OpenShift Container Platform 4.7

マシン管理

クラスターマシンの追加および保守

Last Updated: 2023-02-24

OpenShift Container Platform 4.7 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)、Azure、Google Cloud Platform (GCP)、OpenStack、Red Hat Virtualization (RHV)、および vSphere などの基礎となるインフラストラクチャーを柔軟に使用して OpenShift Container Platform クラスターを管理できます。クラスターを制御し、特定のワークロードポリシーに基づいてクラスターをスケールアップやスケールダウンするなどの自動スケーリングを実行できます。

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

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

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

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

マシンセットで実行できること

クラスター管理者として、以下を実行できます。

- 以下でマシンセットを作成する。
  - AWS
  - Azure
  - GCP
  - OpenStack
  - RHV
  - vSphere
- マシンセットからマシンを追加または削除して マシンセットを手動でスケーリング します。
- MachineSet YAML 設定ファイルを使用して マシンセットを変更 します。
- マシンを 削除 します。
- インフラストラクチャー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します。

Autoscaler

クラスターの自動スケーリングにより、ワークロードの変更に柔軟性を持たせることができます。

OpenShift Container Platform 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マシン数を調整します。

ユーザーによってプロビジョニングされるインフラストラクチャー

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の一環として、あるいはその後に、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。

RHEL コンピュートマシンで実行できること

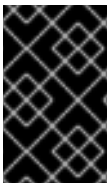
クラスター管理者として、以下を実行できます。

- 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 マシンセットの作成

2.1. AWS でのマシンセットの作成

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



重要

このプロセスは、手動でプロビジョニングされたマシンを持つクラスターには適用されません。高度なマシン管理およびスケーリング機能は、マシン API が機能しているクラスターでのみ使用することができます。

2.1.1. マシン API の概要

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

OpenShift Container Platform 4.7 クラスターの場合、マシン API はクラスターインストールの終了後にすべてのノードホストのプロビジョニングの管理アクションを実行します。このシステムにより、OpenShift Container Platform 4.7 はパブリックまたはプライベートのクラウドインフラストラクチャーに加えて弾力性があり、動的なプロビジョニング方法を提供します。

以下の 2 つのリソースは重要なリソースになります。

Machines

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

マシンセット

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

警告

コントロールプレーンマシンは、マシンセットで管理することはできません。

以下のカスタムリソースは、クラスターに機能を追加します。

Machine Autoscaler

MachineAutoscaler リソースはマシンをクラウドで自動的にスケーリングします。ノードに対する最小および最大のスケーリングの境界を、指定されるマシンセットに設定でき、Machine Autoscaler はノードの該当範囲を維持します。**MachineAutoscaler** オブジェクトは **ClusterAutoscaler** オブ

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

Cluster Autoscaler

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

マシンのヘルスチェック

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

OpenShift Container Platform バージョン 3.11 では、クラスターでマシンのプロビジョニングが管理されないためにマルチゾーンアーキテクチャーを容易に展開することができませんでした。しかし、OpenShift Container Platform バージョン 4.1 以降、このプロセスはより簡単になりました。それぞれのマシンセットのスコープが単一ゾーンに設定されるため、インストールプログラムはユーザーに代わって、アベイラビリティーゾーン全体にマシンセットを送信します。さらに、コンピューティングは動的に展開されるため、ゾーンに障害が発生した場合の、マシンのリバランスが必要な場合に使用するゾーンを常に確保できます。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。

2.1.2. AWS 上のマシンセットカスタムリソースのサンプル YAML

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

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

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
  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: awsproviderconfig.openshift.io/v1beta1
        blockDevices:
          - ebs:
              iops: 0
              volumeSize: 120
              volumeType: gp2
        credentialsSecret:
          name: aws-cloud-credentials
        deviceIndex: 0
        iamInstanceProfile:
          id: <infrastructure_id>-worker-profile 11
        instanceType: m4.large
        kind: AWSMachineProviderConfig
        placement:
          availabilityZone: us-east-1a
          region: us-east-1
        securityGroups:
          - filters:
              - name: tag:Name
                values:
                  - <infrastructure_id>-worker-sg 12
        subnet:
          filters:
            - name: tag:Name
              values:
                - <infrastructure_id>-private-us-east-1a 13
        tags:
          - name: kubernetes.io/cluster/<infrastructure_id> 14
            value: owned
        userDataSecret:
          name: worker-user-data
```

1 3 5 11 12 13 14 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、ノードラベル、およびゾーンを指定します。

6 7 9 追加するノードラベルを指定します。

10 OpenShift Container Platform ノードの AWS ゾーンに有効な Red Hat Enterprise Linux CoreOS (RHCOS) AMI を指定します。

2.1.3. 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ものに加え、独自のマシンセットを作成して、選択する特定のワークロードに対するマシンのコンピュートリソースを動的に管理することができます。

前提条件

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

手順

1. 説明されているようにマシンセット カスタムリソース (CR) サンプルを含む新規 YAML ファイルを作成し、そのファイルに **<file_name>.yaml** という名前を付けます。
<clusterID> および **<role>** パラメーターの値を設定していることを確認します。

   a. 特定のフィールドに設定する値が不明な場合は、クラスターから既存のマシンセットを確認できます。

   ```
   $ 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. 特定のマシンセットの値を確認します。

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

   出力例

   ```
   ...
   template:
     metadata:
       labels:
         machine.openshift.io/cluster-api-cluster: agl030519-vplxk 1
         machine.openshift.io/cluster-api-machine-role: worker 2
         machine.openshift.io/cluster-api-machine-type: worker
         machine.openshift.io/cluster-api-machineset: agl030519-vplxk-worker-us-east-1a
   ```

   1 クラスター ID。

   2 デフォルトのノードラベル。

2. 新規 **MachineSet** CR を作成します。

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

3. マシンセットの一覧を表示します。

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

出力例

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

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

次のステップ

他のアベイラビリティーゾーンで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マシンセットを作成します。

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

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

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

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

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

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

2.1.5. マシンセットの使用による Spot インスタンスの作成

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

手順

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

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

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

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

注記

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

2.1.6. 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。

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

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

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

手順

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

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

2.2. AZURE でのマシンセットの作成

Microsoft Azure 上の OpenShift Container Platform クラスターで特定の目的を果たすように異なるマシンセットを作成することができます。たとえば、インフラストラクチャ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

このプロセスは、手動でプロビジョニングされたマシンを持つクラスターには適用されません。高度なマシン管理およびスケーリング機能は、マシン API が機能しているクラスターでのみ使用することができます。

2.2.1. マシン API の概要

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

OpenShift Container Platform 4.7 クラスターの場合、マシン API はクラスターインストールの終了後にすべてのノードホストのプロビジョニングの管理アクションを実行します。このシステムにより、OpenShift Container Platform 4.7 はパブリックまたはプライベートのクラウドインフラストラクチャーに加えて弾力性があり、動的なプロビジョニング方法を提供します。

以下の 2 つのリソースは重要なリソースになります。

Machines

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

マシンセット

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

警告

コントロールプレーンマシンは、マシンセットで管理することはできません。

以下のカスタムリソースは、クラスターに機能を追加します。

Machine Autoscaler

MachineAutoscaler リソースはマシンをクラウドで自動的にスケーリングします。ノードに対する最小および最大のスケーリングの境界を、指定されるマシンセットに設定でき、Machine Autoscaler はノードの該当範囲を維持します。**MachineAutoscaler** オブジェクトは **ClusterAutoscaler** オブジェクトの設定後に有効になります。**ClusterAutoscaler** および **MachineAutoscaler** リソースは、どちらも **ClusterAutoscalerOperator** オブジェクトによって利用可能にされます。

Cluster Autoscaler

このリソースはアップストリームの Cluster Autoscaler プロジェクトに基づいています。OpenShift Container Platform の実装では、これはマシンセット API を拡張することによってクラスター API

に統合されます。コア、ノード、メモリー、および GPU などのリソースのクラスター全体でのスケーリング制限を設定できます。優先順位を設定することにより、重要度の低い Pod のために新規ノードがオンラインにならないようにクラスターで Pod の優先順位付けを実行できます。また、スケーリングポリシーを設定してノードをスケールダウンせずにスケールアップできるようにすることもできます。

マシンのヘルスチェック

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

OpenShift Container Platform バージョン 3.11 では、クラスターでマシンのプロビジョニングが管理されないためにマルチゾーンアーキテクチャーを容易に展開することができませんでした。しかし、OpenShift Container Platform バージョン 4.1 以降、このプロセスはより簡単になりました。それぞれのマシンセットのスコープが単一ゾーンに設定されるため、インストールプログラムはユーザーに代わって、アベイラビリティーゾーン全体にマシンセットを送信します。さらに、コンピューティングは動的に展開されるため、ゾーンに障害が発生した場合の、マシンのリバランスが必要な場合に使用するゾーンを常に確保できます。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。

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

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

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

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

```
      labels:
        node-role.kubernetes.io/<role>: "" (11)
    providerSpec:
      value:
        apiVersion: azureproviderconfig.openshift.io/v1beta1
        credentialsSecret:
          name: azure-cloud-credentials
          namespace: openshift-machine-api
        image:
          offer: ""
          publisher: ""
          resourceID: /resourceGroups/<infrastructure_id>-
rg/providers/Microsoft.Compute/images/<infrastructure_id> (12)
          sku: ""
          version: ""
        internalLoadBalancer: ""
        kind: AzureMachineProviderSpec
        location: <region> (13)
        managedIdentity: <infrastructure_id>-identity (14)
        metadata:
          creationTimestamp: null
        natRule: null
        networkResourceGroup: ""
        osDisk:
          diskSizeGB: 128
          managedDisk:
            storageAccountType: Premium_LRS
          osType: Linux
        publicIP: false
        publicLoadBalancer: ""
        resourceGroup: <infrastructure_id>-rg (15)
        sshPrivateKey: ""
        sshPublicKey: ""
        subnet: <infrastructure_id>-<role>-subnet (16) (17)
        userDataSecret:
          name: worker-user-data (18)
        vmSize: Standard_DS4_v2
        vnet: <infrastructure_id>-vnet (19)
        zone: "1" (20)
```

(1) (5) (7) (12) (14) (15) (16) (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
```

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 17 18 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。

13 マシンを配置するリージョンを指定します。

20 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。

2.2.3. 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ものに加え、独自のマシンセットを作成して、選択する特定のワークロードに対するマシンのコンピュートリソースを動的に管理することができます。

前提条件

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

手順

1. 説明されているようにマシンセット カスタムリソース (CR) サンプルを含む新規 YAML ファイルを作成し、そのファイルに **<file_name>.yaml** という名前を付けます。
   <clusterID> および **<role>** パラメーターの値を設定していることを確認します。

   a. 特定のフィールドに設定する値が不明な場合は、クラスターから既存のマシンセットを確認できます。

   ```
   $ 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. 特定のマシンセットの値を確認します。

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

   出力例

```
...
template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: agl030519-vplxk 1
        machine.openshift.io/cluster-api-machine-role: worker 2
        machine.openshift.io/cluster-api-machine-type: worker
        machine.openshift.io/cluster-api-machineset: agl030519-vplxk-worker-us-east-1a
```

1. クラスター ID。
2. デフォルトのノードラベル。

2. 新規 **MachineSet** CR を作成します。

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

3. マシンセットの一覧を表示します。

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

出力例

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

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

2.2.4. マシンを Spot 仮想マシンとしてデプロイするマシンセット

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

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

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

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

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

2.2.5. マシンセットの使用による Spot 仮想マシンの作成

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

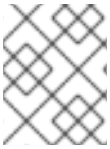
手順

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

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

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

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



注記

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

2.2.6. マシンセットの顧客管理の暗号鍵の有効化

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

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

前提条件

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

手順

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

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

関連情報

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

2.3. GCP でのマシンセットの作成

異なるマシンセットを作成して、Google Cloud Platform (GCP) 上の OpenShift Container Platform クラスタ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

このプロセスは、手動でプロビジョニングされたマシンを持つクラスターには適用されません。高度なマシン管理およびスケーリング機能は、マシン API が機能しているクラスターでのみ使用することができます。

2.3.1. マシン API の概要

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

OpenShift Container Platform 4.7 クラスターの場合、マシン API はクラスターインストールの終了後にすべてのノードホストのプロビジョニングの管理アクションを実行します。このシステムにより、OpenShift Container Platform 4.7 はパブリックまたはプライベートのクラウドインフラストラクチャーに加えて弾力性があり、動的なプロビジョニング方法を提供します。

以下の 2 つのリソースは重要なリソースになります。

Machines

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

マシンセット

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

警告

コントロールプレーンマシンは、マシンセットで管理することはできません。

以下のカスタムリソースは、クラスターに機能を追加します。

Machine Autoscaler

MachineAutoscaler リソースはマシンをクラウドで自動的にスケーリングします。ノードに対する最小および最大のスケーリングの境界を、指定されるマシンセットに設定でき、Machine Autoscaler はノードの該当範囲を維持します。**MachineAutoscaler** オブジェクトは **ClusterAutoscaler** オブジェクトの設定後に有効になります。**ClusterAutoscaler** および **MachineAutoscaler** リソースは、どちらも **ClusterAutoscalerOperator** オブジェクトによって利用可能にされます。

