OpenShift Container Platform 4.4

マシン管理

クラスターマシンの追加および保守

Last Updated: 2022-11-28

OpenShift Container Platform 4.4 マシン管理

クラスターマシンの追加および保守

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

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 マシンセットの作成

1.1. 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。

1.1.1. 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 4.4 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 4.4 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。

マシン

ノードのホストを記述する基本的なユニットです。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。

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。

1.1.2. 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、**<infrastructureID>** 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: <infrastructureID> 1
  name: <infrastructureID>-<role>-<zone> 2
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructureID> 3
      machine.openshift.io/cluster-api-machineset: <infrastructureID>-<role>-<zone> 4
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructureID> 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: <infrastructureID>-<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: <infrastructureID>-worker-profile (11)
      instanceType: m4.large
      kind: AWSMachineProviderConfig
      placement:
        availabilityZone: us-east-1a
        region: us-east-1
      securityGroups:
        - filters:
            - name: tag:Name
              values:
                - <infrastructureID>-worker-sg (12)
      subnet:
        filters:
          - name: tag:Name
            values:
              - <infrastructureID>-private-us-east-1a (13)
      tags:
        - name: kubernetes.io/cluster/<infrastructureID> (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。

1.1.3. 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

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

....

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 です。

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


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。

4. 新規のマシンセットが利用可能になった後に、マシンおよびそれが参照するノードのステータスを確認します。

```
$ oc describe machine <name> -n openshift-machine-api
```

以下に例を示します。

```
$ oc describe machine agl030519-vplxk-infra-us-east-1a -n openshift-machine-api

status:
  addresses:
  - address: 10.0.133.18
    type: InternalIP
  - address: ""
    type: ExternalDNS
  - address: ip-10-0-133-18.ec2.internal
    type: InternalDNS
  lastUpdated: "2019-05-03T10:38:17Z"
  nodeRef:
    kind: Node
    name: ip-10-0-133-18.ec2.internal
    uid: 71fb8d75-6d8f-11e9-9ff3-0e3f103c7cd8
  providerStatus:
    apiVersion: awsproviderconfig.openshift.io/v1beta1
    conditions:
    - lastProbeTime: "2019-05-03T10:34:31Z"
      lastTransitionTime: "2019-05-03T10:34:31Z"
      message: machine successfully created
      reason: MachineCreationSucceeded
      status: "True"
      type: MachineCreation
    instanceId: i-09ca0701454124294
    instanceState: running
    kind: AWSMachineProviderStatus
```

5. 新しいノードを表示し、新規ノードが指定したラベルを持っていることを確認します。

```
$ oc get node <node_name> --show-labels
```

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

注記

マシンセットへの変更は、マシンセットが所有する既存のマシンには適用されません。たとえば、編集されたか、または既存のマシンセットに追加されたラベルは、マシンセットに関連付けられた既存マシンおよびノードには伝播しません。

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。

1.2. 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。

1.2.1. 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 4.4 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 4.4 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。

マシン

ノードのホストを記述する基本的なユニットです。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。

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。

1.2.2. 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、**centralus** 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、**<infrastructureID>** 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: <infrastructureID> 1
    machine.openshift.io/cluster-api-machine-role: <role> 2
    machine.openshift.io/cluster-api-machine-type: <role> 3
  name: <infrastructureID>-<role>-<region> 4
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructureID> 5
      machine.openshift.io/cluster-api-machineset: <infrastructureID>-<role>-<region> 6
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructureID> 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: <infrastructureID>-<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/<infrastructureID>-
rg/providers/Microsoft.Compute/images/<infrastructureID>
            sku: ""
            version: ""
```

```
      internalLoadBalancer: ""
      kind: AzureMachineProviderSpec
      location: centralus
      managedIdentity: <infrastructureID>-identity (12)
      metadata:
        creationTimestamp: null
      natRule: null
      networkResourceGroup: ""
      osDisk:
        diskSizeGB: 128
        managedDisk:
          storageAccountType: Premium_LRS
        osType: Linux
      publicIP: false
      publicLoadBalancer: ""
      resourceGroup: <infrastructureID>-rg (13)
      sshPrivateKey: ""
      sshPublicKey: ""
      subnet: <infrastructureID>-<role>-subnet (14) (15)
      userDataSecret:
        name: <role>-user-data (16)
      vmSize: Standard_D2s_v3
      vnet: <infrastructureID>-vnet (17)
      zone: "1" (18)
```

(1) (5) (7) (12) (13) (14) (17) クラスターのプロビジョニング時に設定したクラスター ID を基にするインフラストラクチャー ID を指定します。OpenShift CLI がインストールされている場合は、以下のコマンドを実行してインフラストラクチャー ID を取得できます。

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

(2) (3) (8) (9) (11) (15) (16) 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。

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

1.2.3. 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

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

....

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 です。

   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


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。

4. 新規のマシンセットが利用可能になった後に、マシンおよびそれが参照するノードのステータスを確認します。

