Red Hat OpenShift Service on AWS 4

Cluster administration

Configuring Red Hat OpenShift Service on AWS clusters

Red Hat OpenShift Service on AWS 4 Cluster administration

Configuring Red Hat OpenShift Service on AWS clusters

Legal Notice

Abstract

This document provides information about configuring Red Hat OpenShift Service on AWS (ROSA) clusters.

Table of Contents

CHAPTER 1. CLUSTER NOTIFICATIONS

Cluster notifications are messages about the status, health, or performance of your cluster.

Cluster notifications are the primary way that Red Hat Site Reliability Engineering (SRE) communicates with you about the health of your managed cluster. SRE may also use cluster notifications to prompt you to perform an action in order to resolve or prevent an issue with your cluster.

Cluster owners and administrators must regularly review and action cluster notifications to ensure clusters remain healthy and supported.

You can view cluster notifications in the Red Hat Hybrid Cloud Console, in the **Cluster history** tab for your cluster. By default, only the cluster owner receives cluster notifications as emails. If other users need to receive cluster notification emails, add each user as a notification contact for your cluster.

1.1. ADDITIONAL RESOURCES

- Customer responsibilities: Review and action cluster notifications

- Cluster notification emails

- Troubleshooting: Cluster notifications

1.2. WHAT TO EXPECT FROM CLUSTER NOTIFICATIONS

As a cluster administrator, you need to be aware of when and why cluster notifications are sent, as well as their types and severity levels, in order to effectively understand the health and administration needs of your cluster.

1.2.1. Cluster notification policy

Cluster notifications are designed to keep you informed about the health of your cluster and high impact events that affect it.

Most cluster notifications are generated and sent automatically to ensure that you are immediately informed of problems or important changes to the state of your cluster.

In certain situations, Red Hat Site Reliability Engineering (SRE) creates and sends cluster notifications to provide additional context and guidance for a complex issue.

Cluster notifications are not sent for low-impact events, low-risk security updates, routine operations and maintenance, or minor, transient issues that are quickly resolved by SRE.

Red Hat services automatically send notifications when:

- Remote health monitoring or environment verification checks detect an issue in your cluster, for example, when a worker node has low disk space.

- Significant cluster life cycle events occur, for example, when scheduled maintenance or upgrades begin, or cluster operations are impacted by an event, but do not require customer intervention.

- Significant cluster management changes occur, for example, when cluster ownership or administrative control is transferred from one user to another.

- Your cluster subscription is changed or updated, for example, when Red Hat makes updates to subscription terms or features available to your cluster.

SRE creates and sends notifications when:

- An incident results in a degradation or outage that impacts your cluster's availability or performance, for example, your cloud provider has a regional outage. SRE sends subsequent notifications to inform you of incident resolution progress, and when the incident is resolved.

- A security vulnerability, security breach, or unusual activity is detected on your cluster.

- Red Hat detects that changes you have made are creating or may result in cluster instability.

- Red Hat detects that your workloads are causing performance degradation or instability in your cluster.

1.2.2. Cluster notification severity levels

Each cluster notification has an associated severity level to help you identify notifications with the greatest impact to your business. You can filter cluster notifications according to these severity levels in the Red Hat Hybrid Cloud Console, in the **Cluster history** tab for your cluster.

Red Hat uses the following severity levels for cluster notifications, from most to least severe:

Critical

Immediate action is required. One or more key functions of a service or cluster is not working, or will stop working soon. A critical alert is important enough to page on-call staff and interrupt regular workflows.

Major

Immediate action is strongly recommended. One or more key functions of the cluster will soon stop working. A major issue may lead to a critical issue if it is not addressed in a timely manner.

Warning

Action is required as soon as possible. One or more key functions of the cluster are not working optimally and may degrade further, but do not pose an immediate danger to the functioning of the cluster.

Info

No action necessary. This severity does not describe problems that need to be addressed, only important information about meaningful or important life cycle, service, or cluster events.

Debug

No action necessary. Debug notifications provide low-level information about less important lifecycle, service, or cluster events to aid in debugging unexpected behavior.

1.2.3. Cluster notification types

Each cluster notification has an associated notification type to help you identify notifications that are relevant to your role and responsibilities. You can filter cluster notifications according to these types in the Red Hat Hybrid Cloud Console, in the **Cluster history** tab for your cluster.

Red Hat uses the following notification types to indicate notification relevance.

Capacity management

Notifications for events related to updating, creating, or deleting node pools, machine pools, compute replicas or quotas (load balancer, storage, etc.).

Cluster access

Notifications for events related to adding or deleting groups, roles or identity providers, for example, when SRE cannot access your cluster because STS credentials have expired, when there is a configuration problem with your AWS roles, or when you add or remove identity providers.

Cluster add-ons

Notifications for events related to add-on management or upgrade maintenance for add-ons, for example, when an add-on is installed, upgraded, or removed, or cannot be installed due to unmet requirements.

Cluster configuration

Notifications for cluster tuning events, workload monitoring, and inflight checks.

Cluster lifecycle

Notifications for cluster or cluster resource creation, deletion, and registration, or change in cluster or resource status (for example, ready or hibernating).

Cluster networking

Notifications related to cluster networking, including HTTP/S proxy, router, and ingress state.

Cluster ownership

Notifications related to cluster ownership transfer from one user to another.

Cluster scaling

Notifications related to updating, creating, or deleting node pools, machine pools, compute replicas or quota.

Cluster security

Events related to cluster security, for example, an increased number of failed access attempts, updates to trust bundles, or software updates with security impact.

Cluster subscription

Cluster expiration, trial cluster notifications, or switching from free to paid.

Cluster updates

Anything relating to upgrades, such as upgrade maintenance or enablement.

Customer support

Updates on support case status.

General notification

The default notification type. This is only used for notifications that do not have a more specific category.

1.3. VIEWING CLUSTER NOTIFICATIONS USING THE RED HAT HYBRID CLOUD CONSOLE

Cluster notifications provide important information about the health of your cluster. You can view notifications that have been sent to your cluster in the **Cluster history** tab on the Red Hat Hybrid Cloud Console.

Prerequisites

- You are logged in to the Hybrid Cloud Console.

Procedure

1. Navigate to the Clusters page of the Hybrid Cloud Console.

2. Click the name of your cluster to go to the cluster details page.

3. Click the **Cluster history** tab.
   Cluster notifications appear under the Cluster history heading.

4. Optional: Filter for relevant cluster notifications
   Use the filter controls to hide cluster notifications that are not relevant to you, so that you can focus on your area of expertise or on resolving a critical issue. You can filter notifications based on text in the notification description, severity level, notification type, when the notification was received, and which system or person triggered the notification.

1.4. CLUSTER NOTIFICATION EMAILS

By default, when a cluster notification is sent to the cluster, it is also sent as an email to the cluster owner. You can configure additional recipients for notification emails to ensure that all appropriate users remain informed about the state of the cluster.

1.4.1. Adding notification contacts to your cluster

Notification contacts receive emails when cluster notifications are sent to the cluster. By default, only the cluster owner receives cluster notification emails. You can configure other cluster users as additional notification contacts in your cluster support settings.

Prerequisites

- Your cluster is deployed and registered to the Red Hat Hybrid Cloud Console.

- You are logged in to the Hybrid Cloud Console.

- The intended notification recipient has a user account on your cluster.

Procedure

1. Navigate to the Clusters page of the Hybrid Cloud Console.

2. Click the name of your cluster to go to the cluster details page.

3. Click the **Support** tab.

4. On the **Support** tab, find the **Notification contacts** section.

5. Click **Add notification contact**.

6. In the **Red Hat username or email** field, enter the email address or the user name of the new recipient.

7. Click **Add contact**.

Verification steps

- The "Notification contact added successfully" message displays.

1.4.2. Removing notification contacts from your cluster

Notification contacts receive emails when cluster notifications are sent to the cluster.

You can remove notification contacts in your cluster support settings to prevent them from receiving notification emails.

Prerequisites

- Your cluster is deployed and registered to the Red Hat Hybrid Cloud Console.

- You are logged in to the Hybrid Cloud Console.

Procedure

1. Navigate to the Clusters page of the Hybrid Cloud Console.

2. Click the name of your cluster to go to the cluster details page.

3. Click the **Support** tab.

4. On the **Support** tab, find the **Notification contacts** section.

5. Click the options menu (⚙) beside the recipient you want to remove.

6. Click **Delete**.

Verification steps

- The "Notification contact deleted successfully" message displays.

1.5. TROUBLESHOOTING

If you are not receiving cluster notification emails

- Ensure that emails sent from **@redhat.com** addresses are not filtered out of your email inbox.

- Ensure that your correct email address is listed as a notification contact for the cluster.

- Ask the cluster owner or administrator to add you as a notification contact: Cluster notification emails.

If your cluster does not receive notifications

- Ensure that your cluster can access resources at **api.openshift.com**.

- Ensure that your firewall is configured according to the documented prerequisites: AWS firewall prerequisites

CHAPTER 2. CONFIGURING PRIVATE CONNECTIONS

2.1. CONFIGURING PRIVATE CONNECTIONS

Private cluster access can be implemented to suit the needs of your Red Hat OpenShift Service on AWS (ROSA) environment.

Procedure

1. Access your ROSA AWS account and use one or more of the following methods to establish a private connection to your cluster:

   - Configuring AWS VPC peering: Enable VPC peering to route network traffic between two private IP addresses.

   - Configuring AWS VPN: Establish a Virtual Private Network to securely connect your private network to your Amazon Virtual Private Cloud.

   - Configuring AWS Direct Connect: Configure AWS Direct Connect to establish a dedicated network connection between your private network and an AWS Direct Connect location.

2. Configure a private cluster on ROSA.

2.2. CONFIGURING AWS VPC PEERING

This sample process configures an Amazon Web Services (AWS) VPC containing an Red Hat OpenShift Service on AWS cluster to peer with another AWS VPC network. For more information about creating an AWS VPC Peering connection or for other possible configurations, see the AWS VPC Peering guide.

2.2.1. VPC peering terms

When setting up a VPC peering connection between two VPCs on two separate AWS accounts, the following terms are used:

Red Hat OpenShift Service on AWS AWS Account	The AWS account that contains the Red Hat OpenShift Service on AWS cluster.
Red Hat OpenShift Service on AWS Cluster VPC	The VPC that contains the Red Hat OpenShift Service on AWS cluster.
Customer AWS Account	Your non-Red Hat OpenShift Service on AWS AWS Account that you would like to peer with.
Customer VPC	The VPC in your AWS Account that you would like to peer with.

Customer VPC Region	The region where the customer's VPC resides.
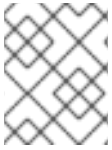
> **NOTE**
>
> As of July 2018, AWS supports inter-region VPC peering between all commercial regions excluding China.

2.2.2. Initiating the VPC peer request

You can send a VPC peering connection request from the Red Hat OpenShift Service on AWS AWS Account to the Customer AWS Account.

Prerequisites

- Gather the following information about the Customer VPC required to initiate the peering request:

  - Customer AWS account number

  - Customer VPC ID

  - Customer VPC Region

  - Customer VPC CIDR

- Check the CIDR block used by the Red Hat OpenShift Service on AWS Cluster VPC. If it overlaps or matches the CIDR block for the Customer VPC, then peering between these two VPCs is not possible; see the Amazon VPC Unsupported VPC Peering Configurations documentation for details. If the CIDR blocks do not overlap, you can continue with the procedure.

Procedure

1. Log in to the Web Console for the Red Hat OpenShift Service on AWS AWS Account and navigate to the **VPC Dashboard** in the region where the cluster is being hosted.

2. Go to the **Peering Connections** page and click the **Create Peering Connection** button.

3. Verify the details of the account you are logged in to and the details of the account and VPC you are connecting to:

   a. **Peering connection name tag** Set a descriptive name for the VPC Peering Connection.

   b. **VPC (Requester)**: Select the Red Hat OpenShift Service on AWS Cluster VPC ID from the dropdown *list.

   c. **Account**: Select **Another account** and provide the Customer AWS Account number * (without dashes).

   d. **Region**: If the Customer VPC Region differs from the current region, select **Another Region** and select the customer VPC Region from the dropdown list.

   e. **VPC (Accepter)**: Set the Customer VPC ID.

4. Click **Create Peering Connection**.

5. Confirm that the request enters a **Pending** state. If it enters a **Failed** state, confirm the details and repeat the process.

2.2.3. Accepting the VPC peer request

After you create the VPC peering connection, you must accept the request in the Customer AWS Account.

Prerequisites

- Initiate the VPC peer request.

Procedure

1. Log in to the AWS Web Console.

2. Navigate to **VPC Service**.

3. Go to **Peering Connections**.

4. Click on **Pending peering connection**.

5. Confirm the AWS Account and VPC ID that the request originated from. This should be from the Red Hat OpenShift Service on AWS AWS Account and Red Hat OpenShift Service on AWS Cluster VPC.

6. Click **Accept Request**.

2.2.4. Configuring the routing tables

After you accept the VPC peering request, both VPCs must configure their routes to communicate across the peering connection.

Prerequisites

- Initiate and accept the VPC peer request.

Procedure

1. Log in to the AWS Web Console for the Red Hat OpenShift Service on AWS AWS Account.

2. Navigate to the **VPC Service**, then **Route Tables**.

3. Select the Route Table for the Red Hat OpenShift Service on AWS Cluster VPC.
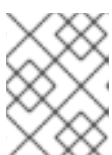
   > **NOTE**
   >
   > On some clusters, there may be more than one route table for a particular VPC. Select the private one that has a number of explicitly associated subnets.

