に etcd の健全性を維持します。

12.3.3. 関連情報

- コントロールプレーンマシンセットの設定
- プロバイダー固有の設定オプション

12.4. コントロールプレーンマシンセットの設定

この例の YAML スニペットは、コントロールプレーンマシンセットのカスタムリソース (CR) の基本構造を示しています。

12.4.1. コントロールプレーンマシンセットのカスタムリソースのサンプル YAML

ControlPlaneMachineSet CR のベースは、すべてのプラットフォームで同じように構築されています。

サンプル ControlPlaneMachineSet CR YAML ファイル

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
  name: cluster 1
  namespace: openshift-machine-api
spec:
  replicas: 3 2
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <cluster_id> 3
      machine.openshift.io/cluster-api-machine-role: master
      machine.openshift.io/cluster-api-machine-type: master
  state: Active 4
```

```
strategy:
  type: RollingUpdate 5
template:
  machineType: machines_v1beta1_machine_openshift_io
  machines_v1beta1_machine_openshift_io:
    failureDomains:
      platform: <platform> 6
      <platform_failure_domains> 7
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <cluster_id>
        machine.openshift.io/cluster-api-machine-role: master
        machine.openshift.io/cluster-api-machine-type: master
    spec:
      providerSpec:
        value:
          <platform_provider_spec> 8
```

1 **cluster** である **ControlPlaneMachineSet** CR の名前を指定します。この値は変更しないでください。

2 コントロールプレーンマシンの数を指定します。3 つのコントロールプレーンマシンを持つクラスターのみがサポートされているため、**replicas** の値は **3** です。水平スケーリングはサポートされていません。この値は変更しないでください。

3 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。**ControlPlaneMachineSet** CR を作成するときに、この値を指定する必要があります。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
```

4 Operator の状態を指定します。状態が **Inactive** の場合、Operator は操作できません。値を **Active** に設定することで、Operator をアクティブ化できます。

重要

Operator をアクティブ化する前に、**ControlPlaneMachineSet** CR 設定がクラスター要件に対して正しいことを確認する必要があります。Control Plane Machine Set Operator のアクティブ化の詳細は、「コントロールプレーンマシンセットの概要」を参照してください。

5 クラスターの更新戦略を指定します。許可される値は **OnDelete** と **RollingUpdate** です。デフォルト値は **RollingUpdate** です。更新戦略の詳細については、コントロールプレーン設定の更新を参照してください。

6 クラウドプロバイダーのプラットフォーム名を指定します。この値は変更しないでください。

7 クラスターの **<platform_failure_domains>** 設定を指定します。このセクションのフォーマットと値はプロバイダー固有です。詳細については、クラウドプロバイダーの障害ドメイン設定サンプルを参照してください。

8

クラスターの **<platform_provider_spec>** 設定を指定します。このセクションのフォーマットと値はプロバイダー固有です。詳細については、クラウドプロバイダーのサンプルプロバイダー仕様

関連情報

- コントロールプレーンマシンセットの概要
- コントロールプレーン設定の更新

12.4.2. プロバイダー固有の設定オプション

コントロールプレーンマシンセットマニフェストの **<platform_provider_spec>** および **<platform_failure_domains>** セクションは、プロバイダー固有です。クラスターのプロバイダー固有の設定オプションについては、次のリソースを参照してください。

- Amazon Web Services のコントロールプレーン設定オプション
- Google Cloud Platform のコントロールプレーン設定オプション
- Microsoft Azure のコントロールプレーン設定オプション
- Nutanix のコントロールプレーン設定オプション
- Red Hat OpenStack Platform (RHOSP) のコントロールプレーン設定オプション
- VMware vSphere のコントロールプレーン設定オプション

12.5. コントロールプレーンマシンの設定オプション

12.5.1. Amazon Web Services のコントロールプレーン設定オプション

コントロールプレーンマシンセットの値を更新することで、Amazon Web Services (AWS) コントロールプレーンマシンの設定を変更し、機能を有効にすることができます。コントロールプレーンマシンセットへの更新を保存すると、設定された 更新ストラテジー に従って Control Plane Machine Set Operator がコントロールプレーンマシンを更新します。

12.5.1.1. Amazon Web Services クラスターを設定するサンプル YAML

次の YAML スニペットの例は、AWS クラスターのプロバイダーの仕様と障害ドメインの設定を示しています。

12.5.1.1.1. サンプル AWS プロバイダー仕様

既存のクラスター用にコントロールプレーンマシンセットを作成する場合、プロバイダーの仕様は、インストールプログラムによって作成されるコントロールプレーンマシンのカスタムリソース (CR) の **providerSpec** 設定と一致する必要があります。CR の障害ドメインセクションに設定されているフィールドは省略できます。

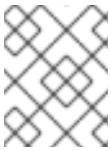
次の例で、**<cluster_id>** は、クラスターをプロビジョニングしたときに設定したクラスター ID に基づくインフラストラクチャー ID です。OpenShift CLI がインストールされている場合は、以下のコマンドを実行してインフラストラクチャー ID を取得できます。

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

サンプル AWS providerSpec 値

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
  name: cluster
  namespace: openshift-machine-api
spec:
# ...
  template:
# ...
      spec:
        providerSpec:
          value:
            ami:
              id: ami-<ami_id_string> (1)
            apiVersion: machine.openshift.io/v1beta1
            blockDevices:
            - ebs: (2)
                encrypted: true
                iops: 0
                kmsKey:
                  arn: ""
                volumeSize: 120
                volumeType: gp3
            credentialsSecret:
              name: aws-cloud-credentials (3)
            deviceIndex: 0
            iamInstanceProfile:
              id: <cluster_id>-master-profile (4)
            instanceType: m6i.xlarge (5)
            kind: AWSMachineProviderConfig (6)
            loadBalancers: (7)
            - name: <cluster_id>-int
              type: network
            - name: <cluster_id>-ext
              type: network
            metadata:
              creationTimestamp: null
            metadataServiceOptions: {}
            placement: (8)
              region: <region> (9)
              availabilityZone: "" (10)
              tenancy: (11)
            securityGroups:
            - filters:
              - name: tag:Name
                values:
                - <cluster_id>-master-sg (12)
            subnet: {} (13)
            userDataSecret:
              name: master-user-data (14)
```

1. クラスターの Red Hat Enterprise Linux CoreOS (RHCOS) Amazon Machine Image (AMI) ID を指定します。AMI はクラスターと同じリージョンに属する必要があります。AWS Marketplace イメー
2. 暗号化された EBS ボリュームの設定を指定します。
3. クラスターのシークレット名を指定します。この値は変更しないでください。
4. AWS Identity and Access Management (IAM) インスタンスプロファイルを指定します。この値は変更しないでください。
5. コントロールプレーンの AWS インスタンスタイプを指定します。
6. クラウドプロバイダープラットフォームのタイプを指定します。この値は変更しないでください。
7. クラスターの内部 (**int**) および外部 (**ext**) ロードバランサーを指定します。



注記

プライベート OpenShift Container Platform クラスターでは、外部 (**ext**) ロードバランサーパラメーターを省略できます。

8. AWS でコントロールプレーンインスタンスを作成する場所を指定します。
9. クラスターの AWS リージョンを指定します。
10. このパラメーターは障害ドメインで設定されます。ここでは空の値が表示されています。このパラメーターに指定された値が障害ドメインの値と異なる場合、Control Plane Machine Set Operator がその値を障害ドメインの値で上書きします。
11. コントロールプレーンの AWS Dedicated Instance 設定を指定します。詳細は、Dedicated Instance に関する AWS ドキュメントを参照してください。次の値が有効です。
    - **default**: Dedicated Instance は共有ハードウェア上で実行されます。
    - **dedicated**: Dedicated Instance はシングルテナントのハードウェア上で実行されます。
    - **host**: Dedicated Instance は Dedicated Host (設定を制御できる分離されたサーバー) 上で実行されます。
12. コントロールプレーンマシンのセキュリティーグループを指定します。
13. このパラメーターは障害ドメインで設定されます。ここでは空の値が表示されています。このパラメーターに指定された値が障害ドメインの値と異なる場合、Control Plane Machine Set Operator がその値を障害ドメインの値で上書きします。

注記

障害ドメイン設定で値が指定されていない場合、プロバイダー仕様の値が使用されます。障害ドメインでサブネットを設定すると、プロバイダー仕様のサブネット値が上書きされます。

14. コントロールプレーンのユーザーデータシークレットを指定します。この値は変更しないでください。

12.5.1.1.2. サンプル AWS 障害ドメインの設定

障害ドメインのコントロールプレーンマシンセットの概念は、既存の AWS の アベイラビリティゾーン (AZ) の概念に似ています。**ControlPlaneMachineSet** CR は、可能な場合、コントロールプレーンマシンを複数の障害ドメインに分散します。

コントロールプレーンマシンセットで AWS 障害ドメインを設定するときは、使用するアベイラビリティゾーン名とサブネットを指定する必要があります。

サンプル AWS 障害ドメインの値

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
  name: cluster
  namespace: openshift-machine-api
spec:
# ...
  template:
# ...
    machines_v1beta1_machine_openshift_io:
      failureDomains:
        aws:
        - placement:
            availabilityZone: <aws_zone_a> 1
          subnet: 2
            filters:
            - name: tag:Name
              values:
              - <cluster_id>-private-<aws_zone_a> 3
            type: Filters 4
        - placement:
            availabilityZone: <aws_zone_b> 5
          subnet:
            filters:
            - name: tag:Name
              values:
              - <cluster_id>-private-<aws_zone_b> 6
            type: Filters
        platform: AWS 7
# ...
```

1. 最初の障害ドメインの AWS アベイラビリティゾーンを指定します。
2. サブネット設定を指定します。この例では、サブネットタイプが **Filters** であるため、**filters** スタンザがあります。
3. インフラストラクチャー ID と AWS アベイラビリティゾーンを使用して、最初の障害ドメインのサブネット名を指定します。
4. サブネットタイプを指定します。許可される値は、**ARN**、**Filters**、および **ID** です。デフォルト値は **Filters** です。
5. インフラストラクチャー ID と AWS アベイラビリティゾーンを使用して、追加の障害ドメインのサブネット名を指定します。

6 クラスターのインフラストラクチャー ID と、追加の障害ドメインの AWS アベイラビリティゾーンを指定します。

7 クラウドプロバイダーのプラットフォーム名を指定します。この値は変更しないでください。

12.5.1.2. コントロールプレーンマシンの Amazon Web Services 機能を有効にする

コントロールプレーンマシンセットの値を更新することで、機能を有効にします。

12.5.1.2.1. API サーバーをプライベートに制限する

クラスターを Amazon Web Services (AWS) にデプロイした後に、プライベートゾーンのみを使用するように API サーバーを再設定することができます。

前提条件

- OpenShift CLI (**oc**) がインストールされている。
- **admin** 権限を持つユーザーとして Web コンソールにアクセスできること。

手順

1. クラウドプロバイダーの Web ポータルまたはコンソールで、次の操作を行います。
   a. 適切なロードバランサーコンポーネントを見つけて削除します。
      - AWS の場合は、外部ロードバランサーを削除します。プライベートゾーンの API DNS エントリーは、同一の設定を使用する内部ロードバランサーをすでに参照するため、内部ロードバランサーを変更する必要はありません。
   b. パブリックゾーンの **api.$clustername.$yourdomain** DNS エントリーを削除します。
2. コントロールプレーンマシンセットのカスタムリソースで次の行を削除して、外部ロードバランサーを削除します。

```
# ...
providerSpec:
  value:
# ...
    loadBalancers:
    - name: lk4pj-ext 1
      type: network 2
    - name: lk4pj-int
      type: network
# ...
```

1 **-ext** で終わる外部ロードバランサーの **name** 値を削除します。

2 外部ロードバランサーの **type** 値を削除します。

関連情報

- Ingress Controller エンドポイント公開スコープの内部への設定

12.5.1.2.2. コントロールプレーンマシンセットを使用して Amazon Web Services インスタンスタイプを変更する

コントロールプレーンマシンセットのカスタムリソース (CR) の仕様を更新することで、コントロールプレーンマシンが使用する Amazon Web Services (AWS) インスタンスタイプを変更できます。

前提条件

- AWS クラスターは、コントロールプレーンマシンセットを使用します。

手順

1. **providerSpec** フィールドの下で以下の行を編集します。

```
providerSpec:
  value:
    ...
    instanceType: <compatible_aws_instance_type> (1)
```

(1) 前の選択と同じベースで、より大きな AWS インスタンスタイプを指定します。たとえば、**m6i.xlarge** を **m6i.2xlarge** または **m6i.4xlarge** に変更できます。

2. 変更を保存します。

12.5.1.2.3. マシンセットを使用した Elastic Fabric Adapter インスタンスの配置グループへのマシンの割り当て

既存の AWS 配置グループ内の Elastic Fabric Adaptor (EFA) インスタンスにマシンをデプロイするようにマシンセットを設定できます。

EFA インスタンスには配置グループは必要なく、EFA の設定以外の目的にも配置グループを使用できます。この例では、両方を使用して、指定された配置グループ内のマシンのネットワークパフォーマンスを向上できる設定を示します。

前提条件

- AWS コンソールで配置グループを作成しました。

注記

作成する配置グループのタイプの ルールと制限が、意図した使用例と互換性があることを確認してください。可能な場合、コントロールプレーンマシンセットは、コントロールプレーンマシンを複数の障害ドメインに分散します。コントロールプレーンに配置グループを使用するには、複数のアベイラビリティゾーンにまたがることができる配置グループタイプを使用する必要があります。

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. **providerSpec** フィールドの下に次の行を編集します。

```
apiVersion: machine.openshift.io/v1
```

```
kind: ControlPlaneMachineSet
# ...
spec:
  template:
    spec:
      providerSpec:
        value:
          instanceType: <supported_instance_type> 1
          networkInterfaceType: EFA 2
          placement:
            availabilityZone: <zone> 3
            region: <region> 4
          placementGroupName: <placement_group> 5
# ...
```

1. EFA をサポートする インスタンスタイプを指定します。
2. **EFA** ネットワークインターフェイスのタイプを指定します。
3. ゾーン (例: **us-east-1a**) を指定します。
4. リージョン (例: **us-east-1**) を指定します。
5. マシンをデプロイする既存の AWS 配置グループの名前を指定します。

検証

- AWS コンソールで、マシンセットが作成したマシンを見つけて、マシンのプロパティーで次のことを確認します。
  - 配置グループフィールドには、マシンセットの **placeGroupName** パラメーターに指定した値が含まれます。
  - インターフェイスタイプフィールドは、EFA を使用することを示します。

12.5.1.2.4. 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マシンを作成できます。マシンセットは、IMDSv1 と IMDSv2 の両方を使用できるマシン、または IMDSv2 の使用を必要とする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。

重要

IMDSv2 を必要とする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。

12.5.1.2.4.1. マシンセットを使用した IMDS の設定

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 1
```

1 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。

12.5.1.2.5. 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つ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。

パブリックテナンシーまたは専用テナンシーのいずれかを持つインスタンスが、Machine 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。

12.5.1.2.5.1. 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

Machine 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
```