Cluster Autoscaler

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

マシンのヘルスチェック

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

OpenShift Container Platform バージョン 3.11 では、クラスターでマシンのプロビジョニングが管理されないためにマルチゾーンアーキテクチャーを容易に展開することができませんでした。しかし、OpenShift Container Platform バージョン 4.1 以降、このプロセスはより簡単になりました。それぞれのマシンセットのスコープが単一ゾーンに設定されるため、インストールプログラムはユーザーに代わって、アベイラビリティーゾーン全体にマシンセットを送信します。さらに、コンピューティングは動的に展開されるため、ゾーンに障害が発生した場合の、マシンのリバランスが必要な場合に使用するゾーンを常に確保できます。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。

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

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

このサンプルでは、**<infrastructure_id>** はクラスタ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>-w-a 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>-w-a 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>-w-a 8
    spec:
      metadata:
        labels:
          node-role.kubernetes.io/<role>: "" 9
      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> 10
            labels: null
            sizeGb: 128
            type: pd-ssd
          gcpMetadata: 11
          - key: <custom_metadata_key>
            value: <custom_metadata_value>
          kind: GCPMachineProviderSpec
          machineType: n1-standard-4
          metadata:
            creationTimestamp: null
          networkInterfaces:
          - network: <infrastructure_id>-network 12
            subnetwork: <infrastructure_id>-worker-subnet 13
          projectID: <project_name> 14
          region: us-central1
          serviceAccounts:
          - email: <infrastructure_id>-w@<project_name>.iam.gserviceaccount.com 15 16
            scopes:
            - https://www.googleapis.com/auth/cloud-platform
          tags:
          - <infrastructure_id>-worker 17
          userDataSecret:
            name: worker-user-data
          zone: us-central1-a
```

1 2 3 4 5 8 12 13 15 17 クラスターのプロビジョニング時に設定したクラスター ID を基にするインフラストラクチャー ID を指定します。OpenShift CLI がインストー

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

6 7 9 追加するノードラベルを指定します。

10 現在の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。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
```

11 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。

14 16 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。

2.3.3. 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ものに加え、独自のマシンセットを作成して、選択する特定のワークロードに対するマシンのコンピュートリソースを動的に管理することができます。

前提条件

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

手順

1. 説明されているようにマシンセット カスタムリソース (CR) サンプルを含む新規 YAML ファイルを作成し、そのファイルに **<file_name>.yaml** という名前を付けます。
   <clusterID> および **<role>** パラメーターの値を設定していることを確認します。

   a. 特定のフィールドに設定する値が不明な場合は、クラスターから既存のマシンセットを確認できます。

   ```
   $ 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. 特定のマシンセットの値を確認します。

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

出力例

```
...
template:
   metadata:
     labels:
       machine.openshift.io/cluster-api-cluster: agl030519-vplxk 1
       machine.openshift.io/cluster-api-machine-role: worker 2
       machine.openshift.io/cluster-api-machine-type: worker
       machine.openshift.io/cluster-api-machineset: agl030519-vplxk-worker-us-east-1a
```

1 クラスター ID。

2 デフォルトのノードラベル。

2. 新規 **MachineSet** CR を作成します。

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

3. マシンセットの一覧を表示します。

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

出力例

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

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

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

マシンを保証されていないプリエン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。マシンセットの **replicas** の量を満たすために、マシンセットはプリエンプション可能な仮想マシンインスタンスを要求するマシンを作成します。

2.3.5. 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。

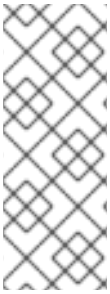
```
providerSpec:
  value:
    preemptible: true
```

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。

2.3.6. マシンセットの顧客管理の暗号鍵の有効化

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

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



注記

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

手順

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

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

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

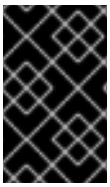
```
providerSpec:
  value:
    # ...
    disks:
    - type:
      # ...
      encryptionKey:
        kmsKey:
          name: machine-encryption-key 1
          keyRing: openshift-encrpytion-ring 2
          location: global 3
          projectID: openshift-gcp-project 4
        kmsKeyServiceAccount: 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規マシンが作成された後に、ディスクの暗号化キーは KMS キーを使用して暗号化されます。

2.4. OPENSTACK でのマシンセットの作成

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



重要

このプロセスは、手動でプロビジョニングされたマシンを持つクラスターには適用されません。高度なマシン管理およびスケーリング機能は、マシン API が機能しているクラスターでのみ使用することができます。

2.4.1. マシン API の概要

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

OpenShift Container Platform 4.7 クラスターの場合、マシン API はクラスターインストールの終了後にすべてのノードホストのプロビジョニングの管理アクションを実行します。このシステムにより、OpenShift Container Platform 4.7 はパブリックまたはプライベートのクラウドインフラストラクチャーに加えて弾力性があり、動的なプロビジョニング方法を提供します。

以下の 2 つのリソースは重要なリソースになります。

Machines

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

マシンセット

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

警告

コントロールプレーンマシンは、マシンセットで管理することはできません。

以下のカスタムリソースは、クラスターに機能を追加します。

Machine Autoscaler

MachineAutoscaler リソースはマシンをクラウドで自動的にスケーリングします。ノードに対する最小および最大のスケーリングの境界を、指定されるマシンセットに設定でき、Machine Autoscaler はノードの該当範囲を維持します。**MachineAutoscaler** オブジェクトは **ClusterAutoscaler** オブ

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

Cluster Autoscaler

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

マシンのヘルスチェック

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

OpenShift Container Platform バージョン 3.11 では、クラスターでマシンのプロビジョニングが管理されないためにマルチゾーンアーキテクチャーを容易に展開することができませんでした。しかし、OpenShift Container Platform バージョン 4.1 以降、このプロセスはより簡単になりました。それぞれのマシンセットのスコープが単一ゾーンに設定されるため、インストールプログラムはユーザーに代わって、アベイラビリティーゾーン全体にマシンセットを送信します。さらに、コンピューティングは動的に展開されるため、ゾーンに障害が発生した場合の、マシンのリバランスが必要な場合に使用するゾーンを常に確保できます。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。

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

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

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

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

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

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

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

2.4.3. 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ものに加え、独自のマシンセットを作成して、選択する特定のワークロードに対するマシンのコンピュートリソースを動的に管理することができます。

前提条件

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

手順

1. 説明されているようにマシンセット カスタムリソース (CR) サンプルを含む新規 YAML ファイルを作成し、そのファイルに **<file_name>.yaml** という名前を付けます。
   <clusterID> および **<role>** パラメーターの値を設定していることを確認します。

   a. 特定のフィールドに設定する値が不明な場合は、クラスターから既存のマシンセットを確認できます。

   ```
   $ 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. 特定のマシンセットの値を確認します。

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

   出力例

   ```
   ...
   template:
     metadata:
       labels:
         machine.openshift.io/cluster-api-cluster: agl030519-vplxk 1
         machine.openshift.io/cluster-api-machine-role: worker 2
         machine.openshift.io/cluster-api-machine-type: worker
         machine.openshift.io/cluster-api-machineset: agl030519-vplxk-worker-us-east-1a
   ```

   1 クラスター ID。

   2 デフォルトのノードラベル。

2. 新規 **MachineSet** CR を作成します。

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

3. マシンセットの一覧を表示します。

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

出力例

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

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

2.5. RHV でのマシンセットの作成

異なるマシンセットを作成して、Red Hat Virtualization (RHV) 上の OpenShift Container Platform クラスタ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

このプロセスは、手動でプロビジョニングされたマシンを持つクラスターには適用されません。高度なマシン管理およびスケーリング機能は、マシン API が機能しているクラスターでのみ使用することができます。

2.5.1. マシン API の概要

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

OpenShift Container Platform 4.7 クラスターの場合、マシン API はクラスターインストールの終了後にすべてのノードホストのプロビジョニングの管理アクションを実行します。このシステムにより、OpenShift Container Platform 4.7 はパブリックまたはプライベートのクラウドインフラストラクチャーに加えて弾力性があり、動的なプロビジョニング方法を提供します。

以下の 2 つのリソースは重要なリソースになります。

Machines

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

マシンセット

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

警告

コントロールプレーンマシンは、マシンセットで管理することはできません。

以下のカスタムリソースは、クラスターに機能を追加します。

Machine Autoscaler

MachineAutoscaler リソースはマシンをクラウドで自動的にスケーリングします。ノードに対する最小および最大のスケーリングの境界を、指定されるマシンセットに設定でき、Machine Autoscaler はノードの該当範囲を維持します。**MachineAutoscaler** オブジェクトは **ClusterAutoscaler** オブジェクトの設定後に有効になります。**ClusterAutoscaler** および **MachineAutoscaler** リソースは、どちらも **ClusterAutoscalerOperator** オブジェクトによって利用可能にされます。

Cluster Autoscaler

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

マシンのヘルスチェック

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

OpenShift Container Platform バージョン 3.11 では、クラスターでマシンのプロビジョニングが管理されないためにマルチゾーンアーキテクチャーを容易に展開することができませんでした。しかし、OpenShift Container Platform バージョン 4.1 以降、このプロセスはより簡単になりました。それぞれのマシンセットのスコープが単一ゾーンに設定されるため、インストールプログラムはユーザーに代わって、アベイラビリティーゾーン全体にマシンセットを送信します。さらに、コンピューティングは動的に展開されるため、ゾーンに障害が発生した場合の、マシンのリバランスが必要な場合に使用するゾーンを常に確保できます。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。

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

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

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

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

```
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
    machine.openshift.io/cluster-api-machine-role: <role> 2
    machine.openshift.io/cluster-api-machine-type: <role> 3
  name: <infrastructure_id>-<role> 4
  namespace: openshift-machine-api
spec:
  replicas: <number_of_replicas> 5
  selector: 6
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 7
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role> 8
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 9
        machine.openshift.io/cluster-api-machine-role: <role> 10
        machine.openshift.io/cluster-api-machine-type: <role> 11
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role> 12
    spec:
      metadata:
        labels:
          node-role.kubernetes.io/<role>: "" 13
      providerSpec:
        value:
          apiVersion: ovirtproviderconfig.machine.openshift.io/v1beta1
          cluster_id: <ovirt_cluster_id> 14
          template_name: <ovirt_template_name> 15
          instance_type_id: <instance_type_id> 16
          cpu: 17
            sockets: <number_of_sockets> 18
            cores: <number_of_cores> 19
            threads: <number_of_threads> 20
          memory_mb: <memory_size> 21
          os_disk: 22
            size_gb: <disk_size> 23
          network_interfaces: 24
            vnic_profile_id:  <vnic_profile_id> 25
          credentialsSecret:
            name: ovirt-credentials 26
          kind: OvirtMachineProviderSpec
          type: <workload_type> 27
          userDataSecret:
            name: worker-user-data
```

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

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

2 3 10 11 13 追加するノードラベルを指定します。

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

5 作成するマシンの数を指定します。

6 マシンのセレクター。

14 この仮想マシンインスタンスが属する RHV クラスターの UUID を指定します。

15 マシンの作成に使用する RHV 仮想マシンテンプレートを指定します。

16 オプション: 仮想マシンインスタンスタイプを指定します。

警告

instance_type_id フィールドは非推奨となり、今後のリリースで削除されます。

このパラメーターを含めると、CPU およびメモリーを含む仮想マシンのハードウェアパラメーターを指定する必要はありません。このパラメーターは、すべてのハードウェアパラメーターを上書きするためです。

17 オプション: CPU フィールドには、ソケット、コア、スレッドを含む CPU の設定が含まれます。

18 オプション: 仮想マシンのソケット数を指定します。

19 オプション: ソケットあたりのコア数を指定します。

20 オプション: コアあたりのスレッド数を指定します。

21 オプション: 仮想マシンのメモリーサイズを MiB 単位で指定します。

22 オプション: ノードのルートディスク。

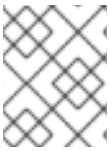
23 オプション: ブート可能なディスクのサイズを GiB 単位で指定します。

24 オプション: 仮想マシンのネットワークインターフェイスの一覧。このパラメーターを含めると、OpenShift Container Platform はテンプレートからすべてのネットワークインターフェイスを破棄し、新規ネットワークインターフェイスを作成します。

25 オプション: vNIC プロファイル ID を指定します。

26 RHV 認証情報を保持するシークレットの名前を指定します。

27 オプション: インスタンスが最適化されるワークロードタイプを指定します。この値は **RHV VM** パラメーターに影響します。サポートされる値: **desktop**、**server** (デフォルト)、**high_performance** です。**high_performance** により、仮想マシンのパフォーマンスが向上しますが、制限があります。たとえば、グラフィカルコンソールを使用して仮想マシンにはアクセスできません。詳細は、**仮想マシン管理ガイド**の ハイパフォーマンス仮想マシン、テンプレート、およびプールの設定 を参照してください。



注記

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

2.5.3. 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ものに加え、独自のマシンセットを作成して、選択する特定のワークロードに対するマシンのコンピュートリソースを動的に管理することができます。

前提条件

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

手順

1. 説明されているようにマシンセット カスタムリソース (CR) サンプルを含む新規 YAML ファイルを作成し、そのファイルに **<file_name>.yaml** という名前を付けます。
   <clusterID> および **<role>** パラメーターの値を設定していることを確認します。

   a. 特定のフィールドに設定する値が不明な場合は、クラスターから既存のマシンセットを確認できます。

   ```
   $ 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. 特定のマシンセットの値を確認します。

