Red Hat OpenShift Service on AWS 4

クラスター管理

Red Hat OpenShift Service on AWS クラスターの設定

Last Updated: 2024-04-08

Red Hat OpenShift Service on AWS 4 クラスター管理

Red Hat OpenShift Service on AWS クラスターの設定

法律上の通知

概要

本書では、Red Hat OpenShift Service on AWS (ROSA) クラスターの設定について説明します。

目次

第1章 プライベート接続の設定

1.1. プライベート接続の設定

プライベートクラスターのアクセスは、Red Hat OpenShift Service on AWS (ROSA) 環境のニーズに合わせて実装できます。

手順

1. ROSA AWS アカウントにアクセスし、以下の方法のいずれかを使用してクラスターへのプライベート接続を確立します。
   - AWS VPC ピアリングの設定: VPC ピアリングを有効にして、2 つのプライベート IP アドレス間のネットワークトラフィックをルーティングします。
   - AWS VPN の設定: プライベートネットワークを Amazon Virtual Private Cloud にセキュアに接続するために、仮想プライベートネットワークを確立します。
   - AWS Direct Connect の設定: プライベートネットワークと AWS Direct Connect の場所との間に専用のネットワーク接続を確立するように AWS Direct Connect を設定します。
2. ROSA でプライベートクラスターの設定

1.2. AWS VPC ピアリングの設定

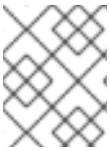
このサンプルプロセスでは、Red Hat OpenShift Service on AWS クラスターが含まれる Amazon Web Services (AWS) VPC を、別の AWS VPC ネットワークとピア接続するように設定します。AWS VPC ピアリング接続の作成または他の設定に関する詳細は、AWS VPC Peering ガイドを参照してください。

1.2.1. VPC ピアリングの用語

2 つの別々の AWS アカウントで 2 つの VPC 間で VPC ピアリング接続を設定する場合は、以下の用語が使用されます。

Red Hat OpenShift Service on AWS の AWS アカウント	Red Hat OpenShift Service on AWS クラスターが含まれる AWS アカウント。
Red Hat OpenShift Service on AWS クラスター VPC	Red Hat OpenShift Service on AWS クラスターが含まれる VPC。
Customer AWS アカウント	ピア接続する Red Hat OpenShift Service on AWS 以外の AWS アカウント。
Customer VPC	ピア接続する AWS アカウントの VPC。

Customer VPC リージョン	お客様の VPC が置かれるリージョン。



注記

2018 年 7 月時点で、AWS は、中国を除く すべての商業地域でのリージョン間の VPC ピアリングをサポートします。

1.2.2. VPC ピア要求の開始

VPC ピアリング接続要求を Red Hat OpenShift Service on AWS アカウントから Customer AWS アカウントに送信できます。

前提条件

- ピア要求を開始するために必要な Customer VPC に関する以下の情報を収集します。
  - Customer AWS アカウント番号
  - Customer VPC ID
  - Customer VPC リージョン
  - Customer VPC CIDR
- Red Hat OpenShift Service on AWS クラスター VPC で使用される CIDR ブロックを確認します。Customer VPC の CIDR ブロックとの重複や一致があると、この 2 つの VPC 間のピアリングは実行できません。詳細は、Amazon VPC の サポートされていない VPC ピアリング設定 を参照してください。CIDR ブロックが重複していない場合は、手順を続行できます。

手順

1. AWS の AWS アカウントで Red Hat OpenShift Service の Web コンソールにログインし、クラスターがホストされるリージョンの **VPC Dashboard** に移動します。
2. **Peering Connections** ページに移動し、 **Create Peering Connection** ボタンをクリックします。
3. ログインしているアカウントの詳細と、接続しているアカウントおよび VPC の詳細を確認します。
   a. **Peering connection name tag** VPC ピアリング接続を説明する名前を設定します。
   b. **VPC (Requester)**: ドロップダウン *リストから Red Hat OpenShift Service on AWS クラスター VPC ID を選択します。
   c. **Account**: **Another account** を選択し、Customer AWS アカウント番号 *(ダッシュなし) を指定します。
   d. **Region**: Customer VPC リージョンが現在のリージョンと異なる場合は、 **Another Region** を選択し、ドロップダウンリストからお客様の VPC リージョンを選択します。
   e. **VPC (Accepter)**: Customer VPC ID を設定します。

4. **Create Peering Connection** をクリックします。

5. 要求の状態が **Pending** になっていることを確認します。**Failed** 状態になった場合は、詳細を確認し、このプロセスを繰り返します。

1.2.3. VPC ピア要求の許可

VPC ピアリング接続を作成したら、Customer AWS アカウントで要求を受け入れる必要があります。

前提条件

- VPC ピア要求を開始します。

手順

1. AWS Web Console にログインします。

2. **VPC Service** に移動します。

3. **Peering Connections** に移動します。

4. **Pending peering connection** をクリックします。

5. 要求の発信元の AWS アカウントおよび VPC ID を確認します。これは、Red Hat OpenShift Service on AWS の AWS アカウントおよび Red Hat OpenShift Service on AWS クラスター VPC から取得する必要があります。

6. **Accept Request** をクリックします。

1.2.4. ルーティングテーブルの設定

VPC のピアリング要求を受け入れた後に、両方の VPC がピアリング接続間で通信するようにルートを設定する必要があります。

前提条件

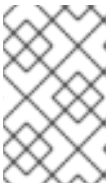
- VPC ピア要求を開始して、受け入れている。

手順

1. Red Hat OpenShift Service on AWS の AWS アカウントで AWS Web Console にログインします。

2. **VPC Service** に移動してから **Route Tables** に移動します。

3. Red Hat OpenShift Service on AWS クラスター VPC の Route Table を選択します。



注記

クラスターによっては、特定の VPC に複数のルートテーブルが存在する場合があります。明示的に関連付けられた多数のサブネットを持つプライベートのルートテーブルを選択します。

4. **Routes** タブを選択してから **Edit** を選択します。

5. **Destination** テキストボックスに Customer VPC CIDR ブロックを入力します。
6. **Target** テキストボックスにピアリング接続 ID を入力します。
7. **Save** をクリックします。
8. 他の VPC の CIDR ブロックで同じプロセスを完了する必要があります。
    a. Customer AWS Web Console → **VPC Service** → **Route Tables** にログインします。
    b. VPC の Route Table を選択します。
    c. **Routes** タブを選択してから **Edit** を選択します。
    d. **Destination** テキストボックスに Red Hat OpenShift Service on AWS クラスター VPC CIDR ブロックを入力します。
    e. **Target** テキストボックスにピアリング接続 ID を入力します。
    f. **Save** をクリックします。

VPC ピアリング接続が完了しました。検証手順に従い、ピアリング接続間の接続が機能していることを確認します。

1.2.5. VPC ピアリング設定の確認およびトラブルシューティング

VPC ピアリング接続を設定したら、これが設定されており、正常に機能していることを確認することが推奨されます。

前提条件

- VPC ピア要求を開始して、受け入れている。
- ルーティングテーブルを設定している。

手順

- AWS コンソールで、ピア接続されるクラスター VPC のルートテーブルを確認します。ルーティングテーブルの設定手順に従い、ピアリング接続のターゲットに VPC CIDR 範囲の宛先を指定するルートテーブルエントリーがあることを確認します。
  正しいルートが Red Hat OpenShift Service on AWS クラスター VPC ルートテーブルおよび Customer VPC ルートテーブルの両方で使用されることが予想される場合は、接続を以下の **netcat** メソッドを使用してテストする必要があります。テスト呼び出しが正常に行われる場合は、VPC ピアリングが正常に機能します。
- エンドポイントデバイスへのネットワーク接続をテストする場合のトラブルシューティングツールとしては、**nc** (または **netcat**) が役に立ちます。これはデフォルトのイメージに含まれ、接続を確立できる場合に迅速かつ明確に出力を提供します。
    a. **busybox** イメージを使用して一時的な Pod を作成します。これは後で自身をクリーンアップします。

```
$ oc run netcat-test \
    --image=busybox -i -t \
    --restart=Never --rm \
    -- /bin/sh
```

b. **nc** を使用して接続を確認します。

- 正常な接続の結果の例:

```
/ nc -zvv 192.168.1.1 8080
10.181.3.180 (10.181.3.180:8080) open
sent 0, rcvd 0
```

- 失敗した接続の結果の例:

```
/ nc -zvv 192.168.1.2 8080
nc: 10.181.3.180 (10.181.3.180:8081): Connection refused
sent 0, rcvd 0
```

c. コンテナーを終了します。これにより、Pod が自動的に削除されます。

```
/ exit
```

1.3. AWS VPN の設定

このサンプルプロセスでは、Amazon Web Services (AWS) Red Hat OpenShift Service on AWS を、お客様のオンサイトのハードウェア VPN デバイスを使用するように設定します。

注記

現時点で AWS VPN は、NAT を VPN トラフィックに適用するための管理オプションを提供しません。詳細は、AWS Knowledge Center を参照してください。

注記

プライベート接続を使用したすべてのトラフィックのルーティング (**0.0.0.0/0** など) はサポートされていません。この場合は、SRE 管理トラフィックを無効にするインターネットゲートウェイを削除する必要があります。

ハードウェア VPN デバイスを使用して AWS VPC をリモートネットワークに接続する方法は、Amazon VPC の VPN Connections ドキュメントを参照してください。

1.3.1. VPN 接続の作成

以下の手順に従って、Amazon Web Services (AWS) Red Hat OpenShift Service on AWS クラスターを、お客様のオンサイトのハードウェア VPN デバイスを使用できるように設定できます。

前提条件

- ハードウェア VPN ゲートウェイデバイスモデルおよびソフトウェアバージョン (例: Cisco ASA バージョン 8.3 を実行)。Amazon VPC の ネットワーク管理者ガイド を参照して、お使いのゲートウェイデバイスが AWS でサポートされているかどうかを確認します。
- VPN ゲートウェイデバイスのパブリック静的 IP アドレス。
- BGP または静的ルーティング: BGP の場合は、ASN が必要です。静的ルーティングの場合は、1 つ以上の静的ルートを設定する必要があります。

- オプション: VPN 接続をテストするための到達可能なサービスの IP およびポート/プロトコル。

1.3.1.1. VPN 接続の設定

手順

1. AWS の AWS Account Dashboard で Red Hat OpenShift Service にログインし、VPC Dashboard に移動します。
2. **Your VPCs** をクリックし、Red Hat OpenShift Service on AWS クラスターが含まれる VPC の名前および VPC ID を特定します。
3. VPC Dashboard から、**Customer Gateway** をクリックします。
4. **Create Customer Gateway** をクリックし、これに意味のある名前を指定します。
5. ルーティング方法 (**Dynamic** または **Static**) を選択します。
6. Dynamic の場合は、表示されるフィールドに BGP ASN を入力します。
7. VPN ゲートウェイエンドポイント IP アドレスに貼り付けます。
8. **Create** をクリックします。
9. 仮想プライベートゲートウェイが目的の VPC に割り当てられていない場合:
    a. VPC Dashboard から、**Virtual Private Gateway** をクリックします。
    b. **Create Virtual Private Gateway**をクリックし、意味のある名前を指定して **Create** をクリックします。
    c. デフォルトの Amazon デフォルト ASN のままにします。
    d. 新たに作成したゲートウェイを選択し、**Attach to VPC** をクリックし、これを以前に指定したクラスター VPC に割り当てます。

1.3.1.2. VPN 接続の確立

手順

1. VPC Dashboard から、**Site-to-Site VPN Connections** をクリックします。
2. **Create VPN Connection**をクリックします。
    a. これに意味のある名前タグを指定します。
    b. 以前に作成した仮想プライベートゲートウェイを選択します。
    c. Customer Gateway については、**Existing** を選択します。
    d. 名前でカスタマーゲートウェイデバイスを選択します。
    e. VPN が BGP を使用する場合は **Dynamic** を選択し、それ以外の場合は **Static** を選択します。静的 IP CIDR を入力します。複数の CIDR がある場合は、各 CIDR を **Another Rule** として追加します。

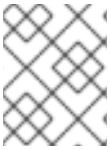
f. **Create** をクリックします。

g. VPN のステータスが **Available** に変更するまで待機します (約 5 分から 10 分)。

3. 作成したばかりの VPN を選択し、**Download Configuration** をクリックします。

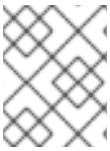
a. ドロップダウンリストから、カスタマーゲートウェイデバイスのベンダー、プラットフォーム、およびバージョンを選択し、**Download** をクリックします。

b. **Generic** ベンダー設定は、プレーンテキスト形式で情報を取得する場合にも利用できます。



注記

VPN 接続が確立されたら、Route Propagation をセットアップしてください。セットアップしない場合、VPN が予想通りに機能しない可能性があります。



注記

VPC サブネット情報をメモします。これは、リモートネットワークとして設定に追加する必要があります。

1.3.1.3. VPN ルート伝播の有効化

VPN 接続を設定したら、必要なルートが VPC のルートテーブルに追加されるように、ルートの伝播が有効にされていることを確認する必要があります。

手順

1. VPC Dashboard から、**Route Tables** をクリックします。

2. Red Hat OpenShift Service on AWS クラスターが含まれる VPC に関連付けられたプライベートルートテーブルを選択します。

注記

クラスターによっては、特定の VPC に複数のルートテーブルが存在する場合があります。明示的に関連付けられた多数のサブネットを持つプライベートのルートテーブルを選択します。

3. **Route Propagation** タブをクリックします。

4. 表示される表に、以前に作成した仮想プライベートゲートウェイが表示されるはずです。**Propagate column** の値を確認します。

a. Propagate (伝播) が **No** に設定されている場合は、**Edit route propagation** をクリックし、仮想プライベートゲートウェイの名前の横にある Propagate チェックボックスを確認して **Save** をクリックします。

VPN トンネルを設定し、AWS がこれを **Up** として検出すると、静的ルートまたは BGP ルートは自動的にルートテーブルに追加されます。

1.3.2. VPN 接続の確認

ご使用の側から VPN トンネルを設定した後に、そのトンネルが AWS コンソールで稼働していること、およびトンネル全体で接続が機能していることを確認します。

前提条件

- VPN 接続を作成します。

手順

1. **トンネルが AWS で稼働していることを確認します。**
   a. VPC Dashboard から、**VPN Connections** をクリックします。
   b. 以前に作成した VPN 接続を選択し、**Tunnel Details** タブをクリックします。
   c. 1 つ以上の VPN トンネルが **Up** になっていることを確認できます。
2. **接続を確認します。**
   エンドポイントデバイスへのネットワーク接続をテストする場合のトラブルシューティングツールとしては、**nc** (または **netcat**) が役に立ちます。これはデフォルトのイメージに含まれ、接続を確立できる場合に迅速かつ明確に出力を提供します。
   a. **busybox** イメージを使用して一時的な Pod を作成します。これは後で自身をクリーンアップします。

```
$ oc run netcat-test \
    --image=busybox -i -t \
    --restart=Never --rm \
    -- /bin/sh
```

   b. **nc** を使用して接続を確認します。
      - 正常な接続の結果の例:

```
/ nc -zvv 192.168.1.1 8080
10.181.3.180 (10.181.3.180:8080) open
sent 0, rcvd 0
```

      - 失敗した接続の結果の例:

```
/ nc -zvv 192.168.1.2 8080
nc: 10.181.3.180 (10.181.3.180:8081): Connection refused
sent 0, rcvd 0
```

   c. コンテナーを終了します。これにより、Pod が自動的に削除されます。

```
/ exit
```

1.3.3. VPN 接続のトラブルシューティング

トンネルが接続されていない
トンネル接続がまだ **ダウン** している場合は、以下を確認できます。

- AWS トンネルは VPN 接続を開始しません。接続の試行はカスタマーゲートウェイから開始する必要があります。
- ソーストラフィックが、設定されたカスタマーゲートウェイと同じ IP から送信されることを確認します。AWS は、ソース IP アドレスが一致しないゲートウェイへのすべてのトラフィックを通知なしでドロップします。
- 設定が AWS でサポートされる 値と一致することを確認します。これには、IKE バージョン、DH グループ、IKE ライフタイムなどが含まれます。
- VPC のルートテーブルを再確認します。伝播が有効で、先にターゲットとして作成した仮想プライベートゲートウェイを持つルートテーブルにエントリーがあることを確認します。
- 中断が発生する可能性があるファイアウォールルールが存在しないことを確認します。
- ポリシーベースの VPN を使用しているかどうかを確認します。これを使用している場合は、その設定によっては複雑な状態が生じる可能性があります。
- トラブルシューティングの手順についての詳細は、AWS ナレッジセンター を参照してください。

トンネルが接続状態にならない
トンネル接続を一貫して **Up** の状態にすることができない場合は、すべての AWS トンネル接続がゲートウェイから開始される必要があることに注意してください。AWS トンネルは トンネリングを開始しません。

Red Hat は、ご使用の側から SLA モニター (Cisco ASA) または一部のデバイスをセットアップすることを推奨しています。これにより、VPC CIDR 範囲内で設定されるすべての IP アドレスで、**ping**、**nc**、**telnet** などの対象 (interesting) トラフィックが絶えず送信されます。接続が成功したかどうかにかかわらず、トラフィックがトンネルにダイレクトされます。

Down 状態のセカンダリートンネル
VPN トンネルが作成されると、AWS は追加のフェイルオーバートンネルを作成します。ゲートウェイデバイスによっては、セカンダリートンネルが **Down** 状態として表示される場合があります。

AWS 通知は以下のようになります。

> You have new non-redundant VPN connections
>
> One or more of your vpn connections are not using both tunnels. This mode of operation is not highly available and we strongly recommend you configure your second tunnel. View your non-redundant VPN connections.

