



OpenShift Enterprise 2 Administration Guide

A Guide to OpenShift Enterprise Operation and Administration

Red Hat OpenShift Documentation
Team

OpenShift Enterprise 2 Administration Guide

A Guide to OpenShift Enterprise Operation and Administration

Red Hat OpenShift Documentation Team

Legal Notice

Copyright © 2017 Red Hat.

This document is licensed by Red Hat under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](https://creativecommons.org/licenses/by-sa/3.0/). If you distribute this document, or a modified version of it, you must provide attribution to Red Hat, Inc. and provide a link to the original. If the document is modified, all Red Hat trademarks must be removed.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

The Administration Guide provides information on advanced configuration and administration of OpenShift Enterprise deployments, and includes the following information: Platform administration User administration Cartridge management Resource monitoring and management Monitoring with the administration console Command reference for broker and node hosts This guide is intended for experienced system administrators.

Table of Contents

Chapter 1. Introduction to OpenShift Enterprise	4
1.1. What's New in Current Release	4
1.2. Upgrading OpenShift Enterprise	4
1.3. Migrating from RHN Classic to RHSM	4
Chapter 2. Platform Administration	7
2.1. Changing the Front-end HTTP Configuration for Existing Deployments	7
2.2. Enabling User Login Normalization	8
2.3. Allowing Multiple HAProxies on a Node Host	9
2.4. Enabling Support for High-Availability Applications	9
2.5. Creating Environment Variables on Node Hosts	11
2.6. Controlling Direct SSL Connections to Gears	11
2.7. Setting Gear Supplementary Groups	12
2.8. Banning IP Addresses That Overload Applications	13
2.9. Enabling Maintenance Mode	13
2.10. Backup and Recovery	14
2.10.1. Backing Up Broker Host Files	14
2.10.2. Backing Up Node Host Files	14
2.10.3. Recovering Failed Node Hosts	15
2.10.4. Recreating /etc/passwd Entries	17
2.11. Component Timeout Value Locations	17
2.12. Enabling Network Isolation for Gears	19
Chapter 3. User Administration	21
3.1. Creating a User	21
3.2. Removing User Applications	21
3.3. Removing User Data	22
3.4. Removing a User	22
3.5. Enabling Users to Add a Kerberos Principal SSH Key	23
3.6. Setting Default Maximum Number of Domains per User	23
3.7. Managing Custom Domain Aliases	23
3.8. Determining Gear Ownership	24
Chapter 4. Team and Global Team Management	25
4.1. Setting the Maximum Number of Teams for Specific Users	25
4.2. Creating Global Teams and Synchronizing with LDAP Groups	25
4.2.1. Encrypting an LDAP Global Team Connection	27
4.2.2. Enabling Global Team Visibility	28
Chapter 5. Cartridge Management	29
5.1. Managing Cartridges on Broker Hosts	29
5.1.1. Importing, Activating, and Deactivating Cartridges	30
5.1.2. Migrating and Upgrading Existing Applications to Active Cartridges	31
5.1.3. Removing Unused Inactive Cartridges	32
5.2. Installing and Removing Custom and Community Cartridges	33
5.3. Upgrading Custom and Community Cartridges	36
5.4. Adding QuickStarts to the Management Console	37
5.5. Disabling Downloadable Cartridges	39
5.6. Disabling Obsolete Cartridges	39
Chapter 6. Resource Management	41
6.1. Adding or Modifying Gear Profiles	41
6.2. Capacity Planning and Districts	42

6.2.1. Hierarchy of OpenShift Enterprise Entities	42
6.2.2. Purpose of Districts	43
6.2.3. Gear Capacity Planning	43
6.2.3.1. Gear Capacity Planning for Nodes	43
6.2.3.2. Gear Capacity Planning for Districts	44
6.3. Managing Districts	45
6.3.1. Enabling Districts	45
6.3.2. Creating and Populating Districts	46
6.3.3. Viewing District Information	47
6.3.4. Viewing Capacity Statistics	48
6.3.5. Moving Gears Between Nodes	49
6.3.6. Removing Nodes from Districts	49
6.3.7. Removing Districts	50
6.4. Managing Regions and Zones	50
6.4.1. Creating a Region with Zones	51
6.4.2. Tagging a Node with a Region and Zone	51
6.4.3. Setting the Default Region For New Applications	52
6.4.4. Disabling Region Selection	52
6.4.5. Additional Region and Zone Tasks	53
6.5. Gear Placement Algorithm	53
6.6. Setting Default Gear Quotas and Sizes	55
6.7. Setting Gear Quotas and Sizes for Specific Users	56
6.8. Restricting Gear Sizes for Cartridges	57
6.9. Viewing Resource Usage on a Node	58
6.10. Enforcing Low Tenancy on Nodes	58
6.11. Managing Capacity on Broker Hosts	58
Chapter 7. Administration Console	60
7.1. Understanding the System Overview	60
7.2. Viewing Gear Profiles	60
7.3. Viewing Suggestions	62
7.4. Searching for Entities	62
7.5. Viewing Statistics	63
7.6. Configuring Suggestions	63
7.7. Loading Capacity Data from a File	63
7.8. Exposed Data	64
Chapter 8. Monitoring	65
8.1. General System Checks	65
8.2. Response Times for Administrative Actions	65
8.3. Testing a Path Through the Whole System	66
8.4. Monitoring Broker Activity	66
8.4.1. Default Broker Log File Locations	66
8.4.2. Verifying Functionality with Administration Commands	66
8.5. Monitoring Node and Gear Activity	67
8.5.1. Default Node Log File Locations	67
8.5.2. Enabling Application and Gear Context in Node Component Logs	68
8.5.3. Viewing Application Details	68
8.5.4. The Watchman Tool	70
8.5.4.1. Enabling Watchman	70
8.5.4.2. Supported Watchman Plug-ins	70
8.5.4.3. Configuring Watchman	72
8.5.5. Testing Node Host Functionality	72
8.5.6. Validating Gears	72

8.5.6. Validating Gears	73
8.5.7. Node Capacity	73
8.6. Monitoring Management Console Activity	73
8.6.1. Default Management Console Log File Locations	73
8.7. Usage Tracking	73
8.7.1. Setting Tracked and Untracked Storage	74
8.7.2. Viewing Accumulated Usage Data	75
8.8. Enabling Syslog	77
8.8.1. Enabling Syslog for Broker Components	77
8.8.2. Enabling Syslog for Node Components	77
8.8.3. Enabling Syslog for Cartridge Logs from Gears	79
8.8.4. Enabling Syslog for Management Console Components	84
Chapter 9. Command Reference	86
9.1. Broker Administration Commands	86
9.1.1. oo-accept-broker	86
9.1.2. oo-accept-systems	86
9.1.3. oo-admin-chk	87
9.1.4. oo-admin-clear-pending-ops	88
9.1.5. oo-admin-console-cache	88
9.1.6. oo-admin-broker-auth	89
9.1.7. oo-admin-broker-cache	89
9.1.8. oo-admin-ctl-app	89
9.1.9. oo-admin-ctl-authorization	89
9.1.10. oo-admin-ctl-district	89
9.1.11. oo-admin-ctl-domain	90
9.1.12. oo-admin-ctl-region	90
9.1.13. oo-admin-ctl-team	90
9.1.14. oo-admin-ctl-usage	91
9.1.15. oo-admin-ctl-user	92
9.1.16. oo-admin-move	93
9.1.17. oo-admin-repair	93
9.1.18. oo-admin-upgrade	93
9.1.19. oo-admin-usage	93
9.1.20. oo-admin-ctl-cartridge	94
9.1.21. oo-register-dns	94
9.2. Node Administration Commands	95
9.2.1. oo-accept-node	95
9.2.2. oo-admin-ctl-gears	95
9.2.3. oo-idler-stats	95
9.2.4. Idler Commands	95
9.2.4.1. oo-last-access	95
9.2.4.2. oo-auto-idler	96
Appendix A. Revision History	97

Chapter 1. Introduction to OpenShift Enterprise

OpenShift Enterprise by Red Hat is a Platform as a Service (PaaS) that provides developers and IT organizations with an auto-scaling, cloud application platform for deploying new applications on secure, scalable resources with minimal configuration and management overhead. OpenShift Enterprise supports a wide selection of programming languages and frameworks, such as Java, Ruby, and PHP. Integrated developer tools, such as Eclipse integration, JBoss Developer Studio, and Jenkins, support the application life cycle.

Built on Red Hat Enterprise Linux, OpenShift Enterprise provides a secure and scalable multi-tenant operating system for today's enterprise-class applications while providing integrated application runtimes and libraries.

OpenShift Enterprise brings the OpenShift PaaS platform to customer data centers, enabling organizations to implement a private PaaS that meets security, privacy, compliance, and governance requirements.

1.1. What's New in Current Release

For a complete list of all the new features available in the current release of OpenShift Enterprise, see the current edition of the *OpenShift Enterprise Release Notes* at <https://access.redhat.com/site/documentation>. New features that are available in the current release are documented in the respective sections of this book.

1.2. Upgrading OpenShift Enterprise

OpenShift Enterprise relies on a complex set of dependencies; to avoid problems, caution is required when applying software upgrades to broker and node hosts.

For bug fixes and other targeted changes, updated RPMs are released in existing channels. Read errata advisories carefully for instructions on how to safely apply upgrades and details of required service restarts and configuration changes. For example, when upgrading *rubygem* packages required by the broker application, it is necessary to restart the **openshift-broker** service. This step regenerates the **bundler** utility's **Gemfile.lock** file and allows the broker application and related administrative commands to use the updated gems. See the latest *OpenShift Enterprise Deployment Guide* at <https://access.redhat.com/site/documentation> for instructions on how to apply asynchronous errata updates.

For systemic upgrades from previous versions of OpenShift Enterprise requiring formal migration scripts and lockstep package updates, see the latest *OpenShift Enterprise Deployment Guide* at <https://access.redhat.com/site/documentation> for instructions on how to use the **ose-upgrade** tool.

1.3. Migrating from RHN Classic to RHSM

The Red Hat Network (RHN) Classic hosted subscription service on the Red Hat Customer Portal is being deprecated. The [product life cycles](#) for OpenShift Enterprise major versions 1 and 2 end before the RHN Classic end date. However, existing OpenShift Enterprise host systems that were registered using the RHN Classic hosted subscription method can be migrated to Red Hat Subscription Management (RHSM). This applies to systems running OpenShift Enterprise version 1 or 2 that host any OpenShift Enterprise component: broker, node, message server, database server, or combinations of these.

Red Hat recommends performing this migration on any affected systems as soon as possible. Once the transition to Red Hat Subscription Management for all Red Hat products is completed, then RHN Classic will no longer provide services to registered systems. More information on this transition can be found at <https://access.redhat.com/rhn-to-rhsm>.

The migration process unregisters the system from Red Hat Network (RHN) Classic, then registers it with

Red Hat Subscription Management and attaches subscriptions using the **subscription-manager** CLI. The migration tools are contained in the *subscription-manager-migration* package. An additional package, *subscription-manager-migration-data*, is required to map the RHN Classic channels to Red Hat Subscription Management product certificates.

The [Red Hat Subscription Management - Migrating from RHN Classic](#) guide provides detailed instructions on migrating Red Hat Enterprise Linux systems to Red Hat Subscription Management.

The following procedure provides a basic overview of this migration and is catered to Red Hat Enterprise Linux systems hosting OpenShift Enterprise components.

Procedure 1.1. To Migrate from RHN Classic to RHSM:

1. Use the **oo-admin-yum-validator** validation tool to verify that the system's **yum** configuration for the current subscription method is valid for the installed OpenShift Enterprise version and components. Use the **-o** option for the version and the **-r** option for the components.

Example 1.1. Verifying a Host With the Validation Tool

The following example is for an OpenShift Enterprise 2.2 broker host:

```
# oo-admin-yum-validator -o 2.2 -r broker
```

If run without options, the validation tool attempts to detect the installed version and components. If any problems are reported, fix them manually or use the validation tool's **--fix** or **--fix-all** options to attempt to fix them automatically.

Additional details on running the validation tool can be found in [this knowledgebase article](#) or in the **oo-admin-yum-validator** man page.

2. Install the migration tool packages:

```
# yum install subscription-manager-migration subscription-manager-migration-data
```

3. Use the **rhn-migrate-classic-to-rhsm** tool to initiate the migration. This tool has many options available, including registering to on-premise services and manually selecting subscriptions. If run without options, this tool migrates the system profile, registers the system with Red Hat Subscription Management, and automatically attaches the system to the best-matched subscriptions:

```
# rhn-migrate-classic-to-rhsm
```

Consult the [Red Hat Subscription Management - Migrating from RHN Classic](#) guide or the **rhn-migrate-classic-to-rhsm** man page for details on additional options that may be relevant to your organization and environment.



Note

A [known issue](#), which will be fixed in Red Hat Enterprise Linux 6.6, prevents the migration tool from automatically enabling the required channels on OpenShift Enterprise 2.1 systems. You can work around this issue by using the migration tool with the **--force** and **--no-auto** options; this continues registering the system to Red Hat Subscription Management, but does not automatically attach a subscription. Once the migration is complete, manually attach the desired OpenShift Enterprise subscription using the **subscription-manager** tool:

```
# subscription-manager attach --pool Pool_ID
```

4. After the migration completes, use the **subscription-manager** tool to list information about the migration including the previous system ID:

Example 1.2. Listing Migration Information

```
# subscription-manager facts --list | grep migr
migration.classic_system_id: 09876
migration.migrated_from: rhn_hosted_classic
migration.migration_date: 2012-09-14T14:55:29.280519
```

5. Use the **oo-admin-yum-validator** validation tool again to verify that the system's **yum** configuration is still valid under the new subscription method, and correct any issues that are reported.

Chapter 2. Platform Administration

This chapter covers tasks related to the various OpenShift Enterprise platform components on broker and node hosts.

2.1. Changing the Front-end HTTP Configuration for Existing Deployments

Starting with OpenShift Enterprise 2.2, the **Apache Virtual Hosts** front-end HTTP proxy is the default for new deployments. If your nodes are currently using the previous default, the **Apache mod_rewrite** plug-in, you can use the following procedure to change the front-end configuration of your existing deployment.

Configuring the HTTP front-end for an already-deployed OpenShift Enterprise instance after it has been configured is possible, but Red Hat recommends caution when doing so. You must first prevent any front-end changes made by the broker, such as creating or deleting application gears, on the node host containing the applications during this configuration change. Performing a verified backup of the node host before commencing configuration is highly recommended.

See the [OpenShift Enterprise Deployment Guide](#) for more information about installing and configuring front-end HTTP server plug-ins.

Procedure 2.1. To Change the Front-end HTTP Configuration on an Existing Deployment:

1. To prevent the broker from making any changes to the front-end during this procedure, stop the `ruby193-mcollective` service on the node host:

```
# service ruby193-mcollective stop
```

Then set the following environment variable to prevent each front-end change from restarting the `httpd` service:

```
# export APACHE_HTTPD_DO_NOT_RELOAD=1
```

2. Back up the existing front-end configuration. You will use this backup to restore the complete state of the front end after the process is complete. Replace *filename* with your desired backup storage location:

```
# oo-frontend-plugin-modify --save > filename
```

3. Delete the existing front-end configuration:

```
# oo-frontend-plugin-modify --delete
```

4. Remove and install the front-end plug-in packages as necessary:

```
# yum remove rubygem-openshift-origin-frontend-apache-mod-rewrite  
# yum -y install rubygem-openshift-origin-frontend-apache-vhost
```

5. Replicate any Apache customizations reliant on the old plug-in onto the new plug-in, then restart the `httpd` service:

```
# service httpd restart
```

- Change the **OPENSIFT_FRONTEND_HTTP_PLUGINS** value in the `/etc/openshift/node.conf` file from **openshift-origin-frontend-apache-mod-rewrite** to **openshift-origin-frontend-apache-vhost**:

```
OPENSIFT_FRONTEND_HTTP_PLUGINS="openshift-origin-frontend-apache-vhost"
```

- Un-set the previous environment variable to restarting the httpd service as normal after any front-end changes:

```
# export APACHE_HTTPD_DO_NOT_RELOAD=""
```

- Restart the MCollective service:

```
# service ruby193-mcollective restart
```

- Restore the HTTP front-end configuration from the backup you created in step one:

```
# oo-frontend-plugin-modify --restore < filename
```

2.2. Enabling User Login Normalization

You can enforce normalization for user logins by enabling a selection of default or custom methods on the broker. This is helpful when using authentication methods like LDAP or Kerberos that can be case-sensitive or use a domain in the login. Without normalization, logins with different letter cases or domain suffixes are stored by the broker as distinct user accounts.

For example, when normalization is enabled using the **lowercase** method, a user logging in as **JDoe** is authenticated using the configured authentication method, then the login is normalized as **jdoe** by the broker to access the **jdoe** user account on OpenShift Enterprise. When normalization is not enabled, a user logging in as **JDoe** is authenticated using the configured authentication method and accesses the **JDoe** user account on OpenShift Enterprise, while a user logging in as **jdoe** ultimately accesses a separate **jdoe** user account.



Warning

Existing logins are not automatically updated when normalization settings are changed. As a result, it is possible for existing user accounts to no longer be accessible if the login was not previously normalized.

The following default methods are available:

Table 2.1. Available Default User Login Normalization Methods

Method	Function
strip	Removes any additional spaces on either side of the login.
lowercase	Changes all characters to lowercase. For example: JDoe --> jdoe
remove_domain	Removes a domain suffix. For example: jdoe@example.com --> jdoe

To enable normalization, edit the `/etc/openshift/broker.conf` file on the broker host and provide one or more methods in the `NORMALIZE_USERNAME_METHOD` parameter using a comma-separated list:

Example 2.1. Setting User Login Normalization Methods

```
NORMALIZE_USERNAME_METHOD="lowercase,remove_domain"
```

Restart the broker service for any changes to take effect:

```
service openshift-broker restart
```

2.3. Allowing Multiple HAProxies on a Node Host

The `ALLOW_MULTIPLE_HAPROXY_ON_NODE` setting, located in the `/etc/openshift/broker.conf` file, is set to `false` by default. In production environments, Red Hat recommends to leave this setting as default. If two or more HAProxies for a single application reside on the same node host, the front-end Apache will map the DNS or alias to one HAProxy gear and not for the remaining HAProxy gears. If, for example, you have only one node host and wish to enable scalability, changing the `ALLOW_MULTIPLE_HAPROXY_ON_NODE` setting to `true` allows multiple HAProxy gears for the same application to reside on the same node host.

Procedure 2.2. To Allow Multiple HAProxies on a Single Node:

1. Open the `/etc/openshift/broker.conf` file on the broker host and set the `ALLOW_MULTIPLE_HAPROXY_ON_NODE` value to `true`:

```
ALLOW_MULTIPLE_HAPROXY_ON_NODE="true"
```

2. Restart the `openshift-broker` service:

```
# service openshift-broker restart
```

2.4. Enabling Support for High-Availability Applications

If you have configured an external routing layer, either the included sample or your own, to route application traffic, you must enable support for high-availability applications and configure specific DNS management options before developers can take advantage of these features.



Note

See the [OpenShift Enterprise Deployment Guide](#) for more information on using an external routing layer for high-availability applications, including how to configure the sample routing plug-in and routing daemon.

Procedure 2.3. To Enable Support for High-Availability Applications:

1. To allow scalable applications to become highly available using the configured external router, edit the `/etc/openshift/broker.conf` file on the broker host and set the **ALLOW_HA_APPLICATIONS** parameter to `"true"`:

```
ALLOW_HA_APPLICATIONS="true"
```

Note that this parameter controls whether high-availability applications are allowed in general, but does not adjust user account capabilities. User account capabilities are discussed in a later step.

2. A scaled application that is not highly available uses the following URL form:

```
http://${APP_NAME}-${DOMAIN_NAME}.${CLOUD_DOMAIN}
```

When high-availability is enabled, HAproxy instances are deployed in multiple gears of the application, which are spread across multiple node hosts. In order to load balance user requests, a high-availability application requires a new high-availability DNS name that points to the external routing layer rather than directly to the application head gear. The routing layer then forwards requests directly to the application's HAproxy instances, which are then distributed to the framework gears. In order to create DNS entries for high-availability applications that point to the routing layer, OpenShift Enterprise adds either a prefix or suffix, or both, to the regular application name:

```
http://${HA_DNS_PREFIX}${APP_NAME}-  
${DOMAIN_NAME}${HA_DNS_SUFFIX}.${CLOUD_DOMAIN}
```

To change the prefix or suffix used in the high-availability URL, you can modify the **HA_DNS_PREFIX** or **HA_DNS_SUFFIX** parameters:

```
HA_DNS_PREFIX="ha - "  
HA_DNS_SUFFIX=""
```

If you modify the **HA_DNS_PREFIX** parameter and are using the sample routing daemon, ensure this parameter and the **HA_DNS_SUFFIX** parameter in the `/etc/openshift/routing-daemon.conf` file are set to the same value.