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

   出力例

   ```
   ...
   template:
     metadata:
       labels:
         machine.openshift.io/cluster-api-cluster: agl030519-vplxk 1
   ```

```
      machine.openshift.io/cluster-api-machine-role: worker 2
      machine.openshift.io/cluster-api-machine-type: worker
      machine.openshift.io/cluster-api-machineset: agl030519-vplxk-worker-us-east-1a
```

1 クラスター ID。

2 デフォルトのノードラベル。

2. 新規 **MachineSet** CR を作成します。

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

3. マシンセットの一覧を表示します。

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

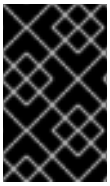
出力例

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

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

2.6. VSPHERE でのマシンセットの作成

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

このプロセスは、手動でプロビジョニングされたマシンを持つクラスターには適用されません。高度なマシン管理およびスケーリング機能は、マシン API が機能しているクラスターでのみ使用することができます。

2.6.1. マシン API の概要

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

OpenShift Container Platform 4.7 クラスターの場合、マシン API はクラスターインストールの終了後にすべてのノードホストのプロビジョニングの管理アクションを実行します。このシステムにより、OpenShift Container Platform 4.7 はパブリックまたはプライベートのクラウドインフラストラクチャーに加えて弾力性があり、動的なプロビジョニング方法を提供します。

以下の 2 つのリソースは重要なリソースになります。

Machines

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

マシンセット

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

警告

コントロールプレーンマシンは、マシンセットで管理することはできません。

以下のカスタムリソースは、クラスターに機能を追加します。

Machine Autoscaler

MachineAutoscaler リソースはマシンをクラウドで自動的にスケーリングします。ノードに対する最小および最大のスケーリングの境界を、指定されるマシンセットに設定でき、Machine Autoscaler はノードの該当範囲を維持します。**MachineAutoscaler** オブジェクトは **ClusterAutoscaler** オブジェクトの設定後に有効になります。**ClusterAutoscaler** および **MachineAutoscaler** リソースは、どちらも **ClusterAutoscalerOperator** オブジェクトによって利用可能にされます。

Cluster Autoscaler

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

マシンのヘルスチェック

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

OpenShift Container Platform バージョン 3.11 では、クラスターでマシンのプロビジョニングが管理されないためにマルチゾーンアーキテクチャーを容易に展開することができませんでした。しかし、OpenShift Container Platform バージョン 4.1 以降、このプロセスはより簡単になりました。それぞれのマシンセットのスコープが単一ゾーンに設定されるため、インストールプログラムはユーザーに代わって、アベイラビリティーゾーン全体にマシンセットを送信します。さらに、コンピューティングは動的に展開されるため、ゾーンに障害が発生した場合の、マシンのリバランスが必要な場合に使用するゾーンを常に確保できます。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。

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

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

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

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  creationTimestamp: null
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 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_datacenter_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 マシンセットをデプロイする vSphere 仮想マシンネットワークを指定します。この仮想マシンネットワークは、他のコンピューティングマシンがクラスター内に存在する場所である必要があります。

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

重要

元の仮想マシンテンプレートは指定しないでください。仮想マシンテンプレートは停止した状態でなければなりません。また、新規 RHCOS マシン用にクローン作成する必要があります。仮想マシンテンプレートを起動すると、仮想マシンテンプレートがプラットフォームの仮想マシンとして設定されるので、マシンセットが設定に適用できるテンプレートとして使用できなくなります。

12 マシンセットをデプロイする vCenter Datacenter を指定します。

13 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。

2.6.3. 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ものに加え、独自のマシンセットを作成して、選択する特定のワークロードに対するマシンのコンピュートリソースを動的に管理することができます。

前提条件

- OpenShift Container Platform クラスターをデプロイすること。
- OpenShift CLI (**oc**) をインストールしている。
- **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。

- クラスター API 名に基づいて vCenter インスタンス内にタグを作成します。このタグは、OpenShift Container Platform ノードをプロビジョニングされた仮想マシン (VM) に関連付けるためにマシンセットによって使用されます。vCenter でタグを作成する方法については、VMware ドキュメントで、vSphere タグおよび属性 について参照してください。
- 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。

手順

1. 説明されているようにマシンセット カスタムリソース (CR) サンプルを含む新規 YAML ファイルを作成し、そのファイルに **<file_name>.yaml** という名前を付けます。
<clusterID> および **<role>** パラメーターの値を設定していることを確認します。

   a. 特定のフィールドに設定する値が不明な場合は、クラスターから既存のマシンセットを確認できます。

   ```
   $ 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. 特定のマシンセットの値を確認します。

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

   出力例

   ```
   ...
   template:
     metadata:
       labels:
         machine.openshift.io/cluster-api-cluster: agl030519-vplxk 1
         machine.openshift.io/cluster-api-machine-role: worker 2
         machine.openshift.io/cluster-api-machine-type: worker
         machine.openshift.io/cluster-api-machineset: agl030519-vplxk-worker-us-east-1a
   ```

   1 クラスター ID。

   2 デフォルトのノードラベル。

2. 新規 **MachineSet** CR を作成します。

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

3. マシンセットの一覧を表示します。

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

出力例

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

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

第3章 マシンセットの手動によるスケーリング

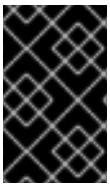
マシンセットのマシンのインスタンスを追加または削除できます。

注記

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

3.1. 前提条件

- クラスター全体のプロキシーを有効にし、インストール設定から **networking.machineNetwork[].cidr** に含まれていないワーカーをスケールアップする場合、ワーカーをプロキシーオブジェクトの **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

このプロセスは、手動でプロビジョニングされたマシンを持つクラスターには適用されません。高度なマシン管理およびスケーリング機能は、マシン API が機能しているクラスターでのみ使用することができます。

3.2. マシンセットの手動によるスケーリング

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

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

前提条件

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

手順

1. クラスターにあるマシンセットを表示します。

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

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

2. クラスター内にあるマシンを表示します。

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

3. 削除するマシンに注釈を設定します。

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

4. 削除するノードを分離して解放します。

```
$ oc adm cordon <node_name>
$ oc adm drain <node_name>
```

5. マシンセットをスケーリングします。

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

または、以下を実行します。

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

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。

検証

- 目的のマシンの削除を確認します。

```
$ oc get machines
```

3.3. マシンセットの削除ポリシー

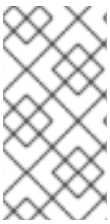
Random、**Newest**、および **Oldest** は 3 つのサポートされる削除オプションです。デフォルトは **Random** であり、これはマシンセットのスケールダウン時にランダムマシンが選択され、削除されることを意味します。削除ポリシーは、特定のマシンセットを変更し、ユースケースに基づいて設定できます。

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

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

重要

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



注記

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

第4章 マシンセットの変更

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

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

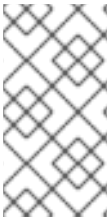
注記

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

4.1. マシンセットの変更

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

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



注記

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

前提条件

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

手順

1. マシンセットを編集します。

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

2. マシンセットを **0** にスケールダウンします。

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

   または、以下を実行します。

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

   マシンが削除されるまで待機します。

3. マシンセットを随時スケールアップします。

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

または、以下を実行します。

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

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

4.2. RHV 上の別のストレージドメインへのノードの移行

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

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

前提条件

- Manager にログインしている。
- ターゲットとなるストレージドメインの名前を把握している。

手順

1. 仮想マシンテンプレートを特定します。

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

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

   ヒント

   新しい仮想マシンを起動する必要はありません。

3. 新規仮想マシンから新規テンプレートを作成します。**Target** にターゲットストレージドメインを指定します。詳細は、Red Hat Virtualization **Virtual Machine Management Guide**の Creating a Template を参照してください。

4. 新規テンプレートを使用して、新規マシンセットを OpenShift Container Platform クラスターに追加します。

   a. 現在のマシンセットの詳細を取得します。

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

   b. これらの詳細を使用してマシンセットを作成します。詳細は、**マシンセットの作成**を参照してください。
   template_name フィールドに新規仮想マシンテンプレート名を入力します。Manager の **New template** ダイアログで使用したものと同じテンプレート名を使用します。

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

5. ワークロードを移行します。

a. 新規のマシンセットをスケールアップします。マシンセットの手動によるスケーリングについての詳細は、**マシンセットの手動によるスケーリング**を参照してください。
OpenShift Container Platform は、古いマシンが削除されると Pod を利用可能なワーカーに移動します。

b. 古いマシンセットをスケールダウンします。

6. 古いマシンセットを削除します。

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

関連情報

- マシンセットの作成
- マシンセットの手動によるスケーリング
- スケジューラーによる Pod 配置の制御

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

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

ノードごとに個別にこの手順を実行します。

前提条件

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

手順

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

8. OpenShift Container Platform クラスターが安定していることを確認します。

```
$ oc get nodes
```

   出力には、ステータスが **Ready** のノードが表示されます。

9. コントロールプレーンノードごとに、この手順を繰り返します。

第5章 マシンの削除

特定のマシンを削除できます。

5.1. 特定マシンの削除

特定のマシンを削除できます。

注記

コントロールプレーンマシンは削除できません。

前提条件

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

手順

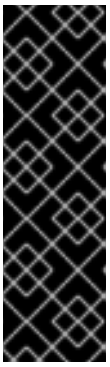
1. クラスターにあるマシンを表示し、削除するマシンを特定します。

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

   コマンド出力には、**<clusterid>-worker-<cloud_region>** 形式のマシンの一覧が含まれます。

2. 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 の Disruption Budget(停止状態の予算) が正しく設定されていない場合などには、ドレイン (解放) の操作を実行してもマシンの削除を防ぐことができない場合があります。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。削除中のマシンがマシンセットに属する場合、指定されたレプリカ数に対応するために新規マシンが即時に作成されます。

5.2. 関連情報

- 正常でない etcd メンバーの置き換え

第6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

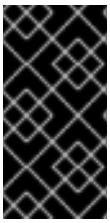
Cluster Autoscaler は、マシン API が機能しているクラスターでのみ設定できます。

6.1. CLUSTER AUTOSCALER について

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

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

Cluster Autoscaler は、コントロールプレーンノードを管理しない場合でも、クラスター内のすべてのノードのメモリー、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。

- ノードで実行されているすべての Pod の CPU およびメモリー要求の合計が、ノードに割り当てられたリソースの 50% 未満である。
- 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。
- Cluster Autoscaler で、スケールダウンが無効にされたアノテーションがない。

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

- 制限のある Pod の Disruption Budget (停止状態の予算、PDB) を持つ Pod。
- デフォルトでノードで実行されない Kube システム Pod。
- PDB を持たないか、または制限が厳しい PDB を持つ Kuber システム Pod。

- デプロイメント、レプリカセット、またはステートフルセットなどのコントローラーオブジェクトによってサポートされない Pod。
- ローカルストレージを持つ Pod。
- リソース不足、互換性のないノードセ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。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。

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

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

6.2. MACHINE AUTOSCALER

Machine Autoscaler は、マシンセットで OpenShift Container Platform クラスターにデプロイするマシン数を調整します。デフォルトの **worker** マシンセットおよび作成する他のマシンセットの両方をスケーリングできます。Machine Autoscaler は、追加のデプロイメントをサポートするのに十分なリソー

スがクラスターにない場合に追加のマシンを作成します。**MachineAutoscaler** リソースの値への変更 (例: インスタンスの最小または最大数) は、それらがターゲットとするマシンセットに即時に適用されます。

重要

マシンをスケーリングするには、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。

6.3. CLUSTER AUTOSCALER の設定

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

注記

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

6.3.1. ClusterAutoscaler リソース定義

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

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