1.4. AWS DIRECT CONNECT の設定

このプロセスでは、Red Hat OpenShift Service on AWS で AWS Direct Connect 仮想インターフェイスを受け入れる方法について説明します。AWS Direct Connect のタイプおよび設定についての詳細は、AWS Direct Connect コンポーネント を参照してください。

1.4.1. AWS Direct Connect の方法

Direct Connect 接続には、Direct Connect Gateway (DXGateway) に接続されるホスト型の仮想インターフェイス (VIF) が必要です。これは、同じまたは別のアカウントでリモート VPC にアクセスするために、Virtual Gateway (VGW) または Transit Gateway に関連付けられます。

既存の DXValidation がない場合、通常のプロセスではホストされた VIF を作成し、Red Hat OpenShift Service on AWS の AWS アカウントに DXValidation および VGW が作成されます。

既存の DXGateway が 1 つ以上の既存の VGW に接続されている場合、このプロセスでは、Red Hat OpenShift Service on AWS の AWS アカウントが Association Proposal (関連付けの提案) を DXGateway の所有者に送信します。DXGateway の所有者は、提案される CIDR が関連付けのあるその他の VGW と競合しないことを確認する必要があります。

詳細は、以下の AWS ドキュメントを参照してください。

- 仮想インターフェイス
- Direct Connect ゲートウェイ
- アカウント間で仮想プライベートゲートウェイを関連付ける

重要

既存の DXValidation への接続は、有料 となります。

選択可能な設定オプションは、以下の 2 つです。

方法 1	ホストされた VIF を作成してから、DXGateway および VGW を作成します。
方法 2	所有する既存の Direct Connect ゲートウェイ経由で接続を要求します。

1.4.2. ホストされた仮想インターフェイスの作成

前提条件

- AWS の AWS Account ID で Red Hat OpenShift Service を収集する。

1.4.2.1. Direct Connect 接続のタイプの決定

Direct Connect 仮想インターフェイスの詳細を表示して、接続の種類を判別します。

手順

1. AWS の AWS Account Dashboard の Red Hat OpenShift Service にログインし、正しいリージョンを選択します。
2. **Services** メニューから **Direct Connect** を選択します。
3. 受け入れを待機している 1 つ以上の仮想インターフェイスがあります。いずれかの仮想インターフェイスを選択して **Summary** を表示します。
4. 仮想インターフェイスタイプ (private または public) を表示します。
5. **Amazon side ASN** の値を記録します。

Direct Connect 仮想インターフェイスの種類が Private の場合は、仮想プライベートゲートウェイが作成されます。Direct Connect Virtual Interface が Public の場合は、Direct Connect Gateway が作成されます。

1.4.2.2. Private Direct Connect の作成

Direct Connect Virtual Interface タイプが Private の場合は、Private Direct Connect が作成されます。

手順

1. AWS の AWS Account Dashboard の Red Hat OpenShift Service にログインし、正しいリージョンを選択します。
2. AWS リージョンで、**Services** メニューから **VPC** を選択します。
3. **VPN Connections** から **Virtual Private Gateways**を選択します。
4. **Create Virtual Private Gateway**をクリックします。
5. 仮想プライベートゲートウェイに適切な名前を指定します。
6. **Custom ASN** を選択し、先に収集された **Amazon side ASN** の値を入力します。
7. 仮想プライベートゲートウェイを作成します。
8. 新規に作成された仮想プライベートゲートウェイをクリックし、**Actions** タブで **Attach to VPC** を選択します。
9. リストから **Red Hat OpenShift Service on AWS Cluster VPC**を選択し、Virtual Private Gateway を VPC に割り当てます。
10. **Services** メニューで、**Direct Connect** をクリックします。リストから Direct Connect 仮想インターフェイスのいずれかを選択します。
11. **I understand that Direct Connect port charges apply once I click Accept Connection**メッセージを確認したら **Accept Connection** を選択します。
12. 仮想プライベートゲートウェイ接続に対して **Accept** を選択し、前の手順で作成した仮想プライベートゲートウェイを選択します。
13. **Accept** を選択して接続を受け入れます。
14. 仮想インターフェイスが複数ある場合は、直前の手順を繰り返します。

1.4.2.3. Public Direct Connect の作成

Direct Connect Virtual Interface タイプが Public の場合は、Public Direct Connect が作成されます。

手順

1. AWS の AWS Account Dashboard の Red Hat OpenShift Service にログインし、正しいリージョンを選択します。
2. AWS の AWS アカウントリージョンの Red Hat OpenShift Service の **Services** メニューから **Direct Connect** を選択します。
3. **Direct Connect Gateways** および **Create Direct Connect Gateway** を選択します。
4. Direct Connect Gateway に適切な名前を付けます。
5. **Amazon side ASN** で、以前に収集した Amazon side ASN 値を入力します。

6. Direct Connect Gateway を作成します。
7. **Services** メニューから **Direct Connect** を選択します。
8. リストから Direct Connect 仮想インターフェイスのいずれかを選択します。
9. **I understand that Direct Connect port charges apply once I click Accept Connection**メッセージを確認したら **Accept Connection** を選択します。
10. Direct Connect Gateway Connection に対して **Accept** を選択し、直前の手順で作成した Direct Connect Gateway を選択します。
11. **Accept** をクリックして接続を受け入れます。
12. 仮想インターフェイスが複数ある場合は、直前の手順を繰り返します。

1.4.2.4. 仮想インターフェイスの確認

Direct Connect 仮想インターフェイスが許可されたら、少しの間待機してから、インターフェイスの状態を確認します。

手順

1. AWS の AWS Account Dashboard の Red Hat OpenShift Service にログインし、正しいリージョンを選択します。
2. AWS の AWS アカウントリージョンの Red Hat OpenShift Service の **Services** メニューから **Direct Connect** を選択します。
3. リストから Direct Connect 仮想インターフェイスのいずれかを選択します。
4. インターフェイスの状態が **Available** になったことを確認します。
5. インターフェイス BGP のステータスが **Up** になったことを確認します。
6. 残りの Direct Connect インターフェイスに対してこの検証を繰り返します。

Direct Connect Virtual Interface が利用可能になった後に、AWS の AWS Account Dashboard で Red Hat OpenShift Service にログインし、ユーザーの側の設定用に Direct Connect 設定ファイルをダウンロードできます。

1.4.3. 既存の Direct Connect ゲートウェイへの接続

前提条件

- AWS VPC の Red Hat OpenShift Service の CIDR 範囲が、関連付けた他の VGW と競合しないことを確認します。
- 以下の情報を収集します。
  - Direct Connect Gateway ID。
  - 仮想インターフェイスに関連付けられた AWS アカウント ID。
  - DXGateway に割り当てられた BGP ASN。必要に応じて、Amazon のデフォルト ASN も使用できます。

手順

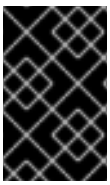
1. AWS の AWS Account Dashboard の Red Hat OpenShift Service にログインし、正しいリージョンを選択します。
2. AWS の AWS アカウントリージョンの Red Hat OpenShift Service の **Services** メニューから **VPC** を選択します。
3. **VPN Connections** から、 **Virtual Private Gateways** を選択します。
4. **Create Virtual Private Gateway** を選択します。
5. 仮想プライベートゲートウェイに適切な名前を指定します。
6. **Custom ASN** をクリックし、以前に使用された **Amazon side ASN** 値を入力するか、Amazon で提供される ASN を使用します。
7. 仮想プライベートゲートウェイを作成します。
8. AWS の AWS Account Dashboard の Red Hat OpenShift Service の **Navigation** ペインで、**Virtual private gateways** を選択してから仮想プライベートゲートウェイを選択します。**View details** を選択します。
9. **Direct Connect gateway associations** を選択し、 **Associate Direct Connect gateway** をクリックします。
10. **Association account type** で、 Account の所有者に **Another account** を選択します。
11. **Direct Connect gateway owner** に、Direct Connect ゲートウェイを所有する AWS アカウントの ID を入力します。
12. Direct Connect ゲートウェイ ID の **Association settings** に、Direct Connect ゲートウェイの ID を入力します。
13. **Association settings** に、仮想インターフェイスの所有者について、関連付けのために仮想インターフェイスを所有する AWS アカウントの ID を入力します。
14. オプション: 接頭辞を Allowed の接頭辞に追加し、それらをコンマで区切ります。
15. **Associate Direct Connect gateway** を選択します。
16. Association Proposal (関連付けの提案) が送信されたら、承認されるのを待っています。実行する必要のある最終手順は、AWS ドキュメンテーション を参照してください。

1.4.4. Direct Connect のトラブルシューティング

トラブルシューティングの詳細は、AWS Direct Connect のトラブルシューティング を参照してください。

第2章 クラスターの自動スケーリング

Red Hat OpenShift Service on AWS に自動スケーリングを適用するには、クラスターオートスケーラーを設定してから、クラスター内の少なくとも 1 つのマシンプールに対してマシンオートスケーラーを設定する必要があります。



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。

クラスターオートスケーラーはクラスターごとに 1 つだけ作成できます。

2.1. 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 Red Hat OpenShift Service on AWS クラスターのサイズを調整します。これは、Kubernetes 形式の宣言引数を使用して、特定のクラウドプロバイダーのオブジェクトに依存しないインフラストラクチャー管理を提供します。Cluster Autoscaler には cluster スコープがあり、特定の namespace には関連付けられていません。

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

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

重要

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

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

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

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

- 制限のある Pod の Disruption Budget (停止状態の予算、PDB) を持つ Pod。

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

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

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

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

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

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

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

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

2.2. OPENSHIFT CLUSTER MANAGER を使用してクラスター作成中に自動スケーリングを有効にする

OpenShift Cluster Manager を使用して、クラスターの作成中に自動スケーリングできます。

手順

1. クラスターの作成中に、**Enable autoscaling** ボックスをオンにします。**Edit cluster autoscaling settings** ボタンが選択可能になります。
    a. 自動スケールするノードの最小数または最大数を選択することもできます。
2. **Edit cluster autoscaling settings**をクリックします。
3. 必要な設定を編集し、**Close** をクリックします。

2.3. OPENSHIFT CLUSTER MANAGER を使用してクラスター作成後に自動スケーリングを有効にする

OpenShift Cluster Manager を使用して、クラスターの作成後に自動スケーリングできます。

手順

1. OpenShift Cluster Manager で、自動スケーリングするクラスターの名前をクリックします。クラスターの概要ページには、**Autoscaling** が有効か無効かを示す項目があります。
2. **Machine pools** タブをクリックします。
3. **Edit cluster autoscaling**ボタンをクリックします。**Edit cluster autoscaling**ウィンドウが表示されます。
4. ウィンドウの上部にある **Autoscale cluster** トグルをクリックします。すべての設定が編集可能になりました。
5. 必要な設定を編集し、**Save** をクリックします。
6. 画面右上の × をクリックして設定画面を閉じます。

すべての自動スケーリング設定が変更されている場合にデフォルトに戻すには、**Revert all to defaults** ボタンをクリックします。

2.4. OPENSHIFT CLUSTER MANAGER を使用したクラスターの自動スケーリング設定

次の表では、OpenShift Cluster Manager でクラスター自動スケーリングを使用する場合に設定可能なすべての UI 設定について説明します。

2.4.1. 一般設定

表2.1 OpenShift Cluster Manager を使用する場合のクラスター自動スケーリングの設定可能な全般設定

設定	説明	型または範囲	デフォルト
log-verbosity	Autoscaler のログレベルを設定します。デフォルト値は 1 です。デバッグには、レベル 4 が推奨されます。レベル 6 はほぼすべてを有効にします。	**integer**	1
skip-nodes-with-local-storage	**true** の場合、クラスターオートスケーラーは、ローカルストレージ (EmptyDir や HostPath など) を持つ Pod を持つノードを削除しません。	**boolean**	true
max-pod-grace-period	スケールダウンする前に Pod に正常な終了時間を秒単位で与えます。	**integer**	600
max-node-provision-time	Cluster Autoscaler がノードのプロビジョニングを待機する最大時間。	**string**	15m
pod-priority-threshold	ユーザーは、クラスターオートスケーラーアクションをトリガーすることが期待されていないベストエフォート Pod をスケジュールできます。これらの Pod は、予備のリソースが利用可能な場合にのみ実行されます。	**integer**	-10
ignore-daemonsets-utilization	スケールダウンのためのリソース使用率を計算するときに、Cluster Autoscaler がデーモンセット Pod を無視するかどうかを決定します。	**boolean**	false
balance-similar-node-groups	**true** の場合、この設定は、同じインスタンスタイプと同じラベルセットを持つノードグループを自動的に識別し、それらのノードグループのそれぞれのサイズのバランスを維持しようとします。	**boolean**	false

設定	説明	型または範囲	デフォルト
balancing-ignored-labels	このオプションは、ノードグループの類似性を考慮するときにクラスターオートスケーラーが無視するラベルを指定します。たとえば、topology.ebs.csi.aws.com/zone ラベルが付いたノードがある場合、このラベルの名前を追加して、クラスターオートスケーラーがその値に基づいてノードを異なるノードグループに分割しないようにすることができます。このオプションにはスペースを使用できません。	**array (string)**	形式は、コンマで区切られたラベルのリストである必要があります。