3. DNS entries for high-availability applications can either be managed by OpenShift Enterprise or externally. By default, this parameter is set to `"false"`, which means the entries must be created externally; failure to do so could prevent the application from receiving traffic through the external routing layer. To allow OpenShift Enterprise to create and delete these entries when applications are created and deleted, set the **MANAGE_HA_DNS** parameter to `"true"`:

```
MANAGE_HA_DNS="true"
```

Then set the **ROUTER_HOSTNAME** parameter to the public hostname of the external routing layer, which the DNS entries for high-availability applications point to. Note that the routing layer host must be resolvable by the broker:

```
ROUTER_HOSTNAME="www.example.com"
```

4. For developers to enable high-availability support with their scalable applications, they must have the **HA allowed** capability enabled on their account. By default, the **DEFAULT_ALLOW_HA** parameter is set to `"false"`, which means user accounts are created with the **HA allowed** capability initially disabled. To have this capability enabled by default for new user accounts, set **DEFAULT_ALLOW_HA** to `"true"`:

```
DEFAULT_ALLOW_HA="true"
```

You can also adjust the **HA allowed** capability per user account using the **oo-admin-ctl-user** command with the **--allowha** option:

```
# oo-admin-ctl-user -l user --allowha true
```

- To make any changes made to the `/etc/openshift/broker.conf` file take effect, restart the broker service:

```
# service openshift-broker restart
```

Note that this procedure only enables the support for high-availability applications. See the [OpenShift Enterprise User Guide](#) for a procedure on how a user can make an application highly-available.

2.5. Creating Environment Variables on Node Hosts

With the release of OpenShift Enterprise 2.1, you can provide environment variables for all applications on a node host by specifying them in the `/etc/openshift/env` directory. By creating a file in the `/etc/openshift/env` directory on a node host, an environment variable is created with the same name as the file name, and the value being set to the contents of the file.

Environment variables set in the `/etc/openshift/env` directory are only set for gear users, and not for system services or other users on the node host. For example, the MCollective service does not have access to these settings during the gear and cartridge creation process.

Application owners can use the **rhc env set** command to override any environment variables set in the `/etc/openshift/env` directory.

Procedure 2.4. Creating Environment Variables on a Node Host

- Create a new file in the `/etc/openshift/env` directory on the node hosts that you want the environment variable set. For example, to allow applications to use an external database, set an external database environment variable `EXT_DB_CONNECTION_URL` with the value of `mysql://host.example.com:3306/`

```
# echo mysql://host.example.com:3306/ >
/etc/openshift/env/EXT_DB_CONNECTION_URL
```

- To make the changes take effect for existing applications, ask affected application owners to restart their applications by running the following commands:

```
$ rhc app stop -a appname
$ rhc app start -a appname
```

Alternatively, you can restart all gears on affected node hosts. The downtime caused by restarting all gears is minimal and around a few seconds.

```
# oo-admin-ctl-gears restartall
```

2.6. Controlling Direct SSL Connections to Gears

In some environments, regulations may require encrypted connections between the client and the server, therefore the need for SSL connections. SSL connections to gears are either allowed, denied, or forced. By default, direct SSL connections to gears are allowed if a cartridge supports the feature and is currently only available for customized cartridges.

Enabling SSL connection allows request to the HTTP front-end to be routed as https to applications. Non-HTTP front-end ports, for example database ports, can have **SSL_TO_GEAR** enabled to be exposed for direct connections using the **PROXY_PORTS** parameter. However, this requires setting up an external router.



Note

As an alternative, you can use a custom cartridge that supports SNI proxy to allow SSL connections over non-HTTP ports. SNI proxy uses a local proxy running on the node host and does not require an external router. Specific ports must be configured to route SSL to gears. See the *OpenShift Enterprise Deployment Guide* at <https://access.redhat.com/site/documentation> for more information. Websocket does not handle SSL connections.

Configure the **SSL_ENDPOINT** setting in the `/etc/openshift/broker.conf` file to one of the following options to control access to cartridges that specify direct connections to gears:

allow

If the cartridge being added to a new application specifies direct SSL connections to gears, configure the appropriate SSL routing. This is the default option.

deny

If the cartridge being added to a new application specifies direct SSL connections to gears, do not allow the application to be created.

force

If the cartridge being added to a new application specifies direct SSL connections to gears, set up the appropriate SSL routing. If the cartridge being added to a new application does not specify direct SSL connections to gears, do not allow the application to be created.

```
# Whether cartridges that specify direct SSL connection to the gear
# are allowed, denied or forced.
SSL_ENDPOINT="allow"
# SSL_ENDPOINT="deny"
# SSL_ENDPOINT="force"
```

2.7. Setting Gear Supplementary Groups

When the broker creates a gear, the gear is assigned a UNIX user UID and a matching group UID. Additional groups to the gears on a node can be assigned so that you can make group-owned files available to all the gears on the node.

Use the **GEAR_SUPL_GRP**s setting in `/etc/openshift/node.conf` file to designate additional groups for the gears on that node. Note that you must create a group using standard system commands before you can add it to **GEAR_SUPL_GRP**s. Separate multiple groups with commas.

```
GEAR_SUPL_GRP="my_group,another_group"
```


**Note**

As a security measure, **root** and **wheel** groups cannot be used as values for **GEAR_SUPL_GRP**s.

2.8. Banning IP Addresses That Overload Applications

If an application user accesses an application with excessive frequency, you can block that user by banning their IP address.

**Note**

The ban instituted by the following procedure applies to all gears on the node host, including the over-accessed gear.

Procedure 2.5. To Ban an IP Address:

1. Run the following command to view a CNAME to the node host where the application's gear is located:

```
# dig appname-domain.example.com
```

2. On the node host identified in the previous step, check the application's apache logs for unusual activity. For example, a high frequency of accesses (3 to 5 per second) from the same IP address in the **access_log** file may indicate abuse:

```
# tail -f /var/lib/openshift/appUUID/appname/logs/*
```

3. Ban the offending IP addresses by placing them in iptables, running the following command for each IP address:

```
# iptables -A INPUT -s IP_address -j DROP
```

4. If you are using a configuration management system, configure it appropriately to ban the offending IP addresses. For non-managed configurations, save your new iptables rules:

```
# service iptables save
```

2.9. Enabling Maintenance Mode

The broker can be put into *maintenance mode*, in which it is running and responding, but developer requests are refused with a predefined notification message. This is useful for keeping developers informed while you perform maintenance on the OpenShift Enterprise deployment, rather than refusing the connections entirely.

Procedure 2.6. To Enable Maintenance Mode:

1. Enable maintenance mode using the **ENABLE_MAINTENANCE_MODE** setting in the **/etc/openshift/broker.conf** file on the broker host:

```
ENABLE_MAINTENANCE_MODE="true"
```

2. Define the location of the notification message using the **MAINTENANCE_NOTIFICATION_FILE** setting:

```
MAINTENANCE_NOTIFICATION_FILE="/etc/openshift/outage_notification.txt"
```

3. Create or edit the file defined in the **MAINTENANCE_NOTIFICATION_FILE** setting to contain the desired notification message seen by developers while the broker is in maintenance mode.
4. Restart the broker service:

```
# service openshift-broker restart
```

2.10. Backup and Recovery

Red Hat recommends backing up important broker and node host files to prevent data loss. This includes platform configuration files and developer application data. The following sections detail which files to back up, and how you can recover them in the event of a failure.

2.10.1. Backing Up Broker Host Files

The authentication service, DNS service, and the MongoDB datastore components of the broker host contain persistent state. Consult your system administrator for advice on how to implement fault tolerance for the authentication and DNS services you have selected for your OpenShift Enterprise installation.

See the [OpenShift Enterprise Deployment Guide](#) for instructions on how to configure redundancy with MongoDB. See the following MongoDB documentation for more information on how to implement fault tolerance with data storage and take regular backups:

- ✦ *Backup Strategies for MongoDB Systems* - <http://docs.mongodb.org/manual/administration/backups/>

In the [OpenShift Enterprise Deployment Guide](#) example installation, the MongoDB data is stored in the `/var/lib/mongodb` directory, which can be used as a potential mount point for fault tolerance or as backup storage.

2.10.2. Backing Up Node Host Files

Backing up certain node host files can help prevent data loss. You can use standard Red Hat Enterprise Linux software, such as **tar** or **cpio**, to perform this backup. Red Hat recommends backing up the following node host files and directories:

- ✦ `/opt/rh/ruby193/root/etc/mcollective`
- ✦ `/etc/passwd`
- ✦ `/var/lib/openshift`
- ✦ `/etc/openshift`



Important

Backing up the `/var/lib/openshift` directory is paramount to recovering a node host, including head gears of scaled applications, which contain data that cannot be recreated. If the file is recoverable, then it is possible to recreate a node from the existing data. Red Hat recommends this directory be backed up on a separate volume from the root file system, preferably on a Storage Area Network.

If the data from these files is lost, see [Section 2.10.3, “Recovering Failed Node Hosts”](#) for instructions on how to recover a failed node host.

Stateless and Stateful Applications

Even though applications on OpenShift Enterprise are stateless by default, developers can also use persistent storage for stateful applications by placing files in their `$OPENSIFT_DATA_DIR` directory. See the [OpenShift Enterprise User Guide](#) for more information.

Stateless applications are more easily recovered; if an application is treated as stateless, then node hosts can easily be added to and destroyed in your deployment and you can create `cron` scripts to clean up these hosts. For stateful applications, Red Hat recommends keeping the state on a separate shared storage volume. This ensures the quick recovery of a node host in the event of a failure.



Note

Developers can also take snapshots of their applications as another way to back up and restore their application data. See the [OpenShift Enterprise User Guide](#) for more information.

See Also:

- ✦ [Section 2.10.3, “Recovering Failed Node Hosts”](#)

2.10.3. Recovering Failed Node Hosts



Important

This section presumes you have backed up the `/var/lib/openshift` directory. See [Section 2.10.2, “Backing Up Node Host Files”](#) for more information.

A failed node host can be recovered if the `/var/lib/openshift` gear directory had fault tolerance and can be restored. SELinux contexts must be preserved with the gear directory in order for recovery to succeed. Note this scenario rarely occurs, especially when node hosts are virtual machines in a fault-tolerant infrastructure rather than physical machines. Note that scaled applications cannot be recovered onto a node host with a different IP address than the original node host.

Procedure 2.7. To Recover a Failed Node Host:

1. Create a node host with the same host name and IP address as the one that failed.

- a. The host name DNS A record can be adjusted if the IP address must be different. However, note that the application CNAME and database records all point to the host name and cannot be easily changed.
- b. Ensure the **ruby193-mcollective** service is not running on the new node host:

```
# service ruby193-mcollective stop
```

- c. Copy all the configuration files in the **/etc/openshift** directory from the failed node host to the new node host and ensure that the gear profile is the same.
2. Attach and mount the backup to **/var/lib/openshift**, ensuring the **usrquota** mount option is used:

```
# echo "/dev/path/to/backup/partition /var/lib/openshift/ ext4
defaults,usrquota 0 0" >> /etc/fstab
# mount -a
```

3. Reinstate quotas on the **/var/lib/openshift** directory:

```
# quotacheck -cmug /var/lib/openshift
# restorecon /var/lib/openshift/aquota.user
# quotaon /var/lib/openshift
```

4. Run the **oo-admin-regenerate-gear-metadata** tool, available starting in OpenShift Enterprise 2.1.6, on the new node host to replace and recover the failed gear data. This browses each existing gear on the gear data volume and ensures it has the correct entries in certain files, and if necessary, performs any fixes:

```
# oo-admin-regenerate-gear-metadata

This script attempts to regenerate gear entries for:
* /etc/passwd
* /etc/shadow
* /etc/group
* /etc/cgrules.conf
* /etc/cgconfig.conf
* /etc/security/limits.d

Proceed? [yes/NO]: yes
```

The **oo-admin-regenerate-gear-metadata** tool will not make any changes unless it notices any missing entries. Note that this tool can be added to a node host deployment script.

Alternatively, if you are using OpenShift Enterprise 2.1.5 or earlier, replace the **/etc/passwd** file on the new node host with the content from the original, failed node host. If this backup file was lost, see [Section 2.10.4, "Recreating /etc/passwd Entries"](#) for instructions on recreating the **/etc/passwd** file.

5. When the **oo-admin-regenerate-gear-metadata** tool completes, it runs the **oo-accept-node** command and reports the output:

```
Running oo-accept-node to check node consistency...
...
FAIL: user 54fe156faf1c09b9a900006f does not have quotas imposed. This
can be addressed by running: oo-devel-node set-quota --with-container-
```

```
uuid 54fe156faf1c09b9a900006f --blocks 2097152 --inodes 80000
```

If there are any quota errors, run the suggested quota command, then run the **oo-accept-node** command again to ensure the problem has been resolved:

```
# oo-devel-node set-quota --with-container-uuid
54fe156faf1c09b9a900006f --blocks 2097152 --inodes 80000
# oo-accept-node
```

6. Reboot the new node host to activate all changes, start the gears, and allow MCollective and other services to run.

2.10.4. Recreating /etc/passwd Entries

For node host recovery, you can recreate the **/etc/passwd** entries for all gears if this backup file was lost.



Note

If you are using OpenShift Enterprise 2.1.6 or later, you can instead run the **oo-admin-regenerate-gear-metadata** tool on a node host to replace and recover the failed gear data, including **/etc/passwd** entries.

Procedure 2.8. To Recreate /etc/passwd Entries:

1. Get a list of UUIDs from the directories in **/var/lib/openshift**.
2. For each UUID, ensure the UNIX UID and GID values correspond to the group ID of the **/var/lib/openshift/UUID** directory. See the fourth value in the output from the following command:

```
# ls -ld -n /var/lib/openshift/UUID
```

3. Create the corresponding entries in **/etc/passwd**, using another node's **/etc/passwd** file for reference.

2.11. Component Timeout Value Locations

Timeouts are useful for testing the interoperability of OpenShift components. A timeout occurs when a component sends a signal to another component, but does not receive a response. The value assigned to the timeout represents how long the component will wait for the returned signal before the process stops. All timeout values are configurable.

The following are scenarios for increasing the default timeout values:

- ✦ When a custom cartridge is taking a long time to be added to a gear.
- ✦ When network latency is forcing requests to take longer than usual.
- ✦ When a high load on the system is causing actions to take longer than usual.

Note that such cases can be resolved by other methods. For example, a high load on the system can be solved by adding extra resources to the environment.

The following table outlines the locations of various component's timeout values, the configurable parameter, and the default values expressed in seconds:

Table 2.2. Timeout Information for Various Components

Type	Location	File	Directive
MCollective	Broker	<code>/etc/openshift/plugins.d/openshift-origin-msg-broker-mcollective.conf</code>	<code>MCOLLECTIVE_TIMEOUT=240</code>
MCollective	Node	<code>/opt/rh/ruby193/root/usr/libexec/mcollective/mcollective/agent/openshift.ddl</code>	<code>:timeout => 360</code>
MCollective Client	Broker	<code>/opt/rh/ruby193/root/etc/mcollective/client.cfg</code>	<code>plugin.activemq.heartbeat_interval = 30</code>
Node Discovery	Broker	<code>/etc/openshift/plugins.d/openshift-origin-msg-broker-mcollective.conf</code>	<code>MCOLLECTIVE_DISCTIMEOUT=5</code>
Facts	Broker	<code>/etc/openshift/plugins.d/openshift-origin-msg-broker-mcollective.conf</code>	<code>MCOLLECTIVE_FACT_TIMEOUT=10</code>
Facts	Node	<code>/opt/rh/ruby193/root/usr/libexec/mcollective/mcollective/agent/rpcutil.rb</code>	<code>:timeout => 10</code>
Apache	Broker	<code>/etc/httpd/conf.d/000002_openshift_origin_broker_proxy.conf</code>	<code>ProxyTimeout 300</code>
Apache	Node	<code>/etc/httpd/conf.d/000001_openshift_origin_node.conf</code>	<code>ProxyTimeout 300</code>
RHC	Client	<code>~/.openshift/express.conf</code>	<code>timeout=300</code>
Background Thread	Broker	<code>/etc/openshift/console.conf</code>	<code>BACKGROUND_REQUEST_TIMEOUT=30</code>



Warning

Any modifications to the `/opt/rh/ruby193/root/usr/libexec/mcollective/mcollective/agent/openshift.ddl` and `/opt/rh/ruby193/root/usr/libexec/mcollective/mcollective/agent/rpcutil.rb` files are unsupported and may be erased by a `yum update`.

MCollective

The MCollective timeout is configured on the broker, and is used for MCollective messages being sent from the broker to the node. If the message is lost after it is sent, or the node takes longer than expected to complete a request, this timeout will be hit.

MCollective Client

The MCollective client timeout is used to ensure that you have a valid and active connection to your messaging broker. Lowering the defined amount causes a quicker switch to a redundant system in the event of a failure.

Node Discovery

The node discovery timeout represents the allowed amount of time a node takes to acknowledge itself in the environment, instead of broadcasting to all nodes. This method of discovery is

generally used in non-direct calls to the nodes. For example, when an application is created, when some administration commands are used, and some ssh key operations are performed.

Facts

The Facts timeout is configured on both the broker and node, and is for determining the allowed amount of time for a fact to be gathered from a node through MCollective. An example of a fact is when an application is created, and in doing so, the node's profile determines which node will perform the action. Facts are gathered often, so this timeout is short.

Apache

The Apache timeout is configured on the broker and node, and represents the timeout of proxy requests. This affects most requests, as they go through a proxy on both the broker and on the node. The ProxyTimeout on the broker affects requests to the broker API and rhc. If the timeout is exceeded due to lengthy requests, the client will receive an uninformative HTTP 502 error, even though the request may have succeeded. The ProxyTimeout on a node affects requests to hosted applications.

RHC

The rhc timeout represents the allowed amount of time that the client tools will wait for a request to be completed before ceasing the attempt. This only has to be configured on the client where rhc is run. If an action is taking longer to complete than expected, this timeout will be hit.

Background Thread

The background thread timeout is found on the broker, and determines how long requests from the console to the broker will take to be completed before ceasing the attempt. This communication is impacted by the amount of applications, domains, and gears an application developer has access to, as well as the locations of the datacenters that make up the OpenShift Enterprise deployment.

2.12. Enabling Network Isolation for Gears

Prior to OpenShift Enterprise 2.2, network isolation for gears was not applied by default. Without isolation, gears could bind and connect to **localhost** as well as IP addresses belonging to other gears on the node, allowing users access to unprotected network resources running in another user's gear. To prevent this, starting with OpenShift Enterprise 2.2 the **oo-gear-firewall** command is invoked by default at installation when using the **oo-install** installation utility or the installation scripts. It must be invoked explicitly on each node host during manual installations.



Note

The **oo-gear-firewall** command is available in OpenShift Enterprise 2.1 starting with release 2.1.9.

The **oo-gear-firewall** command configures nodes with firewall rules using the **iptables** command and SELinux policies using the **semanage** command to prevent gears from binding or connecting on IP addresses that belong to other gears.

Gears are identified as a range of user IDs on the node host. The **oo-gear-firewall** command creates static sets of rules and policies to isolate all possible gears in the range. The UID range must be the same across all hosts in a gear profile. By default, the range used by the **oo-gear-firewall** command is taken from existing district settings if known, or 1000 through 6999 if unknown. The tool can be re-run to apply rules and policies for an updated UID range if the range is changed later.

To enable network isolation for gears using the default range, run the following command on each node host:

```
# oo-gear-firewall -i enable -s enable
```

To specify the UID range:

```
# oo-gear-firewall -i enable -s enable -b District_Beginning_UID -e  
District_Ending_UID
```


Chapter 3. User Administration

OpenShift Enterprise users are typically developers working on and hosting their applications in an OpenShift Enterprise deployment. This chapter covers tasks related to the administration of those user accounts.

3.1. Creating a User

On broker hosts, the `oo-admin-ctl-user` command can be used with the `-c` or `--create` option to create new user accounts for the OpenShift Enterprise environment. The command creates a user record in MongoDB and when used with different options, allows different capabilities to be set for specific users overriding the default settings in the `/etc/openshift/broker.conf` file.

Creating user accounts using the `oo-admin-ctl-user` command does not set up authentication credentials. OpenShift Enterprise allows you to choose from a variety of authentication mechanisms and separates the concept of the user record that it stores in MongoDB from the user credentials that are stored by your chosen authentication mechanism. See the [OpenShift Enterprise Deployment Guide](#) ^[1] for more information on configuring user authentication on the broker.

To create one user at a time, use the following:

```
# oo-admin-ctl-user -c -l Username
```

To create multiple users at once, first create a file containing one login per line, then use the following:

```
# oo-admin-ctl-user -c -f File_Name
```

3.2. Removing User Applications

Use the `oo-admin-ctl-app` command to remove a user's application.



Warning

This procedure deletes all the data for the selected application and cannot be reversed.

Procedure 3.1. To Remove a User Application:

1. Stop the application by running the following command on the broker host:

```
# oo-admin-ctl-app -l username -a appname -c stop
```

2. Delete the application:

```
# oo-admin-ctl-app -l username -a appname -c destroy
```

3. If the standard `stop` and `destroy` commands fail, you can `force-stop` and `force-remove` the application. The `force-` commands do not wait for the proper shutdown sequence, so should only be used if the standard commands fail:

```
# oo-admin-ctl-app -l username -a appname -c force-stop
# oo-admin-ctl-app -l username -a appname -c force-destroy
```

3.3. Removing User Data

Remove a former user's application and domain data if they are no longer required.



