OpenShift Dedicated 3

Cluster Administration

OpenShift Dedicated 3 Cluster Administration

Last Updated: 2019-05-10

OpenShift Dedicated 3 Cluster Administration

OpenShift Dedicated 3 Cluster Administration

Legal Notice

Abstract

OpenShift Cluster Administration topics cover the day to day tasks for managing your OpenShift Dedicated cluster and other advanced configuration topics.

Table of Contents

CHAPTER 1. OVERVIEW

These Cluster Administration topics cover the day-to-day tasks for managing your OpenShift Dedicated cluster and other advanced configuration topics.

1.1. DEDICATED ADMINISTRATOR ROLE

As a dedicated administrator of an OpenShift Dedicated cluster, your account has increased permissions and access to all user-created projects. If you are new to the role, check out the Getting Started topic on Administering an OpenShift Dedicated Cluster for a quick overview.

NOTE

Some configuration changes or procedures discussed in this guide may be performed only by the OpenShift Dedicated Operations Team. They are included in this guide for informational purposes to help you as an OpenShift Dedicated cluster administrator better understand what configuration options are possible. If you would like to request a change to your cluster that you cannot perform using the administrator CLI, open a support case on the Red Hat Customer Portal .

When your account has the **dedicated-cluster-admin** authorization role bound to it, you are automatically bound to the **dedicated-project-admin** for any new projects that are created by users in the cluster.

You can perform actions associated with a set of verbs (e.g., **create**) to operate on a set of resource names (e.g., **templates**). To view the details of these roles and their sets of verbs and resources, run the following:

```
$ oc describe clusterrole/dedicated-cluster-admin
$ oc describe clusterrole/dedicated-project-admin
```

The verb names do not necessarily all map directly to **oc** commands, but rather equate more generally to the types of CLI operations you can perform. For example, having the **list** verb means that you can display a list of all objects of a given resource name (e.g., using **oc get**), while **get** means that you can display the details of a specific object if you know its name (e.g., using **oc describe**).

OpenShift Dedicated administrators can grant users a **dedicated-reader** role, which provides view-only access at the cluster level, as well as view access for all user projects.

1.2. PROJECT-LEVEL PERMISSIONS

At the project level, an administrator of an OpenShift Dedicated cluster can perform all actions that a project administrator can perform. In addition, the OpenShift Dedicated administrator can set resource quotas and limit ranges for the project.

1.3. CLUSTER-LEVEL PERMISSIONS

Ability	Description

Ability	Description
Manage Users and Groups	● Create, update, and delete users and groups within the cluster. ● Add or remove users to and from groups.
Manage Roles and Bindings	Manage roles and bindings for users and groups within the cluster.
Manage Authorization	● Run checks to determine which users or groups can access a certain resource or resource type. ● Check to see whether a particular user or group can access a certain resource or resource type.
View Certain Cluster-level Resources	View (get/list/watch) certain resources like events, nodes, persistent volumes, and security context constraints.
Create Daemon Sets	Create daemon sets, which ensure that all (or some) nodes run a copy of a pod.

CHAPTER 2. BACKING UP AND RESTORING PROJECTS AND APPLICATIONS

You can manually back up and restore data for your projects and applications.

IMPORTANT

Backup and restore is not guaranteed. You are responsible for backing up your own data.

2.1. BACKING UP APPLICATION DATA

In many cases, you can back up application data by using the **oc rsync** command, assuming **rsync** is installed within the container image. The Red Hat **rhel7** base image contains **rsync**. Therefore, all images that are based on **rhel7** contain it as well. See Troubleshooting and Debugging CLI Operations - rsync.

WARNING

This is a *generic* backup of application data and does not take into account application-specific backup procedures, for example, special export and import procedures for database systems.

Other means of backup might exist depending on the type of the persistent volume you use, for example, Cinder, NFS, or Gluster.

The paths to back up are also *application specific*. You can determine what path to back up by looking at the **mountPath** for volumes in the **deploymentconfig**.

NOTE

You can perform this type of application data backup only while the application pod is running.

Procedure

Example of backing up a Jenkins deployment's application data

1. Get the application data **mountPath** from the **deploymentconfig**:

```
$ oc get dc/jenkins -o jsonpath='{ .spec.template.spec.containers[?
(@.name=="jenkins")].volumeMounts[?(@.name=="jenkins-data")].mountPath }'
/var/lib/jenkins
```

2. Get the name of the pod that is currently running:

```
$ oc get pod --selector=deploymentconfig=jenkins -o jsonpath='{ .metadata.name }'
jenkins-1-37nux
```

3. Use the **oc rsync** command to copy application data:

```
$ oc rsync jenkins-1-37nux:/var/lib/jenkins /tmp/
```

2.2. BACKING UP A PROJECT

Creating a backup of all relevant data involves exporting all important information, then restoring into a new project.

> **NOTE**
>
> Currently, a OpenShift Dedicated project back up and restore tool is being developed by Red Hat. See the following bug for more information:
>
> - bugzilla 1303205.

Procedure

1. List all the relevant data to back up:

```
$ oc get all
NAME        TYPE      FROM      LATEST
bc/ruby-ex  Source    Git       1

NAME              TYPE      FROM          STATUS     STARTED         DURATION
builds/ruby-ex-1  Source    Git@c457001   Complete   2 minutes ago   35s

NAME                DOCKER REPO                                     TAGS      UPDATED
is/guestbook        10.111.255.221:5000/myproject/guestbook         latest    2 minutes ago
is/hello-openshift  10.111.255.221:5000/myproject/hello-openshift   latest    2 minutes ago
is/ruby-22-centos7  10.111.255.221:5000/myproject/ruby-22-centos7   latest    2 minutes ago
is/ruby-ex          10.111.255.221:5000/myproject/ruby-ex           latest    2 minutes ago

NAME                REVISION   DESIRED   CURRENT   TRIGGERED BY
dc/guestbook        1          1         1         config,image(guestbook:latest)
dc/hello-openshift  1          1         1         config,image(hello-openshift:latest)
dc/ruby-ex          1          1         1         config,image(ruby-ex:latest)

NAME                 DESIRED   CURRENT   READY     AGE
rc/guestbook-1       1         1         1         2m
rc/hello-openshift-1 1         1         1         2m
rc/ruby-ex-1         1         1         1         2m

NAME                 CLUSTER-IP       EXTERNAL-IP   PORT(S)             AGE
svc/guestbook        10.111.105.84    <none>        3000/TCP            2m
svc/hello-openshift  10.111.230.24    <none>        8080/TCP,8888/TCP   2m
svc/ruby-ex          10.111.232.117   <none>        8080/TCP            2m

NAME                         READY     STATUS      RESTARTS   AGE
po/guestbook-1-c010g         1/1       Running     0          2m
po/hello-openshift-1-4zw2q   1/1       Running     0          2m
po/ruby-ex-1-build           0/1       Completed   0          2m
po/ruby-ex-1-rxc74           1/1       Running     0          2m
```

2. Export the project objects to a **.yaml** or **.json** file.

   - To export the project objects into a **project.yaml** file:

     ```
     $ oc get -o yaml --export all > project.yaml
     ```

   - To export the project objects into a **project.json** file:

     ```
     $ oc get -o json --export all > project.json
     ```

3. Export the project's **role bindings**, **secrets**, **service accounts**, and **persistent volume claims**:

   ```
   $ for object in rolebindings serviceaccounts secrets imagestreamtags cm
   egressnetworkpolicies rolebindingrestrictions limitranges resourcequotas pvc templates
   cronjobs statefulsets hpa deployments replicasets poddisruptionbudget endpoints
   do
     oc get -o yaml --export $object > $object.yaml
   done
   ```

4. To list all the namespaced objects:

   ```
   $ oc api-resources --namespaced=true -o name
   ```

5. Some exported objects can rely on specific metadata or references to unique IDs in the project. This is a limitation on the usability of the recreated objects.
   When using **imagestreams**, the **image** parameter of a **deploymentconfig** can point to a specific **sha** checksum of an image in the internal registry that would not exist in a restored environment. For instance, running the sample "ruby-ex" as **oc new-app centos/ruby-22-centos7~https://github.com/sclorg/ruby-ex.git** creates an **imagestream ruby-ex** using the internal registry to host the image:

   ```
   $ oc get dc ruby-ex -o jsonpath="{.spec.template.spec.containers[].image}"
   10.111.255.221:5000/myproject/ruby-
   ex@sha256:880c720b23c8d15a53b01db52f7abdcbb2280e03f686a5c8edfef1a2a7b21cee
   ```

   If importing the **deploymentconfig** as it is exported with **oc get --export** it fails if the image does not exist.

2.3. RESTORING APPLICATION DATA

You can restore application data by using the **oc rsync** command, assuming **rsync** is installed within the container image. The Red Hat **rhel7** base image contains **rsync**. Therefore, all images that are based on **rhel7** contain it as well. See Troubleshooting and Debugging CLI Operations - rsync.

> **WARNING**
>
> This is a *generic* restoration of application data and does not take into account application-specific backup procedures, for example, special export and import procedures for database systems.

Other means of restoration might exist depending on the type of the persistent volume you use, for example, Cinder, NFS, or Gluster.

Procedure

Example of restoring a Jenkins deployment's application data

1. Verify the backup:

```
$ ls -la /tmp/jenkins-backup/
total 8
drwxrwxr-x.  3 user     user   20 Sep  6 11:14 .
drwxrwxrwt. 17 root     root 4096 Sep  6 11:16 ..
drwxrwsrwx. 12 user     user 4096 Sep  6 11:14 jenkins
```

2. Use the **oc rsync** tool to copy the data into the running pod:

```
$ oc rsync /tmp/jenkins-backup/jenkins jenkins-1-37nux:/var/lib
```

> **NOTE**
>
> Depending on the application, you may be required to restart the application.

