OpenShift Dedicated 4

Cluster administration

Configuring OpenShift Dedicated clusters

Last Updated: 2022-08-03

OpenShift Dedicated 4 Cluster administration

Configuring OpenShift Dedicated clusters

Legal Notice

Abstract

This document provides information about configuring OpenShift Dedicated clusters.

Table of Contents

CHAPTER 1. MANAGING ADMINISTRATION ROLES AND USERS

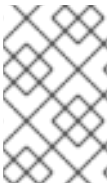
1.1. UNDERSTANDING ADMINISTRATION ROLES

1.1.1. The cluster-admin role

As an administrator of an OpenShift Dedicated cluster with Customer Cloud Subscriptions (CCS), you have access to the **cluster-admin** role. The user who created the cluster can add the **cluster-admin** user role to an account to have the maximum administrator privileges. These privileges are not automatically assigned to your user account when you create the cluster. While logged in to an account with the cluster-admin role, users have mostly unrestricted access to control and configure the cluster. There are some configurations that are blocked with webhooks to prevent destabilizing the cluster, or because they are managed in OpenShift Cluster Manager and any in-cluster changes would be overwritten. Usage of the cluster-admin role is subject to the restrictions listed in your Appendix 4 agreement with Red Hat. As a best practice, limit the number of **cluster-admin** users to as few as possible.

1.1.2. The dedicated-admin role

As an administrator of an OpenShift Dedicated cluster, your account has additional permissions and access to all user-created projects in your organization's cluster. While logged in to an account with the **dedicated-admin** role, the developer CLI commands (under the **oc** command) allow you increased visibility and management capabilities over objects across projects, while the administrator CLI commands (under the **oc adm** command) allow you to complete additional operations.



NOTE

While your account does have these increased permissions, the actual cluster maintenance and host configuration is still performed by the OpenShift Operations Team.

1.2. MANAGING OPENSHIFT DEDICATED ADMINISTRATORS

Administrator roles are managed using a **cluster-admin** or **dedicated-admin** group on the cluster. Existing members of this group can edit membership through OpenShift Cluster Manager.

1.2.1. Adding a user

Procedure

1. Navigate to the **Cluster Details** page and **Access Control** tab.
2. Click the **Add user** button (first user only).
3. Enter the user name and select the group.
4. Click the **Add** button.

NOTE

Adding a user to the **cluster-admin** group can take several minutes to complete.

NOTE

Existing **dedicated-admin** users cannot also be added to the **cluster-admin** group. You must first remove the user from the **dedicated-admin** group before adding the user to the **cluster-admin** group.

1.2.2. Removing a user

Procedure

1. Navigate to the **Cluster Details** page and **Access Control** tab.
2. Click the Options menu  to the right of the user and group combination and click **Delete**.

CHAPTER 2. CONFIGURING PRIVATE CONNECTIONS

2.1. CONFIGURING PRIVATE CONNECTIONS FOR AWS

2.1.1. Understanding AWS cloud infrastructure access

NOTE

AWS cloud infrastructure access does not apply to the Customer Cloud Subscription (CCS) infrastructure type that is chosen when you create a cluster because CCS clusters are deployed onto your account.

Amazon Web Services (AWS) infrastructure access permits Customer Portal Organization Administrators and cluster owners to enable AWS Identity and Access Management (IAM) users to have federated access to the AWS Management Console for their OpenShift Dedicated cluster. AWS access can be granted for customer AWS users, and private cluster access can be implemented to suit the needs of your OpenShift Dedicated environment.

1. Get started with configuring AWS infrastructure access for your OpenShift Dedicated cluster. By creating an AWS user and account and providing that user with access to the OpenShift Dedicated AWS account.

2. After you have access to the OpenShift Dedicated AWS account, 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.

After configuring your cloud infrastructure access, learn more about Configuring a private cluster.

2.1.2. Configuring AWS infrastructure access

Amazon Web Services (AWS) infrastructure access allows Customer Portal Organization Administrators and cluster owners to enable AWS Identity and Access Management (IAM) users to have federated access to the AWS Management Console for their OpenShift Dedicated cluster. Administrators can select between **Network Management** or **Read-only** access options.