```
$ oc describe machine <name> -n openshift-machine-api
```

以下に例を示します。

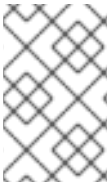
```
$ oc describe machine agl030519-vplxk-infra-us-east-1a -n openshift-machine-api

status:
  addresses:
  - address: 10.0.133.18
    type: InternalIP
  - address: ""
    type: ExternalDNS
  - address: ip-10-0-133-18.ec2.internal
    type: InternalDNS
  lastUpdated: "2019-05-03T10:38:17Z"
  nodeRef:
    kind: Node
    name: ip-10-0-133-18.ec2.internal
    uid: 71fb8d75-6d8f-11e9-9ff3-0e3f103c7cd8
  providerStatus:
    apiVersion: awsproviderconfig.openshift.io/v1beta1
    conditions:
    - lastProbeTime: "2019-05-03T10:34:31Z"
      lastTransitionTime: "2019-05-03T10:34:31Z"
      message: machine successfully created
      reason: MachineCreationSucceeded
      status: "True"
      type: MachineCreation
    instanceId: i-09ca0701454124294
    instanceState: running
    kind: AWSMachineProviderStatus
```

5. 新しいノードを表示し、新規ノードが指定したラベルを持っていることを確認します。

```
$ oc get node <node_name> --show-labels
```

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



注記

マシンセットへの変更は、マシンセットが所有する既存のマシンには適用されません。たとえば、編集されたか、または既存のマシンセットに追加されたラベルは、マシンセットに関連付けられた既存マシンおよびノードには伝播しません。

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。

1.3. 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。

1.3.1. 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 4.4 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 4.4 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。

マシン

ノードのホストを記述する基本的なユニットです。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。

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。

1.3.2. 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、**<infrastructureID>** 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: <infrastructureID> 1
  name: <infrastructureID>-w-a 2
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructureID> 3
      machine.openshift.io/cluster-api-machineset: <infrastructureID>-w-a 4
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructureID> 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: <infrastructureID>-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: <infrastructureID>-rhcos-image 10
            labels: null
            sizeGb: 128
            type: pd-ssd
          kind: GCPMachineProviderSpec
          machineType: n1-standard-4
```

```
    metadata:
      creationTimestamp: null
    networkInterfaces:
    - network: <infrastructureID>-network (11)
      subnetwork: <infrastructureID>-<role>-subnet (12)
    projectID: <project_name> (13)
    region: us-central1
    serviceAccounts:
    - email: <infrastructureID>-w@<project_name>.iam.gserviceaccount.com (14) (15)
      scopes:
      - https://www.googleapis.com/auth/cloud-platform
    tags:
    - <infrastructureID>-<role> (16)
    userDataSecret:
      name: worker-user-data
    zone: us-central1-a
```

(1) (2) (3) (4) (5) (8) (10) (11) (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
```

(12) (16) 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。

(13) (15) 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。

1.3.3. 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
   ```

```
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

....

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 です。

(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


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。

4. 新規のマシンセットが利用可能になった後に、マシンおよびそれが参照するノードのステータスを確認します。

```
$ oc describe machine <name> -n openshift-machine-api
```

以下に例を示します。

```
$ oc describe machine agl030519-vplxk-infra-us-east-1a -n openshift-machine-api

status:
  addresses:
  - address: 10.0.133.18
    type: InternalIP
  - address: ""
    type: ExternalDNS
  - address: ip-10-0-133-18.ec2.internal
    type: InternalDNS
  lastUpdated: "2019-05-03T10:38:17Z"
  nodeRef:
    kind: Node
    name: ip-10-0-133-18.ec2.internal
    uid: 71fb8d75-6d8f-11e9-9ff3-0e3f103c7cd8
  providerStatus:
    apiVersion: awsproviderconfig.openshift.io/v1beta1
    conditions:
    - lastProbeTime: "2019-05-03T10:34:31Z"
      lastTransitionTime: "2019-05-03T10:34:31Z"
      message: machine successfully created
      reason: MachineCreationSucceeded
      status: "True"
      type: MachineCreation
    instanceId: i-09ca0701454124294
    instanceState: running
    kind: AWSMachineProviderStatus
```

5. 新しいノードを表示し、新規ノードが指定したラベルを持っていることを確認します。