Warning

The following procedure removes all of a user's application data from the system and cannot be reversed.

Procedure 3.2. To Remove User Data:

1. Prevent the user from creating more gears by running the following command on the broker host:

```
# oo-admin-ctl-user -l username --setmaxgears 0
```

2. Retrieve the user's domain and application names:

```
# oo-admin-ctl-domain -l username | egrep -i '^name:|^Namespace:'
```

3. Remove the user's applications by running the following commands for each application found in the previous step:

```
# oo-admin-ctl-app -l username -a app1 -c stop
# oo-admin-ctl-app -l username -a app1 -c destroy
```

Use the *force-destroy* parameter to remove particularly troublesome applications:

```
# oo-admin-ctl-app -l username -a app1 -c force-destroy
```

4. Delete the user's domain:

```
# oo-admin-ctl-domain -l username -c delete -n testdomain
```

The user's application data is now removed and the user cannot create any new applications; the account is effectively deactivated.

To reactivate a user's account, set the maximum amount of gears to a desired amount. Note that the *--setmaxgears* option may be restricted based on the user's configuration settings:

```
# oo-admin-ctl-user -l username --setmaxgears 5
```

3.4. Removing a User

Use the *oo-admin-ctl-domain* command to remove a user from an OpenShift Enterprise environment:

```
# oo-admin-ctl-domain -l username -c delete
```



Note

The **oo-admin-ctl-domain** command deletes the user from the OpenShift Enterprise datastore, but does not delete user credentials stored on external databases such as LDAP or Kerberos.

3.5. Enabling Users to Add a Kerberos Principal SSH Key

You can enable developers to be able to add a Kerberos principal SSH key.

The **VALID_SSH_KEY_TYPES** option, in the `/etc/openshift/broker.conf` file, contains a list of supported SSH key types. If **VALID_SSH_KEY_TYPES** is unspecified, all supported types are allowed.

If the **k5login_directory** option is used in the `/etc/krb5.conf` file, ensure SSHD can read the specified directory. For SELinux, the default context might need to be modified, as in the following example:

```
$ semanage fcontext -a -t krb5_home_t "/Path/To/File(/.*)?"
$ restorecon -R -v /Path/To/File
```

3.6. Setting Default Maximum Number of Domains per User

Edit the **DEFAULT_MAX_DOMAINS** setting in the `/etc/openshift/broker.conf` file on the broker host to configure the default maximum number of domains that can be created per user.

```
DEFAULT_MAX_DOMAINS="5"
```

The maximum number of domains a specific user can create is further restricted by the maximum number of gears that user can create. For example, if a user can create three gears, then that user can create three domains, even if the default maximum number of domains is higher. If a user attempts to create more domains than their allowed limit, the attempt fails and an error message is displayed.

3.7. Managing Custom Domain Aliases

Developers can designate custom domain aliases for applications to use DNS entries other than the domains generated by OpenShift Enterprise. By default, developers cannot create aliases that are in the cloud domain where the applications are created. For example, a developer could not create the alias **app.example.com** or **my-app.example.com** for an application that was created in the cloud domain **example.com**. This restriction prevents confusion or possible name collisions.

Enabling the **ALLOW_ALIAS_IN_DOMAIN** setting in the `/etc/openshift/broker.conf` file on the broker host allows developers to create aliases within the cloud domain, provided the alias does not take the form `<name>-<name>.<cloud-domain>`. Aliases taking this standard form of application names are rejected to prevent conflicts. For example, while a developer could now create the alias **app.example.com** for an application that was created in the cloud domain **example.com**, they still could not create the alias **my-app.example.com** because it takes the standard form.



Important

While the **ALLOW_ALIAS_IN_DOMAIN** setting is enabled, only standard name collisions are prevented. Collisions with high-availability application names are not prevented, which, should they occur on the same node host, could result in traffic being routed to the wrong gear on the node host. OpenShift Enterprise still does not create a DNS entry for the alias; that is an external step.

Procedure 3.3. To Allow Custom Domain Aliases in the Cloud Domain:

1. Edit the `/etc/openshift/broker.conf` file on the broker host and set the **ALLOW_ALIAS_IN_DOMAIN** setting to `"true"`:

```
ALLOW_ALIAS_IN_DOMAIN="true"
```

2. Restart the broker service:

```
# service openshift-broker restart
```

3.8. Determining Gear Ownership

On a node host, list the contents of the `/var/lib/openshift/.httpd.d/` directory to view the operational directories for gears. These directories have the format `UUID_domain_appname`. For example, the following command shows a gear with an application named **chess** in the domain **games**:

Example 3.1. Listing the Contents of the `/var/lib/openshift/.httpd.d/` Directory

```
# ls /var/lib/openshift/.httpd.d/  
  
c13aca229215491693202f6ffca1f84a_games_chess
```

[1] https://access.redhat.com/documentation/en-US/OpenShift_Enterprise/2/html-single/Deployment_Guide/index.html#sect-Configuring_User_Authentication

Chapter 4. Team and Global Team Management

Teams contain a group of developers that are part of a conjoined role within a domain, and are created and owned by developers.

As an OpenShift Enterprise administrator, you can create global teams from a preexisting source, such as an LDAP database, and synchronize team membership. Note that each global team must have a unique name.

Table 4.1. Teams and Global Teams

Team Types	Owner	Use	Conditions
Team	Developer	To collaborate on an application	Each team name must be unique name within a domain.
Global team	Administrator	To reuse existing group definitions for user management, such as LDAP groups.	Each global team name must be unique.



Note

By default, developers cannot view and search global teams. As an OpenShift Enterprise administrator, you must enable this capability so that global teams can be viewed and searched by developers.

4.1. Setting the Maximum Number of Teams for Specific Users

On the broker host, you can set a limit to the number of teams a developer can create with the following command:

```
# oo-admin-ctl-user -l username --setmaxteams No_of_Teams
```

The default number is set to **0**. Edit the **DEFAULT_MAX_TEAMS** setting located in the **/etc/openshift/broker.conf** file to change the default setting for any new users created after the setting has been modified. Restart the broker service for the changes to take effect.

For more information on teams, see the *OpenShift Enterprise User Guide* at <https://access.redhat.com/site/documentation>.

4.2. Creating Global Teams and Synchronizing with LDAP Groups

With the release of OpenShift Enterprise 2.1, you can create global teams and synchronize membership from an already existing source, such as an LDAP database. This enables you to have full control over global team membership. For example, if a global team is synchronized to an LDAP database, and a developer leaves your company, the privileges granted through the global team membership will be removed and you will be able to reassign or remove any of the individual's work across the platform.

Create global teams and synchronize membership with LDAP with the following procedure. However, a plain sync file can be created from any source to perform the same process if LDAP is not in use.

**Note**

This is a basic workflow. For more information, consult the **oo-admin-ctl-team** command man pages for detailed descriptions of each command shown in the following instructions.

Procedure 4.1. To Synchronize a Global Team with LDAP Groups:

1. Create an LDAP configuration file in the `/etc/openshift/` directory. This file specifies how your instance will connect to the LDAP server and query for LDAP groups and group membership.
2. Create one or more global teams. If you are not using LDAP groups, then the `--maps-to` option can be specified as anything:

```
# oo-admin-ctl-team -c create --name Team_Name --maps-to
cn=all,ou=Groups,dc=example,dc=com
```

Alternatively, you can create a global team straight from LDAP groups using the `--groups` option. In this case, you must indicate your LDAP config file and the LDAP groups to create the global team from:

```
# oo-admin-ctl-team --config-file /etc/openshift/File_Name.yml -c
create --groups Group_Name1,Group_Name2
```

Example 4.1. Sample LDAP configuration File

```
Host: server.example.com
Port: 389
Get-Group:
  Base: dc=example,dc=com
  Filter: (cn=<group_cn>)
Get-Group-Users:
  Base: <group_dn>
  Attributes: [member]
Get-User:
  Base: dc=example,dc=com
  Filter: (uid=<user_id>)
  Attributes: [emailAddress]
Openshift-Username: emailAddress
```

Example 4.2. Sample Active Directory based LDAP configuration File

```
Host: server.example.com
Port: 389
Username: CN=username.gen,OU=Generics,OU=Company
Users,DC=company,DC=com
Password: xxxxxxxxxxxxxxxx

#get group entry so we can map team to the group distinguished name
Get-Group:
  Base: dc=example,dc=com
```

```

Filter: (cn=<group_cn>)

#get all the users in the group
Get-Group-Users:
  Base: <group_dn>
  Filter: (memberOf=<group_dn>)
  Attributes: [emailaddress]

Openshift-Username: emailaddress

```

- Next, synchronize global team membership with LDAP:

```
# oo-admin-ctl-team --config-file /etc/openshift/File_Name.yml -c sync
--create-new-users --remove-old-users
```

This step can be performed in a cron job in order to regularly synchronize OpenShift Enterprise with LDAP.

Alternatively, use a sync file to synchronize global team membership with LDAP with the following command:

```
# oo-admin-ctl-team --config-file /etc/openshift/File_Name.yml -c
sync-to-file --out-file teams.sync --create-new-users --remove-old-
users
```

This command creates a file you can modify to suit your requirements. The format is the entity to act upon, an action, then the user names.

The following example sync file adds users to an OpenShift Enterprise instance, then adds them as members to the team named "myteam".

Example 4.3. Synchronizing Global Team Membership with a Sync File

```

USER|ADD|user1
...
USER|ADD|user100
MEMBER|ADD|myteam|user1, ..., user100

```

Alternatively, create this file from any source and sync team members from the specified file with the following command:

```
# oo-admin-ctl-team -c sync-from-file --in-file teams.sync
```

4.2.1. Encrypting an LDAP Global Team Connection

When synchronizing a global team with LDAP groups, you can choose to encrypt all communication with the LDAP server by adding a parameter to the LDAP `.yml` file. This encrypts any communication between the LDAP client and server and is only intended for instances where the LDAP server is a trusted source.

`simple_tls` encryption establishes an SSL/TLS encryption with the LDAP server before any LDAP protocol

data is exchanged, meaning that no validation of the LDAP server's SSL certificate is performed. Therefore, no errors are reported if the SSL certificate of the client is not trusted. If you have communication errors, see your LDAP server administrator.

To encrypt an LDAP and global team connection edit the `/etc/openshift/File_Name.yml` file and replace it with the following:

```
Host: server.example.com
Port: 636
Encryption: simple_tls
Get-Group:
  Base: dc=example,dc=com
  Filter: (cn=<group_cn>)
Get-Group-Users:
  Base: <group_dn>
  Attributes: [member]
Get-User:
  Base: dc=example,dc=com
  Filter: (uid=<user_id>)
  Attributes: [emailAddress]
Openshift-Username: emailAddress
```

Note that the port must be changed from the initial example in [Section 4.2, "Creating Global Teams and Synchronizing with LDAP Groups"](#) to the above example for encryption to successfully occur. An LDAP server cannot support both plaintext and ***simple_tls*** connections on the same port.

4.2.2. Enabling Global Team Visibility

Developers cannot search and view global teams because this capability is disabled by default. The following instructions describe how to enable this capability for new or existing user accounts.

Enabling Global Team Visibility for New Accounts

Set the following variable in the `/etc/openshift/broker.conf` file to "true":

```
DEFAULT_VIEW_GLOBAL_TEAMS = "true"
```

Next, restart the broker service for the changes to take effect:

```
# service openshift-broker restart
```

All new developer accounts that are created in the future will have the ability to search and view global teams.

Enabling Global Team Visibility for Existing Accounts

Enable the ability to view and search global teams for existing accounts with the following command:

```
$ oo-admin-ctl-user -l username --allowviewglobalteams true
```

Disable this capability by changing the `--allowviewglobalteams` option to **false**.

Chapter 5. Cartridge Management

This chapter covers the management of cartridges provided by Red Hat, the installation and management of custom and community cartridges, and other cartridge tasks.



Note

Some sections in this chapter assume that you have installed cartridges on node hosts. See the following section of the *OpenShift Enterprise Deployment Guide* for instructions on installing cartridges from RPM packages provided by Red Hat, if required:

https://access.redhat.com/site/documentation/en-US/OpenShift_Enterprise/2/html-single/Deployment_Guide/index.html#sect-Installing_Cartridges



Important

If needed, administrators can choose to configure the JBoss EWS cartridge *tomcat7* binary provided by the EWS 3 product to work around known security issues with the JBoss EWS-2 *tomcat7* binary. See [Known Issues](#) to learn more.

5.1. Managing Cartridges on Broker Hosts



Important

Cartridge management on broker hosts, which includes any usage of the `oo-admin-ctl-cartridge` command, is only applicable to OpenShift Enterprise 2.1 and later.

With the release of OpenShift Enterprise 2.1, cartridges are managed on the broker. While cartridges are still initially installed on nodes, you must then import the cartridge manifests on the broker from the nodes, which creates records in the MongoDB datastore using metadata from the manifests. Cartridges must then be activated before they can be used by developers in new applications or as add-on cartridges to existing applications.

With this cartridge management system, the broker application is able to track which cartridges are deployed on which applications, including the corresponding capabilities for each cartridge. The broker application can then control cartridge actions such as starting, stopping, scaling, and deleting. This system also allows developers to know which cartridges you have activated and made available.

Software Versions Versus Cartridge Versions

To better understand cartridge management on broker hosts, including required tasks such as importing and activating cartridges, it is important to note the distinction between *software versions* and *cartridge versions* in cartridge manifests.

When you install cartridges on nodes, either from RPM packages or source directories, cartridge manifests are installed, which describe the features a cartridge requires and the information to provide to developers about a cartridge. A single manifest can support one or more software versions, which identify the specific version or versions of a web framework or add-on technology that the cartridge is supporting. For example,

installing the *openshift-origin-cartridge-ruby* RPM package provides support for two software versions: Ruby 1.8 and Ruby 1.9.

Each software version is then associated with a *cartridge name* and presented to developers as a distinct cartridge. For example, the Ruby 1.8 software version is presented with the cartridge name **ruby-1.8**, and Ruby 1.9 with the cartridge name **ruby-1.9**.

However, each manifest also has a cartridge version, which is separate from any software version or cartridge name. When a cartridge is updated by the cartridge author, the cartridge version is incremented to identify the new release of that particular cartridge. Therefore, a single cartridge name can be associated with multiple cartridge versions over time, based on the manifests that have been installed. For example, if errata is released that updates *openshift-origin-cartridge-ruby* to a newer package version, this can result in manifests installed on nodes for two cartridge versions: 0.0.17 and 0.0.18. The **ruby-1.8** cartridge name would therefore have two cartridge versions (0.0.17 and 0.0.18), and the **ruby-1.9** cartridge would also have two cartridge versions (0.0.17 and 0.0.18).

Active and Inactive Cartridges

After manifests have been imported on the broker, you can designate cartridges as either *active* or *inactive*. The active cartridge represents the cartridge, based on an imported manifest, that is made available to developers for creating new applications or adding to existing applications. Any inactive cartridges cannot be deployed as new cartridges by developers. Cartridges can be activated automatically when importing the latest manifests from nodes or activated and deactivated manually at any time.

5.1.1. Importing, Activating, and Deactivating Cartridges

With the release of OpenShift Enterprise 2.1, you must import cartridge manifests on the broker host and activate or deactivate cartridges using the **oo-admin-ctl-cartridge** command. Running the **oo-admin-ctl-cartridge** command with the **-c import-profile** option imports the latest manifests for all cartridges installed on a randomly selected node for each gear profile. Importing the latest manifests includes manifests for both newly installed cartridges as well as newly updated cartridges that may have older manifests that were previously imported.

Run the following command on the broker host to import the latest manifests from nodes and mark all new or updated cartridges as active:

```
# oo-admin-ctl-cartridge -c import-profile --activate
```

You can also import manifests for downloadable cartridges to make them persistently available as cartridge options for developers. The cartridge sources for downloadable cartridges remain hosted externally, and they are downloaded when a developer deploys them as a new cartridge. Run the following command to import the latest manifest for a downloadable cartridge and mark all new or updated cartridges as active:

```
# oo-admin-ctl-cartridge -c import --url URL_to_Cartridge_Manifest --activate
```

When importing and activating at the same time, any other previously imported cartridges with the same cartridge name are automatically deactivated, though applications already using previous versions are unaffected and continue to function. This only means that developers cannot deploy new cartridges using the inactive cartridges.

Activating and Deactivating Using Cartridge Names

After manifests have been imported, you can activate and deactivate cartridges manually using their cartridge name. Running the **oo-admin-ctl-cartridge** command with the **-c list** option lists all currently imported cartridges and the timestamp of each import. Active cartridges are identified with an asterisk.

Example 5.1. Listing Imported Cartridges

```
# oo-admin-ctl-cartridge -c list

* cron-1.4          plugin    Cron 1.4          2014/06/16 22:09:55
UTC
* jenkins-client-1 plugin    Jenkins Client   2014/06/16 22:09:55 UTC
  mongodb-2.4      service   MongoDB 2.4      2014/06/16 22:09:55
UTC
* mysql-5.1        service   MySQL 5.1        2014/06/16 22:09:55
UTC
* mysql-5.5        service   MySQL 5.5        2014/06/16 22:09:55
UTC
  ruby-1.8         web       Ruby 1.8          2014/06/16 22:09:55
UTC
* ruby-1.9         web       Ruby 1.9          2014/06/16 22:09:55
UTC
* haproxy-1.4     web_proxy Web Load Balancer 2014/06/16 22:09:55 UTC
```

The following command activates cartridges using one or more cartridge names:

```
# oo-admin-ctl-cartridge -c activate --name Cart_Name1, Cart_Name2, Cart_Name3
```

The following command deactivates cartridges using one or more cartridge names:

```
# oo-admin-ctl-cartridge -c deactivate --name
Cart_Name1, Cart_Name2, Cart_Name3
```

Advanced Managing Using Cartridge IDs

Whenever a new manifest is imported, a record is created in the MongoDB datastore noting the cartridge name, the timestamp of the import, and a unique *cartridge ID*. Cartridge IDs are alphanumeric strings used to identify a cartridge based on an imported manifest and timestamp of the import. Therefore, a single cartridge name can be associated with multiple cartridge IDs.

For most cases, importing and activating the latest manifests at the same time is the workflow recommended by Red Hat when updates are released for cartridges provided by Red Hat. However, if you need developers to go back to using an inactive cartridge when deploying new cartridges, you can activate and deactivate using specific cartridge IDs at any time. For more advanced cartridge management, including activating and deactivating using cartridge IDs, see the man page for **oo-admin-ctl-cartridge**.

5.1.2. Migrating and Upgrading Existing Applications to Active Cartridges

To allow existing applications that are using inactive cartridges to switch to using the currently active cartridges, the following two tasks must be performed for the switch to fully take effect for both new and existing gears.

Migrating Existing Applications to Active Cartridges for New Gears

Existing applications using inactive cartridges continue to use the inactive versions when adding new gears, for example, during scaling operations. Run the following command on the broker host to allow these applications to instead use the currently active cartridges, if active versions are available, when adding new gears:

```
# oo-admin-ctl-cartridge -c migrate
```

This command initiates a migration that updates the MongoDB datastore records all of applications that are using inactive cartridges to refer instead to the currently active cartridges. Existing application gears on nodes, however, are unaffected, and continue to use inactive cartridges.



Note

If the command returns an exit code **2**, wait a few minutes for all applications to finish using the cartridges, then run the command again until it completes successfully.

Upgrading Existing Application Gears to Active Cartridges

You can use the **oo-admin-upgrade** command on the broker host to upgrade existing application gears that are currently using inactive cartridges to instead use active cartridges. The most common scenario that requires this cartridge upgrade process is when applying certain asynchronous errata updates. See the following section of the *OpenShift Enterprise Deployment Guide* for instructions on running the **oo-admin-upgrade** command when applying these types of errata updates:

https://access.redhat.com/site/documentation/en-US/OpenShift_Enterprise/2/html-single/Deployment_Guide/index.html#chap-Asynchronous_Errata_Updates

The **oo-admin-upgrade** command can also be used to upgrade existing application gears that are using inactive versions of custom, community, and downloadable cartridges. See [Section 5.3, “Upgrading Custom and Community Cartridges”](#) for more information.

5.1.3. Removing Unused Inactive Cartridges

When inactive cartridges are no longer being used by any existing applications, you can remove these cartridges from the MongoDB datastore by running the **oo-admin-ctl-cartridge** command with the **-c clean** option on the broker. This command returns a list of the unused inactive cartridges that were removed, but also lists any inactive cartridges that were not removed because they were still in use by an application. Inactive cartridges that were not removed are shown on lines starting with a **#** symbol; the number of applications that are still using the cartridge is shown at the end of the same line.