4. Select the **Routes** tab, then **Edit**.

5. Enter the Customer VPC CIDR block in the **Destination** text box.

6. Enter the Peering Connection ID in the **Target** text box.

7. Click **Save**.

8. You must complete the same process with the other VPC's CIDR block:

    a. Log into the Customer AWS Web Console → **VPC Service** → **Route Tables**.

    b. Select the Route Table for your VPC.

    c. Select the **Routes** tab, then **Edit**.

    d. Enter the Red Hat OpenShift Service on AWS Cluster VPC CIDR block in the **Destination** text box.

    e. Enter the Peering Connection ID in the **Target** text box.

    f. Click **Save**.

The VPC peering connection is now complete. Follow the verification procedure to ensure connectivity across the peering connection is working.

2.2.5. Verifying and troubleshooting VPC peering

After you set up a VPC peering connection, it is best to confirm it has been configured and is working correctly.

Prerequisites

- Initiate and accept the VPC peer request.

- Configure the routing tables.

Procedure

- In the AWS console, look at the route table for the cluster VPC that is peered. Ensure that the steps for configuring the routing tables were followed and that there is a route table entry pointing the VPC CIDR range destination to the peering connection target.
  If the routes look correct on both the Red Hat OpenShift Service on AWS Cluster VPC route table and Customer VPC route table, then the connection should be tested using the **netcat** method below. If the test calls are successful, then VPC peering is working correctly.

- To test network connectivity to an endpoint device, **nc** (or **netcat**) is a helpful troubleshooting tool. It is included in the default image and provides quick and clear output if a connection can be established:

    a. Create a temporary pod using the **busybox** image, which cleans up after itself:

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

    b. Check the connection using **nc**.

- Example successful connection results:

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

- Example failed connection results:

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

c. Exit the container, which automatically deletes the Pod:

```
/ exit
```

2.3. CONFIGURING AWS VPN

This sample process configures an Amazon Web Services (AWS) Red Hat OpenShift Service on AWS cluster to use a customer's on-site hardware VPN device.

> **NOTE**
>
> AWS VPN does not currently provide a managed option to apply NAT to VPN traffic. See the AWS Knowledge Center for more details.

> **NOTE**
>
> Routing all traffic, for example **0.0.0.0/0**, through a private connection is not supported. This requires deleting the internet gateway, which disables SRE management traffic.

For more information about connecting an AWS VPC to remote networks using a hardware VPN device, see the Amazon VPC VPN Connections documentation.

2.3.1. Creating a VPN connection

You can configure an Amazon Web Services (AWS) Red Hat OpenShift Service on AWS cluster to use a customer's on-site hardware VPN device using the following procedures.

Prerequisites

- Hardware VPN gateway device model and software version, for example Cisco ASA running version 8.3. See the Amazon VPC Network Administrator Guide to confirm whether your gateway device is supported by AWS.

- Public, static IP address for the VPN gateway device.

- BGP or static routing: if BGP, the ASN is required. If static routing, you must configure at least one static route.

- Optional: IP and Port/Protocol of a reachable service to test the VPN connection.

2.3.1.1. Configuring the VPN connection

Procedure

1. Log in to the Red Hat OpenShift Service on AWS AWS Account Dashboard, and navigate to the VPC Dashboard.

2. Click on **Your VPCs** and identify the name and VPC ID for the VPC containing the Red Hat OpenShift Service on AWS cluster.

3. From the VPC Dashboard, click **Customer Gateway**.

4. Click **Create Customer Gateway** and give it a meaningful name.

5. Select the routing method: **Dynamic** or **Static**.

6. If Dynamic, enter the BGP ASN in the field that appears.

7. Paste in the VPN gateway endpoint IP address.

8. Click **Create**.

9. If you do not already have a Virtual Private Gateway attached to the intended VPC:

   a. From the VPC Dashboard, click on **Virtual Private Gateway**.

   b. Click **Create Virtual Private Gateway**, give it a meaningful name, and click **Create**.

   c. Leave the default Amazon default ASN.

   d. Select the newly created gateway, click **Attach to VPC**, and attach it to the cluster VPC you identified earlier.

2.3.1.2. Establishing the VPN Connection

Procedure

1. From the VPC dashboard, click on **Site-to-Site VPN Connections**.

2. Click **Create VPN Connection**.

   a. Give it a meaningful name tag.

   b. Select the virtual private gateway created previously.

   c. For Customer Gateway, select **Existing**.

   d. Select the customer gateway device by name.

   e. If the VPN will use BGP, select **Dynamic**, otherwise select **Static**. Enter Static IP CIDRs. If there are multiple CIDRs, add each CIDR as **Another Rule**.

   f. Click **Create**.

   g. Wait for VPN status to change to **Available**, approximately 5 to 10 minutes.

3. Select the VPN you just created and click **Download Configuration**.

a. From the dropdown list, select the vendor, platform, and version of the customer gateway device, then click **Download**.

b. The **Generic** vendor configuration is also available for retrieving information in a plain text format.
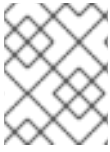
> **NOTE**
>
> After the VPN connection has been established, be sure to set up Route Propagation or the VPN may not function as expected.

> **NOTE**
>
> Note the VPC subnet information, which you must add to your configuration as the remote network.

2.3.1.3. Enabling VPN route propagation

After you have set up the VPN connection, you must ensure that route propagation is enabled so that the necessary routes are added to the VPC's route table.

Procedure

1. From the VPC Dashboard, click on **Route Tables**.

2. Select the private Route table associated with the VPC that contains your Red Hat OpenShift Service on AWS cluster.

   > **NOTE**
   >
   > On some clusters, there may be more than one route table for a particular VPC. Select the private one that has a number of explicitly associated subnets.

3. Click on the **Route Propagation** tab.

4. In the table that appears, you should see the virtual private gateway you created previously. Check the value in the **Propagate column**.

   a. If Propagate is set to **No**, click **Edit route propagation**, check the Propagate checkbox next to the virtual private gateway's name and click **Save**.

After you configure your VPN tunnel and AWS detects it as **Up**, your static or BGP routes are automatically added to the route table.

2.3.2. Verifying the VPN connection

After you have set up your side of the VPN tunnel, you can verify that the tunnel is up in the AWS console and that connectivity across the tunnel is working.

Prerequisites

- Created a VPN connection.

Procedure

1. **Verify the tunnel is up in AWS.**

   a. From the VPC Dashboard, click on **VPN Connections**.

   b. Select the VPN connection you created previously and click the **Tunnel Details** tab.

   c. You should be able to see that at least one of the VPN tunnels is **Up**.

2. **Verify the connection.**
   To test network connectivity to an endpoint device, **nc** (or **netcat**) is a helpful troubleshooting tool. It is included in the default image and provides quick and clear output if a connection can be established:

   a. Create a temporary pod using the **busybox** image, which cleans up after itself:

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

   b. Check the connection using **nc**.

      - Example successful connection results:

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

      - Example failed connection results:

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

   c. Exit the container, which automatically deletes the Pod:

   ```
   / exit
   ```

2.3.3. Troubleshooting the VPN connection

Tunnel does not connect
If the tunnel connection is still **Down**, there are several things you can verify:

- The AWS tunnel will not initiate a VPN connection. The connection attempt must be initiated from the Customer Gateway.

- Ensure that your source traffic is coming from the same IP as the configured customer gateway. AWS will silently drop all traffic to the gateway whose source IP address does not match.

- Ensure that your configuration matches values supported by AWS. This includes IKE versions, DH groups, IKE lifetime, and more.

- Recheck the route table for the VPC. Ensure that propagation is enabled and that there are entries in the route table that have the virtual private gateway you created earlier as a target.

- Confirm that you do not have any firewall rules that could be causing an interruption.

- Check if you are using a policy-based VPN as this can cause complications depending on how it is configured.

- Further troubleshooting steps can be found at the AWS Knowledge Center.

Tunnel does not stay connected
If the tunnel connection has trouble staying **Up** consistently, know that all AWS tunnel connections must be initiated from your gateway. AWS tunnels do not initiate tunneling .

Red Hat recommends setting up an SLA Monitor (Cisco ASA) or some device on your side of the tunnel that constantly sends "interesting" traffic, for example **ping**, **nc**, or **telnet**, at any IP address configured within the VPC CIDR range. It does not matter whether the connection is successful, just that the traffic is being directed at the tunnel.

Secondary tunnel in Down state
When a VPN tunnel is created, AWS creates an additional failover tunnel. Depending upon the gateway device, sometimes the secondary tunnel will be seen as in the **Down** state.

The AWS Notification is as follows:

> 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.

2.4. CONFIGURING AWS DIRECT CONNECT

This process describes accepting an AWS Direct Connect virtual interface with Red Hat OpenShift Service on AWS. For more information about AWS Direct Connect types and configuration, see the AWS Direct Connect components documentation.

2.4.1. AWS Direct Connect methods

A Direct Connect connection requires a hosted Virtual Interface (VIF) connected to a Direct Connect Gateway (DXGateway), which is in turn associated to a Virtual Gateway (VGW) or a Transit Gateway in order to access a remote VPC in the same or another account.

If you do not have an existing DXGateway, the typical process involves creating the hosted VIF, with the DXGateway and VGW being created in the Red Hat OpenShift Service on AWS AWS Account.

If you have an existing DXGateway connected to one or more existing VGWs, the process involves the Red Hat OpenShift Service on AWS AWS Account sending an Association Proposal to the DXGateway owner. The DXGateway owner must ensure that the proposed CIDR will not conflict with any other VGWs they have associated.

See the following AWS documentation for more details:

- Virtual Interfaces

- Direct Connect Gateways

- Associating a VGW across accounts
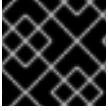
> **IMPORTANT**
>
> When connecting to an existing DXGateway, you are responsible for the costs.

There are two configuration options available:

Method 1	Create the hosted VIF and then the DXGateway and VGW.
Method 2	Request a connection via an existing Direct Connect Gateway that you own.

2.4.2. Creating the hosted Virtual Interface

Prerequisites

- Gather Red Hat OpenShift Service on AWS AWS Account ID.

2.4.2.1. Determining the type of Direct Connect connection

View the Direct Connect Virtual Interface details to determine the type of connection.

Procedure

1. Log in to the Red Hat OpenShift Service on AWS AWS Account Dashboard and select the correct region.

2. Select **Direct Connect** from the **Services** menu.

3. There will be one or more Virtual Interfaces waiting to be accepted, select one of them to view the **Summary**.

4. View the Virtual Interface type: private or public.

5. Record the **Amazon side ASN** value.

If the Direct Connect Virtual Interface type is Private, a Virtual Private Gateway is created. If the Direct Connect Virtual Interface is Public, a Direct Connect Gateway is created.

2.4.2.2. Creating a Private Direct Connect

A Private Direct Connect is created if the Direct Connect Virtual Interface type is Private.

Procedure

1. Log in to the Red Hat OpenShift Service on AWS AWS Account Dashboard and select the correct region.

2. From the AWS region, select **VPC** from the **Services** menu.

3. Select **Virtual Private Gateways** from **VPN Connections**.

4. Click **Create Virtual Private Gateway**.

5. Give the Virtual Private Gateway a suitable name.

6. Select **Custom ASN** and enter the **Amazon side ASN** value gathered previously.

7. Create the Virtual Private Gateway.

8. Click the newly created Virtual Private Gateway and choose **Attach to VPC** from the **Actions** tab.

9. Select the **Red Hat OpenShift Service on AWS Cluster VPC** from the list, and attach the Virtual Private Gateway to the VPC.

10. From the **Services** menu, click **Direct Connect**. Choose one of the Direct Connect Virtual Interfaces from the list.

11. Acknowledge the **I understand that Direct Connect port charges apply once I click Accept Connection** message, then choose **Accept Connection**.

12. Choose to **Accept** the Virtual Private Gateway Connection and select the Virtual Private Gateway that was created in the previous steps.

13. Select **Accept** to accept the connection.

14. Repeat the previous steps if there is more than one Virtual Interface.

2.4.2.3. Creating a Public Direct Connect

A Public Direct Connect is created if the Direct Connect Virtual Interface type is Public.

Procedure

1. Log in to the Red Hat OpenShift Service on AWS AWS Account Dashboard and select the correct region.

2. From the Red Hat OpenShift Service on AWS AWS Account region, select **Direct Connect** from the **Services** menu.

3. Select **Direct Connect Gateways** and **Create Direct Connect Gateway**.

4. Give the Direct Connect Gateway a suitable name.

5. In the **Amazon side ASN**, enter the Amazon side ASN value gathered previously.

6. Create the Direct Connect Gateway.

7. Select **Direct Connect** from the **Services** menu.

8. Select one of the Direct Connect Virtual Interfaces from the list.

9. Acknowledge the **I understand that Direct Connect port charges apply once I click Accept Connection** message, then choose **Accept Connection**.

10. Choose to **Accept** the Direct Connect Gateway Connection and select the Direct Connect Gateway that was created in the previous steps.

11. Click **Accept** to accept the connection.

12. Repeat the previous steps if there is more than one Virtual Interface.

2.4.2.4. Verifying the Virtual Interfaces

After the Direct Connect Virtual Interfaces have been accepted, wait a short period and view the status of the Interfaces.

Procedure

1. Log in to the Red Hat OpenShift Service on AWS AWS Account Dashboard and select the correct region.

2. From the Red Hat OpenShift Service on AWS AWS Account region, select **Direct Connect** from the **Services** menu.

3. Select one of the Direct Connect Virtual Interfaces from the list.

4. Check the Interface State has become **Available**

5. Check the Interface BGP Status has become **Up**.

6. Repeat this verification for any remaining Direct Connect Interfaces.

After the Direct Connect Virtual Interfaces are available, you can log in to the Red Hat OpenShift Service on AWS AWS Account Dashboard and download the Direct Connect configuration file for configuration on your side.

2.4.3. Connecting to an existing Direct Connect Gateway

Prerequisites

- Confirm the CIDR range of the Red Hat OpenShift Service on AWS VPC will not conflict with any other VGWs you have associated.

- Gather the following information:

  - The Direct Connect Gateway ID.

