OpenShift Container Platform 4.1

マシン管理

クラスターマシンの追加および保守

Last Updated: 2020-04-16

OpenShift Container Platform 4.1 マシン管理

クラスターマシンの追加および保守

法律上の通知

概要

本書では、OpenShift Container Platform 4.1 クラスターを構成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 4.1 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 MACHINESET の作成

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 4.1 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 4.1 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 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。

MachineSet

マシンのグループです。MachineSet とマシンの関係は、ReplicaSet 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 MachineSet 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。

MachineAutoscaler

この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 MachineSet に設定でき、MachineAutoscaler 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。ClusterAutoscale および MachineAutoscaler リソースは、どちらも ClusterAutoscalerOperator によって利用可能にされます。

ClusterAutoscaler

このリソースはアップストリームの ClusterAutoscaler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 MachineSet 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、ScalingPolicy 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。

MachineHealthCheck

この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新規マシンを作成します。

注記

バージョン 4.1 では、MachineHealthChecks はテクノロジープレビュー機能です。

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。4.1 以降、このプロセスはより容易になりました。それぞれの MachineSet 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 MachineSet 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. MACHINESET 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) リージョンに MachineSet を定義し、**node-role.kubernetes.io/<role>: ""** というラベルが付けられたノードを作成します。

このサンプルでは、**<clusterID>** 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 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: <clusterID> 1
  name: <clusterID>-<role>-us-east-1a 2
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <clusterID> 3
      machine.openshift.io/cluster-api-machineset: <clusterID>-<role>-us-east-1a 4
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <clusterID> 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: <clusterID>-<role>-us-east-1a 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: <clusterID>-worker-profile 11
          instanceType: m4.large
          kind: AWSMachineProviderConfig
```

```
      placement:
        availabilityZone: us-east-1a
        region: us-east-1
      securityGroups:
        - filters:
            - name: tag:Name
              values:
                - <clusterID>-worker-sg 12
      subnet:
        filters:
          - name: tag:Name
            values:
              - <clusterID>-private-us-east-1a 13
      tags:
        - name: kubernetes.io/cluster/<clusterID> 14
          value: owned
      userDataSecret:
        name: worker-user-data
```

1 3 5 11 12 13 14 クラスターをプロビジョニングしたときに設定したクラスター ID を指定します。

2 4 8 クラスター ID とノードラベルを指定します。

6 7 9 追加するノードラベルを指定します。

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

1.3. MACHINESET 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 MachineSet 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します。
- **oc** として知られる OpenShift コマンドラインインターフェース (CLI) をダウンロードします。
- **cluster-admin** パーミッションを持つユーザーとして、**oc** にログインします。

手順

1. 説明されているように MachineSet カスタムリソース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 MachineSet 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 MachineSet 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。

(2) デフォルトのノードラベル。

2. 新規 **MachineSet** を作成します。

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

3. MachineSet 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 MachineSet が利用可能な場合、 **DESIRED** および **CURRENT** の値は一致します。MachineSet 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 MachineSet 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 get machine -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

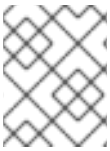
MachineSet への変更は、MachineSet が所有する既存のマシンには適用されません。たとえば、既存の MachineSet へ編集または追加されたラベルは、既存のマシンおよび MachineSet に関連付けられたノードには伝播しません。

次のステップ

他のアベイラビリティーゾーンで MachineSet が必要な場合、このプロセスを繰り返して追加の MachineSet を作成します。

第2章 MACHINESET の手動によるスケーリング

MachineSet 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 MachineSet の要素を変更する必要がある場合は、「MachineSet の変更」を参照してください。

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。

2.1. MACHINESET の手動によるスケーリング

MachineSet 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、MachineSet 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します。
- **cluster-admin** パーミッションを持つユーザーとして、**oc** にログインします。

手順

1. クラスターにある MachineSet を表示します。

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

   MachineSet は **<clusterid>-worker-<aws-region-az>** の形式で一覧表示されます。

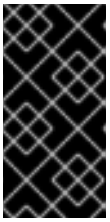
2. MachineSet をスケーリングします。

