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
Abstract

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

## CHAPTER 1. CONFIGURING PRIVATE CONNECTIONS ........................................ 5

1.1. CONFIGURING PRIVATE CONNECTIONS ............................................. 5
1.2. CONFIGURING AWS VPC PEERING ................................................. 5
  1.2.1. VPC peering terms ............................................................. 5
  1.2.2. Initiating the VPC peer request ........................................... 6
  1.2.3. Accepting the VPC peer request ........................................... 7
  1.2.4. Configuring the routing tables ............................................. 7
  1.2.5. Verifying and troubleshooting VPC peering ............................ 8

1.3. CONFIGURING AWS VPN .............................................................. 9
  1.3.1. Creating a VPN connection .................................................. 9
    1.3.1.1. Configuring the VPN connection ..................................... 10
    1.3.1.2. Establishing the VPN Connection .................................... 10
    1.3.1.3. Enabling VPN route propagation .................................... 11
  1.3.2. Verifying the VPN connection ............................................. 11
  1.3.3. Troubleshooting the VPN connection .................................... 12
    Tunnel does not connect ......................................................... 12
    Tunnel does not stay connected ............................................. 13
    Secondary tunnel in Down state ............................................. 13

1.4. CONFIGURING AWS DIRECT CONNECT ........................................ 13
  1.4.1. AWS Direct Connect methods ............................................. 13
  1.4.2. Creating the hosted Virtual Interface ................................... 14
    1.4.2.1. Determining the type of Direct Connect connection ............. 14
    1.4.2.2. Creating a Private Direct Connect .................................. 14
    1.4.2.3. Creating a Public Direct Connect ................................... 15
    1.4.2.4. Verifying the Virtual Interfaces .................................... 16
  1.4.3. Connecting to an existing Direct Connect Gateway .................. 16
  1.4.4. Troubleshooting Direct Connect .......................................... 17

1.5. CONFIGURING A PRIVATE CLUSTER ............................................. 17
  1.5.1. Enabling private cluster on a new cluster ............................. 17
  1.5.2. Enabling private cluster on an existing cluster ....................... 18
  1.5.3. Additional resources ......................................................... 18

## CHAPTER 2. NODES ................................................................. 19

2.1. ABOUT MACHINE POOLS .............................................................. 19
  2.1.1. Machines ............................................................................. 19
  2.1.2. Machine sets ...................................................................... 19
  2.1.3. Machine pools ................................................................. 19
  2.1.4. Machine pools in multiple zone clusters ............................... 19
  2.1.5. Additional resources .......................................................... 20

2.2. MANAGING COMPUTE NODES .................................................... 20
  2.2.1. Creating a machine pool ...................................................... 20
    2.2.1.1. Creating a machine pool using OpenShift Cluster Manager .... 20
    2.2.1.2. Creating a machine pool using the ROSA CLI .................... 22
  2.2.2. Scaling compute nodes manually ....................................... 25
  2.2.3. Adding node labels to a machine pool ................................ 26
  2.2.4. Adding taints to a machine pool ......................................... 28
  2.2.5. Additional resources .......................................................... 30

2.3. ABOUT AUTOSCALING NODES ON A CLUSTER ............................. 31
  2.3.1. Enabling autoscaling nodes on a cluster ............................... 31
    Enabling autoscaling nodes in an existing cluster using Red Hat OpenShift Cluster Manager 31
    Enabling autoscaling nodes in an existing cluster using the rosa CLI 32
2.3.2. Disabling autoscaling nodes on a cluster
   Disabling autoscaling nodes in an existing cluster using Red Hat OpenShift Cluster Manager
   Disabling autoscaling nodes in an existing cluster using the rosa CLI
2.3.3. Additional resources

CHAPTER 3. LOGGING

3.1. ACCESSING THE SERVICE LOGS FOR ROSA CLUSTERS
   3.1.1. Viewing the service logs by using OpenShift Cluster Manager
   3.1.2. Adding cluster notification contacts
3.2. INSTALLING LOGGING ADD-ON SERVICES
   3.2.1. Install the logging add-on service
   3.2.2. Additional resources
3.3. VIEWING CLUSTER LOGS IN THE AWS CONSOLE
   3.3.1. Viewing forwarded logs

CHAPTER 4. MONITORING USER-DEFINED PROJECTS

4.1. UNDERSTANDING THE MONITORING STACK
   4.1.1. Understanding the monitoring stack
      4.1.1.1. Components for monitoring user-defined projects
      4.1.1.2. Monitoring targets for user-defined projects
   4.1.2. Additional resources
   4.1.3. Next steps
4.2. ACCESSING MONITORING FOR USER-DEFINED PROJECTS
   4.2.1. Next steps
4.3. CONFIGURING THE MONITORING STACK
   4.3.1. Maintenance and support for monitoring
      4.3.1.1. Support considerations for monitoring user-defined projects
   4.3.2. Configuring the monitoring stack
   4.3.3. Configurable monitoring components
   4.3.4. Moving monitoring components to different nodes
   4.3.5. Assigning tolerations to components that monitor user-defined projects
   4.3.6. Configuring persistent storage
      4.3.6.1. Persistent storage prerequisites
      4.3.6.2. Configuring a local persistent volume claim
      4.3.6.3. Modifying the retention time for Prometheus metrics data
   4.3.7. Controlling the impact of unbound metrics attributes in user-defined projects
      4.3.7.1. Setting a scrape sample limit for user-defined projects
   4.3.8. Setting log levels for monitoring components
   4.3.9. Next steps
4.4. MANAGING METRICS
   4.4.1. Understanding metrics
   4.4.2. Setting up metrics collection for user-defined projects
      4.4.2.1. Deploying a sample service
      4.4.2.2. Specifying how a service is monitored
   4.4.3. Querying metrics
      4.4.3.1. Querying metrics for all projects as an administrator
      4.4.3.2. Querying metrics for user-defined projects as a developer
      4.4.3.3. Exploring the visualized metrics
   4.4.4. Next steps
4.5. ALERTS
   4.5.1. Next steps
4.6. REVIEWING MONITORING DASHBOARDS
   4.6.1. Reviewing monitoring dashboards as a developer
4.6.2. Next steps

4.7. TROUBLESHOOTING MONITORING ISSUES

4.7.1. Determining why user-defined project metrics are unavailable
CHAPTER 1. CONFIGURING PRIVATE CONNECTIONS

1.1. 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.

1.2. CONFIGURING AWS VPC PEERING

This sample process configures an Amazon Web Services (AWS) VPC containing a 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.

1.2.1. VPC peering terms

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

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red Hat OpenShift Service on AWS AWS Account</td>
<td>The AWS account that contains the Red Hat OpenShift Service on AWS cluster.</td>
</tr>
<tr>
<td>Red Hat OpenShift Service on AWS Cluster VPC</td>
<td>The VPC that contains the Red Hat OpenShift Service on AWS cluster.</td>
</tr>
<tr>
<td>Customer AWS Account</td>
<td>Your non-Red Hat OpenShift Service on AWS AWS Account that you would like to peer with.</td>
</tr>
<tr>
<td>Customer VPC</td>
<td>The VPC in your AWS Account that you would like to peer with.</td>
</tr>
</tbody>
</table>
1.2.2. 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 proceed 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.

---

| 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.
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.

### 1.2.3. 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**.

### 1.2.4. 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.

### 1.2.5. 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:

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

1.3. 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](https://aws.amazon.com) 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](https://aws.amazon.com) documentation.

1.3.1. 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](https://aws.amazon.com) 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.
1.3.1.1. 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.

1.3.1.2. 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.