```
  delayAfterDelete: 5m (13)
  delayAfterFailure: 30s (14)
  unneededTime: 5m (15)
```

(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) オプションで、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) このセクションでは、有効な ParseDuration 期間 (**ns**、**us**、**ms**、**s**、**m**、および **h** を含む) を使用して各アクションについて待機する期間を指定できます。

(11) Cluster Autoscaler が不必要なノードを削除できるかどうかを指定します。

(12) オプションで、ノードが最後に **追加** されてからノードを削除するまで待機する期間を指定します。値を指定しない場合、デフォルト値の **10m** が使用されます。

(13) ノードが最後に **削除** された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 **10s** が使用されます。

(14) スケールダウンが失敗してからノードを削除するまで待機する期間を指定します。値を指定しない場合、デフォルト値の **3m** が使用されます。

(15) 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。

注記

スケーリング操作の実行時に、Cluster Autoscaler は、デプロイするコアの最小および最大数、またはクラスター内のメモリー量などの **ClusterAutoscaler** リソース定義に設定された範囲内に残ります。ただし、Cluster Autoscaler はそれらの範囲内に留まるようクラスターの現在の値を修正しません。

Cluster Autoscaler がノードを管理しない場合でも、最小および最大の CPU、メモリー、および GPU の値は、クラスター内のすべてのノードのこれらのリソースを計算することによって決定されます。たとえば、Cluster Autoscaler がコントロールプレーンノードを管理しない場合でも、コントロールプレーンノードはクラスターのメモリーの合計に考慮されます。

6.3.2. Cluster Autoscaler のデプロイ

Cluster Autoscaler をデプロイするには、**ClusterAutoscaler** リソースのインスタンスを作成します。

手順

1. カスタマイズされたリソース定義を含む **ClusterAutoscaler** リソースの YAML ファイルを作成します。
2. クラスターにリソースを作成します。

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

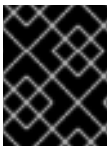
1 **<filename>** は、カスタマイズしたリソースファイルの名前です。

6.4. 次のステップ

- Cluster Autoscaler の設定後に、1 つ以上の Machine Autoscaler を設定する必要があります。

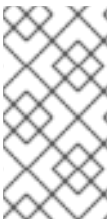
6.5. MACHINE AUTOSCALER の設定

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マシンセットを参照する **MachineAutoscaler** リソースをデプロイします。



重要

ClusterAutoscaler リソースのデプロイ後に、1 つ以上の **MachineAutoscaler** リソースをデプロイする必要があります。



注記

各マシンセットに対して別々のリソースを設定する必要があります。マシンセットはそれぞれのリージョンごとに異なるため、複数のリージョンでマシンのスケー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。スケーリングするマシンセットには 1 つ以上のマシンが必要です。

6.5.1. MachineAutoscaler リソース定義

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

```
apiVersion: "autoscaling.openshift.io/v1beta1"
kind: "MachineAutoscaler"
metadata:
  name: "worker-us-east-1a" (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マシンセットを簡単に特定できるようにするには、スケーリングするマシンセットの名前を指定するか、またはこれを組み込みます。マシンセットの名前は、**<clusterid>-<machineset>-<region>** の形式を使用します。

(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、または RHOSP で実行している場合は、この値は **0** に設定できます。他のプロバイダーの場合は、この値は **0** に設定しないでください。

特殊なワークロードに使用されるコストがかかり、用途が限られたハードウェアを稼働する場合などのユースケースにはこの値を **0** に設定するか、若干大きいマシンを使用してマシンセットをスケーリングすることで、コストを節約できます。Cluster Autoscaler は、マシンが使用されていない場合にマシンセットをゼロにスケールダウンします。

重要

インストーラーでプロビジョニングされるインフラストラクチャーの OpenShift Container Platform インストールプロセス時に作成される 3 つのコンピュートマシンセットについては、**spec.minReplicas** の値を **0** に設定しないでください。

(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。**ClusterAutoscaler** リソース定義の **maxNodesTotal** 値が、Machine AutoScaler がこの数のマシンをデプロイするのに十分な大きさの値であることを確認します。

(4) このセクションでは、スケーリングする既存のマシンセットを記述する値を指定します。

(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マシンセットの名前に一致する必要があります。

6.5.2. Machine Autoscaler のデプロイ

Machine Autoscaler をデプロイするには、 **MachineAutoscaler** リソースのインスタンスを作成します。

手順

1. カスタマイズされたリソース定義を含む **MachineAutoscaler** リソースの YAML ファイルを作成します。
2. クラスターにリソースを作成します。

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

1 **<filename>** は、カスタマイズしたリソースファイルの名前です。

6.6. 関連情報

- Pod の優先順位についての詳細は、Pod スケジューリングの決定に Pod の優先順位を含めるを参照してください。

第7章 インフラストラクチャーマシンセットの作成

重要

このプロセスは、手動でプロビジョニングされたマシンを持つクラスターには適用されません。高度なマシン管理およびスケーリング機能は、マシン API が機能しているクラスターでのみ使用することができます。

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ー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。

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

以下のインフラストラクチャーワークロードでは、OpenShift Container Platform ワーカーのサブスクリプションは不要です。

- マスターで実行される Kubernetes および OpenShift Container Platform コントロールプレーンサービス
- デフォルトルーター
- 統合コンテナ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ー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ーリングサービス
- クラスター集計ロギング
- サービスブローカー
- Red Hat Quay
- Red Hat OpenShift Container Storage
- Red Hat Advanced Cluster Manager
- Kubernetes 用 Red Hat Advanced Cluster Security
- Red Hat OpenShift GitOps
- Red Hat OpenShift Pipelines

他のコンテナー、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。

関連情報

- 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。

7.2. 実稼働環境用のインフラストラクチャーマシンセットの作成

実稼働デプロイメントでは、インフラストラクチャーコンポーネントを保持するために 3 つ以上のマシンセットをデプロイすることが推奨されます。OpenShift Logging と Red Hat OpenShift Service Mesh の両方が Elasticsearch をデプロイします。これには、3 つのインスタンスを異なるノードにインストールする必要があります。これらの各ノードは、高可用性のために異なるアベイラビリティ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。

7.2.1. 異なるクラウドのマシンセットの作成

クラウドのサンプルマシンセットを使用します。

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

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

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

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 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
      taints: 10
        - key: node-role.kubernetes.io/infra
          effect: NoSchedule
      providerSpec:
        value:
          ami:
            id: ami-046fe691f52a953f9 11
          apiVersion: awsproviderconfig.openshift.io/v1beta1
```

```
        blockDevices:
          - ebs:
              iops: 0
              volumeSize: 120
              volumeType: gp2
        credentialsSecret:
          name: aws-cloud-credentials
        deviceIndex: 0
        iamInstanceProfile:
          id: <infrastructure_id>-worker-profile (12)
        instanceType: m4.large
        kind: AWSMachineProviderConfig
        placement:
          availabilityZone: us-east-1a
          region: us-east-1
        securityGroups:
          - filters:
              - name: tag:Name
                values:
                  - <infrastructure_id>-worker-sg (13)
        subnet:
          filters:
            - name: tag:Name
              values:
                - <infrastructure_id>-private-us-east-1a (14)
        tags:
          - name: kubernetes.io/cluster/<infrastructure_id> (15)
            value: owned
        userDataSecret:
          name: worker-user-data
```

(1) (3) (5) (12) (13) (14) (15) 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>** ノードラベル、およびゾーンを指定します。

(6) (7) (9) **<infra>** ノードラベルを指定します。

(10) ユーザーのワークロードが infra ノードにスケジュールされないようにテイントを指定します。

(11) OpenShift Container Platform ノードの AWS ゾーンに有効な Red Hat Enterprise Linux CoreOS (RHCOS) AMI を指定します。

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

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。**spotMarketOptions** を **MachineSet** YAML ファイルに追加して、Spot インスタンスを設定 します。

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

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

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

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

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。

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。**spotVMOptions** を **MachineSet** YAML ファイルに追加して、Spot 仮想マシンを設定 できます。

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

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

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

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
  name: <infrastructure_id>-w-a 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>-w-a 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>-w-a 8
    spec:
      metadata:
        labels:
          node-role.kubernetes.io/infra: "" 9
      taints: 10
      - key: node-role.kubernetes.io/infra
        effect: NoSchedule
      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> 11
            labels: null
            sizeGb: 128
            type: pd-ssd
```

```
        gcpMetadata: (12)
        - key: <custom_metadata_key>
          value: <custom_metadata_value>
        kind: GCPMachineProviderSpec
        machineType: n1-standard-4
        metadata:
          creationTimestamp: null
        networkInterfaces:
        - network: <infrastructure_id>-network (13)
          subnetwork: <infrastructure_id>-worker-subnet (14)
        projectID: <project_name> (15)
        region: us-central1
        serviceAccounts:
        - email: <infrastructure_id>-w@<project_name>.iam.gserviceaccount.com (16) (17)
          scopes:
          - https://www.googleapis.com/auth/cloud-platform
        tags:
        - <infrastructure_id>-worker (18)
        userDataSecret:
          name: worker-user-data
        zone: us-central1-a
```

(1) (2) (3) (4) (5) (8) (13) (14) (16) (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
```

(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。

(11) 現在の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。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
```

(12) 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。

(15) (17) 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。

GCP で実行されるマシンセットは、保証されていない プリエンプション可能な仮想マシンインスタンス をサポートします。GCP の通常のインスタンスと比較して、プリエンプション可能な仮想マシンインスタンスをより低い価格で使用することでコストを節約できます。**preemptible** を **MachineSet** YAML ファイルに追加して、プリエンプション可能な仮想マシンインスタンスを設定 できます。

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

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

このサンプルでは、**infrastructure_id** はクラスターのプロビジョニング時に設定したクラスター ID に基づくインフラストラクチャー 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: openstackproviderconfig.openshift.io/v1alpha1
          cloudName: openstack
          cloudsSecret:
            name: openstack-cloud-credentials
            namespace: openshift-machine-api
          flavor: <nova_flavor>
          image: <glance_image_name_or_location>
          serverGroupID: <optional_UUID_of_server_group> 12
          kind: OpenstackProviderSpec
          networks: 13
          - filter: {}
            subnets:
            - filter:
                name: <subnet_name>
                tags: openshiftClusterID=<infrastructure_id> 14
          primarySubnet: <rhosp_subnet_UUID> 15
          securityGroups:
          - filter: {}
            name: <infrastructure_id>-worker 16
```

```
      serverMetadata:
        Name: <infrastructure_id>-worker (17)
        openshiftClusterID: <infrastructure_id> (18)
      tags:
      - openshiftClusterID=<infrastructure_id> (19)
      trunk: true
      userDataSecret:
        name: worker-user-data (20)
      availabilityZone: <optional_openstack_availability_zone>
```

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

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

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

(13) 複数ネットワークへのデプロイメントに必要です。複数ネットワークにデプロイする場合、この一覧には、**primarySubnet が** の値として使用されるネットワークが含まれる必要があります。

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

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

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

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

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

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

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

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

2 3 10 11 13 追加するノードラベルを指定します。

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

5 作成するマシンの数を指定します。

6 マシンのセレクター。

14 この仮想マシンインスタンスが属する RHV クラスターの UUID を指定します。

15 マシンの作成に使用する RHV 仮想マシンテンプレートを指定します。

16 オプション: 仮想マシンインスタンスタイプを指定します。

警告

instance_type_id フィールドは非推奨となり、今後のリリースで削除されます。

このパラメーターを含めると、CPU およびメモリーを含む仮想マシンのハードウェアパラメーターを指定する必要はありません。このパラメーターは、すべてのハードウェアパラメーターを上書きするためです。

17 オプション: CPU フィールドには、ソケット、コア、スレッドを含む CPU の設定が含まれます。

18 オプション: 仮想マシンのソケット数を指定します。

19 オプション: ソケットあたりのコア数を指定します。

20 オプション: コアあたりのスレッド数を指定します。

21 オプション: 仮想マシンのメモリーサイズを MiB 単位で指定します。

22 オプション: ノードのルートディスク。

23 オプション: ブート可能なディスクのサイズを GiB 単位で指定します。

24 オプション: 仮想マシンのネットワークインターフェイスの一覧。このパラメーターを含めると、OpenShift Container Platform はテンプレートからすべてのネットワークインターフェイスを破棄し、新規ネットワークインターフェイスを作成します。

25 オプション: vNIC プロファイル ID を指定します。

26 RHV 認証情報を保持するシークレットの名前を指定します。

27 オプション: インスタンスが最適化されるワークロードタイプを指定します。この値は **RHV VM** パラメーターに影響します。サポートされる値: **desktop**、**server** (デフォルト)、**high_performance** です。**high_performance** により、仮想マシンのパフォーマンスが向上しますが、制限があります。たとえば、グラフィカルコンソールを使用して仮想マシンにはアクセスできません。詳細は、**仮想マシン管理ガイド**の ハイパフォーマンス仮想マシン、テンプレート、およびプールの設定 を参照してください。

注記

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

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

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

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

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  creationTimestamp: null
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 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_datacenter_name> 13
```

```
        datastore: <vcenter_datastore_name> 14
        folder: <vcenter_vm_folder_path> 15
        resourcepool: <vsphere_resource_pool> 16
        server: <vcenter_server_ip> 17
```

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

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

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

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

13 マシンセットをデプロイする vCenter Datacenter を指定します。

14 マシンセットをデプロイする vCenter Datastore を指定します。

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

16 仮想マシンの vSphere リソースプールを指定します。

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

7.2.2. マシンセットの作成

インストールプログラムによって作成されるものに加え、独自のマシンセットを作成して、選択する特定のワークロードに対するマシンのコンピュートリソースを動的に管理することができます。

前提条件

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

手順

1. 説明されているようにマシンセット カスタムリソース (CR) サンプルを含む新規 YAML ファイルを作成し、そのファイルに **<file_name>.yaml** という名前を付けます。
   <clusterID> および **<role>** パラメーターの値を設定していることを確認します。

a. 特定のフィールドに設定する値が不明な場合は、クラスターから既存のマシンセットを確認できます。

```
$ 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. 特定のマシンセットの値を確認します。

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

出力例