```
$ 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
```

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



重要

デフォルトで、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 MachineSet を **0** にスケーリングできません。

第3章 MACHINESET 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、MachineSet に変更を加えることができます。

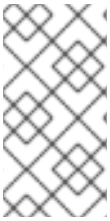
注記

他の変更なしに MachineSet をスケーリングする必要がある場合は、「MachineSet の手動によるスケーリング」を参照してください。

3.1. MACHINESET の変更

MachineSet を変更するには、MachineSet YAML を編集します。次に、マシンを削除するか、またはこれを **0**にスケールダウンして MachineSet 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。MachineSet 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 MachineSet 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 MachineSet を **0** にスケーリングできません。

前提条件

- OpenShift Container Platform クラスターと oc コマンドライン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、**oc** にログインします。

手順

1. MachineSet を編集します。

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

2. MachineSet 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. MachineSet 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 MachineSet 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。
- **oc** として知られる OpenShift コマンドラインインターフェース (CLI) をダウンロードします。
- **cluster-admin** パーミッションを持つユーザーとして、**oc** にログインします。

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 MachineSet 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 ClusterAutoscaler のデプロイと各マシンタイプの MachineAutoscaler のデプロイが必要です。

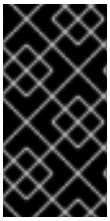
重要

ClusterAutoscaler 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. CLUSTERAUTOSCALER について

ClusterAutoscaler 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。ClusterAutoscaler 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。

ClusterAutoscaler 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。ClusterAutoscaler 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** 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。

ClusterAutoscaler 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、 ClusterAutoscaler はそのノードを削除しません。

- 制限のある PodDisruptionBudget (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。
- PDBB 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。
- Deployment、ReplicaSet、または StatefulSet 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。

ClusterAutoscaler 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 Autoscaler を実行しない。

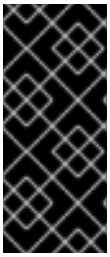
Horizontal Pod Autoscaler (HPA) および ClusterAutoscaler は複数の異なる方法でクラスターリソースを変更します。HPA は、現在の CPU 負荷に基づいてデプロイメント、または ReplicaSet のレプリカ数を変更します。負荷が増大すると、HPA はクラスターで利用できるリソース量に関係なく、新規レプリカを作成します。十分なリソースがない場合、ClusterAutoscaler はリソースを追加し、HPA で作成された Pod が実行できるようにします。負荷が減少する場合、HPA は一部のレプリカを停止します。この動作によって一部のノードの使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

ClusterAutoscaler 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、ClusterAutoscaler 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、ClusterAutoscaler 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 ClusterAutoscaler 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. MACHINEAUTOSCALER について

MachineAutoscaler は、MachineSet 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** MachineSet および作成する他の MachineSet の両方をスケーリングできます。MachineAutoscaler 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 MachineSet に即時に適用されます。

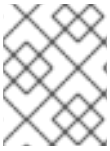


重要

マシンをスケーリングするには、ClusterAutoscaler の MachineAutoscaler をデプロイする必要があります。ClusterAutoscaler は、スケーリングできるリソースを判別するために、MachineAutoscaler が設定するアノテーションを MachineSet で使用します。MachineAutoscaler を定義せずに ClusterAutoscaler を定義する場合、ClusterAutoscaler はクラスターをスケーリングできません。

5.3. CLUSTERAUTOSCALER の設定

まず ClusterAutoscaler 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

ClusterAutoscaler 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 ClusterAutoscaler 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、 ClusterAutoscaler 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) ClusterAutoscaler 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 ClusterAutoscaler 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. ClusterAutoscaler のデプロイ

ClusterAutoscaler 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
```

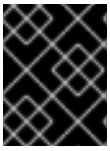
1 **<filename>** は、カスタマイズしたリソースファイルの名前です。

次のステップ

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

5.4. MACHINEAUTOSCALER の設定

ClusterAutoscaler の設定後に、クラスターのスケーリングに使用される MachineSet を参照する MachineAutoscaler リソースをデプロイします。