3. Optionally, restart the application with new data:

```
$ oc delete pod jenkins-1-37nux
```

Alternatively, you can scale down the deployment to 0, and then up again:

```
$ oc scale --replicas=0 dc/jenkins
$ oc scale --replicas=1 dc/jenkins
```

2.3.1. Restoring a project

To restore a project, create the new project, then restore any exported files by running **oc create -f pods.json**. However, restoring a project from scratch requires a specific order because some objects depend on others. For example, you must create the **configmaps** before you create any **pods**.

Procedure

1. If the project was exported as a single file, import it by running the following commands:

```
$ oc new-project <projectname>
$ oc create -f project.yaml
$ oc create -f secret.yaml
$ oc create -f serviceaccount.yaml
$ oc create -f pvc.yaml
$ oc create -f rolebindings.yaml
```

WARNING

Some resources, such as pods and default service accounts, can fail to be created.

CHAPTER 3. MANAGING USERS

3.1. OVERVIEW

A user is an entity that interacts with the OpenShift Dedicated API. These can be a developer for developing applications or an administrator for managing the cluster. Users can be assigned to groups, which set the permissions applied to all the group’s members. For example, you can give API access to a group, which give all members of the group API access.

This topic describes the management of user accounts, including how new user accounts are created in OpenShift Dedicated and how they can be deleted.

3.2. CREATING A USER

The process for creating a user depends on the configured identity provider. By default, OpenShift Dedicated uses the **DenyAll** identity provider, which denies access for all user names and passwords.

3.3. VIEWING USER AND IDENTITY LISTS

OpenShift Dedicated user configuration is stored in several locations within OpenShift Dedicated. Regardless of the identity provider, OpenShift Dedicated internally stores details like role-based access control (RBAC) information and group membership. To completely remove user information, this data must be removed in addition to the user account.

In OpenShift Dedicated, two object types contain user data outside the identification provider: **user** and **identity**.

To get the current list of users:

```
$ oc get user
NAME      UID                                    FULL NAME   IDENTITIES
demo     75e4b80c-dbf1-11e5-8dc6-0e81e52cc949              htpasswd_auth:demo
```

To get the current list of identities:

```
$ oc get identity
NAME                  IDP NAME        IDP USER NAME   USER NAME   USER UID
htpasswd_auth:demo    htpasswd_auth   demo            demo        75e4b80c-dbf1-11e5-8dc6-
0e81e52cc949
```

Note the matching UID between the two object types. If you attempt to change the authentication provider after starting to use OpenShift Dedicated, the user names that overlap will not work because of the entries in the identity list, which will still point to the old authentication method.

3.4. CREATING GROUPS

While a user is an entity making requests to OpenShift Dedicated, users can be organized into one or more groups made up from a set of users. Groups are useful for managing many users at one time, such as for authorization policies, or to grant permissions to multiple users at once.

If your organization is using LDAP, you can synchronize any LDAP records to OpenShift Dedicated so that you can configure groups on one place. This presumes that information about your users is in an MDAP server.

If you are not using LDAP, you can use the following procedure to manually create groups.

To create a new group:

```
# oc adm groups new <group_name> <user1> <user2>
```

For example, to create the **west** groups and in it place the **john** and **betty** users:

```
# oc adm groups new west john betty
```

To verify that the group has been created, and list the users associated with the group, run the following:

```
# oc get groups
NAME      USERS
west      john, betty
```

Next steps:

- Managing role bindings

3.5. MANAGING USER AND GROUP LABELS

To add a label to a user or group:

```
$ oc label user/<user_name> <label_name>
```

For example, if the user name is **theuser** and the label is **level=gold**:

```
$ oc label user/theuser level=gold
```

To remove the label:

```
$ oc label user/<user_name> <label_name>-
```

To show labels for a user or group:

```
$ oc describe user/<user_name>
```

3.6. DELETING A USER

To delete a user:

1. Delete the user record:

   ```
   $ oc delete user demo
   user "demo" deleted
   ```

2. Delete the user identity.
   The identity of the user is related to the identification provider you use. Get the provider name from the user record in **oc get user**.

In this example, the identity provider name is **htpasswd_auth**. The command is:

```
# oc delete identity htpasswd_auth:demo
identity "htpasswd_auth:demo" deleted
```

If you skip this step, the user will not be able to log in again.

After you complete these steps, a new account will be created in OpenShift Dedicated when the user logs in again.

If your intention is to prevent the user from being able to log in again (for example, if an employee has left the company and you want to permanently delete the account), you can also remove the user from your authentication back end (like **htpasswd**, **kerberos**, or others) for the configured identity provider.

For example, if you are using **htpasswd**, delete the entry in the ***htpasswd*** file that is configured for OpenShift Dedicated with the user name and password.

For external identification management like Lightweight Directory Access Protocol (LDAP) or Red Hat Identity Management (IdM), use the user management tools to remove the user entry.

CHAPTER 4. MANAGING PROJECTS

4.1. OVERVIEW

In OpenShift Dedicated, projects are used to group and isolate related objects. As an administrator, you can give developers access to certain projects, allow them to create their own, and give them administrative rights within individual projects.

A dedicated administrator is by default an administrator for all projects on the cluster that are not managed by Red Hat Operations.

4.2. SELF-PROVISIONING PROJECTS

You can allow developers to create their own projects. There is an endpoint that will provision a project according to a template. The web console and **oc new-project** command use this endpoint when a developer creates a new project .

4.2.1. Modifying the Template for New Projects

The API server automatically provisions projects based on the template that is identified by the **projectRequestTemplate** parameter of the ***master-config.yaml*** file If the parameter is not defined, the API server creates a default template that creates a project with the requested name, and assigns the requesting user to the "admin" role for that project.

To create your own custom project template:

1. Start with the current default project template:

   ```
   $ oc adm create-bootstrap-project-template -o yaml > template.yaml
   ```

2. Use a text editor to modify the ***template.yaml*** file by adding objects or modifying existing objects.

3. Load the template:

   ```
   $ oc create -f template.yaml -n default
   ```

4. Modify the ***master-config.yaml*** file to reference the loaded template:

   ```
   ...
   projectConfig:
     projectRequestTemplate: "default/project-request"
     ...
   ```

When a project request is submitted, the API substitutes the following parameters into the template:

Parameter	Description
PROJECT_NAME	The name of the project. Required.
PROJECT_DISPLAYNAME	The display name of the project. May be empty.

Parameter	Description
PROJECT_DESCRIPTION	The description of the project. May be empty.
PROJECT_ADMIN_USER	The username of the administrating user.
PROJECT_REQUESTING_USER	The username of the requesting user.

Access to the API is granted to developers with the **self-provisioner** role and the **self-provisioners** cluster role binding. This role is available to all authenticated developers by default.

4.3. ENFORCING DIFFERENT PROJECT LIMITS FOR USERS

If you are an OpenShift Dedicated administrator and want to enforce different limits on the number of projects that your users can create, apply the appropriate label to the user.

Acceptable Values for Project Limit

```
openshift.io/project-limit: '0'
openshift.io/project-limit: '1'
openshift.io/project-limit: '3'
openshift.io/project-limit: '5'
openshift.io/project-limit: '10'
openshift.io/project-limit: '20'
openshift.io/project-limit: '50'
openshift.io/project-limit: '100'
openshift.io/project-limit: 'unlimited'
```

The default limit is **unlimited**.

4.4. MANAGING LIMITS AND QUOTA ON PROJECT RESOURCES

A default project template is available for any projects created by users on the cluster. The template helps prevent inadvertent consumption of all of the cluster and node resources. The project template is created and configured in the **dedicated-admin** namespace.

Edit the default project template with the following command:

```
$ oc edit template project-request -n dedicated-admin
```

If you modify the default project template and want your changes to remain after you upgrade, you must update the **openshift.io/overwrite-protect** annotation value to **true**.

```
$ oc annotate template project-request -n dedicated-admin openshift.io/overwrite-protect=true
```

The default value is **false**, so by default your changes are not protected against overwrite. You only need to update the value if you make changes to the project template.

Setting the value to **true** results in the upgrade skipping the template and not updating it. Therefore, you must check that newer versions of OpenShift Dedicated provide anything of interest that they would need to incorporate.

See Setting Quotas and Setting Limit Ranges for more information.

CHAPTER 5. MANAGING NETWORKING

5.1. OVERVIEW

This topic describes the management of the overall cluster network, including project isolation and outbound traffic control.

5.2. MANAGING POD NETWORKS

When your cluster is configured to use the **ovs-multitenant** SDN plug-in you can manage the separate pod overlay networks for projects using the administrator CLI.

5.2.1. Joining Project Networks

To join projects to an existing project network:

```
$ oc adm pod-network join-projects --to=<project1> <project2> <project3>
```

In the above example, all the pods and services in **<project2>** and **<project3>** can now access any pods and services in **<project1>** and vice versa. Services can be accessed either by IP or fully qualified DNS name (**<service>.<pod_namespace>.svc.cluster.local**). For example, to access a service named **db** in a project **myproject**, use **db.myproject.svc.cluster.local**.

Alternatively, instead of specifying specific project names, you can use the **--selector=<project_selector>** option.

To verify the networks you have joined together:

```
$ oc get netnamespaces
```

Then look at the **NETID** column. Projects in the same pod-network will have the same NetID.

5.3. ISOLATING PROJECT NETWORKS

To isolate the project network in the cluster and vice versa, run:

```
$ oc adm pod-network isolate-projects <project1> <project2>
```

In the above example, all of the pods and services in **<project1>** and **<project2>** can *not* access any pods and services from other non-global projects in the cluster and vice versa.

Alternatively, instead of specifying specific project names, you can use the **--selector=<project_selector>** option.