12.5.2. Microsoft Azure のコントロールプレーン設定オプション

コントロールプレーンマシンセットの値を更新することで、Microsoft Azure コントロールプレーンマ

シンの設定を変更し、機能を有効にすることができます。コントロールプレーンマシンセットへの更新を保存すると、設定された 更新ストラテジー に従って Control Plane Machine Set Operator がコントロールプレーンマシンを更新します。

12.5.2.1. Microsoft Azure クラスターを設定するためのサンプル YAML

次の YAML スニペットの例は、Azure クラスターのプロバイダーの仕様と障害ドメインの設定を示しています。

12.5.2.1.1. Azure プロバイダー仕様のサンプル

既存クラスター用のコントロールプレーンマシンセットを作成する場合、プロバイダーの仕様は、インストールプログラムによって作成されるコントロールプレーン **machine** CR の **providerSpec** 設定と一致する必要があります。CR の障害ドメインセクションに設定されているフィールドは省略できます。

次の例で、**<cluster_id>** は、クラスターをプロビジョニングしたとき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。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
```

Azure providerSpec 値のサンプル

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
  name: cluster
  namespace: openshift-machine-api
spec:
# ...
  template:
# ...
      spec:
        providerSpec:
          value:
            acceleratedNetworking: true
            apiVersion: machine.openshift.io/v1beta1
            credentialsSecret:
              name: azure-cloud-credentials 1
              namespace: openshift-machine-api
            diagnostics: {}
            image: 2
              offer: ""
              publisher: ""
              resourceID: /resourceGroups/<cluster_id>-
rg/providers/Microsoft.Compute/galleries/gallery_<cluster_id>/images/<cluster_id>-
gen2/versions/412.86.20220930 3
              sku: ""
              version: ""
            internalLoadBalancer: <cluster_id>-internal 4
            kind: AzureMachineProviderSpec 5
            location: <region> 6
            managedIdentity: <cluster_id>-identity
            metadata:
```

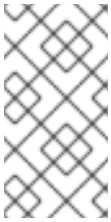
```
    creationTimestamp: null
    name: <cluster_id>
  networkResourceGroup: <cluster_id>-rg
  osDisk: (7)
    diskSettings: {}
    diskSizeGB: 1024
    managedDisk:
      storageAccountType: Premium_LRS
    osType: Linux
  publicIP: false
  publicLoadBalancer: <cluster_id> (8)
  resourceGroup: <cluster_id>-rg
  subnet: <cluster_id>-master-subnet (9)
  userDataSecret:
    name: master-user-data (10)
  vmSize: Standard_D8s_v3
  vnet: <cluster_id>-vnet
  zone: "1" (11)
```

1. クラスターのシークレット名を指定します。この値は変更しないでください。
2. コントロールプレーンマシンセットのイメージの詳細を指定します。
3. 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。
4. コントロールプレーンの内部ロードバランサーを指定します。このフィールドは事前入力されていない可能性がありますが、**ControlPlaneMachineSet** とコントロールプレーン **Machin** CR の両方で必要です。
5. クラウドプロバイダープラットフォームのタイプを指定します。この値は変更しないでください。
6. コントロールプレーンマシンを配置するリージョンを指定します。
7. コントロールプレーンのディスク設定を指定します。
8. コントロールプレーンのパブリックロードバランサーを指定します。

注記

ユーザー定義のアウトバウンドルーティングを持つプライベート OpenShift Container Platform クラスターでは、**publicLoadBalancer** パラメーターを省略できます。

9. コントロールプレーンのサブネットを指定します。
10. コントロールプレーンのユーザーデータシークレットを指定します。この値は変更しないでください。
11. すべての障害ドメインに対して単一のゾーンを使用するクラスターのゾーン設定を指定します。



注記

クラスターが障害ドメインごとに異なるゾーンを使用するように設定されている場合、このパラメーターは障害ドメインで設定されます。各障害ドメインに異なるゾーンを使用する場合にプロバイダー仕様でこの値を指定しても、その値は Control Plane Machine Set Operator によって無視されます。

12.5.2.1.2. Azure 障害ドメイン設定のサンプル

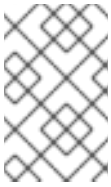
障害ドメインのコントロールプレーンマシンセットの概念は、Azure 可用性ゾーン の既存の Azure 概念に似ています。**ControlPlaneMachineSet** CR は、可能な場合、コントロールプレーンマシンを複数の障害ドメインに分散します。

コントロールプレーンマシンセットで Azure 障害ドメインを設定するときは、可用性ゾーン名を指定する必要があります。Azure クラスターは、複数のゾーンにまたがる単一のサブネットを使用します。

Azure 障害ドメインの値のサンプル

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
  name: cluster
  namespace: openshift-machine-api
spec:
# ...
  template:
# ...
    machines_v1beta1_machine_openshift_io:
      failureDomains:
        azure:
        - zone: "1" (1)
        - zone: "2"
        - zone: "3"
        platform: Azure (2)
# ...
```

(1) **zone** の各インスタンスは、障害ドメインの Azure アベイラビリティーゾーンを指定します。



注記

すべての障害ドメインに対して単一のゾーンを使用するようにクラスターが設定されている場合、**zone** パラメーターは、障害ドメイン設定ではなくプロバイダー仕様で設定されます。

(2) クラウドプロバイダーのプラットフォーム名を指定します。この値は変更しないでください。

12.5.2.2. コントロールプレーンマシンの Microsoft Azure 機能を有効にする

コントロールプレーンマシンセットの値を更新することで、機能を有効にします。

12.5.2.2.1. API サーバーをプライベートに制限する

クラスターを Amazon Web Services (AWS) にデプロイした後に、プライベートゾーンのみを使用するように API サーバーを再設定することができます。

前提条件

- OpenShift CLI (**oc**) がインストールされている。
- **admin** 権限を持つユーザーとして Web コンソールにアクセスできること。

手順

1. クラウドプロバイダーの Web ポータルまたはコンソールで、次の操作を行います。
   a. 適切なロードバランサーコンポーネントを見つけて削除します。
   b. パブリックゾーンの **api.$clustername.$yourdomain** DNS エントリーを削除します。
2. コントロールプレーンマシンセットのカスタムリソースで次の行を削除して、外部ロードバランサーを削除します。

```
# ...
providerSpec:
  value:
# ...
    loadBalancers:
    - name: lk4pj-ext 1
      type: network 2
    - name: lk4pj-int
      type: network
# ...
```

1. **-ext** で終わる外部ロードバランサーの **name** 値を削除します。
2. 外部ロードバランサーの **type** 値を削除します。

関連情報

- Ingress Controller エンドポイント公開スコープの内部への設定

12.5.2.2.2. Azure Marketplace オファリングの使用

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1VM イメージを表します。

重要

Azure マーケットプレイスを使用したイメージのインストールは、64 ビット ARM インスタンスを備えたクラスターではサポートされていません。

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

1. 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       Sku                 Urn                                                                  Version
     -------------  --------------  ------------------  -------------------------------------------------------------------  -----------------
     rh-ocp-worker  RedHat          rh-ocp-worker       RedHat:rh-ocp-worker:rh-ocp-worker:413.92.2023101700                 413.92.2023101700
     rh-ocp-worker  RedHat          rh-ocp-worker-gen1  RedHat:rh-ocp-worker:rh-ocp-worker-gen1:413.92.2023101700         413.92.2023101700
     ```

   - EMEA:

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

     出力例

     ```
     Offer          Publisher       Sku                 Urn                                                                  Version
     -------------  --------------  ------------------  -------------------------------------------------------------------  -----------------
     rh-ocp-worker  redhat-limited  rh-ocp-worker       redhat-limited:rh-ocp-worker:rh-ocp-worker:413.92.2023101700         413.92.2023101700
     rh-ocp-worker  redhat-limited  rh-ocp-worker-gen1  redhat-limited:rh-ocp-worker:rh-ocp-worker-gen1:413.92.2023101700     413.92.2023101700
     ```

注記

コンピュートおよびコントロールプレーンノードで利用可能な最新のイメージを使用します。必要に応じて、VM はインストールプロセスの一部として自動的にアップグレードされます。

2. 次のいずれかのコマンドを実行して、オファ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>
     ```

3. 次のコマンドのいずれかを実行して、オファ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、オファ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. オファーのイメージの詳細 (具体的には **publisher**、**offer**、**sku**、および **version** の値) を記録します。
6. オファー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 マシンのサンプル providerSpec イメージ値

   ```
   providerSpec:
     value:
       image:
         offer: rh-ocp-worker
         publisher: redhat
         resourceID: ""
         sku: rh-ocp-worker
         type: MarketplaceWithPlan
         version: 413.92.2023101700
   ```

12.5.2.2.3. Azure ブート診断の有効化

マシンセットが作成する Azure マシンで起動診断を有効にできます。

前提条件

- 既存の Microsoft Azure クラスターがある。

手順

- ストレージタイプに適用可能な **diagnostics** 設定を、マシンセット YAML ファイルの **providerSpec** フィールドに追加します。
  - Azure Managed ストレージアカウントの場合:

```
providerSpec:
  diagnostics:
    boot:
      storageAccountType: AzureManaged 1
```

1 Azure Managed ストレージアカウントを指定します。

  - Azure Unmanaged ストレージアカウントの場合:

```
providerSpec:
  diagnostics:
    boot:
      storageAccountType: CustomerManaged 1
      customerManaged:
        storageAccountURI: https://<storage-account>.blob.core.windows.net 2
```

1 Azure Unmanaged ストレージアカウントを指定します。

2 **<storage-account>** をストレージアカウントの名前に置き換えます。

注記

Azure Blob Storage データサービスのみサポートされています。

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 **起動診断** ページを確認し、マシンのシリアルログが表示されることを確認します。

12.5.2.2.4. 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。

関連情報

- Microsoft Azure Ultra ディスクのドキュメント

12.5.2.2.4.1. 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 クラスターがある。

手順

1. 次のコマンドを実行して、**master** 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 <role>-user-data \ 1
--template='{{index .data.userData | base64decode}}' | jq > userData.txt 2
```

1. **<role>** を **master** に置き換えます。
2. 新しいカスタムシークレットの名前として **userData.txt** を指定します。

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 <role>-user-data \ 1
--template='{{index .data.disableTemplating | base64decode}}' | jq > disableTemplating.txt
```

1. **<role>** を **master** に置き換えます。

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 <role>-user-data-x5 \ 1
--from-file=userData=userData.txt \
--from-file=disableTemplating=disableTemplating.txt
```

1. **<role>-user-data-x5** には、シークレットの名前を指定します。**<role>** を **master** に置き換えます。

5. 次のコマンドを実行して、コントロールプレーンマシンセットの CR を編集します。

```
$ oc --namespace openshift-machine-api edit controlplanemachineset.machine.openshift.io
cluster
```

6. 示された位置に次の行を追加します。

```
apiVersion: machine.openshift.io/v1beta1
kind: ControlPlaneMachineSet
spec:
  template:
    spec:
      metadata:
        labels:
          disk: ultrassd 1
      providerSpec:
        value:
          ultraSSDCapability: Enabled 2
          dataDisks: 3
          - nameSuffix: ultrassd
            lun: 0
            diskSizeGB: 4
            deletionPolicy: Delete
            cachingType: None
            managedDisk:
              storageAccountType: UltraSSD_LRS
          userDataSecret:
            name: <role>-user-data-x5 4
```

1 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.ultrassd** を使用します。

2 3 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。

4 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。**<role>** を **master** に置き換えます。

7. 変更を保存します。

- デフォルトの **RollingUpdate** 更新戦略を使用するクラスターの場合、Operator は自動的に変更をコントロールプレーン設定に伝達します。
- **OnDelete** 更新戦略を使用するように設定されているクラスターの場合、コントロールプレーンマシンを手動で置き換える必要があります。

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

- コントロールプレーンで Ultra ディスクを使用するには、コントロールプレーンの Ultra ディスクマウントポイントを使用するようにワークロードを再設定します。

12.5.2.2.4.2. 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。

12.5.2.2.4.2.1. 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。

12.5.2.2.4.2.2. 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、アベイラビリティーゾーン、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。

12.5.2.2.4.2.3. 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。

12.5.2.2.5. 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、保存中に管理ディスクのデータを暗号化できます。Machine API を使

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。Machine API を使用すると、顧客管理の鍵によるサーバー側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) がアクセス許可を付与したリソースグループ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

- カスタマーマネージドキーに関する Azure ドキュメント

12.5.2.2.6. マシンセットを使用した Azure 仮想マシンの信頼された起動の設定

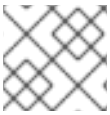
重要

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

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

OpenShift Container Platform 4.16 は、Azure 仮想マシンの信頼された起動をサポートします。マシンセットの YAML ファイルを編集することで、マシンセットがデプロイメントするマシンに使用する信頼できる起動オプションを設定できます。たとえば、セキュアブートや専用の仮想 Trusted Platform

Module (vTPM) インスタンスなどの UEFI セキュリティー機能を使用するようにこれらのマシンを設定できます。



注記

一部の機能の組み合わせでは、無効な設定が発生します。

表12.2 UEFI 機能の組み合わせの互換性

Secure Boot[1]	vTPM[2]	有効な設定
有効	有効	はい
有効	無効	はい
有効	省略	はい
無効	有効	はい
省略	有効	はい
無効	無効	いいえ
省略	無効	いいえ
省略	省略	いいえ

1. **secureBoot** フィールドの使用。
2. **virtualizedTrustedPlatformModule** フィールドの使用。

関連する機能の詳細は、Azure 仮想マシンの信頼された起動 に関する Microsoft Azure のドキュメントを参照してください。

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. **providerSpec** フィールドの下の次のセクションを編集して、有効な設定を指定します。

UEFI セキュアブートと vTPM が有効になっている有効な設定のサンプル

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
# ...
spec:
  template:
    spec:
      providerSpec:
        value:
```