```
$ oc get node <node_name> --show-labels
```

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

注記

マシンセットへの変更は、マシンセットが所有する既存のマシンには適用されません。たとえば、編集されたか、または既存のマシンセットに追加されたラベルは、マシンセットに関連付けられた既存マシンおよびノードには伝播しません。

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。

1.4. 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。

1.4.1. 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 4.4 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 4.4 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。

マシン

ノードのホストを記述する基本的なユニットです。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。

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。

1.4.2. 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.openshift.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 ラベルであり、**node_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: <node_role> 2
    machine.openshift.io/cluster-api-machine-type: <node_role> 3
  name: <infrastructure_ID>-<node_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>-<node_role> 6
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_ID> 7
        machine.openshift.io/cluster-api-machine-role: <node_role> 8
        machine.openshift.io/cluster-api-machine-type: <node_role> 9
        machine.openshift.io/cluster-api-machineset: <infrastructure_ID>-<node_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>
          kind: OpenstackProviderSpec
          networks:
          - filter: {}
            subnets:
            - filter:
                name: <subnet_name>
                tags: openshiftClusterID=<infrastructure_ID>
          securityGroups:
          - filter: {}
            name: <infrastructure_ID>-<node_role>
          serverMetadata:
            Name: <infrastructure_ID>-<node_role>
            openshiftClusterID: <infrastructure_ID>
          tags:
          - openshiftClusterID=<infrastructure_ID>
```

```
        trunk: true
        userDataSecret:
          name: <node_role>-user-data 11
```

1 5 7 クラスターのプロビジョニング時に設定したクラスター ID を基にするインフラストラクチャー ID を指定します。OpenShift CLI 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 11 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。

1.4.3. 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

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

....
```

```
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 です。

(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


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。

4. 新規のマシンセットが利用可能になった後に、マシンおよびそれが参照するノードのステータスを確認します。

```
$ oc describe machine <name> -n openshift-machine-api
```

以下に例を示します。

```
$ oc describe machine agl030519-vplxk-infra-us-east-1a -n openshift-machine-api

status:
  addresses:
  - address: 10.0.133.18
    type: InternalIP
  - address: ""
    type: ExternalDNS
  - address: ip-10-0-133-18.ec2.internal
    type: InternalDNS
  lastUpdated: "2019-05-03T10:38:17Z"
  nodeRef:
    kind: Node
    name: ip-10-0-133-18.ec2.internal
```

```
    uid: 71fb8d75-6d8f-11e9-9ff3-0e3f103c7cd8
  providerStatus:
    apiVersion: awsproviderconfig.openshift.io/v1beta1
    conditions:
    - lastProbeTime: "2019-05-03T10:34:31Z"
      lastTransitionTime: "2019-05-03T10:34:31Z"
      message: machine successfully created
      reason: MachineCreationSucceeded
      status: "True"
      type: MachineCreation
    instanceId: i-09ca0701454124294
    instanceState: running
    kind: AWSMachineProviderStatus
```

5. 新しいノードを表示し、新規ノードが指定したラベルを持っていることを確認します。

```
$ oc get node <node_name> --show-labels
```

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

注記

マシンセットへの変更は、マシンセットが所有する既存のマシンには適用されません。たとえば、編集されたか、または既存のマシンセットに追加されたラベルは、マシンセットに関連付けられた既存マシンおよびノードには伝播しません。

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。

第2章 マシンセットの手動によるスケーリング

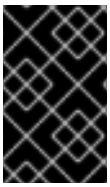
マシンセットのマシンのインスタンスを追加または削除できます。

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。

2.1. 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、マシンを独自に手動でプロビジョニングしている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. 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。