5.3.1. Making Project Networks Global

To allow projects to access all pods and services in the cluster and vice versa:

```
$ oc adm pod-network make-projects-global <project1> <project2>
```

In the above example, all the pods and services in **<project1>** and **<project2>** can now access any pods and services in the cluster and vice versa.

Alternatively, instead of specifying specific project names, you can use the **--selector=<project_selector>** option.

5.4. USING AN EGRESS FIREWALL TO LIMIT ACCESS TO EXTERNAL RESOURCES

As an OpenShift Dedicated cluster administrator, you can use egress firewall policy to limit the external addresses that some or all pods can access from within the cluster, so that:

- A pod can only talk to internal hosts, and cannot initiate connections to the public Internet.
  Or,

- A pod can only talk to the public Internet, and cannot initiate connections to internal hosts (outside the cluster).
  Or,

- A pod cannot reach specified internal subnets/hosts that it should have no reason to contact.

Egress policies can be set at the pod selector-level and project-level. For example, you can allow **<project A>** access to a specified IP range but deny the same access to **<project B>**. Or, you can restrict application developers from updating from (Python) pip mirrors, and force updates to only come from approved sources.

Project administrators can neither create **EgressNetworkPolicy** objects, nor edit the ones you create in their project. There are also several other restrictions on where **EgressNetworkPolicy** can be created:

- The **default** project (and any other project that has been made global via **oc adm pod-network make-projects-global**) cannot have egress policy.

- If you merge two projects together (via **oc adm pod-network join-projects**), then you cannot use egress policy in *any* of the joined projects.

- No project may have more than one egress policy object.

Violating any of these restrictions results in broken egress policy for the project, and may cause all external network traffic to be dropped.

Use the **oc** command or the REST API to configure egress policy. You can use **oc [create|replace|delete]** to manipulate **EgressNetworkPolicy** objects. The ***api/swagger-spec/oapi-v1.json*** file has API-level details on how the objects actually work.

To configure egress policy:

1. Navigate to the project you want to affect.

2. Create a JSON file with the desired policy details. For example:

```
{
    "kind": "EgressNetworkPolicy",
    "apiVersion": "v1",
    "metadata": {
        "name": "default"
    },
    "spec": {
        "egress": [
            {
                "type": "Allow",
                "to": {
                    "cidrSelector": "1.2.3.0/24"
                }
            },
            {
                "type": "Allow",
                "to": {
                    "dnsName": "www.foo.com"
                }
            },
            {
                "type": "Deny",
                "to": {
                    "cidrSelector": "0.0.0.0/0"
                }
            }
        ]
    }
}
```

When the example above is added to a project, it allows traffic to IP range **1.2.3.0/24** and domain name **www.foo.com**, but denies access to all other external IP addresses. Traffic to other pods is not affected because the policy only applies to *external* traffic.

The rules in an **EgressNetworkPolicy** are checked in order, and the first one that matches takes effect. If the three rules in the above example were reversed, then traffic would not be allowed to **1.2.3.0/24** and **www.foo.com** because the **0.0.0.0/0** rule would be checked first, and it would match and deny all traffic.

Domain name updates are polled based on the TTL (time to live) value of the domain returned by the local non-authoritative servers. The pod should also resolve the domain from the same local nameservers when necessary, otherwise the IP addresses for the domain perceived by the egress network policy controller and the pod will be different, and the egress network policy may not be enforced as expected. Since egress network policy controller and pod are asynchronously polling the same local nameserver, there could be a race condition where pod may get the updated IP before the egress controller. Due to this current limitation, domain name usage in **EgressNetworkPolicy** is only recommended for domains with infrequent IP address changes.

5.5. ENABLING HTTP STRICT TRANSPORT SECURITY

HTTP Strict Transport Security (HSTS) policy is a security enhancement, which ensures that only HTTPS traffic is allowed on the host. Any HTTP requests are dropped by default. This is useful for ensuring secure interactions with websites, or to offer a secure application for the user’s benefit.

When HSTS is enabled, HSTS adds a Strict Transport Security header to HTTPS responses from the site. You can use the **insecureEdgeTerminationPolicy** value in a route to redirect to send HTTP to HTTPS. However, when HSTS is enabled, the client changes all requests from the HTTP URL to HTTPS before the request is sent, eliminating the need for a redirect. This is not required to be supported by the client, and can be disabled by setting **max-age=0**.

IMPORTANT

HSTS works only with secure routes (either edge terminated or re-encrypt). The configuration is ineffective on HTTP or passthrough routes.

To enable HSTS to a route, add the **haproxy.router.openshift.io/hsts_header** value to the edge terminated or re-encrypt route:

```
apiVersion: v1
kind: Route
metadata:
  annotations:
    haproxy.router.openshift.io/hsts_header: max-age=31536000;includeSubDomains;preload
```

IMPORTANT

Ensure there are no spaces and no other values in the parameters in the **haproxy.router.openshift.io/hsts_header** value. Only **max-age** is required.

The required **max-age** parameter indicates the length of time, in seconds, the HSTS policy is in effect for. The client updates **max-age** whenever a response with a HSTS header is received from the host. When **max-age** times out, the client discards the policy.

The optional **includeSubDomains** parameter tells the client that all subdomains of the host are to be treated the same as the host.

If **max-age** is greater than 0, the optional **preload** parameter allows external services to include this site in their HSTS preload lists. For example, sites such as Google can construct a list of sites that have **preload** set. Browsers can then use these lists to determine which sites to only talk to over HTTPS, even before they have interacted with the site. Without **preload** set, they need to have talked to the site over HTTPS to get the header.

5.6. TROUBLESHOOTING THROUGHPUT ISSUES

Sometimes applications deployed through OpenShift Dedicated can cause network throughput issues such as unusually high latency between specific services.

Use the following methods to analyze performance issues if pod logs do not reveal any cause of the problem:

- Use a packet analyzer, such as ping or tcpdump to analyze traffic between a pod and its node. For example, run the tcpdump tool on each pod while reproducing the behavior that led to the issue. Review the captures on both sides to compare send and receive timestamps to analyze the latency of traffic to/from a pod. Latency can occur in OpenShift Dedicated if a node interface is overloaded with traffic from other pods, storage devices, or the data plane.

  ```
  $ tcpdump -s 0 -i any -w /tmp/dump.pcap host <podip 1> && host <podip 2> 1
  ```

  1 **podip** is the IP address for the pod. Run the following command to get the IP address of the pods:

  ```
  # oc get pod <podname> -o wide
  ```

tcpdump generates a file at ***/tmp/dump.pcap*** containing all traffic between these two pods. Ideally, run the analyzer shortly before the issue is reproduced and stop the analyzer shortly after the issue is finished reproducing to minimize the size of the file. You can also run a packet analyzer between the nodes (eliminating the SDN from the equation) with:

```
# tcpdump -s 0 -i any -w /tmp/dump.pcap port 4789
```

- Use a bandwidth measuring tool, such as iperf, to measure streaming throughput and UDP throughput. Run the tool from the pods first, then from the nodes to attempt to locate any bottlenecks. The iperf3 tool is included as part of RHEL 7.

CHAPTER 6. CONFIGURING SERVICE ACCOUNTS

6.1. OVERVIEW

When a person uses the OpenShift Dedicated CLI or web console, their API token authenticates them to the OpenShift Dedicated API. However, when a regular user's credentials are not available, it is common for components to make API calls independently. For example:

- Replication controllers make API calls to create or delete pods.
- Applications inside containers can make API calls for discovery purposes.
- External applications can make API calls for monitoring or integration purposes.

Service accounts provide a flexible way to control API access without sharing a regular user's credentials.

6.2. USER NAMES AND GROUPS

Every service account has an associated user name that can be granted roles, just like a regular user. The user name is derived from its project and name:

```
system:serviceaccount:<project>:<name>
```

For example, to add the **view** role to the **robot** service account in the **top-secret** project:

```
$ oc policy add-role-to-user view system:serviceaccount:top-secret:robot
```

IMPORTANT

If you want to grant access to a specific service account in a project, you can use the **-z** flag. From the project to which the service account belongs, use the **-z** flag and specify the **<serviceaccount_name>**. This is highly recommended, as it helps prevent typos and ensures that access is granted only to the specified service account. For example:

```
$ oc policy add-role-to-user <role_name> -z <serviceaccount_name>
```

If not in the project, use the **-n** option to indicate the project namespace it applies to, as shown in the examples below.

Every service account is also a member of two groups:

system:serviceaccount

Includes all service accounts in the system.

system:serviceaccount:<project>

Includes all service accounts in the specified project.

For example, to allow all service accounts in all projects to view resources in the **top-secret** project:

```
$ oc policy add-role-to-group view system:serviceaccount -n top-secret
```

To allow all service accounts in the **managers** project to edit resources in the **top-secret** project:

```
$ oc policy add-role-to-group edit system:serviceaccount:managers -n top-secret
```

6.3. GRANTING SERVICE ACCOUNTS ACCESS TO DEDICATED-ADMIN ROLES

As an OpenShift Dedicated administrator, you can use service accounts to perform any actions that require OpenShift Dedicated **admin** roles.

The **dedicated-admin** service creates the **dedicated-admins** group. This group is granted the roles at the cluster or individual project level. Users can be assigned to this group and group membership defines who has OpenShift Dedicated administrator access. However, by design, service accounts cannot be added to regular groups.

Instead, the **dedicated-admin** service creates a special project for this purpose named **dedicated-admin**. The service account group for this project is granted OpenShift Dedicated **admin** roles, granting OpenShift Dedicated administrator access to all service accounts within the **dedicated-admin** project. These service accounts can then be used to perform any actions that require OpenShift Dedicated administrator access.

Users that are members of the **dedicated-admins** group, and thus have been granted the **dedicated-admin** role, have **edit** access to the **dedicated-admin** project. This allows these users to manage the service accounts in this project and create new ones as needed.