  - The AWS Account ID associated with the virtual interface.

  - The BGP ASN assigned for the DXGateway. Optional: the Amazon default ASN may also be used.

Procedure

1. Log in to the Red Hat OpenShift Service on AWS AWS Account Dashboard and select the correct region.

2. From the Red Hat OpenShift Service on AWS AWS Account region, select **VPC** from the **Services** menu.

3. From **VPN Connections**, select **Virtual Private Gateways**.

4. Select **Create Virtual Private Gateway**.

5. Give the Virtual Private Gateway a suitable name.

6. Click **Custom ASN** and enter the **Amazon side ASN** value gathered previously or use the Amazon Provided ASN.

7. Create the Virtual Private Gateway.

8. In the **Navigation** pane of the Red Hat OpenShift Service on AWS AWS Account Dashboard, choose **Virtual private gateways** and select the virtual private gateway. Choose **View details**.

9. Choose **Direct Connect gateway associations** and click **Associate Direct Connect gateway**.

10. Under **Association account type**, for Account owner, choose **Another account**.

11. For **Direct Connect gateway owner**, enter the ID of the AWS account that owns the Direct Connect gateway.

12. Under **Association settings**, for Direct Connect gateway ID, enter the ID of the Direct Connect gateway.

13. Under **Association settings**, for Virtual interface owner, enter the ID of the AWS account that owns the virtual interface for the association.

14. Optional: Add prefixes to Allowed prefixes, separating them using commas.

15. Choose **Associate Direct Connect gateway**.

16. After the Association Proposal has been sent, it will be waiting for your acceptance. The final steps you must perform are available in the AWS Documentation.

2.4.4. Troubleshooting Direct Connect

Further troubleshooting can be found in the Troubleshooting AWS Direct Connect documentation.

CHAPTER 3. CLUSTER AUTOSCALING

Applying autoscaling to Red Hat OpenShift Service on AWS clusters involves configuring a cluster autoscaler and then configuring a machine autoscaler for at least one machine pool in your cluster.

> **IMPORTANT**
>
> You can configure the cluster autoscaler only in clusters where the machine API is operational.
>
> Only one cluster autoscaler can be created per cluster.

3.1. ABOUT THE CLUSTER AUTOSCALER

The cluster autoscaler adjusts the size of an Red Hat OpenShift Service on AWS cluster to meet its current deployment needs. It uses declarative, Kubernetes-style arguments to provide infrastructure management that does not rely on objects of a specific cloud provider. The cluster autoscaler has a cluster scope, and is not associated with a particular namespace.

The cluster autoscaler increases the size of the cluster when there are pods that fail to schedule on any of the current worker nodes due to insufficient resources or when another node is necessary to meet deployment needs. The cluster autoscaler does not increase the cluster resources beyond the limits that you specify.

The cluster autoscaler computes the total memory and CPU on all nodes the cluster, even though it does not manage the control plane nodes. These values are not single-machine oriented. They are an aggregation of all the resources in the entire cluster. For example, if you set the maximum memory resource limit, the cluster autoscaler includes all the nodes in the cluster when calculating the current memory usage. That calculation is then used to determine if the cluster autoscaler has the capacity to add more worker resources.
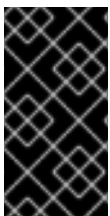
> **IMPORTANT**
>
> Ensure that the **maxNodesTotal** value in the **ClusterAutoscaler** resource definition that you create is large enough to account for the total possible number of machines in your cluster. This value must encompass the number of control plane machines and the possible number of compute machines that you might scale to.

Every 10 seconds, the cluster autoscaler checks which nodes are unnecessary in the cluster and removes them. The cluster autoscaler considers a node for removal if the following conditions apply:

- The node utilization is less than the *node utilization level* threshold for the cluster. The node utilization level is the sum of the requested resources divided by the allocated resources for the node. If you do not specify a value in the **ClusterAutoscaler** custom resource, the cluster autoscaler uses a default value of **0.5**, which corresponds to 50% utilization.

- The cluster autoscaler can move all pods running on the node to the other nodes. The Kubernetes scheduler is responsible for scheduling pods on the nodes.

- The cluster autoscaler does not have scale down disabled annotation.

If the following types of pods are present on a node, the cluster autoscaler will not remove the node:

- Pods with restrictive pod disruption budgets (PDBs).

- Kube-system pods that do not run on the node by default.

- Kube-system pods that do not have a PDB or have a PDB that is too restrictive.

- Pods that are not backed by a controller object such as a deployment, replica set, or stateful set.

- Pods with local storage.

- Pods that cannot be moved elsewhere because of a lack of resources, incompatible node selectors or affinity, matching anti-affinity, and so on.

- Unless they also have a **"cluster-autoscaler.kubernetes.io/safe-to-evict": "true"** annotation, pods that have a **"cluster-autoscaler.kubernetes.io/safe-to-evict": "false"** annotation.

For example, you set the maximum CPU limit to 64 cores and configure the cluster autoscaler to only create machines that have 8 cores each. If your cluster starts with 30 cores, the cluster autoscaler can add up to 4 more nodes with 32 cores, for a total of 62.

If you configure the cluster autoscaler, additional usage restrictions apply:

- Do not modify the nodes that are in autoscaled node groups directly. All nodes within the same node group have the same capacity and labels and run the same system pods.

- Specify requests for your pods.

- If you have to prevent pods from being deleted too quickly, configure appropriate PDBs.

- Confirm that your cloud provider quota is large enough to support the maximum node pools that you configure.

- Do not run additional node group autoscalers, especially the ones offered by your cloud provider.

The horizontal pod autoscaler (HPA) and the cluster autoscaler modify cluster resources in different ways. The HPA changes the deployment's or replica set's number of replicas based on the current CPU load. If the load increases, the HPA creates new replicas, regardless of the amount of resources available to the cluster. If there are not enough resources, the cluster autoscaler adds resources so that the HPA-created pods can run. If the load decreases, the HPA stops some replicas. If this action causes some nodes to be underutilized or completely empty, the cluster autoscaler deletes the unnecessary nodes.

The cluster autoscaler takes pod priorities into account. The Pod Priority and Preemption feature enables scheduling pods based on priorities if the cluster does not have enough resources, but the cluster autoscaler ensures that the cluster has resources to run all pods. To honor the intention of both features, the cluster autoscaler includes a priority cutoff function. You can use this cutoff to schedule "best-effort" pods, which do not cause the cluster autoscaler to increase resources but instead run only when spare resources are available.

Pods with priority lower than the cutoff value do not cause the cluster to scale up or prevent the cluster from scaling down. No new nodes are added to run the pods, and nodes running these pods might be deleted to free resources.

Cluster autoscaling is supported for the platforms that have machine API available on it.

3.2. ENABLE AUTOSCALING DURING CLUSTER CREATION WITH OPENSHIFT CLUSTER MANAGER

You can use OpenShift Cluster Manager to autoscale during cluster creation.

Procedure

1. During cluster creation, check the **Enable autoscaling** box. The **Edit cluster autoscaling settings** button becomes selectable.

   a. You can also choose the minimum or maximum amount of nodes to autoscale.

2. Click **Edit cluster autoscaling settings**

3. Edit any settings you want and then click **Close**.

3.3. ENABLE AUTOSCALING AFTER CLUSTER CREATION WITH OPENSHIFT CLUSTER MANAGER

You can use OpenShift Cluster Manager to autoscale after cluster creation.

Procedure

1. In OpenShift Cluster Manager, click the name of the cluster you want to autoscale. The Overview page for the cluster has a **Autoscaling** item that indicates if it is enabled or disabled.

2. Click the **Machine Pools** tab.

3. Click the **Edit cluster autoscaling** button. The **Edit cluster autoscaling** settings window is shown.

4. Click the **Autoscale cluster** toggle at the top of the window. All the settings are now editable.

5. Edit any settings you want and then click **Save**.

6. Click the **x** at the top right of the screen to close the settings window.

To revert all autoscaling settings to the defaults if they have been changed, click the **Revert all to defaults** button.

3.4. CLUSTER AUTOSCALING SETTINGS USING OPENSHIFT CLUSTER MANAGER

The tables explain all the configurable UI settings when using cluster autoscaling with OpenShift Cluster Manager.

3.4.1. General settings

Table 3.1. Configurable general settings for cluster autoscaling when using the OpenShift Cluster Manager

Setting	Description	Type or Range	Default

Setting	Description	Type or Range	Default
log-verbosity	Sets the autoscaler log level. The default value is 1. Level 4 is recommended for debugging. Level 6 enables almost everything.	**integer**	1
skip-nodes-with-local-storage	If **true**, the cluster autoscaler never deletes nodes with pods with local storage, e.g. EmptyDir or HostPath.	**boolean**	true
max-pod-grace-period	Gives pods graceful termination time in seconds before scaling down.	**integer**	600
max-node-provision-time	Maximum time the cluster autoscaler waits for nodes to be provisioned.	**string**	15m
pod-priority-threshold	Allows users to schedule "best-effort" pods, which are not expected to trigger cluster autoscaler actions. These pods only run when spare resources are available.	**integer**	-10
ignore-daemonsets-utilization	Determines whether the cluster autoscaler ignores daemon set pods when calculating resource utilization for scaling down.	**boolean**	false
balance-similar-node-groups	If **true**, this setting automatically identifies node groups with the same instance type and the same set of labels and tries to keep the respective sizes of those node groups balanced.	**boolean**	false

Setting	Description	Type or Range	Default
balancing-ignored-labels	This option specifies labels that the cluster autoscaler should ignore when considering node group similarity. This option cannot contain spaces.	**array (string)**	Format should be a comma-separated list of labels.

3.4.2. Resource limits

Table 3.2. Configurable resource limit settings for cluster autoscaling when using the OpenShift Cluster Manager

Setting	Description	Type or Range	Default
cores-total-min	Minimum number of cores in cluster. The cluster autoscaler does not scale the cluster less than this number.	**object**	0
cores-total-max	Maximum number of cores in cluster. The cluster autoscaler does not scale the cluster greater than this number.	**object**	180 * 64 (11520)
memory-total-min	Minimum number of gigabytes of memory in cluster. The cluster autoscaler does not scale the cluster less than this number.	**object**	0
memory-total-max	Maximum number of gigabytes of memory in cluster. The cluster autoscaler does not scale the cluster greater than this number.	**object**	180 * 64 * 20 (230400)

Setting	Description	Type or Range	Default
max-nodes-total	Maximum number of nodes in all node groups. Includes all nodes, not just automatically scaled nodes. The cluster autoscaler does not grow the cluster greater than this number.	**integer**	180
GPUs	Minimum and maximum number of different GPUs in cluster. The cluster autoscaler does not scale the cluster less than or greater than these numbers.	**array**	Format should be a comma-separated list of "<gpu_type>:<min>:<max>".

3.4.3. Scale down configuration

Table 3.3. Configurable scale down settings for cluster autoscaling when using the OpenShift Cluster Manager

Setting	Description	Type or Range	Default
scale-down-enabled	Should the cluster autoscaler scale down the cluster.	**boolean**	true
scale-down-utilization-threshold	Node utilization level, defined as the sum of the requested resources divided by capacity, below which a node can be considered for scale down.	**float**	0.5
scale-down-unneeded-time	How long a node should be unneeded before it is eligible for scale down.	**string**	10m
scale-down-delay-after-add	How long after scale up that scale-down evaluation resumes.	**string**	10m
scale-down-delay-after-delete	How long after node deletion that scale-down evaluation resumes.	**string**	0s

Setting	Description	Type or Range	Default
scale-down-delay-after-failure	How long after scale down failure that scale-down evaluation resumes.	**string**	3m

3.5. ENABLE AUTOSCALING DURING CLUSTER CREATION BY USING THE INTERACTIVE MODE WITH THE ROSA CLI

You can use the interactive mode of your terminal, if available, to set cluster-wide autoscaling behavior during cluster creation.

Interactive mode provides more information about available configurable parameters. Interactive mode also does basic checks and preflight validations, meaning that if a provided value is invalid, the terminal outputs a prompt for a valid input.

Procedure

- During cluster creation, use the **--enable-autoscaling** and **--interactive** parameters to enable cluster autoscaling:

  Example:

  ```
  $ rosa create cluster --cluster-name <cluster_name> --enable-autoscaling --interactive
  ```
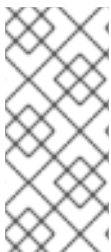
> **NOTE**
>
> If your cluster name is longer than 15 characters, it will contain an autogenerated domain prefix as a sub-domain for your provisioned cluster on ***.openshiftapps.com**.
>
> To customize the subdomain, use the **--domain-prefix** flag. The domain prefix cannot be longer than 15 characters, must be unique, and cannot be changed after cluster creation.

When the following prompt appears, enter **y** to go through all available autoscaling options.

Example interactive prompt:

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

3.5.1. Enable autoscaling after cluster creation by using the interactive mode with the ROSA CLI

You can use the interactive mode of your terminal, if available, to set cluster-wide autoscaling behavior after cluster creation.

Procedure

- After you have created a cluster, type the following command:

Example:

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

You can then set all available autoscaling parameters.

3.6. ENABLE AUTOSCALING DURING CLUSTER CREATION WITH THE ROSA CLI

You can use the ROSA CLI (**rosa**) to set cluster-wide autoscaling behavior during cluster creation. You can enable the autoscaler on the entire machine or just a cluster.

Procedure

- During cluster creation, type **--enable autoscaling** after the cluster name to enable machine autoscaling:

> **NOTE**
>
> If your cluster name is longer than 15 characters, it will contain an autogenerated domain prefix as a sub-domain for your provisioned cluster on ***.openshiftapps.com**.
>
> To customize the subdomain, use the **--domain-prefix** flag. The domain prefix cannot be longer than 15 characters, must be unique, and cannot be changed after cluster creation.