2.4.2. リソース制限

表2.2 OpenShift Cluster Manager を使用する場合のクラスター自動スケーリングの設定可能なリソース制限設定

設定	説明	型または範囲	デフォルト
cores-total-min	クラスター内のコアの最小数。Cluster Autoscaler は、この数値より小さいクラスターをスケーリングしません。	**object**	0
cores-total-max	クラスター内のコアの最大数。クラスターオートスケーラーは、この数値を超えるクラスターをスケーリングしません。	**object**	180 * 64 (11520)
memory-total-min	クラスター内のメモリーの最小ギガバイト数。Cluster Autoscaler は、この数値より小さいクラスターをスケーリングしません。	**object**	0

設定	説明	型または範囲	デフォルト
memory-total-max	クラスター内のメモリーの最大ギガバイト数。クラスターオートスケーラーは、この数値を超えるクラスターをスケーリングしません。	**object**	180 * 64 * 20 (230400)
max-nodes-total	すべてのノードグループのノードの最大数。自動的にスケールされたノードだけでなく、すべてのノードが含まれます。Cluster Autoscaler は、この数値を超えてクラスターを拡大しません。	**integer**	180
GPU	クラスター内の異なる GPU の最小数と最大数。Cluster Autoscaler は、これらの数値より小さくても大きくてもクラスターをスケーリングしません。	**array**	形式は、"<gpu_type>:<min>:<max>" のコンマ区切りのリストである必要があります。

2.4.3. スケールダウン設定

表2.3 OpenShift Cluster Manager を使用する場合の Cluster Autoscaler のスケールダウン設定を設定可能

設定	説明	型または範囲	デフォルト
scale-down-enabled	Cluster Autoscaler はクラスターをスケールダウンする必要があります。	**boolean**	true
scale-down-utilization-threshold	ノード使用率レベル。要求されたリソースの合計を容量で割ったものとして定義され、このレベルを下回るとノードのスケールダウンが考慮されます。	**float**	0.5
scale-down-unneeded-time	スケールダウンの対象となる前にノードが不要になるまでの期間。	**string**	10m

設定	説明	型または範囲	デフォルト
scale-down-delay-after-add	スケールアップ後、スケールダウン評価が再開されるまでの期間。	**string**	10m
scale-down-delay-after-delete	ノードの削除後、スケールダウン評価が再開されるまでの時間。	**string**	0s
scale-down-delay-after-failure	スケールダウンの失敗後、スケールダウン評価が再開されるまでの期間。	**string**	3m

2.5. ROSA CLI の対話モードを使用して、クラスターの作成中に自動スケーリングを有効にする

ターミナルの対話モード (利用可能な場合) を使用して、クラスターの作成中にクラスター全体の自動スケーリング動作を設定できます。

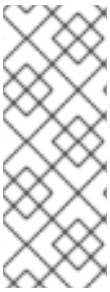
対話型モードでは、使用可能な設定可能なパラメーターに関する詳細情報が提供されます。対話型モードでは、基本的なチェックとプリフライト検証も実行されます。つまり、指定された値が無効な場合、端末は有効な入力を求めるプロンプトを出力します。

手順

- クラスターの作成中に、**--enable-autoscaling** パラメーターと **--interactive** パラメーターを使用してクラスターの自動スケーリングを有効にします。

  以下に例を示します。

```
$ rosa create cluster --cluster-name <cluster_name> --enable-autoscaling --interactive
```



注記

クラスター名が 15 文字を超える場合、***.openshiftapps.com** にプロビジョニングされたクラスターのサブドメインとして自動生成されたドメイン接頭辞が含まれます。

サブドメインをカスタマイズするには、**--domain-prefix** フラグを使用します。ドメイン接頭辞は 15 文字を超えてはならず、一意である必要があり、クラスターの作成後に変更できません。

次のプロンプトが表示されたら、**y** を入力して、使用可能なすべての自動スケーリングオプションを実行します。

対話型プロンプトの例:

```
? Configure cluster-autoscaler (optional): [? for help] (y/N) y <enter>
```

2.5.1. ROSA CLI の対話モードを使用してクラスター作成後に自動スケーリングを有効にする

ターミナルの対話モード (利用可能な場合) を使用して、クラスターの作成後にクラスター全体の自動スケーリング動作を設定できます。

手順

- クラスターを作成した後、次のコマンドを入力します。

  以下に例を示します。

  ```
  $ rosa create autoscaler --cluster=<mycluster> --interactive
  ```

  その後、利用可能なすべての自動スケーリングパラメーターを設定できます。

2.6. ROSA CLI を使用してクラスター作成中に自動スケーリングを有効にする

ROSA CLI (**rosa**) を使用すると、クラスターの作成時にクラスター全体の自動スケーリング動作を設定できます。オートスケーラーはマシン全体で有効にすることも、クラスターのみで有効にすることもできます。

手順

- クラスターの作成中に、クラスター名の後に **--enable autoscaling** と入力して、マシンの自動スケーリングを有効にします。

注記

クラスター名が 15 文字を超える場合、***.openshiftapps.com** にプロビジョニングされたクラスターのサブドメインとして自動生成されたドメイン接頭辞が含まれます。

サブドメインをカスタマイズするには、**--domain-prefix** フラグを使用します。ドメイン接頭辞は 15 文字を超えてはならず、一意である必要があり、クラスターの作成後に変更できません。

以下に例を示します。

```
$ rosa create cluster --cluster-name <cluster_name> --enable-autoscaling
```

次のコマンドを実行して、クラスターの自動スケーリングを有効にするために少なくとも 1 つのパラメーターを設定します。

以下に例を示します。

```
$ rosa create cluster --cluster-name <cluster_name> --enable-autoscaling <parameter>
```

2.6.1. ROSA CLI を使用してクラスター作成後に自動スケーリングを有効にする

ROSA CLI (**rosa**) を使用して、クラスターの作成後にクラスター全体の自動スケーリングを設定できます。

手順

- クラスターを作成した後、オートスケーラーを作成します。

  以下に例を示します。

  ```
  $ rosa create autoscaler --cluster=<mycluster>
  ```

  a. 次のコマンドを使用して、特定のパラメーターを使用してオートスケーラーを作成することもできます。

     以下に例を示します。

     ```
     $ rosa create autoscaler --cluster=<mycluster> <parameter>
     ```

2.6.2. ROSA CLI を使用してクラスター作成後に自動スケーリングを編集する

オートスケーラーの作成後に、Cluster Autoscaler の特定のパラメーターを編集できます。

- Cluster Autoscaler を編集するには、次のコマンドを実行します。

  以下に例を示します。

  ```
  $ rosa edit autoscaler --cluster=<mycluster>
  ```

  a. 特定のパラメーターを編集するには、次のコマンドを実行します。

     以下に例を示します。

     ```
     $ rosa edit autoscaler --cluster=<mycluster> <parameter>
     ```

2.6.3. ROSA CLI を使用して自動スケーリングを削除する

Cluster Autoscaler を使用しなくなった場合は、削除できます。

- Cluster Autoscaler を削除するには、次のコマンドを実行します。

  以下に例を示します。

  ```
  $ rosa delete autoscaler --cluster=<mycluster>
  ```

2.7. ROSA CLI を使用したクラスター自動スケーリングパラメーター

ROSA CLI (**rosa**) を使用する場合、クラスター作成コマンドに次のパラメーターを追加してオートスケーラーパラメーターを設定できます。

表2.4 ROSA CLI (rosa) で利用可能な設定可能なオートスケーラーパラメーター

設定	説明	型または範囲	例/命令
--autoscaler-balance-similar-node-groups	同じインスタンスタイプとラベルセットを持つノードグループを特定し、それらのノードグループのそれぞれのサイズのバランスを試みます。	**boolean**	true に設定するには追加し、false に設定するにはオプションを省略します。
--autoscaler-skip-nodes-with-local-storage	設定されている場合、Cluster Autoscaler は、EmptyDir や HostPath などのローカルストレージを持つ Pod を持つノードを削除しません。	**boolean**	true に設定するには追加し、false に設定するにはオプションを省略します。
--autoscaler-log-verbosity int	Autoscaler のログレベル。コマンドの **int** を使用する数値に置き換えます。	**integer**	**--autoscaler-log-verbosity 4**
--autoscaler-max-pod-grace-period int	スケールダウンする前の Pod の正常な終了時間を秒単位で指定します。コマンドの **int** を、使用する秒数に置き換えます。	**integer**	**--autoscaler-max-pod-grace-period 0**
--autoscaler-pod-priority-threshold int	Cluster Autoscaler が追加のノードをデプロイするために Pod が超える必要がある優先度。コマンドの **int** を使用する数値に置き換えます。負の値も使用できます。	**integer**	**--autoscaler-pod-priority-threshold -10**
--autoscaler-gpu-limit stringArray	クラスター内の異なる GPU の最小数と最大数。Cluster Autoscaler は、これらの数値より小さくても大きくてもクラスターをスケーリングしません。形式は、"<gpu_type>,<min>,<max>" のコンマ区切りのリストである必要があります。	**array**	**--autoscaler-gpu-limit nvidia.com/gpu,0,10 --autoscaler-gpu-limit amd.com/gpu,1,5**

設定	説明	型または範囲	例/命令
--autoscaler-ignore-daemonsets-utilization	設定されている場合、cluster-autoscaler は、スケールダウンのためのリソース使用率を計算するときに、デーモンセット Pod を無視します。	**boolean**	true に設定するには追加し、false に設定するにはオプションを省略します。
--autoscaler-max-node-provision-time string	Cluster Autoscaler がノードのプロビジョニングを待機する最大時間。コマンド内の **文字列** を整数と時間単位 (ns、us、µs、ms、s、m、h) に置き換えます。	**string**	**--autoscaler-max-node-provision-time 35m**
--autoscaler-balancing-ignored-labels strings	ノードグループの類似性を比較するときに Cluster Autoscaler が無視する必要があるラベルキーのコンマ区切りのリスト。コマンド内の **文字列** を関連するラベルに置き換えます。	**string**	**--autoscaler-balancing-ignored-labels topology.ebs.csi.aws.com/zone,alpha.eksctl.io/instance-id**
--autoscaler-max-nodes-total int	クラスター内のノードの最大数 (自動スケールされたノードを含む)。コマンドの **int** を使用する数値に置き換えます。	**integer**	**--autoscaler-max-nodes-total 180**
--autoscaler-min-cores int	クラスターにデプロイするコアの最小数。コマンドの **int** を使用する数値に置き換えます。	**integer**	**--autoscaler-min-cores 0**
--autoscaler-max-cores int	クラスターにデプロイするコアの最大数。コマンドの **int** を使用する数値に置き換えます。	**integer**	**--autoscaler-max-cores 100**
--autoscaler-min-memory int	クラスター内の最小メモリー量 (GiB 単位)。コマンドの **int** を使用する数値に置き換えます。	**integer**	**--autoscaler-min-memory 0**
--autoscaler-max-memory int	クラスター内の最大メモリー量 (GiB 単位)。コマンドの **int** を使用する数値に置き換えます。	**integer**	**--autoscaler-max-memory 4096**

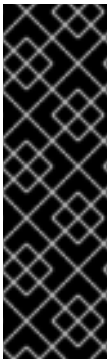
設定	説明	型または範囲	例/命令
--autoscaler-scale-down-enabled	設定されている場合、Cluster Autoscaler はクラスターをスケールダウンする必要があります。	**boolean**	true に設定するには追加し、false に設定するにはオプションを省略します。
--autoscaler-scale-down-unneeded-time string	スケールダウンの対象となる前にノードが不要になるまでの期間。コマンド内の **文字列** を整数と時間単位 (ns、us、µs、ms、s、m、h) に置き換えます。	**string**	**--autoscaler-scale-down-unneeded-time 1h**
--autoscaler-scale-down-utilization-threshold float	ノード使用率レベル。要求されたリソースの合計を容量で割ったものとして定義され、このレベルを下回るとノードのスケールダウンが考慮されます。値は 0 から 1 の間である必要があります。	**float**	**--autoscaler-scale-down-utilization-threshold 0.5**
--autoscaler-scale-down-delay-after-add string	スケールアップ後、スケールダウン評価が再開されるまでの期間。コマンド内の **文字列** を整数と時間単位 (ns、us、µs、ms、s、m、h) に置き換えます。	**string**	**--autoscaler-scale-down-delay-after-add 1h**
--autoscaler-scale-down-delay-after-delete string	ノードの削除後、スケールダウン評価が再開されるまでの時間。コマンド内の **文字列** を整数と時間単位 (ns、us、µs、ms、s、m、h) に置き換えます。	**string**	**--autoscaler-scale-down-delay-after-delete 1h**
--autoscaler-scale-down-delay-after-failure string	スケールダウンの失敗後、スケールダウン評価が再開されるまでの期間。コマンド内の **文字列** を整数と時間単位 (ns、us、µs、ms、s、m、h) に置き換えます。	**string**	**--autoscaler-scale-down-delay-after-failure 1h**

第3章 マシンプールを使用したノードの管理

3.1. マシンプールについて

Red Hat OpenShift Service on AWS は、クラウドインフラストラクチャーで柔軟性があり動的なプロビジョニング方法としてマシンプールを使用します。

主要なリソースは、マシン、コンピューティングマシンセット、およびマシンプールです。



重要

Red Hat OpenShift Service on AWS 4.11 の時点で、Pod ごとのデフォルトの PID 制限は **4096** です。この PID 制限を有効する場合は、Red Hat OpenShift Service on AWS クラスターをこのバージョン以降にアップグレードする必要があります。以前のバージョンで実行されている Red Hat OpenShift Service on AWS クラスターでは、デフォルトの PID 制限である **1024** が使用されます。

ROSA CLI を使用すると、Red Hat OpenShift Service on AWS クラスターの Pod ごとの PID 制限を設定できます。詳細は、「PID 制限の設定」を参照してください。

3.1.1. マシン

マシンは、ワーカーノードのホストを記述する基本的な単位です。

3.1.2. マシンセット

MachineSet リソースは、計算マシンのグループです。より多くのマシンが必要な場合、またはマシンをスケールダウンする必要がある場合は、コンピュートマシンセットが属するマシンプール内のレプリカの数を変更します。

マシンセットは ROSA で直接変更することができません。

3.1.3. マシンプール

マシンプールは、マシンセットを計算するための上位レベルの構造です。

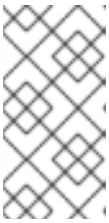
コンピュートマシンプールは、アベイラビリティーゾーン全体で同じ設定のクローンがすべて含まれるマシンセットを作成します。マシンプールは、ワーカーノードですべてのホストノードのプロビジョニング管理アクションを実行します。より多くのマシンが必要な場合、またはマシンをスケールダウンする必要がある場合は、コンピュートのニーズに合わせてマシンプール内のレプリカの数を変更してください。スケーリングは手動または自動の設定ができます。

デフォルトでは、クラスターに 1 つのマシンプールがあります。クラスターのインストール中に、インスタンスのタイプまたはサイズを定義し、このマシンプールにラベルを追加できます。

クラスターのインストール後:

- 任意のマシンプールに対してラベルを削除または追加できます。
- 既存のクラスターに追加のマシンプールを追加できます。
- テイントのないマシンプールが 1 つ存在する場合は、任意のマシンプールにテイントを追加できます。

- 汚染のないマシンプールが1つと、シングル AZ クラスターの場合は少なくとも 2 つのレプリカ、マルチ AZ クラスターの場合は少なくとも 3 つのレプリカが存在する限り、マシンプールを作成または削除できます。