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. 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
```

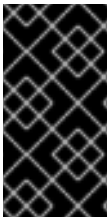
   マシンセットをスケールアップまたはスケールダウンできます。新規マシンが利用可能になるまで数分の時間がかかります。

2.3. 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、削除ポリシーにかかわらず、アノテーション **machine.openshift.io/cluster-api-delete-machine** を関係するマシンに追加して設定できます。



重要

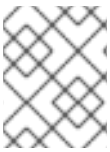
デフォルトで、OpenShift Container Platform ルーター Pod はワーカーにデプロイされます。ルーターは Web コンソールなどの一部のクラスターリソースにアクセスすることが必要であるため、 ルーター Pod をまず再配置しない限り、ワーカーのマシンセットを **0** にスケーリングできません。

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3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。



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。

3.1. 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。

第4章 マシンの削除

特定のマシンを削除できます。

4.1. 特定マシンの削除

特定のマシンを削除できます。

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** にログインすること。

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。

第5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。

5.1. 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。

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、リソースの使用量が少なく、重要な Pod すべてが他のノードに適合する場合など、一部のノードが長い期間にわたって不要な状態が続く場合にクラスターのサイズを縮小します。

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。

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。

5.2. MACHINE AUTOSCALER

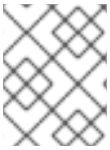
Machine Autoscaler は、マシンセットで OpenShift Container Platform クラスターにデプロイするマシン数を調整します。デフォルトの **worker** マシンセットおよび作成する他のマシンセットの両方をスケーリングできます。Machine Autoscaler は、追加のデプロイメントをサポートするのに十分なリソースがクラスターにない場合に追加のマシンを作成します。**MachineAutoscaler** リソースの値への変更 (例: インスタンスの最小または最大数) は、それらがターゲットとするマシンセットに即時に適用されます。

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。

5.3. 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

クラスター 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 つのクラスター Autoscaler のみを作成できます。

5.3.1. ClusterAutoscaler リソース定義

この **ClusterAutoscaler** リソース定義は、クラスター 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 (10)
        min: 0 (11)
        max: 4 (12)
  scaleDown: (13)
    enabled: true (14)
    delayAfterAdd: 10m (15)
    delayAfterDelete: 5m (16)
    delayAfterFailure: 30s (17)
    unneededTime: 60s (18)
```

(1) Cluster Autoscaler に追加のノードをデプロイさせるために Pod が超えている必要のある優先順位を指定します。32 ビットの整数値を入力します。**podPriorityThreshold** 値は、各 Pod に割り当てる **PriorityClass** の値と比較されます。

(2) デプロイするノードの最大数を指定します。この値は、Autoscaler が制御するマシンだけでなく、クラスターにデプロイされるマシンの合計数です。この値は、すべてのコントロールプレーンおよびコンピュートマシン、および **MachineAutoscaler** リソースに指定するレプリカの合計数に対応するのに十分な大きさの値であることを確認します。

(3) デプロイするコアの最小数を指定します。

(4) デプロイするコアの最大数を指定します。

(5) ノードごとにメモリーの最小量 (GiB 単位) を指定します。

6 ノードごとにメモリーの最大量 (GiB 単位) を指定します。

7 10 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 11 デプロイする GPU の最小数を指定します。

9 12 デプロイする GPU の最大数を指定します。

13 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。

14 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。

15 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。

16 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。

17 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。

18 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。

5.3.2. クラスター 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
```

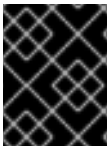
1 **<filename>** は、カスタマイズしたリソースファイルの名前です。

5.4. 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。

5.5. 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。

5.5.1. MachineAutoscaler リソース定義

この **MachineAutoscaler** リソース定義は、マシン 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以下の形式を取ります。 **<clusterid>-<machineset>-<aws-region-az>**

(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。この値は、**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 AWS 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。

5.5.2. 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。

5.6. 関連情報

- Pod の優先順位についての詳細は、 Including pod priority in pod scheduling decisions in OpenShift Container Platform を参照してください。

第6章 インフラストラクチャーマシンセットの作成

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ようにマシンセットを作成することができます。特定の Kubernetes ラベルをこれらのマシンに適用してから、インフラストラクチャーコンポーネントをそれらのマシンでのみ実行されるように更新します。これらのインフラストラクチャーノードは、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。

重要

以前のバージョンの OpenShift Container Platform とは異なり、インフラストラクチャーコンポーネントをマスターマシンに移動することはできません。コンポーネントを移動するには、新規マシンセットを作成する必要があります。

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

以下の OpenShift Container Platform コンポーネントはインフラストラクチャーコンポーネントです。

- 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
- コンテナーイメージレジストリー
- 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

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。

6.2. 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します。ロギング集約ソリューションおよびサービスメッシュはどちらも Elasticsearch をデプロインし、Elasticsearch では複数の異なるノードにインストールされる 3 つのインスタンスが必要です。高可用性を確保するには、これらのノードを複数の異なるアベイラビリティーゾーンにデプロイします。各アベイラビリティーゾーンにそれぞれ異なるマシンセットが必要であるため、3 つ以上のマシンセットを作成します。

6.2.1. 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。

6.2.1.1. 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/<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、**<infrastructureID>** 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: <infrastructureID> 1
  name: <infrastructureID>-<role>-<zone> 2
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructureID> 3
      machine.openshift.io/cluster-api-machineset: <infrastructureID>-<role>-<zone> 4
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructureID> 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: <infrastructureID>-<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: <infrastructureID>-worker-profile 11
          instanceType: m4.large
          kind: AWSMachineProviderConfig
          placement:
            availabilityZone: us-east-1a
            region: us-east-1
          securityGroups:
            - filters:
                - name: tag:Name
                  values:
                    - <infrastructureID>-worker-sg 12
          subnet:
            filters:
```

```
        - name: tag:Name
          values:
            - <infrastructureID>-private-us-east-1a 13
      tags:
        - name: kubernetes.io/cluster/<infrastructureID> 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。

6.2.1.2. 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、**centralus** 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、**<infrastructureID>** 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: <infrastructureID> 1
    machine.openshift.io/cluster-api-machine-role: <role> 2
    machine.openshift.io/cluster-api-machine-type: <role> 3
  name: <infrastructureID>-<role>-<region> 4
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructureID> 5
      machine.openshift.io/cluster-api-machineset: <infrastructureID>-<role>-<region> 6
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructureID> 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: <infrastructureID>-<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/<infrastructureID>-
rg/providers/Microsoft.Compute/images/<infrastructureID>
          sku: ""
          version: ""
        internalLoadBalancer: ""
        kind: AzureMachineProviderSpec
        location: centralus
        managedIdentity: <infrastructureID>-identity 12
        metadata:
          creationTimestamp: null
        natRule: null
        networkResourceGroup: ""
        osDisk:
          diskSizeGB: 128
          managedDisk:
            storageAccountType: Premium_LRS
          osType: Linux
        publicIP: false
        publicLoadBalancer: ""
        resourceGroup: <infrastructureID>-rg 13
        sshPrivateKey: ""
        sshPublicKey: ""
        subnet: <infrastructureID>-<role>-subnet 14 15
        userDataSecret:
          name: <role>-user-data 16
        vmSize: Standard_D2s_v3
        vnet: <infrastructureID>-vnet 17
        zone: "1" 18
```