Users with a **dedicated-reader** role are granted edit and view access to the **dedicated-reader** project and view-only access to the other projects.

6.4. MANAGING SERVICE ACCOUNTS

Service accounts are API objects that exist within each project. To manage service accounts, you can use the **oc** command with the **sa** or **serviceaccount** object type or use the web console.

To get a list of existing service accounts in the current project:

```
$ oc get sa
NAME       SECRETS   AGE
builder    2         2d
default    2         2d
deployer   2         2d
```

To create a new service account:

```
$ oc create sa robot
serviceaccount "robot" created
```

As soon as a service account is created, two secrets are automatically added to it:

- an API token
- credentials for the OpenShift Container Registry

These can be seen by describing the service account:

```
$ oc describe sa robot
```

```
Name:  robot
Namespace: project1
Labels:  <none>
Annotations: <none>

Image pull secrets: robot-dockercfg-qzbhb

Mountable secrets:  robot-token-f4khf
               robot-dockercfg-qzbhb

Tokens:          robot-token-f4khf
               robot-token-z8h44
```

The system ensures that service accounts always have an API token and registry credentials.

The generated API token and registry credentials do not expire, but they can be revoked by deleting the secret. When the secret is deleted, a new one is automatically generated to take its place.

6.5. ENABLING SERVICE ACCOUNT AUTHENTICATION

Service accounts authenticate to the API using tokens signed by a private RSA key. The authentication layer verifies the signature using a matching public RSA key.

To enable service account token generation, update the **serviceAccountConfig** stanza in the ***/etc/origin/master/master-config.yml*** file on the master to specify a **privateKeyFile** (for signing), and a matching public key file in the **publicKeyFiles** list:

```
serviceAccountConfig:
  ...
  masterCA: ca.crt 1
  privateKeyFile: serviceaccount.private.key 2
  publicKeyFiles:
  - serviceaccount.public.key 3
  - ...
```

1 CA file used to validate the API server’s serving certificate.

2 Private RSA key file (for token signing).

3 Public RSA key files (for token verification). If private key files are provided, then the public key component is used. Multiple public key files can be specified, and a token will be accepted if it can be validated by one of the public keys. This allows rotation of the signing key, while still accepting tokens generated by the previous signer.

CHAPTER 7. MANAGING ROLE-BASED ACCESS CONTROL (RBAC)

7.1. OVERVIEW

You can use the CLI to view RBAC resources and the administrator CLI to manage the roles and bindings.

Dedicated administrators can view but not manage cluster roles. They can manage cluster role bindings and manage local roles and bindings.

7.2. VIEWING ROLES AND BINDINGS

Roles can be used to grant various levels of access both cluster-wide as well as at the project-scope. Users and groups can be associated with, or *bound* to, multiple roles at the same time. You can view details about the roles and their bindings using the **oc describe** command.

Users with the **dedicated-cluster-admin** role can view but not manage cluster roles. They can manage cluster role bindings and manage local roles and bindings. Users with the **admin**default cluster role bound locally can manage roles and bindings in that project.

NOTE

Review a full list of verbs in the Evaluating Authorization section.

7.2.1. Viewing cluster roles

7.2.2. Viewing cluster role bindings

To view the current set of cluster role bindings, which show the users and groups that are bound to various roles:

```
$ oc describe clusterrolebinding.rbac
```

7.2.3. Viewing local roles and bindings

All of the default cluster roles can be bound locally to users or groups.

The local role bindings are also viewable.

To view the current set of local role bindings, which show the users and groups that are bound to various roles:

```
$ oc describe rolebinding.rbac
```

By default, the current project is used when viewing local role bindings. Alternatively, a project can be specified with the **-n** flag. This is useful for viewing the local role bindings of another project, if the user already has the **admin**default cluster role in it.

```
$ oc describe rolebinding.rbac -n joe-project
Name:  admin
```

```
Labels:  <none>
Annotations: <none>
Role:
  Kind: ClusterRole
  Name: admin
Subjects:
  Kind Name Namespace
  ---- ---- ---------
  User joe


Name:  system:deployers
Labels:  <none>
Annotations: <none>
Role:
  Kind: ClusterRole
  Name: system:deployer
Subjects:
  Kind   Name  Namespace
  ----   ----  ---------
  ServiceAccount deployer joe-project


Name:  system:image-builders
Labels:  <none>
Annotations: <none>
Role:
  Kind: ClusterRole
  Name: system:image-builder
Subjects:
  Kind   Name Namespace
  ----   ---- ---------
  ServiceAccount builder joe-project


Name:  system:image-pullers
Labels:  <none>
Annotations: <none>
Role:
  Kind: ClusterRole
  Name: system:image-puller
Subjects:
  Kind Name     Namespace
  ---- ----     ---------
  Group system:serviceaccounts:joe-project
```

7.3. MANAGING ROLE BINDINGS

Adding, or *binding*, a role to users or groups gives the user or group the relevant access granted by the role. You can add and remove roles to and from users and groups using **oc adm policy** commands.

When managing a user or group's associated roles for local role bindings using the following operations, a project may be specified with the **-n** flag. If it is not specified, then the current project is used.

Table 7.1. Local role binding operations

Command	Description
$ oc adm policy who-can ***<verb> <resource>***	Indicates which users can perform an action on a resource.
$ oc adm policy add-role-to-user ***<role> <username>***	Binds a given role to specified users in the current project.
$ oc adm policy remove-role-from-user ***<role> <username>***	Removes a given role from specified users in the current project.
$ oc adm policy remove-user ***<username>***	Removes specified users and all of their roles in the current project.
$ oc adm policy add-role-to-group ***<role> <groupname>***	Binds a given role to specified groups in the current project.
$ oc adm policy remove-role-from-group ***<role> <groupname>***	Removes a given role from specified groups in the current project.
$ oc adm policy remove-group ***<groupname>***	Removes specified groups and all of their roles in the current project.
--rolebinding-name=	Can be used with **oc adm policy** commands to retain the rolebinding name assigned to a user or group.

For example, you can add the **admin** role to the **alice** user in **joe-project** by running:

```
$ oc adm policy add-role-to-user admin alice -n joe-project
```

You can then view the local role bindings and verify the addition in the output:

```
$ oc describe rolebinding.rbac -n joe-project
Name:  admin
Labels:  <none>
Annotations: <none>
Role:
  Kind: ClusterRole
  Name: admin
Subjects:
  Kind Name Namespace
  ---- ---- ---------
  User joe
  User alice 1


Name:  system:deployers
Labels:  <none>
Annotations: <none>
Role:
  Kind: ClusterRole
  Name: system:deployer
Subjects:
  Kind   Name  Namespace
  ----   ----  ---------
  ServiceAccount deployer joe-project


Name:  system:image-builders
Labels:  <none>
Annotations: <none>
Role:
  Kind: ClusterRole
  Name: system:image-builder
Subjects:
  Kind   Name Namespace
  ----   ---- ---------
  ServiceAccount builder joe-project


Name:  system:image-pullers
Labels:  <none>
Annotations: <none>
Role:
  Kind: ClusterRole
  Name: system:image-puller
Subjects:
  Kind Name     Namespace
  ---- ----     ---------
  Group system:serviceaccounts:joe-project
```

1 The **alice** user has been added to the **admins RoleBinding**.

CHAPTER 8. MANAGING SECURITY CONTEXT CONSTRAINTS

8.1. OVERVIEW

Security context constraints allow administrators to control permissions for pods. To learn more about this API type, see the security context constraints (SCCs) architecture documentation. You can manage SCCs in your instance as normal API objects using the CLI.

As an OpenShift Dedicated cluster administrator, you can list and view details for SCCs, but cannot edit or delete the default SCCs.

8.2. LISTING SECURITY CONTEXT CONSTRAINTS

To get a current list of SCCs:

```
$ oc get scc

NAME              PRIV     CAPS     SELINUX     RUNASUSER          FSGROUP     SUPGROUP
PRIORITY   READONLYROOTFS   VOLUMES
anyuid            false    []       MustRunAs   RunAsAny           RunAsAny    RunAsAny    10       false
[configMap downwardAPI emptyDir persistentVolumeClaim secret]
hostaccess        false    []       MustRunAs   MustRunAsRange     MustRunAs   RunAsAny    <none>
false           [configMap downwardAPI emptyDir hostPath persistentVolumeClaim secret]
hostmount-anyuid  false    []       MustRunAs   RunAsAny           RunAsAny    RunAsAny    <none>
false           [configMap downwardAPI emptyDir hostPath nfs persistentVolumeClaim secret]
hostnetwork       false    []       MustRunAs   MustRunAsRange     MustRunAs   MustRunAs   <none>
false           [configMap downwardAPI emptyDir persistentVolumeClaim secret]
nonroot           false    []       MustRunAs   MustRunAsNonRoot   RunAsAny    RunAsAny    <none>
false           [configMap downwardAPI emptyDir persistentVolumeClaim secret]
privileged        true     [*]      RunAsAny    RunAsAny           RunAsAny    RunAsAny    <none>
false           [*]
restricted        false    []       MustRunAs   MustRunAsRange     MustRunAs   RunAsAny    <none>
false           [configMap downwardAPI emptyDir persistentVolumeClaim secret]
```

8.3. EXAMINING A SECURITY CONTEXT CONSTRAINTS OBJECT

To examine a particular SCC, use **oc get**, **oc describe**, or **oc edit**. For example, to examine the **restricted** SCC:

```
$ oc describe scc restricted
Name:     restricted
Priority:    <none>
Access:
  Users:    <none>
  Groups:    system:authenticated
Settings:
  Allow Privileged:   false
  Default Add Capabilities:  <none>
  Required Drop Capabilities:  KILL,MKNOD,SYS_CHROOT,SETUID,SETGID
  Allowed Capabilities:   <none>
  Allowed Seccomp Profiles:  <none>
  Allowed Volume Types:
configMap,downwardAPI,emptyDir,persistentVolumeClaim,projected,secret
```

```
Allow Host Network:   false
Allow Host Ports:   false
Allow Host PID:   false
Allow Host IPC:   false
Read Only Root Filesystem:  false
Run As User Strategy: MustRunAsRange
  UID:    <none>
  UID Range Min:   <none>
  UID Range Max:   <none>
SELinux Context Strategy: MustRunAs
  User:    <none>
  Role:    <none>
  Type:    <none>
  Level:    <none>
FSGroup Strategy: MustRunAs
  Ranges:    <none>
Supplemental Groups Strategy: RunAsAny
  Ranges:    <none>
```

CHAPTER 9. SETTING QUOTAS

9.1. OVERVIEW

A resource quota, defined by a **ResourceQuota** object, provides constraints that limit aggregate resource consumption per project. It can limit the quantity of objects that can be created in a project by type, as well as the total amount of compute resources and storage that may be consumed by resources in that project.