注記

マシンプールノードのタイプやサイズは変更できません。マシンプールノードのタイプまたはサイズは、作成時にのみ指定されます。別のノードタイプまたはサイズが必要な場合は、マシンプールを再作成し、必要なノードタイプまたはサイズの値を指定する必要があります。

- 追加された各マシンプールにラベルを追加できます。

単一のクラスター上に複数のマシンプールが存在でき、各マシンプールには一意のノードタイプとノードサイズ設定を含めることができます。

3.1.4. 複数のゾーンクラスターのマシンプール

複数のアベイラビリティーゾーン (AZ) にわたって作成されたクラスターでは、3 つの AZ すべてまたは、任意の単一 AZ にわたるマシンプールを作成できます。デフォルトでクラスターの作成時に作られるマシンプールは、3 つの AZ すべてでマシンを作成し、3 の倍数にスケーリングします。

新しい Multi-AZ クラスターを作成すると、マシンプールはそれらのゾーンに自動的に複製されます。デフォルトでは、既存の Multi-AZ クラスターにマシンプールを追加すると、新しいマシンプールがすべてのゾーンに自動的に作成されます。

注記

このデフォルト設定をオーバーライドして、選択した Single-AZ にマシンプールを作成できます。

同様に、マシンプールを削除するとすべてのゾーンから削除されます。この相乗効果により、Multi-AZ クラスターでマシンプールを使用すると、マシンプールを作成するときに、特定のリージョンに対するプロジェクトのクォータをより多く消費する可能性があります。

3.1.5. 関連情報

- コンピュートノードの管理
- 自動スケーリングについて
- PID 制限の設定

3.2. コンピュートノードの管理

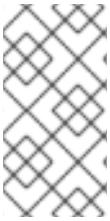
本書では、Red Hat OpenShift Service on AWS (ROSA) でコンピュート (ワーカーとも呼ばれる) ノードを管理する方法について説明します。

コンピュートノードの変更の大半は、マシンプールで設定されます。マシンプール は、管理を容易にするために、同じ設定を持つクラスター内のコンピュートノードのグループです。

スケーリング、ノードラベルの追加、テイントの追加などのマシンプール設定オプションを編集できます。

3.2.1. マシンセットの作成

マシンプールは、Red Hat OpenShift Service on AWS (ROSA) クラスターのインストール時に作成されます。インストール後に、OpenShift Cluster Manager または ROSA CLI (**rosa**) を使用して、クラスターの追加のマシンプールを作成できます。



注記

ROSA CLI **rosa** バージョン 1.2.25 以前のバージョンのユーザーの場合、クラスターとともに作成されたマシンプールは **Default** として識別されます。ROSA CLI **rosa** バージョン 1.2.26 以降のユーザーの場合、クラスターとともに作成されたマシンプールは **worker** として識別されます。

3.2.1.1. OpenShift Cluster Manager を使用したマシンプールの作成

OpenShift Cluster Manager を使用して、Red Hat OpenShift Service on AWS (ROSA) クラスターに追加のマシンプールを作成できます。

前提条件

- ROSA クラスターを作成している。

手順

1. OpenShift Cluster Manager に移動し、クラスターを選択します。
2. **Machine pools** タブで、**Add machine pool** をクリックします。
3. **マシンプール名** を追加します。
4. ドロップダウンメニューから **Compute node instance type**を選択します。インスタンスタイプは、マシンプール内の各コンピュートノードの仮想 CPU およびメモリー割り当てを定義します。

注記

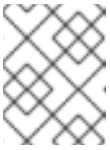
プールを作成した後に、マシンプールのインスタンスタイプを変更することはできません。

5. オプション: マシンプールの自動スケーリングを設定します。
   a. **Enable autoscaling** を選択し、デプロイメントのニーズを満たすためにマシンプール内のマシン数を自動的にスケーリングします。
   b. 自動スケーリングの最小および最大のノード数制限を設定します。Cluster Autoscaler は、指定する制限を超えてマシンプールノード数を減らしたり、増やしたりできません。
      - 単一アベイラビリティーゾーンを使用してクラスターをデプロイした場合は、**最小および最大のノード数** を設定します。これは、アベイラビリティーゾーンのコンピュートノードの最小および最大の制限を定義します。
      - 複数のアベイラビリティーゾーンを使用してクラスターをデプロイした場合は、**Minimum nodes per zone**および **Maximum nodes per zone** を設定します。これは、ゾーンごとの最小および最大のコンピュート制限を定義します。

注記

または、マシンプールの作成後にマシンプールの自動スケーリングを設定できます。

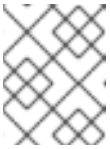
6. 自動スケーリングを有効にしていない場合は、コンピュートノードの数を選択します。
   - 単一アベイラビリティーゾーンを使用してクラスターをデプロイした場合は、ドロップダウンメニューから **コンピュートノード数** を選択します。これは、ゾーンのマシンプールにプロビジョニングするコンピュートノードの数を定義します。
   - 複数のアベイラビリティーゾーンを使用してクラスターをデプロイした場合は、ドロップダウンメニューから **コンピュートノードの数 (ゾーンごと)** を選択します。これは、ゾーンごとにマシンプールにプロビジョニングするコンピュートノードの数を定義します。
7. オプション: **ルートディスクのサイズ** を設定します。
8. オプション: マシンプールのノードラベルおよびテイントを追加します。
   a. **Edit node labels and taints** メニューをデプロイメントします。
   b. **Node labels** で、ノードラベルの **Key** および **Value** のエントリーを追加します。
   c. **Taints** で、テイントの **Key** および **Value** エントリーを追加します。



注記

テイントを含むマシンプールの作成は、クラスターにテイントのないマシンプールが少なくとも 1 つすでに存在する場合にのみ可能です。

   d. テイントごとに、ドロップダウンメニューから **Effect** を選択します。使用できるオプションには、**NoSchedule**、**PreferNoSchedule**、および **NoExecute** が含まれます。



注記

または、マシンプールの作成後にノードラベルおよびテイントを追加できます。

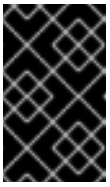
9. オプション: このマシンプール内のノードに使用する追加のカスタムセキュリティーグループを選択します。すでにセキュリティーグループを作成し、このクラスター用に選択した VPC にそのグループを関連付けている必要があります。マシンプールの作成後は、セキュリティーグループを追加または編集することはできません。詳細は、関連情報セクションのセキュリティーグループの要件を参照してください。

重要

ROSA with HCP クラスターのマシンプールには、最大 10 個の追加セキュリティーグループを使用できます。

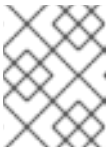
10. オプション: マシンプールを保証なしの AWS Spot インスタンスとしてデプロイするように設定するには、Amazon EC2 Spot インスタンスを使用します。
    a. **Use Amazon EC2 Spot Instances** を選択します。

b. オンデマンドのインスタンス価格を使用するには、**Use On-Demand instance price** を選択したままにします。または、**Set maximum price** を選択して、Spot インスタンスの 1 時間ごとの最大価格を定義します。
Amazon EC2 Spot インスタンスの詳細は、AWS のドキュメント を参照してください。



重要

Amazon EC2 Spot インスタンスはいつでも中断する可能性があります。Amazon EC2 Spot インスタンスは、中断に対応できるワークロードにのみ使用します。



注記

マシンプールに **Use Amazon EC2 Spot Instances** を選択すると、マシンプールの作成後にオプションを無効にすることはできません。

11. **Add machine pool** をクリックしてマシンプールを作成します。

検証

- マシンプールが **Machine pools** ページに表示され、設定が想定どおりに表示されていることを確認します。

関連情報

- 追加のカスタムセキュリティーグループ

3.2.1.2. ROSA CLI を使用したマシンプールの作成

ROSA CLI (**rosa**) を使用して、Red Hat OpenShift Service on AWS (ROSA) クラスターの追加のマシンプールを作成できます。

前提条件

- 最新の Red Hat OpenShift Service on AWS (ROSA) CLI (**rosa**) をワークステーションにインストールして設定している。
- ROSA CLI (**rosa**) を使用して Red Hat アカウントにログインしている。
- ROSA クラスターを作成している。

手順

- 自動スケーリングを使用しないマシンプールを追加するには、マシンプールを作成し、インスタンスタイプ、コンピュート (ワーカーとも呼ばれる) ノード数、およびノードラベルを定義します。

```
$ rosa create machinepool --cluster=<cluster-name> \
                          --name=<machine_pool_id> \ 1
                          --replicas=<replica_count> \ 2
                          --instance-type=<instance_type> \ 3
                          --labels=<key>=<value>,<key>=<value> \ 4
                          --taints=<key>=<value>:<effect>,<key>=<value>:<effect> \ 5
```

```
--use-spot-instances \ 6
--spot-max-price=0.5 \ 7
--disk-size=<disk_size> 8
--availability-zone=<availability_zone_name> 9
--additional-security-group-ids <sec_group_id> 10
--subnet string 11
```

1. マシンプールの名前を指定します。**<machine_pool_id>** をマシンプールの名前に置き換えます。

2. プロビジョニングするコンピュートノードの数を指定します。単一アベイラビリティーゾーンを使用して ROSA をデプロイしている場合は、ゾーンのマシンプールにプロビジョニングするコンピュートノードの数を定義します。複数のアベイラビリティーゾーンを使用してクラスターをデプロイしている場合は、全ゾーンでプロビジョニングするコンピュートノードの数を定義し、その数は 3 の倍数である必要があります。**--replicas** 引数は、自動スケーリングが設定されていない場合に必要です。

3. オプション: マシンプールのコンピュートノードのインスタンスタイプを設定します。インスタンスタイプは、プール内の各コンピュートノードの仮想 CPU およびメモリー割り当てを定義します。**<instance_type>** をインスタンスタイプに置き換えます。デフォルトは **m5.xlarge** です。プールを作成した後に、マシンプールのインスタンスタイプを変更することはできません。

4. オプション: マシンプールのラベルを定義します。**<key>=<value>,<key>=<value>** は、キーと値のペアのコンマ区切りリストに置き換えます (例: **--labels=key1=value1,key2=value2**)。

5. オプション: マシンプールのテイントを定義します。**<key>=<value>:<effect>,<key>=<value>:<effect>** は、各テイントのキー、値、および影響に置き換えます (例: **--taints=key1=value1:NoSchedule,key2=value2:NoExecute**)。利用可能な影響には、**NoSchedule**、**PreferNoSchedule**、および **NoExecute** が含まれます。

6. オプション: マシンプールは、保証なしの AWS Spot インスタンスとしてデプロイするように設定します。詳細は、AWS ドキュメントの Amazon EC2 Spot Instances を参照してください。マシンプールに **Use Amazon EC2 Spot Instances** を選択すると、マシンプールの作成後にオプションを無効にすることはできません。

7. オプション: Spot インスタンスを使用する場合は、この引数を指定して Spot インスタンスの 1 時間ごとの最大価格を定義できます。この引数が指定されていない場合は、オンデマンドの価格が使用されます。

重要

Amazon EC2 Spot インスタンスはいつでも中断する可能性があります。Amazon EC2 Spot インスタンスは、中断に対応できるワークロードにのみ使用します。

8. オプション: ワーカーノードのディスクサイズを指定します。値は GB、GiB、TB、または TiB 単位で指定できます。**<disk_size>** は、数値と単位に置き換えます (例: **--disk-size=200GiB**)。

9. オプション: Multi-AZ クラスターの場合、選択した Single-AZ にマシンプールを作成できます。**<az>** は、Single-AZ 名に置き換えます。

注記

Multi-AZ クラスターは、Multi-AZ コントロールプレーンを保持し、Single-AZ または Multi-AZ 全体でワーカーマシンプールを持つことができます。マシンプールは、アベイラビリティーゾーン全体にマシン (ノード) を均等に分散します。

警告

Single-AZ のワーカーマシンプールを選択した場合、マシンレプリカの数に関係なく、そのマシンプールには耐障害性がありません。耐障害性のあるワーカーマシンプールの場合は、Multi-AZ マシンプールを選択すると、アベイラビリティーゾーン全体に 3 の倍数でマシンが分散されます。

- 3 つのアベイラビリティーゾーンを持つマルチ AZ マシンプールは、3、6、9 など、3 の倍数でのみマシン数を指定できます。
- アベイラビリティーゾーンが 1 つの Single-AZ マシンプールは、1、2、3、4 など、1 の倍数のマシン数を指定できます。

10 オプション: Red Hat 管理の VPC がないクラスター内のマシンプールの場合、マシンプールで使用する追加のカスタムセキュリティーグループを選択できます。すでにセキュリティーグループを作成し、このクラスター用に選択した VPC にそのグループを関連付けている必要があります。マシンプールの作成後は、セキュリティーグループを追加または編集することはできません。詳細は、関連情報セクションのセキュリティーグループの要件を参照してください。

重要

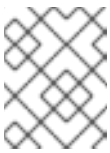
ROSA with HCP クラスターのマシンプールには、最大 10 個の追加セキュリティーグループを使用できます。

11 オプション: BYO VPC クラスターの場合、サブネットを選択してシングル AZ マシンプールを作成できます。サブネットがクラスター作成サブネットの外にある場合は、キーが **kubernetes.io/cluster/<infra-id>** で値が **shared** のタグが必要です。お客様は、次のコマンドを使用して Infra ID を取得できます。

```
$ rosa describe cluster -c <cluster name>|grep "Infra ID:"
```

出力例

```
Infra ID:                   mycluster-xqvj7
```



注記

--subnet と **--availability-zone** の両方を同時に設定できません。Single-AZ マシンプールの作成には 1 つだけが許可されます。

以下の例では、**m5.xlarge** インスタンスタイプを使用し、コンピュートノードレプリカが 2 つ含まれる **mymachinepool** という名前のマシンプールを作成します。この例では、ワークロード固有のラベルも 2 つ追加します。

```
$ rosa create machinepool --cluster=mycluster --name=mymachinepool --replicas=2 --
instance-type=m5.xlarge --labels=app=db,tier=backend
```

出力例

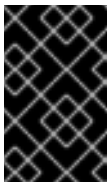
```
I: Machine pool 'mymachinepool' created successfully on cluster 'mycluster'
I: To view all machine pools, run 'rosa list machinepools -c mycluster'
```

- 自動スケーリングを使用するマシンプールを追加するには、マシンプールを作成して、自動スケーリング設定、インスタンスタイプ、およびノードラベルを定義します。

```
$ rosa create machinepool --cluster=<cluster-name> \
                          --name=<machine_pool_id> \ 1
                          --enable-autoscaling \ 2
                          --min-replicas=<minimum_replica_count> \ 3
                          --max-replicas=<maximum_replica_count> \ 4
                          --instance-type=<instance_type> \ 5
                          --labels=<key>=<value>,<key>=<value> \ 6
                          --taints=<key>=<value>:<effect>,<key>=<value>:<effect> \ 7
                          --use-spot-instances \ 8
                          --spot-max-price=0.5 9
                          --availability-zone=<availability_zone_name> 10
```