Prerequisites

- An AWS account with IAM permissions.

Procedure

1. Log in to your AWS account. If necessary, you can create a new AWS account by following the AWS documentation.

2. Create an IAM user with **STS:AllowAssumeRole** permissions within the AWS account.

a. Open the IAM dashboard of the AWS Management Console.

b. In the **Policies** section, click **Create Policy**.

c. Select the **JSON** tab and replace the existing text with the following:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "*"
    }
  ]
}
```

d. Click **Next:Tags**.

e. Optional: Add tags. Click **Next:Review**

f. Provide an appropriate name and description, then click **Create Policy**.

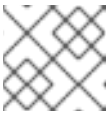
g. In the **Users** section, click **Add user**.

h. Provide an appropriate user name.

i. Select **AWS Management Console access** as the AWS access type.

j. Adjust the password requirements as necessary for your organization, then click **Next:Permissions**.

k. Click the **Attach existing policies directly** option. Search for and check the policy created in previous steps.



NOTE

It is not recommended to set a permissions boundary.

l. Click **Next: Tags**, then click **Next: Review**. Confirm the configuration is correct.

m. Click **Create user**, a success page appears.

n. Gather the IAM user's Amazon Resource Name (ARN). The ARN will have the following format: **arn:aws:iam::000111222333:user/username**. Click **Close**.

3. Open OpenShift Cluster Manager in your browser and select the cluster you want to allow AWS infrastructure access.

4. Select the **Access control** tab, and scroll to the **AWS Infrastructure Access** section.

5. Paste the **AWS IAM ARN** and select **Network Management** or **Read-only** permissions, then click **Grant role**.

6. Copy the **AWS OSD console URL** to your clipboard.

7. Sign in to your AWS account with your Account ID or alias, IAM user name, and password.
8. In a new browser tab, paste the AWS OSD Console URL that will be used to route to the AWS Switch Role page.
9. Your account number and role will be filled in already. Choose a display name if necessary, then click **Switch Role**.

Verification

- You now see **VPC** under **Recently visited services**.

2.1.3. Configuring AWS VPC peering

A Virtual Private Cloud (VPC) peering connection is a networking connection between two VPCs that enables you to route traffic between them using private IPv4 addresses or IPv6 addresses. You can configure an Amazon Web Services (AWS) VPC containing an OpenShift Dedicated cluster to peer with another AWS VPC network.

WARNING

Private clusters cannot be fully deleted by Red Hat OpenShift Cluster Manager if the VPC the cluster is installed in is peered.

AWS supports inter-region VPC peering between all commercial regions excluding China.

Prerequisites

- Gather the following information about the Customer VPC that is 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 OpenShift Dedicated 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. Initiate the VPC peering request.
2. Accept the VPC peering request.

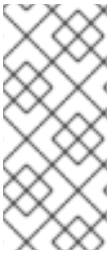
3. Update your Route tables for the VPC peering connection .

Additional resources

- For more information and troubleshooting help, see the AWS VPC guide.

2.1.4. Configuring an AWS VPN

You can configure an Amazon Web Services (AWS) OpenShift Dedicated cluster to use a customer's on-site hardware Virtual Private Network (VPN) device. By default, instances that you launch into an AWS Virtual Private Cloud (VPC) cannot communicate with your own (remote) network. You can enable access to your remote network from your VPC by creating an AWS Site-to-Site VPN connection, and configuring routing to pass traffic through the connection.



NOTE

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

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.

Prerequisites

- Hardware VPN gateway device model and software version, for example Cisco ASA running version 8.3. See the AWS documentation 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.

Procedure

1. Create a customer gateway to configure the VPN connection.
2. If you do not already have a Virtual Private Gateway attached to the intended VPC, create and attach a Virtual Private Gateway.
3. Configure routing and enable VPN route propagation .
4. Update your security group .
5. Establish the Site-to-Site VPN connection .

NOTE

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

Additional resources