Example 5.2. Listing Imported Cartridges And Removing Unused Inactive Cartridges

```
# oo-admin-ctl-cartridge -c list

* cron-1.4          plugin    Cron 1.4          2014/06/16 22:09:55
UTC
* jenkins-client-1 plugin    Jenkins Client    2014/06/16 22:09:55 UTC
  mongodb-2.4      service   MongoDB 2.4      2014/06/16 22:09:55
UTC
* mysql-5.1        service   MySQL 5.1        2014/06/16 22:09:55
UTC
* mysql-5.5        service   MySQL 5.5        2014/06/16 22:09:55
```

```

UTC
  ruby-1.8          web          Ruby 1.8          2014/06/16 22:09:55
UTC
* ruby-1.9         web          Ruby 1.9          2014/06/16 22:09:55
UTC
* haproxy-1.4     web_proxy  Web Load Balancer 2014/06/16 22:09:55 UTC

# oo-admin-ctl-cartridge -c clean

Deleting all unused cartridges from the broker ...
539f6b336892dff17900000f # ruby-1.8
# 539f6b336892dff179000012 mongodb-2.4          1

```

In the above example, the **mongodb-2.4** and **ruby-1.8** cartridges were both inactive cartridges. The **ruby-1.8** cartridge was successfully removed, however the **mongodb-2.4** cartridge was not because it was still in use by one application. Listing the imported cartridges again confirms the removal of only the **ruby-1.8** cartridge:

Example 5.3. Listing Imported Cartridges After Removing Unused Inactive Cartridges

```

# oo-admin-ctl-cartridge -c list

* cron-1.4         plugin    Cron 1.4          2014/06/16 22:09:55
UTC
* jenkins-client-1 plugin    Jenkins Client    2014/06/16 22:09:55 UTC
  mongodb-2.4     service   MongoDB 2.4       2014/06/16 22:09:55
UTC
* mysql-5.1       service   MySQL 5.1         2014/06/16 22:09:55
UTC
* mysql-5.5       service   MySQL 5.5         2014/06/16 22:09:55
UTC
* ruby-1.9        web       Ruby 1.9          2014/06/16 22:09:55
UTC
* haproxy-1.4     web_proxy Web Load Balancer 2014/06/16 22:09:55 UTC

```

5.2. Installing and Removing Custom and Community Cartridges

In addition to cartridges provided and supported by Red Hat, you can install *custom* and *community cartridges* for developers to use in their applications. The following table describes the cartridge types available and indicates their level of Red Hat support.

Table 5.1. Cartridge Types

Type	Description	Red Hat Supported?
Standard cartridges	These cartridges are shipped with OpenShift Enterprise.	Yes. Requires base OpenShift Enterprise entitlement.
Premium cartridges	These cartridges are shipped with OpenShift Enterprise.	Yes. Requires premium add-on OpenShift Enterprise entitlement.

Type	Description	Red Hat Supported?
Custom cartridges	These cartridges are developed by users and can be based on other cartridges. See the <i>OpenShift Enterprise Cartridge Specification Guide</i> for more information on creating custom cartridges.	No.
Community cartridges	These cartridges are contributed by the community. See the OpenShift Origin Index at http://origin.ly to browse and search for many community cartridges.	No.
Partner cartridges	These cartridges are developed by third-party partners.	No, but can possibly be directly supported by the third-party developer.



Note

Red Hat supports the base OpenShift Enterprise platform on which custom and community cartridges run, but does not support or maintain the custom and community cartridges themselves. See <https://access.redhat.com/support/policy/updates/openshift/policies.html> for more information about Red Hat's support for OpenShift Enterprise.

Custom and Community Cartridges Versus Downloadable Cartridges

Custom and community cartridges are installed locally on your OpenShift Enterprise deployment and appear as cartridge options for developers when using the Management Console or client tools. However, installing custom or community cartridges locally as an administrator is not to be confused with developers using downloadable cartridges, which are custom or community cartridges that are hosted externally. See the *OpenShift Enterprise User Guide* for more information on developers using downloadable cartridges in applications:

https://access.redhat.com/site/documentation/en-US/OpenShift_Enterprise/2/html-single/User_Guide/index.html#Downloadable_Cartridges2

If you are using OpenShift Enterprise 2.1 or later, you can also see [Section 5.1.1, “Importing, Activating, and Deactivating Cartridges”](#) for instructions on managing downloadable cartridges locally in the MongoDB database. By importing the downloadable cartridge manifests on the broker, downloadable cartridges can be made persistently available as cartridge options for all developers while the cartridge sources are still hosted externally.

Installing Custom and Community Cartridges

To use custom or community cartridges in any release of OpenShift Enterprise 2, you must install the cartridges from a source directory using the **oo-admin-cartridge** command on each node host. In OpenShift Enterprise 2.1 and later, you must then import the newly installed cartridge manifests on the broker using the **oo-admin-ctl-cartridge** command before the cartridges are usable in applications.

Procedure 5.1. To Install Custom or Community Cartridges:

1. Run the following command on each node host, specifying the source directory of the custom or community cartridge to install:

```
# oo-admin-cartridge --action install --source /path/to/cartridge/
```

2. Verify that the list of installed cartridges on each node host is updated with the newly added custom or community cartridge:

Example 5.4. Listing Installed Cartridges

```
# oo-admin-cartridge --list

(redhat, jenkins-client, 1.4, 0.0.1)
(redhat, haproxy, 1.4, 0.0.1)
(redhat, jenkins, 1.4, 0.0.1)
(redhat, mock, 0.1, 0.0.1)
(redhat, tomcat, 8.0, 0.0.1)
(redhat, cron, 1.4, 0.0.1)
(redhat, php, 5.3, 0.0.1)
(myvendor, mycart, 1.1, 0.0.1)
(redhat, ruby, 1.9, 0.0.1)
(redhat, perl, 5.10, 0.0.1)
(redhat, diy, 0.1, 0.0.1)
(redhat, mysql, 5.1, 0.2.0)
```

This command displays the vendor name, cartridge name, software version, and cartridge version of each installed cartridge.

- Restart the MCollective service on each node host:

```
# service ruby193-mcollective restart
```

- Update the cartridge lists on the broker. For releases prior to OpenShift Enterprise 2.1, run the following command on the broker host to clear the broker cache and, if installed, the Management Console cache:

```
# oo-admin-broker-cache --clear --console
```

For OpenShift Enterprise 2.1 and later, run the following commands on the broker host to import and activate the latest cartridges from the nodes and, if installed, clear the Management Console cache:

```
# oo-admin-ctl-cartridge -c import-profile --activate
# oo-admin-console-cache --clear
```

Removing Custom and Community Cartridges

You can also use the **oo-admin-cartridge** command to remove cartridges from the cartridge repositories on a node host. Cartridges should only be removed from cartridge repositories after they are no longer in use by any existing applications. When removing a cartridge, ensure the same cartridge is removed from each node host.

Procedure 5.2. To Remove Custom and Community Cartridges:

- For OpenShift Enterprise 2.1 and later, deactivate the cartridge to be removed by running the following command on the broker host:

```
# oo-admin-ctl-cartridge -c deactivate --name Cart_Name
```

Deactivating the cartridge ensures it can no longer be used by developers in new applications or as add-on cartridges to existing applications. This step is not applicable for releases prior to OpenShift Enterprise 2.1.

2. List the installed cartridges by running the following command on each node host:

```
# oo-admin-cartridge --list
```

Identify in the output the cartridge name, software version, and cartridge version of the cartridge to be removed.

3. Remove the cartridge from the cartridge repository by running the following command on each node host with the cartridge information identified in the previous step:

```
# oo-admin-cartridge --action erase --name Cart_Name --version Software_Version_Number --cartridge_version Cart_Version_Number
```

4. Update the relevant cartridge lists. For releases prior to OpenShift Enterprise 2.1, clear the cache for the broker and, if installed, the Management Console by running the following command on the broker host:

```
# oo-admin-broker-cache --clear --console
```

For OpenShift Enterprise 2.1 and later, clear the cache for only the Management Console, if installed, by running the following command on the broker host:

```
# oo-admin-console-cache --clear
```

5.3. Upgrading Custom and Community Cartridges

The OpenShift Enterprise runtime contains a system for upgrading custom cartridges on a gear to the latest available version and for applying gear-level changes that affect cartridges.

The **oo-admin-upgrade** command on the broker host provides the command line interface for the upgrade system and can upgrade all the gears in an OpenShift Enterprise environment, all the gears on a node, or a single gear. This command queries the OpenShift Enterprise broker to determine the locations of the gears to migrate and uses MCollective calls to trigger the upgrade for a gear.

Upgrade Process Overview

1. Load the gear upgrade extension, if configured.
2. Inspect the gear state.
3. Run the gear extension's **pre-upgrade** script, if it exists.
4. Compute the upgrade itinerary for the gear.
5. If the itinerary contains an incompatible upgrade, stop the gear.
6. Upgrade the cartridges in the gear according to the itinerary.
7. Run the gear extension's **post-upgrade** script, if it exists.
8. If the itinerary contains an incompatible upgrade, restart and validate the gear.

- Clean up after the upgrade by deleting pre-upgrade state and upgrade metadata.

See the *OpenShift Enterprise Cartridge Specification Guide* at <https://access.redhat.com/site/documentation> for more information on the cartridge upgrade process.

The **oo-admin-upgrade** command can perform the following tasks, as described by the **oo-admin-upgrade help** command:

oo-admin-upgrade archive

Archives existing upgrade data in order to begin a completely new upgrade attempt.

oo-admin-upgrade help <task>

List available tasks or describe the designated task and its options.

oo-admin-upgrade upgrade-gear --app-name=<app_name>

Upgrades only the specified gear.

oo-admin-upgrade upgrade-node --version=<version>

Upgrades all gears on one or all nodes.



Important

Do not use the **oo-admin-upgrade upgrade-from-file** task. The **help** output of the **oo-admin-upgrade** command does list **upgrade-from-file** as a valid task. However, it is not meant for direct use by an administrator and can invalidate an upgrade process.

5.4. Adding QuickStarts to the Management Console

Developers can create applications using *QuickStarts*, which are preconfigured applications installed from a specific source. However, QuickStarts are not available to developers by default in OpenShift Enterprise. You can browse the OpenShift Origin Index at <http://origin.ly> to search for QuickStarts created by the OpenShift community or see the *OpenShift QuickStart Developer's Guide* to learn more about creating your own:

<https://www.openshift.com/developers/get-involved/creating-quickstarts>

While applications created from web framework cartridges can be automatically updated, applications created from QuickStarts cannot. Applications created using web framework cartridges are created from a designated runtime. If the runtime is updated, the cartridge automatically updates when the cartridge is restarted. However, applications created using QuickStarts require an update using Git to update the application.

You can add QuickStarts to the Management Console so that developers using your OpenShift Enterprise instance can use them to create applications. However, you must first create or obtain a configuration for the QuickStart in JSON format. When searching the OpenShift Origin Index at <http://origin.ly> for community QuickStarts, you can click the gift icon next to any result to get the JSON relevant to that QuickStart.



Warning

While QuickStarts can be added to the Management Console, QuickStarts themselves, including any community cartridges used by a QuickStart, are not supported by Red Hat and can require more configuration to work with your version of OpenShift Enterprise.

To add QuickStarts to the Management Console, edit the `/etc/openshift/quickstarts.json` file on the broker host and add entries for one or more QuickStart configurations. The following shows the basic format of a `/etc/openshift/quickstarts.json` file with two QuickStarts using some common parameters:

```
[
  {"quickstart": {
    "id": "QuickStart1_ID",
    "name": "QuickStart1_Name",
    "website": "QuickStart1_Website",
    "initial_git_url": "QuickStart1_Location_URL",
    "cartridges": ["Cart_Name"],
    "summary": "QuickStart1_Description",
    "tags": ["Tags"],
    "admin_tags": ["Tags"]
  }},
  {"quickstart": {
    "id": "QuickStart2_ID",
    "name": "QuickStart2_Name",
    "website": "QuickStart2_Website",
    "initial_git_url": "QuickStart2_Location_URL",
    "cartridges": ["Cart_Name"],
    "summary": "QuickStart2_Description",
    "tags": ["Tags"],
    "admin_tags": ["Tags"]
  }}
]
```

You must ensure that any cartridges defined in the **"cartridges"** parameter of a QuickStart configuration are available to developers in your OpenShift Enterprise instance. These can be cartridges local to your instance or downloadable cartridges. If the web framework cartridge required by a QuickStart is unavailable, developers are unable to create applications using the QuickStart, even if the QuickStart appears as an option in the Management Console. See the *OpenShift Enterprise Deployment Guide* for information on installing cartridges:

https://access.redhat.com/documentation/en-US/OpenShift_Enterprise/2/html/Deployment_Guide/sect-Installing_Cartridges.html

For example, the following shows a Django QuickStart configuration that requires the **python-2.7** cartridge:

Example 5.5. `/etc/openshift/quickstarts.json` File with a Django QuickStart Entry

```
[
  {"quickstart": {
    "id": "2",
    "name": "Django",
    "website": "https://www.djangoproject.com/",
```

```

    "initial_git_url": "git://github.com/openshift/django-example.git",
    "cartridges": ["python-2.7"],
    "summary": "A high-level Python web framework that encourages rapid
development and clean, pragmatic design. Administrator user name and
password are written to $OPENSIFT_DATA_DIR/CREDENTIALS.",
    "tags": ["python", "django", "framework"],
    "admin_tags": []
  }}
]

```

After adding entries for QuickStart configurations to the `/etc/openshift/quickstarts.json` file, clear the Management Console cache to ensure the QuickStart appears immediately for developers. For releases prior to OpenShift Enterprise 2.1, run the following command on the broker host:

```
# oo-admin-broker-cache --clear --console
```

For OpenShift Enterprise 2.1 and later, run the following command on the broker host:

```
# oo-admin-console-cache --clear
```

5.5. Disabling Downloadable Cartridges

The `DOWNLOAD_CARTRIDGES_ENABLED` setting, located in the `/etc/openshift/broker.conf` file, is set to `true` by default. Set it to `false` to disable the ability to use downloadable cartridges.

Procedure 5.3. To Disable Downloadable Cartridges:

1. Open the `/etc/openshift/broker.conf` file on the broker host and set the `DOWNLOAD_CARTRIDGES_ENABLED` value to `false`:

```
DOWNLOAD_CARTRIDGES_ENABLED="false"
```

2. Restart the `openshift-broker` service:

```
# service openshift-broker restart
```

5.6. Disabling Obsolete Cartridges

Cartridges are updated over time, leaving older versions of a cartridge with fewer advantages. To acknowledge this, cartridges can be marked *obsolete* in their cartridge manifests. Obsolete cartridges represent technologies, or versions of technologies, for which you do not want developers to be able to deploy new applications or add-on cartridges, but that are still required by the applications already using them.

By default, obsolete cartridges are still available to developers when deploying new applications or add-on cartridges. However, you can disable the use of all obsolete cartridges, preventing developers from using them in these cases. Whether the use of obsolete cartridges is enabled or disabled, applications already using obsolete cartridges continue to function normally and can add new gears using the obsolete cartridges automatically, for example during scaling operations.

Procedure 5.4. To Disable Obsolete Cartridges:

1. Ensure the **ALLOW_OBSOLETE_CARTRIDGES** parameter in the `/etc/openshift/broker.conf` file on the broker host is set to **false**:

```
ALLOW_OBSOLETE_CARTRIDGES="false"
```

2. Add the **obsolete: true** parameter to the `/usr/libexec/openshift/cartridges/Cart_Name/metadata/manifest.yml` file on each node host for any cartridge being marked obsolete:

```
Obsolete: true
```

3. Restart the MCollective service on each node host:

```
# service ruby193-mcollective restart
```

4. Update the cartridge lists on the broker. For releases prior to OpenShift Enterprise 2.1, run the following command on the broker host to clear the broker cache and, if installed, the Management Console cache:

```
# oo-admin-broker-cache --clear --console
```

For OpenShift Enterprise 2.1 and later, run the following commands on the broker host to import the latest cartridge manifests from the nodes and, if installed, clear the Management Console cache:

```
# oo-admin-ctl-cartridge -c import-profile  
# oo-admin-console-cache --clear
```

5. Restart the broker service:

```
# service openshift-broker restart
```

Chapter 6. Resource Management

This chapter covers tasks related to the management of resources on OpenShift Enterprise hosts and capacity planning, mostly focusing on node hosts.

6.1. Adding or Modifying Gear Profiles

Adding or modifying gear profiles in your OpenShift Enterprise deployment requires three main tasks:

1. Define the new gear profile on the node host.
2. Update the list of valid gear sizes on the broker host.
3. Grant users access to the new gear size.

The following instructions detail how to perform these tasks.

Procedure 6.1. To Define a New Gear Profile:

The default node host installation configures a gear profile named **small**. Edit the `/etc/openshift/resource_limits.conf` file on the node host to define a new gear profile.



Note

Starting with OpenShift Enterprise 2.1.6, additional example `resource_limits.conf` files based on other gear profile and host type configurations are included in the `/etc/openshift/` directory on nodes. For example, files for **medium** and **large** example profiles are included, as well as an **xpaas** profile for use on nodes hosting xPaaS cartridges. These files are available as a reference or can be used to copy over the existing `/etc/openshift/resource_limits.conf` file.

1. Edit the `/etc/openshift/resource_limits.conf` file on the node host and modify its parameters to your desired specifications. See the file's commented lines for information on available parameters.
2. Modify the `node_profile` parameter to set a new name for the gear profile, if desired.
3. Restart the **ruby193-mcollective** service on the node host:

```
# service ruby193-mcollective restart
```

4. If Traffic Control is enabled in the `/etc/openshift/node.conf` file, run the following command to apply any bandwidth setting changes:

```
# oo-admin-ctl-tc restart
```

5. If gears already exist on the node host, run the following commands to ensure the resource limits for the new gear profile are applied to the existing gears:

```
# oo-cgroup-enable --with-all-containers
# oo-pam-enable --with-all-containers
```

Procedure 6.2. To Update the List of Valid Gear Sizes:

If you defined a new gear profile or modified the name of an existing gear profile, you must update the broker host configuration to enable administrators to create districts for the profile and to enable developers to create gears of that size.

1. Edit the `/etc/openshift/broker.conf` file on the broker host and modify the comma-separated list in the `VALID_GEAR_SIZES` parameter to include the new gear profile.
2. Consider adding the new gear profile to the comma-separated list in the `DEFAULT_GEAR_CAPABILITIES` parameter as well, which determines the default available gear sizes for new users.
3. Restart the broker service:

```
# service openshift-broker restart
```

4. For existing users, you must grant their accounts access to the new gear size before they can create gears of that size. Run the following command on the broker host for the relevant user name and gear size:

```
# oo-admin-ctl-user -l Username --addgearsizes Gear_Size
```

5. See [Section 6.3.2, “Creating and Populating Districts”](#) for more information on how to create and populate a district, which are required for gear deployment, using the new gear profile.

6.2. Capacity Planning and Districts

Red Hat recommends that you plan for your OpenShift Enterprise deployment's expected capacity to better ensure resource availability for gears. This is best accomplished through the use of *districts*. Districts facilitate gear movement between node hosts in order to manage resource usage. Districts also allow node deactivation to ensure a node receives no additional gears.



Note

Red Hat requires using districts to provide several administrative benefits. Districts are difficult to introduce after the initial OpenShift Enterprise deployment process, therefore it is required to create districts before creating any applications.

See Also:

- » [Section 6.3.1, “Enabling Districts”](#)

6.2.1. Hierarchy of OpenShift Enterprise Entities

To better understand the role of districts, examine their relationship with other OpenShift Enterprise entities:

Table 6.1. OpenShift Enterprise Container Hierarchy

Entity	Description
Gears	Gears are at the bottom of the hierarchy, and contain instances of one or more cartridges.

Entity	Description
Nodes	Nodes contain gears. Each gear UUID has a local UNIX user UID on the node host with storage and processes constrained by various mechanisms.
Districts	When used, districts contain a set of nodes, including the gears that reside on them.
Node profiles	Node profiles are at the top of the hierarchy, and are also referred to as gear profiles or gear sizes. They are conceptually similar to a label attached to a set of nodes. Node profiles are assigned to districts, and all nodes in a district must have that node profile. Nodes or districts can only contain gears for one node profile.
Applications	Applications contain one or more gears, which currently must all have the same node profile. Application gears can span multiple nodes in multiple districts. However, no mechanism exists for placing gears on specific nodes or districts.

6.2.2. Purpose of Districts

Districts define a set of node hosts that gears can reliably move between to manage node host resource usage. Red Hat requires using districts for production deployments, and they are enabled by default for deploying gears on new installations.

OpenShift Enterprise allocates resources to gears including an external port range and IP address range, calculated according to their numeric Linux user ID (UID) on the node. A gear can only move to a node where its UID is not already in use. Districts reserve a UID for the gear across all nodes in the district, meaning only the node hosting the gear uses its UID. This allows the gear to maintain the same UID and related resources when moved to any other node within the district.

A district's UID pool includes 6000 UIDs due to the limited range of external ports. Districts allocate these UIDs to gears randomly rather than sequentially. This random allocation method makes the availability of a gear's UID more likely, even when moving the gear to a new district. Without districts, nodes allocate gear UIDs locally and sequentially, making it extremely likely that a gear's UID will be in use on other nodes.