重要

ClusterAutoscaler リソースの設定後に、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 MachineSet に対して別々のリソースを設定する必要があります。MachineSet 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 MachineSet には1つ以上のマシンが必要です。

5.4.1. MachineAutoscaler リソース定義

この MachineAutoscaler リソース定義は、 MachineAutoscaler 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) **MachineAutoscaler** の名前を指定します。この MachineAutoscaler がスケーリングする MachineSet を簡単に特定できるようにするには、スケーリングする MachineSet の名前を指定するか、またはこれを組み込みます。MachineSet の名前は以下の形式を取ります。 **<clusterid>-<machineset>-<aws-region-az>**

(2) ClusterAutoscaler 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 AWS 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) ClusterAutoscaler がクラス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最大数マシンを指定します。**ClusterAutoscaler** 定義の **maxNodesTotal** 値が、MachineAutoScaler 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 MachineSet 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 MachineSet の名前に一致する必要があります。

5.4.2. MachineAutoscaler のデプロイ

MachineAutoscaler 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
```

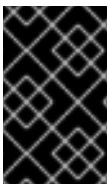
1 **<filename>** は、カスタマイズしたリソースファイルの名前です。

5.5. 追加リソース

- 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 MACHINESET 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 MachineSet 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 MachineSet 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 MACHINESET 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 MachineSet 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インストールし、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 MachineSet が必要であるため、3 つ以上の MachineSet を作成します。

6.2.1. MachineSet 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) リージョンに MachineSet を定義し、**node-role.kubernetes.io/<role>: ""** というラベルが付けられたノードを作成します。

このサンプルでは、**<clusterID>** 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 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: <clusterID> 1
  name: <clusterID>-<role>-us-east-1a 2
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <clusterID> 3
      machine.openshift.io/cluster-api-machineset: <clusterID>-<role>-us-east-1a 4
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <clusterID> 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: <clusterID>-<role>-us-east-1a 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: <clusterID>-worker-profile 11
          instanceType: m4.large
          kind: AWSMachineProviderConfig
          placement:
            availabilityZone: us-east-1a
            region: us-east-1
          securityGroups:
            - filters:
                - name: tag:Name
                  values:
                    - <clusterID>-worker-sg 12
          subnet:
            filters:
              - name: tag:Name
                values:
                  - <clusterID>-private-us-east-1a 13
          tags:
            - name: kubernetes.io/cluster/<clusterID> 14
```

```
      value: owned
  userDataSecret:
    name: worker-user-data
```

1 3 5 11 12 13 14 クラスターをプロビジョニングしたときに設定したクラスター ID を指定します。

2 4 8 クラスター ID とノードラベルを指定します。

6 7 9 追加するノードラベルを指定します。

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

6.2.2. MachineSet 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 MachineSet 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します。
- **oc** として知られる OpenShift コマンドラインインターフェース (CLI) をダウンロードします。
- **cluster-admin** パーミッションを持つユーザーとして、**oc** にログインします。

手順

1. 説明されているように MachineSet カスタムリソース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 MachineSet 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 MachineSet 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。

2 デフォルトのノードラベル。

2. 新規 **MachineSet** を作成します。

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

3. MachineSet 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 MachineSet が利用可能な場合、 **DESIRED** および **CURRENT** の値は一致します。MachineSet 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 MachineSet 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。

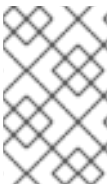
```
$ oc get machine -n openshift-machine-api

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

MachineSet への変更は、MachineSet が所有する既存のマシンには適用されません。たとえば、既存の MachineSet へ編集または追加されたラベルは、既存のマシンおよび MachineSet に関連付けられた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 MachineSet が必要な場合、このプロセスを繰り返して追加の MachineSet を作成します。

6.3. リソースのインフラストラクチャー MACHINESET 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 MachineSet に移行できます。

6.3.1. ルーターの移動

ルーター Pod を異なる MachineSet 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表示されます。

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. 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 -o yaml
```

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

AME                              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       v1.11.0+406fc897d8
```

-

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。

6.3.2. 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 MachineSet 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: ""
```