1 5 7 12 13 14 17 クラスターのプロビジョニング時に設定したクラスター ID を基にするインフラストラクチャー ID を指定します。OpenShift CLI がインストールされている場合は、以下のコマンドを実行してインフラストラクチャー ID を取得できます。

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

2 3 8 9 11 15 16 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。

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

6.2.1.3. 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/<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、**<infrastructureID>** 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: <infrastructureID> 1
  name: <infrastructureID>-w-a 2
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructureID> 3
      machine.openshift.io/cluster-api-machineset: <infrastructureID>-w-a 4
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructureID> 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: <infrastructureID>-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: <infrastructureID>-rhcos-image 10
            labels: null
            sizeGb: 128
            type: pd-ssd
          kind: GCPMachineProviderSpec
          machineType: n1-standard-4
          metadata:
```

```
    creationTimestamp: null
  networkInterfaces:
  - network: <infrastructureID>-network 11
    subnetwork: <infrastructureID>-<role>-subnet 12
  projectID: <project_name> 13
  region: us-central1
  serviceAccounts:
  - email: <infrastructureID>-w@<project_name>.iam.gserviceaccount.com 14 15
    scopes:
    - https://www.googleapis.com/auth/cloud-platform
  tags:
  - <infrastructureID>-<role> 16
  userDataSecret:
    name: worker-user-data
  zone: us-central1-a
```

1 2 3 4 5 8 10 11 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
```

12 16 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。

13 15 クラスターに使用する GCP プロジェクトの名前を指定します。

6.2.2. 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

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

....

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 です。

(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


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。

4. 新規のマシンセットが利用可能になった後に、マシンおよびそれが参照するノードのステータスを確認します。

```
$ oc describe machine <name> -n openshift-machine-api
```

以下に例を示します。

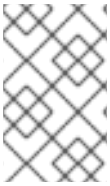
```
$ oc describe machine agl030519-vplxk-infra-us-east-1a -n openshift-machine-api

status:
  addresses:
  - address: 10.0.133.18
    type: InternalIP
  - address: ""
    type: ExternalDNS
  - address: ip-10-0-133-18.ec2.internal
    type: InternalDNS
  lastUpdated: "2019-05-03T10:38:17Z"
  nodeRef:
    kind: Node
    name: ip-10-0-133-18.ec2.internal
    uid: 71fb8d75-6d8f-11e9-9ff3-0e3f103c7cd8
  providerStatus:
    apiVersion: awsproviderconfig.openshift.io/v1beta1
    conditions:
    - lastProbeTime: "2019-05-03T10:34:31Z"
      lastTransitionTime: "2019-05-03T10:34:31Z"
      message: machine successfully created
      reason: MachineCreationSucceeded
      status: "True"
      type: MachineCreation
    instanceId: i-09ca0701454124294
    instanceState: running
    kind: AWSMachineProviderStatus
```