### 1.3.1.3. 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.

### 1.3.2. 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:
      ```bash
      $ oc run netcat-test \
          --image=busybox -i -t \
          --restart=Never --rm \
          -- /bin/sh
      ```
   b. Check the connection using **nc**.
      - Example successful connection results:
        ```bash
        / 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:
        ```bash
        / 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:
      ```bash
      / exit
      ```

1.3.3. **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.

### 1.4. 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.

#### 1.4.1. 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:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method 1</td>
<td>Create the hosted VIF and then the DXGateway and VGW.</td>
</tr>
<tr>
<td>Method 2</td>
<td>Request a connection via an existing Direct Connect Gateway that you own.</td>
</tr>
</tbody>
</table>

1.4.2. Creating the hosted Virtual Interface

Prerequisites

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

1.4.2.1. 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.

1.4.2.2. 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.

1.4.2.3. 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.
1.4.2.4. 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.

1.4.3. 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.


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**.

### 1.4.4. Troubleshooting Direct Connect

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

### 1.5. CONFIGURING A PRIVATE CLUSTER

A Red Hat OpenShift Service on AWS cluster can be made private so that internal applications can be hosted inside a corporate network. In addition, private clusters can be configured to have only internal API endpoints for increased security.

Privacy settings can be configured during cluster creation or after a cluster is established.

#### 1.5.1. Enabling private cluster on a new cluster

You can enable the private cluster setting when creating a new Red Hat OpenShift Service on AWS cluster.

**IMPORTANT**

Private clusters cannot be used with AWS security token service (STS). However, STS supports AWS PrivateLink clusters.

**Prerequisites**

AWS VPC Peering, VPN, DirectConnect, or **TransitGateway** has been configured to allow private access.

**Procedure**
Enter the following command to create a new private cluster.

```bash
$ rosa create cluster --cluster-name=<cluster_name> --private
```

**NOTE**
Alternatively, use `--interactive` to be prompted for each cluster option.

### 1.5.2. Enabling private cluster on an existing cluster

After a cluster has been created, you can later enable the cluster to be private.

**IMPORTANT**
Private clusters cannot be used with AWS security token service (STS). However, STS supports AWS PrivateLink clusters.

**Prerequisites**
AWS VPC Peering, VPN, DirectConnect, or TransitGateway has been configured to allow private access.

**Procedure**

Enter the following command to enable the `--private` option on an existing cluster.

```bash
$ rosa edit cluster --cluster-name=<cluster_name> --private
```

**NOTE**
Transitioning your cluster between private and public can take several minutes to complete.

### 1.5.3. Additional resources

- Creating an AWS PrivateLink cluster on ROSA
CHAPTER 2. NODES

2.1. 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, machine sets, and machine pools.

IMPORTANT

As of the Red Hat OpenShift Service on AWS versions 4.8.35, 4.9.26, 4.10.6, the Red Hat OpenShift Service on AWS default per-pod pid limit is **4096**. If you want to enable this PID limit, you must upgrade your Red Hat OpenShift Service on AWS clusters to these versions or later. Red Hat OpenShift Service on AWS clusters with prior versions use a default PID limit of **1024**.

You cannot configure the per-pod PID limit on any Red Hat OpenShift Service on AWS cluster.

2.1.1. Machines

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

2.1.2. Machine sets

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

Machine sets are not directly modifiable in ROSA.

2.1.3. Machine pools

Machine pools are a higher level construct to machine sets.

A machine pool creates 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 is created with one machine pool. You can add additional machine pools to an existing cluster, modify the default machine pool, and delete machine pools.

Multiple machine pools can exist on a single cluster, and they can each have different types or different size nodes.

2.1.4. Machine pools in multiple zone clusters

When you create a machine pool in a multiple availability zone (Multi-AZ) cluster, that one machine pool has 3 zones. The machine pool, in turn, creates a total of 3 machine sets - one machine set for each zone in the cluster. Each of those machine sets manages one or more machines in its respective availability zone.

If you create a new Multi-AZ cluster, the machine pools are replicated to those zones automatically. If
you add a machine pool to an existing Multi-AZ, the new pool is automatically created in those zones. 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.

2.1.5. Additional resources

- Managing worker nodes
- About autoscaling

2.2. 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.

2.2.1. Creating a machine pool

A default 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).

2.2.1.1. 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 Worker 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:
Select Enable autoscaling to automatically scale the number of machines in your machine pool to meet the deployment needs.

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.

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

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

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

- Expand the Edit node labels and taints menu.
- Under Node labels, add Key and Value entries for your node labels.
- Under Taints, add Key and Value entries for your taints.
- 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.

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

- Select Use Amazon EC2 Spot Instances.
- 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.

9. 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.

2.2.1.2. 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 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 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> \  
  --replicas=<replica_count> \  
  --instance-type=<instance_type> \  
  --labels=<key>=<value>,<key>=<value> \  
  --taints=<key>=<value>:<effect>,<key>=<value>:<effect> \  
  --use-spot-instances \  
  --spot-max-price=0.5
```

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 have opted 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.

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:

```sh
$ 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> \ 
   --enable-autoscaling \ 
   --min-replicas=<minimum_replica_count> \ 
   --max-replicas=<maximum_replica_count> \ 
   --instance-type=<instance_type> \ 
   --labels=<key>=<value>,<key>=<value>
```

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. 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.

4. 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.

5. 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`.

6. 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`.

7. 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 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.

8. Optional: If you have opted 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.

9. IMPORTANT

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

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
Verification

1. List the available machine pools in your cluster:

   $ rosa list machinepools --cluster=<cluster_name>

   **Example output**

<table>
<thead>
<tr>
<th>ID</th>
<th>AUTOSCALING</th>
<th>REPLICA</th>
<th>INSTANCE TYPE</th>
<th>LABELS</th>
<th>TAINTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>No</td>
<td>3</td>
<td>m5.xlarge</td>
<td>us-east-1a, us-east-1b, us-east-1c</td>
<td>N/A</td>
</tr>
<tr>
<td>mymachinepool</td>
<td>Yes</td>
<td>3-6</td>
<td>m5.xlarge</td>
<td>app=db, tier=backend</td>
<td>us-east-1a, us-east-1b, us-east-1c</td>
</tr>
</tbody>
</table>

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

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](#).

### 2.2.2. 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 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.

**Procedure**

1. List the machine pools in the cluster:

   $ rosa list machinepools --cluster=<cluster_name>

   **Example output**

<table>
<thead>
<tr>
<th>ID</th>
<th>AUTOSCALING</th>
<th>REPLICA</th>
<th>INSTANCE TYPE</th>
<th>LABELS</th>
<th>TAINTS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>AUTOSCALING</th>
<th>REPLICA</th>
<th>INSTANCE TYPE</th>
<th>LABELS</th>
<th>TAINTS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2. Increase or decrease the number of compute node replicas in a machine pool:

```bash
$ 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:

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

**Example output**

```
<table>
<thead>
<tr>
<th>ID</th>
<th>AUTOSCALING</th>
<th>REPLICA</th>
<th>INSTANCE TYPE</th>
<th>LABELS</th>
<th>TAINTS</th>
<th>AVAILABILITY ZONES</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>No</td>
<td>2</td>
<td>m5.xlarge</td>
<td>us-east-1a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mp1</td>
<td>No</td>
<td>3</td>
<td>m5.xlarge</td>
<td>us-east-1a</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

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.

### 2.2.3. 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 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.
Procedure

1. List the machine pools in the cluster:

   $ rosa list machinepools --cluster=<cluster_name>

Example output

<table>
<thead>
<tr>
<th>ID</th>
<th>AUTOSCALING</th>
<th>REPLICAS</th>
<th>INSTANCE TYPE</th>
<th>LABELS</th>
<th>TAINTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>No</td>
<td>2</td>
<td>m5.xlarge</td>
<td></td>
<td>us-east-1a N/A</td>
</tr>
<tr>
<td>db-nodes-mp</td>
<td>No</td>
<td>2</td>
<td>m5.xlarge</td>
<td></td>
<td>us-east-1a No</td>
</tr>
</tbody>
</table>

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> \
     --labels=<key>=<value>,<key>=<value> \
     <machine_pool_id>

     1 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> \
     --max-replicas=<maximum_replica_count> \

     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.

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:

```bash
$ 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. List the available machine pools in your cluster:

   ```bash
   $ 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      app=db, tier=backend              us-east-1a
   No
   ```

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

### 2.2.4. 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.

**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.

Procedure

1. List the machine pools in the cluster:

   $ rosa list machinepools --cluster=<cluster_name>

   Example output

<table>
<thead>
<tr>
<th>ID</th>
<th>AUTOSCALING</th>
<th>REPLICAS</th>
<th>INSTANCE TYPE</th>
<th>LABELS</th>
<th>TAINTS</th>
<th>AVAILABILITY ZONES</th>
<th>SPOT INSTANCES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>No</td>
<td>2</td>
<td>m5.xlarge</td>
<td></td>
<td></td>
<td>us-east-1a</td>
<td>N/A</td>
</tr>
<tr>
<td>db-nodes-mp</td>
<td>No</td>
<td>2</td>
<td>m5.xlarge</td>
<td></td>
<td></td>
<td>us-east-1a</td>
<td>No</td>
</tr>
</tbody>
</table>

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> \1
   --taints=<key>=<value>:<effect>,<key>=<value>:<effect> \2
   <machine_pool_id>

   1 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.

   2 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:

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

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.

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:

```bash
$ 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. List the available machine pools in your cluster:

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

**Example output**

```
<table>
<thead>
<tr>
<th>ID</th>
<th>AUTOSCALING</th>
<th>REPLICA</th>
<th>INSTANCE TYPE</th>
<th>LABELS</th>
<th>TAITNS</th>
<th>AVAILABILITY ZONES</th>
<th>SPOT INSTANCES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>No</td>
<td>2</td>
<td>m5.xlarge</td>
<td></td>
<td>us-east-1a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>db-nodes-mp</td>
<td>No</td>
<td>2</td>
<td>m5.xlarge</td>
<td>key1=value1:NoSchedule, key2=value2:NoExecute</td>
<td>us-east-1a</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

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

**2.2.5. Additional resources**
2.3. 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.

2.3.1. 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.
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:

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

   **Example output**

<table>
<thead>
<tr>
<th>ID</th>
<th>AUTOSCALING</th>
<th>REPLICA</th>
<th>INSTANCE TYPE</th>
<th>LABELS</th>
<th>ZONE</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>No</td>
<td>2</td>
<td>m5.xlarge</td>
<td>us-east-1a</td>
<td></td>
</tr>
<tr>
<td>mp1</td>
<td>No</td>
<td>2</td>
<td>m5.xlarge</td>
<td>us-east-1a</td>
<td></td>
</tr>
</tbody>
</table>

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:

   ```bash
   $ 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:

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

2.3.2. 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.
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.

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

2.3.3. Additional resources

- About machinepools
- Managing worker nodes
- Managing objects with the rosa CLI
CHAPTER 3. LOGGING

3.1. ACCESSING THE SERVICE LOGS FOR ROSA CLUSTERS

You can view the service logs for your Red Hat OpenShift Service on AWS (ROSA) clusters by using Red Hat OpenShift Cluster Manager. The service logs detail cluster events such as load balancer quota updates and scheduled maintenance upgrades. The logs also show cluster resource changes such as the addition or deletion of users, groups, and identity providers.

Additionally, you can add notification contacts for a ROSA cluster. Subscribed users receive emails about cluster events that require customer action, known cluster incidents, upgrade maintenance, and other topics.

3.1.1. Viewing the service logs by using OpenShift Cluster Manager

You can view the service logs for a Red Hat OpenShift Service on AWS (ROSA) cluster by using Red Hat OpenShift Cluster Manager.

**Prerequisites**

- You have installed a ROSA cluster.

**Procedure**

1. Navigate to OpenShift Cluster Manager and select your cluster.
2. In the Overview page for your cluster, view the service logs in the Cluster history section.
3. Optional: Filter the cluster service logs by Description or Severity from the drop-down menu. You can filter further by entering a specific item in the search bar.
4. Optional: Click Download history to download the service logs for your cluster in JSON or CSV format.

3.1.2. Adding cluster notification contacts

You can add notification contacts for your Red Hat OpenShift Service on AWS (ROSA) cluster. When an event occurs that triggers a cluster notification email, subscribed users are notified.

**Procedure**

1. Navigate to OpenShift Cluster Manager and select your cluster.
2. On the Support tab, under the Notification contacts heading, click Add notification contact.
3. Enter the Red Hat username or email of the contact you want to add.

**NOTE**

The username or email address must relate to a user account in the Red Hat organization where the cluster is deployed.

4. Click Add contact.
Verification

- You see a confirmation message when you have successfully added the contact. The user appears under the **Notification contacts** heading on the **Support** tab.

### 3.2. INSTALLING LOGGING ADD-ON SERVICES

This section describes how to install the logging add-on and Amazon Web Services (AWS) CloudWatch log forwarding add-on services on Red Hat OpenShift Service on AWS (ROSA).

The AWS CloudWatch log forwarding service on ROSA has the following approximate log throughput rates. Message rates greater than these can result in dropped log messages.

<table>
<thead>
<tr>
<th>Message size (bytes)</th>
<th>Maximum expected rate (messages/second/node)</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>1,000</td>
</tr>
<tr>
<td>1,024</td>
<td>650</td>
</tr>
<tr>
<td>2,048</td>
<td>450</td>
</tr>
</tbody>
</table>

### 3.2.1. Install the logging add-on service

Red Hat OpenShift Service on AWS (ROSA) provides logging through the **cluster-logging-operator** add-on. This add-on service offers an optional application log forwarding solution based on AWS CloudWatch. This logging solution can be installed after the ROSA cluster is provisioned.

**Procedure**

1. Enter the following command:

   ```bash
   $ rosa install addon cluster-logging-operator --cluster=<cluster_name> --interactive
   ```

   For `<cluster_name>`, enter the name of your cluster.

2. When prompted, accept the default **yes** to install the **cluster-logging-operator**.

3. When prompted, accept the default **yes** to install the optional Amazon CloudWatch log forwarding add-on or enter **no** to decline the installation of this add-on.

   **NOTE**

   It is not necessary to install the AWS CloudWatch service when you install the **cluster-logging-operator**. You can install the AWS CloudWatch service at any time through OpenShift Cluster Manager console from the cluster’s **Add-ons** tab.

4. For the collection of applications, infrastructure, and audit logs, accept the default values or change them as needed:
- **Applications logs**: Lets the Operator collect application logs, which includes everything that is not deployed in the openshift-, kube-, and default namespaces. Default: yes

- **Infrastructure logs**: Lets the Operator collect logs from OpenShift Container Platform, Kubernetes, and some nodes. Default: yes

- **Audit logs**: Type yes to let the Operator collect node logs related to security audits. By default, Red Hat stores audit logs outside the cluster through a separate mechanism that does not rely on the Cluster Logging Operator. For more information about default audit logging, see the ROSA Service Definition. Default: no

5. For the Amazon CloudWatch region, use the default cluster region, leave the Cloudwatch region value empty.

**Example output**