Example:

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

Set at least one parameter to enable cluster autoscaling by running the following command:

Example:

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

3.6.1. Enable autoscaling after cluster creation with the ROSA CLI

You can use the ROSA CLI (**rosa**) to set cluster-wide autoscaling after cluster creation.

Procedure

- After you have created a cluster, create the autoscaler:

  Example:

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

  a. You can also create the autoscaler with specific parameters using the following command:

     Example:

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

3.6.2. Edit autoscaling after cluster creation with the ROSA CLI

You can edit any specific parameters of the cluster autoscaler after creating the autoscaler.

- To edit the cluster autoscaler, run the following command:

  Example:

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

  a. To edit a specific parameter, run the following command:

     Example:

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

3.6.3. Delete autoscaling using the ROSA CLI

You can delete the cluster autoscaler if you no longer want to use it.

- To delete the cluster autoscaler, run the following command:

  Example:

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

3.7. CLUSTER AUTOSCALING PARAMETERS USING THE ROSA CLI

You can add the following parameters to the cluster creation command to configure autoscaler parameters when using the ROSA CLI (**rosa**).

Table 3.4. Configurable autoscaler parameters available with the ROSA CLI (**rosa**)

Setting	Description	Type or Range	Example/Instruction
--autoscaler-balance-similar-node-groups	Identify node groups with the same instance type and label set, and try to balance respective sizes of those node groups.	**boolean**	Add it to set to true, omit the option to set to false.
--autoscaler-skip-nodes-with-local-storage	If set, the cluster autoscaler does not delete nodes with pods that have local storage, for example, EmptyDir or HostPath.	**boolean**	Add it to set to true, omit the option to set to false.

Setting	Description	Type or Range	Example/Instruction
--autoscaler-log-verbosity *int*	Autoscaler log level. Replace *int* in the command with the number you want to use.	**integer**	**--autoscaler-log-verbosity 4**
--autoscaler-max-pod-grace-period *int*	Gives pods graceful termination time before scaling down, measured in seconds. Replace *int* in the command with the number of seconds you want to use.	**integer**	**--autoscaler-max-pod-grace-period 0**
--autoscaler-pod-priority-threshold *int*	The priority that a pod must exceed to cause the cluster autoscaler to deploy additional nodes. Replace *int* in the command with the number you want to use, can be negative.	**integer**	**--autoscaler-pod-priority-threshold -10**
--autoscaler-gpu-limit *stringArray*	Minimum and maximum number of different GPUs in cluster. Cluster autoscaler does not scale the cluster less than or greater than these numbers. The format must be a comma-separated list of "<gpu_type>,<min>, <max>".	**array**	**--autoscaler-gpu-limit nvidia.com/gpu,0,10 --autoscaler-gpu-limit amd.com/gpu,1,5**
--autoscaler-ignore-daemonsets-utilization	If set, the cluster-autoscaler ignores daemon set pods when calculating resource utilization for scaling down.	**boolean**	Add it to set to true, omit the option to set to false.
--autoscaler-max-node-provision-time *string*	Maximum time that the cluster autoscaler waits for a node to be provisioned. Replace *string* in the command with an integer and time unit (ns,us,μs,ms,s,m,h).	**string**	**--autoscaler-max-node-provision-time 35m**

Setting	Description	Type or Range	Example/Instruction
--autoscaler-balancing-ignored-labels *strings*	A comma-separated list of label keys that the cluster autoscaler should ignore when comparing node groups for similarity. Replace *strings* in the command with the relevant labels..	**string**	**--autoscaler-balancing-ignored-labels topology.ebs.csi.aws.com/zone,alpha.eksctl.io/instance-id**
--autoscaler-max-nodes-total *int*	Maximum amount of nodes in the cluster, including the autoscaled nodes. Replace *int* in the command with the number you want to use.	**integer**	**--autoscaler-max-nodes-total 180**
--autoscaler-min-cores *int*	Minimum number of cores to deploy in the cluster. Replace *int* in the command with the number you want to use.	**integer**	**--autoscaler-min-cores 0**
--autoscaler-max-cores *int*	Maximum number of cores to deploy in the cluster. Replace *int* in the command with the number you want to use.	**integer**	**--autoscaler-max-cores 100**
--autoscaler-min-memory *int*	Minimum amount of memory, in GiB, in the cluster. Replace *int* in the command with the number you want to use.	**integer**	**--autoscaler-min-memory 0**
--autoscaler-max-memory *int*	Maximum amount of memory, in GiB, in the cluster. Replace *int* in the command with the number you want to use.	**integer**	**--autoscaler-max-memory 4096**
--autoscaler-scale-down-enabled	If set, the cluster autoscaler should scale down the cluster.	**boolean**	Add it to set to true, omit the option to set to false.

Setting	Description	Type or Range	Example/Instruction
--autoscaler-scale-down-unneeded-time *string*	How long a node should be unneeded before it is eligible for scale down. Replace *string* in the command with an integer and time unit (ns,us,μs,ms,s,m,h).	**string**	**--autoscaler-scale-down-unneeded-time 1h**
--autoscaler-scale-down-utilization-threshold *float*	Node utilization level, defined as sum of requested resources divided by capacity, below which a node can be considered for scale down. Value must be between 0 and 1.	**float**	**--autoscaler-scale-down-utilization-threshold 0.5**
--autoscaler-scale-down-delay-after-add *string*	How long after scale up that scale down evaluation resumes. Replace *string* in the command with an integer and time unit (ns,us,μs,ms,s,m,h).	**string**	**--autoscaler-scale-down-delay-after-add 1h**
--autoscaler-scale-down-delay-after-delete *string*	How long after node deletion that scale down evaluation resumes. Replace *string* in the command with an integer and time unit (ns,us,μs,ms,s,m,h).	**string**	**--autoscaler-scale-down-delay-after-delete 1h**
--autoscaler-scale-down-delay-after-failure *string*	How long after scale down failure that scale down evaluation resumes. Replace *string* in the command with an integer and time unit (ns,us,μs,ms,s,m,h).	**string**	**--autoscaler-scale-down-delay-after-failure 1h**

CHAPTER 4. MANAGE NODES USING MACHINE POOLS

4.1. ABOUT MACHINE POOLS

Red Hat OpenShift Service on AWS uses machine pools as an elastic, dynamic provisioning method on top of your cloud infrastructure.

The primary resources are machines, compute machine sets, and machine pools.

4.1.1. Machines

A machine is a fundamental unit that describes the host for a worker node.

4.1.2. Machine sets

MachineSet resources are groups of compute machines. If you need more machines or must scale them down, change the number of replicas in the machine pool to which the compute machine sets belong.

Machine sets are not directly modifiable in ROSA.

4.1.3. Machine pools

Machine pools are a higher level construct to compute machine sets.

A machine pool creates compute machine sets that are all clones of the same configuration across availability zones. Machine pools perform all of the host node provisioning management actions on a worker node. If you need more machines or must scale them down, change the number of replicas in the machine pool to meet your compute needs. You can manually configure scaling or set autoscaling.

By default, a cluster has one machine pool. During cluster installation, you can define instance type or size and add labels to this machine pool.

After a cluster's installation:

- You can remove or add labels to any machine pool.

- You can add additional machine pools to an existing cluster.

- You can add taints to any machine pool as long as there is one machine pool without any taints.

- You can create or delete a machine pool as long as there is one machine pool without any taints and at least two replicas for a Single-AZ cluster or three replicas for a Multi-AZ cluster.

> **NOTE**
>
> You cannot change the machine pool node type or size. The machine pool node type or size is specified during their creation only. If you need a different node type or size, you must re-create a machine pool and specify the required node type or size values.

- You can add a label to each added machine pool.

Multiple machine pools can exist on a single cluster, and each machine pool can contain a unique node type and node size configuration.

4.1.4. Machine pools in multiple zone clusters

In a cluster created across multiple Availability Zones (AZ), the machine pools can be created across either all of the three AZs or any single AZ of your choice. The machine pool created by default at the time of cluster creation will be created with machines in all three AZs and scale in multiples of three.

If you create a new Multi-AZ cluster, the machine pools are replicated to those zones automatically. By default, if you add a machine pool to an existing Multi-AZ cluster, the new machine pool is automatically created in all of the zones.
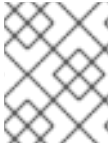
> **NOTE**
>
> You can override this default setting and create a machine pool in a Single-AZ of your choice.

Similarly, deleting a machine pool will delete it from all zones. Due to this multiplicative effect, using machine pools in Multi-AZ cluster can consume more of your project's quota for a specific region when creating machine pools.

4.1.5. Machine pools in ROSA with HCP clusters

In ROSA with HCP clusters, the hosted control plane spans three availability zones (AZ) in the installed cloud region. Each machine pool in a ROSA with HCP cluster deploys in a single subnet within a single AZ. Each of these AZs can have only one machine pool.

Each machine pool in an ROSA with HCP cluster upgrades independently. Because the machine pools upgrade independently, they must remain within 2 minor (Y-stream) versions of the hosted control plane. For example, if your hosted control plane is 4.16.z, your machine pools must be at least 4.14.z.

The following image depicts how machine pools work within ROSA and ROSA with HCP clusters:

HCP (>=X.1)

ROSA hosted control plane 4.y.z

VPC

Machine pool 1 – 4.y.z
Replica 2
Nodes
Availability zone 1

Machine pool 2 – 4.(y-1).z
Replica 3
Nodes
Availability zone 2

Machine pool 3 – 4.(y-2).z
Replica 4
Nodes
Availability zone 3

Classic Multi-AZ

VPC

Control plane 4.y.z

Machine pool 1 – 4.y.z
Minimum replica 3
Nodes

Machine pool 2 – 4.y.z
Replica 1
Nodes
Availability zone 1

Nodes

Machine pool 3 – 4.y.z
Replica 1
Nodes
Availability zone 2

Nodes

Availability zone 3

Classic Single-AZ

Not highly available

VPC

Control plane 4.y.z

Machine pool 1 – 4.y.z
Nodes

Machine pool 2 – 4.y.z
Nodes

Availability zone

638_OpenShift_0524

NOTE

Machine pools in ROSA with HCP clusters each upgrade independently and the machine pool versions must remain within two minor (Y-stream) versions of the control plane.

4.1.6. Additional resources

- Managing compute nodes

- About autoscaling

- [Configuring PID limits](#)

4.2. MANAGING COMPUTE NODES

This document describes how to manage compute (also known as worker) nodes with Red Hat OpenShift Service on AWS (ROSA).

The majority of changes for compute nodes are configured on machine pools. A machine pool is a group of compute nodes in a cluster that have the same configuration, providing ease of management.

You can edit machine pool configuration options such as scaling, adding node labels, and adding taints.

4.2.1. Creating a machine pool

A machine pool is created when you install a Red Hat OpenShift Service on AWS (ROSA) cluster. After installation, you can create additional machine pools for your cluster by using OpenShift Cluster Manager or the ROSA CLI (**rosa**).

> **NOTE**
>
> For users of ROSA CLI **rosa** version 1.2.25 and earlier versions, the machine pool created along with the cluster is identified as **Default**. For users of ROSA CLI **rosa** version 1.2.26 and later, the machine pool created along with the cluster is identified as **worker**.

4.2.1.1. Creating a machine pool using OpenShift Cluster Manager

You can create additional machine pools for your Red Hat OpenShift Service on AWS (ROSA) cluster by using OpenShift Cluster Manager.

Prerequisites

- You created a ROSA cluster.

Procedure

1. Navigate to [OpenShift Cluster Manager](#) and select your cluster.

2. Under the **Machine pools** tab, click **Add machine pool**.

3. Add a **Machine pool name**.

4. Select a **Compute node instance type** from the drop-down menu. The instance type defines the vCPU and memory allocation for each compute node in the machine pool.

   > **NOTE**
   >
   > You cannot change the instance type for a machine pool after the pool is created.

5. Optional: Configure autoscaling for the machine pool:

   a. Select **Enable autoscaling** to automatically scale the number of machines in your machine pool to meet the deployment needs.

b. Set the minimum and maximum node count limits for autoscaling. The cluster autoscaler does not reduce or increase the machine pool node count beyond the limits that you specify.

- If you deployed your cluster using a single availability zone, set the **Minimum and maximum node count**. This defines the minimum and maximum compute node limits in the availability zone.

- If you deployed your cluster using multiple availability zones, set the **Minimum nodes per zone** and **Maximum nodes per zone**. This defines the minimum and maximum compute node limits per zone.

> **NOTE**
>
> Alternatively, you can set your autoscaling preferences for the machine pool after the machine pool is created.

6. If you did not enable autoscaling, select a compute node count:

- If you deployed your cluster using a single availability zone, select a **Compute node count** from the drop-down menu. This defines the number of compute nodes to provision to the machine pool for the zone.

- If you deployed your cluster using multiple availability zones, select a **Compute node count (per zone)** from the drop-down menu. This defines the number of compute nodes to provision to the machine pool per zone.

7. Optional: Configure **Root disk size**.

8. Optional: Add node labels and taints for your machine pool:

   a. Expand the **Edit node labels and taints** menu.

   b. Under **Node labels**, add **Key** and **Value** entries for your node labels.

   c. Under **Taints**, add **Key** and **Value** entries for your taints.

   > **NOTE**
   >
   > Creating a machine pool with taints is only possible if the cluster already has at least one machine pool without a taint.

   d. For each taint, select an **Effect** from the drop-down menu. Available options include **NoSchedule**, **PreferNoSchedule**, and **NoExecute**.
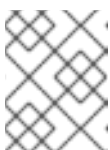
   > **NOTE**
   >
   > Alternatively, you can add the node labels and taints after you create the machine pool.

9. Optional: Select additional custom security groups to use for nodes in this machine pool. You must have already created the security groups and associated them with the VPC that you selected for this cluster. You cannot add or edit security groups after you create the machine pool. For more information, see the requirements for security groups in the "Additional resources" section.
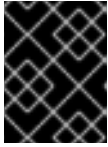
> **IMPORTANT**
>
> You can use up to ten additional security groups for machine pools on ROSA with HCP clusters.