これらを保存し、終了した後に、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できます。

6.3.3. モニタ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 ConfigMap 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: ""
```

この ConfigMap 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 ConfigMap 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'
```

6.3.4. 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

MachineSet 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と。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** プロジェクトでクラスターロギングのカスタムリソースを編集します。

```
$ oc edit ClusterLogging instance
```

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

....

spec:
  collection:
    logs:
      fluentd:
        resources: null
      rsyslog:
        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。

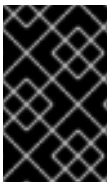
第7章 RHEL コンピュートマシンの OPENSHIFT CONTAINER PLATFORM クラスターへの追加

OpenShift Container Platform では、Red Hat Enterprise Linux (RHEL) のコンピュートまたはワーカーマシンをユーザーによってプロビジョニングされるインフラストラクチャークラスターに追加できます。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. RHEL コンピュートノードのクラスターへの追加について

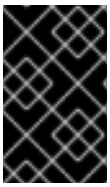
OpenShift Container Platform 4.1 には、ユーザーによってプロビジョニングされるインフラストラクチャーを使用する場合、Red Hat Enterprise Linux (RHEL) マシンをクラスター内のコンピュートまたはワーカーマシンとして使用するオプションがあります。クラスター内のコントロールプレーンまたはマスターマシンには Red Hat Enterprise Linux CoreOS (RHCOS) マシンを使用する必要があります。

ユーザーによってプロビジョニングされるインフラストラクチャーを使用するすべてのインストールの場合、クラスターで RHEL コンピュートマシンを使用する選択をする場合には、システム更新の実行や、パッチの適用、またその他の必要なすべてのタスクの実行を含むオペレーティングシステムのライフサイクル管理およびメンテナンスのすべてを独自に実行する必要があります。



重要

OpenShift Container Platform をクラスター内のマシンから削除するには、オペレーティングシステムを破棄する必要があるため、クラスターに追加する RHEL マシンについては専用のハードウェアを使用する必要があります。



重要

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

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

7.2. 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。OpenShift Container Platform クラスターの管理者は、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 (「最小」のインストールオプション)

重要

OpenShift Container Platform 4.1 でサポートされるのは RHEL 7.6 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。

- 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。

7.2.1. 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。

7.3. 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 4.1 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

- **oc** として知られる OpenShift コマンドラインインターフェース (CLI) を、Playbook を実行するマシンに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 4.1 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-2.7-rpms" \
    --enable="rhel-7-server-ose-4.1-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。

7.4. 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 \*
   ```

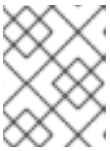
   利用可能なリポジトリーが多い場合には、数分の時間がかかることがあります。

6. OpenShift Container Platform 4.1 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-4.1-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。

7.5. 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 4.1 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る。
- インストール用の RHEL ホストを準備している。

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. クラスターのプルシークレットを抽出します。

```
$ oc -n openshift-config get -o jsonpath='{.data.\.dockerconfigjson}' secret pull-secret |
base64 -d | jq .
```

2. **pull-secret.txt** という名前のファイルにプルシークレットを保存します。
3. 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
openshift_pull_secret_path="~/pull-secret.txt" 4

[new_workers] 5
mycluster-worker-0.example.com
mycluster-worker-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. クラスターのイメージレジストリーのプルシークレットが含まれるファイルへのパスを指定します。
5. 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。

4. 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。

7.6. マシンの CSR 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。

前提条件

- マシンをクラスターに追加していること。
- **jq** パッケージのインストール。

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-0  Ready     master  63m  v1.13.4+b626c2fe1
master-1  Ready     master  63m  v1.13.4+b626c2fe1
master-2  Ready     master  64m  v1.13.4+b626c2fe1
worker-0  NotReady  worker  76s  v1.13.4+b626c2fe1
worker-1  NotReady  worker  70s  v1.13.4+b626c2fe1
```

出力には作成したすべてのマシンが一覧表示されます。

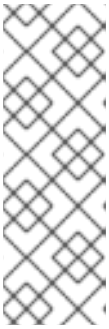
2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr

NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending 1
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending 2
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

1. クライアント要求の CSR。
2. サーバー要求の CSR。

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 CSR が有効な場合、以下のコマンドを実行してそれらすべてを承認します。

```
$ oc get csr -ojson | jq -r '.items[] | select(.status == {} ) | .metadata.name' | xargs oc adm certificate approve
```

7.7. 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。

パラメーター	説明	値
openshift_kubeconfig_path	クラスターの **kubeconfig** ファイルが含まれるローカルディレクトリーへのパスを指定します。	設定ファイルのパスと名前。
openshift_pull_secret_path	クラスターのイメージレジストリーに対するプルシークレットが含まれるテキストファイルへのパスを指定します。OpenShift インフラストラクチャープロバイダーページから取得したプルシークレットを使用します。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する、Quay.io などの組み込まれた各種の認証局によって提供されるサービスで認証できます。	プルシークレットファイルのパスと名前。

7.7.1. 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、Red Hat Enterprise Linux CoreOS (RHCOS) コンピュートマシンを削除できます。

前提条件

- RHEL コンピュートマシンをクラスターに追加している。

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。

第8章 RHEL コンピュートマシンの OPENSHIFT CONTAINER PLATFORM クラスターへのさらなる追加

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

8.1. RHEL コンピュートノードのクラスターへの追加について

OpenShift Container Platform 4.1 には、ユーザーによってプロビジョニングされるインフラストラクチャーを使用する場合、Red Hat Enterprise Linux (RHEL) マシンをクラスター内のコンピュートまたはワーカーマシンとして使用するオプションがあります。クラスター内のコントロールプレーンまたはマスターマシンには Red Hat Enterprise Linux CoreOS (RHCOS) マシンを使用する必要があります。

ユーザーによってプロビジョニングされるインフラストラクチャーを使用するすべてのインストールの場合、クラスターで RHEL コンピュートマシンを使用する選択をする場合には、システム更新の実行や、パッチの適用、またその他の必要なすべてのタスクの実行を含むオペレーティングシステムのライフサイクル管理およびメンテナンスのすべてを独自に実行する必要があります。

重要

OpenShift Container Platform をクラスター内のマシンから削除するには、オペレーティングシステムを破棄する必要があるため、クラスターに追加する RHEL マシンについては専用のハードウェアを使用する必要があります。

重要

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

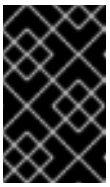
RHEL コンピュートマシンは、コントロールプレーンを初期化してからクラスターに追加する必要があります。

8.2. RHEL コンピュートノードのシステム要件

OpenShift Container Platform 環境の Red Hat Enterprise Linux (RHEL) コンピュートマシンホスト (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。OpenShift Container Platform クラスターの管理者は、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 (「最小」のインストールオプション)



重要

OpenShift Container Platform 4.1 でサポートされるのは RHEL 7.6 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。

- 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。

8.2.1. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

8.3. 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 4.1 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-4.1-rpms"
```

7. ホストで firewalld を停止し、無効にします。

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

注記

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

8.4. 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 4.1 クラスター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る。

- 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** および **pull-secret.txt** ファイルが Playbook の実行に使用するマシン上にある。
- 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る。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る。
- インストール用の RHEL ホストを準備している。
- 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。
- **oc** として知られる OpenShift コマンドラインインターフェース (CLI) を、Playbook を実行するマシンに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"
openshift_pull_secret_path="~/pull-secret.txt"

[workers]
mycluster-worker-0.example.com
mycluster-worker-1.example.com

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

この例では、**mycluster-worker-0.example.com** および **mycluster-worker-1.example.com** マシンがクラスターにあり、**mycluster-worker-2.example.com** および **mycluster-worker-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。

8.5. マシンの CSR 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。

前提条件

- マシンをクラスターに追加していること。
- **jq** パッケージのインストール。

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-0  Ready     master  63m  v1.13.4+b626c2fe1
master-1  Ready     master  63m  v1.13.4+b626c2fe1
master-2  Ready     master  64m  v1.13.4+b626c2fe1
worker-0  NotReady  worker  76s  v1.13.4+b626c2fe1
worker-1  NotReady  worker  70s  v1.13.4+b626c2fe1
```