```
          securityProfile:
            settings:
              securityType: TrustedLaunch (1)
              trustedLaunch:
                uefiSettings: (2)
                  secureBoot: Enabled (3)
                  virtualizedTrustedPlatformModule: Enabled (4)
# ...
```

(1) Azure 仮想マシンの信頼された起動の使用を有効にします。この値は、すべての有効な設定に必要です。

(2) 使用する UEFI セキュリティー機能を指定します。このセクションは、すべての有効な設定に必要です。

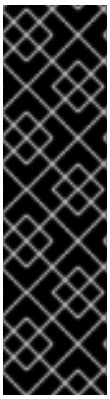
(3) UEFI セキュアブートを有効にします。

(4) vTPM の使用を有効にします。

検証

- 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詳細を確認し、信頼された起動オプションが設定した値と一致することを確認します。

12.5.2.2.7. マシンセットを使用した Azure 機密仮想マシンの設定



重要

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

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

OpenShift Container Platform 4.16 は、Azure 機密仮想マシンをサポートします。

注記

現在、機密仮想マシンは 64 ビット ARM アーキテクチャーではサポートされていません。

マシンセットの YAML ファイルを編集することにより、マシンセットがデプロイするマシンに使用する Confidential VM オプションを設定できます。たとえば、セキュアブートや専用の仮想 Trusted Platform Module (vTPM) インスタンスなどの UEFI セキュリティー機能を使用するようにこれらのマシンを設定できます。

警告

すべてのインスタンスタイプが機密仮想マシンをサポートしているわけではありません。機密仮想マシンを使用するように設定されているコントロールプレーンマシンセットのインスタンスタイプを、互換性のないタイプに変更しないでください。互換性のないインスタンスタイプを使用すると、クラスターが不安定になる可能性があります。

関連する機能の詳細は、機密仮想マシン に関する Microsoft Azure のドキュメントを参照してください。

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. **providerSpec** フィールドの下の次のセクションを編集します。

設定サンプル

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
# ...
spec:
  template:
    spec:
      providerSpec:
        value:
          osDisk:
            # ...
            managedDisk:
              securityProfile: (1)
                securityEncryptionType: VMGuestStateOnly (2)
            # ...
          securityProfile: (3)
            settings:
              securityType: ConfidentialVM (4)
              confidentialVM:
                uefiSettings: (5)
                  secureBoot: Disabled (6)
                  virtualizedTrustedPlatformModule: Enabled (7)
          vmSize: Standard_DC16ads_v5 (8)
# ...
```

(1) 機密仮想マシンを使用する場合のマネージドディスクのセキュリティープロファイル設定を指定します。

(2) Azure 仮想マシンゲスト状態 (VMGS) ブロブの暗号化を有効にします。この設定には vTPM の使用が必要です。

3 機密仮想マシンのセキュリティープロファイル設定を指定します。

4 機密仮想マシンの使用を有効にします。この値は、すべての有効な設定に必要です。

5 使用する UEFI セキュリティー機能を指定します。このセクションは、すべての有効な設定に必要です。

6 UEFI セキュアブートを無効にします。

7 vTPM の使用を有効にします。

8 機密仮想マシンをサポートするインスタンスタイプを指定します。

検証

- 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詳細を確認し、Confidential VM オプションが設定した値に一致していることを確認します。

12.5.2.2.8. 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、インストール後に有効にすることができます。

12.5.2.2.8.1. 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。

- Accelerated Networking は、Machine API が動作しているクラスターでのみサポートされます。
- 高速ネットワークには、少なくとも 4 つの vCPU を含む Azure VM 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。

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

マシンセット YAML ファイルに **acceleratedNetworking** を追加することで、Azure で Accelerated Networking を有効にすることができます。

前提条件

- Machine API が動作している既存の Microsoft Azure クラスターがある。

手順

- 以下を **providerSpec** フィールドに追加します。

```
providerSpec:
  value:
    acceleratedNetworking: true 1
    vmSize: <azure-vm-size> 2
```

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

2 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。

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。

12.5.3. Google Cloud Platform のコントロールプレーン設定オプション

コントロールプレーンマシンセットの値を更新することで、Google Cloud Platform (GCP) コントロールプレーンマシンの設定を変更し、機能を有効にすることができます。コントロールプレーンマシンセットへの更新を保存すると、設定された 更新ストラテジー に従って Control Plane Machine Set Operator がコントロールプレーンマシンを更新します。

12.5.3.1. Google Cloud Platform クラスターを設定するためのサンプル YAML

次の YAML スニペットの例は、GCP クラスターのプロバイダーの仕様と障害ドメインの設定を示しています。

12.5.3.1.1. サンプル GCP プロバイダーの仕様

既存のクラスター用にコントロールプレーンマシンセットを作成する場合、プロバイダーの仕様は、インストールプログラムによって作成されるコントロールプレーンマシンのカスタムリソース (CR) の **providerSpec** 設定と一致する必要があります。CR の障害ドメインセクションに設定されているフィールドは省略できます。

OpenShift CLI を使用して取得した値
以下の例では、OpenShift CLI を使用してクラスターの値の一部を取得できます。

インフラストラクチャー ID

<cluster_id> 文字列は、クラスターをプロビジョニングしたとき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。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
```

イメージパス

<path_to_image> 文字列は、ディスクの作成に使用されたイメージへのパスです。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.machines_v1beta1_machine_openshift_io.spec.providerSpec.value.disks[
].image}{"\n"}' \
  get ControlPlaneMachineSet/cluster
```

サンプル GCP providerSpec 値

```
apiVersion: machine.openshift.io/v1
```

```
kind: ControlPlaneMachineSet
metadata:
  name: cluster
  namespace: openshift-machine-api
spec:
# ...
  template:
# ...
      spec:
        providerSpec:
          value:
            apiVersion: machine.openshift.io/v1beta1
            canIPForward: false
            credentialsSecret:
              name: gcp-cloud-credentials (1)
            deletionProtection: false
            disks:
            - autoDelete: true
              boot: true
              image: <path_to_image> (2)
              labels: null
              sizeGb: 200
              type: pd-ssd
            kind: GCPMachineProviderSpec (3)
            machineType: e2-standard-4
            metadata:
              creationTimestamp: null
            metadataServiceOptions: {}
            networkInterfaces:
            - network: <cluster_id>-network
              subnetwork: <cluster_id>-master-subnet
            projectID: <project_name> (4)
            region: <region> (5)
            serviceAccounts:
            - email: <cluster_id>-m@<project_name>.iam.gserviceaccount.com
              scopes:
              - https://www.googleapis.com/auth/cloud-platform
            shieldedInstanceConfig: {}
            tags:
            - <cluster_id>-master
            targetPools:
            - <cluster_id>-api
            userDataSecret:
              name: master-user-data (6)
            zone: "" (7)
```

(1) クラスターのシークレット名を指定します。この値は変更しないでください。

(2) ディスクの作成に使用されたイメージへのパスを指定します。

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-413-x86-64-202305021736**

- OpenShift Platform Plus: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-opp-413-x86-64-202305021736**
- OpenShift Kubernetes Engine: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-oke-413-x86-64-202305021736**

3 クラウドプロバイダープラットフォームのタイプを指定します。この値は変更しないでください。

4 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。

5 クラスターの GCP リージョンを指定します。

6 コントロールプレーンのユーザーデータシークレットを指定します。この値は変更しないでください。

7 このパラメーターは障害ドメインで設定され、ここでは空の値で表示されます。このパラメーターに指定された値が障害ドメインの値と異なる場合、Operator はそれを障害ドメインの値で上書きします。

12.5.3.1.2. GCP 障害ドメインの設定例

障害ドメインのコントロールプレーンマシンセットの概念は、既存の GCP の ゾーン の概念に似ています。**ControlPlaneMachineSet** CR は、可能な場合、コントロールプレーンマシンを複数の障害ドメインに分散します。

コントロールプレーンマシンセットで GCP 障害ドメインを設定する場合は、使用するゾーン名を指定する必要があります。

GCP 障害ドメインの値の例

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
  name: cluster
  namespace: openshift-machine-api
spec:
# ...
  template:
# ...
    machines_v1beta1_machine_openshift_io:
      failureDomains:
        gcp:
        - zone: <gcp_zone_a> 1
        - zone: <gcp_zone_b> 2
        - zone: <gcp_zone_c>
        - zone: <gcp_zone_d>
        platform: GCP 3
# ...
```

1 最初の障害ドメインの GCP ゾーンを指定します。

2 追加の障害ドメインを指定します。さらに障害ドメインが同じ方法で追加されます。

3 クラウドプロバイダーのプラットフォーム名を指定します。この値は変更しないでください。

12.5.3.2. コントロールプレーンマシンの Google Cloud Platform 機能を有効にする

コントロールプレーンマシンセットの値を更新することで、機能を有効にします。

12.5.3.2.1. 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。

永続ディスクのタイプ、互換性、リージョン別の可用性、制限事項の詳細は、永続ディスク に関する GCP Compute Engine のドキュメントを参照してください。

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. **providerSpec** フィールドの下で以下の行を編集します。

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
...
spec:
  template:
    spec:
      providerSpec:
        value:
          disks:
            type: pd-ssd 1
```

1 コントロールプレーンノードは、**pd-ssd** ディスクタイプを使用する必要があります。

検証

- Google Cloud コンソールを使用して、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に一致していることを確認します。

12.5.3.2.2. マシンセットを使用した Confidential VM の設定

マシンセットの YAML ファイルを編集することにより、マシンセットがデプロイするマシンに使用する Confidential VM オプションを設定できます。

機密仮想マシンの機能、互換性の詳細は、Confidential VM に関する GCP Compute Engine のドキュメントを参照してください。

注記

現在、機密仮想マシンは 64 ビット ARM アーキテクチャーではサポートされていません。

重要

OpenShift Container Platform 4.16 は、AMD Secure Encrypted Virtualization Secure Nested Paging (SEV-SNP) を備えた機密仮想マシンなど、一部の Confidential Compute 機能をサポートしていません。

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. **providerSpec** フィールドの下の次のセクションを編集します。

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
...
spec:
  template:
    spec:
      providerSpec:
        value:
          confidentialCompute: Enabled (1)
          onHostMaintenance: Terminate (2)
          machineType: n2d-standard-8 (3)
...
```

(1) Confidential VM を有効にするかどうかを指定します。有効な値は **Disabled** または **Enabled** です。

(2) ハードウェアやソフトウェアの更新など、ホストのメンテナンスイベント中の VM の動作を指定します。Confidential VM を使用するマシンの場合は、この値を **Terminate** に設定する必要があります。これにより、VM が停止します。Confidential VM はライブ VM 移行をサポートしていません。

(3) Confidential VM をサポートするマシンタイプを指定します。Confidential VM は、N2D および C2D シリーズのマシンタイプをサポートしています。

検証

- GCP 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、Confidential VM オプションが設定した値に一致していることを確認します。

12.5.3.2.3. マシンセットを使用した Shielded VM オプションの設定

マシンセットの YAML ファイルを編集することで、マシンセットがデプロイメントするマシンに使用する Shielded VM オプションを設定できます。

Shielded VM の特徴と機能の詳細については、Shielded VM に関する GCP Compute Engine のドキュメントを参照してください。

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. **providerSpec** フィールドの下の次のセクションを編集します。

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
# ...
spec:
  template:
    spec:
      providerSpec:
        value:
          shieldedInstanceConfig: 1
            integrityMonitoring: Enabled 2
            secureBoot: Disabled 3
            virtualizedTrustedPlatformModule: Enabled 4
# ...
```

1. このセクションでは、必要な Shielded VM オプションを指定します。
2. 整合性監視を有効にするかどうかを指定します。有効な値は **Disabled** または **Enabled** です。

注記

整合性監視が有効になっている場合、仮想トラステッドプラットフォームモジュール (vTPM) を無効にしないでください。

3. UEFI セキュアブートを有効にするかどうかを指定します。有効な値は **Disabled** または **Enabled** です。
4. vTPM を有効にするかどうかを指定します。有効な値は **Disabled** または **Enabled** です。

検証

- Google Cloud コンソールを使用して、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、Shielded VM オプションが設定した値に一致していることを確認します。

関連情報

- Shielded VM とは
  - セキュアブート
  - 仮想トラステッドプラットフォームモジュール (vTPM)
  - 整合性監視

12.5.3.2.4. 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。

Machine API を使用するクラスターでは、顧客管理の鍵による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. 特定のサービスアカウントが KMS キーを使用できるようにし、サービスアカウントに正しい IAM ロールを付与するには、KMS キー名、キーリング名、場所を指定して次のコマンドを実行します。

```
$ 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。

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
...
spec:
  template:
    spec:
      providerSpec:
        value:
          disks:
          - type:
            encryptionKey:
              kmsKey:
                name: machine-encryption-key 1
                keyRing: openshift-encrpytion-ring 2
                location: global 3
                projectID: openshift-gcp-project 4
              kmsKeyServiceAccount: openshift-service-account@openshift-gcp-
project.iam.gserviceaccount.com 5
```

1. ディスク暗号化に使用される顧客管理の暗号鍵の名前。
2. KMS キーが属する KMS キーリングの名前。
3. KMS キーリングが存在する GCP の場所。

4 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。

5 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しいマシンが作成されると、ディスク暗号化キーが KMS キーで暗号化されます。

12.5.4. Nutanix のコントロールプレーン設定オプション

コントロールプレーンマシンセットの値を更新することで、Nutanix コントロールプレーンマシンの設定を変更できます。コントロールプレーンマシンセットへの更新を保存すると、設定された 更新ストラテジー に従って Control Plane Machine Set Operator がコントロールプレーンマシンを更新します。

12.5.4.1. Nutanix クラスターを設定するためのサンプル YAML

次の YAML スニペットの例は、Nutanix クラスターのプロバイダー仕様の設定を示しています。

12.5.4.1.1. Nutanix プロバイダー仕様のサンプル

既存のクラスター用にコントロールプレーンマシンセットを作成する場合、プロバイダーの仕様は、インストールプログラムによって作成されるコントロールプレーンマシンのカスタムリソース (CR) の **providerSpec** 設定と一致する必要があります。

OpenShift CLI を使用して取得した値
以下の例では、OpenShift CLI を使用してクラスターの値の一部を取得できます。

インフラストラクチャー ID

<cluster_id> 文字列は、クラスターをプロビジョニングしたとき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。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。

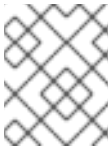
```
$ oc get -o jsonpath='{.status.infrastructureName}{"\n"}' infrastructure cluster
```

Nutanix の providerSpec 値のサンプル

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
  name: cluster
  namespace: openshift-machine-api
spec:
# ...
  template:
# ...
      spec:
        providerSpec:
          value:
            apiVersion: machine.openshift.io/v1
            bootType: "" 1
            categories: 2
```