10. Optional: Use Amazon EC2 Spot Instances if you want to configure your machine pool to deploy machines as non-guaranteed AWS Spot Instances:

    a. Select **Use Amazon EC2 Spot Instances**.

    b. Leave **Use On-Demand instance price** selected to use the on-demand instance price. Alternatively, select **Set maximum price** to define a maximum hourly price for a Spot Instance.
       For more information about Amazon EC2 Spot Instances, see the AWS documentation.
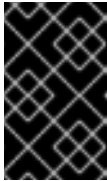
    > **IMPORTANT**
    >
    > Your Amazon EC2 Spot Instances might be interrupted at any time. Use Amazon EC2 Spot Instances only for workloads that can tolerate interruptions.

    > **NOTE**
    >
    > If you select **Use Amazon EC2 Spot Instances** for a machine pool, you cannot disable the option after the machine pool is created.

11. Click **Add machine pool** to create the machine pool.

Verification

- Verify that the machine pool is visible on the **Machine pools** page and the configuration is as expected.

Additional resources

- Additional custom security groups

4.2.1.2. Creating a machine pool using the ROSA CLI

You can create additional machine pools for your Red Hat OpenShift Service on AWS (ROSA) cluster by using the ROSA CLI (**rosa**).

Prerequisites

- You installed and configured the latest Red Hat OpenShift Service on AWS (ROSA) CLI, **rosa**, on your workstation.

- You logged in to your Red Hat account using the ROSA CLI (**rosa**).

- You created a ROSA cluster.

Procedure

- To add a machine pool that does not use autoscaling, create the machine pool and define the instance type, compute (also known as worker) node count, and node labels:

```
$ 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 Specifies the name of the machine pool. Replace **<machine_pool_id>** with the name of your machine pool.

2 Specifies the number of compute nodes to provision. If you deployed ROSA using a single availability zone, this defines the number of compute nodes to provision to the machine pool for the zone. If you deployed your cluster using multiple availability zones, this defines the number of compute nodes to provision in total across all zones and the count must be a multiple of 3. The **--replicas** argument is required when autoscaling is not configured.

3 Optional: Sets the instance type for the compute nodes in your machine pool. The instance type defines the vCPU and memory allocation for each compute node in the pool. Replace **<instance_type>** with an instance type. The default is **m5.xlarge**. You cannot change the instance type for a machine pool after the pool is created.

4 Optional: Defines the labels for the machine pool. Replace **<key>=<value>,<key>=<value>** with a comma-delimited list of key-value pairs, for example **--labels=key1=value1,key2=value2**.

5 Optional: Defines the taints for the machine pool. Replace **<key>=<value>:<effect>,<key>=<value>:<effect>** with a key, value, and effect for each taint, for example **--taints=key1=value1:NoSchedule,key2=value2:NoExecute**. Available effects include **NoSchedule**, **PreferNoSchedule**, and **NoExecute**.

6 Optional: Configures your machine pool to deploy machines as non-guaranteed AWS Spot Instances. For information, see Amazon EC2 Spot Instances in the AWS documentation. If you select **Use Amazon EC2 Spot Instances** for a machine pool, you cannot disable the option after the machine pool is created.

7 Optional: If you choose to use Spot Instances, you can specify this argument to define a maximum hourly price for a Spot Instance. If this argument is not specified, the on-demand price is used.

> **IMPORTANT**
>
> Your Amazon EC2 Spot Instances might be interrupted at any time. Use Amazon EC2 Spot Instances only for workloads that can tolerate interruptions.

8 Optional: Specifies the worker node disk size. The value can be in GB, GiB, TB, or TiB. Replace **<disk_size>** with a numeric value and unit, for example **--disk-size=200GiB**.

9 Optional: For Multi-AZ clusters, you can create a machine pool in a Single-AZ of your choice. Replace **<az>** with a Single-AZ name.

> **NOTE**
>
> Multi-AZ clusters retain a Multi-AZ control plane and can have worker machine pools across a Single-AZ or Multi-AZ. Machine pools distribute machines (nodes) evenly across availability zones.

> **WARNING**
>
> If you choose a worker machine pool with a Single-AZ, there is no fault tolerance for that machine pool, regardless of machine replica count. For fault-tolerant worker machine pools, choosing a Multi-AZ machine pool distributes machines in multiples of 3 across availability zones.
>
> - A Multi-AZ machine pool with three availability zones can have a machine count in multiples of 3 only, such as 3, 6, 9, and so on.
>
> - A Single-AZ machine pool with one availability zone can have a machine count in multiples of 1, such as 1,2,3,4 and so on.

10 Optional: For machine pools in clusters that do not have Red Hat managed VPCs, you can select additional custom security groups to use in your machine pools. You must have already created the security groups and associated them with the VPC that you selected for this cluster. You cannot add or edit security groups after you create the machine pool. For more information, see the requirements for security groups in the "Additional resources" section.

> **IMPORTANT**
>
> You can use up to ten additional security groups for machine pools on ROSA with HCP clusters.

11 Optional: For BYO VPC clusters, you can select a subnet to create a Single-AZ machine pool. If the subnet is out of your cluster creation subnets, there must be a tag with a key **kubernetes.io/cluster/<infra-id>** and value **shared**. Customers can obtain the Infra ID by using the following command:

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

Example output

```
Infra ID:               mycluster-xqvj7
```
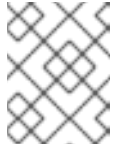
> **NOTE**
>
> You cannot set both **--subnet** and **--availability-zone** at the same time, only 1 is allowed for a Single-AZ machine pool creation.

The following example creates a machine pool called **mymachinepool** that uses the **m5.xlarge** instance type and has 2 compute node replicas. The example also adds 2 workload-specific labels:

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

Example output

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

- To add a machine pool that uses autoscaling, create the machine pool and define the autoscaling configuration, instance type and node labels:

```
$ 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 Specifies the name of the machine pool. Replace **<machine_pool_id>** with the name of your machine pool.

2 Enables autoscaling in the machine pool to meet the deployment needs.

3 4 Defines the minimum and maximum compute node limits. The cluster autoscaler does not reduce or increase the machine pool node count beyond the limits that you specify. If you deployed ROSA using a single availability zone, the **--min-replicas** and **--max-replicas** arguments define the autoscaling limits in the machine pool for the zone. If you deployed your cluster using multiple availability zones, the arguments define the autoscaling limits in total across all zones and the counts must be multiples of 3.

5 Optional: Sets the instance type for the compute nodes in your machine pool. The instance type defines the vCPU and memory allocation for each compute node in the pool. Replace **<instance_type>** with an instance type. The default is **m5.xlarge**. You cannot change the instance type for a machine pool after the pool is created.

6 Optional: Defines the labels for the machine pool. Replace **<key>=<value>,<key>=<value>** with a comma-delimited list of key-value pairs, for example **--labels=key1=value1,key2=value2**.

7 Optional: Defines the taints for the machine pool. Replace **<key>=<value>:<effect>, <key>=<value>:<effect>** with a key, value, and effect for each taint, for example **--**

8 Optional: Configures your machine pool to deploy machines as non-guaranteed AWS Spot Instances. For information, see Amazon EC2 Spot Instances in the AWS documentation. If you select **Use Amazon EC2 Spot Instances** for a machine pool, you cannot disable the option after the machine pool is created.

> **IMPORTANT**
>
> Your Amazon EC2 Spot Instances might be interrupted at any time. Use Amazon EC2 Spot Instances only for workloads that can tolerate interruptions.

9 Optional: If you choose to use Spot Instances, you can specify this argument to define a maximum hourly price for a Spot Instance. If this argument is not specified, the on-demand price is used.

10 Optional: For Multi-AZ clusters, you can create a machine pool in a Single-AZ of your choice. Replace **<az>** with a Single-AZ name.

The following example creates a machine pool called **mymachinepool** that uses the **m5.xlarge** instance type and has autoscaling enabled. The minimum compute node limit is 3 and the maximum is 6 overall. The example also adds 2 workload-specific labels:

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

Example output

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

Verification

You can list all machine pools on your cluster or describe individual machine pools.

1. List the available machine pools on your cluster:

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

   Example output

   ```
   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. Describe the information of a specific machine pool in your cluster:

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

–

Example output

```
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. Verify that the machine pool is included in the output and the configuration is as expected.

Additional resources

- [Additional custom security groups](#)

4.2.2. Configuring machine pool disk volume

Machine pool disk volume size can be configured for additional flexibility. The default disk size is 300 GiB. For cluster version 4.13 or earlier, the disk size can be configured to a minimum of 128 GiB to a maximum of 1 TiB. For cluster version 4.14 and later, the disk size can be configured to a minimum of 128 GiB to a maximum of 16 TiB.

You can configure the machine pool disk size for your cluster by using OpenShift Cluster Manager or the ROSA CLI (**rosa**).

> **NOTE**
>
> Existing cluster and machine pool node volumes cannot be resized.

> **IMPORTANT**
>
> The default disk size is 300 GiB. For cluster version 4.13 or earlier, the disk size can be configured to a minimum of 128 GiB to a maximum of 1 TiB. For cluster version 4.14 and later, the disk size can be configured to a minimum of 128 GiB to a maximum of 16 TiB.

4.2.2.1. Configuring machine pool disk volume using OpenShift Cluster Manager

Prerequisite for cluster creation

- You have the option to select the node disk sizing for the default machine pool during cluster installation.

Procedure for cluster creation

1. From ROSA cluster wizard, navigate to Cluster settings.

2. Navigate to **Machine pool** step.

3. Select the desired **Root disk size**.

4. Select **Next** to continue creating your cluster.

Prerequisite for machine pool creation

- You have the option to select the node disk sizing for the new machine pool after the cluster has been installed.

Procedure for machine pool creation

1. Navigate to [OpenShift Cluster Manager](#) and select your cluster.

2. Navigate to **Machine pool tab**.

3. Click **Add machine pool**.

4. Select the desired **Root disk size**.

5. Select **Add machine pool** to create the machine pool.

4.2.2.2. Configuring machine pool disk volume using the ROSA CLI

Prerequisite for cluster creation

- You have the option to select the root disk sizing for the default machine pool during cluster installation.

Procedure for cluster creation

- Run the following command when creating your OpenShift cluster for the desired root disk size:

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

The value can be in GB, GiB, TB, or TiB. Replace '<disk_size>' with a numeric value and unit, for example '--worker-disk-size=200GiB'. You cannot separate the digit and the unit. No spaces are allowed.

Prerequisite for machine pool creation

- You have the option to select the root disk sizing for the new machine pool after the cluster has been installed.

Procedure for machine pool creation

1. Scale up the cluster by executing the following command:

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

1   Specifies the ID or name of your existing OpenShift cluster

2 Specifies the worker node disk size. The value can be in GB, GiB, TB, or TiB. Replace '<disk_size>' with a numeric value and unit, for example '--disk-size=200GiB'. You cannot separate the digit and the unit. No spaces are allowed.

2. Confirm new machine pool disk volume size by logging into the AWS console and find the EC2 virtual machine root volume size.

Additional resources

- For a detailed list of the arguments that are available for the **rosa create machinepool** subcommand, see Managing objects with the ROSA CLI.

4.2.3. Deleting a machine pool

You can delete a machine pool in the event that your workload requirements have changed and your current machine pools no longer meet your needs.

You can delete machine pools using the OpenShift Cluster Manager or the ROSA CLI (**rosa**).

4.2.3.1. Deleting a machine pool using OpenShift Cluster Manager

You can delete a machine pool for your Red Hat OpenShift Service on AWS (ROSA) cluster by using OpenShift Cluster Manager.

Prerequisites

- You created a ROSA cluster.

- The cluster is in the ready state.

- You have an existing machine pool without any taints and with at least two instances for a single-AZ cluster or three instances for a multi-AZ cluster.

Procedure

1. From OpenShift Cluster Manager, navigate to the **Clusters** page and select the cluster that contains the machine pool that you want to delete.

2. On the selected cluster, select the **Machine pools** tab.

3. Under the **Machine pools** tab, click the options menu ⋮ for the machine pool that you want to delete.

4. Click Delete.

The selected machine pool is deleted.

4.2.3.2. Deleting a machine pool using the ROSA CLI

You can delete a machine pool for your Red Hat OpenShift Service on AWS (ROSA) cluster by using the ROSA CLI.

> **NOTE**
>
> For users of ROSA CLI **rosa** version 1.2.25 and earlier versions, the machine pool (ID='Default') that is created along with the cluster cannot be deleted. For users of ROSA CLI **rosa** version 1.2.26 and later, the machine pool (ID='worker') that is created along with the cluster can be deleted as long as there is one machine pool within the cluster that contains no taints, and at least two replicas for a Single-AZ cluster or three replicas for a Multi-AZ cluster.

Prerequisites

- You created a ROSA cluster.

- The cluster is in the ready state.

- You have an existing machine pool without any taints and with at least two instances for a Single-AZ cluster or three instances for a Multi-AZ cluster.

Procedure

1. From the ROSA CLI, run the following command:

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

   Example output

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

2. Enter 'y' to delete the machine pool.
   The selected machine pool is deleted.

4.2.4. Scaling compute nodes manually

If you have not enabled autoscaling for your machine pool, you can manually scale the number of compute (also known as worker) nodes in the pool to meet your deployment needs.

You must scale each machine pool separately.

Prerequisites

- You installed and configured the latest Red Hat OpenShift Service on AWS (ROSA) CLI, **rosa**, on your workstation.

- You logged in to your Red Hat account using the ROSA CLI (**rosa**).

- You created a Red Hat OpenShift Service on AWS (ROSA) cluster.

- You have an existing machine pool.