5. 新しいノードを表示し、新規ノードが指定したラベルを持っていることを確認します。

```
$ oc get node <node_name> --show-labels
```

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



注記

マシンセットへの変更は、マシンセットが所有する既存のマシンには適用されません。たとえば、編集されたか、または既存のマシンセットに追加されたラベルは、マシンセットに関連付けられた既存マシンおよびノードには伝播しません。

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。

6.3. 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。

6.3.1. 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。

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

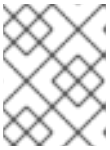
1. 以下のコマンドを使用して、テイントを infra ノードに追加し、ユーザーのワークロードをこれにスケジュールできないようにします。

```
$ oc adm taint nodes <node_name> <key>:<effect>
```

以下は例になります。

```
$ 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

- Pod のノードへのスケジュールの一般的な情報については、スケジューラーを使用した Pod の配置の制御 について参照してください。
- Pod を infra ノードにスケジュールする方法については、リソースのインフラストラクチャーマシンセットへの移動 について参照してください。

6.4. 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。

6.4.1. 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

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)

NAME                         STATUS  ROLES        AGE   VERSION
ip-10-0-217-226.ec2.internal  Ready   infra,worker  17h   v1.17.1
```

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

   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。

6.4.2. 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 config/cluster -o yaml
```

出力は以下のテキストのようになります。

```
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 config/cluster
```

3. テキストの以下の行を、オブジェクトの **spec** セクションに追加します。

```
  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。

6.4.3. 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。

追加リソース

- 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。

6.4.4. クラスターロギングリソースの移動

すべてのクラスターロギングコンポーネント、Elasticsearch、Kibana、および Curator 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。

注記

マシンセットを 6 つ以上のレプリカを使用するように設定する必要があります。

前提条件

- クラスターロギングおよび 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
  curation:
    curator:
      nodeSelector: 1
        node-role.kubernetes.io/infra: ''
      resources: null
      schedule: 30 3 * * *
    type: curator
  logStore:
    elasticsearch:
      nodeCount: 3
      nodeSelector: 2
        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: 3
        node-role.kubernetes.io/infra: '' 4
      proxy:
        resources: null
      replicas: 1
      resources: null
    type: kibana

....
```

1 2 3 4 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。

検証手順

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
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
NAME                                         STATUS   ROLES          AGE   VERSION
ip-10-0-133-216.us-east-2.compute.internal   Ready    master         60m   v1.17.1
ip-10-0-139-146.us-east-2.compute.internal   Ready    master         60m   v1.17.1
ip-10-0-139-192.us-east-2.compute.internal   Ready    worker         51m   v1.17.1
ip-10-0-139-241.us-east-2.compute.internal   Ready    worker         51m   v1.17.1
ip-10-0-147-79.us-east-2.compute.internal    Ready    worker         51m   v1.17.1
ip-10-0-152-241.us-east-2.compute.internal   Ready    master         60m   v1.17.1
ip-10-0-139-48.us-east-2.compute.internal    Ready    infra          51m   v1.17.1
```

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

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: '' 2
      proxy:
        resources: null
      replicas: 1
      resources: null
    type: kibana
```

1 2 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
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
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
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
```

第7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

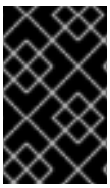
7.1. RHEL コンピュートマシンの OPENSHIFT CONTAINER PLATFORM クラスターへの追加

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。

7.1.1. 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 4.4 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ー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。

7.1.2. RHEL コンピュートノードのシステム要件

OpenShift Container Platform 環境の Red Hat Enterprise Linux (RHEL) コンピュートマシンホスト (またはワーカーマシンホストとしても知られる) 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 7.6-7.8 (最小のインストールオプション)

重要

OpenShift Container Platform 4.4 でサポートされるのは RHEL 7.6-7.8 のみになります。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。

- 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。
- 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。

7.1.2.1. 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。

7.1.3. Playbook 実行のためのマシンの準備

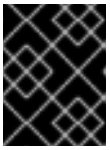
Red Hat Enterprise Linux をオペレーティングシステムとして使用するコンピュートマシンを OpenShift Container Platform 4.4 クラスターに追加する前に、Playbook を実行するマシンを準備する必要があります。このマシンはクラスターの一部にはなりませんが、クラスターにアクセスできる必要があります。

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 4.4 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-4.4-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。

7.1.4. 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 4.4 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-server-extras-rpms" \
    --enable="rhel-7-server-ose-4.4-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。

7.1.5. 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 4.4 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. Playbook を実行します。