   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 (1)
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending (2)
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

   (1) クライアント要求の CSR。

   (2) サーバー要求の CSR。

   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 CSR が有効な場合、以下のコマンドを実行してそれらすべてを承認します。

```
$ oc get csr -ojson | jq -r '.items[] | select(.status == {} ) | .metadata.name' | xargs oc adm certificate approve
```

8.6. 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を指定します。	設定ファイルのパスと名前。

パラメーター	説明	値
openshift_pull_secret_path	クラスターのイメージレジストリーに対するプルシークレットが含まれるテキストファイルへのパスを指定します。OpenShift インフラストラクチャープロバイダーページから取得したプルシークレットを使用します。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する、Quay.io などの組み込まれた各種の認証局によって提供されるサービスで認証できます。	プルシークレットファイルのパスと名前。

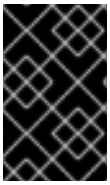
第9章 マシンヘルスチェックのデプロイ

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

マシンヘルスチェックはテクノロジープレビュー機能です。テクノロジープレビュー機能は Red Hat の実稼働環境でのサービスレベルアグリーメント (SLA) ではサポートされていないため、Red Hat では実稼働環境での使用を推奨していません。Red Hat は実稼働環境でこれらを使用することを推奨していません。これらの機能は、近々発表予定の製品機能をリリースに先駆けてご提供することにより、お客様は機能性をテストし、開発プロセス中にフィードバックをお寄せいただくことができます。

Red Hat のテクノロジープレビュー機能のサポート範囲についての詳細は、https://access.redhat.com/ja/support/offerings/techpreview/ を参照してください。

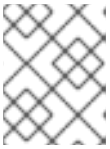


重要

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。

前提条件

- FeatureGate を有効にして、テクノロジープレビュー機能にアクセスできること。



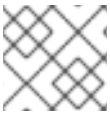
注記

テクノロジープレビュー機能をオンにすると元に戻すことができなくなり、アップグレードができなくなります。

9.1. MACHINEHEALTHCHECK について

MachineHealthCheck は特定の MachinePool の正常ではないマシンを自動的に修復します。

マシンの正常性を監視するには、リソースを作成し、コントローラーの設定を定義します。15 分間 **NotReady** ステータスにすることや、 node-problem-detector に永続的な条件を表示すること、また監視する一連のマシンのラベルなど、チェックする条件を設定します。



注記

マスターロールのあるマシンに MachineHealthCheck を適用することはできません。

MachineHealthCheck リソースを観察するコントローラーは、定義したステータスをチェックします。マシンがヘルスチェックに失敗した場合、これは自動的に検出され、新規マシンがこれに代わって作成されます。マシンが削除されると、**machine deleted** イベントが表示されます。マシンの削除による破壊的な影響を制限するために、コントローラーは 1 度に 1 つのノードのみをドレイン (解放) し、これを削除します。

チェックを停止するには、リソースを削除します。

9.2. 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: healthchecking.openshift.io/v1alpha1
kind: MachineHealthCheck
metadata:
 name: example (1)
 namespace: openshift-machine-api
Spec:
  Selector:
    matchLabels:

      machine.openshift.io/cluster-api-machine-role: <label> (2)
      machine.openshift.io/cluster-api-machine-type: <label> (3)
      machine.openshift.io/cluster-api-machineset: <cluster_name>-<label>-<AWS-zone> (4)
```

(1) デプロイする MachineHealthCheck の名前を指定します。追跡する MachinePool の名前を含めます。

(2) (3) チェックする必要のある MachinePool のラベルを指定します。

(4) 追跡する MachineSet 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。

9.3. MACHINEHEALTHCHECK リソースの作成

クラスターに、**master** プール以外のすべての MachinePools の MachineHealthCheck リソースを作成できます。

前提条件

- **oc** コマンドラインインターフェースをインストールします。

手順

1. MachineHealthCheck 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
```