Procedure

1. List the machine pools in the cluster:

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

Example output

```
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. Increase or decrease the number of compute node replicas in a machine pool:

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

1 If you deployed Red Hat OpenShift Service on AWS (ROSA) using a single availability zone, the replica count defines the number of compute nodes to provision to the machine pool for the zone. If you deployed your cluster using multiple availability zones, the count defines the total number of compute nodes in the machine pool across all zones and must be a multiple of 3.

2 Replace **<machine_pool_id>** with the ID of your machine pool, as listed in the output of the preceding command.

Verification

1. List the available machine pools in your cluster:

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

Example output

```
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. In the output of the preceding command, verify that the compute node replica count is as expected for your machine pool. In the example output, the compute node replica count for the **mp1** machine pool is scaled to 3.

4.2.5. Node labels

A label is a key-value pair applied to a **Node** object. You can use labels to organize sets of objects and control the scheduling of pods.

You can add labels during cluster creation or after. Labels can be modified or updated at any time.

Additional resources

- For more information about labels, see Kubernetes Labels and Selectors overview .

4.2.5.1. Adding node labels to a machine pool

Add or edit labels for compute (also known as worker) nodes at any time to manage the nodes in a manner that is relevant to you. For example, you can assign types of workloads to specific nodes.

Labels are assigned as key–value pairs. Each key must be unique to the object it is assigned to.

Prerequisites

- You installed and configured the latest Red Hat OpenShift Service on AWS (ROSA) CLI, **rosa**, on your workstation.

- You logged in to your Red Hat account using the ROSA CLI (**rosa**).

- You created a Red Hat OpenShift Service on AWS (ROSA) cluster.

- You have an existing machine pool.

Procedure

1. List the machine pools in the cluster:

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

   Example output

   ```
   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. Add or update the node labels for a machine pool:

   - To add or update node labels for a machine pool that does not use autoscaling, run the following command:

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

     ❶ For machine pools that do not use autoscaling, you must provide a replica count when adding node labels. If you do not specify the **--replicas** argument, you are prompted for a replica count before the command completes. If you deployed Red Hat OpenShift Service on AWS (ROSA) using a single availability zone, the replica count defines the number of compute nodes to provision to the machine pool for the zone. If you deployed your cluster using multiple availability zones, the count defines the total number of compute nodes in the machine pool across all zones and must be a multiple of 3.

2   Replace **<key>=<value>,<key>=<value>** with a comma-delimited list of key-value pairs, for example **--labels=key1=value1,key2=value2**. This list overwrites any modifications made to node labels on an ongoing basis.

The following example adds labels to the **db-nodes-mp** machine pool:

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

Example output

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

- To add or update node labels for a machine pool that uses autoscaling, run the following command:

```
$ 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** For machine pools that use autoscaling, you must provide minimum and maximum compute node replica limits. If you do not specify the arguments, you are prompted for the values before the command completes. The cluster autoscaler does not reduce or increase the machine pool node count beyond the limits that you specify. If you deployed ROSA using a single availability zone, the **--min-replicas** and **--max-replicas** arguments define the autoscaling limits in the machine pool for the zone. If you deployed your cluster using multiple availability zones, the arguments define the autoscaling limits in total across all zones and the counts must be multiples of 3.

3   Replace **<key>=<value>,<key>=<value>** with a comma-delimited list of key-value pairs, for example **--labels=key1=value1,key2=value2**. This list overwrites any modifications made to node labels on an ongoing basis.

The following example adds labels to the **db-nodes-mp** machine pool:

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

Example output

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

Verification

1. Describe the details of the machine pool with the new labels:

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

Example output

```
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. Verify that the labels are included for your machine pool in the output.

4.2.6. Adding tags to a machine pool

You can add tags for compute nodes, also known as worker nodes, in a machine pool to introduce custom user tags for AWS resources that are generated when you provision your machine pool.

4.2.6.1. Adding tags to a machine pool using the ROSA CLI

You can add tags to a machine pool for your Red Hat OpenShift Service on AWS cluster by using the ROSA command line interface (CLI).

> **IMPORTANT**
>
> You must ensure that your tag keys are not **aws**, **red-hat-managed**, **red-hat-clustertype**, or **Name**. In addition, you must not set a tag key that begins with **kubernetes.io/cluster/**. Your tag's key cannot be longer than 128 characters, while your tag's value cannot be longer than 256 characters. Red Hat reserves the right to add additional reserved tags in the future.

Prerequisites

- You installed and configured the latest AWS (**aws**), ROSA (**rosa**), and OpenShift (**oc**) CLIs on your workstation.

- You logged in to your Red Hat account by using the **rosa** CLI.

- You created a Red Hat OpenShift Service on AWS (ROSA) cluster.

Procedure

- Create a machine pool with a custom tag by running the following command:

  ```
  $ rosa create machinepools --cluster=<name> --replicas=<replica_count> \
        --name <mp_name> --tags='<key> <value>,<key> <value>' \    1
  ```

  1 Replace **<key> <value>,<key> <value>** with a key and value for each tag.

  Example output

```
$ rosa create machinepools --cluster=mycluster --replicas 2 --tags='tagkey1
tagvalue1,tagkey2 tagvaluev2'

I: Checking available instance types for machine pool 'mp-1'
I: Machine pool 'mp-1' created successfully on cluster 'mycluster'
I: To view the machine pool details, run 'rosa describe machinepool --cluster mycluster --
machinepool mp-1'
I: To view all machine pools, run 'rosa list machinepools --cluster mycluster'
```

Verification

- Use the **describe** command to see the details of the machine pool with the tags, and verify that the tags are included for your machine pool in the output:

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

Example output

```
$ rosa describe machinepool --cluster classic-rosa --machinepool mp-1

ID:                         mp-1
Cluster ID:                 2baiirqa2141oreotoivp4sipq84vp5g
Autoscaling:                No
Replicas:                   2
Instance type:              m5.xlarge
Labels:
Taints:
Availability zones:         us-east-1a
Subnets:
Spot instances:             No
Disk size:                  300 GiB
Additional Security Group IDs:
Tags:                       red-hat-clustertype=rosa, red-hat-managed=true,
tagkey1=tagvalue1, tagkey2=tagvaluev2
```

4.2.7. Adding taints to a machine pool

You can add taints for compute (also known as worker) nodes in a machine pool to control which pods are scheduled to them. When you apply a taint to a machine pool, the scheduler cannot place a pod on the nodes in the pool unless the pod specification includes a toleration for the taint. Taints can be added to a machine pool using the OpenShift Cluster Manager or the Red Hat OpenShift Service on AWS (ROSA) CLI, **rosa**.

> **NOTE**
>
> A cluster must have at least one machine pool that does not contain any taints.

4.2.7.1. Adding taints to a machine pool using OpenShift Cluster Manager

You can add taints to a machine pool for your Red Hat OpenShift Service on AWS (ROSA) cluster by using OpenShift Cluster Manager.

Prerequisites

- You created a Red Hat OpenShift Service on AWS (ROSA) cluster.

- You have an existing machine pool that does not contain any taints and contains at least two instances.

Procedure

1. Navigate to OpenShift Cluster Manager and select your cluster.

2. Under the **Machine pools** tab, click the options menu ⋮ for the machine pool that you want to add a taint to.

3. Select **Edit taints**.

4. Add **Key** and **Value** entries for your taint.

5. Select an **Effect** for your taint from the drop-down menu. Available options include **NoSchedule**, **PreferNoSchedule**, and **NoExecute**.

6. Optional: Select **Add taint** if you want to add more taints to the machine pool.

7. Click **Save** to apply the taints to the machine pool.

Verification

1. Under the **Machine pools** tab, select > next to your machine pool to expand the view.

2. Verify that your taints are listed under **Taints** in the expanded view.

4.2.7.2. Adding taints to a machine pool using the ROSA CLI

You can add taints to a machine pool for your Red Hat OpenShift Service on AWS (ROSA) cluster by using the ROSA CLI.

> **NOTE**
>
> For users of ROSA CLI **rosa** version 1.2.25 and prior versions, the number of taints cannot be changed within the machine pool (ID=**Default**) created along with the cluster. For users of ROSA CLI **rosa** version 1.2.26 and beyond, the number of taints can be changed within the machine pool (ID=**worker**) created along with the cluster. There must be at least one machine pool without any taints and with at least two replicas for a Single-AZ cluster or three replicas for a Multi-AZ cluster.

Prerequisites

- You installed and configured the latest AWS (**aws**), ROSA (**rosa**), and OpenShift (**oc**) CLIs on your workstation.

- You logged in to your Red Hat account by using the **rosa** CLI.

- You created a Red Hat OpenShift Service on AWS (ROSA) cluster.

- You have an existing machine pool that does not contain any taints and contains at least two instances.

Procedure

1. List the machine pools in the cluster by running the following command:

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

   Example output

   ```
   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. Add or update the taints for a machine pool:

   - To add or update taints for a machine pool that does not use autoscaling, run the following command:

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

     ❶ For machine pools that do not use autoscaling, you must provide a replica count when adding taints. If you do not specify the **--replicas** argument, you are prompted for a replica count before the command completes. If you deployed Red Hat OpenShift Service on AWS (ROSA) using a single availability zone, the replica count defines the number of compute nodes to provision to the machine pool for the zone. If you deployed your cluster using multiple availability zones, the count defines the total number of compute nodes in the machine pool across all zones and must be a multiple of 3.

     ❷ Replace **<key>=<value>:<effect>,<key>=<value>:<effect>** with a key, value, and effect for each taint, for example **--taints=key1=value1:NoSchedule,key2=value2:NoExecute**. Available effects include **NoSchedule**, **PreferNoSchedule**, and **NoExecute**.This list overwrites any modifications made to node taints on an ongoing basis.

     The following example adds taints to the **db-nodes-mp** machine pool:

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

     Example output

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

   - To add or update taints for a machine pool that uses autoscaling, run the following command:

     ```
     $ rosa edit machinepool --cluster=<cluster_name> \
     ```

```
                           --min-replicas=<minimum_replica_count> \ ❶
                           --max-replicas=<maximum_replica_count> \ ❷
                           --taints=<key>=<value>:<effect>,<key>=<value>:<effect> \ ❸
                           <machine_pool_id>
```

❶ ❷ For machine pools that use autoscaling, you must provide minimum and maximum compute node replica limits. If you do not specify the arguments, you are prompted for the values before the command completes. The cluster autoscaler does not reduce or increase the machine pool node count beyond the limits that you specify. If you deployed ROSA using a single availability zone, the **--min-replicas** and **--max-replicas** arguments define the autoscaling limits in the machine pool for the zone. If you deployed your cluster using multiple availability zones, the arguments define the autoscaling limits in total across all zones and the counts must be multiples of 3.

❸ Replace **<key>=<value>:<effect>,<key>=<value>:<effect>** with a key, value, and effect for each taint, for example **--taints=key1=value1:NoSchedule,key2=value2:NoExecute**. Available effects include **NoSchedule**, **PreferNoSchedule**, and **NoExecute**. This list overwrites any modifications made to node taints on an ongoing basis.

The following example adds taints to the **db-nodes-mp** machine pool:

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

Example output

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

Verification

1. Describe the details of the machine pool with the new taints:

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

Example output

```
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. Verify that the taints are included for your machine pool in the output.

4.2.8. Adding node tuning to a machine pool

You can add tunings for compute, also called worker, nodes in a machine pool to control their configuration on Red Hat OpenShift Service on AWS (ROSA) with hosted control planes (HCP) clusters.

> **NOTE**
>
> This feature is only supported on Red Hat OpenShift Service on AWS (ROSA) with hosted control planes (HCP) clusters.

Prerequisites

- You installed and configured the latest Red Hat OpenShift Service on AWS (ROSA) CLI, **rosa**, on your workstation.

- You logged in to your Red Hat account by using the ROSA CLI.

- You created a Red Hat OpenShift Service on AWS (ROSA) with hosted control planes (HCP) cluster.

- You have an existing machine pool.

- You have an existing tuning configuration.

Procedure

1. List all of the machine pools in the cluster:

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

   Example output

   ```
   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. You can add tuning configurations to an existing or new machine pool.

   a. Add tunings when creating a machine pool:

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

      Example output

      ```
      ? 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. Add or update the tunings for a machine pool:

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

Example output

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

Verification

1. List the available machine pools in your cluster:

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

   Example output

   > 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. Verify that the tuning config is included for your machine pool in the output.

4.2.9. Configuring node drain grace periods in ROSA with HCP clusters

You can configure the node drain grace period for machine pools in your cluster. The node drain grace period for a machine pool is how long the cluster respects the Pod Disruption Budget protected workloads when upgrading or replacing the machine pool. After this grace period, all remaining workloads are forcibly evicted. The value range for the node drain grace period is from **0** to **1 week**. With the default value **0**, or empty value, the machine pool drains without any time limitation until complete.

Prerequisites

- You installed and configured the latest Red Hat OpenShift Service on AWS (ROSA) CLI, **rosa**, on your workstation.

- You created a Red Hat OpenShift Service on AWS (ROSA) with hosted control planes (HCP) cluster.

- You have an existing machine pool.

Procedure

1. List all of the machine pools in the cluster by running the following command:

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

   Example output

   > ID   AUTOSCALING REPLICAS INSTANCE TYPE [...] AVAILABILITY ZONES
   > SUBNET VERSION AUTOREPAIR TUNING CONFIGS
   > workers   No   2   m5.xlarge   [...] us-east-1a   N/A   4.14.18 Yes
   > db-nodes-mp No   2   m5.xlarge   [...] us-east-1a   No   4.14.18 Yes

2. Check the node drain grace period for a machine pool by running the following command:

```
$ rosa describe machinepool --cluster <cluster_name> <machinepool_name>
```

Example output

```
ID:                        workers
Cluster ID:                2a90jdl0i4p9r9k9956v5ocv40se1kqs
Node drain grace period:          1
```

1   If this value is empty, the machine pool drains without any time limitation until complete.

3. Optional: Update the node drain grace period for a machine pool by running the following command:

```
$ rosa edit machinepool --node-drain-grace-period="<node_drain_grace_period_value>" --cluster=<cluster_name>  <machinepool_name>
```

NOTE

Changing the node drain grace period during a machine pool upgrade applies to future upgrades, not in progress upgrades.