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

7.1.6. 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

NAME                 STATUS   ROLES    AGE   VERSION
master-01.example.com   Ready    master   40d   v1.17.1
master-02.example.com   Ready    master   40d   v1.17.1
master-03.example.com   Ready    master   40d   v1.17.1
worker-01.example.com   Ready    worker   40d   v1.17.1
worker-02.example.com   Ready    worker   40d   v1.17.1
```

出力には作成したすべてのマシンが一覧表示されます。

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

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。最初の CSR の承認後、後続のノードクライアント CSR はクラスターの **kube-controller-manger** によって自動的に承認されます。kubelet 提供証明書の要求を自動的に承認する方法を実装する必要があります。

- 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
```

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。

7.1.7. 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。

7.1.7.1. 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-local-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。

7.2. 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。

7.2.1. RHEL コンピュートノードのクラスターへの追加について

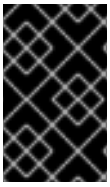
OpenShift Container Platform 4.4 には、ユーザーによってプロビジョニングされるインフラストラクチャーを使用する場合、Red Hat Enterprise Linux (RHEL) マシンをクラスター内のコンピュートまたは

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ー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

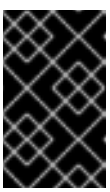
swap メモリーは、OpenShift Container Platform クラスターに追加されるすべての RHEL マシンで無効にされます。これらのマシンで swap メモリーを有効にすることはできません。

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。

7.2.2. RHEL コンピュートノードのシステム要件

OpenShift Container Platform 環境の Red Hat Enterprise Linux (RHEL) コンピュートマシンホスト (またはワーカーマシンホストとしても知られる) 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 7.6-7.8 (最小のインストールオプション)

    重要

    OpenShift Container Platform 4.4 でサポートされるのは RHEL 7.6-7.8 のみになります。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。

  - 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。

- 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。

7.2.2.1. 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。

7.2.3. 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 4.4 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-server-extras-rpms" \
    --enable="rhel-7-server-ose-4.4-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。

7.2.4. 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 4.4 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. スケールアップ Playbook を実行します。

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

7.2.5. 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

NAME                 STATUS   ROLES    AGE   VERSION
master-01.example.com   Ready    master   40d   v1.17.1
master-02.example.com   Ready    master   40d   v1.17.1
master-03.example.com   Ready    master   40d   v1.17.1
worker-01.example.com   Ready    worker   40d   v1.17.1
worker-02.example.com   Ready    worker   40d   v1.17.1
```

   出力には作成したすべてのマシンが一覧表示されます。

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

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。最初の CSR の承認後、後続のノードクライアント CSR はクラスターの **kube-controller-manger** によって自動的に承認されます。kubelet 提供証明書の要求を自動的に承認する方法を実装する必要があります。

   - 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
```

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。

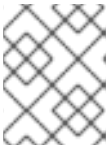
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

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。

7.2.6. 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。

7.3. 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。

7.3.1. 前提条件

- クラスターを vSphere にインストールしている。

7.3.2. 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。

7.3.3. 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

NAME                 STATUS   ROLES    AGE   VERSION
```

```
master-01.example.com   Ready    master   40d   v1.17.1
master-02.example.com   Ready    master   40d   v1.17.1
master-03.example.com   Ready    master   40d   v1.17.1
worker-01.example.com   Ready    worker   40d   v1.17.1
worker-02.example.com   Ready    worker   40d   v1.17.1
```

出力には作成したすべてのマシンが一覧表示されます。

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

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。最初の CSR の承認後、後続のノードクライアント CSR はクラスターの **kube-controller-manger** によって自動的に承認されます。kubelet 提供証明書の要求を自動的に承認する方法を実装する必要があります。

- 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
```

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。

7.4. 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。

7.4.1. 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。

7.4.2. 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 ブートを使用する手順を実行してマシンを作成することができます。

7.4.2.1. ISO イメージを使用した追加の Red Hat Enterprise Linux CoreOS (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 BIOS または UEFI RHCOS イメージファイルの URL を取得します。

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=yes
coreos.inst.install_dev=sda 1
coreos.inst.image_url=<bare_metal_image_URL> 2
coreos.inst.ignition_url=http://example.com/worker.ign 3
```

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. サーバーにアップロードした UEFI または BIOS イメージの URL を指定します。
3. コンピュート Ignition 設定ファイルの URL を指定します。

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。

7.4.2.2. PXE または iPXE ブートによる追加の Red Hat Enterprise Linux CoreOS (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>-installer-kernel-<architecture> (1)
    APPEND ip=dhcp rd.neednet=1 initrd=http://<HTTP_server>/rhcos-<version>-installer-initramfs.<architecture>.img console=tty0 console=ttyS0 coreos.inst=yes coreos.inst.install_dev=sda coreos.inst.image_url=http://<HTTP_server>/rhcos-<version>-metal.<architecture>.raw.gz coreos.inst.ignition_url=http://<HTTP_server>/worker.ign (2) (3)
```