In previous versions of OpenShift Enterprise, it was possible to change a gear's UID on a gear move. However, this required reconfiguration of the related resources, impeded cartridge maintenance, and caused trouble for application developers with hard-coded resource settings, which could not be updated automatically. Disallowing UID changes during a gear move and using districts to reserve UIDs saves developers and administrators time and trouble.

Districts also allow you to mark a node as deactivated to ensure it receives no additional gears from the broker host. The existing gears continue to run until they are destroyed or moved to another node. This enables the decommissioning of a node with minimal disruption to its gears.

6.2.3. Gear Capacity Planning

Districts and nodes have separate capacity limits for the number of gears allowed on each. Districts allocate UIDs from a fixed pool and can only contain 6000 gears, regardless of their state. Nodes, however, only constrain the number of active gears on that host.

6.2.3.1. Gear Capacity Planning for Nodes

Use the `max_active_gears` parameter in the `/etc/openshift/resource_limits.conf` file to specify the maximum number of active gears allowed per node. By default, this value is set to **100**, but most administrators will need to modify this value over time. Stopped or idled gears do not count toward this limit; a node can have any number of inactive gears, constrained only by storage. However, starting inactive gears after the `max_active_gears` limit has been reached may exceed the limit, which cannot be prevented or corrected. Reaching the limit exempts the node from future gear placement by the broker.

The safest way to calculate the *max_active_gears* limit on nodes is to consider the resource most likely to be exhausted first (typically RAM) and divide the amount of available resource by the resource limit per gear. For example, consider a node with 7.5 GB of RAM available and gears constrained to 0.5 GB of RAM:

Example 6.1. Example *max_active_gears* Calculation

```
max_active_gears = 7.5 GB / 0.5 GB = 15 gears
```

Most gears do not consume their entire resource quota, so this conservative limit can leave some resources unused. Most administrators should overcommit at least some of their nodes by allowing more gears than would fit if all gears used all of their resources. Experimentation is recommended to discover optimal settings for your OpenShift Enterprise deployment. Based on the types of cartridges and applications expected, as well as the amount of scarce resources actually used (such as RAM, CPU, network bandwidth, processes, inodes, etc.), determine an overcommit percent by which to increase your limits.

Changing the *max_active_gears* parameter after installation is harmless. Consider beginning with conservative limits and adjust accordingly after empirical evidence of usage becomes available. It is easier to add more active gears than to move them away.

6.2.3.2. Gear Capacity Planning for Districts

Due to current constraints, each district can only contain 6000 gears. Therefore, Red Hat recommends that you avoid placing a large number of nodes in a district. When a district's UID pool is exhausted its nodes will no longer receive additional gears even if they have the capacity, thereby wasting resources. You can remove excess nodes from a district by deactivating them and moving all of their gears away, which is a process known as *compacting* a district. However, avoid this process if possible to minimize disruption to the gears, and because mass gear movement can be slow and is prone to failure.

Districts exist to facilitate gear movement; the only advantage to having more than two or three nodes in a district is that fewer districts exist requiring maintenance. It is easy to add nodes to a district, and difficult to remove them. Therefore, adding nodes to districts conservatively is wise, and it is simplest to plan for districts with two or three nodes.

With perfect knowledge, calculating how many nodes to put in each district is a function of the following values:

```
D = district capacity (6000)
G = total number of gears per node
```

However, the total number of gears per node is not limited. To project this number, one of the values to consider is the node capacity for active gears:

```
C = node capacity (max_active_gears)
```

For deployments that use the idler on inactive gears, or that stop many applications, the percentage of active gears over a long period of time may be very low. Remember that even though the broker continues to fill the nodes to the active limit when gears are stopped or idled, the district capacity must also contain all of those inactive gears.

Therefore, to roughly project how many gears a full node can ultimately contain (G), determine the following value (estimating at first, then adjusting):

```
A = percentage of gears that are active
```


Then the estimate of how many gears a full node can ultimately contain is:

$$G = C * 100 / A$$

Thus, the formula for determining the number of nodes per district is:

$$N = 6000 * A / (100 * C)$$

Using the above formula, consider the following example.

Example 6.2. Example Nodes per District Calculation

If only 10% of gears are active over time, and *max_active_gears* is 50, calculate the following:

$$6000 * 10 / (100 * 50) = 12 \text{ (round down if needed)}$$

In this example, twelve nodes should be added per district.

However, in performing this calculation with imperfect knowledge, it is best to be conservative by guessing a low value of active gears and a high value for the node capacity. Adding nodes later is much better than compacting districts.

6.3. Managing Districts

Districts facilitate gear movement between node hosts in order to manage resource usage. See [Section 6.2, “Capacity Planning and Districts”](#) for more information on the concepts behind districts.

6.3.1. Enabling Districts

MCollective is responsible for communication between the broker and node hosts. This communication can fail unless the **MCollective** plug-in on the broker host is configured to enable districts.

The following parameters in the `/etc/openshift/plugins.d/openshift-origin-msg-broker-mcollective.conf` file on the broker host enable and enforce district use, all of which are set to **true** by default:

```
DISTRICTS_ENABLED=true
NODE_PROFILE_ENABLED=true
DISTRICTS_REQUIRE_FOR_APP_CREATE=true
```



Note

Though not supported for production deployments, you can disable districts by setting the above parameters to **false** and restarting the **openshift-broker** service.

The default value of **true** for the **DISTRICTS_REQUIRE_FOR_APP_CREATE** parameter prevents gear placement if no district exists with capacity for the chosen gear profile, therefore preventing the use of node hosts that are outside of districts. Setting the value to **false** and restarting the **openshift-broker** service enables immediate use of node hosts without having to understand or implement districts. While this

immediate usage may be helpful in an evaluation setting, it is neither desirable nor recommended in a production setting where districts are used to place gears on a node host before being placed in a district. This is because nodes cannot be placed in a district after they are hosting gears.

6.3.2. Creating and Populating Districts

Use the `oo-admin-ctl-district` command on the broker host to administer districts.



Note

Districts work with gear profiles to manage nodes. A default gear profile is defined in the `/etc/openshift/broker.conf` file on the broker host, and is created in the following procedure. For information on how to change the default gear profile, see [Section 6.1, “Adding or Modifying Gear Profiles”](#).

Procedure 6.3. To Create and Populate Districts:

1. Create a district using the following command:

```
# oo-admin-ctl-district -c create -n District_Name -p Gear_Profile
```

2. Add a node to the district using the following command:

```
# oo-admin-ctl-district -c add-node -n District_Name -i Node_Hostname
```

Alternatively, create a district and add nodes to it simultaneously with the following command. Note that you can add multiple node hosts with the `-i` option and any node hostnames, or use the `--available` option to add all undistricted nodes of the specified size:

```
# oo-admin-ctl-district -c add-node -n District_Name -p Gear_Profile -i Node_Hostname1,Node_Hostname2
```

The following examples use the `small` gear profile to create a district named `small_district`, then add the node host `node1.example.com` to the new district:

Example 6.3. Creating a District Named `small_district`:

```
# oo-admin-ctl-district -c create -n small_district -p small

Successfully created district: 7521a7801686477f8409e74f67b693f4

{"_id"=>"53443b8b87704f23db000001",
 "active_servers_size"=>1,
 "available_capacity"=>6000,
 "available_uids"=>"<6000 uids hidden>",
 "created_at"=>2014-04-08 18:10:19 UTC,
 "gear_size"=>"small",
 "max_capacity"=>6000,
 "max_uid"=>6999,
```

```
"name"=>"default-small-0",
"servers"=> [],
"updated_at"=>2014-04-08 18:10:19 UTC,
"uuid"=>"53443b8b87704f23db000001"}
```

Example 6.4. Adding node1.example.com to the District:

```
# oo-admin-ctl-district -c add-node -n small_district -i node1.example.com

Success!

{"_id"=>"53443b8b87704f23db000001",
 "active_servers_size"=>1,
 "available_capacity"=>6000,
 "available_uids"=>"<6000 uids hidden>",
 "created_at"=>2014-04-08 18:10:19 UTC,
 "gear_size"=>"small",
 "max_capacity"=>6000,
 "max_uid"=>6999,
 "name"=>"default-small-0",
 "servers"=>
  [{"_id"=>"53443bbc87704f49bd000001",
   "active"=>true,
   "name"=>"node1.example.com",
   "unresponsive"=>false}],
 "updated_at"=>2014-04-08 18:10:19 UTC,
 "uuid"=>"53443b8b87704f23db000001"}
```



Important

The server identity, **node1.example.com** in the above example, is the node's host name as configured on that server, which could be different from the **PUBLIC_HOSTNAME** configured in the `/etc/openshift/node.conf` file on the node. CNAME records use the **PUBLIC_HOSTNAME** parameter, which must resolve to the host through DNS; the host name could be something completely different and may not resolve in DNS at all.

MongoDB records the host name both in the district and with any gears hosted on the node host, so changing the node's host name disrupts the broker's ability to use the node. Red Hat recommends using the host name as the DNS name and not changing either after deployment.

6.3.3. Viewing District Information

This section describes how to view information about a district on your system. Note that the resulting output is in JSON format.

View all available districts with the **oo-admin-ctl-district** command, or use the **-n** option with the district's name to view a single district.

Example 6.5. Viewing All Districts

```
# oo-admin-ctl-district

{ ...
"uuid"=>"7521a7801686477f8409e74f67b693f4",
...}
```

Example 6.6. Viewing a Single District

```
# oo-admin-ctl-district -n small_district
```

District Representation on the Broker

During district creation, the broker creates a new document in its **MongoDB** database. Run the following command to view these documents inside of the **openshift_broker** database, replacing the login credentials from the **/etc/openshift/broker.conf** file, if needed:

```
# mongo -u openshift -p password openshift_broker
```

From the **mongo** shell, you can perform commands against the broker database. Run the following command to list all of the available collections in the **openshift_broker** database:

```
> db.getCollectionNames()
```

Observe the collections returned, noting the **districts** collection:

```
[ "applications", "auth_user", "cloud_users", "districts", "domains",
"locks", "system.indexes", "system.users", "usage", "usage_records" ]
```

Query the **districts** collection to verify the creation of your districts. District information is output in JSON format:

```
> db.districts.find()
```

Exit the **mongo** shell using the **exit** command:

```
> exit
```

6.3.4. Viewing Capacity Statistics

Run the following command on the broker host to view gear usage across nodes and districts:

```
# oo-stats
```

Consult the command's man page or **--help** option for script arguments. By default, this tool summarizes district and profile gear usage in a human-readable format, and produces several computer-readable formats for use by automation or monitoring.

6.3.5. Moving Gears Between Nodes

The **oo-admin-move** command moves a gear from one node to another. Note that moving gears requires a **rsync_id_rsa** private key in the broker host's **/etc/openshift/** directory and a matching public key in each node host's **/root/.ssh/authorized_keys** file as explained in the *OpenShift Enterprise Deployment Guide* at <https://access.redhat.com/site/documentation>.

A gear retains its UID when moved, therefore cross-district moves are only allowed when the destination district has the same gear UID available.

Run the **oo-admin-move** command on the broker host to move a gear from one node to another:

Example 6.7. Moving a Gear from One Node to Another

```
# oo-admin-move --gear_uuid 3baf79139b0b449d90303464dfa8dd6f -i
node2.example.com

URL: http://app3-username.example.com
Login: username
App UUID: 3baf79139b0b449d90303464dfa8dd6f
Gear UUID: 3baf79139b0b449d90303464dfa8dd6f
DEBUG: Source district uuid: NONE
DEBUG: Destination district uuid: NONE
[...]
DEBUG: Starting cartridge 'ruby-1.8' in 'app3' after move on
node2.example.com
DEBUG: Fixing DNS and mongo for gear 'app3' after move
DEBUG: Changing server identity of 'app3' from 'node1.example.com' to
'node2.example.com'
DEBUG: The gear's node profile changed from medium to small
DEBUG: Deconfiguring old app 'app3' on node1.example.com after move
Successfully moved 'app3' with gear uuid
'3baf79139b0b449d90303464dfa8dd6f' from 'node1.example.com' to
'node2.example.com'
```

6.3.6. Removing Nodes from Districts

If many gears on a node host become idle over time, you can compact the district by decommissioning or repurposing the node host. Use the **oo-admin-ctl-district** and **oo-admin-move** commands in combination to remove the gears from the node host, and then remove the host from its district.

Procedure 6.4. To Remove Nodes from Districts:

The following steps demonstrate an example situation where district **small_district** has two node hosts, **node1.example.com** and **node2.example.com**. The second node host, **node2.example.com**, has a high number of idle gears.

1. Run the following commands and fix any problems that are found. This prevents future problems caused by moving a broken gear. On the broker host, run:

```
# oo-admin-chk
```

On the node hosts, run:

```
# oo-accept -node
```

- Deactivate the node you want to remove to prevent applications from being created on or moved to the node. Existing gears continue running. On the broker host, run:

```
# oo-admin-ctl-district -c deactivate-node -n small_district -i node2.example.com
```

- Move all the gears from `node2.example.com` to `node1.example.com` by repeating the following command on the broker host for each gear on `node2.example.com`:

```
# oo-admin-move --gear_uuid UUID -i node1.example.com
```

- Remove `node2.example.com` from the district:

```
# oo-admin-ctl-district -c remove-node -n small_district -i node2.example.com
```

6.3.7. Removing Districts

When deleting a district, first remove all the node hosts from the district, then delete the district.

Procedure 6.5. To Remove Districts:

- On the broker host, set the district's capacity to 0:

```
# oo-admin-ctl-district -c remove-capacity -n district_name -s 6000
```

- Remove all the node hosts from the district you want to delete by running the following commands for each node:

```
# oo-admin-ctl-district -c deactivate-node -i node_hostname
# oo-admin-ctl-district -c remove-node -n district_name -i node_hostname
```

- Delete the empty district:

```
# oo-admin-ctl-district -c destroy -n district_name
```

6.4. Managing Regions and Zones

Prerequisites:

- » [Section 6.3, "Managing Districts"](#)

With the release of OpenShift Enterprise 2.1, you can group nodes into *regions* and *zones*. Regions and zones provide a way for brokers to manage several distinct geographies by controlling application deployments across a selected group of nodes. You can group nodes into zones, and group zones into regions. These groups can represent physical geographies, such as different countries or data centers, or can be used to provide network level separation between node environments.

Use regions when you require all application developers to use the same OpenShift Enterprise deployment,

but they are separated by their geographical location. Developers can tag nodes into zones, then group the zones into regions, and their workload is placed inside the corresponding hardware, zone, and region. If you have deployed a high-availability deployment, you can use a new zone for each rack in a datacenter, and, due to gear anti-affinity, any new gears will be spread across multiple zones, ensuring availability. Configuring regions in your deployment can also help avoid latency issues. You can maximize your application performance with less latency by deploying applications geographically closer to your expected users. For example, if your application is hosted in the US, and European application users are experiencing latency, you can use regions to extend your application to European-hosted datacenters to ease the application end-users' experience.

The current implementation of regions requires the use of districts. Nodes in districts can be tagged with a region and zone, while districts themselves can span several regions or a single region. Any single application is restricted to one region at a time, while gears within an application gear group are distributed across available zones in the current region. The broker attempts to distribute new gears evenly across the available zones; if the default gear placement algorithm is not desired, a custom gear placement plug-in can be implemented.



Note

When regions are in use, gear moves are allowed using the **oo-admin-move** tool if the move is between districted nodes and all gears in the application remain in a single region.

See Also:

- [Section 6.5, “Gear Placement Algorithm”](#)
- [Section 6.3.5, “Moving Gears Between Nodes”](#)

6.4.1. Creating a Region with Zones

Use the **oo-admin-ctl-region** tool to create, list, or destroy regions and add or remove zones within a given region.

Procedure 6.6. To Create a Region with Zones:

1. Create a new region. Region names can include alphanumeric characters, underscores, hyphens, and dots:

```
# oo-admin-ctl-region -c create -r region_name
```

2. Add zones to the region. Zone names can include alphanumeric characters, underscores, hyphens, and dots:

```
# oo-admin-ctl-region -c add-zone -r region_name -z zone_name
```

3. Verify the new region and zones:

```
# oo-admin-ctl-region -c list -r region_name
```

6.4.2. Tagging a Node with a Region and Zone

Prerequisites:

» [Section 6.3.2, “Creating and Populating Districts”](#)

Use the `oo-admin-ctl-district` tool to tag nodes in districts with a region and zone.

Procedure 6.7. To Tag a Node with a Region and Zone:

1. Create a district if one does not already exist:

```
# oo-admin-ctl-district -c create -n district_name -p gear_profile
```

2. While adding a node to a district, tag the node with a region and zone. Note that you can add multiple nodes with the `-i` option:

```
# oo-admin-ctl-district -c add-node -n district_name -i  
Node_Hostname1,Node_Hostname2 -r region_name -z zone_name
```

Alternatively, tag a node previously added to a district with a region and zone:

```
# oo-admin-ctl-district -c set-region -n district_name -i  
Node_Hostname -r region_name -z zone_name
```

6.4.3. Setting the Default Region For New Applications

You can have multiple regions at one time. Unless specified differently with the `--region` option, new applications are created in the default region set in the `/etc/openshift/broker.conf` file.

Procedure 6.8. To Set the Default Region for New Applications:

1. Change the following parameter to the desired default region for new applications:

```
DEFAULT_REGION_NAME="Region_Name"
```

2. Restart the broker service:

```
# service openshift-broker restart
```

6.4.4. Disabling Region Selection

By default, an application developer can select the region in which to create an application by using the `--region` option. Use the following procedure to disable a developer's ability to create an application in a specific region. Paired with [Section 6.4.3, “Setting the Default Region For New Applications”](#), this gives you control over the region in which newly-created applications are located.

Procedure 6.9. To Disable Region Selection When Creating Applications:

1. Change the following setting in the `/etc/openshift/broker.conf` file to false:

```
ALLOW_REGION_SELECTION="false"
```

2. Restart the broker service for the changes to take effect:

```
# service openshift-broker restart
```


6.4.5. Additional Region and Zone Tasks

List all available regions with the following command:

```
# oo-admin-ctl-region -c list
```

Remove region and zone tags from a node with the following command:

```
# oo-admin-ctl-district -c unset-region -n district_name -i server_identity
```

Remove zones from a region with the following command:

```
# oo-admin-ctl-region -c remove-zone -r region_name -z zone_name
```

Destroy empty regions with the following command:

```
# oo-admin-ctl-region -c destroy -r region_name
```

Procedure 6.10. To Require New Applications Use Zones:

1. In the `/etc/openshift/plugins.d/openshift-origin-msg-broker-mcollective.conf` file on the broker host, set the `ZONES_REQUIRE_FOR_APP_CREATE` parameter to `true` to require that new applications only use nodes tagged with a zone. When `true`, gear placement will fail if there are no zones available with the correct gear profile:

```
ZONES_REQUIRE_FOR_APP_CREATE=true
```

2. Restart the broker service:

```
# service openshift-broker restart
```

Procedure 6.11. To Enforce the Minimum Number of Zones per Gear Group

1. In the `/etc/openshift/plugins.d/openshift-origin-msg-broker-mcollective.conf` file on the broker host, set the `ZONES_MIN_PER_GEAR_GROUP` parameter to the desired minimum number of zones between which gears in application gear groups are distributed:

```
ZONES_MIN_PER_GEAR_GROUP=number_of_zones
```

2. Restart the broker service:

```
# service openshift-broker restart
```

6.5. Gear Placement Algorithm

Prerequisites:

- » [Section 6.2, “Capacity Planning and Districts”](#)
- » [Section 6.4, “Managing Regions and Zones”](#)

When new gears are added to an application, a gear placement algorithm is used to find an available node

on which to place each gear. You can either use the default algorithm or implement the gear placement plugin to use a custom algorithm; see [OpenShift Enterprise Deployment Guide](#) for more information on using a custom algorithm.

This section details the default gear placement algorithm and assumes the use of districts, which are required for production deployments and enabled by default.

Gear Placement and Districts

MongoDB configures district capacity. Gear status does not affect district capacity, because districts reserve resources; they do not account for actual usage. In the JSON record for a district, *max_capacity* indicates the maximum number of gears that can be placed in the district, while *available_capacity* indicates the number of gears still available in that district. See [Section 6.3.3, "Viewing District Information"](#) for details on viewing the JSON record of a district.

Districts have a hard limit of 6000 gears, because each member node reserves resources for the entire district to ensure availability when a gear moves between nodes. This limit means that in a district with only one node, the district is full and cannot accept additional gears if that node reaches 6000 gears. Consider a district with more than one node full when each node has a number of gears equal to 6000 divided by the number of nodes. For example, the default gear placement algorithm keeps a district with two nodes at approximately 3000 gears on each node.

Use caution when manually migrating, as well. For example, starting with three nodes in a district then removing one manually can result in the remaining two nodes being unbalanced with 4000 and 2000 gears each.

Least Preferred and Restricted Servers

When choosing nodes for new gears, the default gear placement algorithm also considers any *least preferred servers* and *restricted servers* to help maintain high availability for applications. Least preferred servers are nodes that already have gears on them for the given application gear group; it is preferable to find other nodes instead so that high availability is ensured. Restricted servers are nodes that should not be chosen at all. For example, restricted servers would be identified for high-availability applications when two **HAProxy** gears are created to ensure they are placed on different nodes.