```
    - key: <category_name>
      value: <category_value>
  cluster: 3
    type: uuid
    uuid: <cluster_uuid>
  credentialsSecret:
    name: nutanix-credentials 4
  image: 5
    name: <cluster_id>-rhcos
    type: name
  kind: NutanixMachineProviderConfig 6
  memorySize: 16Gi 7
  metadata:
    creationTimestamp: null
  project: 8
    type: name
    name: <project_name>
  subnets: 9
  - type: uuid
    uuid: <subnet_uuid>
  systemDiskSize: 120Gi 10
  userDataSecret:
    name: master-user-data 11
  vcpuSockets: 8 12
  vcpusPerSocket: 1 13
```

1 コントロールプレーンマシンが使用するブートタイプを指定します。ブートタイプの詳細については、仮想化環境内の UEFI、セキュアブート、および TPM について を参照してください。有効な値は、**Legacy**、**SecureBoot**、または **UEFI** です。デフォルトは、**Legacy** です。

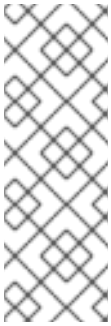


注記

OpenShift Container Platform 4.16 では、**Legacy** ブートタイプを使用する必要があります。

2 コントロールプレーンマシンに適用する 1 つ以上の Nutanix Prism カテゴリーを指定します。このスタンザには、Prism Central に存在するカテゴリーのキーと値のペアの **key** および **value** パラメーターが必要です。カテゴリーの詳細は、カテゴリー管理 を参照してください。

3 Nutanix Prism Element のクラスター設定を指定します。この例のクラスタータイプは **uuid** であるため、**uuid** スタンザがあります。



注記

OpenShift Container Platform バージョン 4.15 以降を使用するクラスターは、障害ドメイン設定を使用できます。

クラスターが障害ドメインを使用するように設定されている場合、このパラメーターは障害ドメインで設定されます。障害ドメインを使用する場合にプロバイダー仕様でこの値を指定しても、その値は Control Plane Machine Set Operator によって無視されます。

4 クラスターのシークレット名を指定します。この値は変更しないでください。

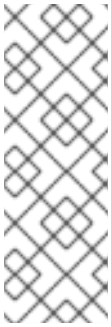
5 ディスクの作成に使用されたイメージを指定します。

6 クラウドプロバイダープラットフォームのタイプを指定します。この値は変更しないでください。

7 コントロールプレーンマシンに割り当てられるメモリーを指定します。

8 クラスターに使用する Nutanix プロジェクトを指定します。この例のプロジェクトタイプは **name** であるため、**name** スタンザがあります。

9 サブネット設定を指定します。この例では、サブネットタイプは **uuid** であるため、**uuid** スタンザがあります。



注記

OpenShift Container Platform バージョン 4.15 以降を使用するクラスターは、障害ドメイン設定を使用できます。

クラスターが障害ドメインを使用するように設定されている場合、このパラメーターは障害ドメインで設定されます。障害ドメインを使用する場合にプロバイダー仕様でこの値を指定しても、その値は Control Plane Machine Set Operator によって無視されます。

10 コントロールプレーンマシンの VM ディスクサイズを指定します。

11 コントロールプレーンのユーザーデータシークレットを指定します。この値は変更しないでください。

12 コントロールプレーンマシンに割り当てられる vCPU ソケットの数を指定します。

13 各コントロールプレーン vCPU ソケットの vCPU の数を指定します。

12.5.4.1.2. Nutanix クラスターの障害ドメイン

Nutanix クラスターの障害ドメイン設定を追加または更新するには、整合性のある変更を複数のリソースに加える必要があります。次の操作が必要です。

1. クラスターインフラストラクチャーのカスタムリソース (CR) を変更します。
2. クラスターコントロールプレーンマシンセットの CR を変更します。
3. コンピュートマシンセットの CR を変更または置き換えます。

詳細は、**インストール後の設定** コンテンツの「既存の Nutanix クラスターへの障害ドメインの追加」を参照してください。

関連情報

- 既存の Nutanix クラスターへの障害ドメインの追加

12.5.5. Red Hat OpenStack Platform (RHOSP) のコントロールプレーン設定オプション

コントロールプレーンマシンセットの値を更新することで、Red Hat OpenStack Platform (RHOSP) コントロールプレーンマシンの設定を変更し、機能を有効にすることができます。コントロールプレーン

マシンセットへの更新を保存すると、設定された 更新ストラテジー に従って Control Plane Machine Set Operator がコントロールプレーンマシンを更新します。

12.5.5.1. Red Hat OpenStack Platform (RHOSP) クラスターを設定するためのサンプル YAML

次の YAML スニペットの例は、RHOSP クラスターのプロバイダーの仕様と障害ドメインの設定を示しています。

12.5.5.1.1. RHOSP プロバイダー仕様のサンプル

既存のクラスター用にコントロールプレーンマシンセットを作成する場合、プロバイダーの仕様は、インストールプログラムによって作成されるコントロールプレーンマシンのカスタムリソース (CR) の **providerSpec** 設定と一致する必要があります。

OpenStack の providerSpec 値のサンプル

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
  name: cluster
  namespace: openshift-machine-api
spec:
# ...
  template:
# ...
      spec:
        providerSpec:
          value:
            apiVersion: machine.openshift.io/v1alpha1
            cloudName: openstack
            cloudsSecret:
              name: openstack-cloud-credentials 1
              namespace: openshift-machine-api
            flavor: m1.xlarge 2
            image: ocp1-2g2xs-rhcos
            kind: OpenstackProviderSpec 3
            metadata:
              creationTimestamp: null
            networks:
            - filter: {}
              subnets:
              - filter:
                  name: ocp1-2g2xs-nodes
                  tags: openshiftClusterID=ocp1-2g2xs
            securityGroups:
            - filter: {}
              name: ocp1-2g2xs-master 4
            serverGroupName: ocp1-2g2xs-master
            serverMetadata:
              Name: ocp1-2g2xs-master
              openshiftClusterID: ocp1-2g2xs
            tags:
            - openshiftClusterID=ocp1-2g2xs
```

```
        trunk: true
        userDataSecret:
          name: master-user-data
```

1. クラスターのシークレット名。この値は変更しないでください。
2. コントロールプレーンの RHOSP フレーバータイプ。
3. RHOSP クラウドプロバイダーのプラットフォームタイプ。この値は変更しないでください。
4. コントロールプレーンマシンのセキュリティーグループ。

12.5.5.1.2. RHOSP 障害ドメイン設定のサンプル

障害ドメインのコントロールプレーンマシンセットの概念は、既存の Red Hat OpenStack Platform (RHOSP) の アベイラビリティーゾーン の概念に似ています。**ControlPlaneMachineSet** CR は、可能な場合、コントロールプレーンマシンを複数の障害ドメインに分散します。

次の例は、複数の Nova アベイラビリティーゾーンと Cinder アベイラビリティーゾーンの使用を示しています。

OpenStack 障害ドメイン値のサンプル

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
  name: cluster
  namespace: openshift-machine-api
spec:
# ...
  template:
# ...
    machines_v1beta1_machine_openshift_io:
      failureDomains:
        platform: OpenStack
        openstack:
        - availabilityZone: nova-az0
          rootVolume:
            availabilityZone: cinder-az0
        - availabilityZone: nova-az1
          rootVolume:
            availabilityZone: cinder-az1
        - availabilityZone: nova-az2
          rootVolume:
            availabilityZone: cinder-az2
# ...
```

12.5.5.2. コントロールプレーンマシンの Red Hat OpenStack Platform (RHOSP) 機能の有効化

コントロールプレーンマシンセットの値を更新することで、機能を有効にします。

12.5.5.2.1. コントロールプレーンマシンセットを使用した RHOSP コンピュートフレーバーの変更

コントロールプレーンマシンセットのカスタムリソースの仕様を更新することで、コントロールプレーンマシンが使用する Red Hat OpenStack Platform (RHOSP) コンピュートサービス (Nova) フレーバーを変更できます。

RHOSP では、フレーバーはコンピューティングインスタンスのコンピュート、メモリー、およびストレージ容量を定義します。フレーバーのサイズを増減することで、コントロールプレーンを垂直方向にスケールできます。

前提条件

- RHOSP クラスターはコントロールプレーンマシンセットを使用します。

手順

1. **providerSpec** フィールドの下で以下の行を編集します。

```
providerSpec:
  value:
# ...
    flavor: m1.xlarge (1)
```

(1) 既存の選択と同じベースを持つ RHOSP フレーバータイプを指定します。たとえば、**m6i.xlarge** を **m6i.2xlarge** または **m6i.4xlarge** に変更できます。垂直方向のスケーリングのニーズに応じて、より大きいフレーバーまたはより小さいフレーバーを選択できます。

2. 変更を保存します。

変更を保存すると、マシンは選択したフレーバーを使用するマシンに置き換えられます。

12.5.6. VMware vSphere のコントロールプレーン設定オプション

コントロールプレーンマシンセットの値を更新することで、VMware vSphere コントロールプレーンマシンの設定を変更できます。コントロールプレーンマシンセットへの更新を保存すると、設定された 更新ストラテジー に従って Control Plane Machine Set Operator がコントロールプレーンマシンを更新します。

12.5.6.1. VMware vSphere クラスターを設定するためのサンプル YAML

次の YAML スニペットの例は、vSphere クラスターのプロバイダーの仕様と障害ドメインの設定を示しています。

12.5.6.1.1. VMware vSphere プロバイダー仕様のサンプル

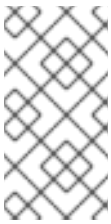
既存のクラスター用にコントロールプレーンマシンセットを作成する場合、プロバイダーの仕様は、インストールプログラムによって作成されるコントロールプレーンマシンのカスタムリソース (CR) の **providerSpec** 設定と一致する必要があります。

サンプルの vSphere providerSpec 値

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
```

```
  name: cluster
  namespace: openshift-machine-api
spec:
# ...
  template:
# ...
      spec:
        providerSpec:
          value:
            apiVersion: machine.openshift.io/v1beta1
            credentialsSecret:
              name: vsphere-cloud-credentials 1
            diskGiB: 120 2
            kind: VSphereMachineProviderSpec 3
            memoryMiB: 16384 4
            metadata:
              creationTimestamp: null
            network: 5
              devices:
              - networkName: <vm_network_name>
            numCPUs: 4 6
            numCoresPerSocket: 4 7
            snapshot: ""
            template: <vm_template_name> 8
            userDataSecret:
              name: master-user-data 9
            workspace: 10
              datacenter: <vcenter_data_center_name> 11
              datastore: <vcenter_datastore_name> 12
              folder: <path_to_vcenter_vm_folder> 13
              resourcePool: <vsphere_resource_pool> 14
              server: <vcenter_server_ip> 15
```

1. クラスターのシークレット名を指定します。この値は変更しないでください。
2. コントロールプレーンマシンの VM ディスクサイズを指定します。
3. クラウドプロバイダープラットフォームのタイプを指定します。この値は変更しないでください。
4. コントロールプレーンマシンに割り当てられるメモリーを指定します。
5. コントロールプレーンがデプロイされるネットワークを指定します。



注記

クラスターが障害ドメインを使用するように設定されている場合、このパラメーターは障害ドメインで設定されます。障害ドメインを使用する場合にプロバイダー仕様でこの値を指定しても、その値は Control Plane Machine Set Operator によって無視されます。

6. コントロールプレーンマシンに割り当てられる CPU の数を指定します。
7. 各コントロールプレーン CPU のコア数を指定します。

8 **user-5ddjd-rhcos** など、使用する vSphere VM テンプレートを指定します。

9 コントロールプレーンのユーザーデータシークレットを指定します。この値は変更しないでください。

10 コントロールプレーンのワークスペースの詳細を指定します。

注記

クラスターが障害ドメインを使用するように設定されている場合、これらのパラメーターは障害ドメインで設定されます。障害ドメインを使用する場合にプロバイダー仕様でこれらの値を指定しても、それらの値は Control Plane Machine Set Operator によって無視されます。

11 コントロールプレーンの vCenter データセンターを指定します。

12 コントロールプレーンの vCenter データストアを指定します。

13 **/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。

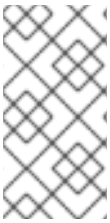
14 仮想マシンの vSphere リソースプールを指定します。

15 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。

12.5.6.1.2. VMware vSphere 障害ドメイン設定のサンプル

VMware vSphere インフラストラクチャーでは、クラスター全体のインフラストラクチャーのカスタムリソース定義 (CRD) である **infrastructures.config.openshift.io** によってクラスターの障害ドメインが定義されます。**ControlPlaneMachineSet** カスタムリソース (CR) の **providerSpec** は、コントロールプレーンマシンセットがコントロールプレーンノードが適切な障害ドメインにデプロイされるようにするために使用する障害ドメインの名前を指定します。障害ドメインは、コントロールプレーンマシンセット、vCenter データセンター、vCenter データストア、およびネットワークで構成されるインフラストラクチャーリソースです。

障害ドメインリソースを使用すると、コントロールプレーンマシンセットを使用して、コントロールプレーンマシンを別のクラスターまたはデータセンターにデプロイできます。また、コントロールプレーンマシンセットは、定義された障害ドメイン全体でコントロールプレーンマシンのバランスをとり、インフラストラクチャーにフォールトトレランス機能を提供します。



注記

ControlPlaneMachineSet CR の **ProviderSpec** 設定を変更すると、コントロールプレーンマシンセットによって、プライマリーインフラストラクチャーと各障害ドメインインフラストラクチャーにデプロイされているすべてのコントロールプレーンマシンが更新されます。

VMware vSphere 障害ドメインの値の例

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
  name: cluster
  namespace: openshift-machine-api
```

```
spec:
# ...
  template:
# ...
    machines_v1beta1_machine_openshift_io:
      failureDomains: 1
        platform: VSphere
        vsphere: 2
        - name: <failure_domain_name1>
        - name: <failure_domain_name2>
# ...
```

1 OpenShift Container Platform クラスターノードの vCenter の場所を指定します。

2 コントロールプレーンマシンセットの障害ドメインを名前で指定します。

重要

このセクションの各 **name** フィールドの値は、クラスター全体のインフラストラクチャー CRD の **failureDomains.name** フィールドの対応する値と一致する必要があります。次のコマンドを実行すると、**failureDomains.name** フィールドの値を見つけることができます。

```
$ oc get infrastructure cluster -o=jsonpath=
{.spec.platformSpec.vsphere.failureDomains[0].name
```

name フィールドは、**ControlPlaneMachineSet** CR で指定できる、サポートされている唯一の障害ドメインフィールドです。

各障害ドメインのリソースを定義するクラスター全体のインフラストラクチャー CRD の例については、「vSphere 上のクラスターに複数のリージョンとゾーンを指定する」を参照してください。

関連情報

- vSphere 上のクラスターに複数のリージョンとゾーンを指定する

12.5.6.2. コントロールプレーンマシンの VMware vSphere 機能を有効にする

コントロールプレーンマシンセットの値を更新することで、機能を有効にします。

12.5.6.2.1. マシンセットの使用によるマシンへのタグの追加

OpenShift Container Platform は、作成する各仮想マシンにクラスター固有のタグを追加します。インストールプログラムはこれらのタグを使用して、クラスターをアンインストールするときに削除する仮想マシンを選択します。

仮想マシンに割り当てられたクラスター固有のタグのほかに、プロビジョニングする仮想マシンに最大 10 個の vSphere タグを追加するようにマシンセットを設定できます。

前提条件

- **cluster-admin** 権限を持つアカウントを使用して、vSphere にインストールされた OpenShift Container Platform クラスターにアクセスできる。

- クラスターに関連付けられた VMware vCenter コンソールにアクセスできる。
- vCenter コンソールにタグを作成している。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. vCenter コンソールを使用して、マシンに追加するタグのタグ ID を検索します。
   a. vCenter コンソールにログインします。
   b. **Home** メニューから、**Tags & Custom Attributes**をクリックします。
   c. マシンに追加するタグを選択します。
   d. 選択したタグのブラウザー URL を使用して、タグ ID を特定します。

   タグ URL の例

   ```
   https://vcenter.example.com/ui/app/tags/tag/urn:vmomi:InventoryServiceTag:208e713c-cae3-4b7f-918e-4051ca7d1f97:GLOBAL/permissions
   ```

   タグ ID の例

   ```
   urn:vmomi:InventoryServiceTag:208e713c-cae3-4b7f-918e-4051ca7d1f97:GLOBAL
   ```

2. 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。
3. **providerSpec** フィールドの下に次の行を編集します。

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
# ...
spec:
  template:
    spec:
      providerSpec:
        value:
          tagIDs: 1
          - <tag_id_value> 2
# ...
```