1. マシンプールの名前を指定します。**<machine_pool_id>** をマシンプールの名前に置き換えます。
2. マシンプールの自動スケーリングを有効にし、デプロイメントのニーズに対応します。
3. 4. コンピュートノードの最小および最大の制限を定義します。Cluster Autoscaler は、指定する制限を超えてマシンプールノード数を減らしたり、増やしたりできません。単一アベイラビリティーゾーンを使用して ROSA をデプロイしている場合、**--min-replicas** 引数および **--max-replicas** 引数は、ゾーンのマシンプールに自動スケーリング制限を定義します。複数のアベイラビリティーゾーンを使用してクラスターをデプロイしている場合に、すべてのゾーンにおける自動スケーリングの制限を引数で定義し、その数は 3 の倍数である必要があります。
5. オプション: マシンプールのコンピュートノードのインスタンスタイプを設定します。インスタンスタイプは、プール内の各コンピュートノードの仮想 CPU およびメモリー割り当てを定義します。**<instance_type>** をインスタンスタイプに置き換えます。デフォルトは **m5.xlarge** です。プールを作成した後に、マシンプールのインスタンスタイプを変更することはできません。
6. オプション: マシンプールのラベルを定義します。**<key>=<value>,<key>=<value>** は、キーと値のペアのコンマ区切りリストに置き換えます (例: **--labels=key1=value1,key2=value2**)。
7. オプション: マシンプールのテイントを定義します。**<key>=<value>:<effect>,<key>=<value>:<effect>** は、各テイントのキー、値、および影響に置き換えます (例: **--taints=key1=value1:NoSchedule,key2=value2:NoExecute**)。利用可能な影響には、**NoSchedule**、**PreferNoSchedule**、および **NoExecute** が含まれます。

8 オプション: マシンプールは、保証なしの AWS Spot インスタンスとしてデプロイするように設定します。詳細は、AWS ドキュメントの Amazon EC2 Spot Instances を参照して



重要

Amazon EC2 Spot インスタンスはいつでも中断する可能性があります。Amazon EC2 Spot インスタンスは、中断に対応できるワークロードにのみ使用します。

9 オプション: Spot インスタンスを使用する場合は、この引数を指定して Spot インスタンスの 1 時間ごとの最大価格を定義できます。この引数が指定されていない場合は、オンデマンドの価格が使用されます。

10 オプション: Multi-AZ クラスターの場合、選択した Single-AZ にマシンプールを作成できます。**<az>** は、Single-AZ 名に置き換えます。

以下の例では、**m5.xlarge** インスタンスタイプを使用し、自動スケーリングが有効になっている **mymachinepool** という名前のマシンプールを作成します。コンピュートノードの最小制限は 3 で、最大制限は全体で 6 です。この例では、ワークロード固有のラベルも 2 つ追加します。

```
$ rosa create machinepool --cluster=mycluster --name=mymachinepool --enable-autoscaling
--min-replicas=3 --max-replicas=6 --instance-type=m5.xlarge --labels=app=db,tier=backend
```

出力例

```
I: Machine pool 'mymachinepool' created successfully on cluster 'mycluster'
I: To view all machine pools, run 'rosa list machinepools -c mycluster'
```

検証

クラスターのすべてのマシンプールをリストするか、個々のマシンプールの詳細情報を表示します。

1. クラスターで使用可能なマシンプールをリスト表示します。

```
$ rosa list machinepools --cluster=<cluster_name>
```

出力例

```
ID           AUTOSCALING  REPLICAS  INSTANCE TYPE  LABELS              TAINTS
AVAILABILITY ZONES                SPOT INSTANCES
Default      No           3         m5.xlarge                          us-east-1a, us-east-1b, us-
east-1c    N/A
mymachinepool  Yes        3-6       m5.xlarge      app=db, tier=backend            us-east-1a,
us-east-1b, us-east-1c    No
```

2. クラスター内の特定のマシンプールの詳細情報を表示します。

```
$ rosa describe machinepool --cluster=<cluster_name> mymachinepool
```

出力例

```
ID:                 mymachinepool
Cluster ID:         27iimopsg1mge0m81l0sqivkne2qu6dr
Autoscaling:        Yes
Replicas:           3-6
Instance type:      m5.xlarge
Labels:             app=db, tier=backend
Taints:
Availability zones: us-east-1a, us-east-1b, us-east-1c
Subnets:
Spot instances:     No
Disk size:          300 GiB
Security Group IDs:
```

3. マシンプールが出力に含まれ、設定が想定どおりであることを確認します。

関連情報

- 追加のカスタムセキュリティーグループ

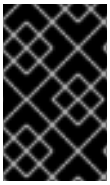
3.2.2. マシンプールディスクボリュームの設定

マシンプールのディスクボリュームサイズは、柔軟性を高めるために設定できます。デフォルトのディスクサイズは 300 GiB です。クラスターバージョン 4.13 以前の場合、ディスクサイズは最小 128 GiB から最大 1 TiB まで設定できます。クラスターバージョン 4.14 以降の場合、ディスクサイズは最小 128 GiB から最大 16 TiB まで設定できます。

OpenShift Cluster Manager または ROSA CLI (**rosa**) を使用して、クラスターのマシンプールのディスクサイズを設定できます。

注記

既存のクラスターおよびマシンプールノードのボリュームのサイズは変更できません。



重要

デフォルトのディスクサイズは 300 GiB です。クラスターバージョン 4.13 以前の場合、ディスクサイズは最小 128 GiB から最大 1 TiB まで設定できます。クラスターバージョン 4.14 以降の場合、ディスクサイズは最小 128 GiB から最大 16 TiB まで設定できます。

3.2.2.1. OpenShift Cluster Manager を使用したマシンプールのディスクボリュームの設定

クラスター作成の前提条件

- クラスターのインストール中に、デフォルトのマシンプールのノードディスクサイズを選択するオプションがあります。

クラスター作成の手順

1. ROSA クラスターウィザードから、Cluster settings に移動します。
2. **Machine pool** の手順に移動します。
3. 目的の **Root disk size** を選択します。

4. **Next** を選択してクラスターの作成を続行します。

マシンプール作成の前提条件

- クラスターのインストール後に、新しいマシンプールのノードディスクサイズを選択するオプションがあります。

マシンプール作成の手順

1. OpenShift Cluster Manager に移動し、クラスターを選択します。
2. **Machine pool タブ** に移動します。
3. **Add machine pool** をクリックします。
4. 目的の **Root disk size** を選択します。
5. **Add machine pool** を選択してマシンプールを作成します。

3.2.2.2. ROSA CLI を使用したマシンプールディスクボリュームの設定

クラスター作成の前提条件

- クラスターのインストール中に、デフォルトのマシンプールのルートディスクのサイズを選択するオプションがあります。

クラスター作成の手順

- 必要なルートディスクサイズの OpenShift クラスターを作成するときに、次のコマンドを実行します。

```
$ rosa create cluster --worker-disk-size=<disk_size>
```

値は GB、GiB、TB、または TiB 単位で指定できます。<disk_size> は、数値と単位に置き換えます (例: --worker-disk-size=200GiB)。数字と単位は、分離できません。空白は、使用できます。

マシンプール作成の前提条件

- クラスターのインストール後に、新しいマシンプールのルートディスクのサイジングを選択するオプションがあります。

マシンプール作成の手順

1. 以下のコマンドを実行してクラスターをスケールアップします。

```
$ rosa create machinepool --cluster=<cluster_id> 1
                          --disk-size=<disk_size> 2
```

1. 既存の OpenShift クラスターの ID または名前を指定します。
2. ワーカーノードのディスクサイズを指定します。値は GB、GiB、TB、または TiB 単位で指定できます。<disk_size> は、数値と単位に置き換えます (例: --disk-size=200GiB)。数字と単位は、分離できません。空白は、使用できます。

2. AWS コンソールにログインして新規のマシンプールのディスクサイズを確認し、EC2 仮想マシンのルートボリュームサイズを見つけます。

関連情報

- **rosa create machinepool** サブコマンドで利用可能な引数の詳細な一覧は、rosa CLI を使用したオブジェクトの管理 を参照してください。

3.2.3. マシンプールの削除

ワークロード要件が変更され、現在のマシンプールがニーズを満たさなくなった場合は、マシンプールを削除できます。

マシンプールは、OpenShift Cluster Manager または ROSA CLI (**rosa**) を使用して削除できます。

3.2.3.1. OpenShift Cluster Manager を使用したマシンプールの作成

OpenShift Cluster Manager を使用して、Red Hat OpenShift Service on AWS (ROSA) クラスターのマシンプールを削除できます。

前提条件

- ROSA クラスターを作成している。
- クラスターが準備状態にある。
- テイントのない既存のマシンプールがあり、シングル AZ クラスターの場合は少なくとも 2 つのインスタンス、マルチ AZ クラスターの場合は少なくとも 3 つのインスタンスがある。

手順

1. OpenShift Cluster Manager から、**Clusters** ページに移動し、削除するマシンプールを含むクラスターを選択します。
2. 選択したクラスターで、**Machine pools** タブを選択します。
3. **Machine pools** タブで、削除するマシンプールのオプションメニュー をクリックします。
4. 削除をクリックします。

選択したマシンプールが削除されます。

3.2.3.2. ROSA CLI を使用したマシンプールの削除

ROSA CLI を使用して、Red Hat OpenShift Service on AWS (ROSA) クラスターのマシンプールを削除できます。

注記

ROSA CLI **rosa** バージョン 1.2.25 以前のユーザーの場合、クラスターと一緒に作成されたマシンプール (ID='Default') は削除できません。ROSA CLI **rosa** バージョン 1.2.26 以降を使用している場合、クラスター内にテイントを含まないマシンプールが 1 つある限り、クラスターとともに作成されたマシンプール (ID='worker') を削除できます。シングル AZ クラスターの場合は少なくとも 2 つのレプリカ、マルチ AZ クラスターの場合は少なくとも 3 つのレプリカがあれば、削除できます。

前提条件

- ROSA クラスターを作成している。
- クラスターが準備状態にある。
- テイントのない既存のマシンプールがあり、シングル AZ クラスターの場合は少なくとも 2 つのインスタンス、マルチ AZ クラスターの場合は少なくとも 3 つのインスタンスがある。

手順

1. ROSA CLI から次のコマンドを実行します。

```
$ rosa delete machinepool -c=<cluster_name> <machine_pool_ID>
```

出力例

```
? Are you sure you want to delete machine pool <machine_pool_ID> on cluster
<cluster_name>? (y/N)
```

2. y を入力してマシンプールを削除します。
選択したマシンプールが削除されます。

3.2.4. コンピュートノードの手動によるスケーリング

マシンプールの自動スケーリングを有効にしていない場合は、デプロイメントのニーズに合わせてプール内のコンピュート (ワーカーとも呼ばれる) ノードの数を手動でスケーリングできます。

各マシンプールを個別にスケーリングする必要があります。

前提条件

- 最新の Red Hat OpenShift Service on AWS (ROSA) CLI (**rosa**) をワークステーションにインストールして設定している。
- ROSA CLI (**rosa**) を使用して Red Hat アカウントにログインしている。
- Red Hat OpenShift Service on AWS (ROSA) クラスターを作成している。
- 既存のマシンプールがある。

手順

1. クラスターのマシンプールをリスト表示します。

```
$ rosa list machinepools --cluster=<cluster_name>
```

出力例

```
ID       AUTOSCALING   REPLICAS    INSTANCE TYPE  LABELS    TAINTS
AVAILABILITY ZONES    DISK SIZE   SG IDs
default   No           2           m5.xlarge                  us-east-1a          300GiB      sg-
0e375ff0ec4a6cfa2
mp1       No           2           m5.xlarge                  us-east-1a          300GiB      sg-
0e375ff0ec4a6cfa2
```

2. マシンプール内のコンピュートノードのレプリカ数を増減します。

```
$ rosa edit machinepool --cluster=<cluster_name> \
                        --replicas=<replica_count> \ 1
                        <machine_pool_id> 2
```

1 単一アベイラビリティーゾーンを使用して Red Hat OpenShift Service on AWS (ROSA) をデプロイしている場合、レプリカ数はゾーンのマシンプールにプロビジョニングするコンピュートノードの数を定義します。複数のアベイラビリティーゾーンを使用してクラスターをデプロイしている場合は、すべてのゾーンでマシンプール内のコンピュートノードの合計数を定義し、その数は 3 の倍数である必要があります。

2 **<machine_pool_id>** を、前述のコマンドの出力に表示されているマシンプールの ID に置き換えます。

検証

1. クラスターで利用可能なマシンプールをリスト表示します。

```
$ rosa list machinepools --cluster=<cluster_name>
```

出力例

```
ID       AUTOSCALING   REPLICAS    INSTANCE TYPE  LABELS    TAINTS
AVAILABILITY ZONES    DISK SIZE   SG IDs
default   No           2           m5.xlarge                  us-east-1a          300GiB      sg-
0e375ff0ec4a6cfa2
mp1       No           3           m5.xlarge                  us-east-1a          300GiB      sg-
0e375ff0ec4a6cfa2
```

2. 上記のコマンドの出力で、コンピュートノードのレプリカ数がマシンプールで想定通りに設定されていることを確認します。この出力例では、**mp1** マシンプールのコンピュートノードレプリカ数は 3 にスケーリングされています。

3.2.5. ノードラベル

ラベルは、**Node** オブジェクトに適用されるキーと値のペアです。ラベルを使用して一連のオブジェクトを整理し、Pod のスケジューリングを制御できます。

クラスターの作成中または後にラベルを追加できます。ラベルはいつでも変更または更新できます。

関連情報

- ラベルの詳細は、Kubernetes ラベルおよびセレクターの概要 を参照してください。

3.2.5.1. ノードラベルのマシンプールへの追加

いつでもコンピュート (ワーカーとも呼ばれる) ノードのラベルを追加または編集して、適切な方法でノードを管理します。たとえば、ワークロードのタイプを特定のノードに割り当てることができます。

ラベルは key-value ペアとして割り当てられます。各キーは、割り当てられたオブジェクトに固有のものである必要があります。

前提条件

- 最新の Red Hat OpenShift Service on AWS (ROSA) CLI (**rosa**) をワークステーションにインストールして設定している。
- ROSA CLI (**rosa**) を使用して Red Hat アカウントにログインしている。
- Red Hat OpenShift Service on AWS (ROSA) クラスターを作成している。
- 既存のマシンプールがある。

手順

1. クラスターのマシンプールをリスト表示します。

```
$ rosa list machinepools --cluster=<cluster_name>
```

出力例

```
ID          AUTOSCALING  REPLICAS  INSTANCE TYPE  LABELS    TAINTS
AVAILABILITY ZONES    SPOT INSTANCES
Default     No           2         m5.xlarge                  us-east-1a          N/A
db-nodes-mp  No          2         m5.xlarge                  us-east-1a          No
```

2. マシンプールのノードラベルを追加または更新します。

   - 自動スケーリングを使用しないマシンプールのノードラベルを追加または更新するには、以下のコマンドを実行します。

```
$ rosa edit machinepool --cluster=<cluster_name> \
                        --replicas=<replica_count> \ 1
                        --labels=<key>=<value>,<key>=<value> \ 2
                        <machine_pool_id>
```

1 自動スケーリングを使用しないマシンプールの場合は、ノードラベルの追加時にレプリカ数を指定する必要があります。**--replicas** 引数を指定しないと、コマンドが完了する前にレプリカ数の入力を求めるプロンプトが出されます。単一アベイラビリティーゾーンを使用して Red Hat OpenShift Service on AWS (ROSA) をデプロイしている場合、レプリカ数はゾーンのマシンプールにプロビジョニングするコンピュートノードの数を定義します。複数のアベイラビリティーゾーンを使用してクラスターをデプロイしている場合は、すべてのゾーンでマシンプール内のコンピュートノードの合計数を定義し、その数は 3 の倍数である必要があります。