If no other nodes are available, a least preferred server can be chosen, however a restricted node cannot, resulting in the failure of the gear creation process and a rollback of the operation.

Default Gear Placement Algorithm

The following steps describe the default algorithm for selecting a node on which to place a new gear for an application:

1. Find all the districts.
2. Find the nodes that have the least *active_capacity*.
3. Filter nodes based on given criteria to ensure gears within scalable applications are spread across multiple nodes.
4. Filter non-districted nodes when districts are required.
5. When regions and zones are present:
 - a. Filter nodes without zones when zones are required.
 - b. If the application already has gears on a node tagged with a region, exclude nodes that do not belong to the current region.

- c. Verify whether the minimum required number of zones for application gear groups is met.
 - d. Filter zones to ensure that gears within the application gear group do not exist solely in a single zone.
 - e. Choose zones that are least consumed to evenly distribute gears among zones.
 - f. When zone nodes available, exclude nodes without zones.
6. When districted nodes are available, exclude nodes without districts.
 7. Among remaining nodes, choose the ones with plenty of available capacity that are in districts with available UIDs.
 8. Randomly choose one of the nodes with the lower levels of *active_capacity*.

6.6. Setting Default Gear Quotas and Sizes

This section describes how to set default gear quotas and sizes for all users.

In the `/etc/openshift/broker.conf` file, modify the `VALID_GEAR_SIZES` value to include a list of gear sizes available to create districts and applications. The `DEFAULT_GEAR_SIZE` value and `DEFAULT_GEAR_CAPABILITIES` value must be set to a size listed available. To modify gear quotas and sizes for specific users, see [Section 6.7, “Setting Gear Quotas and Sizes for Specific Users”](#).

Table 6.2. Default Gear Quotas and Sizes

Configuration Setting	Description
<code>VALID_GEAR_SIZES</code>	Specifies a list of gear sizes that are available in the system.
<code>DEFAULT_MAX_GEAR_S</code>	Specifies the default maximum number of a gears a new user is entitled to.
<code>DEFAULT_GEAR_SIZE</code>	Specifies the default gear size when a new gear is created.
<code>DEFAULT_GEAR_CAPABILITIES</code>	Specifies a list of gear sizes available to a new user.

Example 6.8. Setting Default Gear Quotas and Sizes

Edit the `/etc/openshift/broker.conf` file on the broker host and modify the following defaults as desired to set the default gear quotas and sizes:

```
# Comma-separated list of valid gear sizes available anywhere in the
installation
VALID_GEAR_SIZES="small,medium"

# Default number of gears to assign to a new user
DEFAULT_MAX_GEAR_S="100"

# Default gear size for a new gear if not otherwise specified
DEFAULT_GEAR_SIZE="small"

# Default gear sizes (comma-separated) allowed to a new user
DEFAULT_GEAR_CAPABILITIES="small"
```

6.7. Setting Gear Quotas and Sizes for Specific Users

Use the **oo-admin-ctl-user** command to set individual user gear parameters to limit the number of gears a user is allowed, and change user access to gear sizes.

On the broker host, display gear information for a user with the following command. Replace **username** with the relevant user name:

```
# oo-admin-ctl-user -l username
```

Example 6.9.

```
# oo-admin-ctl-user -l user  
  
User user:  
  consumed gears: 3  
  max gears: 100  
  gear sizes: small
```

On the broker host, limit the number of gears a user is allowed with the following command. Replace **username** with the relevant user name and **101** with the desired number of gears:

```
# oo-admin-ctl-user -l username --setmaxgears 101
```

Example 6.10. Limiting a User's Number of Gears

```
# oo-admin-ctl-user -l user --setmaxgears 101  
Setting max_gears to 101... Done.  
  
User user:  
  consumed gears: 3  
  max gears: 101  
  gear sizes: small
```

On the broker host, add a gear size for a user with the following command. Replace **username** with the relevant user name and **medium** with the desired gear size:

```
# oo-admin-ctl-user -l username --addgearsizes medium
```

Example 6.11. Enabling a Gear Size For a User

```
# oo-admin-ctl-user -l user --addgearsizes medium  
Adding gear size medium for user user... Done.  
  
User user:
```

```
consumed gears: 3
max gears: 101
gear sizes: small, medium
```

On the broker host, remove a gear size from a user with the following command. Replace **username** with the relevant user name and **medium** with the desired gear size:

```
# oo-admin-ctl-user -l username --removegearsize medium
```

Example 6.12. Removing a Gear Size For a User

```
# oo-admin-ctl-user -l user --removegearsize medium
Removing gear size medium for user user... Done.

User user:
  consumed gears: 3
  max gears: 101
  gear sizes: small
```

6.8. Restricting Gear Sizes for Cartridges

With the release of OpenShift Enterprise 2.2, you can associate cartridges with specific gear sizes to restrict the size of deployed applications. This allows application developers to deploy certain applications on appropriate infrastructures. For example, you can set a gear size to a corresponding cartridge for applications that requires a faster CPU or more RAM to run at a higher proficiency.

To restrict gear sizes for a cartridge, add the following to the `/etc/openshift/broker.conf` file:

```
VALID_GEAR_SIZES_FOR_CARTRIDGE = "Cart_Name|Gear_Size"
```

Ensure the appropriate cartridge version number is included.

Example 6.13. Restricting Gears Sizes for Specific Cartridges

```
VALID_GEAR_SIZES_FOR_CARTRIDGE = "php-5.3|medium,large jbossews-2.0|large"
```

Restart the broker service to load the changes:

```
# service openshift-broker restart
```

If a cartridge is configured to use specific gear sizes, then a developer must be able to create applications using the gear size directly or indirectly through domain membership. Any client tools and API commands that access the cartridge listing filter out any cartridges that cannot be used by specific developers.

If the application developer attempts to use a cartridge when they do not have the corresponding gear size capability, the operation terminates and they are notified of the error. See [Section 6.6, "Setting Default Gear Quotas and Sizes"](#) and [Section 6.7, "Setting Gear Quotas and Sizes for Specific Users"](#) for information on specifying gear sizes for a developer.

6.9. Viewing Resource Usage on a Node

On a node host, use standard tools such as **free** and **vmstat** to report memory usage. Note that tools such as **vmstat**, **top**, and **uptime** report the number of processes waiting to execute. These tools might be artificially inflated, because cgroups restrict each gear to a specific time slice rather than time-sharing between processes. This restriction enables OpenShift Enterprise to provide consistent CPU availability for each gear, but also means that active processes on gears may wait while the CPU is allocated to a less active gear. As a result, reported load average may routinely be close to the number of active gears.

The **oo-idler-stats** command returns a status summary of all the gears on a node host.

Example 6.14. Returning a Status Summary of Gears on a Node Host

```
# oo-idler-stats  
  
1 running, 1 idled, 0 half-idled for a total 2 of 3 (66.67 %)
```

This example shows three gears on the node: one is running, one is idle, and one is stopped. The **half-idled** state is deprecated and always returns 0.

6.10. Enforcing Low Tenancy on Nodes

Set the **max_active_gears** value in a node's gear profile to a low number to achieve low or single tenancy on nodes. In environments with low numbers of gears per node and large gear sizes, it is important to avoid overcommitting resources and exceeding the **max_active_gears** value.

Set the **no_overcommit_active** value in the node's gear profile to **true** to avoid overcommitting resources and to enforce the desired tenancy. This setting verifies node capacity when a gear is being created on a node. If sufficient capacity is not available on the selected node, the gear is not created and an error message is displayed.

Example 6.15. Node Capacity Exceeded Error Message

```
Gear creation failed (chosen node capacity exceeded).
```

This message is also displayed if another application is already being created on the selected node.

Procedure 6.12. To Enforce Low Tenancy on Nodes:

1. Open the **/etc/openshift/resource_limits.conf** file on the node host and set the **no_overcommit_active** value to **true**:

```
no_overcommit_active=true
```

2. Restart the **ruby193-mcollective** service:

```
# service ruby193-mcollective restart
```

6.11 Managing Capacity on Broker Hosts

6.11. Managing Capacity on Broker Hosts

It is highly unlikely that a need to add more broker hosts will arise in terms of capacity. A broker host is typically used to create and delete applications. The majority of developer and end-user traffic, including updates and actual usage of the applications, is handled by node hosts. However, because the process of creating and deleting applications is sporadic, high availability is the main reason to have additional broker hosts. For example, the service at <https://openshift.redhat.com> only uses four broker hosts in EC2, mainly for high availability.

See the *OpenShift Enterprise Deployment Guide* at <https://access.redhat.com/site/documentation> for more information on how to add a broker host.

Chapter 7. Administration Console

An optional Administration Console is available for OpenShift Enterprise that allows administrators to search and navigate entities to plan the capacity of an OpenShift Enterprise deployment. Note that the current iteration of the Administration Console is read-only, so the settings or data cannot be modified.

The Administration Console's default URI is `/admin-console`, however external access is disabled by default. See the *OpenShift Enterprise Deployment Guide* at <https://access.redhat.com/site/documentation> for information on installing and configuring the Administration Console, including options for configuring external access.

7.1. Understanding the System Overview

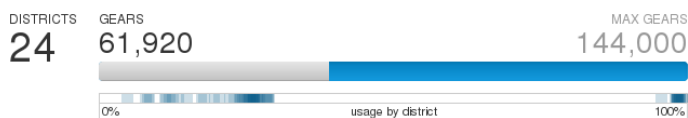
The front page of the Administration Console displays the **System Overview**. This page displays a summary of each gear profile for a user and offers suggestions based on configured targets, such as adding nodes or creating new districts. See [Section 7.6, “Configuring Suggestions”](#) for more on configuring suggestions.

The gear profile summaries provide information about total gears, active gears, the current maximums for both total and active gears, and progress bars relating these values. It also provides heat maps for district usage that show how many total gears the district is using versus its maximum, and for node usage showing how many active gears a node is using versus its maximum. More intense color in these heat maps indicates more districts or nodes at that percentage usage. For nodes, the color transitions to orange when it reaches the configured threshold, and transitions to red for nodes that exceed their active capacity. Click on the **DISTRICTS** or **NODES** section to view details for that gear profile, including its nodes and districts.

System Overview

Data collected less than a minute ago. 

Small gears



Medium gears



Suggestions







-  Add capacity for small2 gear profile
Gear profile small2 has capacity for 100 active gears, which is below the threshold of 200.
-  Add capacity for small gear profile
Gear profile small has capacity for 100 active gears, which is below the threshold of 200.
-  Add capacity for small1 gear profile
Gear profile small1 has capacity for 100 active gears, which is below the threshold of 200.
-  Gear down threshold for small gear profile needs adjustment.
The config variable GEAR_DOWN_THRESHOLD for profile small conflicts with other config values.
-  Invalid configuration value for gear profile small
The config variable "gear_up_threshold" has an unexpected value of -1.
-  Invalid configuration value

Figure 7.1. System Overview

7.2. Viewing Gear Profiles

Click on a gear profile's **Details . . .** link from the **System Overview** page to view more information about it. Each gear profile page provides the same summary for the respective gear profile as seen on the **System Overview** page and allows you to toggle between viewing the relevant districts or nodes. The **DISTRICTS** view shows all of the districts in that gear profile, and by default sorts by the fewest total gears remaining, or the most full districts. Each district displays a progress bar of the total gears and a link to view the nodes for that district.

The **DISTRICTS** view also displays a threshold indicator. The threshold is a configurable value for the target number of active gears available on a node. Each node for the district appears as either over (displayed in red) or under (displayed in green) the threshold. Each bar is slightly opaque to allow for multiple bars of the same type to show through. Therefore, if there is an intense red or green color, then several nodes are either over or under the threshold.

Small gears

Data collected 21 minutes ago.

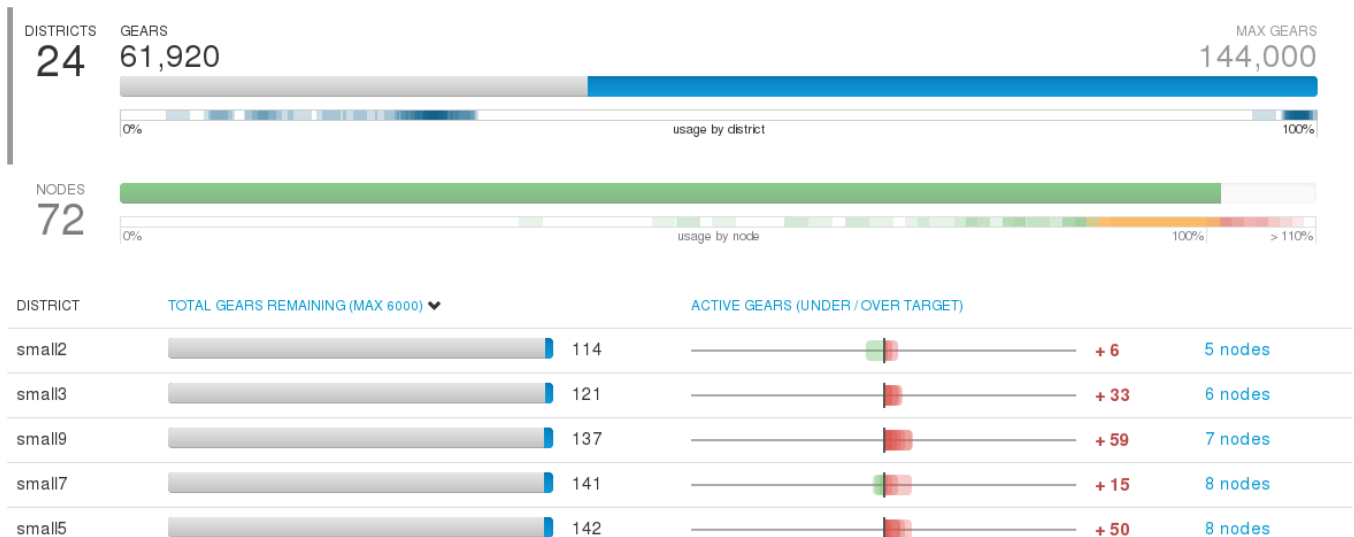


Figure 7.2. Gear Profile Districts View

Each gear profile page's **NODES** view shows all nodes in that gear profile, and by default sorts by the fewest active gears remaining. A progress bar indicates how many active gears the node is using; a full bar is 110% or more of the active capacity. The bars transition to orange when it reaches the configured threshold, and transitions to red for nodes that exceed their active capacity. The **Show** drop-down menu allows you to filter the nodes displayed, with the options for either all nodes in the gear profile, only nodes in a particular district, or nodes that are currently not in a district. Selecting a node takes you to the node's summary page.

Small gears

Data collected 36 minutes ago.

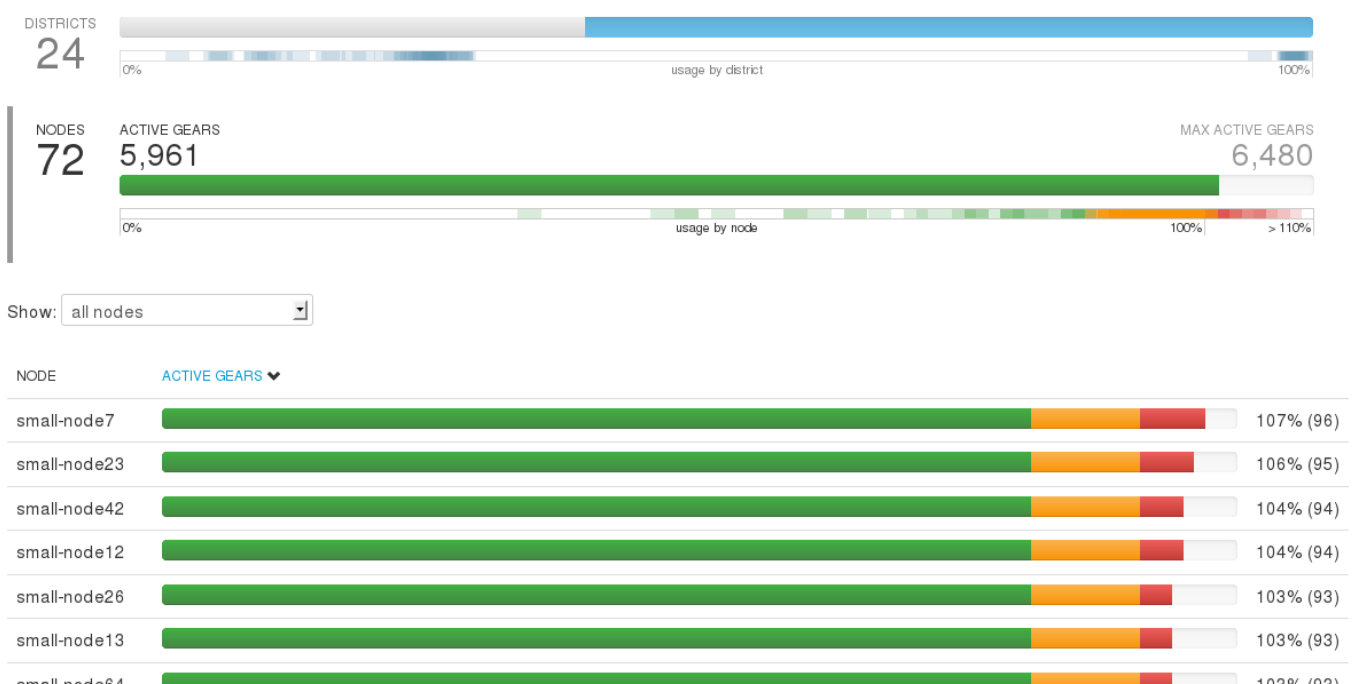


Figure 7.3. Gear Profile Nodes View

7.3. Viewing Suggestions

The suggestions on the **System Overview** page are only a short summary of the full suggestion. All of the suggestions are also available on the **Suggestions** page, and are expandable to a more detailed description with recommended commands where applicable. The triggers for many of these suggestions are configurable values. See [Section 7.6, “Configuring Suggestions”](#) for more on configuring suggestions.

! Add capacity for small2 gear profile ^
 Gear profile small2 has capacity for 100 active gears, which is below the threshold of 200.
 Create 6 nodes with profile small2 to increase the active gear capacity. This suggestion is based on:

- 200 gears needed to both reach the threshold and meet the configured GEAR_UP_SIZE of this profile.
- The estimated maximum active gears of 50 for nodes in this profile.

Add 1 node to district district_2, which according to an active gear percent of 10% has room for 1 more node.

```
oo-admin-ctl-district -c add-node -n 'district_2' -i <server_identity>
```

Add 1 node to district district_3, which according to an active gear percent of 10% has room for 1 more node.

```
oo-admin-ctl-district -c add-node -n 'district_3' -i <server_identity>
```

Create 2 new districts.

```
oo-admin-ctl-district -c create -p 'small2' -n <name>
```

Then add 2 nodes to each new district.

```
oo-admin-ctl-district -c add-node -n '<district_name>' -i <server_identity>
```

Figure 7.4. Suggestions for Adding Capacity

! 4 nodes failed to respond ^
 The nodes may be down, or may not have responded within the configured timeout.

Server Identity	District Name	Profile
small-node5.example.com	small23	small
small-node17.example.com	small3	small
medium-node6.example.com	medium22	medium
large-node51.example.com	large14	large

From the broker, check if a node is still responding with:

```
mco ping -I <server_identity>
```

If the node is still responding then the configured timeout may be too short. Compare the response time from mco ping to the STATS_GATHER_TIMEOUT in

```
/etc/openshift/plugins.d/openshift-origin-admin-console.conf
```

If the node does not respond at all, the node may actually be unresponsive, or there may be a network or service outage blocking communication.
 While a node is not responding, its district is exempt from recommendations, because accurate district statistics cannot be calculated.

Figure 7.5. Suggestions for Missing Nodes

7.4. Searching for Entities

The upper right section of every page of the Administration Console contains a search box, providing a quick way to find OpenShift Enterprise entities. Additionally, the dedicated **Search** page provides more information on the expected search queries for the different entities, such as **Applications**, **Users**, **Gears**, and **Nodes**.

The search does not intend to provide a list of possible matches; it is a quick access method that attempts to directly match the search query. **Applications**, **User**, **Gear**, and **Node** pages link to each other where possible. For example, a **User** page links to all of the user's applications, and vice versa.

7.5. Viewing Statistics

The **Stats** page provides counts of applications, users, and domains. It also provides histograms for the number of applications per domain, number of gears per user, and number of domains owned per user.

7.6. Configuring Suggestions

The **System Overview** front page of the Administration Console provides a visual and numeric summary of the capacity and usage of the entire OpenShift Enterprise deployment. You can configure it to provide suggestions for when the capacity may have to be adjusted. Because OpenShift Enterprise environments vary, thresholds are not set by default, and thus capacity suggestions are initially absent. Settings for capacity planning are configured in the Administration Console configuration file, located at `/etc/openshift/plugins.d/openshift-origin-admin-console.conf`, to enable suggestions that warn of current or impending capacity problems. For example, the Administration Console can suggest where to add nodes to ensure a particular profile can continue to create gears, or where capacity is poorly utilized.