```
...
template:
  metadata:
    labels:
      machine.openshift.io/cluster-api-cluster: agl030519-vplxk (1)
      machine.openshift.io/cluster-api-machine-role: worker (2)
      machine.openshift.io/cluster-api-machine-type: worker
      machine.openshift.io/cluster-api-machineset: agl030519-vplxk-worker-us-east-1a
```

(1) クラスター ID。

(2) デフォルトのノードラベル。

2. 新規 **MachineSet** CR を作成します。

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

3. マシンセットの一覧を表示します。

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

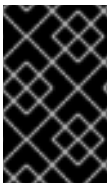
出力例

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

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

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

7.2.3. 専用インフラストラクチャーノードの作成



重要

インストーラーでプロビジョニングされるインフラストラクチャー環境またはコントロールプレーンノード (別名マスターノード) 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は、インフラストラクチャーマシンセットの作成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=""
```

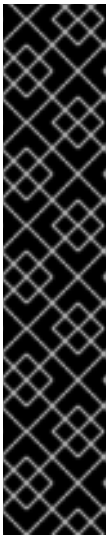
2. インフラストラクチャーノードとして機能する必要のあるワーカーノードにラベルを追加します。

```
$ 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ター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: topology.kubernetes.io/region=us-east-1 1
...
```

1 このサンプルノードセレクターは、デフォルトで **us-east-1** リージョンのノードに Pod をデプロイします。

c. 変更を適用するためにファイルを保存します。

これで、インフラストラクチャ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

7.2.4. 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。

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。

7.3. マシンセットリソースのインフラストラクチャーノードへの割り当て

インフラストラクチャーマシンセットの作成後、**worker** および **infra** ロールが新規の infra ノードに適用されます。**infra** ロールが適用されるノードは、**worker** ロールも適用されている場合でも、環境を実行するために必要なサブスクリプションの合計数にはカウントされません。

ただし、infra ノードがワーカーとして割り当てられると、ユーザーのワークロードが誤って infra ノードに割り当てられる可能性があります。これを回避するには、テイントを、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。

7.3.1. テイントおよび容認を使用したインフラストラクチャーノードのワークロードのバインディング

infra および **worker** ロールが割り当てられている infra ノードがある場合、ユーザーのワークロードがこれに割り当てられないようにノードを設定する必要があります。

重要

infra ノード用に作成されたデュアル **infra,worker** ラベルを保持し、テイントおよび容認 (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>:<effect>
   ```

   以下に例を示します。

   ```
   $ oc adm taint nodes node1 node-role.kubernetes.io/infra:NoSchedule
   ```

この例では、テイントを、キー **node-role.kubernetes.io/infra** およびテイントの effect **NoSchedule** を持つ **node1** に配置します。effect が **NoSchedule** のノードは、テイントを容認する Pod のみをスケジュールしますが、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。

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)
    operator: Exists (3)
```

(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) **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。

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は、スケジューラーを使用した Pod の配置の制御 について参照してください。
- Pod を infra ノードにスケジュールする方法については、リソースのインフラストラクチャーマシンセットへの移動 について参照してください。

7.4. リソースのインフラストラクチャ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。それらは、作成したインフラストラクチャーマシンセットに移行できます。

7.4.1. ルーターの移動

ルーター Pod を異なるマシンセットにデプロイできます。デフォルトで、この Pod はワーカーノードにデプロイされます。

前提条件

- 追加のマシンセットを OpenShift Container Platform クラスターに設定します。

手順

1. ルーター Operator の **IngressController** カスタムリソースを表示します。

```
$ oc get ingresscontroller default -n openshift-ingress-operator -o yaml
```

コマンド出力は以下のテキストのようになります。

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  creationTimestamp: 2019-04-18T12:35:39Z
  finalizers:
  - ingresscontroller.operator.openshift.io/finalizer-ingresscontroller
  generation: 1
  name: default
  namespace: openshift-ingress-operator
  resourceVersion: "11341"
  selfLink: /apis/operator.openshift.io/v1/namespaces/openshift-ingress-
operator/ingresscontrollers/default
  uid: 79509e05-61d6-11e9-bc55-02ce4781844a
spec: {}
status:
  availableReplicas: 2
  conditions:
  - lastTransitionTime: 2019-04-18T12:36:15Z
    status: "True"
    type: Available
  domain: apps.<cluster>.example.com
  endpointPublishingStrategy:
    type: LoadBalancerService
  selector: ingresscontroller.operator.openshift.io/deployment-ingresscontroller=default
```

2. **ingresscontroller** リソースを編集し、 **nodeSelector** を **infra** ラベルを使用するように変更します。

```
$ oc edit ingresscontroller default -n openshift-ingress-operator
```

以下に示すように、**infra** ラベルを参照する **nodeSelector** スタンザを **spec** セクションに追加します。

```
  spec:
    nodePlacement:
      nodeSelector:
```

```
      matchLabels:
        node-role.kubernetes.io/infra: ""
```

3. ルーター Pod が **infra** ノードで実行されていることを確認します。

   a. ルーター Pod の一覧を表示し、実行中の Pod のノード名をメモします。

   ```
   $ oc get pod -n openshift-ingress -o wide
   ```

   出力例

   ```
   NAME                              READY   STATUS        RESTARTS   AGE     IP           NODE
   NOMINATED NODE   READINESS GATES
   router-default-86798b4b5d-bdlvd   1/1     Running       0          28s     10.130.2.4   ip-10-
   0-217-226.ec2.internal   <none>           <none>
   router-default-955d875f4-255g8    0/1     Terminating   0          19h     10.129.2.4   ip-10-
   0-148-172.ec2.internal   <none>           <none>
   ```

   この例では、実行中の Pod は **ip-10-0-217-226.ec2.internal** ノードにあります。

   b. 実行中の Pod のノードのステータスを表示します。

   ```
   $ oc get node <node_name> 1
   ```

   1 Pod の一覧より取得した **<node_name>** を指定します。

   出力例

   ```
   NAME                           STATUS  ROLES        AGE   VERSION
   ip-10-0-217-226.ec2.internal   Ready   infra,worker  17h   v1.20.0
   ```

   ロールの一覧に **infra** が含まれているため、Pod は正しいノードで実行されます。

7.4.2. デフォルトレジストリーの移行

レジストリー Operator を、その Pod を複数の異なるノードにデプロイするように設定します。

前提条件

- 追加のマシンセットを OpenShift Container Platform クラスターに設定します。

手順

1. **config/instance** オブジェクトを表示します。

   ```
   $ oc get configs.imageregistry.operator.openshift.io/cluster -o yaml
   ```

   出力例

   ```
   apiVersion: imageregistry.operator.openshift.io/v1
   kind: Config
   metadata:
   ```

```
  creationTimestamp: 2019-02-05T13:52:05Z
  finalizers:
  - imageregistry.operator.openshift.io/finalizer
  generation: 1
  name: cluster
  resourceVersion: "56174"
  selfLink: /apis/imageregistry.operator.openshift.io/v1/configs/cluster
  uid: 36fd3724-294d-11e9-a524-12ffeee2931b
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
```

3. オブジェクトの **spec** セクションを以下の YAML のように変更します。

```
spec:
  affinity:
    podAntiAffinity:
      preferredDuringSchedulingIgnoredDuringExecution:
      - podAffinityTerm:
          namespaces:
          - openshift-image-registry
          topologyKey: kubernetes.io/hostname
        weight: 100
  logLevel: Normal
  managementState: Managed
  nodeSelector:
    node-role.kubernetes.io/infra: ""
```

4. レジストリー Pod がインフラストラクチャーノードに移動していることを確認します。

   a. 以下のコマンドを実行して、レジストリー Pod が置かれているノードを特定します。

   ```
   $ oc get pods -o wide -n openshift-image-registry
   ```

   b. ノードに指定したラベルがあることを確認します。

   ```
   $ oc describe node <node_name>
   ```

コマンド出力を確認し、**node-role.kubernetes.io/infra** が **LABELS** 一覧にあることを確認します。

7.4.3. モニターリングソリューションの移動

デフォルトでは、Prometheus、Grafana、および AlertManager が含まれる Prometheus Cluster Monitoring スタックはクラスターモニターリングをデプロイするためにデプロイされます。これは Cluster Monitoring Operator によって管理されます。このコンポーネント異なるマシンに移行するには、カスタム設定マップを作成し、これを適用します。

手順

1. 以下の **ConfigMap** 定義を **cluster-monitoring-configmap.yaml** ファイルとして保存します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |+
    alertmanagerMain:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    prometheusK8s:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    prometheusOperator:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    grafana:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    k8sPrometheusAdapter:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    kubeStateMetrics:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    telemeterClient:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    openshiftStateMetrics:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    thanosQuerier:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
```

この設定マップを実行すると、モニタリングスタックのコンポーネントがインフラストラクチャーノードに再デプロイされます。

2. 新規の設定マップを適用します。

```
$ oc create -f cluster-monitoring-configmap.yaml
```

3. 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'
```

4. コンポーネントが **infra** ノードに移動していない場合は、このコンポーネントを持つ Pod を削除します。

```
$ oc delete pod -n openshift-monitoring <pod>
```

削除された Pod からのコンポーネントが **infra** ノードに再作成されます。

7.4.4. OpenShift Logging リソースの移動

Elasticsearch および Kibana などの OpenShift Logging コンポーネントの Pod を異なるノードにデプロイするように Cluster Logging Operator を設定できます。Cluster Logging Operator Pod については、インストールされた場所から移動することはできません。

たとえば、Elasticsearch Pod の CPU、メモリーおよびディスクの要件が高いために、この Pod を別のノードに移動できます。

前提条件

- OpenShift Logging および Elasticsearch がインストールされている。これらの機能はデフォルトでインストールされません。

手順

1. **openshift-logging** プロジェクトで **ClusterLogging** カスタムリソース (CR) を編集します。

```
$ oc edit ClusterLogging instance
```

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging

...

spec:
  collection:
    logs:
      fluentd:
        resources: null
      type: fluentd
  logStore:
    elasticsearch:
      nodeCount: 3
      nodeSelector: 1
        node-role.kubernetes.io/infra: ''
      redundancyPolicy: SingleRedundancy
      resources:
        limits:
          cpu: 500m
          memory: 16Gi
        requests:
          cpu: 500m
```

```
        memory: 16Gi
    storage: {}
  type: elasticsearch
managementState: Managed
visualization:
  kibana:
    nodeSelector: 2
      node-role.kubernetes.io/infra: ''
    proxy:
      resources: null
    replicas: 1
    resources: null
  type: kibana

...
```

1 2 適切な値が設定された **nodeSelector** パラメーターを、移動する必要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。

検証

コンポーネントが移動したことを確認するには、**oc get pod -o wide** コマンドを使用できます。

以下に例を示します。

- Kibana Pod を **ip-10-0-147-79.us-east-2.compute.internal** ノードから移動する必要がある場合、以下を実行します。

```
$ oc get pod kibana-5b8bdf44f9-ccpq9 -o wide
```

出力例

```
NAME                     READY   STATUS    RESTARTS   AGE   IP            NODE
NOMINATED NODE   READINESS GATES
kibana-5b8bdf44f9-ccpq9   2/2     Running   0          27s   10.129.2.18   ip-10-0-147-79.us-
east-2.compute.internal   <none>           <none>
```

- Kibana Pod を、専用インフラストラクチャーノードである **ip-10-0-139-48.us-east-2.compute.internal** ノードに移動する必要がある場合、以下を実行します。

```
$ oc get nodes
```

出力例

```
NAME                                         STATUS   ROLES          AGE   VERSION
ip-10-0-133-216.us-east-2.compute.internal   Ready    master         60m   v1.20.0
ip-10-0-139-146.us-east-2.compute.internal   Ready    master         60m   v1.20.0
ip-10-0-139-192.us-east-2.compute.internal   Ready    worker         51m   v1.20.0
ip-10-0-139-241.us-east-2.compute.internal   Ready    worker         51m   v1.20.0
ip-10-0-147-79.us-east-2.compute.internal    Ready    worker         51m   v1.20.0
ip-10-0-152-241.us-east-2.compute.internal   Ready    master         60m   v1.20.0
ip-10-0-139-48.us-east-2.compute.internal    Ready    infra          51m   v1.20.0
```

ノードには **node-role.kubernetes.io/infra: ''** ラベルがあることに注意してください。

```
$ oc get node ip-10-0-139-48.us-east-2.compute.internal -o yaml
```

出力例

```
kind: Node
apiVersion: v1
metadata:
  name: ip-10-0-139-48.us-east-2.compute.internal
  selfLink: /api/v1/nodes/ip-10-0-139-48.us-east-2.compute.internal
  uid: 62038aa9-661f-41d7-ba93-b5f1b6ef8751
  resourceVersion: '39083'
  creationTimestamp: '2020-04-13T19:07:55Z'
  labels:
    node-role.kubernetes.io/infra: ''
...
```