2 **<key>=<value>,<key>=<value>** は、キーと値のペアのコンマ区切りリストに置き換えます (例: **--labels=key1=value1,key2=value2**)。このリストは、継続的にノードラベルに加えられるすべての変更を上書きします。

以下の例では、ラベルを **db-nodes-mp** マシンプールに追加します。

```
$ rosa edit machinepool --cluster=mycluster --replicas=2 --labels=app=db,tier=backend
db-nodes-mp
```

出力例

```
I: Updated machine pool 'db-nodes-mp' on cluster 'mycluster'
```

- 自動スケーリングを使用するマシンプールのノードラベルを追加または更新するには、以下のコマンドを実行します。

```
$ rosa edit machinepool --cluster=<cluster_name> \
                        --min-replicas=<minimum_replica_count> \ 1
                        --max-replicas=<maximum_replica_count> \ 2
                        --labels=<key>=<value>,<key>=<value> \ 3
                        <machine_pool_id>
```

1 2 自動スケーリングを使用するマシンプールの場合は、最小および最大のコンピュートノードレプリカ制限を指定する必要があります。引数を指定しないと、コマンドが完了する前に値の入力が求められます。Cluster Autoscaler は、指定する制限を超えてマシンプールノード数を減らしたり、増やしたりできません。単一アベイラビリティーゾーンを使用して ROSA をデプロイしている場合、**--min-replicas** 引数および **--max-replicas** 引数は、ゾーンのマシンプールに自動スケーリング制限を定義します。複数のアベイラビリティーゾーンを使用してクラスターをデプロイしている場合に、すべてのゾーンにおける自動スケーリングの制限を引数で定義し、その数は 3 の倍数である必要があります。

3 **<key>=<value>,<key>=<value>** は、キーと値のペアのコンマ区切りリストに置き換えます (例: **--labels=key1=value1,key2=value2**)。このリストは、継続的にノードラベルに加えられるすべての変更を上書きします。

以下の例では、ラベルを **db-nodes-mp** マシンプールに追加します。

```
$ rosa edit machinepool --cluster=mycluster --min-replicas=2 --max-replicas=3 --
labels=app=db,tier=backend db-nodes-mp
```

出力例

```
I: Updated machine pool 'db-nodes-mp' on cluster 'mycluster'
```

検証

1. 新しいラベルを持つマシンプールの詳細情報を表示します。

```
$ rosa describe machinepool --cluster=<cluster_name> <machine-pool-name>
```

出力例

出力例

```
ID:                 db-nodes-mp
Cluster ID:         <ID_of_cluster>
Autoscaling:        No
Replicas:           2
Instance type:      m5.xlarge
Labels:             app=db, tier=backend
Taints:
Availability zones: us-east-1a
Subnets:
Spot instances:     No
Disk size:          300 GiB
Security Group IDs:
```

2. 出力内のマシンプールにラベルが含まれていることを確認します。

3.2.6. マシンプールへのテイントの追加

マシンプールにコンピュート (ワーカーとも呼ばれる) ノードにテイントを追加して、そのノードにスケジュールされる Pod を制御できます。テイントをマシンプールに適用すると、Pod 仕様にテイントの容認が含まれない限り、スケジューラーは Pod をプールに配置できません。テイントは、OpenShift Cluster Manager または Red Hat OpenShift Service on AWS (ROSA) CLI、**rosa** を使用してマシンプールに追加できます。

注記

クラスターにはテイントを含まないマシンプールが少なくとも 1 つ必要です。

3.2.6.1. Openshift Cluster Manager を使用したマシンプールへのテイントの追加

OpenShift Cluster Manager を使用して、Red Hat OpenShift Service on AWS (ROSA) クラスターのマシンプールにテイントを追加できます。

前提条件

- Red Hat OpenShift Service on AWS (ROSA) クラスターを作成している。
- テイントを含まず、少なくとも 2 つのインスタンスを含む既存のマシンプールがある。

手順

1. OpenShift Cluster Manager に移動し、クラスターを選択します。
2. **Machine pools** タブで、テイントを追加するマシンプールのオプションメニュー  をクリックします。
3. **Edit taints** を選択します。
4. テイントの**Key** と **Value** のエントリーを追加します。
5. ドロップダウンメニューからテイントの **Effect** を選択します。使用できるオプションには、**NoSchedule**、**PreferNoSchedule**、および **NoExecute** が含まれます。

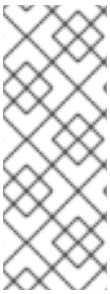
6. 任意: マシンプールにテイントを追加する場合は、**Add taint** を選択します。
7. **Save** をクリックして、テイントをマシンプールに適用します。

検証

1. **Machine pools** タブで、マシンプールの横にある > を選択して、ビューをデプロイメントします。
2. デプロイメントされたビューの **Taints** の下にテイントがリストされていることを確認します。

3.2.6.2. ROSA CLI を使用したマシンプールへのテイントの追加

ROSA CLI を使用して、Red Hat OpenShift Service on AWS (ROSA) クラスターのマシンプールにテイントを追加できます。



注記

ROSA CLI **rosa** バージョン 1.2.25 以前のバージョンを使用している場合、クラスターとともに作成されたマシンプール (ID= **Default**) 内のテイント数を変更できません。ROSA CLI **rosa** バージョン 1.2.26 以降を使用している場合、クラスターとともに作成されたマシンプール (ID= **worker**) 内でテイントの数を変更できます。テイントのないマシンプールが 1 つ以上、Single-AZ クラスターの場合はレプリカが 2 つ以上、Multi-AZ クラスターの場合はレプリカが 3 つ以上ある必要があります。

前提条件

- ワークステーションに最新の AWS (**aws**)、ROSA (**rosa**)、OpenShift (**oc**) の CLI をインストールして設定している。
- **rosa** CLI を使用して Red Hat アカウントにログインしている。
- Red Hat OpenShift Service on AWS (ROSA) クラスターを作成している。
- テイントを含まず、少なくとも 2 つのインスタンスを含む既存のマシンプールがある。

手順

1. 次のコマンドを実行して、クラスター内のマシンプールをリスト表示します。

```
$ rosa list machinepools --cluster=<cluster_name>
```

出力例

```
ID          AUTOSCALING  REPLICAS  INSTANCE TYPE  LABELS    TAINTS
AVAILABILITY ZONES    SPOT INSTANCES     DISK SIZE   SG IDs
Default     No           2         m5.xlarge                 us-east-1a          N/A              300 GiB
sg-0e375ff0ec4a6cfa2
db-nodes-mp  No           2         m5.xlarge                 us-east-1a          No               300
GiB     sg-0e375ff0ec4a6cfa2
```

2. マシンプールのテイントを追加または更新します。

- 自動スケーリングを使用しないマシンプールのテイントを追加または更新するには、以下のコマンドを実行します。

```
$ rosa edit machinepool --cluster=<cluster_name> \
                        --replicas=<replica_count> \ 1
                        --taints=<key>=<value>:<effect>,<key>=<value>:<effect> \ 2
                        <machine_pool_id>
```

1 自動スケーリングを使用しないマシンプールの場合は、テイントの追加時にレプリカ数を指定する必要があります。**--replicas** 引数を指定しないと、コマンドが完了する前にレプリカ数の入力を求めるプロンプトが出されます。単一アベイラビリティーゾーンを使用して Red Hat OpenShift Service on AWS (ROSA) をデプロイしている場合、レプリカ数はゾーンのマシンプールにプロビジョニングするコンピュートノードの数を定義します。複数のアベイラビリティーゾーンを使用してクラスターをデプロイしている場合は、すべてのゾーンでマシンプール内のコンピュートノードの合計数を定義し、その数は 3 の倍数である必要があります。

2 **<key>=<value>:<effect>,<key>=<value>:<effect>** は、各テイントのキー、値、および影響に置き換えます (例: **--taints=key1=value1:NoSchedule,key2=value2:NoExecute**)。影響として **NoSchedule**、**PreferNoSchedule**、および **NoExecute** が使用できます。このリストは、ノードテイントに加えられた変更を継続的に上書きします。

以下の例では、テイントを **db-nodes-mp** マシンプールに追加します。

```
$ rosa edit machinepool --cluster=mycluster --replicas 2 --
taints=key1=value1:NoSchedule,key2=value2:NoExecute db-nodes-mp
```

出力例

```
I: Updated machine pool 'db-nodes-mp' on cluster 'mycluster'
```

- 自動スケーリングを使用するマシンプールのテイントを追加または更新するには、以下のコマンドを実行します。

```
$ rosa edit machinepool --cluster=<cluster_name> \
                        --min-replicas=<minimum_replica_count> \ 1
                        --max-replicas=<maximum_replica_count> \ 2
                        --taints=<key>=<value>:<effect>,<key>=<value>:<effect> \ 3
                        <machine_pool_id>
```

1 2 自動スケーリングを使用するマシンプールの場合は、最小および最大のコンピュートノードレプリカ制限を指定する必要があります。引数を指定しないと、コマンドが完了する前に値の入力が求められます。Cluster Autoscaler は、指定する制限を超えてマシンプールノード数を減らしたり、増やしたりできません。単一アベイラビリティーゾーンを使用して ROSA をデプロイしている場合、**--min-replicas** 引数および **--max-replicas** 引数は、ゾーンのマシンプールに自動スケーリング制限を定義します。複数のアベイラビリティーゾーンを使用してクラスターをデプロイしている場合に、すべてのゾーンにおける自動スケーリングの制限を引数で定義し、その数は 3 の倍数である必要があります。

3

<key>=<value>:<effect>,<key>=<value>:<effect> は、各テイントのキー、値、および影響に置き換えます (例: **--**

以下の例では、テイントを **db-nodes-mp** マシンプールに追加します。

```
$ rosa edit machinepool --cluster=mycluster --min-replicas=2 --max-replicas=3 --
taints=key1=value1:NoSchedule,key2=value2:NoExecute db-nodes-mp
```

出力例

```
I: Updated machine pool 'db-nodes-mp' on cluster 'mycluster'
```

検証

1. 新しいテイントを含むマシンプールの詳細情報を表示します。

```
$ rosa describe machinepool --cluster=<cluster_name> <machine-pool-name>
```

出力例

```
ID:                 db-nodes-mp
Cluster ID:         <ID_of_cluster>
Autoscaling:        No
Replicas:           2
Instance type:      m5.xlarge
Labels:
Taints:             key1=value1:NoSchedule, key2=value2:NoExecute
Availability zones: us-east-1a
Subnets:
Spot instances:     No
Disk size:          300 GiB
Security Group IDs:
```

2. 出力にマシンプールのテイントが含まれていることを確認します。

3.2.7. マシンプールへのノードチューニングの追加

マシンプール内のコンピュートノード (ワーカーノードとも呼ばれる) のチューニングを追加して、Hosted Control Plane (HCP) を備えた Red Hat OpenShift Service on AWS (ROSA) クラスターの設定を制御できます。

注記

この機能は、Hosted Control Plane (HCP) を備えた Red Hat OpenShift Service on AWS (ROSA) クラスターでのみサポートされます。

前提条件

- 最新の Red Hat OpenShift Service on AWS (ROSA) CLI (**rosa**) をワークステーションにインストールして設定している。
- ROSA CLI を使用して Red Hat アカウントにログインしている。

- Hosted Control Plane (HCP) を備えた Red Hat OpenShift Service on AWS (ROSA) を作成している。
- 既存のマシンプールがある。
- 既存のチューニング設定がある。

手順

1. クラスター内のすべてのマシンプールをリスト表示します。

```
$ rosa list machinepools --cluster=<cluster_name>
```

出力例

```
ID          AUTOSCALING  REPLICAS  INSTANCE TYPE [...] AVAILABILITY ZONES
SUBNET  VERSION  AUTOREPAIR  TUNING CONFIGS
workers     No           2         m5.xlarge    [...] us-east-1a          N/A     4.12.14  Yes
db-nodes-mp  No          2         m5.xlarge    [...] us-east-1a          No      4.12.14  Yes
```

2. 既存または新規のマシンプールにチューニング設定を追加できます。

   a. マシンプールの作成時にチューニングを追加します。

```
$ rosa create machinepool -c <cluster-name> <machinepoolname> --tuning-configs
<tuning_config_name>
```

出力例

```
? Tuning configs: sample-tuning
I: Machine pool 'db-nodes-mp' created successfully on hosted cluster 'sample-cluster'
I: To view all machine pools, run 'rosa list machinepools -c sample-cluster'
```

   b. マシンプールのチューニングを追加または更新します。

```
$ rosa edit machinepool -c <cluster-name> <machinepoolname> --tuning-configs
<tuningconfigname>
```

出力例

```
I: Updated machine pool 'db-nodes-mp' on cluster 'mycluster'
```

検証

1. クラスターで利用可能なマシンプールをリスト表示します。

```
$ rosa list machinepools --cluster=<cluster_name>
```

出力例

```
ID          AUTOSCALING  REPLICAS  INSTANCE TYPE [...] AVAILABILITY ZONES
SUBNET  VERSION  AUTOREPAIR  TUNING CONFIGS
```

```
workers      No          2         m5.xlarge    [...] us-east-1a          N/A     4.12.14  Yes
db-nodes-mp  No          2         m5.xlarge    [...] us-east-1a          No      4.12.14  Yes
sample-tuning
```

2. マシンプールのチューニング設定が出力に含まれていることを確認します。

3.2.8. 関連情報

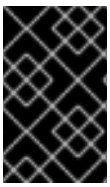
- マシンプールについて
- 自動スケーリングについて
- 自動スケーリングの有効化
- 自動スケーリングの無効化
- ROSA サービス定義

3.3. ローカルゾーンでのマシンプールの設定

このドキュメントでは、Red Hat OpenShift Service on AWS (ROSA) を使用してマシンプールでローカルゾーンを設定する方法を説明します。

3.3.1. ローカルゾーンでのマシンプールの設定

ローカルゾーンでマシンプールを設定するには、次の手順を使用します。

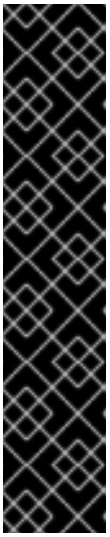


重要

AWS ローカルゾーンは Red Hat OpenShift Service on AWS 4.12 でサポートされています。ローカルゾーンを有効にする方法は、Red Hat ナレッジベースの記事 を参照してください。

前提条件

- Red Hat OpenShift Service on AWS (ROSA) が、選択した親リージョンで一般提供されている。特定の AWS リージョンで利用できるローカルゾーンを確認するには、AWS の一般利用可能なロケーションのリスト を参照してください。
- ROSA クラスターは当初、既存の Amazon VPC (BYO-VPC) 内に構築されている。
- ROSA クラスターの最大伝送単位 (MTU) は 1200 に設定されている。



重要

通常、ローカルゾーンの Amazon EC2 インスタンスとリージョンの Amazon EC2 インスタンス間の最大転送単位 (MTU) は 1300 です。AWS ドキュメントの How Local Zones work を参照してください。オーバーヘッドを考慮して、クラスターネットワーク MTU は常に EC2 MTU より小さくなければなりません。特定のオーバーヘッドは、ネットワークプラグインにより決定されます。以下に例を示します。

- OVN-Kubernetes: **100 bytes**
- OpenShift SDN: **50 bytes**

ネットワークプラグインは、MTU を減らす可能性のある追加機能を提供する可能性があります。詳細は、ドキュメントを参照してください。

- AWS アカウントでは、ローカルゾーンが有効 になっている。
- AWS アカウントには、クラスターと同じ VPC の ローカルゾーンサブネット がある。
- AWS アカウントに、NAT ゲートウェイへのルートを持つルーティングテーブルに関連付けられたサブネットがある。
- AWS アカウントには、関連付けられたサブネット上にタグ kubernetes.io/cluster/<infra_id>:shared がある。