Note

See [Section 6.2, “Capacity Planning and Districts”](#) for more information on capacity planning in OpenShift Enterprise to better understand the information that is displayed by the Administration Console and the importance of the suggestion settings.

Both the configuration file settings and the existing capacity data determine the suggestions for adding and removing capacity, which are conservative in terms of placing nodes in districts. For example, when calculating district size suggestions, the Administration Console uses the observed gear percentage if that percentage is lower than expected; if all nodes do not have the same `max_active_gears` limit, it uses the largest.

The `STATS_CACHE_TIMEOUT` parameter in the configuration file, set by default to one hour, determines how long to keep capacity and suggestions statistics cached. If you do not immediately see changes that you expect in the Administration Console, refresh the data by clicking the refresh icon near the upper right of any page.

7.7. Loading Capacity Data from a File

The Administration Console uses the same **Admin Stats** library used by the `oo-stats` command to collect capacity data. Record the YAML or JSON output from the `oo-stats` command and use the output directly instead of the actual system data using the following procedure:

Procedure 7.1. To Load Capacity Data from a File:

1. Run the following command to gather capacity data on a broker host, which then records the output to a file. Replace `yaml` with `json`, if needed:

```
# oo-stats -f yaml > /tmp/stats.yaml
```

2. Copy `/tmp/stats.yaml` to the host running the Administration Console, if needed.
3. Set `/tmp/stats.yaml` in the `STATS_FROM_FILE` parameter in the `/etc/openshift/plugins.d/openshift-origin-admin-console.conf` file.
4. SELinux limits what the broker application can read (for example, it cannot ordinarily read `/tmp` entries). To ensure that the broker can read the data file, adjust its context as follows:

```
# chcon system_u:object_r:httpd_sys_content_t:s0 /tmp/stats.yaml
```

5. Restart the broker:

```
# service openshift-broker restart
```

The Administration Console will now use the loaded file for capacity views and suggestions, although navigation still only works for entities actually present.

7.8. Exposed Data

The Administration Console exposes OpenShift Enterprise system data for use by external tools. In the current iteration of the Administration Console, you can retrieve the raw data from some of the application controllers in JSON format. This is not a long-term API however, and is likely to change in future releases. You can access the following URLs by appending them to the appropriate host name:

Exposed Data Points

- `/admin-console/capacity/profiles.json` returns all profile summaries from the **Admin Stats** library (the same library used by the `oo-stats` command). Add the `?reload=1` parameter to ensure the data is current rather than cached.
- `/admin-console/stats/gears_per_user.json` returns frequency data for gears owned by a user.
- `/admin-console/stats/apps_per_domain.json` returns frequency data for applications belonging to a domain.
- `/admin-console/stats/domains_per_user.json` returns frequency data for domains owned by a user.

The following example shows how to access `/admin-console/capacity/profiles.json` on the broker host:

```
# curl http://localhost:8080/admin-console/capacity/profiles.json
```

Chapter 8. Monitoring

This chapter covers recommended practices for evaluating the overall health and performance of an OpenShift Enterprise deployment, as well as configuration options for gathering logs and metrics.

With the release of OpenShift Enterprise 2.1, you can now choose to collocate log files to **Syslog** instead of their default locations, which are found in several locations across an OpenShift Enterprise instance. Placing them into a single location helps you to analyze broker, node, gear, and Management Console errors. See the following sections for more information on how to enable **Syslog** for OpenShift Enterprise components.

8.1. General System Checks

1. Use standard system administration checks to monitor the basic health of your system. For example:
 - ✦ ensure adequate memory
 - ✦ minimize disk swapping
 - ✦ ensure adequate disk space
 - ✦ monitor file system health
2. Monitor the services used by OpenShift Enterprise. Ensure the following are running and configured correctly:
 - ✦ MCollective
 - ✦ Mongo
 - ✦ Apache
 - ✦ ActiveMQ
 - ✦ SELinux and cgroups
3. Use custom scripts to run checks specific to your system. Confirm that the entire system is working by checking:
 - ✦ nodes and gears are valid and consistent system-wide by running **oo-admin-chk** on a broker host
 - ✦ gears are created and deleted correctly
 - ✦ available statistics and capacities
 - ✦ hosts respond to MCollective using **oo-mco ping**

8.2. Response Times for Administrative Actions

The following sections provide approximate response times for various administrative tasks.

Application Creation

The time it takes to create an application depends on the application type and how long DNS propagation takes. Apart from the time spent propagating DNS, applications are generally created in approximately 35 seconds.

Restarting a Node

The length of time required to restart a node host depends on the number of gears on the node host, and how many of those gears are active. Node host restarts can take approximately five to thirty minutes.

8.3. Testing a Path Through the Whole System

Every major component of a system can be tested by creating a test user, and then creating and deleting an application. This includes user authentication, the broker application, the MongoDB datastore, the DNS, MCollective and other messaging services, and node host and gear functionality.

8.4. Monitoring Broker Activity

Monitoring broker activity provides insight into the usage of your OpenShift Enterprise deployment and can help diagnose problems with it. Note that problems on the broker only affect actions that interact with the broker, such as creating applications. Deployed applications continue to function normally on their nodes even if the broker is unavailable, which means that developers can still access and update their applications using **SSH** and Git, and applications are still available online.

8.4.1. Default Broker Log File Locations

By default, the actions of a broker host are written locally to certain log files. The following table provides the location and a description of important broker host log files:

Table 8.1. Default Broker Log Files

File	Description
<code>/var/log/openshift/broker/production.log</code>	This file contains any log requests processed by the broker application.
<code>/var/log/openshift/broker/user_action.log</code>	This file logs any user actions, including the creation and deletion of gears. Similar to production.log , but less verbose.
<code>/var/log/openshift/broker/httpd/access_log</code>	This file logs any calls made to the REST API.
<code>/var/log/openshift/broker/httpd/error_log</code>	This file logs any Rails errors that occur on start-up.
<code>/var/log/openshift/broker/usage.log</code>	This file logs information on gear or filesystem resource usage, but only if tracking is enabled in the <code>/etc/openshift/broker.conf</code> file.

8.4.2. Verifying Functionality with Administration Commands

There are several commands that test the basic functionality of an OpenShift Enterprise instance.

Each command is outlined with examples in [Chapter 9, Command Reference](#).



Note

In order to prevent the overloading of your instance, Red Hat recommends running the following commands no less than twelve hours apart.

Verifying Broker Host Functionality

The following table outlines administration commands for testing functionality on a broker host:

Table 8.2. Verification Commands for a Broker Host

Command Name	Description
<code>oo-accept-broker</code>	Use this command to test basic functionality before performing more intensive tests.
<code>oo-accept-systems</code>	Use this command to verify that the settings on node hosts are valid and can be used by a broker host.
<code>oo-admin-chk</code>	Use this command to verify consistency throughout all node hosts and gears in an OpenShift Enterprise deployment.

Verifying Node Host Functionality

The following table contains administration commands for testing functionality on a node host:

Table 8.3. Verification Commands for a Node Host

Command Name	Description
<code>oo-accept-node</code>	Use this command to test basic functionality before performing more intensive tests.

8.5. Monitoring Node and Gear Activity

Problems with node hosts can affect a single gear, several gears, or all the gears on the host. A common indication of a node host problem is an inability to create or remove gears on that host.

8.5.1. Default Node Log File Locations

By default, node components write log messages locally to their configured log file destination. The following table provides the default locations of important log files for node components, summarizes the information they contain, and identifies the configuration setting that changes their default location:

Table 8.4. Node Log Files

File	Description	Configuration Setting
<code>/var/log/openshift/node/platform.log</code>	Primary log for node platform actions including MCollective actions performed on the node host.	<code>PLATFORM_LOG_FILE</code> setting in <code>/etc/openshift/node.conf</code> .
<code>/var/log/openshift/node/platform-trace.log</code>	Logs node platform trace actions.	<code>PLATFORM_TRACE_LOG_FILE</code> setting in <code>/etc/openshift/node.conf</code> .
<code>/var/log/openshift/node/ruby193-mcollective.log</code>	Logs MCollective messages communicated between broker and node hosts. Read to confirm proper gear creation.	<code>logfile</code> setting in <code>/opt/rh/ruby193/root/etc/mcollective/server.cfg</code> .
<code>/var/log/httpd/openshift_log</code>	Logs gear access from the front-end Apache.	<code>APACHE_ACCESS_LOG</code> setting in <code>/etc/openshift/node.conf</code> .

8.5.2. Enabling Application and Gear Context in Node Component Logs

Further context, such as application names and gear UUIDs, can be included in log messages from node components, which adds visibility by associating entries with specific applications or gears. This can also improve the ability to correlate log entries using reference IDs from the broker.

Procedure 8.1. To Enable Application and Gear Context in Apache Logs:

1. Configure **Apache** to include application names and gear UUIDs in its log messages by editing the `/etc/sysconfig/httpd` file and adding the following line:

```
OPTIONS="-DopenShiftAnnotateFrontendAccessLog"
```



Important

All options must be on the same line. For example, in [Section 8.8.2, "Enabling Syslog for Node Components"](#) another option for Apache log files is explained. If both options are desired, the line must use the following syntax:

```
OPTIONS="-Option1 -Option2"
```

2. Restart the **httpd** service for the **Apache** changes to take effect for new applications:

```
# service httpd restart
```

Procedure 8.2. To Enable Application and Gear Context in Node Platform Logs:

1. Configure the node platform to include application and gear context in its log messages by editing the `/etc/openshift/node.conf` file and adding the following line:

```
PLATFORM_LOG_CONTEXT_ENABLED=1
```

2. Add the following line to specify which attributes are included. Set any or all of the following options in a comma-delimited list:

```
PLATFORM_LOG_CONTEXT_ATTRS=request_id,container_uuid,app_uuid
```

This produces key-value pairs for the specified attributes. If no context attribute configuration is present, all context attributes are printed.

3. Restart the **ruby193-mcollective** service for the node platform changes to take effect:

```
# service ruby193-mcollective restart
```

8.5.3. Viewing Application Details

Use the `oo-app-info` command to view the information about applications and gears.

On the broker host, view application details with the following command, replacing *options* with the desired values:

oo-app-info options

Option	Description
-a, --app [NAME]	Specify a comma-delimited list of application names, without domains. Alternatively, specify a regular expression instead of application names.
-d, --domain [NAME]	Specify a comma-delimited list of domain namespaces, without application names. Alternatively, specify a regular expression instead of a domain namespace.
-f, --fqdn [NAME]	Specify a comma-delimited list of application FQDNs. Alternatively, specify a regular expression instead of an application FQDN.
-l, --login [NAME]	Specify a comma-delimited list of OpenShift user logins. Alternatively, specify a regular expression instead of a login.
-u, --gear_uuid [uuid]	Specify a comma-delimited list of application or gear UUIDs. Alternatively, specify a regular expression instead of a UUID.
--deleted	Search for deleted applications.
--raw	Display raw data structure without formatting.

Example 8.1. Viewing Application Details for a Specific Login:

```
# oo-app-info --login login --app py33s

Loading broker environment... Done.

=====
=====
Login:          demo
Plan:          ()

App Name:      py33s
App UUID:      54471801f09833e74300001e
Creation Time: 2014-10-22 02:35:45 AM
URL:           http://py33s-demo.example.com

Group Instance[0]:
  Components:
    Cartridge Name: python-3.3
    Component Name: python-3.3
  Gear[0]
    Server Identity: node.hosts.example.com
    Gear UUID:      54471801f09833e74300001e
    Gear UID:       1224

Current DNS
-----
py33s-demo.example.com is an alias for node.hosts.example.com.
node.hosts.example.com has address 172.16.4.136
```

8.5.4. The Watchman Tool

The Watchman tool is a daemon that is used to protect your OpenShift Enterprise instance against common issues found by Red Hat. The Watchman tool solves these common issues autonomously, and includes the following built-in features:

- Watchman searches cgroup event flow through syslog to determine when a gear is destroyed. If the pattern does not match a clean gear removal, the gear will be restarted.
- Watchman monitors the application server logs for messages hinting at out of memory, then restarts the gear if needed.
- Watchman compares the user-defined status of a gear, then the actual status of the gear, and fixes any dependencies.
- Watchman searches processes to ensure they belong to the correct cgroup. It kills abandoned processes associated with a stopped gear, or restarts a gear that has zero running processes.
- Watchman monitors the usage rate of CPU cycles and restricts a gear's CPU consumption if the rate of change is too aggressive.

Watchman capabilities can be expanded with plug-ins. See [Section 8.5.4.2, “Supported Watchman Plug-ins”](#) for more information.

8.5.4.1. Enabling Watchman

Watchman is an optional tool that monitors the state of gears and cartridges on a node. It is primarily used to automatically attempt to resolve problems and, if required, restore any gears that have ceased to function.

Enable the Watchman tool persistently using the following command on a node host:

```
# chkconfig openshift-watchman on
```

8.5.4.2. Supported Watchman Plug-ins

Plug-ins are used to expand the events and conditions on which the Watchman tool takes action. These plug-ins are located in the `/etc/openshift/watchman/plugins.d` directory, and are outlined in the following table.

The Watchman tool automatically loads any plug-in whose file is added to the `/etc/openshift/watchman/plugins.d` directory. To disable a plug-in, move the plug-in file from the `plugins.d` directory and place it into an unused directory for backup. Ensure to restart the Watchman tool any time a change to the `plugins.d` directory is made:

```
# service openshift-watchman restart
```

Table 8.5. Supported Watchman Plug-ins

Watchman Plug-in Name	Plug-in Filename	Function
Syslog	<code>syslog_plugin.rb</code>	This searches the <code>/var/log/messages</code> file for any messages logged by cgroups when a gear is destroyed and restarts the gear if required.

Watchman Plug-in Name	Plug-in Filename	Function
JBoss	jboss_plugin.rb	This searches JBoss cartridge server.log files for out-of-memory exceptions and restarts gears if required.
Gear State	gear_state_plugin.rb	This compares the last state change commanded by the user against the current status of the gear in order to find the best use for resources. For example, this plug-in kills any processes running on a stopped gear, and restarts a started gear if it has no running processes.
Throttler	throttler_plugin.rb	This uses cgroups to monitor CPU usage and restricts usage if required.
Metrics	metrics_plugin.rb	This gathers and publishes gear-level metrics such as cgroups data for all gears on a node host at a configurable interval.
OOM	oom_plugin.rb	Available starting with OpenShift Enterprise 2.1.4, this monitors for gears under out-of-memory (OOM) state, attempts to resolve problems, and restarts gears if required.

Enabling the Watchman Metrics Plug-in

As well as adding it to the **plugins.d** directory, as outlined above, enabling the Metrics plug-in requires an extra step. Edit the **/etc/openshift/node.conf** file and ensure the following line is uncommented to enable the Metrics plug-in:

```
WATCHMAN_METRICS_ENABLED=true
```

Restart the Watchman service for the changes to take effect

The Metrics plug-in logs its data to **/var/log/openshift/node/platform.log** by default, which is the node platform log file. However, if you have **Syslog** enabled, log data is sent to the syslog file with **type=metric** in each Metrics logline.

Example 8.2. Logged Metrics Data

```
Jun 10 16:25:39 vm openshift-platform[29398]: type=metric appName=php6
gear=53961099e659c55b08000102 app=53961099e659c55b08000102 ns=demo
quota.blocks.used=988 quota.blocks.limit=1048576 quota.files.used=229
quota.files.limit=80000
```

Configuring the Throttler Plug-in

To configure the throttler plug-in, edit the following parameters in the **/etc/openshift/resource_limits.conf** file on a node host:

Example 8.3. Throttler Plug-in Configuration Parameters

```
[cg_template_throttled]
cpu_shares=128
cpu_cfs_quota_us=100000
apply_period=120
```

```
apply_percent=30
restore_percent=70
```

The throttler plug-in works with cgroups. It watches for gears that are using an excessive amount of CPU time, and when the gear's CPU time is being used more than the amount defined by the **apply_percent** parameter the gear is placed into a 'throttled' cgroup quota. The throttler plug-in continues to watch the gear until it is using the amount of CPU time defined by the **restore_percent** parameter or less. When the amount is stabilized, the gear is placed back into the default cgroup limit.

The **cpu_shares** and **cpu_cfs_quota** parameters define the throttle cgroup templates to apply to any throttled gears. The **apply_period** parameter defines how long a gear is throttled before it is restored into the default cgroup limit. For example, using the default parameters, a gear is throttled once it uses over 30 percent of a gear's CPU usage for 120 seconds, and a throttled gear is unthrottled once it is using less than 70 percent of the throttled CPU usage for 120 seconds.

8.5.4.3. Configuring Watchman

Set any of the following parameters in the `/etc/sysconfig/watchman` file to configure Watchman. This file is available by default starting with OpenShift Enterprise 2.1.4.

Table 8.6. Watchman Configuration Parameters

Parameter	Function
GEAR_RETRIES	This sets the number of gear restarts attempted before a RETRY_PERIOD .
RETRY_DELAY	This sets the number of seconds to wait before attempting to restart the gear.
RETRY_PERIOD	This sets the number of seconds to wait before resetting the GEAR_RETRIES entry.
STATE_CHANGE_DELAY	This sets the number of seconds the gear remains broken before Watchman attempts to fix the issue.
STATE_CHECK_PERIOD	Available starting with OpenShift Enterprise 2.1.4, this sets the number of seconds to wait since the last check before checking the gear state. Increase this to reduce the impact of the Watchman Gear State plug-in.
THROTTLER_CHECK_PERIOD	Available starting with OpenShift Enterprise 2.1.4, this sets the number of seconds to wait before checking the cgroup state of the gear. Increase this to reduce the impact of the Watchman Throttler plug-in.
OOM_CHECK_PERIOD	Available starting with OpenShift Enterprise 2.1.4, this sets the number of seconds to wait since the last check before looking for gears under out-of-memory (OOM) state.

Restart the Watchman tool for any changes to take effect:

```
# service openshift-watchman restart
```

8.5.5. Testing Node Host Functionality

Use the **oo-accept-node** command to test the basic functionality of the node host before performing more intensive tests.

8.5.6. Validating Gears

Validate the gears registered in the MongoDB datastore against those on all node hosts by running **oo-admin-chk** on a broker host. This command lists gears that partially failed creation or deletion, as well as nodes with incorrect gear counts.

8.5.7. Node Capacity

When a node host reaches capacity, extend the capabilities of a node by adding storage and RAM, or by moving gears to different nodes with available resources. On a node with 100 applications, 100 active gears out of 100 capacity indicates a full node. If an application on a full node is idle, and is restored, the node will exceed capacity (101/100). The node will continue to function normally, but you can move gears to another node to reduce the active applications.

To determine the active gear capacity of a node host, view the **active_capacity** entry in **/opt/rh/ruby193/root/etc/mcollective/facts.yaml**.

To determine the maximum number of active applications on a node host, view the **max_active_gears** entry in **/etc/openshift/resource_limits.conf**.

8.6. Monitoring Management Console Activity

Monitoring console log files can provide insight into any problems with the Management Console.

8.6.1. Default Management Console Log File Locations

Management Console log files are found in the **/var/log/openshift/console** directory.

Table 8.7. Default Management Console Log Files

File	Description
/var/log/openshift/console/production.log	This file contains any log requests processed by the Management Console.
/var/log/openshift/console/httpd/access_log	This file logs any calls made to the REST API.
/var/log/openshift/console/httpd/error_log	This file logs any Rails errors that occur on start-up.

8.7. Usage Tracking

Monitoring of resource usage per user is enabled by default in OpenShift Enterprise. This includes gears that are created, additional gear storage, and gear age.

User resource tracking consumes space in the **MongoDB** datastore. Therefore, Red Hat recommends that you disable resource tracking if it is not required.

Procedure 8.3. To Disable Usage Tracking:

1. Open **/etc/openshift/broker.conf** on the broker host.
2. Set the value of **ENABLE_USAGE_TRACKING_DATASTORE** to **"false"**.

- a. Alternatively, set **ENABLE_USAGE_TRACKING_AUDIT_LOG** to **false** to disable audit logging for usage tracking.
3. Restart the broker service:

```
# service openshift-broker restart
```

8.7.1. Setting Tracked and Untracked Storage

The **oo-admin-ctl-user** command allows you to manage a user's available tracked and untracked gear storage. Both types of storage provide additional storage to a user's gears, but untracked storage is not included in usage reports. The total storage available to a user's gear is the sum of the tracked and untracked storage.

When a user adds storage to a gear, their untracked allowance is applied first. When the untracked storage is depleted, further storage is drawn from their tracked allowance.

After you set the gear storage maximums, a user can add their allotted additional storage to their applications using the Management Console or the client tools. See the *OpenShift Enterprise User Guide* at <https://access.redhat.com/site/documentation> for more information on storage management.



Note

Red Hat recommends that you only set the maximum untracked storage amount if resource usage is not being tracked for users.