- For more information and troubleshooting help, see the AWS VPN guide.

2.1.5. Configuring AWS Direct Connect

Amazon Web Services (AWS) Direct Connect 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 Virtual Private Cloud (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 your AWS account.

If you have an existing DXGateway connected to one or more existing VGWs, the process involves your 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.

Prerequisites

- Confirm the CIDR range of the OpenShift Dedicated 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. Create a VIF or view your existing VIFs to determine the type of direct connection you need to create.
2. Create your gateway.
   a. If the Direct Connect VIF type is **Private**, create a virtual private gateway.
   b. If the Direct Connect VIF is **Public**, create a Direct Connect gateway.
3. If you have an existing gateway you want to use, create an association proposal and send the proposal to the DXGateway owner for approval.

WARNING

When connecting to an existing DXGateway, you are responsible for the costs.

Additional resources

- For more information and troubleshooting help, see the AWS Direct Connect guide.

2.2. CONFIGURING A PRIVATE CLUSTER

An OpenShift Dedicated 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.

OpenShift Dedicated administrators can choose between public and private cluster configuration from within **OpenShift Cluster Manager**. Privacy settings can be configured during cluster creation or after a cluster is established.

2.2.1. Enabling a private cluster during cluster creation

You can enable private cluster settings when creating a new cluster.

Prerequisites

- The following private connections must be configured to allow private access:
  - VPC Peering
  - Cloud VPN
  - DirectConnect (AWS only)
  - TransitGateway (AWS only)
  - Cloud Interconnect (GCP only)

Procedure

1. Log in to OpenShift Cluster Manager.
2. Click **Create cluster → OpenShift Dedicated → Create cluster**.
3. Configure your cluster details.
4. When selecting your preferred network configuration, select **Advanced**.
5. Select **Private**.

WARNING

When set to **Private**, you cannot access your cluster unless you have configured the private connections in your cloud provider as outlined in the prerequisites.

6. Click **Create cluster**. The cluster creation process begins and takes about 30-40 minutes to complete.

Verification

- The **Installing cluster** heading, under the **Overview** tab, indicates that the cluster is installing and you can view the installation logs from this heading. The **Status** indicator under the **Details** heading indicates when your cluster is **Ready** for use.

2.2.2. Enabling an existing cluster to be private

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

Prerequisites

- The following private connections must be configured to allow private access:
  - VPC Peering
  - Cloud VPN
  - DirectConnect (AWS only)
  - TransitGateway (AWS only)
  - Cloud Interconnect (GCP only)

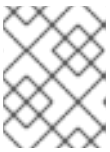
Procedure

1. Log in to OpenShift Cluster Manager.
2. Select the public cluster you would like to make private.
3. On the **Networking** tab, select **Make API private** under **Control Plane API endpoint**

WARNING

When set to **Private**, you cannot access your cluster unless you have configured the private connections in your cloud provider as outlined in the prerequisites.

4. Click **Change settings**.



NOTE

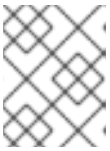
Transitioning your cluster between private and public can take several minutes to complete.

2.2.3. Enabling an existing private cluster to be public

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

Procedure

1. Log in to OpenShift Cluster Manager.
2. Select the private cluster you would like to make public.
3. On the **Networking** tab, deselect **Make API private** under **Control Plane API endpoint**
4. Click **Change settings**.



NOTE

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

CHAPTER 3. NODES

3.1. ABOUT MACHINE POOLS

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

You cannot configure the per-pod PID limit on any OpenShift Dedicated cluster.

3.1.1. Machines

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

3.1.2. 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.

3.1.3. 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.

3.1.4. 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.

3.1.5. Additional resources