NOTE

See the Developer Guide for more on compute resources.

9.2. RESOURCES MANAGED BY QUOTA

The following describes the set of compute resources and object types that may be managed by a quota.

NOTE

A pod is in a terminal state if **status.phase in (Failed, Succeeded)** is true.

Table 9.1. Compute Resources Managed by Quota

Resource Name	Description
cpu	The sum of CPU requests across all pods in a non-terminal state cannot exceed this value. **cpu** and **requests.cpu** are the same value and can be used interchangeably.
memory	The sum of memory requests across all pods in a non-terminal state cannot exceed this value. **memory** and **requests.memory** are the same value and can be used interchangeably.
ephemeral-storage	The sum of local ephemeral storage requests across all pods in a non-terminal state cannot exceed this value. **ephemeral-storage** and **requests.ephemeral-storage** are the same value and can be used interchangeably. This resource is available only if you enabled the ephemeral storage technology preview. This feature is disabled by default.
requests.cpu	The sum of CPU requests across all pods in a non-terminal state cannot exceed this value. **cpu** and **requests.cpu** are the same value and can be used interchangeably.
requests.memory	The sum of memory requests across all pods in a non-terminal state cannot exceed this value. **memory** and **requests.memory** are the same value and can be used interchangeably.

Resource Name	Description
requests.ephemeral-storage	The sum of ephemeral storage requests across all pods in a non-terminal state cannot exceed this value. **ephemeral-storage** and **requests.ephemeral-storage** are the same value and can be used interchangeably. This resource is available only if you enabled the ephemeral storage technology preview. This feature is disabled by default.
limits.cpu	The sum of CPU limits across all pods in a non-terminal state cannot exceed this value.
limits.memory	The sum of memory limits across all pods in a non-terminal state cannot exceed this value.
limits.ephemeral-storage	The sum of ephemeral storage limits across all pods in a non-terminal state cannot exceed this value. This resource is available only if you enabled the ephemeral storage technology preview. This feature is disabled by default.

Table 9.2. Storage Resources Managed by Quota

Resource Name	Description
requests.storage	The sum of storage requests across all persistent volume claims in any state cannot exceed this value.
persistentvolumeclaims	The total number of persistent volume claims that can exist in the project.
<storage-class-name>.storageclass.storage.k8s.io/requests.storage	The sum of storage requests across all persistent volume claims in any state that have a matching storage class, cannot exceed this value.
<storage-class-name>.storageclass.storage.k8s.io/persistentvolumeclaims	The total number of persistent volume claims with a matching storage class that can exist in the project.

Table 9.3. Object Counts Managed by Quota

Resource Name	Description
pods	The total number of pods in a non-terminal state that can exist in the project.
replicationcontrollers	The total number of replication controllers that can exist in the project.
resourcequotas	The total number of resource quotas that can exist in the project.

Resource Name	Description
services	The total number of services that can exist in the project.
secrets	The total number of secrets that can exist in the project.
configmaps	The total number of **ConfigMap** objects that can exist in the project.
persistentvolumeclaims	The total number of persistent volume claims that can exist in the project.
openshift.io/imagestreams	The total number of image streams that can exist in the project.

You can configure an object count quota for these standard namespaced resource types using the **count/<resource>.<group>** syntax while creating a quota.

```
$ oc create quota <name> --hard=count/<resource>.<group>=<quota> 1
```

1 **<resource>** is the name of the resource, and **<group>** is the API group, if applicable. Use the **kubectl api-resources** command for a list of resources and their associated API groups.

9.2.1. Setting Resource Quota for Extended Resources

Overcommitment of resources is not allowed for extended resources, so you must specify **requests** and **limits** for the same extended resource in a quota. Currently, only quota items with the prefix **requests.** are allowed for extended resources. The following is an example scenario of how to set resource quota for the GPU resource **nvidia.com/gpu**.

Procedure

1. Determine how many GPUs are available on a node in your cluster. For example:

```
# oc describe node ip-172-31-27-209.us-west-2.compute.internal | egrep
'Capacity|Allocatable|gpu'
                openshift.com/gpu-accelerator=true
Capacity:
 nvidia.com/gpu:  2
Allocatable:
 nvidia.com/gpu:  2
  nvidia.com/gpu  0           0
```

In this example, 2 GPUs are available.

2. Set a quota in the namespace **nvidia**. In this example, the quota is **1**:

```
# cat gpu-quota.yaml
apiVersion: v1
kind: ResourceQuota
```

```
metadata:
  name: gpu-quota
  namespace: nvidia
spec:
  hard:
    requests.nvidia.com/gpu: 1
```

3. Create the quota:

```
# oc create -f gpu-quota.yaml
resourcequota/gpu-quota created
```

4. Verify that the namespace has the correct quota set:

```
# oc describe quota gpu-quota -n nvidia
Name:                    gpu-quota
Namespace:               nvidia
Resource                 Used  Hard
--------                 ----  ----
requests.nvidia.com/gpu  0     1
```

5. Run a pod that asks for a single GPU:

```
# oc create pod gpu-pod.yaml
```

```
apiVersion: v1
kind: Pod
metadata:
  generateName: gpu-pod-
  namespace: nvidia
spec:
  restartPolicy: OnFailure
  containers:
  - name: rhel7-gpu-pod
    image: rhel7
    env:
      - name: NVIDIA_VISIBLE_DEVICES
        value: all
      - name: NVIDIA_DRIVER_CAPABILITIES
        value: "compute,utility"
      - name: NVIDIA_REQUIRE_CUDA
        value: "cuda>=5.0"

    command: ["sleep"]
    args: ["infinity"]

    resources:
      limits:
        nvidia.com/gpu: 1
```

6. Verify that the pod is running:

```
# oc get pods
NAME           READY    STATUS     RESTARTS   AGE
gpu-pod-s46h7     1/1       Running     0          1m
```

7. Verify that the quota **Used** counter is correct:

```
# oc describe quota gpu-quota -n nvidia
Name:                  gpu-quota
Namespace:              nvidia
Resource                 Used  Hard
--------                 ----  ----
requests.nvidia.com/gpu  1     1
```

8. Attempt to create a second GPU pod in the **nvidia** namespace. This is technically available on the node because it has 2 GPUs:

```
# oc create -f gpu-pod.yaml
Error from server (Forbidden): error when creating "gpu-pod.yaml": pods "gpu-pod-f7z2w" is
forbidden: exceeded quota: gpu-quota, requested: requests.nvidia.com/gpu=1, used:
requests.nvidia.com/gpu=1, limited: requests.nvidia.com/gpu=1
```

This **Forbidden** error message is expected because you have a quota of 1 GPU and this pod tried to allocate a second GPU, which exceeds its quota.

9.3. QUOTA SCOPES

Each quota can have an associated set of *scopes*. A quota will only measure usage for a resource if it matches the intersection of enumerated scopes.

Adding a scope to a quota restricts the set of resources to which that quota can apply. Specifying a resource outside of the allowed set results in a validation error.

Scope	Description
Terminating	Match pods where **spec.activeDeadlineSeconds >= 0**.
NotTerminating	Match pods where **spec.activeDeadlineSeconds** is **nil**.
BestEffort	Match pods that have best effort quality of service for either **cpu** or **memory**.
NotBestEffort	Match pods that do not have best effort quality of service for **cpu** and **memory**.

A **BestEffort** scope restricts a quota to limiting the following resources:

- **pods**

A **Terminating**, **NotTerminating**, and **NotBestEffort** scope restricts a quota to tracking the following resources:

- **pods**

- **memory**
- **requests.memory**
- **limits.memory**
- **cpu**
- **requests.cpu**
- **limits.cpu**
- **ephemeral-storage**
- **requests.ephemeral-storage**
- **limits.ephemeral-storage**

NOTE

Ephemeral storage requests and limits apply only if you enabled the ephemeral storage technology preview. This feature is disabled by default.

9.4. QUOTA ENFORCEMENT

After a resource quota for a project is first created, the project restricts the ability to create any new resources that may violate a quota constraint until it has calculated updated usage statistics.

After a quota is created and usage statistics are updated, the project accepts the creation of new content. When you create or modify resources, your quota usage is incremented immediately upon the request to create or modify the resource.

When you delete a resource, your quota use is decremented during the next full recalculation of quota statistics for the project. A configurable amount of time determines how long it takes to reduce quota usage statistics to their current observed system value.

If project modifications exceed a quota usage limit, the server denies the action, and an appropriate error message is returned to the user explaining the quota constraint violated, and what their currently observed usage stats are in the system.

9.5. REQUESTS VERSUS LIMITS

When allocating compute resources, each container may specify a request and a limit value each for CPU, memory, and ephemeral storage. Quotas can restrict any of these values.