1. このマシンセットがプロビジョニングするマシンに追加する最大 10 個のタグのリストを指定します。
2. マシンに追加するタグの値を指定します。たとえば、**urn:vmomi:InventoryServiceTag:208e713c-cae3-4b7f-918e-4051ca7d1f97:GLOBAL** です。

12.6. コントロールプレーンの回復力と回復

コントロールプレーンマシンセットを使用して、OpenShift Container Platform クラスターのコントロールプレーンの回復力を向上させることができます。

12.6.1. 障害ドメインによる高可用性と耐障害性

可能な場合、コントロールプレーンマシンセットは、コントロールプレーンマシンを複数の障害ドメインに分散します。この設定は、コントロールプレーン内で高可用性とフォールトトレランスを提供します。この戦略は、インフラストラクチャープロバイダー内で問題が発生した場合に、コントロールプレーンを保護するのに役立ちます。

12.6.1.1. 障害ドメインプラットフォームのサポートと設定

障害ドメインのコントロールプレーンマシンセットの概念は、クラウドプロバイダーの既存の概念に似ています。すべてのプラットフォームが障害ドメインの使用をサポートしているわけではありません。

表12.3 障害ドメインのサポートマトリックス

クラウドプロバイダー	障害ドメインのサポート	プロバイダーの命名法
Amazon Web Services (AWS)	X	アベイラビリティーゾーン (AZ)
Google Cloud Platform (GCP)	X	zone
Microsoft Azure	X	Azure アベイラビリティゾーン
Nutanix	X	障害ドメイン
Red Hat OpenStack Platform (RHOSP)	X	OpenStack Nova アベイラビリティーゾーン と OpenStack Cinder アベイラビリティーゾーン
VMware vSphere	X	vSphere Zone にマッピングされた障害ドメイン [1]

1. 詳細は、「VMware vCenter のリージョンとゾーン」を参照してください。

コントロールプレーンマシンセットカスタムリソース (CR) の障害ドメイン設定は、プラットフォーム固有です。CR の障害ドメインパラメーターの詳細については、プロバイダーの障害ドメイン設定のサンプルを参照してください。

関連情報

- アマゾンウェブサービス障害ドメイン設定のサンプル
- サンプルの Google Cloud Platform 障害ドメイン設定
- サンプルの Microsoft Azure 障害ドメイン設定
- 既存の Nutanix クラスターへの障害ドメインの追加
- Red Hat OpenStack Platform (RHOSP) 障害ドメイン設定のサンプル

- VMware vSphere 障害ドメイン設定のサンプル
- VMware vCenter のリージョンとゾーン

12.6.1.2. コントロールプレーンマシンのバランス調整

コントロールプレーンマシンセットは、カスタムリソース (CR) で指定された障害ドメイン全体でコントロールプレーンマシンのバランスをとります。

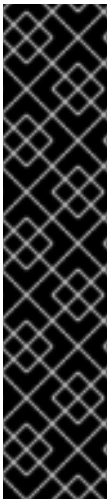
可能な場合、コントロールプレーンマシンセットは各障害ドメインを均等に使用して、適切なフォールトトレランスを確保します。コントロールプレーンマシンよりも障害ドメインが少ない場合、障害ドメインは名前のアルファベット順に選択されて再利用されます。障害ドメインが指定されていないクラスターの場合、すべてのコントロールプレーンマシンが単一の障害ドメイン内に配置されます。

障害ドメインの設定にいくつかの変更を加えると、コントロールプレーンマシンセットがコントロールプレーンマシンのバランスを再調整します。たとえば、障害ドメインがコントロールプレーンマシンよりも少ないクラスターに障害ドメインを追加すると、コントロールプレーンマシンセットは、使用可能なすべての障害ドメイン間でマシンのバランスを再調整します。

12.6.2. 障害が発生したコントロールプレーンマシンの復旧

Control Plane Machine Set Operator は、コントロールプレーンマシンの復旧を自動化します。コントロールプレーンマシンが削除されると、Operator は **ControlPlaneMachineSet** カスタムリソース (CR) で指定された設定で置換を作成します。

コントロールプレーンマシンセットを使用するクラスターの場合、マシンのヘルスチェックを設定できます。マシンのヘルスチェックでは、異常なコントロールプレーンマシンが削除され、置き換えられます。



重要

コントロールプレーンの **MachineHealthCheck** リソースを設定する場合は、**maxUnhealthy** の値を **1** に設定します。

この設定により、複数のコントロールプレーンマシンが異常であると思われる場合に、マシンのヘルスチェックがアクションを実行しないことが保証されます。複数の異常なコントロールプレーンマシンは、etcd クラスターが劣化していること、または障害が発生したマシンを置き換えるためのスケーリング操作が進行中であることを示している可能性があります。

etcd クラスターが劣化している場合は、手動での介入が必要になる場合があります。スケーリング操作が進行中の場合は、マシンのヘルスチェックで完了できるようにする必要があります。

関連情報

- マシンヘルスチェックのデプロイ

12.6.3. 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 クラスターの具体的な運用上の知識がなくても、etcd Operator によって 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、Machine API Operator によって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。

12.6.3.1. 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

コントロールプレーンマシンセットを使用する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ー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 1
      owner: clusteroperator/etcd 2
  ...
```

1 **preDrain** ライフサイクルフックの名前。

2 **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。

関連情報

- マシン削除フェーズのライフサイクルフック

12.7. コントロールプレーンマシンセット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。

12.7.1. コントロールプレーンマシンセットのカスタムリソースの状態を確認する

ControlPlaneMachineSet カスタムリソース (CR) の存在と状態を確認できます。

手順

- 次のコマンドを実行して、CR の状態を確認します。

```
$ oc get controlplanemachineset.machine.openshift.io cluster \
  --namespace openshift-machine-api
```

  - **Active** の結果は、**ControlPlaneMachineSet** CR が存在し、アクティブ化されていることを示します。管理者の操作は必要ありません。
  - **Inactive** の結果は、**ControlPlaneMachineSet** CR が存在するがアクティブ化されていないことを示します。
  - **NotFound** の結果は、既存の **ControlPlaneMachineSet** CR がないことを示します。

次のステップ

コントロールプレーンマシンセットを使用するには、クラスターの正しい設定を持つ **ControlPlaneMachineSet** CR が存在することを確認する必要があります。

- クラスターに既存の CR がある場合は、CR の設定がクラスターに対して正しいことを確認する必要があります。
- クラスターに既存の CR がない場合は、クラスターの正しい設定で CR を作成する必要があります。

関連情報

- コントロールプレーンマシンセットカスタムリソースの有効化
- コントロールプレーンマシンセットのカスタムリソースの作成

12.7.2. 不足している Azure 内部ロードバランサーの追加

Azure の **ControlPlaneMachineSet** とコントロールプレーン **Machine** のカスタムリソース (CR) の両方で **internalLoadBalancer** パラメーターが必要です。このパラメーターがクラスターで事前設定されていない場合は、両方の CR に追加する必要があります。

このパラメーターが Azure プロバイダー仕様のどこにあるかについて詳しくは、Azure プロバイダー仕様のサンプルを参照してください。コントロールプレーン **Machine** CR での配置も同様です。

手順

1. 次のコマンドを実行して、クラスター内のコントロールプレーンマシンを一覧表示します。

```
$ oc get machines \
  -l machine.openshift.io/cluster-api-machine-role==master \
  -n openshift-machine-api
```

2. コントロールプレーンマシンごとに、次のコマンドを実行して CR を編集します。

```
$ oc edit machine <control_plane_machine_name>
```

3. クラスターの正しい詳細を含む **internalLoadBalancer** パラメーターを追加し、変更を保存します。
4. 次のコマンドを実行して、コントロールプレーンマシンセットの CR を編集します。

```
$ oc edit controlplanemachineset.machine.openshift.io cluster \
  -n openshift-machine-api
```

5. クラスターの正しい詳細を含む **internalLoadBalancer** パラメーターを追加し、変更を保存します。

次のステップ

- デフォルトの **RollingUpdate** 更新戦略を使用するクラスターの場合、Operator は自動的に変更をコントロールプレーン設定に伝達します。
- **OnDelete** 更新戦略を使用するように設定されているクラスターの場合、コントロールプレーンマシンを手動で置き換える必要があります。

関連情報

- Microsoft Azure プロバイダー仕様のサンプル

12.7.3. 劣化した etcd Operator のリカバリー

特定の状況では、etcd Operator が劣化する可能性があります。

たとえば、修復の実行中に、マシンのヘルスチェックによって、etcd をホストしているコントロールプレーンマシンが削除される場合があります。その時点で etcd メンバーにアクセスできない場合、etcd Operator は劣化します。

etcd Operator が劣化している場合、Operator に障害のあるメンバーを強制的に削除させ、クラスターの状態を復元させるには、手動の介入が必要です。

手順

1. 次のコマンドを実行して、クラスター内のコントロールプレーンマシンを一覧表示します。

```
$ oc get machines \
  -l machine.openshift.io/cluster-api-machine-role==master \
  -n openshift-machine-api \
  -o wide
```

次のいずれかの状態は、コントロールプレーンマシンの障害を示している可能性があります。

- **STATE** 値は **stopped** です。
- **PHASE** 値は **Failed** です。
- **PHASE** 値が 10 分以上 **Deleting** です。

重要

続行する前に、クラスターに 2 つの正常なコントロールプレーンマシンがあることを確認します。この手順のアクションを複数のコントロールプレーンマシンで実行すると、etcd クォーラムが失われるリスクがあり、データが失われる可能性があります。

大多数のコントロールプレーンホストが失われ、etcd のクォーラム (定足数) の損失が発生した場合は、この手順ではなく、「直前のクラスター状態への復元に向けた障害復旧」手順を実行する必要があります。

2. 次のコマンドを実行して、障害が発生したコントロールプレーンマシンのマシン CR を編集します。

```
$ oc edit machine <control_plane_machine_name>
```

3. 障害が発生したコントロールプレーンマシンから **lifecycleHooks** パラメーターの内容を削除し、変更を保存します。
etcd Operator は、障害が発生したマシンをクラスターから削除し、新しい etcd メンバーを安全に追加できるようにします。

関連情報

- クラスターの直前の状態への復元

12.7.4. RHOSP 上で実行されるクラスターのアップグレード

OpenShift Container Platform 4.13 以前で作成された Red Hat OpenStack Platform (RHOSP) 上で実行されるクラスターの場合は、コントロールプレーンマシンセットを使用する前にアップグレード後のタスクを実行する必要がある場合があります。

12.7.4.1. アップグレード後のルートボリュームアベイラビリティーゾーンを持つマシンを含む RHOSP クラスターの設定

アップグレードする Red Hat OpenStack Platform (RHOSP) 上で実行する一部のクラスターでは、次の設定に該当する場合、コントロールプレーンマシンセットを使用する前に、マシンリソースを手動で更新する必要があります。

- アップグレードされたクラスターは、OpenShift Container Platform 4.13 以前で作成されました。
- クラスターインフラストラクチャーはインストーラーによってプロビジョニングされます。
- マシンは複数のアベイラビリティーゾーンに分散されました。
- マシンは、ブロックストレージのアベイラビリティーゾーンが定義されていないルートボリュームを使用するように設定されていました。

この手順が必要な理由を理解するには、Solution #7024383 を参照してください。

手順

1. すべてのコントロールプレーンマシンについて、環境に一致するすべてのコントロールプレーンマシンのプロバイダー仕様を編集します。たとえば、マシン **master-0** を編集するには、次のコマンドを入力します。

   ```
   $ oc edit machine/<cluster_id>-master-0 -n openshift-machine-api
   ```

   ここでは、以下のようになります。

   <cluster_id>
   アップグレードされたクラスターの ID を指定します。

2. プロバイダー仕様で、プロパティー **rootVolume.availabilityZone** の値を、使用するアベイラビリティーゾーンのボリュームに設定します。

   RHOSP プロバイダー仕様の例

   ```
   providerSpec:
     value:
       apiVersion: machine.openshift.io/v1alpha1
       availabilityZone: az0
         cloudName: openstack
       cloudsSecret:
         name: openstack-cloud-credentials
         namespace: openshift-machine-api
       flavor: m1.xlarge
       image: rhcos-4.14
       kind: OpenstackProviderSpec
       metadata:
         creationTimestamp: null
       networks:
       - filter: {}
         subnets:
         - filter:
             name: refarch-lv7q9-nodes
             tags: openshiftClusterID=refarch-lv7q9
   ```

```
    rootVolume:
      availabilityZone: nova 1
      diskSize: 30
      sourceUUID: rhcos-4.12
      volumeType: fast-0
  securityGroups:
  - filter: {}
    name: refarch-lv7q9-master
  serverGroupName: refarch-lv7q9-master
  serverMetadata:
    Name: refarch-lv7q9-master
    openshiftClusterID: refarch-lv7q9
  tags:
  - openshiftClusterID=refarch-lv7q9
  trunk: true
  userDataSecret:
    name: master-user-data
```

1 ゾーン名をこの値として設定します。

注記

最初のクラスターデプロイメント後にマシンリソースを編集または再作成した場合は、これらの手順を設定に合わせて調整が必要になる場合があります。

RHOSP クラスターで、マシンのルートボリュームのアベイラビリティーゾーンを見つけて、それを値として使用します。

3. 次のコマンドを実行して、コントロールプレーンマシンセットリソースに関する情報を取得します。

```
$ oc describe controlplanemachineset.machine.openshift.io/cluster --namespace openshift-machine-api
```

4. 以下のコマンドを実行してリソースを編集します。

```
$ oc edit controlplanemachineset.machine.openshift.io/cluster --namespace openshift-machine-api
```

5. そのリソースについて、**spec.state** プロパティーの値を **Active** に設定して、クラスターのコントロールプレーンマシンセットをアクティブにします。

コントロールプレーンは、Cluster Control Plane Machine Set Operator によって管理される準備が整いました。

12.7.4.2. アップグレード後のアベイラビリティーゾーンを備えたコントロールプレーンマシンを含む RHOSP クラスターの設定

アップグレードする Red Hat OpenStack Platform (RHOSP) 上で実行する一部のクラスターでは、次の設定に該当する場合、コントロールプレーンマシンセットを使用する前に、マシンリソースを手動で更新する必要があります。

- アップグレードされたクラスターは、OpenShift Container Platform 4.13 以前で作成されました。
- クラスターインフラストラクチャーはインストーラーによってプロビジョニングされます。
- コントロールプレーンマシンは、複数のコンピュートアベイラビリティーゾーンに分散されました。

この手順が必要な理由を理解するには、Solution #7013893 を参照してください。

手順

1. **master-1** および **master-2** コントロールプレーンマシンの場合は、プロバイダー仕様を開いて編集します。たとえば、最初のマシンを編集するには、次のコマンドを入力します。