- About autoscaling

3.2. MANAGING COMPUTE NODES

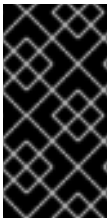
This document describes how to manage compute (also known as worker) nodes with OpenShift Dedicated.

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.

3.2.1. Creating a machine pool

A default machine pool is created when you install an OpenShift Dedicated cluster. After installation, you can create additional machine pools for your cluster by using OpenShift Cluster Manager.



IMPORTANT

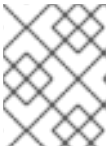
The compute (also known as worker) node instance types, autoscaling options, and node counts that are available to you depend on your OpenShift Dedicated subscriptions, resource quotas and deployment scenario. For more information, contact your sales representative or Red Hat support.

Prerequisites

- You created an OpenShift Dedicated cluster.

Procedure

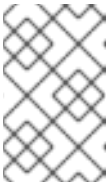
1. Navigate to OpenShift Cluster Manager and select your cluster.
2. Under the **Machine pools** tab, click **Add machine pool**.
3. Add a **Machine pool name**.
4. Select a **Worker node instance type** from the drop-down menu. The instance type defines the vCPU and memory allocation for each compute 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:
   a. Select **Enable autoscaling** to automatically scale the number of machines in your machine pool to meet the deployment needs.



NOTE

The **Enable autoscaling** option is only available for OpenShift Dedicated if you have the **capability.cluster.autoscale_clusters** subscription. For more information, contact your sales representative or Red Hat support.

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

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

NOTE

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

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

- If you deployed your cluster using a single availability zone, select a **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.

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

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

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

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

d. For each taint, select an **Effect** from the drop-down menu. Available options include **NoSchedule**, **PreferNoSchedule**, and **NoExecute**.

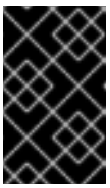
NOTE

Alternatively, you can add the node labels and taints after you create the machine pool.

8. Optional: If you deployed OpenShift Dedicated on AWS using the Customer Cloud Subscription (CCS) model, use Amazon EC2 Spot Instances if you want to configure your machine pool to deploy machines as non-guaranteed AWS Spot Instances:

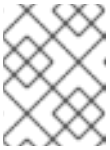
a. Select **Use Amazon EC2 Spot Instances**.

b. Leave **Use On-Demand instance price** selected to use the on-demand instance price. Alternatively, select **Set maximum price** to define a maximum hourly price for a Spot Instance.
For more information about Amazon EC2 Spot Instances, see the AWS documentation.



IMPORTANT

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



NOTE

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

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.

3.2.2. 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 created an OpenShift Dedicated cluster.
- You have an existing machine pool.

Procedure

1. Navigate to OpenShift Cluster Manager and select your cluster.
2. Under the **Machine pools** tab, click the options menu  for the machine pool that you want to scale.
3. Select **Scale**.
4. Specify the node count:
   - If you deployed your cluster using a single availability zone, specify the **Node count** in the drop-down menu.
   - If you deployed your cluster using multiple availability zones, specify the **Node count per zone** in the drop-down menu.

NOTE

Your subscription determines the number of nodes that you can select.

5. Click **Apply** to scale the machine pool.

Verification

- Under the **Machine pools** tab, verify that the **Node count** for your machine pool is as expected.

3.2.3. 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 created an OpenShift Dedicated cluster.
- You have an existing machine pool.

Procedure

1. Navigate to OpenShift Cluster Manager and select your cluster.
2. Under the **Machine pools** tab, click the options menu  for the machine pool that you want to add a label to.
3. Select **Edit labels**.