- Kibana Pod を移動するには、**ClusterLogging** CR を編集してノードセレクターを追加します。

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging

...

spec:

...

  visualization:
    kibana:
      nodeSelector: 1
        node-role.kubernetes.io/infra: ''
      proxy:
        resources: null
      replicas: 1
      resources: null
    type: kibana
```

1 ノード仕様のラベルに一致するノードセレクターを追加します。

- CR を保存した後に、現在の Kibana Pod は終了し、新規 Pod がデプロイされます。

```
$ oc get pods
```

出力例

```
NAME                                  READY   STATUS      RESTARTS   AGE
cluster-logging-operator-84d98649c4-zb9g7       1/1     Running       0          29m
elasticsearch-cdm-hwv01pf7-1-56588f554f-kpmlg   2/2     Running       0          28m
elasticsearch-cdm-hwv01pf7-2-84c877d75d-75wqj   2/2     Running       0          28m
elasticsearch-cdm-hwv01pf7-3-f5d95b87b-4nx78    2/2     Running       0          28m
fluentd-42dzz                                   1/1     Running       0          28m
```

```
fluentd-d74rq                           1/1     Running       0          28m
fluentd-m5vr9                           1/1     Running       0          28m
fluentd-nkxl7                           1/1     Running       0          28m
fluentd-pdvqb                           1/1     Running       0          28m
fluentd-tflh6                           1/1     Running       0          28m
kibana-5b8bdf44f9-ccpq9                 2/2     Terminating   0          4m11s
kibana-7d85dcffc8-bfpfp                 2/2     Running       0          33s
```

- 新規 Pod が **ip-10-0-139-48.us-east-2.compute.internal** ノードに置かれます。

```
$ oc get pod kibana-7d85dcffc8-bfpfp -o wide
```

出力例

```
NAME                    READY   STATUS       RESTARTS   AGE   IP          NODE
NOMINATED NODE   READINESS GATES
kibana-7d85dcffc8-bfpfp   2/2     Running      0          43s   10.131.0.22   ip-10-0-139-48.us-
east-2.compute.internal   <none>          <none>
```

- しばらくすると、元の Kibana Pod が削除されます。

```
$ oc get pods
```

出力例

```
NAME                                READY   STATUS    RESTARTS   AGE
cluster-logging-operator-84d98649c4-zb9g7       1/1     Running   0          30m
elasticsearch-cdm-hwv01pf7-1-56588f554f-kpmlg   2/2     Running   0          29m
elasticsearch-cdm-hwv01pf7-2-84c877d75d-75wqj   2/2     Running   0          29m
elasticsearch-cdm-hwv01pf7-3-f5d95b87b-4nx78    2/2     Running   0          29m
fluentd-42dzz                           1/1     Running   0          29m
fluentd-d74rq                           1/1     Running   0          29m
fluentd-m5vr9                           1/1     Running   0          29m
fluentd-nkxl7                           1/1     Running   0          29m
fluentd-pdvqb                           1/1     Running   0          29m
fluentd-tflh6                           1/1     Running   0          29m
kibana-7d85dcffc8-bfpfp                 2/2     Running   0          62s
```

関連情報

- OpenShift Container Platform コンポーネントの移動についての一般的な情報は、モニターリングについてのドキュメント を参照してください。

第8章 RHEL コンピュートマシンの OPENSHIFT CONTAINER PLATFORM クラスターへの追加

OpenShift Container Platform では、Red Hat Enterprise Linux (RHEL) のコンピュートまたはワーカーマシンを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。

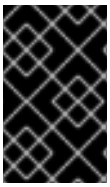
8.1. RHEL コンピュートノードのクラスターへの追加について

OpenShift Container Platform 4.7 には、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を使用する場合、Red Hat Enterprise Linux (RHEL) マシンをクラスター内のコンピュートまたはワーカーマシンとして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またはマスターマシンには Red Hat Enterprise Linux CoreOS (RHCOS) マシンを使用する必要があります。

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を使用するすべてのインストールの場合、クラスターで RHEL コンピュートマシンを使用する選択をする場合には、システム更新の実行や、パッチの適用、またその他の必要なすべてのタスクの実行を含むオペレーティングシステムのライフサイクル管理およびメンテナンスのすべてを独自に実行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。



重要

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。

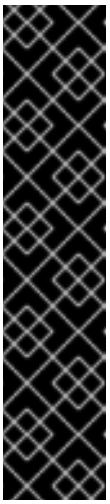
RHEL コンピュートマシンは、コントロールプレーンを初期化してからクラスターに追加する必要があります。

8.2. 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) コンピュートまたはワーカー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 パーセント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: RHEL 7.9 (最小のインストールオプション)。



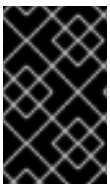
重要

RHEL 7 コンピュートマシンの OpenShift Container Platform クラスターへの追加は非推奨となりました。非推奨の機能は依然として OpenShift Container Platform に含まれており、引き続きサポートされますが、本製品の今後のリリースで削除されるため、新規デプロイメントでの使用は推奨されません。

また、本リリースではサポートされていないため、コンピュートマシンを RHEL 8 にアップグレードすることはできません。

OpenShift Container Platform で非推奨となったか、または削除された主な機能の最新の一覧については、OpenShift Container Platform リリースノートの **非推奨および削除された機能**セクションを参照してください。

- FIPS モードで OpenShift Container Platform をデプロイしている場合、起動する前に FIPS を RHEL マシン上で有効にする必要があります。詳細は、RHEL 7 のドキュメントの FIPS モードの有効化 を参照してください。



重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

- 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。
- システムの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。

8.2.1. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズム

を提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

8.3. クラウド用イメージの準備

各種のイメージ形式は AWS で直接使用できないので、Amazon Machine Images (AMI) が必要です。Red Hat が提供している AMI を使用するか、または独自のイメージを手動でインポートできます。EC2 インスタンスをプロビジョニングする前に AMI が存在している必要があります。コンピュー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。

8.3.1. AWS で利用可能な最新の RHEL イメージの一覧表示

AMI ID は、AWS のネイティブブートイメージに対応します。EC2 インスタンスがプロビジョニングされる前に AMI が存在している必要があるため、設定前に AMI ID を把握しておく必要があります。AWS コマンドラインインターフェイス (CLI) は、利用可能な Red Hat Enterprise Linux (RHEL) イメージ ID の一覧を表示するために使用されます。

前提条件

- AWS CLI をインストールしている。

手順

- このコマンドを使用して、RHEL 7.9 Amazon Machine Images (AMI) の一覧を表示します。

```
$ aws ec2 describe-images --owners 309956199498 \ 1
--query 'sort_by(Images, &CreationDate)[*].[CreationDate,Name,ImageId]' \ 2
--filters "Name=name,Values=RHEL-7.9*" \ 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-7.9*"** で設定されているため、RHEL 7.9 AMI が表示されます。
4. **--region** コマンドオプションは、AMI が保存されるリージョンを設定します。
5. **--output** コマンドオプションは、結果の表示方法を設定します。

注記

AWS 用の RHEL コンピュートマシンを作成する場合、AMI が RHEL 7.9 であることを確認します。

出力例

```
----------------------------------------------------------------------------------------------------
|                                          DescribeImages                                          |
+--------------------------+-----------------------------------------------+----------------------+
|  2020-05-13T09:50:36.000Z|  RHEL-7.9_HVM_BETA-20200422-x86_64-0-Hourly2-GP2  |  ami-038714142142a6a64 |
|  2020-09-18T07:51:03.000Z|  RHEL-7.9_HVM_GA-20200917-x86_64-0-Hourly2-GP2    |  ami-005b7876121b7244d |
|  2021-02-09T09:46:19.000Z|  RHEL-7.9_HVM-20210208-x86_64-0-Hourly2-GP2       |  ami-030e754805234517e |
+--------------------------+-----------------------------------------------+----------------------+
```

関連情報

- RHEL イメージを AWS に手動でインポートする こともできます。

8.4. PLAYBOOK 実行のためのマシンの準備

Red Hat Enterprise Linux (RHEL) をオペレーティングシステムとして使用するコンピュートマシンを OpenShift Container Platform 4.7 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 7 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** パーミッションを持つユーザーとしてログインします。

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一覧表示します。

   ```
   # subscription-manager list --available --matches '*OpenShift*'
   ```

   d. 直前のコマンドの出力で、OpenShift Container Platform サブスクリプションのプール ID を見つけ、これをアタッチします。

   ```
   # subscription-manager attach --pool=<pool_id>
   ```

5. OpenShift Container Platform 4.7 で必要なリポジトリーを有効にします。

```
# subscription-manager repos \
    --enable="rhel-7-server-rpms" \
    --enable="rhel-7-server-extras-rpms" \
    --enable="rhel-7-server-ansible-2.9-rpms" \
    --enable="rhel-7-server-ose-4.7-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。

8.5. 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一覧表示します。

```
# subscription-manager list --available --matches '*OpenShift*'
```

4. 直前のコマンドの出力で、OpenShift Container Platform サブスクリプションのプール ID を見つけ、これをアタッチします。

```
# subscription-manager attach --pool=<pool_id>
```

5. yum リポジトリ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 リポジトリーを一覧表示し、**repo id** にあるそれらの名前をメモします (ある場合)。

```
# yum repolist
```

   c. **yum-config-manager** を使用して、残りの yum リポジトリーを無効にします。

```
# 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 4.7 で必要なリポジトリーのみを有効にします。

```
# subscription-manager repos \
    --enable="rhel-7-server-rpms" \
    --enable="rhel-7-fast-datapath-rpms" \
    --enable="rhel-7-server-extras-rpms" \
    --enable="rhel-7-server-optional-rpms" \
    --enable="rhel-7-server-ose-4.7-rpms"
```

7. ホストで firewalld を停止し、無効にします。

```
# systemctl disable --now firewalld.service
```

注記

firewalld は、後で有効にすることはできません。これを実行する場合、ワーカー上の OpenShift Container Platform ログにはアクセスできません。

8.6. AWS での RHEL インスタンスへのロールパーミッションの割り当て

ブラウザーで Amazon IAM コンソールを使用して、必要なロールを選択し、ワーカーノードに割り当てることができます。

手順

1. AWS IAM コンソールから、任意の IAM ロール を作成します。
2. IAM ロール を必要なワーカーノードに割り当てます。

関連情報

- IAM ロールに必要な AWS パーミッション について参照してください。

8.7. 所有または共有されている RHEL ワーカーノードへのタグ付け

クラスターは **kubernetes.io/cluster/<clusterid>,Value=(owned|shared)** タグの値を使用して、AWS クラスターに関するリソースの有効期間を判別します。

- リソースをクラスターの破棄の一環として破棄する必要がある場合は、**owned** タグの値を追加する必要があります。
- クラスターが破棄された後にリソースが引き続いて存在する場合、**shared** タグの値を追加する必要があります。このタグ付けは、クラスターがこのリソースを使用することを示しますが、リソースには別の所有者が存在します。

手順

- RHEL コンピュートマシンの場合、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。

8.8. 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 4.7 クラスターに追加することができます。

前提条件

- Playbook を実行するマシンに必要なパッケージをインストールし、必要な設定が行われています。
- インストール用の RHEL ホストを準備しています。

手順

Playbook を実行するために準備しているマシンで以下の手順を実行します。

1. コンピュー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-rhel7-0.example.com
mycluster-rhel7-1.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 マシンを一覧表示します。各ホストについて完全修飾ドメイン名を指定する必要があります。この名前は、クラスターがマシンにアクセスするために使用するホスト名であるため、マシンにアクセスできるように正しいパブリックまたはプライベートの名前を設定します。

2. Ansible Playbook ディレクトリーに移動します。

```
$ cd /usr/share/ansible/openshift-ansible
```

3. Playbook を実行します。

```
$ ansible-playbook -i /<path>/inventory/hosts playbooks/scaleup.yml 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。

8.9. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

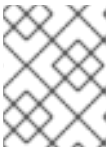
1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME     STATUS   ROLES   AGE  VERSION
master-0 Ready    master  63m  v1.20.0
master-1 Ready    master  63m  v1.20.0
master-2 Ready    master  64m  v1.20.0
```

出力には作成したすべてのマシンが一覧表示されます。



注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

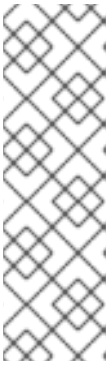
```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

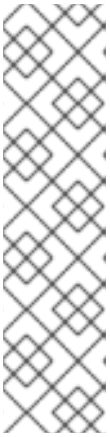
この例では、2 つのマシンがクラスターに参加しています。この一覧にはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。



注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認されたら、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、手動の承認が必要です。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。



注記

ベアメタルおよび他のユーザーによってプロビジョニングされるインフラストラクチャーなどのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、 **oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

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

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

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

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

注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

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

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

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

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  v1.20.0
master-1  Ready     master  73m  v1.20.0
master-2  Ready     master  74m  v1.20.0
worker-0  Ready     worker  11m  v1.20.0
worker-1  Ready     worker  11m  v1.20.0
```