手順

1. 次の ROSA CLI (**rosa**) コマンドを実行して、クラスター上にマシンプールを作成します。

```
$ rosa create machinepool -c <cluster-name> -i
```

2. ROSA CLI でマシンプールのサブネットとインスタンスタイプを追加します。数分後、クラスターはノードをプロビジョニングします。

```
I: Enabling interactive mode (1)
? Machine pool name: xx-lz-xx (2)
? Create multi-AZ machine pool: No (3)
? Select subnet for a single AZ machine pool (optional): Yes (4)
? Subnet ID: subnet-<a> (region-info) (5)
? Enable autoscaling (optional): No (6)
? Replicas: 2 (7)
I: Fetching instance types (8)
? disk-size (optional): (9)
```

(1) 対話モードを有効にします。

(2) マシンプールに名前を付けます。これは英数字に制限されており、最大長は 30 文字です。

(3) このオプションを no に設定します。

(4) このオプションを yes に設定します。

5. リストからサブネット ID を選択します。
6. 自動スケーリングを有効にする場合は yes を選択し、自動スケーリングを無効にする場合は no を選択します。
7. マシンプールのマシンの数を選択します。この数値は 1 ~ 180 の範囲で指定できます。
8. リストからインスタンスタイプを選択します。選択したローカルゾーンでサポートされているインスタンスタイプのみが表示されます。
9. オプション: ワーカーノードのディスクサイズを指定します。値は GB、GiB、TB、または TiB 単位で指定できます。数値と単位を設定します (例: 200GiB)。数字と単位は、分離できません。空白は、使用できます。

3. サブネット ID を指定して、ローカルゾーンにマシンプールをプロビジョニングします。

一般に利用可能および発表された AWS Local Zone の場所については、AWS の Local Zones の場所 リストを参照してください。

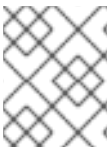
3.4. クラスターでのノードの自動スケーリングについて

自動スケーリングオプションは、クラスター内のマシンの数を自動的にスケーリングするように設定できます。

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 Pod をスケジュールできない場合、または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、指定の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。

さらに、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。

自動スケーリングを有効にする場合は、ワーカーノードの最小数および最大数も設定する必要があります。



注記

クラスターの所有者と組織管理者のみがクラスターのスケーリングまたは削除が可能です。

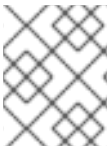
3.4.1. クラスターでの自動スケーリングノードの有効化

ワーカーノードで自動スケーリングを有効にし、既存クラスターのマシンプール定義を編集して利用可能なノード数を増減できます。

Red Hat OpenShift Cluster Manager を使用した既存のクラスターでの自動スケーリングノードの有効化
OpenShift Cluster Manager コンソールからマシンプール定義でワーカーノードの自動スケーリングを有効にします。

手順

1. OpenShift Cluster Manager で、**Clusters** ページに移動し、自動スケーリングを有効にするクラスターを選択します。
2. 選択したクラスターで、**Machine pools** タブを選択します。
3. 自動スケーリングを有効にするマシンプールの最後にある Options メニュー をクリックし、**Scale** を選択します。
4. **Edit node count** ダイアログで、**Enable autoscaling** チェックボックスを選択します。
5. **Apply** を選択してこれらの変更を保存し、クラスターの自動スケーリングを有効にします。



注記

さらに、対話モードでクラスターを作成する 場合に、デフォルトのマシンプールに自動スケーリングを設定できます。

rosa CLI を使用した既存クラスターでの自動スケーリングノードの有効化
負荷に基づいてワーカーノード数を動的にスケールアップまたはスケールダウンできるように自動スケーリングを設定します。

自動スケーリングが正常に実行されるかどうかは、AWS アカウントに正しい AWS リソースクォータがあることかどうかに依存します。AWS コンソール でリソースクォータおよび要求クォータの増加を確認します。

手順

1. クラスター内のマシンプール ID を識別するには、以下のコマンドを実行します。

```
$ rosa list machinepools --cluster=<cluster_name>
```

出力例

```
ID       AUTOSCALING   REPLICAS    INSTANCE TYPE  LABELS    TAINTS
AVAILABILITY ZONES    DISK SIZE   SG IDs
default   No           2           m5.xlarge                  us-east-1a          300GiB     sg-
0e375ff0ec4a6cfa2
mp1       No           2           m5.xlarge                  us-east-1a          300GiB     sg-
0e375ff0ec4a6cfa2
```

2. 設定する必要のあるマシンプールの ID を取得します。
3. マシンプールで自動スケーリングを有効にするには、以下のコマンドを実行します。

```
$ rosa edit machinepool --cluster=<cluster_name> <machinepool_ID> --enable-autoscaling --min-replicas=<number> --max-replicas=<number>
```

例

mp1 という ID を **mycluster** という名前のクラスターに設定し、レプリカの数が 2 から 5 ワーカーノード間でスケーリングするように設定された状態でマシンプールで自動スケーリングを有効にします。

```
$ rosa edit machinepool --cluster=mycluster mp1 --enable-autoscaling --min-replicas=2 --
max-replicas=5
```

3.4.2. クラスターでの自動スケーリングノードの無効化

ワーカーノードで自動スケーリングを無効にし、既存クラスターのマシンプール定義を編集して利用可能なノード数を増減できます。

OpenShift Cluster Manager コンソールまたは Red Hat OpenShift Service on AWS CLI を使用して、クラスターでの自動スケーリングを無効にできます。

注記

さらに、対話モードでクラスターを作成する 場合に、デフォルトのマシンプールに自動スケーリングを設定できます。

Red Hat OpenShift Cluster Manager を使用した既存のクラスターでの自動スケーリングノードの無効化
OpenShift Cluster Manager コンソールからマシンプール定義でワーカーノードの自動スケーリングを無効にします。

手順

1. OpenShift Cluster Manager で、**Clusters** ページに移動し、無効にする必要のある自動スケーリングでクラスターを選択します。
2. 選択したクラスターで、**Machine pools** タブを選択します。
3. 自動スケーリングのあるマシンプールの最後にある Options メニュー  をクリックし、**Scale** を選択します。
4. ノード数の編集ダイアログで、**Enable autoscaling** チェックボックスの選択を解除します。
5. **Apply** を選択してこれらの変更を保存し、クラスターから自動スケーリングを無効にします。

ROSA CLI を使用した既存クラスターでの自動スケーリングノードの無効化
Red Hat OpenShift Service on AWS (ROSA) CLI **rosa** を使用して、マシンプール定義内のワーカーノードの自動スケーリングを無効にします。

手順

1. 以下のコマンドを実行します。

```
$ rosa edit machinepool --cluster=<cluster_name> <machinepool_ID> --enable-
autoscaling=false --replicas=<number>
```

例

mycluster という名前のクラスターで、**default** マシンプールの自動スケーリングを無効にします。

```
$ rosa edit machinepool --cluster=mycluster default --enable-autoscaling=false --replicas=3
```

3.4.3. 関連情報

- Troubleshooting: Autoscaling is not scaling down nodes
- マシンプールについて
- コンピュートノードの管理
- ROSA CLI を使用したオブジェクトの管理

3.5. コンテナーメモリーとリスク要件を満たすためのクラスターメモリーの設定

クラスター管理者は、以下を実行し、クラスターがアプリケーションメモリーの管理を通じて効率的に動作するようにすることができます。

- コンテナー化されたアプリケーションコンポーネントのメモリーおよびリスク要件を判別し、それらの要件を満たすようコンテナーメモリーパラメーターを設定する
- コンテナー化されたアプリケーションランタイム (OpenJDK など) を、設定されたコンテナーメモリーパラメーターに基づいて最適に実行されるよう設定する
- コンテナーでの実行に関連するメモリー関連のエラー状態を診断し、これを解決する

3.5.1. アプリケーションメモリーの管理について

まず Red Hat OpenShift Service on AWS によるコンピュートリソースの管理方法の概要をよく読んでから次の手順に進むことを推奨します。

各種のリソース (メモリー、cpu、ストレージ) に応じて、Red Hat OpenShift Service on AWS ではオプションの **要求** および **制限** の値を Pod の各コンテナーに設定できます。

メモリー要求とメモリー制限について、以下の点に注意してください。

- **メモリー要求**
  - メモリー要求値が指定されている場合、Red Hat OpenShift Service on AWS スケジューラーに影響します。スケジューラーは、コンテナーのノードへのスケジュール時にメモリー要求を考慮し、コンテナーの使用のために選択されたノードで要求されたメモリーをフェンスオフします。
  - ノードのメモリーが使い切られると、Red Hat OpenShift Service on AWS はメモリー使用がメモリー要求を最も超過しているコンテナーのエビクションを優先します。深刻なメモリー枯渇の場合、ノード OOM キラーが同様のメトリックに基づいてコンテナー内のプロセスを選択して強制終了することがあります。
  - クラスター管理者は、メモリー要求値に対してクォータを割り当てるか、デフォルト値を割り当てることができます。
  - クラスター管理者は、クラスターのオーバーコミットを管理するために開発者が指定するメモリー要求の値を上書きできます。
- **メモリー制限**
  - メモリー制限値が指定されている場合、コンテナーのすべてのプロセスに割り当て可能なメモリーにハード制限を指定します。

- コンテナーのすべてのプロセスで割り当てられるメモリーがメモリー制限を超過する場合、ノードの OOM (Out of Memory) killer はコンテナーのプロセスをすぐに選択し、これを強制終了します。

- メモリー要求とメモリー制限の両方が指定される場合、メモリー制限の値はメモリー要求の値よりも大きいか、これと等しくなければなりません。

- クラスター管理者は、メモリーの制限値に対してクォータを割り当てるか、デフォルト値を割り当てることができます。

- 最小メモリー制限は 12 MB です。**Cannot allocate memory** Pod イベントのためにコンテナーの起動に失敗すると、メモリー制限は低くなります。メモリー制限を引き上げるか、これを削除します。制限を削除すると、Pod は制限のないノードのリソースを消費できるようになります。

3.5.1.1. アプリケーションメモリーストラテジーの管理

Red Hat OpenShift Service on AWS でアプリケーションメモリーをサイジングする手順は以下の通りです。

1. **予想されるコンテナーのメモリー使用の判別**
   必要時に予想される平均およびピーク時のコンテナーのメモリー使用を判別します (例: 別の負荷テストを実行)。コンテナーで並行して実行されている可能性のあるすべてのプロセスを必ず考慮に入れるようにしてください。 たとえば、メインのアプリケーションは付属スクリプトを生成しているかどうかを確認します。

2. **リスク選好 (risk appetite) の判別**
   エビクションのリスク選好を判別します。リスク選好のレベルが低い場合、コンテナーは予想されるピーク時の使用量と安全マージンのパーセンテージに応じてメモリーを要求します。リスク選好が高くなる場合、予想される平均の使用量に応じてメモリーを要求することがより適切な場合があります。

3. **コンテナーのメモリー要求の設定**
   上記に基づいてコンテナーのメモリー要求を設定します。要求がアプリケーションのメモリー使用をより正確に表示することが望ましいと言えます。要求が高すぎる場合には、クラスターおよびクォータの使用が非効率となります。要求が低すぎる場合、アプリケーションのエビクションの可能性が高くなります。

4. **コンテナーのメモリー制限の設定 (必要な場合)**
   必要時にコンテナーのメモリー制限を設定します。制限を設定すると、コンテナーのすべてのプロセスのメモリー使用量の合計が制限を超える場合にコンテナーのプロセスがすぐに強制終了されるため、いくつかの利点をもたらします。まずは予期しないメモリー使用の超過を早期に明確にする (fail fast (早く失敗する)) ことができ、次にプロセスをすぐに中止できます。

   一部の Red Hat OpenShift Service on AWS クラスターでは制限値を設定する必要があります。制限に基づいて要求を上書きする場合があります。 また、一部のアプリケーションイメージは、要求値よりも検出が簡単なことから設定される制限値に依存します。

   メモリー制限が設定される場合、これは予想されるピーク時のコンテナーのメモリー使用量と安全マージンのパーセンテージよりも低い値に設定することはできません。

5. **アプリケーションが調整されていることの確認**
   適切な場合は、設定される要求および制限値に関連してアプリケーションが調整されていることを確認します。この手順は、JVM などのメモリーをプールするアプリケーションにおいてとくに当てはまります。残りの部分では、これについて説明します。

3.5.2. Red Hat OpenShift Service on AWS の OpenJDK 設定について

デフォルトの OpenJDK 設定はコンテナー化された環境では機能しません。そのため、コンテナーで OpenJDK を実行する場合は常に追加の Java メモリー設定を指定する必要があります。

JVM のメモリーレイアウトは複雑で、バージョンに依存しており、本書ではこれについて詳細には説明しません。ただし、コンテナーで OpenJDK を実行する際のスタートにあたって少なくとも以下の 3 つのメモリー関連のタスクが主なタスクになります。

1. JVM 最大ヒープサイズを上書きする。
2. JVM が未使用メモリーをオペレーティングシステムに解放するよう促す (適切な場合)。
3. コンテナー内のすべての JVM プロセスが適切に設定されていることを確認する。

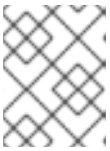
コンテナーでの実行に向けて JVM ワークロードを最適に調整する方法については本書では扱いませんが、これには複数の JVM オプションを追加で設定することが必要になる場合があります。

3.5.2.1. JVM の最大ヒープサイズを上書きする方法について

数多くの Java ワークロードにおいて、JVM ヒープはメモリーの最大かつ単一のコンシューマーです。現時点で OpenJDK は、OpenJDK がコンテナー内で実行されているかにかかわらず、ヒープに使用されるコンピュートノードのメモリーの最大 1/4 (1/**-XX:MaxRAMFraction**) を許可するようデフォルトで設定されます。そのため、コンテナーのメモリー制限も設定されている場合には、この動作をオーバーライドすることが **必須** です。

上記を実行する方法として、2 つ以上の方法を使用できます:

- コンテナーのメモリー制限が設定されており、JVM で実験的なオプションがサポートされている場合には、**-XX:+UnlockExperimentalVMOptions -XX:+UseCGroupMemoryLimitForHeap** を設定します。

  注記

  JDK 11 では **UseCGroupMemoryLimitForHeap** オプションが削除されました。**-XX:+UseContainerSupport** を代わりに使用します。

  これにより、**-XX:MaxRAM** がコンテナーのメモリー制限に設定され、最大ヒープサイズ (**-XX:MaxHeapSize** / **-Xmx**) が 1/**-XX:MaxRAMFraction** に設定されます (デフォルトでは 1/4)。

- **-XX:MaxRAM**、**-XX:MaxHeapSize** または **-Xmx** のいずれかを直接上書きします。
  このオプションには、値のハードコーディングが必要になりますが、安全マージンを計算できるという利点があります。

3.5.2.2. JVM で未使用メモリーをオペレーティングシステムに解放するよう促す方法について

デフォルトで、OpenJDK は未使用メモリーをオペレーティングシステムに積極的に返しません。これは多くのコンテナー化された Java ワークロードには適していますが、例外として、コンテナー内に JVM と共存する追加のアクティブなプロセスがあるワークロードの場合を考慮する必要があります。それらの追加のプロセスはネイティブのプロセスである場合や追加の JVM の場合、またはこれら 2 つの組み合わせである場合もあります。