4. If you have existing labels in the machine pool that you want to remove, select **x** next to the label to delete it.
5. Add a label using the format **<key>=<value>** and press enter. For example, add **app=db** and then press enter. If the format is correct, the key value pair is then highlighted.
6. Repeat the previous step if you want to add additional labels.
7. Click **Save** to apply the labels to the machine pool.

Verification

1. Under the **Machine pools** tab, select **>** next to your machine pool to expand the view.
2. Verify that your labels are listed under **Labels** in the expanded view.

3.2.4. 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 created an OpenShift Dedicated cluster.
- You have an existing machine pool.

Procedure

1. Navigate to OpenShift Cluster Manager and select your cluster.
2. Under the **Machine pools** tab, click the options menu  for the machine pool that you want to add a taint to.
3. Select **Edit taints**.
4. Add **Key** and **Value** entries for your taint.
5. Select an **Effect** for your taint from the drop-down menu. Available options include **NoSchedule**, **PreferNoSchedule**, and **NoExecute**.
6. Select **Add taint** if you want to add more taints to the machine pool.
7. Click **Save** to apply the taints to the machine pool.

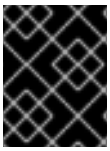
Verification

1. Under the **Machine pools** tab, select > next to your machine pool to expand the view.
2. Verify that your taints are listed under **Taints** in the expanded view.

3.2.5. Additional resources

- About machine pools
- Enabling autoscaling
- Disabling autoscaling
- OpenShift Dedicated service definition

3.3. ABOUT AUTOSCALING NODES ON A CLUSTER



IMPORTANT

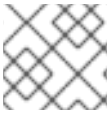
Autoscaling is available only on clusters that were purchased through the Red Hat Marketplace.

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.

3.3.1. 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.

3.3.2. 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.

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

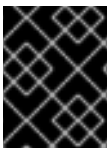
Procedure

1. From OpenShift Cluster Manager, navigate to the **Clusters** page and select the cluster with autoscaling that must be disabled.
2. On the selected cluster, select the **Machine pools** tab.
3. Click the Options menu  at the end of the machine pool with autoscaling and select **Scale**.

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

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

Applying autoscaling to an OpenShift Dedicated cluster involves deploying a cluster autoscaler and then deploying machine autoscalers for each machine type in your cluster.



IMPORTANT

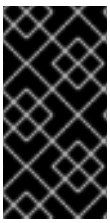
You can configure the cluster autoscaler only in clusters where the Machine API is operational.

3.3.3. About the cluster autoscaler

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

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

The cluster autoscaler computes the total memory and CPU on all nodes the cluster, even though it does not manage the control plane nodes. These values are not single-machine oriented. They are an aggregation of all the resources in the entire cluster. For example, if you set the maximum memory resource limit, the cluster autoscaler includes all the nodes in the cluster when calculating the current memory usage. That calculation is then used to determine if the cluster autoscaler has the capacity to add more worker resources.



IMPORTANT

Ensure that the **maxNodesTotal** value in the **ClusterAutoscaler** resource definition that you create is large enough to account for the total possible number of machines in your cluster. This value must encompass the number of control plane machines and the possible number of compute machines that you might scale to.

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

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

- The cluster autoscaler can move all pods running on the node to the other nodes.

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

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

- Pods with restrictive pod disruption budgets (PDBs).

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

- Kube-system pods that do not have a PDB or have a PDB that is too restrictive.
- Pods that are not backed by a controller object such as a deployment, replica set, or stateful set.
- Pods with local storage.