Verification

1. Check the node drain grace period for a machine pool by running the following command:

```
$ rosa describe machinepool --cluster <cluster_name> <machinepool_name>
```

Example output

```
ID:                        workers
Cluster ID:                2a90jdl0i4p9r9k9956v5ocv40se1kqs
Node drain grace period:        30 minutes
```

2. Verify the correct **Node drain grace period** for your machine pool in the output.

4.2.10. Additional resources

- [About machine pools](#)

- [About autoscaling](#)

- [Enabling autoscaling](#)

- [Disabling autoscaling](#)

- [ROSA Service Definition](#)

4.3. CONFIGURING MACHINE POOLS IN LOCAL ZONES

This document describes how to configure Local Zones in machine pools with Red Hat OpenShift Service on AWS (ROSA).

4.3.1. Configuring machine pools in Local Zones

Use the following steps to configure machine pools in Local Zones.

> **IMPORTANT**
>
> AWS Local Zones are supported on Red Hat OpenShift Service on AWS 4.12. See the Red Hat Knowledgebase article for information on how to enable Local Zones.

Prerequisites

- Red Hat OpenShift Service on AWS (ROSA) is generally available in the parent region of choice. See the AWS generally available locations list to determine the Local Zone available to specific AWS regions.

- The ROSA cluster was initially built in an existing Amazon VPC (BYO-VPC).

- The maximum transmission unit (MTU) for the ROSA cluster is set at 1200.
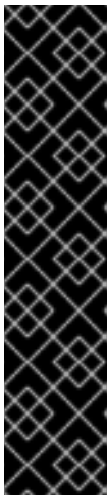
> **IMPORTANT**
>
> Generally, the Maximum Transmission Unit (MTU) between an Amazon EC2 instance in a Local Zone and an Amazon EC2 instance in the Region is 1300. See How Local Zones work in the AWS documentation. The cluster network MTU must always be less than the EC2 MTU to account for the overhead. The specific overhead is determined by your network plugin, for example:
>
> - OVN-Kubernetes: **100 bytes**
>
> - OpenShift SDN: **50 bytes**
>
> The network plugin could provide additional features that may also decrease the MTU. Check the documentation for additional information.

- The AWS account has Local Zones enabled.

- The AWS account has a Local Zone subnet for the same VPC as the cluster.

- The AWS account has a subnet that is associated with a routing table that has a route to a NAT gateway.

- The AWS account has the tag `kubernetes.io/cluster/<infra_id>: shared' on the associated subnet.

Procedure

1. Create a machine pool on the cluster by running the following ROSA CLI (**rosa**) command.

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

2. Add the subnet and instance type for the machine pool in the ROSA CLI. After several minutes, the cluster will provision the nodes.

   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 Enables interactive mode.

   2 Names the machine pool. This is limited to alphanumeric and a maximum length of 30 characters.

   3 Set this option to no.

   4 Set this option to yes.

   5 Selects a subnet ID from the list.

   6 Select yes to enable autoscaling or no to disable autoscaling.

   7 Selects the number of machines for the machine pool. This number can be anywhere from 1 – 180.

   8 Selects an instance type from the list. Only instance types that are supported in the selected Local Zone will appear.

   9 Optional: Specifies the worker node disk size. The value can be in GB, GiB, TB, or TiB. Set a numeric value and unit, for example '200GiB'. You cannot separate the digit and the unit. No spaces are allowed.

3. Provide the subnet ID to provision the machine pool in the Local Zone.

See the AWS Local Zones locations list on AWS for generally available and announced AWS Local Zone locations.

4.4. ABOUT AUTOSCALING NODES ON A CLUSTER

The autoscaler option can be configured to automatically scale the number of machines in a cluster.

The cluster autoscaler increases the size of the cluster when there are pods that failed to schedule on any of the current nodes due to insufficient resources or when another node is necessary to meet deployment needs. The cluster autoscaler does not increase the cluster resources beyond the limits that you specify.

Additionally, the cluster autoscaler decreases the size of the cluster when some nodes are consistently not needed for a significant period, such as when it has low resource use and all of its important pods can fit on other nodes.

When you enable autoscaling, you must also set a minimum and maximum number of worker nodes.

> **NOTE**
>
> Only cluster owners and organization admins can scale or delete a cluster.

4.4.1. Enabling autoscaling nodes on a cluster

You can enable autoscaling on worker nodes to increase or decrease the number of nodes available by editing the machine pool definition for an existing cluster.

Enabling autoscaling nodes in an existing cluster using Red Hat OpenShift Cluster Manager
Enable autoscaling for worker nodes in the machine pool definition from OpenShift Cluster Manager console.

Procedure

1. From OpenShift Cluster Manager, navigate to the **Clusters** page and select the cluster that you want to enable autoscaling for.

2. On the selected cluster, select the **Machine pools** tab.

3. Click the Options menu ⋮ at the end of the machine pool that you want to enable autoscaling for and select **Scale**.

4. On the **Edit node count** dialog, select the **Enable autoscaling** checkbox.

5. Select **Apply** to save these changes and enable autoscaling for the cluster.

> **NOTE**
>
> Additionally, you can configure autoscaling on the default machine pool when you create the cluster using interactive mode.

Enabling autoscaling nodes in an existing cluster using the ROSA CLI
Configure autoscaling to dynamically scale the number of worker nodes up or down based on load.

Successful autoscaling is dependent on having the correct AWS resource quotas in your AWS account. Verify resource quotas and request quota increases from the AWS console.

Procedure

1. To identify the machine pool IDs in a cluster, enter the following command:

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

   Example output

   ```
   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. Get the ID of the machine pools that you want to configure.

3. To enable autoscaling on a machine pool, enter the following command:

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

   ### Example

   Enable autoscaling on a machine pool with the ID **mp1** on a cluster named **mycluster**, with the number of replicas set to scale between 2 and 5 worker nodes:

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

4.4.2. Disabling autoscaling nodes on a cluster

You can disable autoscaling on worker nodes to increase or decrease the number of nodes available by editing the machine pool definition for an existing cluster.

You can disable autoscaling on a cluster using OpenShift Cluster Manager console or the Red Hat OpenShift Service on AWS CLI.

> **NOTE**
>
> Additionally, you can configure autoscaling on the default machine pool when you create the cluster using interactive mode.

Disabling autoscaling nodes in an existing cluster using Red Hat OpenShift Cluster Manager
Disable autoscaling for worker nodes in the machine pool definition from OpenShift Cluster Manager console.

Procedure

1. From OpenShift Cluster Manager, navigate to the **Clusters** page and select the cluster with autoscaling that must be disabled.

2. On the selected cluster, select the **Machine pools** tab.

3. Click the Options menu ⋮ at the end of the machine pool with autoscaling and select **Scale**.

4. On the "Edit node count" dialog, deselect the **Enable autoscaling** checkbox.

5. Select **Apply** to save these changes and disable autoscaling from the cluster.

Disabling autoscaling nodes in an existing cluster using the ROSA CLI
Disable autoscaling for worker nodes in the machine pool definition using the Red Hat OpenShift Service on AWS (ROSA) CLI, **rosa**.

Procedure

1. Enter the following command:

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

   Example

   Disable autoscaling on the **default** machine pool on a cluster named **mycluster**:

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

4.4.3. Additional resources

- Troubleshooting: Autoscaling is not scaling down nodes

- About machinepools

- Managing compute nodes

- Managing objects with the ROSA CLI

4.5. CONFIGURING CLUSTER MEMORY TO MEET CONTAINER MEMORY AND RISK REQUIREMENTS

As a cluster administrator, you can help your clusters operate efficiently through managing application memory by:

- Determining the memory and risk requirements of a containerized application component and configuring the container memory parameters to suit those requirements.

- Configuring containerized application runtimes (for example, OpenJDK) to adhere optimally to the configured container memory parameters.

- Diagnosing and resolving memory-related error conditions associated with running in a container.

4.5.1. Understanding managing application memory

It is recommended to fully read the overview of how Red Hat OpenShift Service on AWS manages Compute Resources before proceeding.

For each kind of resource (memory, CPU, storage), Red Hat OpenShift Service on AWS allows optional **request** and **limit** values to be placed on each container in a pod.

Note the following about memory requests and memory limits:

- **Memory request**

  - The memory request value, if specified, influences the Red Hat OpenShift Service on AWS scheduler. The scheduler considers the memory request when scheduling a container to a node, then fences off the requested memory on the chosen node for the use of the container.

- If a node's memory is exhausted, Red Hat OpenShift Service on AWS prioritizes evicting its containers whose memory usage most exceeds their memory request. In serious cases of memory exhaustion, the node OOM killer may select and kill a process in a container based on a similar metric.

- The cluster administrator can assign quota or assign default values for the memory request value.

- The cluster administrator can override the memory request values that a developer specifies, to manage cluster overcommit.

- **Memory limit**

  - The memory limit value, if specified, provides a hard limit on the memory that can be allocated across all the processes in a container.

  - If the memory allocated by all of the processes in a container exceeds the memory limit, the node Out of Memory (OOM) killer will immediately select and kill a process in the container.

  - If both memory request and limit are specified, the memory limit value must be greater than or equal to the memory request.

  - The cluster administrator can assign quota or assign default values for the memory limit value.

  - The minimum memory limit is 12 MB. If a container fails to start due to a **Cannot allocate memory** pod event, the memory limit is too low. Either increase or remove the memory limit. Removing the limit allows pods to consume unbounded node resources.

4.5.1.1. Managing application memory strategy

The steps for sizing application memory on Red Hat OpenShift Service on AWS are as follows:

1. **Determine expected container memory usage**
   Determine expected mean and peak container memory usage, empirically if necessary (for example, by separate load testing). Remember to consider all the processes that may potentially run in parallel in the container: for example, does the main application spawn any ancillary scripts?

2. **Determine risk appetite**
   Determine risk appetite for eviction. If the risk appetite is low, the container should request memory according to the expected peak usage plus a percentage safety margin. If the risk appetite is higher, it may be more appropriate to request memory according to the expected mean usage.

3. **Set container memory request**
   Set container memory request based on the above. The more accurately the request represents the application memory usage, the better. If the request is too high, cluster and quota usage will be inefficient. If the request is too low, the chances of application eviction increase.

4. **Set container memory limit, if required**
   Set container memory limit, if required. Setting a limit has the effect of immediately killing a container process if the combined memory usage of all processes in the container exceeds the limit, and is therefore a mixed blessing. On the one hand, it may make unanticipated excess memory usage obvious early ("fail fast"); on the other hand it also terminates processes abruptly.

Note that some Red Hat OpenShift Service on AWS clusters may require a limit value to be set; some may override the request based on the limit; and some application images rely on a limit value being set as this is easier to detect than a request value.

If the memory limit is set, it should not be set to less than the expected peak container memory usage plus a percentage safety margin.

5. **Ensure application is tuned**
   Ensure application is tuned with respect to configured request and limit values, if appropriate. This step is particularly relevant to applications which pool memory, such as the JVM. The rest of this page discusses this.

4.5.2. Understanding OpenJDK settings for Red Hat OpenShift Service on AWS

The default OpenJDK settings do not work well with containerized environments. As a result, some additional Java memory settings must always be provided whenever running the OpenJDK in a container.

The JVM memory layout is complex, version dependent, and describing it in detail is beyond the scope of this documentation. However, as a starting point for running OpenJDK in a container, at least the following three memory-related tasks are key:

1. Overriding the JVM maximum heap size.

2. Encouraging the JVM to release unused memory to the operating system, if appropriate.

3. Ensuring all JVM processes within a container are appropriately configured.

Optimally tuning JVM workloads for running in a container is beyond the scope of this documentation, and may involve setting multiple additional JVM options.

4.5.2.1. Understanding how to override the JVM maximum heap size

For many Java workloads, the JVM heap is the largest single consumer of memory. Currently, the OpenJDK defaults to allowing up to 1/4 (1/**-XX:MaxRAMFraction**) of the compute node's memory to be used for the heap, regardless of whether the OpenJDK is running in a container or not. It is therefore **essential** to override this behavior, especially if a container memory limit is also set.

There are at least two ways the above can be achieved:

- If the container memory limit is set and the experimental options are supported by the JVM, set **-XX:+UnlockExperimentalVMOptions -XX:+UseCGroupMemoryLimitForHeap**.

  > **NOTE**
  >
  > The **UseCGroupMemoryLimitForHeap** option has been removed in JDK 11. Use **-XX:+UseContainerSupport** instead.

  This sets **-XX:MaxRAM** to the container memory limit, and the maximum heap size (**-XX:MaxHeapSize** / **-Xmx**) to 1/**-XX:MaxRAMFraction** (1/4 by default).

- Directly override one of **-XX:MaxRAM**, **-XX:MaxHeapSize** or **-Xmx**.
  This option involves hard-coding a value, but has the advantage of allowing a safety margin to be calculated.

4.5.2.2. Understanding how to encourage the JVM to release unused memory to the operating system

By default, the OpenJDK does not aggressively return unused memory to the operating system. This may be appropriate for many containerized Java workloads, but notable exceptions include workloads where additional active processes co-exist with a JVM within a container, whether those additional processes are native, additional JVMs, or a combination of the two.

Java-based agents can use the following JVM arguments to encourage the JVM to release unused memory to the operating system:

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

These arguments are intended to return heap memory to the operating system whenever allocated memory exceeds 110% of in-use memory (**-XX:MaxHeapFreeRatio**), spending up to 20% of CPU time in the garbage collector (**-XX:GCTimeRatio**). At no time will the application heap allocation be less than the initial heap allocation (overridden by **-XX:InitialHeapSize** / **-Xms**). Detailed additional information is available Tuning Java's footprint in OpenShift (Part 1) , Tuning Java's footprint in OpenShift (Part 2) , and at OpenJDK and Containers.

4.5.2.3. Understanding how to ensure all JVM processes within a container are appropriately configured

In the case that multiple JVMs run in the same container, it is essential to ensure that they are all configured appropriately. For many workloads it will be necessary to grant each JVM a percentage memory budget, leaving a perhaps substantial additional safety margin.

Many Java tools use different environment variables (**JAVA_OPTS**, **GRADLE_OPTS**, and so on) to configure their JVMs and it can be challenging to ensure that the right settings are being passed to the right JVM.

The **JAVA_TOOL_OPTIONS** environment variable is always respected by the OpenJDK, and values specified in **JAVA_TOOL_OPTIONS** will be overridden by other options specified on the JVM command line. By default, to ensure that these options are used by default for all JVM workloads run in the Java-based agent image, the Red Hat OpenShift Service on AWS Jenkins Maven agent image sets:

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

> **NOTE**
>
> The **UseCGroupMemoryLimitForHeap** option has been removed in JDK 11. Use **-XX:+UseContainerSupport** instead.

This does not guarantee that additional options are not required, but is intended to be a helpful starting point.

4.5.3. Finding the memory request and limit from within a pod

An application wishing to dynamically discover its memory request and limit from within a pod should use the Downward API.

Procedure

1. Configure the pod to add the **MEMORY_REQUEST** and **MEMORY_LIMIT** stanzas:

   a. Create a YAML file similar to the following:

   ```
   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 ❶
        valueFrom:
          resourceFieldRef:
            containerName: test
            resource: requests.memory
      - name: MEMORY_LIMIT ❷
        valueFrom:
          resourceFieldRef:
            containerName: test
            resource: limits.memory
      resources:
        requests:
          memory: 384Mi
        limits:
          memory: 512Mi
      securityContext:
        allowPrivilegeEscalation: false
        capabilities:
          drop: [ALL]
   ```

   ❶ Add this stanza to discover the application memory request value.

   ❷ Add this stanza to discover the application memory limit value.

   b. Create the pod by running the following command:

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

Verification

1. Access the pod using a remote shell:

```
$ oc rsh test
```

2. Check that the requested values were applied:

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

Example output

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

> **NOTE**
>
> The memory limit value can also be read from inside the container by the
> **/sys/fs/cgroup/memory/memory.limit_in_bytes** file.

4.5.4. Understanding OOM kill policy

Red Hat OpenShift Service on AWS can kill a process in a container if the total memory usage of all the processes in the container exceeds the memory limit, or in serious cases of node memory exhaustion.

When a process is Out of Memory (OOM) killed, this might result in the container exiting immediately. If the container PID 1 process receives the **SIGKILL**, the container will exit immediately. Otherwise, the container behavior is dependent on the behavior of the other processes.

For example, a container process exited with code 137, indicating it received a SIGKILL signal.

If the container does not exit immediately, an OOM kill is detectable as follows:

1. Access the pod using a remote shell:

```
# oc rsh test
```

2. Run the following command to see the current OOM kill count in **/sys/fs/cgroup/memory/memory.oom_control**:

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

Example output

```
oom_kill 0
```

3. Run the following command to provoke an OOM kill:

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

Example output

```
Killed
```

4. Run the following command to view the exit status of the **sed** command:

```
$ echo $?
```

Example output

```
137
```

The **137** code indicates the container process exited with code 137, indicating it received a SIGKILL signal.

5. Run the following command to see that the OOM kill counter in **/sys/fs/cgroup/memory/memory.oom_control** incremented:

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

Example output

```
oom_kill 1
```

If one or more processes in a pod are OOM killed, when the pod subsequently exits, whether immediately or not, it will have phase **Failed** and reason **OOMKilled**. An OOM-killed pod might be restarted depending on the value of **restartPolicy**. If not restarted, controllers such as the replication controller will notice the pod's failed status and create a new pod to replace the old one.

Use the follwing command to get the pod status:

```
$ oc get pod test
```

Example output

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

- If the pod has not restarted, run the following command to view the pod:

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

  Example output

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

- If restarted, run the following command to view the pod:

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

Example output

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

4.5.5. Understanding pod eviction

Red Hat OpenShift Service on AWS may evict a pod from its node when the node's memory is exhausted. Depending on the extent of memory exhaustion, the eviction may or may not be graceful. Graceful eviction implies the main process (PID 1) of each container receiving a SIGTERM signal, then some time later a SIGKILL signal if the process has not exited already. Non-graceful eviction implies the main process of each container immediately receiving a SIGKILL signal.

An evicted pod has phase **Failed** and reason **Evicted**. It will not be restarted, regardless of the value of **restartPolicy**. However, controllers such as the replication controller will notice the pod's failed status and create a new pod to replace the old one.

```
$ oc get pod test
```

Example output

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

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

Example output

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

CHAPTER 5. CONFIGURING PID LIMITS

A process identifier (PID) is a unique identifier assigned by the Linux kernel to each process or thread currently running on a system. The number of processes that can run simultaneously on a system is limited to 4,194,304 by the Linux kernel. This number might also be affected by limited access to other system resources such as memory, CPU, and disk space.

In Red Hat OpenShift Service on AWS 4.11 and later, by default, a pod can have a maximum of 4,096 PIDs. If your workload requires more than that, you can increase the allowed maximum number of PIDs by configuring a **KubeletConfig** object.

Red Hat OpenShift Service on AWS clusters running versions earlier than 4.11 use a default PID limit of **1024**.

5.1. UNDERSTANDING PROCESS ID LIMITS

In Red Hat OpenShift Service on AWS, consider these two supported limits for process ID (PID) usage before you schedule work on your cluster:

- Maximum number of PIDs per pod.
  The default value is 4,096 in Red Hat OpenShift Service on AWS 4.11 and later. This value is controlled by the **podPidsLimit** parameter set on the node.

- Maximum number of PIDs per node.
  The default value depends on node resources. In Red Hat OpenShift Service on AWS, this value is controlled by the **--system-reserved** parameter, which reserves PIDs on each node based on the total resources of the node.

When a pod exceeds the allowed maximum number of PIDs per pod, the pod might stop functioning correctly and might be evicted from the node. See the Kubernetes documentation for eviction signals and thresholds for more information.

When a node exceeds the allowed maximum number of PIDs per node, the node can become unstable because new processes cannot have PIDs assigned. If existing processes cannot complete without creating additional processes, the entire node can become unusable and require reboot. This situation can result in data loss, depending on the processes and applications being run. Customer administrators and Red Hat Site Reliability Engineering are notified when this threshold is reached, and a **Worker node is experiencing PIDPressure** warning will appear in the cluster logs.

5.2. RISKS OF SETTING HIGHER PROCESS ID LIMITS FOR RED HAT OPENSHIFT SERVICE ON AWS PODS

The **podPidsLimit** parameter for a pod controls the maximum number of processes and threads that can run simultaneously in that pod.

You can increase the value for **podPidsLimit** from the default of 4,096 to a maximum of 16,384. Changing this value might incur downtime for applications, because changing the **podPidsLimit** requires rebooting the affected node.

If you are running a large number of pods per node, and you have a high **podPidsLimit** value on your nodes, you risk exceeding the PID maximum for the node.

To find the maximum number of pods that you can run simultaneously on a single node without exceeding the PID maximum for the node, divide 3,650,000 by your **podPidsLimit** value. For example, if your **podPidsLimit** value is 16,384, and you expect the pods to use close to that number of process

IDs, you can safely run 222 pods on a single node.

> **NOTE**
>
> Memory, CPU, and available storage can also limit the maximum number of pods that can run simultaneously, even when the **podPidsLimit** value is set appropriately. For more information, see "Planning your environment" and "Limits and scalability".

Additional resources

- Instance types

- Planning your environment

- Limits and scalability

5.3. CONFIGURING PID LIMITS ON ROSA CLASSIC CLUSTERS

5.3.1. Setting a higher process ID limit on an existing Red Hat OpenShift Service on AWS cluster

You can set a higher **podPidsLimit** on an existing Red Hat OpenShift Service on AWS (ROSA) cluster by creating or editing a **KubeletConfig** object that changes the **--pod-pids-limit** parameter.
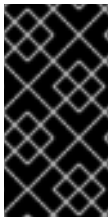
> **IMPORTANT**
>
> Changing the **podPidsLimit** on an existing cluster will trigger non-control plane nodes in the cluster to reboot one at a time. Make this change outside of peak usage hours for your cluster and avoid upgrading or hibernating your cluster until all nodes have rebooted.

Prerequisites

- You have a Red Hat OpenShift Service on AWS cluster.

- You have installed the ROSA CLI (**rosa**).

- You have installed the OpenShift CLI (**oc**).

- You have logged in to your Red Hat account by using the ROSA CLI.

Procedure

1. Create or edit the **KubeletConfig** object to change the PID limit.

   - If this is the first time you are changing the default PID limit, create the **KubeletConfig** object and set the **--pod-pids-limit** value by running the following command:

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

> **NOTE**
>
> The **--name** parameter is optional on ROSA Classic clusters, because only one **KubeletConfig** object is supported per ROSA Classic cluster.

For example, the following command sets a maximum of 16,384 PIDs per pod for cluster **my-cluster**:

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

- If you previously created a **KubeletConfig** object, edit the existing **KubeletConfig** object and set the **--pod-pids-limit** value by running the following command:

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

A cluster-wide rolling reboot of worker nodes is triggered.

2. Verify that all of the worker nodes rebooted by running the following command:

```
$ oc get machineconfigpool
```

Example output

```
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
```

Verification

When each node in the cluster has rebooted, you can verify that the new setting is in place.

- Check the Pod Pids limit in the **KubeletConfig** object:

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

The new PIDs limit appears in the output, as shown in the following example:

Example output

```
Pod Pids Limit:            16384
```

5.3.2. Removing custom configuration from a cluster

You can remove custom configuration from your cluster by removing the **KubeletConfig** object that contains the configuration details.

Prerequisites

- You have an existing Red Hat OpenShift Service on AWS cluster.

- You have installed the ROSA CLI (rosa).

- You have logged in to your Red Hat account by using the ROSA CLI.

Procedure

- Remove custom configuration from the cluster by deleting the relevant custom **KubeletConfig** object:

  ```
  $ rosa delete kubeletconfig --cluster <cluster_name> --name <kubeletconfig_name>
  ```

Verification steps

- Confirm that the custom **KubeletConfig** object is not listed for the cluster:

  ```
  $ rosa describe kubeletconfig --name <cluster_name>
  ```

5.4. CONFIGURING PID LIMITS ON ROSA WITH HCP CLUSTERS

5.4.1. Setting a higher process ID limit on a machine pool in a Red Hat OpenShift Service on AWS cluster

You can set a higher **podPidsLimit** for machine pools in an existing Red Hat OpenShift Service on AWS (ROSA) cluster by creating or editing a **KubeletConfig** object that changes the **--pod-pids-limit** parameter.
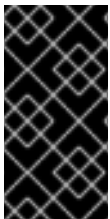
IMPORTANT

Changing the **podPidsLimit** on an existing machine pool triggers nodes in the machine pool to reboot one at a time. Make this change outside of peak usage hours for workloads in your machine pool and avoid upgrading or hibernating your cluster until all nodes have rebooted.

Prerequisites

- You have a Red Hat OpenShift Service on AWS cluster.

- You have installed the ROSA CLI (**rosa**).

- You have logged in to your Red Hat account by using the ROSA CLI.

Procedure

1. Create a new **KubeletConfig** object for your cluster that specifies a new **--pod-pids-limit**:

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

   For example, the following command creates a **set-high-pids KubeletConfig** object for the **my-cluster** cluster that sets a maximum of 16,384 PIDs per pod:

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

2. Associate the new **KubeletConfig** object with a new or existing machine pool.

   - For a new machine pool:

     ```
     $ rosa create machinepool -c <cluster_name> --kubelet-configs=<kubeletconfig_name> -
     -name <machinepool_name>
     ```

   - For an existing machine pool:

     ```
     $ rosa edit machinepool -c <cluster_name> --kubelet-configs=<kubeletconfig_name> --
     name <machinepool_name>
     ```

   For example, the following command associates the **set-high-pids KubeletConfig** object with the **high-pid-pool** machine pool in the **my-cluster** cluster:

   ```
   $ rosa edit machinepool -c my-cluster --kubelet-configs=set-high-pids --name=high-pid-pool
   ```

   A rolling reboot of worker nodes is triggered when a new **KubeletConfig** object is attached to an existing machine pool. You can check the progress of the rollout in the machine pool description:

   ```
   $ rosa describe machinepool --cluster <cluster_name> --name <machinepool_name>
   ```

Verification

- Confirm that the new setting is in place on nodes in the machine pool:

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

  The new PIDs limit appears in the output, as shown in the following example:

  Example output

  ```
  Pod Pids Limit:              16384
  ```

5.4.2. Removing custom configuration from a machine pool

You can remove custom configuration on your machine pools by removing the **KubeletConfig** object that contains the configuration details.

Prerequisites

- You have an existing Red Hat OpenShift Service on AWS cluster.

- You have installed the ROSA CLI (rosa).

- You have logged in to your Red Hat account by using the ROSA CLI.

Procedure

- Edit the machine pool and set the **--kubeletconfigs** parameter so that the **KubeletConfig** object you want to remove is omitted.
  To remove all **KubeletConfig** objects from the machine pool, set an empty value for the **--kubeletconfigs** parameter, for example:

  ```
  $ rosa edit machinepool -c <cluster_name> --kubeletconfigs="" --name
  <machinepool_name>
  ```

Verification steps

- Confirm that the **KubeletConfig** object you removed is not visible in the machine pool description:

  ```
  $ rosa describe machinepool --cluster <cluster_name> --name <machinepool_name>
  ```