注記

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

関連情報

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

8.10. ANSIBLE ホストファイルの必須パラメーター

Red Hat Enterprise Linux (RHEL) コンピュー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。

8.10.1. 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. マシンの一覧を表示し、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一覧を確認し、RHEL ノードのみが残っていることを確認します。

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

4. RHCOS マシンをクラスターのコンピュー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9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。

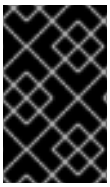
9.1. RHEL コンピュートノードのクラスターへの追加について

OpenShift Container Platform 4.7 には、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を使用する場合、Red Hat Enterprise Linux (RHEL) マシンをクラスター内のコンピュートまたはワーカーマシンとして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またはマスターマシンには Red Hat Enterprise Linux CoreOS (RHCOS) マシンを使用する必要があります。

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を使用するすべてのインストールの場合、クラスターで RHEL コンピュートマシンを使用する選択をする場合には、システム更新の実行や、パッチの適用、またその他の必要なすべてのタスクの実行を含むオペレーティングシステムのライフサイクル管理およびメンテナンスのすべてを独自に実行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。



重要

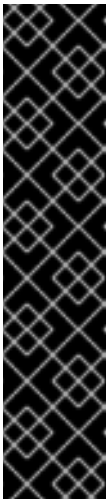
swap メモリーは、OpenShift Container Platform クラスターに追加されるすべての RHEL マシンで無効にされます。これらのマシンで swap メモリーを有効にすることはできません。

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) コンピュートまたはワーカーマシンは以下の最低のハードウェア仕様およびシステムレベルの要件を満たしている必要があります。

- まず、お使いの Red Hat アカウントに有効な OpenShift Container Platform サブスクリプションがなければなりません。これがない場合は、営業担当者にお問い合わせください。
- 実稼働環境では予想されるワークロードに対応するコンピューターノードを提供する必要があります。クラスター管理者は、予想されるワークロードを計算し、オーバーヘッドの約 10 パーセント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: RHEL 7.9 (最小のインストールオプション)。



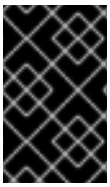
重要

RHEL 7 コンピュートマシンの OpenShift Container Platform クラスターへの追加は非推奨となりました。非推奨の機能は依然として OpenShift Container Platform に含まれており、引き続きサポートされますが、本製品の今後のリリースで削除されるため、新規デプロイメントでの使用は推奨されません。

また、本リリースではサポートされていないため、コンピュートマシンを RHEL 8 にアップグレードすることはできません。

OpenShift Container Platform で非推奨となったか、または削除された主な機能の最新の一覧については、OpenShift Container Platform リリースノートの **非推奨および削除された機能**セクションを参照してください。

- FIPS モードで OpenShift Container Platform をデプロイしている場合、起動する前に FIPS を RHEL マシン上で有効にする必要があります。詳細は、RHEL 7 のドキュメントの FIPS モードの有効化 を参照してください。



重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

- 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。
- システムの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。

9.2.1. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認しま

す。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

9.3. クラウド用イメージの準備

各種のイメージ形式は AWS で直接使用できないので、Amazon Machine Images (AMI) が必要です。Red Hat が提供している AMI を使用するか、または独自のイメージを手動でインポートできます。EC2 インスタンスをプロビジョニングする前に AMI が存在している必要があります。コンピュー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 を一覧表示する必要があります。

9.3.1. AWS で利用可能な最新の RHEL イメージの一覧表示

AMI ID は、AWS のネイティブブートイメージに対応します。EC2 インスタンスがプロビジョニングされる前に AMI が存在している必要があるため、設定前に AMI ID を把握しておく必要があります。AWS コマンドラインインターフェイス (CLI) は、利用可能な Red Hat Enterprise Linux (RHEL) イメージ ID の一覧を表示するために使用されます。

前提条件

- AWS CLI をインストールしている。

手順

- このコマンドを使用して、RHEL 7.9 Amazon Machine Images (AMI) の一覧を表示します。

```
$ aws ec2 describe-images --owners 309956199498 \ 1
--query 'sort_by(Images, &CreationDate)[*].[CreationDate,Name,ImageId]' \ 2
--filters "Name=name,Values=RHEL-7.9*" \ 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-7.9*"** で設定されているため、RHEL 7.9 AMI が表示されます。
4. **--region** コマンドオプションは、AMI が保存されるリージョンを設定します。
5. **--output** コマンドオプションは、結果の表示方法を設定します。

注記

AWS 用の RHEL コンピュートマシンを作成する場合、AMI が RHEL 7.9 であることを確認します。

出力例

```
----------------------------------------------------------------------------------------------------
|                                          DescribeImages                                          |
+---------------------------+------------------------------------------------+---------------------+
|  2020-05-13T09:50:36.000Z |  RHEL-7.9_HVM_BETA-20200422-x86_64-0-Hourly2-GP2  |  ami-038714142142a6a64 |
|  2020-09-18T07:51:03.000Z |  RHEL-7.9_HVM_GA-20200917-x86_64-0-Hourly2-GP2    |  ami-005b7876121b7244d |
|  2021-02-09T09:46:19.000Z |  RHEL-7.9_HVM-20210208-x86_64-0-Hourly2-GP2       |  ami-030e754805234517e |
+---------------------------+------------------------------------------------+---------------------+
```

関連情報

- RHEL イメージを AWS に手動でインポートする こともできます。

9.4. RHEL コンピュートノードの準備

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一覧表示します。

```
# subscription-manager list --available --matches '*OpenShift*'
```

4. 直前のコマンドの出力で、OpenShift Container Platform サブスクリプションのプール ID を見つけ、これをアタッチします。

```
# subscription-manager attach --pool=<pool_id>
```

5. yum リポジトリ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 リポジトリーを一覧表示し、**repo id** にあるそれらの名前をメモします (ある場合)。

```
# yum repolist
```

c. **yum-config-manager** を使用して、残りの yum リポジトリーを無効にします。

```
# 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 \*
```

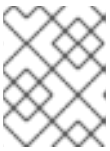
利用可能なリポジトリーが多い場合には、数分の時間がかかることがあります。

6. OpenShift Container Platform 4.7 で必要なリポジトリーのみを有効にします。

```
# subscription-manager repos \
    --enable="rhel-7-server-rpms" \
    --enable="rhel-7-fast-datapath-rpms" \
    --enable="rhel-7-server-extras-rpms" \
    --enable="rhel-7-server-optional-rpms" \
    --enable="rhel-7-server-ose-4.7-rpms"
```

7. ホストで firewalld を停止し、無効にします。

```
# systemctl disable --now firewalld.service
```



注記

firewalld は、後で有効にすることはできません。これを実行する場合、ワーカー上の OpenShift Container Platform ログにはアクセスできません。

9.5. AWS での RHEL インスタンスへのロールパーミッションの割り当て

ブラウザーで Amazon IAM コンソールを使用して、必要なロールを選択し、ワーカーノードに割り当てることができます。

手順

1. AWS IAM コンソールから、任意の IAM ロール を作成します。
2. IAM ロール を必要なワーカーノードに割り当てます。

関連情報

- IAM ロールに必要な AWS パーミッション について参照してください。

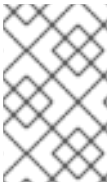
9.6. 所有または共有されている RHEL ワーカーノードへのタグ付け

クラスターは **kubernetes.io/cluster/<clusterid>,Value=(owned|shared)** タグの値を使用して、AWS クラスターに関するリソースの有効期間を判別します。

- リソースをクラスターの破棄の一環として破棄する必要がある場合は、**owned** タグの値を追加する必要があります。
- クラスターが破棄された後にリソースが引き続いて存在する場合、**shared** タグの値を追加する必要があります。このタグ付けは、クラスターがこのリソースを使用することを示しますが、リソースには別の所有者が存在します。

手順

- RHEL コンピュートマシンの場合、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。

9.7. 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 4.7 クラスターにさらに追加することができます。

前提条件

- OpenShift Container Platform クラスターに RHEL コンピュー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-rhel7-0.example.com
mycluster-rhel7-1.example.com

[new_workers]
mycluster-rhel7-2.example.com
mycluster-rhel7-3.example.com
```

この例では、**mycluster-rhel7-0.example.com** および **mycluster-rhel7-1.example.com** マシンがクラスターにあり、**mycluster-rhel7-2.example.com** および **mycluster-rhel7-3.example.com** マシンを追加します。

4. Ansible Playbook ディレクトリーに移動します。

```
$ cd /usr/share/ansible/openshift-ansible
```

5. スケールアップ Playbook を実行します。

```
$ ansible-playbook -i /<path>/inventory/hosts playbooks/scaleup.yml 1
```

1 **<path>** については、作成した Ansible インベントリーファイルへのパスを指定します。

9.8. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

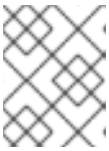
```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.20.0
```

```
master-1  Ready    master  63m  v1.20.0
master-2  Ready    master  64m  v1.20.0
```

出力には作成したすべてのマシンが一覧表示されます。



注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

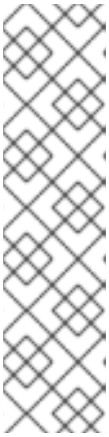
```
NAME        AGE     REQUESTOR                                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。この一覧にはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認されたら、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、手動の承認が必要です。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。



注記

ベアメタルおよび他のユーザーによってプロビジョニングされるインフラストラクチャーなどのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、 **oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

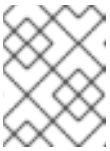
- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

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

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

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



注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

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

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

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

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  v1.20.0
master-1  Ready     master  73m  v1.20.0
master-2  Ready     master  74m  v1.20.0
worker-0  Ready     worker  11m  v1.20.0
worker-1  Ready     worker  11m  v1.20.0
```

注記

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

関連情報

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

9.9. ANSIBLE ホストファイルの必須パラメーター

Red Hat Enterprise Linux (RHEL) コンピュー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。

パラメーター	説明	値
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10章 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

10.1. 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を使用したクラスターへのコンピュートマシンの追加

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。

10.1.1. 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。

10.1.2. 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追加するには、Creating additional worker machines in Azure を参照してください。

10.1.3. コンピュートマシンの 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追加するには、Creating additional worker machines in GCP を参照してください。

10.1.4. コンピュートマシンの vSphere への追加

vSphere 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 vSphere への追加 を参照してください。

10.1.5. 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。

10.2. 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。

10.2.1. 前提条件

- 提供される AWS CloudFormation テンプレート を使用して AWS にクラスターをインストールしている。
- クラスターのインストール時にコンピュー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 に従ってこれらを作成する必要があります。

10.2.2. CloudFormation テンプレートの使用によるコンピュートマシンの 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。

10.2.3. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

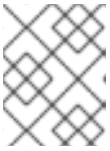
1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.20.0
master-1  Ready     master  63m  v1.20.0
master-2  Ready     master  64m  v1.20.0
```

出力には作成したすべてのマシンが一覧表示されます。



注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

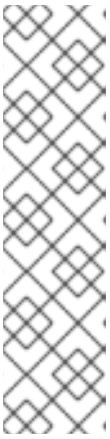
```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。この一覧にはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認されたら、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、手動の承認が必要です。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。



注記

ベアメタルおよび他のユーザーによってプロビジョニングされるインフラストラクチャーなどのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、 **oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

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

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

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

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

注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
```

```
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

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

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

     1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

   - すべての保留中の CSR を承認するには、以下のコマンドを実行します。

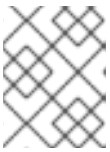
     ```
     $ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
     {{end}}{{end}}' | xargs oc adm certificate approve
     ```

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  v1.20.0
   master-1  Ready     master  73m  v1.20.0
   master-2  Ready     master  74m  v1.20.0
   worker-0  Ready     worker  11m  v1.20.0
   worker-1  Ready     worker  11m  v1.20.0
   ```



注記

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

関連情報

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

10.3. コンピュートマシンの VSPHERE への追加

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。

10.3.1. 前提条件

- クラスターを vSphere にインストールしている。

- クラスターの作成に使用したインストールメディアおよび Red Hat Enterprise Linux CoreOS (RHCOS) イメージがある。これらのファイルがない場合は、インストール手順 に従ってこれらを取得する必要があります。

重要

クラスターの作成に使用された Red Hat Enterprise Linux CoreOS (RHCOS) イメージへのアクセスがない場合、より新しいバージョンの Red Hat Enterprise Linux CoreOS (RHCOS) イメージと共にコンピュートマシンを 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。

10.3.2. vSphere での追加の Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成

VMware vSphere で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を使用するクラスターのコンピュートマシンを追加で作成できます。

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. テンプレートがデプロイされた後に、マシンの仮想マシンをクラスターにデプロイします。
   a. テンプレートの名前を右クリックし、**Clone → Clone to Virtual Machine**をクリックします。
   b. **Select a name and folder**タブで、仮想マシンの名前を指定します。**compute-1** などのように、マシンタイプを名前に含めることができるかもしれません。
   c. **Select a name and folder**タブで、クラスターに作成したフォルダーの名前を選択します。
   d. **Select a compute resource** タブで、データセンター内のホストの名前を選択します。
   e. オプション: **Select storage** タブで、ストレージオプションをカスタマイズします。
   f. **Select clone options** で、**Customize this virtual machine's hardware**を選択します。
   g. **Customize hardware** タブで、**VM Options → Advanced** をクリックします。
      - **Latency Sensitivity** 一覧から、**High** を選択します。
      - **Edit Configuration** をクリックし、**Configuration Parameters** ウィンドウで **Add Configuration Params** をクリックします。以下のパラメーター名および値を定義します。
        - **guestinfo.ignition.config.data**: このマシンファイルの base64 でエンコードしたコンピュート Ignition 設定ファイルの内容を貼り付けます。
        - **guestinfo.ignition.config.data.encoding**: **base64** を指定します。
        - **disk.EnableUUID**: **TRUE** を指定します。

h. **Customize hardware** タブの **Virtual Hardware** パネルで、必要に応じて指定した値を変更します。RAM、CPU、およびディスクストレージの量がマシンタイプの最小要件を満たすことを確認してください。また、ネットワークが複数利用可能な場合は、必ず **Add network adapter** に正しいネットワークを選択してください。

i. 設定を完了し、仮想マシンの電源をオンにします。

2. 継続してクラスター用の追加のコンピュートマシンを作成します。

10.3.3. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.20.0
master-1  Ready     master  63m  v1.20.0
master-2  Ready     master  64m  v1.20.0
```