If the quota has a value specified for **requests.cpu** or **requests.memory**, then it requires that every incoming container make an explicit request for those resources. If the quota has a value specified for **limits.cpu** or **limits.memory**, then it requires that every incoming container specify an explicit limit for those resources.

9.6. SAMPLE RESOURCE QUOTA DEFINITIONS

core-object-counts.yaml

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: core-object-counts
spec:
  hard:
    configmaps: "10" 1
    persistentvolumeclaims: "4" 2
    replicationcontrollers: "20" 3
    secrets: "10" 4
    services: "10" 5
```

1 The total number of **ConfigMap** objects that can exist in the project.

2 The total number of persistent volume claims (PVCs) that can exist in the project.

3 The total number of replication controllers that can exist in the project.

4 The total number of secrets that can exist in the project.

5 The total number of services that can exist in the project.

openshift-object-counts.yaml

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: openshift-object-counts
spec:
  hard:
    openshift.io/imagestreams: "10" 1
```

1 The total number of image streams that can exist in the project.

compute-resources.yaml

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: compute-resources
spec:
  hard:
    pods: "4" 1
    requests.cpu: "1" 2
    requests.memory: 1Gi 3
    requests.ephemeral-storage: 2Gi 4
    limits.cpu: "2" 5
    limits.memory: 2Gi 6
    limits.ephemeral-storage: 4Gi 7
```

1 The total number of pods in a non-terminal state that can exist in the project.

2 Across all pods in a non-terminal state, the sum of CPU requests cannot exceed 1 core.

3 Across all pods in a non-terminal state, the sum of memory requests cannot exceed 1Gi.

4 Across all pods in a non-terminal state, the sum of ephemeral storage requests cannot exceed 2Gi.

5 Across all pods in a non-terminal state, the sum of CPU limits cannot exceed 2 cores.

6 Across all pods in a non-terminal state, the sum of memory limits cannot exceed 2Gi.

7 Across all pods in a non-terminal state, the sum of ephemeral storage limits cannot exceed 4Gi.

besteffort.yaml

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: besteffort
spec:
  hard:
    pods: "1" 1
  scopes:
  - BestEffort 2
```

1 The total number of pods in a non-terminal state with **BestEffort** quality of service that can exist in the project.

2 Restricts the quota to only matching pods that have **BestEffort** quality of service for either memory or CPU.

compute-resources-long-running.yaml

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: compute-resources-long-running
spec:
  hard:
    pods: "4" 1
    limits.cpu: "4" 2
    limits.memory: "2Gi" 3
    limits.ephemeral-storage: "4Gi" 4
  scopes:
  - NotTerminating 5
```

1 The total number of pods in a non-terminal state.

2 Across all pods in a non-terminal state, the sum of CPU limits cannot exceed this value.

3 Across all pods in a non-terminal state, the sum of memory limits cannot exceed this value.

4 Across all pods in a non-terminal state, the sum of ephemeral storage limits cannot exceed this value.

5 Restricts the quota to only matching pods where **spec.activeDeadlineSeconds** is set to **nil**. Build pods will fall under **NotTerminating** unless the **RestartNever** policy is applied.

compute-resources-time-bound.yaml

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: compute-resources-time-bound
spec:
  hard:
    pods: "2" 1
    limits.cpu: "1" 2
    limits.memory: "1Gi" 3
    limits.ephemeral-storage: "1Gi" 4
  scopes:
  - Terminating 5
```

1 The total number of pods in a non-terminal state.

2 Across all pods in a non-terminal state, the sum of CPU limits cannot exceed this value.

3 Across all pods in a non-terminal state, the sum of memory limits cannot exceed this value.

4 Across all pods in a non-terminal state, the sum of ephemeral storage limits cannot exceed this value.

5 Restricts the quota to only matching pods where **spec.activeDeadlineSeconds >=0**. For example, this quota would charge for build or deployer pods, but not long running pods like a web server or database.

storage-consumption.yaml

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: storage-consumption
spec:
  hard:
    persistentvolumeclaims: "10" 1
    requests.storage: "50Gi" 2
    gold.storageclass.storage.k8s.io/requests.storage: "10Gi" 3
    silver.storageclass.storage.k8s.io/requests.storage: "20Gi" 4
    silver.storageclass.storage.k8s.io/persistentvolumeclaims: "5" 5
    bronze.storageclass.storage.k8s.io/requests.storage: "0" 6
    bronze.storageclass.storage.k8s.io/persistentvolumeclaims: "0" 7
```

1 The total number of persistent volume claims in a project

2 Across all persistent volume claims in a project, the sum of storage requested cannot exceed this value.

3. Across all persistent volume claims in a project, the sum of storage requested in the gold storage class cannot exceed this value.

4. Across all persistent volume claims in a project, the sum of storage requested in the silver storage class cannot exceed this value.

5. Across all persistent volume claims in a project, the total number of claims in the silver storage class cannot exceed this value.

6. Across all persistent volume claims in a project, the sum of storage requested in the bronze storage class cannot exceed this value. When this is set to **0**, it means bronze storage class cannot request storage.

7. Across all persistent volume claims in a project, the sum of storage requested in the bronze storage class cannot exceed this value. When this is set to **0**, it means bronze storage class cannot create claims.

9.7. CREATING A QUOTA

To create a quota, first define the quota in a file, such as the examples in Sample Resource Quota Definitions. Then, create using that file to apply it to a project:

```
$ oc create -f <resource_quota_definition> [-n <project_name>]
```

For example:

```
$ oc create -f core-object-counts.yaml -n demoproject
```

9.7.1. Creating Object Count Quotas

You can create an object count quota for all OpenShift Dedicated standard namespaced resource types, such as **BuildConfig**, and **DeploymentConfig**. An object quota count places a defined quota on all standard namespaced resource types.

When using a resource quota, an object is charged against the quota if it exists in server storage. These types of quotas are useful to protect against exhaustion of storage resources.

To configure an object count quota for a resource, run the following command:

```
$ oc create quota <name> --hard=count/<resource>.<group>=<quota>,count/<resource>.<group>=
<quota>
```

For example:

```
$ oc create quota test --
hard=count/deployments.extensions=2,count/replicasets.extensions=4,count/pods=3,count/secrets=4
resourcequota "test" created

$ oc describe quota test
Name:                   test
Namespace:              quota
Resource                Used  Hard
--------                ----  ----
```

```
count/deployments.extensions  0     2
count/pods                    0     3
count/replicasets.extensions  0     4
count/secrets                 0     4
```

This example limits the listed resources to the hard limit in each project in the cluster.

9.8. VIEWING A QUOTA

You can view usage statistics related to any hard limits defined in a project's quota by navigating in the web console to the project's **Quota** page.

You can also use the CLI to view quota details:

1. First, get the list of quotas defined in the project. For example, for a project called **demoproject**:

   ```
   $ oc get quota -n demoproject
   NAME                AGE
   besteffort          11m
   compute-resources   2m
   core-object-counts  29m
   ```

2. Then, describe the quota you are interested in, for example the **core-object-counts** quota:

   ```
   $ oc describe quota core-object-counts -n demoproject
   Name:   core-object-counts
   Namespace:  demoproject
   Resource  Used Hard
   --------  ---- ----
   configmaps  3 10
   persistentvolumeclaims 0 4
   replicationcontrollers 3 20
   secrets   9 10
   services  2 10
   ```

9.9. ACCOUNTING FOR QUOTA IN DEPLOYMENT CONFIGURATIONS

If a quota has been defined for your project, see Deployment Resources for considerations on any deployment configurations.

9.10. REQUIRE EXPLICIT QUOTA TO CONSUME A RESOURCE

If a resource is not managed by quota, a user has no restriction on the amount of resource that can be consumed. For example, if there is no quota on storage related to the gold storage class, the amount of gold storage a project can create is unbounded.

For high-cost compute or storage resources, administrators may want to require an explicit quota be granted in order to consume a resource. For example, if a project was not explicitly given quota for storage related to the gold storage class, users of that project would not be able to create any storage of that type.

In order to require explicit quota to consume a particular resource, the following stanza should be added to the master-config.yaml.

```
admissionConfig:
  pluginConfig:
    ResourceQuota:
      configuration:
        apiVersion: resourcequota.admission.k8s.io/v1alpha1
        kind: Configuration
        limitedResources:
        - resource: persistentvolumeclaims 1
        matchContains:
        - gold.storageclass.storage.k8s.io/requests.storage 2
```

1. The group/resource to whose consumption is limited by default.
2. The name of the resource tracked by quota associated with the group/resource to limit by default.

In the above example, the quota system will intercept every operation that creates or updates a **PersistentVolumeClaim**. It checks what resources understood by quota would be consumed, and if there is no covering quota for those resources in the project, the request is denied. In this example, if a user creates a **PersistentVolumeClaim** that uses storage associated with the gold storage class, and there is no matching quota in the project, the request is denied.

9.11. MANAGING QUOTA ON PROJECT RESOURCES

For information on managing limits and quota on project resources, see Managing Projects.

9.12. KNOWN ISSUES

- Invalid objects can cause quota resources for a project to become exhausted. Quota is incremented in admission prior to validation of the resource. As a result, quota can be incremented even if the pod is not ultimately persisted. This will be resolved in a future release. (BZ1485375)

CHAPTER 10. SETTING LIMIT RANGES

10.1. OVERVIEW

A limit range, defined by a **LimitRange** object, enumerates compute resource constraints in a project at the pod, container, image, image stream, and persistent volume claim level, and specifies the amount of resources that a pod, container, image, image stream, or persistent volume claim can consume.

All resource create and modification requests are evaluated against each **LimitRange** object in the project. If the resource violates any of the enumerated constraints, then the resource is rejected. If the resource does not set an explicit value, and if the constraint supports a default value, then the default value is applied to the resource.

Core Limit Range Object Definition