- Pods that cannot be moved elsewhere because of a lack of resources, incompatible node selectors or affinity, matching anti-affinity, and so on.
- Unless they also have a **"cluster-autoscaler.kubernetes.io/safe-to-evict": "true"** annotation, pods that have a **"cluster-autoscaler.kubernetes.io/safe-to-evict": "false"** annotation.

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

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

- Do not modify the nodes that are in autoscaled node groups directly. All nodes within the same node group have the same capacity and labels and run the same system pods.
- Specify requests for your pods.
- If you have to prevent pods from being deleted too quickly, configure appropriate PDBs.
- Confirm that your cloud provider quota is large enough to support the maximum node pools that you configure.
- Do not run additional node group autoscalers, especially the ones offered by your cloud provider.

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

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

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

3.3.4. Additional resources

- About machinepools

CHAPTER 4. LOGGING

4.1. ACCESSING THE SERVICE LOGS FOR OPENSHIFT DEDICATED CLUSTERS

You can view the service logs for your OpenShift Dedicated 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 an OpenShift Dedicated cluster. Subscribed users receive emails about cluster events that require customer action, known cluster incidents, upgrade maintenance, and other topics.

4.1.1. Viewing the service logs by using OpenShift Cluster Manager

You can view the service logs for an OpenShift Dedicated cluster by using Red Hat OpenShift Cluster Manager.

Prerequisites

- You have installed an OpenShift Dedicated 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.

4.1.2. Adding cluster notification contacts

You can add notification contacts for your OpenShift Dedicated 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.

CHAPTER 5. MONITORING USER-DEFINED PROJECTS

5.1. UNDERSTANDING THE MONITORING STACK

In OpenShift Dedicated, 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

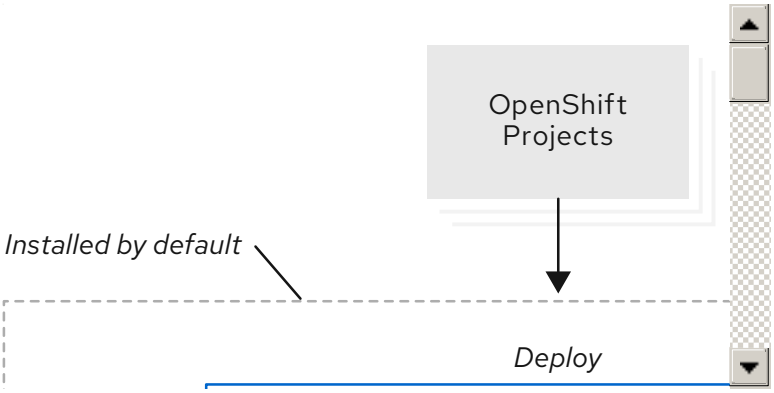
Follow the instructions in this document carefully to configure a supported Prometheus instance for monitoring user-defined projects. Custom Prometheus instances are not supported by OpenShift Dedicated.

5.1.1. Understanding the monitoring stack

The OpenShift Dedicated 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 an OpenShift Dedicated 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.



5.1.1.1. Components for monitoring user-defined projects

OpenShift Dedicated 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 5.1. Components for monitoring user-defined projects

Component	Description

Component	Description
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 OpenShift Dedicated 4, Thanos Ruler provides rule and alerting evaluation for the monitoring of user-defined projects.

All of these components are monitored by the stack and are automatically updated when OpenShift Dedicated is updated.

5.1.1.2. Monitoring targets for user-defined projects

Monitoring is enabled by default for OpenShift Dedicated user-defined projects. You can monitor:

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

5.1.2. Additional resources

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

5.1.3. Next steps

- Accessing monitoring for user-defined projects

5.2. ACCESSING MONITORING FOR USER-DEFINED PROJECTS

When you install an OpenShift Dedicated cluster, monitoring for user-defined projects is enabled by default. With monitoring for user-defined projects enabled, you can monitor your own OpenShift Dedicated 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.

5.2.1. Next steps

- Configuring the monitoring stack

5.3. CONFIGURING THE MONITORING STACK

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

5.3.1. Maintenance and support for monitoring

The supported way of configuring OpenShift Dedicated 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.

5.3.1.1. Support considerations for monitoring user-defined projects

The following modifications are explicitly not supported:

- Installing custom Prometheus instances on OpenShift Dedicated

5.3.2. Configuring the monitoring stack

In OpenShift Dedicated, 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:

   ```
   $ 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 **<component_name>: <component_configuration>**:

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

   ```
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: user-workload-monitoring-config
     namespace: openshift-user-workload-monitoring
   data:
     config.yaml: |
       prometheus: 1
         retention: 24h 2
         resources:
           requests:
             cpu: 200m 3
             memory: 2Gi 4
   ```

   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. 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.

5.3.3. 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 5.2. Configurable monitoring components

Component	user-workload-monitoring-config config map key
Prometheus Operator	**prometheusOperator**
Prometheus	**prometheus**
Thanos Ruler	**thanosRuler**

5.3.4. 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:

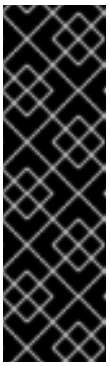
```
$ oc -n openshift-user-workload-monitoring edit configmap user-workload-monitoring-config
```

   b. Specify the **nodeSelector** constraint for the component under **data.config.yaml**:

```
apiVersion: v1
kind: 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>
        <...>
```

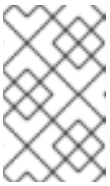
Substitute **<component>** accordingly and substitute **<node_key>: <node_value>** with the map of key-value pairs that specifies the destination nodes. Often, only a single key-value pair is used.

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:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    prometheusOperator:
      nodeSelector:
        nodename: worker1
    prometheus:
      nodeSelector:
        nodename: worker1
        nodename: worker2
    thanosRuler:
```

```
    nodeSelector:
      nodename: worker1
      nodename: worker2
```

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.

Additional resources

- Understanding how to update labels on nodes
- Placing pods on specific nodes using node selectors
- See the Kubernetes documentation for details on the **nodeSelector** constraint

5.3.5. 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:

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

   b. Specify **tolerations** for the component:

```
apiVersion: v1
kind: ConfigMap
metadata:
```

```
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    <component>:
      tolerations:
        <toleration_specification>
```

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:

```
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 on taints and tolerations
- See the Kubernetes documentation on taints and tolerations

5.3.6. 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 .

5.3.6.1. Persistent storage prerequisites

- Use the block type of storage.

5.3.6.2. 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:

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

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

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

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:

```
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.

5.3.6.3. Modifying the retention time for Prometheus metrics data

By default, the OpenShift Dedicated 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**).

Procedure

1. 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: <time_specification>
```

   Substitute **<time_specification>** 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

- Understanding persistent storage
- Optimizing storage

5.3.7. 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.

5.3.7.1. 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:

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

2. 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 1
```

1 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.

3. 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

5.3.8. 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:

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

   b. Add **logLevel: <log_level>** for a component under **data.config.yaml**:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    <component>: 1
      logLevel: <log_level> 2
```

   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:

```
$ 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.

5.3.9. Next steps

- Managing metrics

5.4. MANAGING METRICS

You can collect metrics to monitor how cluster components and your own workloads are performing.

5.4.1. Understanding metrics

In OpenShift Dedicated, 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 OpenShift Dedicated, 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.

5.4.2. 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.

5.4.2.1. 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:

```
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
  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
  name: prometheus-example-app
  namespace: ns1
spec:
  ports:
  - port: 8080
    protocol: TCP
    targetPort: 8080
    name: web
  selector:
    app: prometheus-example-app
  type: ClusterIP
```

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

```
NAME                                  READY     STATUS    RESTARTS   AGE
prometheus-example-app-7857545cb7-sbgwq   1/1       Running   0          81m
```

5.4.2.2. Specifying how a service is monitored

To use the metrics exposed by your service, you must configure OpenShift Dedicated 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 OpenShift Dedicated, 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:

```
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

- See the Prometheus Operator API documentation for more information on **ServiceMonitor** and **PodMonitor** resources.
- Accessing monitoring for user-defined projects.

5.4.3. 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.

5.4.3.1. 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 OpenShift Dedicated and user-defined projects in the Metrics UI.

NOTE

Only dedicated administrators have access to the third-party UIs provided with OpenShift Dedicated Monitoring.

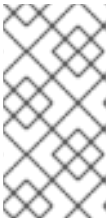
Prerequisites

- 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.

5.4.3.2. 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 OpenShift Dedicated 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 OpenShift Dedicated 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 the Querying metrics for user-defined projects as a developer for details on accessing non-cluster metrics as a developer or a privileged user

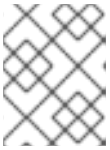
5.4.3.3. Exploring the visualized metrics

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

5.4.4. Next steps

- Alerts

5.5. ALERTS

Alerts for monitoring workloads in user-defined projects are not currently supported in this OpenShift Dedicated.

5.5.1. Next steps

- Reviewing monitoring dashboards

5.6. REVIEWING MONITORING DASHBOARDS

OpenShift Dedicated 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 5.1. Example dashboard in the Developer perspective

NOTE

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

5.6.1. 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 OpenShift Dedicated 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.

5.6.2. Next steps

- Troubleshooting monitoring issues

5.7. TROUBLESHOOTING MONITORING ISSUES

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

5.7.1. 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**.

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

```
  % 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.

   ```
   $ oc get service
   ```

   Example output

   ```
   apiVersion: v1
   kind: Service (1)
   metadata:
     labels: (2)
       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  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.