```
$ oc edit machine/<cluster_id>-master-1 -n openshift-machine-api
```

   ここでは、以下のようになります。

   <cluster_id>
   アップグレードされたクラスターの ID を指定します。

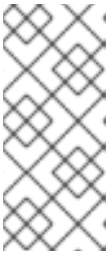
2. **master-1** および **master-2** コントロールプレーンマシンの場合は、プロバイダー仕様の **serverGroupName** プロパティーの値を編集して、マシン **master-0** の値と一致させます。

   RHOSP プロバイダー仕様の例

```
providerSpec:
  value:
    apiVersion: machine.openshift.io/v1alpha1
    availabilityZone: az0
      cloudName: openstack
    cloudsSecret:
      name: openstack-cloud-credentials
      namespace: openshift-machine-api
    flavor: m1.xlarge
    image: rhcos-4.16
    kind: OpenstackProviderSpec
    metadata:
      creationTimestamp: null
    networks:
    - filter: {}
      subnets:
      - filter:
          name: refarch-lv7q9-nodes
          tags: openshiftClusterID=refarch-lv7q9
    securityGroups:
    - filter: {}
      name: refarch-lv7q9-master
    serverGroupName: refarch-lv7q9-master-az0 (1)
    serverMetadata:
      Name: refarch-lv7q9-master
      openshiftClusterID: refarch-lv7q9
    tags:
    - openshiftClusterID=refarch-lv7q9
```

```
    trunk: true
    userDataSecret:
      name: master-user-data
```

1 この値は、マシン **master-0**、**master-1**、および **master-3** と一致する必要があります。



注記

最初のクラスターデプロイメント後にマシンリソースを編集または再作成した場合は、これらの手順を設定に合わせて調整が必要になる場合があります。

RHOSP クラスターで、コントロールプレーンインスタンスが含まれるサーバーグループを見つけて、それを値として使用します。

3. 次のコマンドを実行して、コントロールプレーンマシンセットリソースに関する情報を取得します。

```
$ oc describe controlplanemachineset.machine.openshift.io/cluster --namespace openshift-machine-api
```

4. 以下のコマンドを実行してリソースを編集します。

```
$ oc edit controlplanemachineset.machine.openshift.io/cluster --namespace openshift-machine-api
```

5. そのリソースについて、**spec.state** プロパティーの値を **Active** に設定して、クラスターのコントロールプレーンマシンセットをアクティブにします。

コントロールプレーンは、Cluster Control Plane Machine Set Operator によって管理される準備が整いました。

12.8. コントロールプレーンマシンセットの無効化

アクティブ化された **ControlPlaneMachineSet** カスタムリソース (CR) の **.spec.state** フィールドは、**Active** から **Inactive** に変更できません。コントロールプレーンマシンセットを無効にするには、CR を削除してクラスターから削除する必要があります。

CR を削除すると、Control Plane Machine Set Operator がクリーンアップ操作を実行し、コントロールプレーンマシンセットを無効にします。その後、Operator はクラスターから CR を削除し、デフォルト設定で非アクティブなコントロールプレーンマシンセットを作成します。

12.8.1. コントロールプレーンマシンセットの削除

クラスター上のコントロールプレーンマシンセットによるコントロールプレーンマシンの管理を停止するには、**ControlPlaneMachineSet** カスタムリソース (CR) を削除する必要があります。

手順

- 次のコマンドを実行して、コントロールプレーンマシンセット CR を削除します。

```
$ oc delete controlplanemachineset.machine.openshift.io cluster \
  -n openshift-machine-api
```

検証

- コントロールプレーンマシンセットのカスタムリソースの状態を確認します。**Inactive** の結果は、取り外しと交換のプロセスが成功したことを示します。**ControlPlaneMachineSet** CR は存在しますが、アクティブ化されていません。

12.8.2. コントロールプレーンマシンセットのカスタムリソースの状態を確認する

ControlPlaneMachineSet カスタムリソース (CR) の存在と状態を確認できます。

手順

- 次のコマンドを実行して、CR の状態を確認します。

```
$ oc get controlplanemachineset.machine.openshift.io cluster \
  --namespace openshift-machine-api
```

  - **Active** の結果は、**ControlPlaneMachineSet** CR が存在し、アクティブ化されていることを示します。管理者の操作は必要ありません。
  - **Inactive** の結果は、**ControlPlaneMachineSet** CR が存在するがアクティブ化されていないことを示します。
  - **NotFound** の結果は、既存の **ControlPlaneMachineSet** CR がないことを示します。

12.8.3. コントロールプレーンマシンセットの再有効化

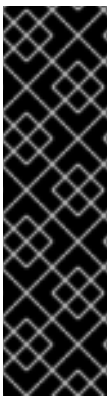
コントロールプレーンマシンセットを再度有効にするには、CR の設定がクラスターに対して正しいことを確認し、アクティブ化する必要があります。

関連情報

- コントロールプレーンマシンセットカスタムリソースの有効化

第13章 CLUSTER API によるマシンの管理

13.1. CLUSTER API について



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

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

Cluster API は、Amazon Web Services (AWS)、Google Cloud Platform (GCP)、VMware vSphere のテクノロジープレビューとして OpenShift Container Platform に統合されているアップストリームプロジェクトです。

13.1.1. Cluster API の概要

Cluster API を使用すると、OpenShift Container Platform クラスター内のコンピュートマシンセットとコンピュート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 4.16 クラスターの場合、Cluster 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トマシンおよびコンピュート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。

13.1.1.1. Cluster API の利点

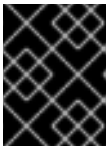
Cluster API を使用すると、OpenShift Container Platform のユーザーと開発者は次の利点を得ることができます。

- Machine API でサポートされていない可能性のあるアップストリームコミュニティーの Cluster API インフラストラクチャープロバイダーを使用する選択肢。
- 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 を使用してコンピュートマシンセットを作成する機能。

13.1.1.2. Cluster API の制限

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。

- 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。



重要

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。

- Cluster API を使用できるのは、Amazon Web Services (AWS)、Google Cloud Platform (GCP)、および VMware vSphere クラスターのみです。
- Cluster API に必要な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。詳細は、「Cluster API の使用開始」を参照してください。
- Cluster API を使用してコントロールプレーンマシンを管理することはできません。
- Machine API によって作成された既存のマシンセットの、Cluster API コンピュート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ーは利用できません。
- Cluster API を使用するクラスターの場合、OpenShift CLI (**oc**) コマンドで、Machine API オブジェクトよりも Cluster API オブジェクトが優先されます。この動作は、Cluster API と Machine API の両方で表されるオブジェクトに対して動作する **oc** コマンドに影響します。この問題の詳細と回避策については、トラブルシューティングコンテンツの「CLI 使用時に目的のオブジェクトを参照する」を参照してください。

関連情報

- フィーチャーゲートを使用した機能の有効化
- Cluster API の使用開始
- CLI 使用時に目的のオブジェクトを参照する

13.1.2. 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。

13.1.2.1. 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 コンポーネントをクラスターにインストールします。

詳細は、**クラスター Operator リファレンス** の「クラスター CAPI Operator」を参照してください。

関連情報

- Cluster CAPI Operator

13.1.2.2. Cluster API の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構成されています。この機能のテクノロジープレビュー版では、**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コンピュートマシンセットで共有されるプロパティー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

コンピュートマシンセットが作成するマシンのプロパティー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

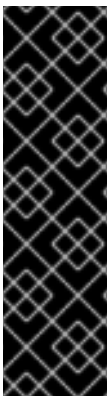
マシンのグループ。
コンピュートマシンセットは、レプリカセットと Pod の関係としてマシンに対して行われます。マシンを追加したりスケールダウンしたりするには、コンピュートマシンの **replicas** フィールドを変更して、コンピュートニーズに合わせてカスタムリソースを設定します。

Cluster API を使用すると、コンピュートマシンセットは **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。

13.2. CLUSTER API の使用開始



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

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

テクノロジープレビューの Cluster API では、Cluster API に必要な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。

13.2.1. Cluster API のプライマリーリソースの作成

Cluster API のプライマリーリソースを作成するには、クラスター ID 値を取得する必要があります。この値は、クラスターリソースマニフェストの **<cluster_name>** パラメーターに使用します。

13.2.1.1. クラスター ID 値の取得

クラスター ID 値は、OpenShift CLI (**oc**) を使用して確認できます。

前提条件

- OpenShift Container Platform クラスターをデプロイした。
- **cluster-admin** 権限を持つアカウントを使用してクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。

手順

- 次のコマンドを実行して、クラスター ID 値を取得します。

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

YAML マニフェストファイルを作成し、OpenShift CLI (**oc**) を使用して適用することで、Cluster API のプライマリーリソースを手動で作成できます。

13.2.1.2. Cluster API のクラスターリソースの作成

YAML マニフェストファイルを作成し、それを OpenShift CLI (**oc**) で適用することで、クラスターリソースを作成できます。

前提条件

- OpenShift Container Platform クラスターをデプロイした。
- Cluster API の使用を有効にした。
- **cluster-admin** 権限を持つアカウントを使用してクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。
- クラスター ID 値を取得した。

手順

1. 以下のような YAML ファイルを作成します。この手順では、ファイル名の例として **<cluster_resource_file>.yaml** を使用します。

```
apiVersion: cluster.x-k8s.io/v1beta1
kind: Cluster
metadata:
  name: <cluster_name> 1
  namespace: openshift-cluster-api
spec:
  infrastructureRef:
    apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
```

```
kind: <infrastructure_kind> 2
name: <cluster_name>
namespace: openshift-cluster-api
```

1 クラスターの名前としてクラスター ID を指定します。

2 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。次の値が有効です。

- **AWSCluster**: クラスターが Amazon Web Services (AWS) で実行されている場合。
- **GCPCluster**: クラスターが Google Cloud Platform (GCP) で実行されている場合。
- **VSphereCluster**: クラスターが VMware vSphere 上で実行されている場合。

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

- 次のコマンドを実行して、クラスター CR が存在することを確認します。

```
$ oc get cluster
```

出力例

```
NAME           PHASE          AGE    VERSION
<cluster_name>   Provisioning   4h6m
```

PHASE の値が **Provisioned** の場合、クラスターリソースの準備が完了しています。

関連情報

- Cluster API の設定

13.2.1.3. Cluster API のインフラストラクチャーリソースの作成

YAML マニフェストファイルを作成し、それを OpenShift CLI (**oc**) で適用することで、プロバイダー固有のインフラストラクチャーリソースを作成できます。

前提条件

- OpenShift Container Platform クラスターをデプロイした。
- Cluster API の使用を有効にした。
- **cluster-admin** 権限を持つアカウントを使用してクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。
- クラスター ID 値を取得した。
- クラスターリソースを作成して適用した。

手順

1. 以下のような YAML ファイルを作成します。この手順では、ファイル名の例として **<infrastructure_resource_file>.yaml** を使用します。

```
apiVersion: infrastructure.cluster.x-k8s.io/<version> (1)
kind: <infrastructure_kind> (2)
metadata:
  name: <cluster_name> (3)
  namespace: openshift-cluster-api
spec: (4)
```

(1) **apiVersion** はプラットフォームによって異なります。詳細は、各プロバイダーの Cluster API インフラストラクチャーリソースのサンプル YAML を参照してください。次の値が有効です。

- **infrastructure.cluster.x-k8s.io/v1beta2**: Amazon Web Services (AWS) クラスターが使用するバージョン。
- **Infrastructure.cluster.x-k8s.io/v1beta1**: Google Cloud Platform (GCP) および VMware vSphere クラスターが使用するバージョン。

(2) 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。次の値が有効です。

- **AWSCluster**: クラスターが AWS 上で実行されている場合。
- **GCPCluster**: クラスターが GCP 上で実行されている場合。
- **VSphereCluster**: クラスターが vSphere で実行されている場合。

(3) クラスターの名前を指定します。

(4) 環境の詳細を指定します。これらのパラメーターはプロバイダー固有です。詳細は、各プロバイダーの Cluster API インフラストラクチャーリソースのサンプル YAML を参照してください。

2. 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

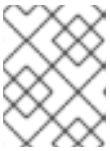
- 次のコマンドを実行して、インフラストラクチャー CR が作成されたことを確認します。

```
$ 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
<cluster_name>   <cluster_name>   true
```



注記

この出力には、クラウドプロバイダーに固有の追加の列が含まれている場合があります。

関連情報

- 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
- 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
- VMware vSphere 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

13.2.1.4. Cluster API のマシンテンプレートの作成

YAML マニフェストファイルを作成し、それを OpenShift CLI (**oc**) で適用することで、プロバイダー固有のマシンテンプレートリソースを作成できます。

前提条件

- OpenShift Container Platform クラスターをデプロイした。
- Cluster API の使用を有効にした。
- **cluster-admin** 権限を持つアカウントを使用してクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。
- クラスターとインフラストラクチャーリソースを作成して適用した。

手順

1. 以下のような YAML ファイルを作成します。この手順では、ファイル名の例として **<machine_template_resource_file>.yaml** を使用します。

```
apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: <machine_template_kind> 1
metadata:
  name: <template_name> 2
  namespace: openshift-cluster-api
spec:
  template:
    spec: 3
```

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。次の値が有効です。

   - **AWSMachineTemplate**: クラスターが Amazon Web Services (AWS) 上で実行されている場合。
   - **GCPMachineTemplate**: クラスターが Google Cloud Platform (GCP) 上で実行されている場合。
   - **VSphereMachineTemplate**: クラスターが VMware vSphere で実行されている場合。

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。これらのパラメーターはプロバイダー固有です。詳細は、各プロバイダーの Cluster API マシンテンプレートのサンプル YAML を参照してください。

2. 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

- 次のコマンドを実行して、マシンテンプレート CR が作成されたことを確認します。

```
$ 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
```

関連情報

- 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
- 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
- VMware vSphere 上の Cluster API マシンテンプレートリソースのサンプル YAML

13.2.1.5. Cluster API のコンピュート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ト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した。
- Cluster API の使用を有効にした。
- **cluster-admin** 権限を持つアカウントを使用してクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。
- クラスターリソース、インフラストラクチャーリソース、およびマシンテンプレートリソースを作成した。

手順

1. 以下のような YAML ファイルを作成します。この手順では、ファイル名の例として **<machine_set_resource_file>.yaml** を使用します。

```
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: example
  template:
    metadata:
      labels:
        test: example
    spec: 3
# ...
```

1. コンピュートマシンセットの名前を指定します。
2. クラスターの名前を指定します。
3. 環境の詳細を指定します。これらのパラメーターはプロバイダー固有です。詳細は、各プロバイダーの Cluster API コンピュートマシンセットのサンプル YAML を参照してください。

2. 次のコンピュート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
```