```bash
? Are you sure you want to install add-on 'cluster-logging-operator' on cluster '<cluster_name>'? Yes
? Use AWS CloudWatch (optional): Yes
? Collect Applications logs (optional): Yes
? Collect Infrastructure logs (optional): Yes
? Collect Audit logs (optional): No
? CloudWatch region (optional):
I: Add-on 'cluster-logging-operator' is now installing. To check the status run 'rosa list addons --cluster=<cluster_name>'
```

**NOTE**

The installation can take approximately 10 minutes to complete.

**Verification steps**

1. To verify the logging installation status, enter the following command:

   ```bash
   $ rosa list addons --cluster=<cluster_name>
   ```

2. To verify which pods are deployed by **cluster-logging-operator** and their state of readiness:

   a. Log in to the oc CLI using **cluster-admin** credentials:

   ```bash
   $ oc login https://api.mycluster.abwp.s1.example.org:6443 \
   --username cluster-admin \
   --password <password>
   ```

   b. Enter the following command to get information about the pods for the default project. Alternatively, you can specify a different project.

   ```bash
   $ oc get pods -n openshift-logging
   ```

**Example output**

<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>cluster-logging-operator-&lt;pod_ID &gt;</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
<td>7m1s</td>
</tr>
<tr>
<td>fluentd-4mnwp</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>6m3s</td>
</tr>
</tbody>
</table>
3. Optional: To get information about the `clusterlogging` instance, enter the following command:

```
$ oc get clusterlogging -n openshift-logging
```

4. Optional: To get information about `clusterlogforwarders` instances, enter the following command:

```
$ oc get clusterlogforwarders -n openshift-logging
```

### 3.2.2. Additional resources

- Adding services to your cluster

### 3.3. VIEWING CLUSTER LOGS IN THE AWS CONSOLE

View forwarded cluster logs in the AWS console.

#### 3.3.1. Viewing forwarded logs

Logs that are being forwarded from Red Hat OpenShift Service on AWS are viewed in the Amazon Web Services (AWS) console.

**Prerequisites**

- The `cluster-logging-operator` add-on service is installed and **Cloudwatch** is enabled.

**Procedure**

1. Log in to the AWS console.

2. Select the region the cluster is deployed in.

3. Select the **CloudWatch** service.

4. Select **Logs** from the left column, and select **Log Groups**.

5. Select a log group to explore. You can view application, infrastructure, or audit logs, depending on which types were enabled during the add-on service installation. See the **Amazon CloudWatch User Guide** for more information.
CHAPTER 4. MONITORING USER-DEFINED PROJECTS

4.1. UNDERSTANDING THE MONITORING STACK

In Red Hat OpenShift Service on AWS, you can monitor your own projects in isolation from Red Hat Site Reliability Engineer (SRE) platform metrics. You can monitor your own projects without the need for an additional monitoring solution.

NOTE

Follow the instructions in this document carefully to configure a supported Prometheus instance for monitoring user-defined projects. Custom Prometheus instances are not supported by Red Hat OpenShift Service on AWS.

4.1.1. Understanding the monitoring stack

The Red Hat OpenShift Service on AWS (ROSA) monitoring stack is based on the Prometheus open source project and its wider ecosystem. The monitoring stack includes the following:

- **Default platform monitoring components.** A set of platform monitoring components are installed in the `openshift-monitoring` project and enabled by default during a ROSA installation. This provides monitoring for core cluster components. The default monitoring stack also enables remote health monitoring for clusters. Critical metrics, such as CPU and memory, are collected from all of the workloads in every namespace and are made available for your use. These components are illustrated in the **Installed by default** section in the following diagram.

- **Components for monitoring user-defined projects** This feature is enabled by default and provides monitoring for user-defined projects. These components are illustrated in the **User** section in the following diagram.

4.1.1.1. Components for monitoring user-defined projects

Red Hat OpenShift Service on AWS includes an optional enhancement to the monitoring stack that enables you to monitor services and pods in user-defined projects. This feature includes the following components:

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenShift Projects</td>
<td>Installed by default</td>
</tr>
</tbody>
</table>

Deploy
### Prometheus Operator

The Prometheus Operator in the `openshift-user-workload-monitoring` project creates, configures, and manages Prometheus and Thanos Ruler instances in the same project.

### Prometheus

Prometheus is the monitoring system through which monitoring is provided for user-defined projects. Prometheus sends alerts to Alertmanager for processing. However, alert routing is not currently supported.

### Thanos Ruler

The Thanos Ruler is a rule evaluation engine for Prometheus that is deployed as a separate process. In Red Hat OpenShift Service on AWS 4, Thanos Ruler provides rule and alerting evaluation for the monitoring of user-defined projects.

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prometheus Operator</td>
<td>The Prometheus Operator in the <code>openshift-user-workload-monitoring</code> project creates, configures, and manages Prometheus and Thanos Ruler instances in the same project.</td>
</tr>
<tr>
<td>Prometheus</td>
<td>Prometheus is the monitoring system through which monitoring is provided for user-defined projects. Prometheus sends alerts to Alertmanager for processing. However, alert routing is not currently supported.</td>
</tr>
<tr>
<td>Thanos Ruler</td>
<td>The Thanos Ruler is a rule evaluation engine for Prometheus that is deployed as a separate process. In Red Hat OpenShift Service on AWS 4, Thanos Ruler provides rule and alerting evaluation for the monitoring of user-defined projects.</td>
</tr>
</tbody>
</table>

All of these components are monitored by the stack and are automatically updated when Red Hat OpenShift Service on AWS is updated.

#### 4.1.1.2. Monitoring targets for user-defined projects

Monitoring is enabled by default for Red Hat OpenShift Service on AWS user-defined projects. You can monitor:

- Metrics provided through service endpoints in user-defined projects.
- Pods running in user-defined projects.

#### 4.1.2. Additional resources

- Accessing monitoring for user-defined projects
- Default monitoring components
- Default monitoring targets

#### 4.1.3. Next steps

- Accessing monitoring for user-defined projects

#### 4.2. ACCESSING MONITORING FOR USER-DEFINED PROJECTS

When you install a Red Hat OpenShift Service on AWS (ROSA) cluster, monitoring for user-defined projects is enabled by default. With monitoring for user-defined projects enabled, you can monitor your own ROSA projects without the need for an additional monitoring solution.

The `dedicated-admin` user has default permissions to configure and access monitoring for user-defined projects.
NOTE

Custom Prometheus instances and the Prometheus Operator installed through Operator Lifecycle Manager (OLM) can cause issues with user-defined project monitoring if it is enabled. Custom Prometheus instances are not supported.

Optionally, you can disable monitoring for user-defined projects during or after a cluster installation.

4.2.1. Next steps

- Configuring the monitoring stack

4.3. CONFIGURING THE MONITORING STACK

After you configure the monitoring stack, you can review common configuration scenarios and configure monitoring of user-defined projects.

4.3.1. Maintenance and support for monitoring

The supported way of configuring Red Hat OpenShift Service on AWS Monitoring is by using the options described in this document. **Do not use other configurations, as they are unsupported.**

IMPORTANT

Installing another Prometheus instance is not supported by the Red Hat Site Reliability Engineers (SREs).

Configuration paradigms can change across Prometheus releases, and such cases can only be handled gracefully if all configuration possibilities are controlled. If you use configurations other than those described in this section, your changes will disappear because the `cluster-monitoring-operator` reconciles any differences. The Operator resets everything to the defined state by default and by design.

4.3.1.1. Support considerations for monitoring user-defined projects

The following modifications are explicitly not supported:

- Installing custom Prometheus instances on Red Hat OpenShift Service on AWS

4.3.2. Configuring the monitoring stack

In Red Hat OpenShift Service on AWS, you can configure the stack that monitors workloads for user-defined projects by using the `user-workload-monitoring-config ConfigMap` object. Config maps configure the Cluster Monitoring Operator (CMO), which in turn configures the components of the stack.

**Prerequisites**

- You have access to the cluster as a user with the `dedicated-admin` role.
- You have created the `user-workload-monitoring-config ConfigMap` object.
- You have installed the OpenShift CLI (`oc`).
Procedure

1. Edit the **ConfigMap** object.
   a. Edit the `user-workload-monitoring-config` **ConfigMap** object in the `openshift-user-workload-monitoring` project:

   ```bash
   $ oc -n openshift-user-workload-monitoring edit configmap user-workload-monitoring-config
   ```

   b. Add your configuration under `data.config.yaml` as a key-value pair:

   ```yaml
   <component_name>: <configuration_for_the_component>
   ```

   Substitute `<component>` and `<configuration_for_the_component>` accordingly.

   The following example **ConfigMap** object configures a data retention period and minimum container resource requests for Prometheus. This relates to the Prometheus instance that monitors user-defined projects only:

   ```yaml
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: user-workload-monitoring-config
     namespace: openshift-user-workload-monitoring
   data:
     config.yaml: |
     <component>:
       <configuration_for_the_component>
   ```

   1. Defines the Prometheus component and the subsequent lines define its configuration.
   2. Configures a 24 hour data retention period for the Prometheus instance that monitors user-defined projects.
   3. Defines a minimum resource request of 200 millicores for the Prometheus container.
   4. Defines a minimum pod resource request of 2 GiB of memory for the Prometheus container.

2. Save the file to apply the changes to the **ConfigMap** object.
4. Save the file to apply the changes to the ConfigMap object. The pods affected by the new configuration are restarted automatically.

**WARNING**

When changes are saved to a monitoring config map, the pods and other resources in the related project might be redeployed. The running monitoring processes in that project might also be restarted.

4.3.3. Configurable monitoring components

This table shows the monitoring components you can configure and the keys used to specify the components in the user-workload-monitoring-config ConfigMap objects:

<table>
<thead>
<tr>
<th>Component</th>
<th>user-workload-monitoring-config config map key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prometheus Operator</td>
<td>prometheusOperator</td>
</tr>
<tr>
<td>Prometheus</td>
<td>prometheus</td>
</tr>
<tr>
<td>Thanos Ruler</td>
<td>thanosRuler</td>
</tr>
</tbody>
</table>

4.3.4. Moving monitoring components to different nodes

You can move any of the components that monitor workloads for user-defined projects to specific worker nodes. It is not permitted to move components to control plane or infrastructure nodes.

**Prerequisites**

- You have access to the cluster as a user with the dedicated-admin role.
- You have created the user-workload-monitoring-config ConfigMap object.
- You have installed the OpenShift CLI (oc).

**Procedure**

1. To move a component that monitors user-defined projects, edit the ConfigMap object:
   a. Edit the user-workload-monitoring-config ConfigMap object in the openshift-user-workload-monitoring project:

```
$ oc -n openshift-user-workload-monitoring edit configmap user-workload-monitoring-config
```

   b. Specify the nodeSelector constraint for the component under data.config.yaml:
Substitute `<component>` accordingly and substitute `<node_key>: <node_value>` with the map of key-value pairs that specifies the destination nodes. Often, only a single key-value pair is used.

The component can only run on nodes that have each of the specified key-value pairs as labels. The nodes can have additional labels as well.

**IMPORTANT**

Many of the monitoring components are deployed by using multiple pods across different nodes in the cluster to maintain high availability. When moving monitoring components to labeled nodes, ensure that enough matching nodes are available to maintain resilience for the component. If only one label is specified, ensure that enough nodes contain that label to distribute all of the pods for the component across separate nodes. Alternatively, you can specify multiple labels each relating to individual nodes.

**NOTE**

If monitoring components remain in a Pending state after configuring the nodeSelector constraint, check the pod logs for errors relating to taints and tolerations.

For example, to move monitoring components for user-defined projects to specific worker nodes labeled `nodename: worker1`, `nodename: worker2`, and `nodename: worker2`, use:

```yaml
apiVersion: v1
class: ConfigMap
metadata:
  name: user-workload-monitoring-config
namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    <component>:
      nodeSelector:
        <node_key>: <node_value>
        <node_key>: <node_value>
        ...
```
2. Save the file to apply the changes. The components affected by the new configuration are moved to the new nodes automatically.

**WARNING**

When changes are saved to a monitoring config map, the pods and other resources in the related project might be redeployed. The running monitoring processes in that project might also be restarted.

### 4.3.5. Assigning tolerations to components that monitor user-defined projects

You can assign tolerations to the components that monitor user-defined projects, to enable moving them to tainted worker nodes. Scheduling is not permitted on control plane or infrastructure nodes.

**Prerequisites**

- You have access to the cluster as a user with the `dedicated-admin` role.
- You have created the `user-workload-monitoring-config` ConfigMap object in the `openshift-user-workload-monitoring` namespace.
- The OpenShift CLI (`oc`) is installed.

**Procedure**

1. Edit the ConfigMap object:
   a. Edit the `user-workload-monitoring-config` ConfigMap object in the `openshift-user-workload-monitoring` project:

   ```bash
   $ oc -n openshift-user-workload-monitoring edit configmap user-workload-monitoring-config
   ```
   b. Specify tolerations for the component:

   ```yaml
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: user-workload-monitoring-config
     namespace: openshift-user-workload-monitoring
   data:
     config.yaml: |
     <component>:
       tolerations:
         <tolerationSpecification>
   ```
Substitute `<component>` and `<toleration_specification>` accordingly.

For example, `oc adm taint nodes node1 key1=value1:NoSchedule` adds a taint to `node1` with the key `key1` and the value `value1`. This prevents monitoring components from deploying pods on `node1` unless a toleration is configured for that taint. The following example configures the `thanosRuler` component to tolerate the example taint:

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    thanosRuler:
      tolerations:
        - key: "key1"
          operator: "Equal"
          value: "value1"
          effect: "NoSchedule"
```

2. Save the file to apply the changes. The new component placement configuration is applied automatically.

**WARNING**

When changes are saved to a monitoring config map, the pods and other resources in the related project might be redeployed. The running monitoring processes in that project might also be restarted.

Additional resources

- See the [OpenShift Container Platform documentation](https://docs.openshift.com/) on taints and tolerations
- See the [Kubernetes documentation](https://kubernetes.io/) on taints and tolerations

### 4.3.6. Configuring persistent storage

Running cluster monitoring with persistent storage means that your metrics are stored to a persistent volume (PV) and can survive a pod being restarted or recreated. This is ideal if you require your metrics data to be guarded from data loss. For production environments, it is highly recommended to configure persistent storage. Because of the high IO demands, it is advantageous to use local storage.

**IMPORTANT**

See [Recommended configurable storage technology](https://kubernetes.io/docs/concepts/storage/storage-best-practices/).

#### 4.3.6.1. Persistent storage prerequisites

- Use the block type of storage.
4.3.6.2. Configuring a local persistent volume claim

For monitoring components to use a persistent volume (PV), you must configure a persistent volume claim (PVC).

Prerequisites

- You have access to the cluster as a user with the dedicated-admin role.
- You have created the user-workload-monitoring-config ConfigMap object.
- You have installed the OpenShift CLI (oc).