出力には作成したすべてのマシンが一覧表示されます。

注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

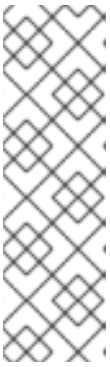
出力例

```
NAME        AGE     REQUESTOR                                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
```

```
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
...
```

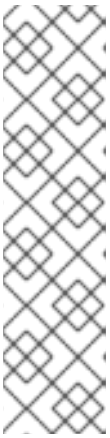
この例では、2 つのマシンがクラスターに参加しています。この一覧にはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。



注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認されたら、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、手動の承認が必要です。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。



注記

ベアメタルおよび他のユーザーによってプロビジョニングされるインフラストラクチャーなどのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、 **oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

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

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

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

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

注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

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

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

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

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  v1.20.0
master-1  Ready     master  73m  v1.20.0
master-2  Ready     master  74m  v1.20.0
worker-0  Ready     worker  11m  v1.20.0
worker-1  Ready     worker  11m  v1.20.0
```

注記

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

関連情報

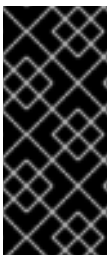
- CSR の詳細は、Certificate Signing Requests を参照してください。

10.4. 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。

10.4.1. 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 に従ってこれらを取得する必要があります。

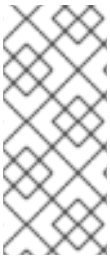


重要

クラスターの作成に使用された Red Hat Enterprise Linux CoreOS (RHCOS) イメージへのアクセスがない場合、より新しいバージョンの Red Hat Enterprise Linux CoreOS (RHCOS) イメージと共にコンピュートマシンを 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。

10.4.2. 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。

10.4.2.1. ISO イメージを使用した追加の RHCOS マシンの作成

ISO イメージを使用して、ベアメタルクラスターの追加の Red Hat Enterprise Linux CoreOS (RHCOS) コンピュー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。

手順

1. 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。

2. インスタンスの起動後に、**TAB** または **E** キーを押してカーネルコマンドラインを編集します。

3. パラメーターをカーネルコマンドラインに追加します。

```
coreos.inst.install_dev=sda 1
coreos.inst.ignition_url=http://example.com/worker.ign 2
```

1 インストール先のシステムのブロックデバイスを指定します。

2 コンピュート Ignition 設定ファイルの URL を指定します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。

4. **Enter** を押してインストールを完了します。RHCOS のインストール後に、システムは再起動します。システムの再起動後、指定した Ignition 設定ファイルを適用します。

5. 継続してクラスター用の追加のコンピュートマシンを作成します。

10.4.2.2. PXE または iPXE ブートによる追加の RHCOS マシンの作成

PXE または iPXE ブートを使用して、ベアメタルクラスターの追加の Red Hat Enterprise Linux CoreOS (RHCOS) コンピュー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。

この設定では、グラフィカルコンソールを使用するマシンでシリアルコンソールアクセスを有効にしません。別のコンソールを設定するには、**APPEND** 行に 1 つ以上の **console=** 引数を追加します。たとえば、**console=tty0 console=ttyS0** を追加して、最初の PC シリアルポートをプライマリーコンソールとして、グラフィカルコンソールをセカンダリーコンソールとして設定します。詳細は、How does one set up a serial terminal and/or console in Red Hat Enterprise Linux? を参照してください。

- iPXE の場合:

```
kernel http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> initrd=main
coreos.inst.install_dev=/dev/sda coreos.inst.ignition_url=http://<HTTP_server>/worker.ign
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.<architecture>.img
1
initrd --name main http://<HTTP_server>/rhcos-<version>-live-initramfs.<architecture>.img
2
```

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

この設定では、グラフィカルコンソールを使用するマシンでシリアルコンソールアクセスを有効にしません。別のコンソールを設定するには、**kernel** 行に **console=** 引数を 1 つ以上追加します。たとえば、**console=tty0 console=ttyS0** を追加して、最初の PC シリアルポートをプライマリーコンソールとして、グラフィカルコンソールをセカンダリーコンソールとして設定します。詳細は、How does one set up a serial terminal and/or console in Red Hat Enterprise Linux? を参照してください。

1. PXE または iPXE インフラストラクチャーを使用して、クラスターに必要なコンピュートマシンを作成します。

10.4.3. 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、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.20.0
master-1  Ready     master  63m  v1.20.0
master-2  Ready     master  64m  v1.20.0
```

出力には作成したすべてのマシンが一覧表示されます。

注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

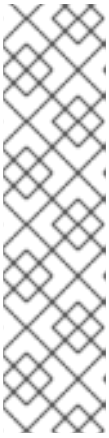
```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。この一覧にはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認されたら、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、手動の承認が必要です。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。



注記

ベアメタルおよび他のユーザーによってプロビジョニングされるインフラストラクチャーなどのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、 **oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

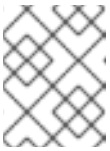
- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

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

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

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



注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

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

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

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

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  v1.20.0
master-1  Ready     master  73m  v1.20.0
master-2  Ready     master  74m  v1.20.0
worker-0  Ready     worker  11m  v1.20.0
worker-1  Ready     worker  11m  v1.20.0
```

注記

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

関連情報

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

第11章 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

このプロセスは、手動でプロビジョニングされたマシンを持つクラスターには適用されません。高度なマシン管理およびスケーリング機能は、マシン API が機能しているクラスターでのみ使用することができます。

11.1. マシンのヘルスチェック

マシンのヘルスチェックは特定のマシンプールの正常ではないマシンを自動的に修復します。

マシンの正常性を監視するには、リソースを作成し、コントローラーの設定を定義します。5 分間 **NotReady** ステータスにすることや、 node-problem-detector に永続的な条件を表示すること、および監視する一連のマシンのラベルなど、チェックする条件を設定します。

注記

マスターロールのあるマシンにマシンヘルスチェックを適用することはできません。

MachineHealthCheck リソースを監視するコントローラーは定義済みのステータスをチェックします。マシンがヘルスチェックに失敗した場合、このマシンは自動的に検出され、その代わりとなるマシンが作成されます。マシンが削除されると、**machine deleted** イベントが表示されます。

マシンの削除による破壊的な影響を制限するために、コントローラーは 1 度に 1 つのノードのみをドレイン (解放) し、これを削除します。マシンのターゲットプールで許可される **maxUnhealthy** しきい値を上回る数の正常でないマシンがある場合、修復が停止するため、手動による介入が可能になります。

注記

タイムアウトについて注意深い検討が必要であり、ワークロードと要件を考慮してください。

- タイムアウトの時間が長くなると、正常でないマシンのワークロードのダウンタイムが長くなる可能性があります。
- タイムアウトが短すぎると、修復ループが生じる可能性があります。たとえば、**NotReady** ステータスを確認するためのタイムアウトについては、マシンが起動プロセスを完了できるように十分な時間を設定する必要があります。

チェックを停止するには、リソースを削除します。

11.1.1. マシンヘルスチェックのデプロイ時の制限

マシンヘルスチェックをデプロイする前に考慮すべき制限事項があります。

- マシンセットが所有するマシンのみがマシンヘルスチェックによって修復されます。
- コントロールプレーンマシンは現在サポートされておらず、それらが正常でない場合にも修正されません。

- マシンのノードがクラスターから削除される場合、マシンヘルスチェックはマシンが正常ではないとみなし、すぐにこれを修復します。
- **nodeStartupTimeout** の後にマシンの対応するノードがクラスターに加わらない場合、マシンは修復されます。
- **Machine** リソースフェーズが **Failed** の場合、マシンはすぐに修復されます。

関連情報

- **MachineHealthCheck** CR で定義できるノードの状態についての詳細は、クラスター内のすべてのノードの一覧表示について 参照してください。
- 一時停止 (short-circuiting) についての詳細は、マシンヘルスチェックによる修復の一時停止 (short-circuiting) を参照してください。

11.2. 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ンタイムが長くなる可能性があります。
7. ターゲットプールで同時に修復できるマシンの数を指定します。これはパーセンテージまたは整数として設定できます。正常でないマシンの数が **maxUnhealthy** で設定された制限を超える場合、

修復は実行されません。

8 マシンが正常でないと判別される前に、ノードがクラスターに参加するまでマシンヘルスチェックが待機する必要のあるタイムアウト期間を指定します。

注記

matchLabels はあくまでもサンプルであるため、特定のニーズに応じてマシングループをマッピングする必要があります。

11.2.1. 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。

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。

11.2.1.1. 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。

11.2.1.2. 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。

11.3. MACHINEHEALTHCHECK リソースの作成

クラスターに、すべての **MachineSets** の **MachineHealthCheck** リソースを作成できます。コントロールプレーンマシンをターゲットとする **MachineHealthCheck** リソースを作成することはできません。

前提条件

- **oc** コマンドラインインターフェイスをインストールします。

手順

1. マシンヘルスチェックの定義を含む **healthcheck.yml** ファイルを作成します。
2. **healthcheck.yml** ファイルをクラスターに適用します。

```
$ oc apply -f healthcheck.yml
```

マシンヘルスチェックを設定し、デプロイして、正常でないベアメタルノードを検出し、修復することができます。

11.4. 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

11.4.1. ベアメタル上の MachineHealthCheck

ベアメタルクラスターでのマシンの削除により、ベアメタルホストの再プロビジョニングがトリガーされます。通常、ベアメタルの再プロビジョニングは長いプロセスで、クラスターにコンピュートリソー

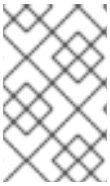
スがなくなり、アプリケーションが中断される可能性があります。デフォルトの修復プロセスをマシンの削除からホストの電源サイクルに切り換えるには、**MachineHealthCheck** リソースに **machine.openshift.io/remediation-strategy: external-baremetal** アノテーションを付けます。

アノテーションの設定後に、BMC 認証情報を使用して正常でないマシンの電源が入れ直されます。

11.4.2. 修復プロセスについて

修復プロセスは以下のように機能します。

1. MachineHealthCheck (MHC) コントローラーは、ノードが正常ではないことを検知します。
2. MHC は、正常でないノードの電源オフを要求するベアメタルマシンコントローラーに通知します。
3. 電源がオフになった後にノードが削除され、クラスターは影響を受けたワークロードを他のノードで再スケジューリングできます。
4. ベアメタルマシンコントローラーはノードの電源をオンにするよう要求します。
5. ノードの起動後、ノードはクラスターに自らを再登録し、これにより新規ノードが作成されます。
6. ノードが再作成されると、ベアメタルマシンコントローラーは、削除前に正常でないノードに存在したアノテーションとラベルを復元します。



注記

電源操作が完了していない場合、ベアメタルマシンコントローラーは、外部でプロビジョニングされたコントロールプレーンノード (別称マスターノード) 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。

11.4.3. ベアメタルの MachineHealthCheck リソースの作成

前提条件

- OpenShift Container Platform は、インストーラーでプロビジョニングされるインフラストラクチャー (IPI) を使用してインストールされます。
- ベースボード管理コントローラー (BMC) 認証情報へのアクセス (または 各ノードへの BMC アクセス)。
- 正常でないノードの BMC インターフェイスへのネットワークアクセス。

手順

1. マシンヘルスチェックの定義を含む **healthcheck.yaml** ファイルを作成します。
2. 以下のコマンドを使用して、**healthcheck.yaml** ファイルをクラスターに適用します。

```
$ oc apply -f healthcheck.yaml
```

ベアメタルのサンプル MachineHealthCheck リソース

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineHealthCheck
metadata:
  name: example (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。

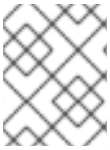
(3) (4) チェックする必要のあるマシンプールのラベルを指定します。

(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。

<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>