     (1) HTTP サーバーにアップロードした **kernel** ファイルの場所を指定します。

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

     (3) 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** パラメーター値は **initramfs** ファイルの場所であり、**coreos.inst.image_url** パラメーター値は圧縮された metal RAW イメージの場所、および **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になります。

   - iPXE の場合:

```
kernel http://<HTTP_server>/rhcos-<version>-installer-kernel-<architecture> ip=dhcp
```

```
rd.neednet=1 initrd=http://<HTTP_server>/rhcos-<version>-installer-initramfs.
<architecture>.img console=tty0 console=ttyS0 coreos.inst=yes
coreos.inst.install_dev=sda coreos.inst.image_url=http://<HTTP_server>/rhcos-
<version>-metal.<arhcitectutre>.raw.gz
coreos.inst.ignition_url=http://<HTTP_server>/worker.ign (1) (2)
initrd http://<HTTP_server>/rhcos-<version>-installer-initramfs.<architecture>.img (3)
boot
```

(1) HTTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**kernel** パラメーター値は **kernel** ファイルの場所であり、**initrd** パラメーター値は **initramfs** ファイルの場所、**coreos.inst.image_url** パラメーター値は圧縮された metal RAW イメージの場所、および **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になります。

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

(3) HTTP サーバーにアップロードした **initramfs** ファイルの場所を指定します。

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

7.4.3. 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

NAME                 STATUS   ROLES    AGE   VERSION
master-01.example.com   Ready    master   40d   v1.17.1
master-02.example.com   Ready    master   40d   v1.17.1
master-03.example.com   Ready    master   40d   v1.17.1
worker-01.example.com   Ready    worker   40d   v1.17.1
worker-02.example.com   Ready    worker   40d   v1.17.1
```

出力には作成したすべてのマシンが一覧表示されます。

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
```

```
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。最初の CSR の承認後、後続のノードクライアント CSR はクラスターの **kube-controller-manger** によって自動的に承認されます。kubelet 提供証明書の要求を自動的に承認する方法を実装する必要があります。

- 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
```

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

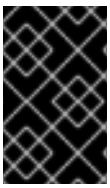
サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

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

第8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

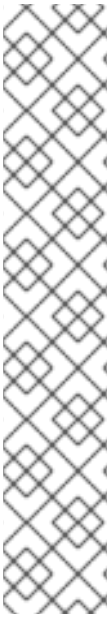
このプロセスは、マシンを独自に手動でプロビジョニングしているクラスターには適用されません。高度なマシン管理およびスケーリング機能は、マシン API が機能しているクラスターでのみ使用することができます。

8.1. マシンのヘルスチェック

MachineHealthCheck リソースを使用して、クラスター内のマシンが正常ではないとみなされる条件を定義できます。条件に一致するマシンは自動的に修復されます。

マシンの正常性を監視するには、監視する一連のマシンのラベルや、**NotReady** ステータスの期間を 15 分にしたり、 node-problem-detector に永続的な条件を表示したりするなど、チェックする条件を含む **MachineHealthCheck** カスタムリソース (CR) を作成します。

MachineHealthCheck CR を観察するコントローラーは定義した条件の有無をチェックします。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、新規マシンが代わりに作成されます。マシンが削除されると、**machine deleted** イベントが表示されます。



注記

マスターロールを持つマシンの場合、マシンのヘルスチェックは正常でないノードの数を報告しますが、マシンは削除されません。以下は例になります。

出力例

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

```
NAME      MAXUNHEALTHY   EXPECTEDMACHINES   CURRENTHEALTHY
example   40%            3                  1
```

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、コントローラーはマシンの削除を停止し、手動で介入する必要があります。

チェックを停止するには、カスタムリソースを削除します。

8.1.1. 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。

追加リソース

- 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。

8.2. サンプル MACHINEHEALTHCHECK リソース

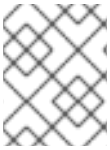
MachineHealthCheck リソースは以下の YAML ファイルのようになります。

MachineHealthCheck

```
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許可される正常でないマシンの量を指定します。これはパーセンテージまたは整数として設定できます。

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。

8.2.1. 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。

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。

8.2.1.1. 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。

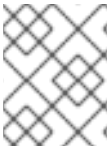
8.2.1.2. パーセンテージを使用した maxUnhealthy の設定

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。

8.3. 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
```