Java ベースのエージェントは、次の JVM 引数を使用して、JVM が未使用のメモリーをオペレーティングシステムに解放するように促すことができます。

```
-XX:+UseParallelGC
-XX:MinHeapFreeRatio=5 -XX:MaxHeapFreeRatio=10 -XX:GCTimeRatio=4
-XX:AdaptiveSizePolicyWeight=90.
```

これらの引数は、割り当てられたメモリーが使用中のメモリー (**-XX:MaxHeapFreeRatio**) の 110% を超え、ガベージコレクター (**-XX:GCTimeRatio**) での CPU 時間の 20% を使用する場合は常にヒープメモリーをオペレーティングシステムに返すことが意図されています。アプリケーションのヒープ割り当てが初期のヒープ割り当て (**-XX:InitialHeapSize** / **-Xms** で上書きされる) を下回ることはありません。詳細情報については、Tuning Java's footprint in OpenShift (Part 1) 、Tuning Java's footprint in OpenShift (Part 2)、および OpenJDK and Containers を参照してください。

3.5.2.3. コンテナー内のすべての JVM プロセスが適切に設定されていることを確認する方法について

複数の JVM が同じコンテナーで実行される場合、それらすべてが適切に設定されていることを確認する必要があります。多くのワークロードでは、それぞれの JVM に memory budget のパーセンテージを付与する必要があります。 これにより大きな安全マージンが残される場合があります。

多くの Java ツールは JVM を設定するために各種の異なる環境変数 (**JAVA_OPTS**、**GRADLE_OPTS** など) を使用します。 適切な設定が適切な JVM に渡されていることを確認するのが容易でない場合もあります。

JAVA_TOOL_OPTIONS 環境変数は常に OpenJDK によって考慮され、**JAVA_TOOL_OPTIONS** に指定された値は、JVM コマンドラインに指定される他のオプションによって上書きされます。デフォルトでは、Java ベースのエージェントイメージで実行されるすべての JVM ワークロードに対してこれらのオプションがデフォルトで使用されるように、Red Hat OpenShift Service on AWS Jenkins Maven エージェントイメージを次のように設定します。

```
JAVA_TOOL_OPTIONS="-XX:+UnlockExperimentalVMOptions
-XX:+UseCGroupMemoryLimitForHeap -Dsun.zip.disableMemoryMapping=true"
```

注記

JDK 11 では **UseCGroupMemoryLimitForHeap** オプションが削除されました。**-XX:+UseContainerSupport** を代わりに使用します。

この設定は、追加オプションが要求されないことを保証する訳ではなく、有用な開始点になることを意図しています。

3.5.3. Pod 内でのメモリー要求および制限の検索

Pod 内からメモリー要求および制限を動的に検出するアプリケーションでは Downward API を使用する必要があります。

手順

1. **MEMORY_REQUEST** と **MEMORY_LIMIT** スタンザを追加するように Pod を設定します。
   a. 以下のような YAML ファイルを作成します。

```
apiVersion: v1
kind: Pod
metadata:
  name: test
```

```
spec:
  securityContext:
    runAsNonRoot: true
    seccompProfile:
      type: RuntimeDefault
  containers:
  - name: test
    image: fedora:latest
    command:
    - sleep
    - "3600"
    env:
    - name: MEMORY_REQUEST (1)
      valueFrom:
        resourceFieldRef:
          containerName: test
          resource: requests.memory
    - name: MEMORY_LIMIT (2)
      valueFrom:
        resourceFieldRef:
          containerName: test
          resource: limits.memory
    resources:
      requests:
        memory: 384Mi
      limits:
        memory: 512Mi
    securityContext:
      allowPrivilegeEscalation: false
      capabilities:
        drop: [ALL]
```

(1) このスタンザを追加して、アプリケーションメモリーの要求値を見つけます。

(2) このスタンザを追加して、アプリケーションメモリーの制限値を見つけます。

b. 以下のコマンドを実行して Pod を作成します。

```
$ oc create -f <file-name>.yaml
```

検証

1. リモートシェルを使用して Pod にアクセスします。

```
$ oc rsh test
```

2. 要求された値が適用されていることを確認します。

```
$ env | grep MEMORY | sort
```

出力例

```
MEMORY_LIMIT=536870912
MEMORY_REQUEST=402653184
```

注記

メモリー制限値は、**/sys/fs/cgroup/memory/memory.limit_in_bytes** ファイルによってコンテナー内から読み取ることもできます。

3.5.4. OOM の強制終了ポリシーについて

Red Hat OpenShift Service on AWS は、コンテナーのすべてのプロセスのメモリー使用量の合計がメモリー制限を超えるか、またはノードのメモリーを使い切られるなどの深刻な状態が生じる場合にコンテナーのプロセスを強制終了する場合があります。

プロセスが OOM (Out of Memory) によって強制終了される場合、コンテナーがすぐに終了する場合があります。コンテナーの PID 1 プロセスが **SIGKILL** を受信する場合、コンテナーはすぐに終了します。それ以外の場合、コンテナーの動作は他のプロセスの動作に依存します。

たとえば、コンテナーのプロセスは、SIGKILL シグナルを受信したことを示すコード 137 で終了します。

コンテナーがすぐに終了しない場合、OOM による強制終了は以下のように検出できます。

1. リモートシェルを使用して Pod にアクセスします。

```
# oc rsh test
```

2. 以下のコマンドを実行して、**/sys/fs/cgroup/memory/memory.oom_control** で現在の OOM kill カウントを表示します。

```
$ grep '^oom_kill ' /sys/fs/cgroup/memory/memory.oom_control
```

出力例

```
oom_kill 0
```

3. 以下のコマンドを実行して、Out Of Memory (OOM) による強制終了を促します。

```
$ sed -e '' </dev/zero
```

出力例

```
Killed
```

4. 以下のコマンドを実行して、**sed** コマンドの終了ステータスを表示します。

```
$ echo $?
```

出力例

```
137
```

137 コードは、コンテナーのプロセスが、SIGKILL シグナルを受信したことを示すコード 137 で終了していることを示唆します。

5. 以下のコマンドを実行して、**/sys/fs/cgroup/memory/memory.oom_control** の OOM kill カウンターの増分を表示します。

```
$ grep '^oom_kill ' /sys/fs/cgroup/memory/memory.oom_control
```

出力例

```
oom_kill 1
```

Pod の 1 つ以上のプロセスが OOM で強制終了され、Pod がこれに続いて終了する場合 (即時であるかどうかは問わない)、フェーズは **Failed**、理由は **OOMKilled** になります。OOM で強制終了された Pod は **restartPolicy** の値によって再起動する場合があります。再起動されない場合は、レプリケーションコントローラーなどのコントローラーが Pod の失敗したステータスを認識し、古い Pod に置き換わる新規 Pod を作成します。

以下のコマンドを使用して Pod のステータスを取得します。

```
$ oc get pod test
```

出力例

```
NAME      READY     STATUS      RESTARTS   AGE
test      0/1       OOMKilled   0          1m
```

- Pod が再起動されていない場合は、以下のコマンドを実行して Pod を表示します。

```
$ oc get pod test -o yaml
```

出力例

```
...
status:
  containerStatuses:
  - name: test
    ready: false
    restartCount: 0
    state:
      terminated:
        exitCode: 137
        reason: OOMKilled
  phase: Failed
```

- 再起動した場合は、以下のコマンドを実行して Pod を表示します。

```
$ oc get pod test -o yaml
```

出力例

```
...
```

```
status:
  containerStatuses:
  - name: test
    ready: true
    restartCount: 1
    lastState:
      terminated:
        exitCode: 137
        reason: OOMKilled
    state:
      running:
  phase: Running
```

3.5.5. Pod エビクションについて

Red Hat OpenShift Service on AWS は、ノードのメモリーが使い切られると、そのノードから Pod をエビクトする場合があります。メモリー消費の度合いによって、エビクションは正常に行われる場合もあれば、そうでない場合もあります。正常なエビクションは、各コンテナーのメインプロセス (PID 1) が SIGTERM シグナルを受信してから、プロセスがすでに終了していない場合は後になって SIGKILL シグナルを受信することを意味します。正常ではないエビクションは各コンテナーのメインプロセスが SIGKILL シグナルを即時に受信することを示します。

エビクトされた Pod のフェーズは **Failed** になり、理由 は **Evicted** になります。この場合、**restartPolicy** の値に関係なく再起動されません。ただし、レプリケーションコントローラーなどのコントローラーは Pod の失敗したステータスを認識し、古い Pod に置き換わる新規 Pod を作成します。

```
$ oc get pod test
```

出力例

```
NAME     READY     STATUS    RESTARTS   AGE
test     0/1       Evicted   0          1m
```

```
$ oc get pod test -o yaml
```

出力例

```
...
status:
  message: 'Pod The node was low on resource: [MemoryPressure].'
  phase: Failed
  reason: Evicted
```

第4章 PID 制限の設定

プロセス識別子 (PID) は、システム上で現在実行されている各プロセスまたはスレッドに Linux カーネルによって割り当てられる一意の識別子です。システム上で同時に実行できるプロセスの数は、Linux カーネルによって 4,194,304 に制限されています。この数値は、メモリー、CPU、ディスク容量などの他のシステムリソースへのアクセス制限によっても影響を受ける可能性があります。

Red Hat OpenShift Service on AWS 4.11 の Pod は、デフォルトで最大 4,096 個の PID を持つことができます。ワークロードでそれ以上の数が必要な場合は、**KubeletConfig** オブジェクトを設定して、許可される PID の最大数を増やせます。

重要

PID の最大数を設定することは、Hosted Control Plane (HCP) を備えた Red Hat OpenShift Service on AWS (ROSA) ではサポートされていません。

4.1. プロセス ID の制限について

Red Hat OpenShift Service on AWS では、クラスターでの作業をスケジュールする前に、プロセス ID (PID) の使用に関してサポートされている次の 2 つの制限を考慮してください。

- Pod あたりの PID の最大数。
  Red Hat OpenShift Service on AWS 4.11 以降のデフォルト値は 4,096 です。この値はノードに設定された **podPidsLimit** パラメーターによって制御されます。

- ノードあたりの PID の最大数。
  デフォルト値は ノードのリソース によって異なります。Red Hat OpenShift Service on AWS では、この値は **--system-reserved** パラメーターによって制御されます。このパラメーターにより、ノードの合計リソースに基づいて各ノードの PID が予約されます。

Pod あたりの PID の最大許容数を超えると、Pod が正しく機能しなくなり、ノードから削除される可能性があります。詳細は、エビクションシグナルとしきい値についての Kubernetes ドキュメント を参照してください。

ノードあたりの PID の最大許容数を超えると、ノードが不安定になる可能性があります。これは新しいプロセスに PID を割り当てることができなくなるためです。追加のプロセスを作成せずに既存のプロセスを完了できない場合、ノード全体が使用できなくなり、リブートが必要になる可能性があります。このような状況では、実行中のプロセスやアプリケーションによっては、データ損失が発生する可能性があります。このしきい値に達すると、お客様の管理者と Red Hat Site Reliability Engineering に通知が送信されます。また、**Worker node is experiencing PIDPressure** という警告がクラスターログに表示されます。

4.2. RED HAT OPENSHIFT SERVICE ON AWS の POD のプロセス ID 制限を引き上げることのリスク

Pod の **podPidsLimit** パラメーターは、その Pod 内で同時に実行できるプロセスとスレッドの最大数を制御するものです。

podPidsLimit の値は、デフォルトの 4,096 から最大 16,384 まで増やすことができます。**podPidsLimit** を変更するには、影響を受けるノードをリブートする必要があります。そのため、この値を変更すると、アプリケーションのダウンタイムが発生する可能性があります。

各ノードで多数の Pod を実行しており、ノードの **podPidsLimit** 値が高い場合は、ノードの PID 最大値を超えるおそれがあります。

1 つのノードでノードの PID 最大値を超えずに同時に実行できる Pod の最大数を確認するには、3,650,000 を **podPidsLimit** 値で割ります。たとえば、**podPidsLimit** 値が 16,384 で、Pod がそれに近い数のプロセス ID を使用すると予想される場合、1 つのノードで 222 個の Pod を安全に実行できます。

注記

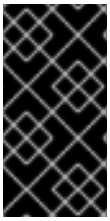
podPidsLimit 値が適切に設定されている場合でも、メモリー、CPU、および利用可能なストレージによって、同時に実行できる Pod の最大数が制限される場合があります。詳細は、「環境のプランニング」と「制限およびスケーラビリティー」を参照してください。

関連情報

- インスタンスタイプ
- 環境のプランニング
- 制限およびスケーラビリティー

4.3. 既存の RED HAT OPENSHIFT SERVICE ON AWS クラスターの PID 制限を引き上げる

--pod-pids-limit パラメーターを変更する **KubeletConfig** オブジェクトを作成または編集することで、既存の Red Hat OpenShift Service on AWS クラスターの **podPidsLimit** 設定を引き上げることができます。

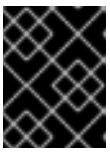


重要

既存のクラスターの **podPidsLimit** を変更すると、クラスター内のコントロールプレーン以外のノードが一度に 1 つずつリブートします。この変更はクラスターのピーク使用時間外に行い、すべてのノードがリブートするまでクラスターのアップグレードやハイバネートを実行しないでください。

前提条件

- ROSA Classic クラスターがある。



重要

PID の最大数を設定することは、Hosted Control Plane (HCP) を備えた Red Hat OpenShift Service on AWS (ROSA) ではサポートされていません。

- OpenShift CLI (**oc**) がインストールされている。
- ROSA CLI を使用して Red Hat アカウントにログインしている。

手順

1. **KubeletConfig** オブジェクトを作成または編集して、PID 制限を変更します。
   - デフォルトの PID 制限を初めて変更する場合は、次のコマンドを実行して **KubeletConfig** オブジェクトを作成し、**--pod-pids-limit** 値を設定します。
     -

```
$ rosa create kubeletconfig -c <cluster_name> --pod-pids-limit=<value>
```

たとえば、次のコマンドは、クラスター **my-cluster** の Pod あたりの PID 最大数を 16,384 に設定します。

```
$ rosa create kubeletconfig -c my-cluster --pod-pids-limit=16384
```

- 以前に **KubeletConfig** オブジェクトを作成した場合は、次のコマンドを実行して既存の **KubeletConfig** オブジェクトを編集し、**--pod-pids-limit** 値を設定します。

```
$ rosa edit kubeletconfig -c <cluster_name> --pod-pids-limit=<value>
```

クラスター全体のワーカーノードのローリングリブートが開始します。

2. 次のコマンドを実行して、すべてのワーカーノードがリブートしたことを確認します。

```
$ oc get machineconfigpool
```

出力例

```
NAME     CONFIG                 UPDATED  UPDATING   DEGRADED  MACHINECOUNT
READYMACHINECOUNT  UPDATEDMACHINECOUNT DEGRADEDMACHINECOUNT
AGE
master   rendered-master-06c9c4…  True     False      False     3          3              3
0                  4h42m
worker   rendered-worker-f4b64…   True     False      False     4          4              4
0                  4h42m
```

検証

クラスター内の各ノードがリブートしたら、新しい設定が適用されていることを確認できます。

- **KubeletConfig** オブジェクトで Pod の Pid 制限を確認します。

```
$ rosa describe kubeletconfig --cluster=<cluster_name>
```

次の例に示すように、新しい PID 制限が出力に表示されます。

出力例

```
Pod Pids Limit:                16384
```