3. 次のコマンドを実行して、コンピュートマシンセット CR が作成されたことを確認します。

```
$ oc get machineset -n openshift-cluster-api 1
```

1. **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
```

新しいコンピュート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。コンピュートマシンセットが使用できない場合は、数分待ってからコマンドを再実行してください。

検証

- コンピュートマシンセットが必要な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、次のコマンドを実行して、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します。

```
$ oc get machine -n openshift-cluster-api 1
```

1 **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します。

```
$ oc get node
```

出力例

```
NAME                                STATUS   ROLES    AGE     VERSION
<ip_address_1>.<region>.compute.internal   Ready    worker   5h14m   v1.28.5
<ip_address_2>.<region>.compute.internal   Ready    master   5h19m   v1.28.5
<ip_address_3>.<region>.compute.internal   Ready    worker   7m      v1.28.5
```

関連情報

- Amazon Web Services の Cluster API コンピューティングマシンセットリソースのサンプル YAML
- Google Cloud Platform 上の Cluster API コンピュートマシンセットリソースのサンプル YAML
- VMware vSphere 上の Cluster API コンピュートマシンセットリソースのサンプル YAML

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

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

13.3.1. Cluster API のマシンテンプレートの変更

YAML マニフェストファイルを変更し、OpenShift CLI (**oc**) を使用して適用することで、クラスターのマシンテンプレートリソースを更新できます。

前提条件

- Cluster API を使用する OpenShift Container Platform クラスターをデプロイした。
- **cluster-admin** 権限を持つアカウントを使用してクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. 次のコマンドを実行して、クラスターのマシンテンプレートリソースをリスト表示します。

```
$ oc get <machine_template_kind> 1
```

1 お使いのプラットフォームに応じた値を指定します。次の値が有効です。

   - **AWSMachineTemplate**: クラスターが Amazon Web Services (AWS) 上で実行されている場合。
   - **GCPMachineTemplate**: クラスターが Google Cloud Platform (GCP) 上で実行されている場合。
   - **VSphereMachineTemplate**: クラスターが VMware vSphere で実行されている場合。

出力例

```
NAME            AGE
<template_name>   77m
```

2. 次のコマンドを実行して、クラスターのマシンテンプレートリソースを編集可能なファイルに書き込みます。

```
$ oc get <template_name> -o yaml > <template_name>.yaml
```

<template_name> は、クラスターのマシンテンプレートリソースの名前です。

3. **<template_name>.yaml** ファイルのコピーを作成し、別の名前を付けます。この手順では、ファイル名の例として **<modified_template_name>.yaml** を使用します。

4. テキストエディターを使用し、**<modified_template_name>.yaml** ファイルに変更を加えて、クラスターの更新されたマシンテンプレートリソースを定義します。マシンテンプレートリソースを編集するときは、次の点に注意してください。

   - **spec** スタンザ内のパラメーターはプロバイダー固有です。詳細は、各プロバイダーの Cluster API マシンテンプレートのサンプル YAML を参照してください。
   - **metadata.name** パラメーターには、既存の値とは異なる値を使用する必要があります。

重要

このテンプレートを参照する Cluster API コンピュートマシンセットがある場合、**spec.template.spec.infrastructureRef.name** パラメーターを、新しいマシンテンプレートリソースの **metadata.name** 値と一致するように更新する必要があります。

5. 次のコマンドを実行して、マシンテンプレート CR を適用します。

```
$ oc apply -f <modified_template_name>.yaml 1
```

1 新しい名前を付けた編集済み YAML ファイルを使用します。

次のステップ

- このテンプレートを参照する Cluster API コンピュートマシンセットがある場合、**spec.template.spec.infrastructureRef.name** パラメーターを、新しいマシンテンプレートリソースの **metadata.name** 値と一致するように更新します。詳細は、「CLI を使用してコンピュートマシンセットを変更する」を参照してください。

関連情報

- 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
- 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
- VMware vSphere 上の Cluster API マシンテンプレートリソースのサンプル YAML
- CLI を使用してコンピュートマシンセットを変更する

13.3.2. CLI を使用してコンピュートマシンセットを変更する

コンピュートマシンセットを変更すると、その変更は、更新された **MachineSet** カスタムリソース (CR) を保存した後に作成されたコンピュートマシンにのみ適用されます。この変更は既存のマシンには影響しません。コンピュートマシンセットをスケーリングすることで、既存のマシンを、更新された設定を反映した新しいマシンに置き換えることができます。

他の変更を加えずに、コンピュート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

デフォルトでは、OpenShift Container Platform ルーター Pod はコンピュートマシン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、コンピュートマシンセットを **0** にスケーリングできません。

この手順の出力例では、AWS クラスターの値を使用します。

前提条件

- OpenShift Container Platform クラスターが Cluster API を使用している。
- OpenShift CLI (**oc**) を使用して、管理者としてクラスターにログインしている。

手順

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

```
$ oc get machinesets.cluster.x-k8s.io -n openshift-cluster-api
```

出力例

```
NAME                        CLUSTER          REPLICAS   READY   AVAILABLE   AGE
VERSION
<compute_machine_set_name_1>  <cluster_name>     1        1       1         26m
<compute_machine_set_name_2>  <cluster_name>     1        1       1         26m
```

2. 次のコマンドを実行して、コンピュートマシンセットを編集します。

```
$ oc edit machinesets.cluster.x-k8s.io <machine_set_name> \
  -n openshift-cluster-api
```

3. **spec.replicas** フィールドの値をメモします。この値は、変更を適用するためにコンピュートマシンセットをスケーリングする際に必要になるためです。

```
apiVersion: cluster.x-k8s.io/v1beta1
kind: MachineSet
metadata:
  name: <machine_set_name>
  namespace: openshift-cluster-api
spec:
  replicas: 2 (1)
# ...
```

(1) この手順例では、**replicas** 値が **2** のコンピュートマシンセットを示しています。

4. 必要な設定オプションを使用してコンピュートマシンセット CR を更新し、変更を保存します。

5. 次のコマンドを実行して、更新されたコンピュートマシンセットによって管理されているマシンをリスト表示します。

```
$ oc get machines.cluster.x-k8s.io \
  -n openshift-cluster-api \
  -l cluster.x-k8s.io/set-name=<machine_set_name>
```

AWS クラスターの出力例

```
NAME                      CLUSTER        NODENAME                                PROVIDERID
PHASE         AGE     VERSION
<machine_name_original_1>  <cluster_name>  <original_1_ip>.<region>.compute.internal
aws:///us-east-2a/i-04e7b2cbd61fd2075  Running       4h
<machine_name_original_2>  <cluster_name>  <original_2_ip>.<region>.compute.internal
aws:///us-east-2a/i-04e7b2cbd61fd2075  Running       4h
```

6. 次のコマンドを実行して、更新されたコンピュートマシンセットで管理されるマシンごとに **delete** アノテーションを設定します。

```
$ oc annotate machines.cluster.x-k8s.io/<machine_name_original_1> \
  -n openshift-cluster-api \
  cluster.x-k8s.io/delete-machine="true"
```

7. 代わりとなるマシンを新しい設定で作成するために、次のコマンドを実行して、コンピュートマシンセットをレプリカ数の 2 倍にスケーリングします。

```
$ oc scale --replicas=4 \ 1
  machinesets.cluster.x-k8s.io <machine_set_name> \
  -n openshift-cluster-api
```

1 元の例の値 **2** は 2 倍の **4** になります。

8. 次のコマンドを実行して、更新されたコンピュートマシンセットによって管理されているマシンをリスト表示します。

```
$ oc get machines.cluster.x-k8s.io \
  -n openshift-cluster-api \
  -l cluster.x-k8s.io/set-name=<machine_set_name>
```

AWS クラスターの出力例

```
NAME                      CLUSTER          NODENAME                             PROVIDERID
PHASE         AGE     VERSION
<machine_name_original_1>   <cluster_name>   <original_1_ip>.<region>.compute.internal
aws:///us-east-2a/i-04e7b2cbd61fd2075   Running         4h
<machine_name_original_2>   <cluster_name>   <original_2_ip>.<region>.compute.internal
aws:///us-east-2a/i-04e7b2cbd61fd2075   Running         4h
<machine_name_updated_1>    <cluster_name>   <updated_1_ip>.
<region>.compute.internal    aws:///us-east-2a/i-04e7b2cbd61fd2075   Provisioned     55s
<machine_name_updated_2>    <cluster_name>   <updated_2_ip>.
<region>.compute.internal    aws:///us-east-2a/i-04e7b2cbd61fd2075   Provisioning    55s
```

新しいマシンが **Running** フェーズにある場合、コンピュートマシンセットを元のレプリカ数にスケーリングできます。

9. 古い設定で作成されたマシンを削除するために、次のコマンドを実行して、コンピュートマシンセットを元のレプリカ数にスケーリングします。

```
$ oc scale --replicas=2 \ 1
  machinesets.cluster.x-k8s.io <machine_set_name> \
  -n openshift-cluster-api
```

1 元の例の値は **2** です。

検証

- 更新されたマシンセットによって作成されたマシンの設定が正しいことを確認するには、次のコマンドを実行して、新しいマシンの 1 つで CR の関連フィールドを調べます。

```
$ oc describe machines.cluster.x-k8s.io <machine_name_updated_1> \
  -n openshift-cluster-api
```

- 設定が更新されていないコンピュートマシンが削除されたことを確認するには、次のコマンドを実行して、更新されたコンピュートマシンセットによって管理されているマシンをリスト表示します。

```
$ oc get machines.cluster.x-k8s.io \
  -n openshift-cluster-api \
  cluster.x-k8s.io/set-name=<machine_set_name>
```

AWS クラスターの削除進行中の出力例

```
NAME                    CLUSTER         NODENAME                                PROVIDERID
PHASE     AGE     VERSION
<machine_name_original_1>  <cluster_name>  <original_1_ip>.<region>.compute.internal
aws:///us-east-2a/i-04e7b2cbd61fd2075  Running  18m
<machine_name_original_2>  <cluster_name>  <original_2_ip>.<region>.compute.internal
aws:///us-east-2a/i-04e7b2cbd61fd2075  Running  18m
<machine_name_updated_1>  <cluster_name>  <updated_1_ip>.
<region>.compute.internal  aws:///us-east-2a/i-04e7b2cbd61fd2075  Running  18m
<machine_name_updated_2>  <cluster_name>  <updated_2_ip>.
<region>.compute.internal  aws:///us-east-2a/i-04e7b2cbd61fd2075  Running  18m
```

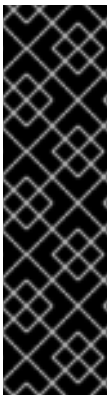
AWS クラスターの削除完了時の出力例

```
NAME                    CLUSTER         NODENAME                                PROVIDERID
PHASE     AGE     VERSION
<machine_name_updated_1>  <cluster_name>  <updated_1_ip>.
<region>.compute.internal  aws:///us-east-2a/i-04e7b2cbd61fd2075  Running  18m
<machine_name_updated_2>  <cluster_name>  <updated_2_ip>.
<region>.compute.internal  aws:///us-east-2a/i-04e7b2cbd61fd2075  Running  18m
```

関連情報

- Amazon Web Services の Cluster API コンピューティングマシンセットリソースのサンプル YAML
- Google Cloud Platform 上の Cluster API コンピュートマシンセットリソースのサンプル YAML
- VMware vSphere 上の Cluster API コンピュートマシンセットリソースのサンプル YAML

13.4. CLUSTER API の設定



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

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

次の YAML ファイルの例は、Cluster API のプライマリーリソースを連携させ、リソースによって作成されるマシンの設定を環境に合わせて適切に設定する方法を示しています。

13.4.1. 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ー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> 1
  namespace: openshift-cluster-api
spec:
  controlPlaneEndpoint: 2
    host: <control_plane_endpoint_address>
    port: 6443
  infrastructureRef:
    apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
    kind: <infrastructure_kind> 3
    name: <cluster_name>
    namespace: openshift-cluster-api
```

1. クラスターの名前を指定します。
2. コントロールプレーンエンドポイントの IP アドレスと、それにアクセスするために使用するポートを指定します。
3. 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**: クラスターが Amazon Web Services で実行されている場合。
   - **GCPCluster**: クラスターが Google Cloud Platform で実行されている場合。
   - **VSphereCluster**: クラスターが VMware vSphere 上で実行されている場合。

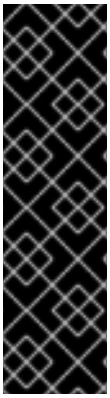
13.4.2. プロバイダー固有の設定オプション

残りはプロバイダー固有の Cluster API リソースです。クラスターのプロバイダー固有の設定オプションについては、次のリソースを参照してください。

- Amazon Web Services の Cluster API 設定オプション
- Google Cloud Platform の Cluster API 設定オプション
- VMware vSphere のクラスター API 設定オプション

13.5. CLUSTER API マシンの設定オプション

13.5.1. Amazon Web Services の Cluster API 設定オプション



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

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

Cluster API カスタムリソースマニフェストの値を更新することで、Amazon Web Services (AWS) Cluster API マシンの設定を変更できます。

13.5.1.1. Amazon Web Services クラスターを設定するサンプル YAML

次の YAML ファイルの例は、Amazon Web Services クラスターの設定を示しています。

13.5.1.1.1. 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コンピュートマシンセットで共有されるプロパティーを定義します。コンピュートマシンセットは、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/v1beta2
kind: AWSCluster (1)
metadata:
  name: <cluster_name> (2)
  namespace: openshift-cluster-api
spec:
  controlPlaneEndpoint: (3)
    host: <control_plane_endpoint_address>
    port: 6443
  region: <region> (4)
```

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. クラスターの名前としてクラスター ID を指定します。
3. コントロールプレーンエンドポイントのアドレスと、それにアクセスするために使用するポートを指定します。
4. AWS リージョンを指定します。

13.5.1.1.2. 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、コンピュートマシンセットが作成するマシンの基本的なプロパティーを定義します。コンピュートマシンセットは、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/v1beta2
```

```
kind: AWSMachineTemplate (1)
metadata:
  name: <template_name> (2)
  namespace: openshift-cluster-api
spec:
  template:
    spec: (3)
      uncompressedUserData: true
      iamInstanceProfile: # ...
      instanceType: m5.large
      ignition:
        storageType: UnencryptedUserData
        version: "3.2"
      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。

13.5.1.1.3. Amazon Web Services の Cluster API コンピューティングマシンセットリソースのサンプル YAML

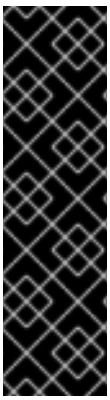
コンピュートマシンセットリソースは、作成するマシンの追加プロパティーを定義します。コンピュートマシンセットは、マシン作成時にインフラストラクチャーリソースとマシンテンプレートも参照します。