```
apiVersion: "v1"
kind: "LimitRange"
metadata:
  name: "core-resource-limits" 1
spec:
  limits:
    - type: "Pod"
      max:
        cpu: "2" 2
        memory: "1Gi" 3
      min:
        cpu: "200m" 4
        memory: "6Mi" 5
    - type: "Container"
      max:
        cpu: "2" 6
        memory: "1Gi" 7
      min:
        cpu: "100m" 8
        memory: "4Mi" 9
      default:
        cpu: "300m" 10
        memory: "200Mi" 11
      defaultRequest:
        cpu: "200m" 12
        memory: "100Mi" 13
      maxLimitRequestRatio:
        cpu: "10" 14
```

1 The name of the limit range object.

2 The maximum amount of CPU that a pod can request on a node across all containers.

3 The maximum amount of memory that a pod can request on a node across all containers.

4 The minimum amount of CPU that a pod can request on a node across all containers.

5 The minimum amount of memory that a pod can request on a node across all containers.

⑥ The maximum amount of CPU that a single container in a pod can request.

⑦ The maximum amount of memory that a single container in a pod can request.

⑧ The minimum amount of CPU that a single container in a pod can request.

⑨ The minimum amount of memory that a single container in a pod can request.

⑩ The default amount of CPU that a container will be limited to use if not specified.

⑪ The default amount of memory that a container will be limited to use if not specified.

⑫ The default amount of CPU that a container will request to use if not specified.

⑬ The default amount of memory that a container will request to use if not specified.

⑭ The maximum amount of CPU burst that a container can make as a ratio of its limit over request.

For more information on how CPU and memory are measured, see Compute Resources.

OpenShift Dedicated Limit Range Object Definition

```
apiVersion: "v1"
kind: "LimitRange"
metadata:
  name: "openshift-resource-limits"
spec:
  limits:
    - type: openshift.io/Image
      max:
        storage: 1Gi ①
    - type: openshift.io/ImageStream
      max:
        openshift.io/image-tags: 20 ②
        openshift.io/images: 30 ③
    - type: "Pod"
      max:
        cpu: "2" ④
       memory: "1Gi" ⑤
       ephemeral-storage: "1Gi" ⑥
```

① The maximum size of an image that can be pushed to an internal registry.

② The maximum number of unique image tags per image stream’s spec.

③ The maximum number of unique image references per image stream’s status.

④ The maximum amount of CPU that a pod can request on a node across all containers.

⑤ The maximum amount of memory that a pod can request on a node across all containers.

⑥ The maximum amount of ephemeral storage that a pod can request on a node across all containers, if the ephemeral storage technology preview is enabled.

Both core and OpenShift Dedicated resources can be specified in just one limit range object. They are separated here into two examples for clarity.

10.1.1. Container Limits

Supported Resources:

- CPU
- Memory

Supported Constraints:

Per container, the following must hold true if specified:

Table 10.1. Container

Constraint	Behavior
Min	**Min[resource]** less than or equal to **container.resources.requests[resource]** (required) less than or equal to **container/resources.limits[resource]** (optional) If the configuration defines a **min** CPU, then the request value must be greater than the CPU value. A limit value does not need to be specified.
Max	**container.resources.limits[resource]** (required) less than or equal to **Max[resource]** If the configuration defines a **max** CPU, then you do not need to define a request value, but a limit value does need to be set that satisfies the maximum CPU constraint.
MaxLimitRequestRatio	**MaxLimitRequestRatio[resource]** less than or equal to (**container.resources.limits[resource]** / **container.resources.requests[resource]**) If a configuration defines a **maxLimitRequestRatio** value, then any new containers must have both a request and limit value. Additionally, OpenShift Dedicated calculates a limit to request ratio by dividing the limit by the request. This value should be a non-negative integer greater than 1. For example, if a container has **cpu: 500** in the **limit** value, and **cpu: 100** in the **request** value, then its limit to request ratio for**cpu** is **5**. This ratio must be less than or equal to the **maxLimitRequestRatio**.

Supported Defaults:

Default[resource]
Defaults **container.resources.limit[resource]** to specified value if none.

Default Requests[resource]
Defaults **container.resources.requests[resource]** to specified value if none.

10.1.2. Pod Limits

Supported Resources:

- CPU
- Memory

Supported Constraints:

Across all containers in a pod, the following must hold true:

Table 10.2. Pod

Constraint	Enforced Behavior
Min	**Min[resource]** less than or equal to **container.resources.requests[resource]** (required) less than or equal to **container.resources.limits[resource]** (optional)
Max	**container.resources.limits[resource]** (required) less than or equal to **Max[resource]**
MaxLimitRequestRatio	**MaxLimitRequestRatio[resource]** less than or equal to (**container.resources.limits[resource]** / **container.resources.requests[resource]**)

10.1.3. Image Limits

Supported Resources:

- Storage

Resource type name:

- **openshift.io/Image**

Per image, the following must hold true if specified:

Table 10.3. Image

Constraint	Behavior
Max	**image.dockerimagemetadata.size** less than or equal to**Max[resource]**

WARNING

The image size is not always available in the manifest of an uploaded image. This is especially the case for images built with Docker 1.10 or higher and pushed to a v2 registry. If such an image is pulled with an older Docker daemon, the image manifest will be converted by the registry to schema v1 lacking all the size information. No storage limit set on images will prevent it from being uploaded.

The issue is being addressed.

10.1.4. Image Stream Limits

Supported Resources:

- **openshift.io/image-tags**
- **openshift.io/images**

Resource type name:

- **openshift.io/ImageStream**

Per image stream, the following must hold true if specified:

Table 10.4. ImageStream

Constraint	Behavior
Max[openshift.io/image-tags]	**length(uniqueimagetags(imagestream.spec.tags))** less than or equal to **Max[openshift.io/image-tags]** **uniqueimagetags** returns unique references to images of given spec tags.
Max[openshift.io/images]	**length(uniqueimages(imagestream.status.tags))** less than or equal to **Max[openshift.io/images]** **uniqueimages** returns unique image names found in status tags. The name equals image's digest.

10.1.4.1. Counting of Image References

Resource **openshift.io/image-tags** represents unique image references. Possible references are an **ImageStreamTag**, an **ImageStreamImage** and a **DockerImage**. They may be created using commands **oc tag** and **oc import-image** or by using tag tracking. No distinction is made between internal and external references. However, each unique reference tagged in the image stream's specification is counted just once. It does not restrict pushes to an internal container image registry in any way, but is useful for tag restriction.

Resource **openshift.io/images** represents unique image names recorded in image stream status. It allows for restriction of a number of images that can be pushed to the internal registry. Internal and external references are not distinguished.

10.1.5. PersistentVolumeClaim Limits

Supported Resources:

- Storage

Supported Constraints:

Across all persistent volume claims in a project, the following must hold true:

Table 10.5. Pod

Constraint	Enforced Behavior
Min	Min[resource] ⇐ claim.spec.resources.requests[resource] (required)
Max	claim.spec.resources.requests[resource] (required) ⇐ Max[resource]

Limit Range Object Definition

```
{
  "apiVersion": "v1",
  "kind": "LimitRange",
  "metadata": {
    "name": "pvcs" 1
  },
  "spec": {
    "limits": [{
        "type": "PersistentVolumeClaim",
        "min": {
          "storage": "2Gi" 2
        },
        "max": {
          "storage": "50Gi" 3
        }
      }
    ]
  }
}
```

1 The name of the limit range object.

2 The minimum amount of storage that can be requested in a persistent volume claim

3 The maximum amount of storage that can be requested in a persistent volume claim

10.1.6. Project Limits

For information on enforcing different limits on the number of projects that your users can create, as well as on managing limits and quota on project resources, see Managing Projects.

10.2. CREATING A LIMIT RANGE

To apply a limit range to a project, create a limit range object definition on your file system to your desired specifications, then run:

```
$ oc create -f <limit_range_file> -n <project>
```

10.3. VIEWING LIMITS

You can view any limit ranges defined in a project by navigating in the web console to the project's **Quota** page.

You can also use the CLI to view limit range details:

1. First, get the list of limit ranges defined in the project. For example, for a project called **demoproject**:

   ```
   $ oc get limits -n demoproject
   NAME              AGE
   resource-limits   6d
   ```

2. Then, describe the limit range you are interested in, for example the **resource-limits** limit range:

   ```
   $ oc describe limits resource-limits -n demoproject
   Name:                           resource-limits
   Namespace:                      demoproject
   Type                            Resource                Min     Max     Default Request Default Limit   Max
   Limit/Request Ratio
   ----                            --------                ---     ---     --------------- -------------   -----------------------
   Pod                             cpu                     200m    2       -               -               -
   Pod                             memory                  6Mi     1Gi     -               -               -
   Container                       cpu                     100m    2       200m            300m            10
   Container                       memory                  4Mi     1Gi     100Mi           200Mi           -
   openshift.io/Image              storage                 -       1Gi     -               -               -
   openshift.io/ImageStream        openshift.io/image      -       12      -               -               -
   openshift.io/ImageStream        openshift.io/image-tags -       10      -               -               -
   ```

10.4. DELETING LIMITS

Remove any active limit range to no longer enforce the limits of a project:

```
$ oc delete limits <limit_name>
```

CHAPTER 11. NODE PROBLEM DETECTOR

11.1. OVERVIEW

The Node Problem Detector monitors the health of your nodes by finding certain problems and reporting these problems to the API server. The detector runs as a daemonset on each node.

> **IMPORTANT**
>
> The Node Problem Detector is a Technology Preview feature only.

The Node Problem Detector reads system logs and watches for specific entries and makes these problems visible to the control plane, which you can view using OpenShift Dedicated commands, such as **oc get node** and **oc get event**. You could then take action to correct these problems as appropriate or capture the messages using a tool of your choice, such as the OpenShift Dedicated log monitoring. Detected problems can be in one of the following categories:

- **NodeCondition**: A permanent problem that makes the node unavailable for pods. The node condition will not be cleared until the host is rebooted.
- **Event**: A temporary problem that has limited impact on a node, but is informative.

The Node Problem Detector can detect:

- container runtime issues:
  - unresponsive runtime daemons
- hardware issues:
  - bad CPU
  - bad memory
  - bad disk
- kernel issues:
  - kernel deadlock conditions
  - corrupted file systems
  - unresponsive runtime daemons
- infrastructure daemon issues:
  - NTP service outages

11.2. EXAMPLE NODE PROBLEM DETECTOR OUTPUT

The following examples show output from the Node Problem Detector watching for kernel deadlock node condition on a specific node. The command uses **oc get node** to watch a specific node filtering for a **KernelDeadlock** entry in a log.