Procedure

1. To configure a PVC for a component that monitors user-defined projects, edit the ConfigMap object:
   a. Edit the user-workload-monitoring-config ConfigMap object in the openshift-user-workload-monitoring project:

   ```bash
   $ oc -n openshift-user-workload-monitoring edit configmap user-workload-monitoring-config
   ```

   b. Add your PVC configuration for the component under data.config.yaml:

   ```yaml
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: user-workload-monitoring-config
     namespace: openshift-user-workload-monitoring
   data:
     config.yaml: |
     <component>:
       volumeClaimTemplate:
         spec:
           storageClassName: <storage_class>
           resources:
             requests:
               storage: <amount_of_storage>
   ```

   See the Kubernetes documentation on PersistentVolumeClaims for information on how to specify volumeClaimTemplate.

   The following example configures a PVC that claims local persistent storage for the Prometheus instance that monitors user-defined projects:

   ```yaml
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: user-workload-monitoring-config
     namespace: openshift-user-workload-monitoring
   data:
     config.yaml: |
     prometheus:
   ```
In the above example, the storage class created by the Local Storage Operator is called `local-storage`.

The following example configures a PVC that claims local persistent storage for Thanos Ruler:

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    thanosRuler:
      volumeClaimTemplate:
        spec:
          storageClassName: local-storage
          resources:
            requests:
              storage: 40Gi
```

2. Save the file to apply the changes. The pods affected by the new configuration are restarted automatically and the new storage configuration is applied.

**WARNING**

When changes are saved to a monitoring config map, the pods and other resources in the related project might be redeployed. The running monitoring processes in that project might also be restarted.

### 4.3.6.3. Modifying the retention time for Prometheus metrics data

By default, the Red Hat OpenShift Service on AWS monitoring stack configures the retention time for Prometheus data to be 15 days. You can modify the retention time for the Prometheus instance that monitors user-defined projects, to change how soon the data is deleted.

**Prerequisites**

- You have access to the cluster as a user with the `dedicated-admin` role.
- You have created the `user-workload-monitoring-config ConfigMap` object.
- You have installed the OpenShift CLI (`oc`).
To modify the retention time for the Prometheus instance that monitors user-defined projects, edit the ConfigMap object:

a. Edit the `user-workload-monitoring-config` ConfigMap object in the `openshift-user-workload-monitoring` project:

```
$ oc -n openshift-user-workload-monitoring edit configmap user-workload-monitoring-config
```

b. Add your retention time configuration under `data.config.yaml`:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    prometheus:
      retention: <timeSpecification>
```

Substitute `<timeSpecification>` with a number directly followed by `ms` (milliseconds), `s` (seconds), `m` (minutes), `h` (hours), `d` (days), `w` (weeks), or `y` (years).

The following example sets the retention time to 24 hours for the Prometheus instance that monitors user-defined projects:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    prometheus:
      retention: 24h
```

2. Save the file to apply the changes. The pods affected by the new configuration are restarted automatically.

**WARNING**

When changes are saved to a monitoring config map, the pods and other resources in the related project might be redeployed. The running monitoring processes in that project might also be restarted.

**Additional resources**
4.3.7. Controlling the impact of unbound metrics attributes in user-defined projects

Developers can create labels to define attributes for metrics in the form of key-value pairs. The number of potential key-value pairs corresponds to the number of possible values for an attribute. An attribute that has an unlimited number of potential values is called an unbound attribute. For example, a `customer_id` attribute is unbound because it has an infinite number of possible values.

Every assigned key-value pair has a unique time series. The use of many unbound attributes in labels can result in an exponential increase in the number of time series created. This can impact Prometheus performance and can consume a lot of disk space.

A **dedicated-admin** can use the following measure to control the impact of unbound metrics attributes in user-defined projects:

- Limit the number of samples that can be accepted per target scrape in user-defined projects

**NOTE**

Limiting scrape samples can help prevent the issues caused by adding many unbound attributes to labels. Developers can also prevent the underlying cause by limiting the number of unbound attributes that they define for metrics. Using attributes that are bound to a limited set of possible values reduces the number of potential key-value pair combinations.

4.3.7.1. Setting a scrape sample limit for user-defined projects

You can limit the number of samples that can be accepted per target scrape in user-defined projects.

**WARNING**

If you set a sample limit, no further sample data is ingested for that target scrape after the limit is reached.

**Prerequisites**

- You have access to the cluster as a user with the **dedicated-admin** role.
- You have created the **user-workload-monitoring-config ConfigMap** object.
- You have installed the OpenShift CLI (**oc**).

**Procedure**

1. Edit the **user-workload-monitoring-config ConfigMap** object in the **openshift-user-workload-monitoring** project:
Add the `enforcedSampleLimit` configuration to `data.config.yaml` to limit the number of samples that can be accepted per target scrape in user-defined projects:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml:
    prometheus:
      enforcedSampleLimit: 50000
```

A value is required if this parameter is specified. This `enforcedSampleLimit` example limits the number of samples that can be accepted per target scrape in user-defined projects to 50,000.

Save the file to apply the changes. The limit is applied automatically.

### WARNING

When changes are saved to the `user-workload-monitoring-config ConfigMap` object, the pods and other resources in the `openshift-user-workload-monitoring` project might be redeployed. The running monitoring processes in that project might also be restarted.

### Additional resources

- [Determining why Prometheus is consuming a lot of disk space](#) for steps to query which metrics have the highest number of scrape samples

### 4.3.8. Setting log levels for monitoring components

You can configure the log level for Prometheus Operator, Prometheus, and Thanos Ruler.

The following log levels can be applied to each of those components in the `user-workload-monitoring-config ConfigMap` object:

- **debug**: Log debug, informational, warning, and error messages.
- **info**: Log informational, warning, and error messages.
- **warn**: Log warning and error messages only.
- **error**: Log error messages only.

The default log level is **info**.
Prerequisites

- You have access to the cluster as a user with the `dedicated-admin` role.
- You have created the `user-workload-monitoring-config ConfigMap` object.
- You have installed the OpenShift CLI (`oc`).

Procedure

1. Edit the `ConfigMap` object:
   
   a. Edit the `user-workload-monitoring-config ConfigMap` object in the `openshift-user-workload-monitoring` project:

   ```bash
   $ oc -n openshift-user-workload-monitoring edit configmap user-workload-monitoring-config
   
   b. Add `logLevel: <log_level>` for a component under `data.config.yaml`:

   ```yaml
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: user-workload-monitoring-config
     namespace: openshift-user-workload-monitoring
   data:
     config.yaml:
       <component>:  
         logLevel: <log_level>  
   
   1. The monitoring component that you are applying a log level to.
   2. The log level to apply to the component.

2. Save the file to apply the changes. The pods for the component restarts automatically when you apply the log-level change.

   **WARNING**
   
   When changes are saved to a monitoring config map, the pods and other resources in the related project might be redeployed. The running monitoring processes in that project might also be restarted.

3. Confirm that the log level has been applied by reviewing the deployment or pod configuration in the related project. The following example checks the log level in the `prometheus-operator` deployment in the `openshift-user-workload-monitoring` project:

   ```bash
   $ oc -n openshift-user-workload-monitoring get deploy prometheus-operator -o yaml | grep "log-level"
   ```
Example output

```
- --log-level=debug
```

4. Check that the pods for the component are running. The following example lists the status of pods in the `openshift-user-workload-monitoring` project:

```
$ oc -n openshift-user-workload-monitoring get pods
```

**NOTE**

If an unrecognized `loglevel` value is included in the `ConfigMap` object, the pods for the component might not restart successfully.