```
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: example
  template:
    metadata:
      labels:
```

```
      test: example
  spec:
    bootstrap:
      dataSecretName: worker-user-data 3
    clusterName: <cluster_name>
    infrastructureRef:
      apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
      kind: AWSMachineTemplate 4
      name: <template_name> 5
```

1. コンピュートマシンセットの名前を指定します。
2. クラスターの名前としてクラスター ID を指定します。
3. テクノロジープレビューの Cluster API では、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。
4. 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。
5. マシンテンプレート名を指定します。

13.5.2. Google Cloud Platform の Cluster API 設定オプション



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

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

Cluster API カスタムリソースマニフェストの値を更新することで、Google Cloud Platform (GCP) Cluster API マシンの設定を変更できます。

13.5.2.1. Google Cloud Platform クラスターを設定するためのサンプル YAML

次の YAML ファイルの例は、Google Cloud Platform クラスターの設定を示しています。

13.5.2.1.1. 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

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コンピュートマシンセットで共有されるプロパティーを定義します。コンピュートマシンセットは、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 1
metadata:
  name: <cluster_name> 2
```

```
spec:
  controlPlaneEndpoint: 3
    host: <control_plane_endpoint_address>
    port: 6443
  network:
    name: <cluster_name>-network
  project: <project> 4
  region: <region> 5
```

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 クラスターの名前としてクラスター ID を指定します。

3 コントロールプレーンエンドポイントの IP アドレスと、それにアクセスするために使用するポートを指定します。

4 GCP プロジェクト名を指定します。

5 GCP リージョンを指定します。

13.5.2.1.2. 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、コンピュートマシンセットが作成するマシンの基本的なプロパティーを定義します。コンピュートマシンセットは、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 1
metadata:
  name: <template_name> 2
  namespace: openshift-cluster-api
spec:
  template:
    spec: 3
      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。

13.5.2.1.3. Google Cloud Platform 上の Cluster API コンピュートマシンセットリソースのサンプル YAML

コンピュート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ーを定義します。コンピュートマシンセットは、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: example
  template:
    metadata:
      labels:
        test: example
    spec:
      bootstrap:
        dataSecretName: worker-user-data 3
      clusterName: <cluster_name>
      infrastructureRef:
        apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
        kind: GCPMachineTemplate 4
        name: <template_name> 5
      failureDomain: <failure_domain> 6
```

1 コンピュートマシンセットの名前を指定します。

2 クラスターの名前としてクラスター ID を指定します。

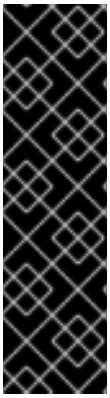
3 テクノロジープレビューの Cluster API では、Operator は **openshift-machine-api** namespace のワーカーユーザーデータシークレットを使用できます。

4 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。

5 マシンテンプレート名を指定します。

6 GCP リージョン内の障害ドメインを指定します。

13.5.3. VMware vSphere のクラスター API 設定オプション



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

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

Cluster API カスタムリソースマニフェストの値を更新することで、VMware vSphere Cluster API マシンの設定を変更できます。

13.5.3.1. VMware vSphere クラスターを設定するためのサンプル YAML

次の YAML ファイルの例は、VMware vSphere クラスターの設定を示しています。

13.5.3.1.1. VMware vSphere 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コンピュートマシンセットで共有されるプロパティーを定義します。コンピュートマシンセットは、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: VSphereCluster 1
metadata:
  name: <cluster_name> 2
spec:
  controlPlaneEndpoint: 3
    host: <control_plane_endpoint_address>
    port: 6443
  identityRef:
    kind: Secret
    name: <cluster_name>
  server: <vsphere_server> 4
```

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. クラスターの名前としてクラスター ID を指定します。
3. コントロールプレーンエンドポイントの IP アドレスと、それにアクセスするために使用するポートを指定します。
4. クラスターの vSphere サーバーを指定します。次のコマンドを実行すると、既存の vSphere クラスターでこの値を見つけることができます。

```
$ oc get infrastructure cluster \
  -o jsonpath="{.spec.platformSpec.vsphere.vcenters[0].server}"
```

13.5.3.1.2. VMware vSphere 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、コンピュートマシンセットが作成するマシンの基本的なプロパティーを定義します。コンピュートマシンセットは、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: VSphereMachineTemplate 1
metadata:
  name: <template_name> 2
  namespace: openshift-cluster-api
spec:
  template:
    spec: 3
      template: <vm_template_name> 4
      server: <vcenter_server_ip> 5
      diskGiB: 128
      cloneMode: linkedClone 6
      datacenter: <vcenter_data_center_name> 7
      datastore: <vcenter_datastore_name> 8
      folder: <vcenter_vm_folder_path> 9
      resourcePool: <vsphere_resource_pool> 10
      numCPUs: 4
      memoryMiB: 16384
      network:
        devices:
        - dhcp4: true
          networkName: "<vm_network_name>" 11
```

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。
4. **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。
5. 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。
6. 使用する仮想マシンクローンのタイプを指定します。次の値が有効です。
   - **fullClone**
   - **linkedClone**

   linkedClone タイプを使用する場合、ディスクサイズは **diskGiB** 値を使用するのではなく、クローンソースと同じになります。詳細は、仮想マシンのクローンタイプに関する vSphere のドキュメントを参照してください。
7. コンピュートマシンセットをデプロイする vCenter Datacenter を指定します。
8. コンピュートマシンセットをデプロイする vCenter Datastore を指定します。
9. **/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。

10 仮想マシンの vSphere リソースプールを指定します。

11 コンピュート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。

13.5.3.1.3. VMware vSphere 上の Cluster API コンピュートマシンセットリソースのサンプル YAML

コンピュート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ーを定義します。コンピュートマシンセットは、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: example
  template:
    metadata:
      labels:
        test: example
    spec:
      bootstrap:
        dataSecretName: worker-user-data 3
      clusterName: <cluster_name>
      infrastructureRef:
        apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
        kind: VSphereMachineTemplate 4
        name: <template_name> 5
      failureDomain: 6
        - name: <failure_domain_name>
          region: <region_a>
          zone: <zone_a>
          server: <vcenter_server_name>
          topology:
            datacenter: <region_a_data_center>
            computeCluster: "</region_a_data_center/host/zone_a_cluster>"
            resourcePool: "</region_a_data_center/host/zone_a_cluster/Resources/resource_pool>"
            datastore: "</region_a_data_center/datastore/datastore_a>"
            networks:
            - port-group
```

1 コンピュートマシンセットの名前を指定します。

2 クラスターの名前としてクラスター ID を指定します。

3 テクノロジープレビューの Cluster API では、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。

4 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。

5 マシンテンプレート名を指定します。

6 障害ドメイン設定の詳細を指定します。

注記

Cluster API を使用する vSphere クラスターで複数のリージョンとゾーンを使用することは、検証済みの設定ではありません。

13.6. CLUSTER API を使用するクラスターのトラブルシューティング

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

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

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、Machine 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、Machine 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。

13.6.1. CLI 使用時に目的のオブジェクトを参照する

Cluster API を使用するクラスターの場合、OpenShift CLI (**oc**) コマンドで、Machine API オブジェクトよりも Cluster API オブジェクトが優先されます。

この動作は、Cluster API と Machine API の両方で表されるオブジェクトに対して動作する **oc** コマンドに影響します。この解説では、マシンを削除する **oc delete machine** コマンドを例として使用します。

原因
oc コマンドを実行すると、**oc** は Kube API サーバーと通信して、どのオブジェクトに対して操作を行うかを決定します。Kube API サーバーは、**oc** コマンドが実行されると、アルファベット順で最初に検出されたインストール済みのカスタムリソース定義 (CRD) を使用します。

Cluster API オブジェクトの CRD は **cluster.x-k8s.io** グループにあり、Machine API オブジェクトの CRD は **machine.openshift.io** グループにあります。アルファベット順で文字 **c** は文字 **m** の前に来るため、Kube API サーバーは Cluster API オブジェクトの CRD とマッチします。そのため、**oc** コマンドは Cluster API オブジェクトに対して動作します。

結果

このような動作により、Cluster API を使用するクラスターで次の予期しない結果が発生する可能性があります。

- 両方のタイプのオブジェクトを含む namespace の場合、**oc get machine** などのコマンドが Cluster API オブジェクトのみを返します。
- Machine API オブジェクトのみを含む namespace の場合、**oc get machine** などのコマンドが結果を返しません。

回避策
それぞれの完全修飾名を使用することで、意図したタイプのオブジェクトに対して **oc** コマンドを動作させることができます。

前提条件

- **cluster-admin** 権限を持つアカウントを使用してクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。

手順

- Machine API マシンを削除するには、**oc delete machine** コマンドを実行するときに、完全修飾名 **machine.machine.openshift.io** を使用します。

```
$ oc delete machine.machine.openshift.io <machine_name>
```

- Cluster API マシンを削除するには、**oc delete machine** コマンドを実行するときに、完全修飾名 **machine.cluster.x-k8s.io** を使用します。

```
$ oc delete machine.cluster.x-k8s.io <machine_name>
```

第14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、Machine API が動作しているクラスターでのみ使用できます。user-provisioned infrastructure を持つクラスターでは、Machine 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** のクラスターでは、Machine 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}'
```

14.1. マシンのヘルスチェック

注記

マシンのヘルスチェックは、コンピュートマシンセットまたはコントロールプレーンマシンセットにより管理されるマシンにのみ適用できます。

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。

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。

14.1.1. 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。
- 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

- クラスター内のすべてのノードの一覧表示について
- マシンヘルスチェックによる修復の一時停止 (short-circuiting)
- Control Plane Machine Set Operator について

14.2. 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 1
  namespace: openshift-machine-api
spec:
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-machine-role: <role> 2
      machine.openshift.io/cluster-api-machine-type: <role> 3
      machine.openshift.io/cluster-api-machineset: <cluster_name>-<label>-<zone> 4
  unhealthyConditions:
  - type:    "Ready"
    timeout: "300s" 5
    status: "False"
  - type:    "Ready"
    timeout: "300s" 6
    status: "Unknown"
  maxUnhealthy: "40%" 7
  nodeStartupTimeout: "10m" 8
```

1. 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

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。

14.2.1. マシンヘルスチェックによる修復の一時停止 (short-circuiting)

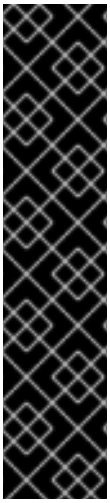
一時停止 (short-circuiting) が実行されることにより、マシンのヘルスチェックはクラスターが正常な場合にのみマシンを修復するようになります。一時停止 (short-circuiting) は、**MachineHealthCheck** リソースの **maxUnhealthy** フィールドで設定されます。

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ー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ーゾーンを持たないグローバル Azure リージョンでは、アベイラビリティーセットを使用して高可用性を確保できます。



重要

コントロールプレーンの **MachineHealthCheck** リソースを設定する場合は、**maxUnhealthy** の値を **1** に設定します。

この設定により、複数のコントロールプレーンマシンが異常であると思われる場合に、マシンのヘルスチェックがアクションを実行しないことが保証されます。複数の異常なコントロールプレーンマシンは、etcd クラスターが劣化していること、または障害が発生したマシンを置き換えるためのスケーリング操作が進行中であることを示している可能性があります。

etcd クラスターが劣化している場合は、手動での介入が必要になる場合があります。スケーリング操作が進行中の場合は、マシンのヘルスチェックで完了できるようにする必要があります。

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。

14.2.1.1. 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。

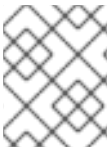
14.2.1.2. パーセンテージを使用した maxUnhealthy の設定

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。

14.3. マシンヘルスチェックリソースの作成

クラスター内のマシンセットの **MachineHealthCheck** リソースを作成できます。

注記

マシンのヘルスチェックは、コンピュートマシンセットまたはコントロールプレーンマシンセットにより管理されるマシンにのみ適用できます。

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。

14.4. 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。

電源バ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

14.4.1. 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パワーサイクルに変更する方法は 2 つあります。

1. **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。
2. **Metal3RemediationTemplate** リソースを作成し、これを **MachineHealthCheck** の **spec.remediationTemplate** で参照します。

いずれかの方法を実行すると、ベースボード管理コントローラー (BMC) 認証情報を使用して、正常でないマシンでパワーサイクルが適用されます。

14.4.2. アノテーションベースの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。

14.4.3. metal3 ベースの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、metal3 修復コントローラーの metal3 修復カスタムリソースを作成します。これは、正常でないノードの電源をオフにするように要求します。
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. metal3 修復コントローラーが、ノードの電源を入れるよう要求します。
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、metal3 修復コントローラーは、削除する前に存在していた正常でないノードのアノテーションとラベルを復元します。

注記

電源操作が完了しなかった場合、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ければ、metal3 修復コントローラーがノードの再プロビジョニングをトリガーします。

14.4.4. 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。
- 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 (アノテーションベースの修復)

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineHealthCheck
metadata:
  name: example (1)
  namespace: openshift-machine-api
  annotations:
    machine.openshift.io/remediation-strategy: external-baremetal (2)
spec:
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-machine-role: <role> (3)
      machine.openshift.io/cluster-api-machine-type: <role> (4)
      machine.openshift.io/cluster-api-machineset: <cluster_name>-<label>-<zone> (5)
  unhealthyConditions:
  - type:    "Ready"
    timeout: "300s" (6)
    status: "False"
  - type:    "Ready"
    timeout: "300s" (7)
    status: "Unknown"
  maxUnhealthy: "40%" (8)
  nodeStartupTimeout: "10m" (9)
```

(1) 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) 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。

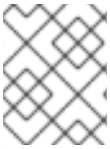
(3) (4) チェックする必要のあるマシンプールのラベルを指定します。

(5) 追跡するコンピュートマシンセットを **<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。

(6) (7) 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。

(8) 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。

(9) 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。

ベアメタルのサンプル MachineHealthCheck リソース (metal3 ベースの修復)

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineHealthCheck
metadata:
  name: example
  namespace: openshift-machine-api
spec:
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-machine-role: <role>
      machine.openshift.io/cluster-api-machine-type: <role>
      machine.openshift.io/cluster-api-machineset: <cluster_name>-<label>-<zone>
  remediationTemplate:
    apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
    kind: Metal3RemediationTemplate
    name: metal3-remediation-template
    namespace: openshift-machine-api
  unhealthyConditions:
  - type:    "Ready"
    timeout: "300s"
```

ベアメタルのサンプル Metal3RemediationTemplate リソース (metal3 ベースの修復)

```
apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: Metal3RemediationTemplate
metadata:
  name: metal3-remediation-template
  namespace: openshift-machine-api
spec:
  template:
    spec:
      strategy:
        type: Reboot
        retryLimit: 1
        timeout: 5m0s
```

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。**annotations** セクションは metal3 ベースの修復には適用されません。アノテーションベースの修復と metal3 ベースの修復は相互に排他的です。

<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>