```
# oc get node <node> -o yaml | grep -B5 KernelDeadlock
```

Sample Node Problem Detector output with no issues

```
message: kernel has no deadlock
reason: KernelHasNoDeadlock
status: false
type: KernelDeadLock
```

Sample output for KernelDeadLock condition

```
message: task docker:1234 blocked for more than 120 seconds
reason: DockerHung
status: true
type: KernelDeadLock
```

This example shows output from the Node Problem Detector watching for events on a node. The following command uses **oc get event** against the **default** project watching for events listed in the **kernel-monitor.json** section of the Node Problem Detector configuration map.

```
# oc get event -n default --field-selector=source=kernel-monitor --watch
```

Sample output showing events on nodes

```
LAST SEEN                    FIRST SEEN                   COUNT NAME     KIND  SUBOBJECT TYPE
REASON      SOURCE                    MESSAGE
2018-06-27 09:08:27 -0400 EDT   2018-06-27 09:08:27 -0400 EDT 1     my-node1 node
Warning TaskHunk    kernel-monitor.my-node1  docker:1234 blocked for more than 300 seconds
2018-06-27 09:08:27 -0400 EDT   2018-06-27 09:08:27 -0400 EDT 3     my-node2 node
Warning KernelOops  kernel-monitor.my-node2  BUG: unable to handle kernel NULL pointer
deference at nowhere
2018-06-27 09:08:27 -0400 EDT   2018-06-27 09:08:27 -0400 EDT 1     my-node1 node
Warning KernelOops  kernel-monitor.my-node2  divide error 0000 [#0] SMP
```

NOTE

The Node Problem Detector consumes resources. If you use the Node Problem Detector, make sure you have enough nodes to balance cluster performance.

11.3. INSTALLING THE NODE PROBLEM DETECTOR

If **openshift_node_problem_detector_install** was set to **true** in the ***/etc/ansible/hosts*** inventory file, the installation creates a Node Problem Detector daemonset by default and creates a project for the detector, called **openshift-node-problem-detector**.

NOTE

Because the Node Problem Detector is in Technology Preview, the **openshift_node_problem_detector_install** is set to **false** by default. You must manually change the parameter to **true** when installing the Node Problem Detector.

If the Node Problem Detector is not installed, change to the playbook directory and run the **openshift-node-problem-detector/config.yml** playbook to install Node Problem Detector:

```
$ cd /usr/share/ansible/openshift-ansible
$ ansible-playbook playbooks/openshift-node-problem-detector/config.yml
```

11.4. CUSTOMIZING DETECTED CONDITIONS

You can configure the Node Problem Detector to watch for any log string by editing the Node Problem Detector configuration map.

Sample Node Problem Detector Configuration Map

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: node-problem-detector
data:
  docker-monitor.json: |  ①
    {
        "plugin": "journald",  ②
        "pluginConfig": {
                "source": "docker"
        },
        "logPath": "/host/log/journal",  ③
        "lookback": "5m",
        "bufferSize": 10,
        "source": "docker-monitor",
        "conditions": [],
        "rules": [              ④
                {
                        "type": "temporary",  ⑤
                        "reason": "CorruptDockerImage",  ⑥
                        "pattern": "Error trying v2 registry: failed to register layer: rename
/var/lib/docker/image/(.+) /var/lib/docker/image/(.+): directory not empty.*"  ⑦
                }
        ]
    }
  kernel-monitor.json: |  ⑧
    {
        "plugin": "journald",  ⑨
        "pluginConfig": {
                "source": "kernel"
        },
        "logPath": "/host/log/journal",  ⑩
        "lookback": "5m",
        "bufferSize": 10,
        "source": "kernel-monitor",
        "conditions": [           ⑪
                {
                        "type": "KernelDeadlock",  ⑫
                        "reason": "KernelHasNoDeadlock",  ⑬
                        "message": "kernel has no deadlock"  ⑭
                }
        ],
        "rules": [
                {
                        "type": "temporary",
                        "reason": "OOMKilling",
                        "pattern": "Kill process \\d+ (.+) score \\d+ or sacrifice child\\nKilled process \\d+ (.+) total-vm:\\d+kB, anon-rss:\\d+kB, file-rss:\\d+kB"
                },
                {
                        "type": "temporary",
                        "reason": "TaskHung",
                        "pattern": "task \\S+:\\w+ blocked for more than \\w+ seconds\\."
                },
                {
                        "type": "temporary",
                        "reason": "UnregisterNetDevice",
                        "pattern": "unregister_netdevice: waiting for \\w+ to become free. Usage count = \\d+"
                },
                {
                        "type": "temporary",
                        "reason": "KernelOops",
                        "pattern": "BUG: unable to handle kernel NULL pointer dereference at .*"
                },
                {
                        "type": "temporary",
                        "reason": "KernelOops",
                        "pattern": "divide error: 0000 \\[#\\d+\\] SMP"
                },
                {
                        "type": "permanent",
                        "condition": "KernelDeadlock",
                        "reason": "AUFSUmountHung",
                        "pattern": "task umount\\.aufs:\\w+ blocked for more than \\w+ seconds\\."
                },
                {
                        "type": "permanent",
                        "condition": "KernelDeadlock",
                        "reason": "DockerHung",
                        "pattern": "task docker:\\w+ blocked for more than \\w+ seconds\\."
                }
        ]
    }
```

1 Rules and conditions that apply to container images.

2 9 Monitoring services, in a comma-separated list.

3 10 Path to the monitoring service log.

4 11 List of events to be monitored.

5 12 Label to indicate the error is an event (**temporary**) or NodeCondition (**permanent**).

6 13 Text message to describe the error.

7 14 Error message that the Node Problem Detector watches for.

8 Rules and conditions that apply to the kernel.

To configure the Node Problem Detector, add or remove problem conditions and events.

1. Edit the Node Problem Detector configuration map with a text editor.

   ```
   oc edit configmap -n openshift-node-problem-detector node-problem-detector
   ```

2. Remove, add, or edit any node conditions or events as needed.

   ```
   {
          "type": <`temporary` or `permanent`>,
          "reason": <free-form text describing the error>,
          "pattern": <log message to watch for>
   },
   ```

   For example:

   ```
   {
          "type": "temporary",
          "reason": "UnregisterNetDevice",
          "pattern": "unregister_netdevice: waiting for \\w+ to become free. Usage count = \\d+"
   },
   ```

3. Restart running pods to apply the changes. To restart pods, you can delete all existing pods:

   ```
   # oc delete pods -n openshift-node-problem-detector -l name=node-problem-detector
   ```

4. To display Node Problem Detector output to standard output (stdout) and standard error (stderr) add the following to the configuration map:

   ```
   spec:
     template:
       spec:
         containers:
         - name: node-problem-detector
           command:
           - node-problem-detector
           - --alsologtostderr=true ①
           - --log_dir="/tmp" ②
           - --system-log-monitors=/etc/npd/kernel-monitor.json,/etc/npd/docker-monitor.json ③
   ```

   ① Sends the output to standard output (stdout).

   ② Path to the error log.

   ③ Comma-separated path to the plug-in configuration files.

11.5. VERIFYING THAT THE NODE PROBLEM DETECTOR IS RUNNING

To verify that the Node Problem Detector is active:

- Run the following command to get the name of the Problem Node Detector pod:

```
# oc get pods -n openshift-node-problem-detector

NAME                               READY     STATUS    RESTARTS   AGE
node-problem-detector-8z8r8   1/1       Running   0          1h
node-problem-detector-nggjv   1/1       Running   0          1h
```

- Run the following command to view log information on the Problem Node Detector pod:

```
# oc logs -n openshift-node-problem-detector <pod_name>
```

The output should be similar to the following:

```
# oc logs -n openshift-node-problem-detector node-problem-detector-c6kng
I0416 23:22:00.641354       1 log_monitor.go:63] Finish parsing log monitor config file:
{WatcherConfig:{Plugin:journald PluginConfig:map[source:kernel] LogPath:/host/log/journal
Lookback:5m} BufferSize:10 Source:kernel-monitor DefaultConditions:
[{Type:KernelDeadlock Status:false Transition:0001-01-01 00:00:00 +0000 UTC
Reason:KernelHasNoDeadlock Message:kernel has no deadlock}]
```

- Test the Node Problem Detector by simulating an event on the node:

```
# echo "kernel: divide error: 0000 [#0] SMP." >> /dev/kmsg
```

- Test the Node Problem Detector by simulating a condition on the node:

```
# echo "kernel: task docker:7 blocked for more than 300 seconds." >> /dev/kmsg
```

11.6. UNINSTALL THE NODE PROBLEM DETECTOR

To uninstall the Node Problem Detector:

1. Add following options in Ansible inventory file:

```
[OSEv3:vars]
openshift_node_problem_detector_state=absent
```

2. Change to the playbook directory and run the ***config.yml*** Ansible playbook:

```
$ cd /usr/share/ansible/openshift-ansible
$ ansible-playbook playbooks/openshift-node-problem-detector/config.yml
```