4.3.9. Next steps

- Managing metrics

4.4. MANAGING METRICS

Red Hat OpenShift Service on AWS collects metrics for cluster components, and you can use Prometheus to query and visualize them.

4.4.1. Understanding metrics

In Red Hat OpenShift Service on AWS, cluster components are monitored by scraping metrics exposed through service endpoints. You can also configure metrics collection for user-defined projects. Metrics enable you to monitor how cluster components and your own workloads are performing.

You can define the metrics that you want to provide for your own workloads by using Prometheus client libraries at the application level.

In Red Hat OpenShift Service on AWS, metrics are exposed through an HTTP service endpoint under the `/metrics` canonical name. You can list all available metrics for a service by running a `curl` query against `http://<endpoint>/metrics`. For instance, you can expose a route to the `prometheus-example-app` example application and then run the following to view all of its available metrics:

```
$ curl http://<example_app_endpoint>/metrics
```

**Example output**

```
# HELP http_requests_total Count of all HTTP requests
# TYPE http_requests_total counter
http_requests_total{code="200",method="get"} 4
http_requests_total{code="404",method="get"} 2
# HELP version Version information about this binary
# TYPE version gauge
version{version="v0.1.0"} 1
```

Additional resources

- See the Prometheus documentation for details on Prometheus client libraries.
4.4.2. Setting up metrics collection for user-defined projects

You can create a ServiceMonitor resource to scrape metrics from a service endpoint in a user-defined project. This assumes that your application uses a Prometheus client library to expose metrics to the /metrics canonical name.

This section describes how to deploy a sample service in a user-defined project and then create a ServiceMonitor resource that defines how that service should be monitored.

4.4.2.1. Deploying a sample service

To test monitoring of a service in a user-defined project, you can deploy a sample service.

Procedure

1. Create a YAML file for the service configuration. In this example, it is called prometheus-example-app.yaml.

2. Add the following deployment and service configuration details to the file:

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: ns1
---
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: prometheus-example-app
    name: prometheus-example-app
    namespace: ns1
spec:
  replicas: 1
  selector:
    matchLabels:
      app: prometheus-example-app
spec:
  template:
    metadata:
      labels:
        app: prometheus-example-app
    spec:
      containers:
      - image: quay.io/brancz/prometheus-example-app:v0.2.0
        imagePullPolicy: IfNotPresent
        name: prometheus-example-app
---
apiVersion: v1
kind: Service
metadata:
  labels:
    app: prometheus-example-app
spec:
  namespace: ns1
  ports:
```
This configuration deploys a service named `prometheus-example-app` in the user-defined `ns1` project. This service exposes the custom `version` metric.

3. Apply the configuration to the cluster:

```
$ oc apply -f prometheus-example-app.yaml
```

It takes some time to deploy the service.

4. You can check that the pod is running:

```
$ oc -n ns1 get pod
```

Example output

<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>prometheus-example-app-7857545cb7-sbgwq</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>81m</td>
</tr>
</tbody>
</table>

**4.4.2.2. Specifying how a service is monitored**

To use the metrics exposed by your service, you must configure Red Hat OpenShift Service on AWS monitoring to scrape metrics from the `/metrics` endpoint. You can do this using a `ServiceMonitor` custom resource definition (CRD) that specifies how a service should be monitored, or a `PodMonitor` CRD that specifies how a pod should be monitored. The former requires a `Service` object, while the latter does not, allowing Prometheus to directly scrape metrics from the metrics endpoint exposed by a pod.

**NOTE**

In Red Hat OpenShift Service on AWS, you can use the `tlsConfig` property for a `ServiceMonitor` resource to specify the TLS configuration to use when scraping metrics from an endpoint. The `tlsConfig` property is not yet available for `PodMonitor` resources. If you need to use a TLS configuration when scraping metrics, you must use the `ServiceMonitor` resource.

This procedure shows you how to create a `ServiceMonitor` resource for a service in a user-defined project.

**Prerequisites**

- You have access to the cluster as a user with the `dedicated-admin` role or the `monitoring-edit` role.
- For this example, you have deployed the `prometheus-example-app` sample service in the `ns1` project.
Procedure

1. Create a YAML file for the ServiceMonitor resource configuration. In this example, the file is called example-app-service-monitor.yaml.

2. Add the following ServiceMonitor resource configuration details:

```yaml
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  labels:
    k8s-app: prometheus-example-monitor
name: prometheus-example-monitor
namespace: ns1
spec:
  endpoints:
  - interval: 30s
    port: web
    scheme: http
  selector:
    matchLabels:
      app: prometheus-example-app
```

This defines a ServiceMonitor resource that scrapes the metrics exposed by the prometheus-example-app sample service, which includes the version metric.

3. Apply the configuration to the cluster:

```
$ oc apply -f example-app-service-monitor.yaml
```

It takes some time to deploy the ServiceMonitor resource.

4. You can check that the ServiceMonitor resource is running:

```
$ oc -n ns1 get servicemonitor
```

Example output

```
NAME                        AGE
prometheus-example-monitor   81m
```

Additional resources

- Accessing monitoring for user-defined projects.

4.4.3. Querying metrics

The OpenShift monitoring dashboard lets you run Prometheus Query Language (PromQL) queries to examine metrics visualized on a plot. This functionality provides information about the state of a cluster and any user-defined projects that you are monitoring.

As a dedicated-admin, you can query one or more namespaces at a time for metrics about user-defined projects.
As a developer, you must specify a project name when querying metrics. You must have the required privileges to view metrics for the selected project.

4.4.3.1. Querying metrics for all projects as an administrator

As a **dedicated-admin** or as a user with view permissions for all projects, you can access metrics for all default Red Hat OpenShift Service on AWS and user-defined projects in the Metrics UI.

**NOTE**

Only dedicated administrators have access to the third-party UIs provided with Red Hat OpenShift Service on AWS Monitoring.

**Prerequisites**

- You have access to the cluster as a user with the **dedicated-admin** role or with view permissions for all projects.

**Procedure**

1. From the **Administrator** perspective in the OpenShift web console, select **Observe → Metrics**.

2. Select **Insert Metric at Cursor** to view a list of predefined queries.

3. To create a custom query, add your Prometheus Query Language (PromQL) query to the **Expression** field.

4. To add multiple queries, select **Add Query**.

5. To delete a query, select **next to the query, then choose Delete query**.

6. To disable a query from being run, select **next to the query and choose Disable query**.

7. Select **Run Queries** to run the queries that you have created. The metrics from the queries are visualized on the plot. If a query is invalid, the UI shows an error message.

**NOTE**

Queries that operate on large amounts of data might time out or overload the browser when drawing time series graphs. To avoid this, select **Hide graph** and calibrate your query using only the metrics table. Then, after finding a feasible query, enable the plot to draw the graphs.

8. Optional: The page URL now contains the queries you ran. To use this set of queries again in the future, save this URL.

**Additional resources**

- See the [Prometheus query documentation](#) for more information about creating PromQL queries.
4.4.3.2. Querying metrics for user-defined projects as a developer

You can access metrics for a user-defined project as a developer or as a user with view permissions for the project.

In the Developer perspective, the Metrics UI includes some predefined CPU, memory, bandwidth, and network packet queries for the selected project. You can also run custom Prometheus Query Language (PromQL) queries for CPU, memory, bandwidth, network packet and application metrics for the project.

**NOTE**

Developers can only use the Developer perspective and not the Administrator perspective. As a developer you can only query metrics for one project at a time. Developers cannot access the third-party UIs provided with Red Hat OpenShift Service on AWS monitoring.

**Prerequisites**

- You have access to the cluster as a developer or as a user with view permissions for the project that you are viewing metrics for.
- You have enabled monitoring for user-defined projects.
- You have deployed a service in a user-defined project.
- You have created a ServiceMonitor custom resource definition (CRD) for the service to define how the service is monitored.

**Procedure**

1. From the Developer perspective in the Red Hat OpenShift Service on AWS web console, select Observe → Metrics.
2. Select the project that you want to view metrics for in the Project: list.
3. Choose a query from the Select Query list, or run a custom PromQL query by selecting Show PromQL.

**NOTE**

In the Developer perspective, you can only run one query at a time.

**Additional resources**

- See the Prometheus query documentation for more information about creating PromQL queries.
- See Querying metrics for user-defined projects as a developer for details on accessing non-cluster metrics as a developer or a privileged user

4.4.3.3. Exploring the visualized metrics

After running the queries, the metrics are displayed on an interactive plot. The X-axis in the plot represents time and the Y-axis represents metrics values. Each metric is shown as a colored line on the graph. You can manipulate the plot interactively and explore the metrics.
Procedure

In the Administrator perspective:

1. Initially, all metrics from all enabled queries are shown on the plot. You can select which metrics are shown.

   **NOTE**
   
   By default, the query table shows an expanded view that lists every metric and its current value. You can select "\[
   ˅\]
   to minimize the expanded view for a query.

   - To hide all metrics from a query, click for the query and click Hide all series.
   - To hide a specific metric, go to the query table and click the colored square near the metric name.

2. To zoom into the plot and change the time range, do one of the following:
   - Visually select the time range by clicking and dragging on the plot horizontally.
   - Use the menu in the left upper corner to select the time range.

3. To reset the time range, select Reset Zoom.

4. To display outputs for all queries at a specific point in time, hover over the plot at that point. The query outputs appear in a pop-up box.

5. To hide the plot, select Hide Graph.

In the Developer perspective:

1. To zoom into the plot and change the time range, do one of the following:
   - Visually select the time range by clicking and dragging on the plot horizontally.
   - Use the menu in the left upper corner to select the time range.

2. To reset the time range, select Reset Zoom.

3. To display outputs for all queries at a specific point in time, hover over the plot at that point. The query outputs appear in a pop-up box.

Additional resources

- See the Querying metrics section on using the PromQL interface
- Troubleshooting monitoring issues

4.4.4. Next steps

- Alerts

4.5. ALERTS
Alerts for monitoring workloads in user-defined projects are not currently supported in this Red Hat OpenShift Service on AWS.

4.5.1. Next steps

- Reviewing monitoring dashboards

4.6. REVIEWING MONITORING DASHBOARDS

Red Hat OpenShift Service on AWS provides monitoring dashboards that help you understand the state of user-defined projects.

In the Developer perspective, you can access dashboards that provide the following statistics for a selected project:

- CPU usage
- Memory usage
- Bandwidth information
- Packet rate information

Figure 4.1. Example dashboard in the Developer perspective

NOTE

In the Developer perspective, you can view dashboards for only one project at a time.

4.6.1. Reviewing monitoring dashboards as a developer
In the Developer perspective, you can view dashboards relating to a selected project. You must have access to monitor a project to view dashboard information for it.

**Prerequisites**

- You have access to the cluster as a dedicated-admin or as a user with view permissions for the project that you are viewing the dashboard for.

**Procedure**

1. In the Developer perspective in the Red Hat OpenShift Service on AWS web console, navigate to Observe → Dashboard.
2. Choose a project in the Project: list.
3. Choose a workload in the All Workloads list.
4. Optional: Select a time range for the graphs in the Time Range list.
5. Optional: Select a Refresh Interval
6. Hover over each of the graphs within a dashboard to display detailed information about specific items.

**4.6.2. Next steps**

- Troubleshooting monitoring issues

**4.7. TROUBLESHOOTING MONITORING ISSUES**

Find troubleshooting steps for common monitoring issues with user-defined projects.

**4.7.1. Determining why user-defined project metrics are unavailable**

If metrics are not displaying when monitoring user-defined projects, follow these steps to troubleshoot the issue.

**Procedure**

1. Query the metric name and verify that the project is correct:
   a. From the Developer perspective in the OpenShift Container Platform web console, select Observe → Metrics.
   b. Select the project that you want to view metrics for in the Project: list.
   c. Choose a query from the Select Query list, or run a custom PromQL query by selecting Show PromQL. The Select Query pane shows the metric names.

   Queries must be done on a per-project basis. The metrics that are shown relate to the project that you have selected.

   2. Verify that the pod that you want metrics from is actively serving metrics. Run the following `oc exec` command into a pod to target the podIP, port, and /metrics.
NOTE

You must run the command on a pod that has `curl` installed.

The following example output shows a result with a valid version metric.

**Example output**

```bash
$ oc exec <sample_pod> -n <sample_namespace> -- curl <target_pod_IP>:<port>/metrics

% Total  % Received % Xferd  Average Speed   Time    Time     Time  Current
Dload  Upload   Total   Spent    Left  Speed

# HELP version Version information about this binary-- --:--:-- --:--:-- 0
# TYPE version gauge
version{version="v0.1.0"} 1
100 102 100 102 0 0 51000 0 --:--:-- --:--:-- --:--:-- 51000
```

An invalid output indicates that there is a problem with the corresponding application.

3. If you are using a **PodMonitor** CRD, verify that the **PodMonitor** CRD is configured to point to the correct pods using label matching. For more information, see the Prometheus Operator documentation.

4. If you are using a **ServiceMonitor** CRD, and if the `/metrics` endpoint of the pod is showing metric data, follow these steps to verify the configuration:
   a. Verify that the service is pointed to the correct `/metrics` endpoint. The service labels in this output must match the services monitor labels and the `/metrics` endpoint defined by the service in the subsequent steps.

```bash
$ oc get service
```

**Example output**

```yaml
apiVersion: v1
kind: Service
metadata:
  labels: 1
    app: prometheus-example-app
    name: prometheus-example-app
    namespace: ns1
spec:
  ports:
    - port: 8080
      protocol: TCP
      targetPort: 8080
      name: web
  selector:
    app: prometheus-example-app
    type: ClusterIP

1 Specifies that this is a service API.
```
2. Specifies the labels that are being used for this service.

b. Query the `serviceIP`, `port`, and `/metrics` endpoints to see if the same metrics from the `curl` command you ran on the pod previously:

i. Run the following command to find the service IP:

```
$ oc get service -n <target_namespace>
```

ii. Query the `/metrics` endpoint:

```
$ oc exec <sample_pod> -n <sample_namespace> -- curl <service_IP>:<port>/metrics
```

Valid metrics are returned in the following example.

**Example output**

```
% Total  % Received % Xferd  Average Speed Time      Time  Time     Time  Current
Dload  Upload Total Spent    Left  Speed
100   102  100   102    0     0  51000      0 --:--:-- --:--:-- --:--:--   99k
# HELP version Version information about this binary
# TYPE version gauge
version{version="v0.1.0"} 1
```

c. Use label matching to verify that the `ServiceMonitor` object is configured to point to the desired service. To do this, compare the `Service` object from the `oc get service` output to the `ServiceMonitor` object from the `oc get servicemonitor` output. The labels must match for the metrics to be displayed.

For example, from the previous steps, notice how the `Service` object has the `app: prometheus-example-app` label and the `ServiceMonitor` object has the same `app: prometheus-example-app` match label.

5. If everything looks valid and the metrics are still unavailable, please contact the support team for further help.