On the broker host, set the maximum amount of tracked storage per gear with the following command. Replace the example user name and amount of tracked gear storage to suit your requirements:

```
# oo-admin-ctl-user -l username --setmaxtrackedstorage 10
```

Example 8.4. Setting the Maximum Amount of Tracked Storage

```
# oo-admin-ctl-user -l user --setmaxtrackedstorage 10
Setting max_tracked_addtl_storage_per_gear to 10... Done.

User user:
           consumed gears: 2
           max gears: 100
max tracked storage per gear: 10
max untracked storage per gear: 0
plan upgrade enabled:
           gear sizes: small
sub accounts allowed: false
```

On the broker host, set the maximum amount of untracked storage per gear with the following command. Replace the example user name and amount of untracked gear storage to suit your requirements:

```
# oo-admin-ctl-user -l username --setmaxuntrackedstorage 10
```

Example 8.5. Setting the Maximum Amount of Untracked Storage

```
# oo-admin-ctl-user -l user --setmaxuntrackedstorage 10
Setting max_tracked_addtl_storage_per_gear to 10... Done.

User user:
           consumed gears: 2
           max gears: 100
max tracked storage per gear: 10
max untracked storage per gear: 10
plan upgrade enabled:
           gear sizes: small
sub accounts allowed: false
```

8.7.2. Viewing Accumulated Usage Data

Use the **oo-admin-usage** or **oo-admin-ctl-usage** command to view resource usage reports per user. Usage reports include how long a user has been using a gear and any additional storage. Red Hat recommends using the **oo-admin-usage** command for listing a single user's usage data, because it contains more detailed information. Use the **oo-admin-ctl-usage** command to list all users' basic usage data at one time.

On the broker host, view resource usage per user with the following command, replacing *username* with the desired value:

```
# oo-admin-usage -l username
```

Example 8.6. Viewing a User's Resource Usage

```
# oo-admin-usage -l user
Usage for user
-----
#1
Usage Type: GEAR_USAGE (small)
  Gear ID: 519262ef6892df43f7000001 (racecar)
  Duration: 3 hours and 19 minutes (2013-05-14 12:14:45 - PRESENT)

#2
Usage Type: ADDTL_FS_GB (3)
  Gear ID: 5192624e6892dfcb3f00000e (foo)
  Duration: 15 seconds (2013-05-14 12:16:33 - 2013-05-14 12:16:48)

#3
Usage Type: ADDTL_FS_GB (2)
  Gear ID: 5192624e6892dfcb3f00000e (foo)
  Duration: 3 hours and 17 minutes (2013-05-14 12:16:48 - PRESENT)
```

The following table provides more information on the output of this command.

Field	Description
-------	-------------

Field	Description
Usage Type	GEAR_USAGE is related to how long a gear has been in use with the gear size in parentheses. ADDTL_FS_GB is related to how long additional storage has been in use on a gear with the number of GBs in parentheses.
Gear ID	Gear ID indicates the UUID of the relevant gear with the associated application name in parentheses.
Duration	Duration indicates the start and end time of the gear (or start time and PRESENT if still in use).

On the broker host, view resource usage for all users with:

```
# oo-admin-ctl-usage --list
```

Example 8.7. Viewing Resource Usage for All Users

```
# oo-admin-ctl-usage --list
Errors/Warnings will be logged to terminal
2013-05-14 15:48:54 -0400 INFO::
----- STARTED -----

User: username1
  Gear: 518bcaa26892dfcb74000001, UsageType: GEAR_USAGE, Usage:
23.32543548687111
  Gear: 518bcb876892dfcb74000017, UsageType: GEAR_USAGE, Usage:
23.32543548687111
  Gear: 519254d36892df8f9000000b, UsageType: ADDTL_FS_GB, Usage:
0.05429166666666666
  Gear: 519254d36892df8f9000000b, UsageType: GEAR_USAGE, Usage:
0.08019000000000001
  Gear: 519258d46892df156600001f, UsageType: GEAR_USAGE, Usage:
4.287655764648889
User: username2
  Gear: 5192624e6892dfcb3f00000e, UsageType: ADDTL_FS_GB, Usage: 0.0042325
  Gear: 5192624e6892dfcb3f00000e, UsageType: ADDTL_FS_GB, Usage:
3.5350574313155554
  Gear: 519262ef6892df43f7000001, UsageType: GEAR_USAGE, Usage:
3.5691388202044445
2013-05-14 15:48:54 -0400 INFO::
----- ENDED, #Errors: 0, #Warnings: 0 -----
```

The following table provides more information on the output of this command.

Field	Description
User	User names the user accumulating the resource usage.
Gear	Gear indicates the UUID of the relevant gear.

Field	Description
UsageType	GEAR_USAGE is related to how long a gear has been in use. ADDTL_FS_GB is related to how long additional storage has been in use on a gear.
Usage	Usage lists the duration of the gear (in hours).

8.8. Enabling Syslog

With the release of OpenShift Enterprise 2.1, you can now choose to send log files to **Syslog** instead of their default locations, which are found in varied locations across an OpenShift Enterprise instance. Placing them into a single location helps you to analyze broker, node, gear, and Management Console errors. See the following sections for more information on how to enable **Syslog** for OpenShift Enterprise components.



Note

Instructions for configuring a **Syslog** implementation to route to alternative destinations, such as a remote logging server, are outside of the scope of this guide. The implementation provided by Red Hat Enterprise Linux 6 is **Rsyslog**, which writes to the `/var/log/messages` file by default. See the [Red Hat Enterprise Linux 6 Deployment Guide](#) for information on viewing and managing log files if using **Rsyslog**.

8.8.1. Enabling Syslog for Broker Components

Set the **SYSLOG_ENABLED** variable in the `/etc/openshift/broker.conf` file to **true** in order to group `production.log`, `user_action.log`, and `usage.log` into the **syslog** file:

```
SYSLOG_ENABLED=true
```

The default location for the **syslog** file is `/var/log/messages`, but this is configurable. However, in the **syslog** file, these share the same program name. In order to distinguish between the log files, the following applies:

- ✦ Messages usually sent to `production.log` will have `src=app` in each log line.
- ✦ Messages usually sent to `user_action.log` will have `src=useraction` in each log line.
- ✦ Messages usually sent to `usage.log` will have `src=usage` in each log line.

8.8.2. Enabling Syslog for Node Components

You can configure node hosts to send node platform and **Apache** logs to **Syslog** instead of writing to their default log file locations. After enabling **Syslog** logging, messages from the various node components are grouped and sent to the configured **Syslog** implementation. This method of aggregating log messages gives you the option to further configure your **Syslog** implementation to send logs directly to a remote logging server or monitoring solution without the messages being written locally to disk.



Note

Instructions for configuring a **Syslog** implementation to route to alternate destinations, such as a remote logging server, are outside of the scope of this guide. The implementation provided by Red Hat Enterprise Linux 6 is **Rsyslog**, which writes to the `/var/log/messages` file by default. See the *Red Hat Enterprise Linux 6 Deployment Guide* for information on viewing and managing log files if using **Rsyslog**.

Procedure 8.4. To Enable Syslog for Apache:

1. Configure **Apache** to send log messages to **Syslog** by adding the following option in the `/etc/sysconfig/httpd` file:

```
OPTIONS="-DOpenShiftFrontendSyslogEnabled"
```



Important

All options must be on the same line. For example, in [Section 8.5.2, “Enabling Application and Gear Context in Node Component Logs”](#) another option for **Apache** log files is explained. If both options are desired, the line must use the following syntax:

```
OPTIONS="-Option1 -Option2"
```

2. Restart the **httpd** service for the **Apache** changes to take effect:

```
# service httpd restart
```

Procedure 8.5. To Enable Syslog for the Node Platform:

1. Configure the node platform to send log messages to **Syslog** by editing the `/etc/openshift/node.conf` file. Add the following line and any or all of the described optional settings that follow:

```
PLATFORM_LOG_CLASS=SyslogLogger
```

Optional Threshold Setting:

Add the following line to include messages with priorities up to and including the set threshold. Replace **priority** in the following line with one of the levels listed at <http://ruby-doc.org/stdlib-1.9.3/libdoc/syslog/rdoc/Syslog.html#method-c-log>:

```
PLATFORM_SYSLOG_THRESHOLD=priority
```

Optional Trace Log Setting:

Add the following line to include trace logs that were previously directed to the default `/var/log/openshift/node/platform-trace.log` file:

```
PLATFORM_SYSLOG_TRACE_ENABLED=1
```

- Restart the **ruby193-mcollective** service for the node platform changes to take effect:

```
# service ruby193-mcollective restart
```

- When **Syslog** support is enabled for the node platform, the **local0 Syslog** facility is used to log messages. By default, the **/etc/rsyslog.conf** file does not log platform debug messages. If you are using **Rsyslog** as your **Syslog** implementation, add the following line to the **/etc/rsyslog.conf** file to enable platform debug message logging. If necessary, replace **/var/log/messages** with your chosen logging destination:

```
local0.*;*.info;mail.none;authpriv.none;cron.none /var/log/messages
```

Then restart the **rsyslog** service:

```
# service rsyslog restart
```

With this change, all log messages using the **local0** facility are sent to the configured logging destination.

8.8.3. Enabling Syslog for Cartridge Logs from Gears

By default, cartridge logs are written to the **\$OPENSIFT_LOG_DIR** directory of an application. You can configure **logshifter** on node hosts to instead have gears send their cartridge logs to **Syslog**. Starting with OpenShift Enterprise 2.1.7, you can also have them sent to both **Syslog** and the **\$OPENSIFT_LOG_DIR** directory at the same time.

Procedure 8.6. To Enable Syslog for Cartridge Logs from Gears:

- Edit the **/etc/openshift/logshifter.conf** file on the node host. The default value for the **outputType** setting is **file**, which results in gears sending cartridge logs to the **\$OPENSIFT_LOG_DIR** directory. Change this setting to **syslog** to have them sent to **Syslog**:

```
outputType=syslog
```

Alternatively, starting with OpenShift Enterprise 2.1.7, you can choose to change the **outputType** setting instead to **multi**, which results in logs being written using both **file** and **syslog** at the same time.

- Ask affected owners of existing applications to restart their applications for the changes to take effect. They can restart their applications with the following commands:

```
$ rhc app stop -a appname
$ rhc app start -a appname
```

Alternatively, you can restart all gears on affected node hosts. The downtime caused by restarting all gears is minimal and normally lasts a few seconds:

```
# oo-admin-ctl-gears restartall
```



Important

If the **outputTypeFromEnviron** setting in the `/etc/openshift/logshifter.conf` file is set to **true**, application owners are allowed to override the global **outputType** setting using a **LOGSHIFTER_OUTPUT_TYPE** environment variable in their application. See the [OpenShift Enterprise User Guide](#) for more information.

Cron Syslog Configuration

With the **mmopenshift** plug-in, all **Cron** cartridges will output all log information to the configured gear log file (`/var/log/openshift_gears` in the example 8.1). It may be necessary for system-level **Cron** logs to be separated from the gear logs for troubleshooting purposes. System-level **Cron** messages are tagged with **cron_sys_log** and can be separated into another file by adding the below to the `/etc/syslog.conf` **Syslog** configuration file:

```
:syslogtag, contains, "cron_sys_log:" /var/log/openshift_cron_cartridges.log
&~

action(type="mmopenshift")

if $!OpenShift!OPENSIFT_APP_UUID != '' then
  # annotate and log syslog output from gears specially
  *.*      action(type="omfile" file="/var/log/openshift_gears"
template="OpenShift")
else
  # otherwise send syslog where it usually goes
  *.info;mail.none;authpriv.none;cron.none  action(type="omfile"
file="/var/log/messages")
```

The **:syslogtag** entry must be placed before the ***.* mmopenshift** entry to prevent **Cron** system logs from going to both the **openshift_cron_cartridges** log and the **openshift_gears** log. The **&~** tells **Rsyslog** to stop processing log entries if the filter condition on the previous line is met.

Enabling Application and Gear Context for Cartridge Logs

To provide context to cartridge logs aggregated to **Syslog**, a message modification plug-in for **Rsyslog** called **mmopenshift** can be used to add gear metadata to the cartridge logs. The plug-in can be configured to add metadata items to the JSON properties of each message that **Rsyslog** receives from a gear.

Due to configuration file format requirements, a newer version of **Rsyslog**, provided by the `rsyslog7` package, must be installed and configured to enable this feature. The **mmopenshift** plug-in also only works for messages that have the **\$!uid** JSON property, which can be added automatically when the **imuxsock** plug-in is enabled with the following options:

- » SysSock.Annotate
- » SysSock.ParseTrusted
- » SysSock.UsePIDFromSystem

Procedure 8.7. To Enable Application and Gear Context for Cartridge Logs:

1. Install the **mmopenshift** plug-in, which requires the *rsyslog7* package, on the node host. Because installing the *rsyslog7* package where the *rsyslog* package is already installed can cause conflicts, consult the following instructions relevant to your node host.

If the *rsyslog* package is already installed, use a **yum** shell to remove the *rsyslog* package and install the *rsyslog7* and *rsyslog7-mmopenshift* packages safely:

- a. Stop the **Rsyslog** service:

```
# service rsyslog stop
```

- b. Open a **yum** shell:

```
# yum shell
```

- c. Run the following commands inside of the **yum** shell:

```
> erase rsyslog
> install rsyslog7 rsyslog7-mmopenshift
> transaction run
> quit
```

The *rsyslog* package is uninstalled and a newer version of **Rsyslog** takes its place. The *rsyslog7-mmopenshift* package is also installed, which provides the **mmopenshift** module.

Alternatively, if the *rsyslog* package is not already installed, or if *rsyslog7* is already the only version of **Rsyslog** installed, install the **mmopenshift** module using the following command:

```
# yum install rsyslog7 rsyslog7-mmopenshift
```

2. Review the existing `/etc/rsyslog.conf` file, if relevant, and note any important default or custom settings. This includes changes that were made with the instructions described in [Section 8.8.2, “Enabling Syslog for Node Components”](#). Next, make any required changes to ensure that the new `/etc/rsyslog7.conf` file contains those changes. Note that some settings may be different between `/etc/rsyslog.conf` and `/etc/rsyslog7.conf.rpmnew`; see <http://www.rsyslog.com/doc/v7-stable/> for more information. Once complete, take a backup of `/etc/rsyslog.conf` and move `/etc/rsyslog.conf.rpmnew` to `/etc/rsyslog.conf`



Important

A sample section of an `/etc/rsyslog7.conf.rpmnew` file is provided at [Example 8.8, “Sample Configuration Settings in /etc/rsyslog7.conf”](#) which depicts how the **mmopenshift** plug-in can be enabled for **Rsyslog**. However, it is not meant to represent a comprehensive `/etc/rsyslog7.conf` file or be fully comparable to the standard `/etc/rsyslog.conf` configuration.

3. Edit the `/etc/rsyslog7.conf` file and add the following lines under the **MODULES** section to enable the **imuxsock** plug-in and the **mmopenshift** plug-in:

```
module(load="imuxsock" SysSock.Annotate="on" SysSock.ParseTrusted="on"
SysSock.UsePIDFromSystem="on")
module(load="mmopenshift")
```

4. Edit the `/etc/rsyslog7.conf` file and comment out the following line under the **MODULES** section to configure the **imuxsock** plug-in:

```
#ModLoad imuxsock
```

5. Edit the `/etc/rsyslog7.conf` file and comment out the following lines to disable the **imjournal** plug-in:

```
$ModLoad imjournal
$OmitLocalLogging on
$IMJournalStateFile imjournal.state
```

6. Edit the `/etc/rsyslog7.conf` file to have **Syslog** search the `/etc/rsyslog7.d` directory for configuration files:

```
#$IncludeConfig /etc/rsyslog.d/*.conf
$IncludeConfig /etc/rsyslog7.d/*.conf
```

7. Examine the `/etc/rsyslog.d` directory and copy any configuration files that are needed in `/etc/rsyslog7.d` directory for the **Rsyslog7** logging configuration.
8. Create a gear log template file in the **Rsyslog7** directory. This defines the format of the gear logs, including sufficient parameters to distinguish gears from each other. This example template can be modified to suit the requirements of your log analysis tools. For more information on template configuration instructions, see <http://www.rsyslog.com/doc/v7-stable/configuration/templates.html>:

```
# vi /etc/rsyslog7.d/openshift-gear-template.conf
template(name="OpenShift" type="list") {
    property(name="timestamp" dateFormat="rfc3339")
    constant(value=" ")
    property(name="hostname")
    constant(value=" ")
    property(name="syslogtag")
    constant(value=" app=")
    property(name="$!OpenShift!OPENSIFT_APP_NAME")
    constant(value=" ns=")
    property(name="$!OpenShift!OPENSIFT_NAMESPACE")
    constant(value=" appUuid=")
    property(name="$!OpenShift!OPENSIFT_APP_UUID")
    constant(value=" gearUuid=")
    property(name="$!OpenShift!OPENSIFT_GEAR_UUID")
    property(name="msg" spifno1stsp="on")
    property(name="msg" droplastlf="on")
    constant(value="\n")
}
```

9. Add the following lines to the `/etc/rsyslog7.conf` file under the **RULES** section to configure the **mmopenshift** plug-in to use the template from the previous step. The following example logs all gear messages to the `/var/log/openshift_gears` file and all other messages to the `/var/log/messages` file, but these destinations are configurable to a different destination:

```
module(load="mmopenshift")
action(type="mmopenshift")
```

```

if $!OpenShift!OPENSIFT_APP_UUID != '' then
    *.* action(type="omfile" file="/var/log/openshift_gears"
template="OpenShift")
else {
    *.info;mail.none;authpriv.none;cron.none    action(type="omfile"
file="/var/log/messages")
}

```

Also, comment out the following line:

```

# *.info;mail.none;authpriv.none;cron.none
/var/log/messages

```

10. Start or restart the **rsyslog** service and ensure it starts persistently across reboots:

```

# service rsyslog restart
# chkconfig rsyslog on

```

Example 8.8. Sample Configuration Settings in /etc/rsyslog7.conf

```

##### MODULES #####

# The imjournal module bellow is now used as a message source instead of
imuxsock.
#$ModLoad imuxsock # provides support for local system logging (e.g. via
logger command)
#$ModLoad imjournal # provides access to the systemd journal
$ModLoad imklog    # provides kernel logging support (previously done by
rklogd)
#$ModLoad immark  # provides --MARK-- message capability
module(load="imuxsock" SysSock.Annotate="on" SysSock.ParseTrusted="on"
SysSock.UsePIDFromSystem="on")
module(load="mmopenshift")

# Provides UDP syslog reception
#$ModLoad imudp
#$UDPServerRun 514

# Provides TCP syslog reception
#$ModLoad imtcp
#$InputTCPListenerRun 514

##### GLOBAL DIRECTIVES #####

# Where to place auxiliary files
$WorkDirectory /var/lib/rsyslog

# Use default timestamp format
$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat

# File syncing capability is disabled by default. This feature is usually
not required,
# not useful and an extreme performance hit
#$ActionFileEnableSync on

```

```

# Include all config files in /etc/rsyslog7.d/
#$IncludeConfig /etc/rsyslog.d/*.conf
$IncludeConfig /etc/rsyslog7.d/*.conf

# Turn off message reception via local log socket;
# local messages are retrieved through imjournal now.
#$OmitLocalLogging on

# File to store the position in the journal
#$IMJournalStateFile imjournal.state

#### RULES ####

# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.* /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
#*.info;mail.none;authpriv.none;cron.none
/var/log/messages
action(type="mmopenshift")
if $!OpenShift!OPENSIFT_APP_UUID != '' then
  # annotate and log syslog output from gears specially
  *.* action(type="omfile" file="/var/log/openshift_gears"
template="OpenShift")
else
  # otherwise send syslog where it usually goes
  *.info;mail.none;authpriv.none;cron.none action(type="omfile"
file="/var/log/messages")

# The authpriv file has restricted access.
authpriv.* /var/log/secure

# Log all the mail messages in one place.
mail.* -
/var/log/maillog

# Log cron stuff
cron.* /var/log/cron

# Everybody gets emergency messages
*.emerg :omusrmsg:*

# Save news errors of level crit and higher in a special file.
uucp,news.crit /var/log/spooler

# Save boot messages also to boot.log
local7.*
/var/log/boot.log

```

8.8.4. Enabling Syslog for Management Console Components

Set the **SYSLOG_ENABLED** variable in the `/etc/openshift/console.conf` file to **true** in order to send **production.log** log messages to the **syslog** file:

```
SYSLOG_ENABLED=true
```

The default location for the **syslog** file is `/var/log/messages`, but this is configurable. However, in the **syslog** file, different log files share the same program name. Management Console log messages will have **src=app** inserted into each log line.

Chapter 9. Command Reference

9.1. Broker Administration Commands

These tools are installed with the `openshift-origin-broker` and `openshift-origin-broker-util` RPMs.

9.1.1. oo-accept-broker

This command checks that your broker setup is valid and functional. It is run without options on a broker host.

If there are no errors, it displays **PASS** and exits with return code 0. With the `-v` option added, it displays the current checks that are being performed.

If there are errors, they are displayed, and the return code is the number of errors.

Example 9.1. Checking For Errors With `oo-accept-broker`

```
# oo-accept-broker -v

INFO: SERVICES: DATA: mongo, Auth: mongo, Name bind
INFO: AUTH_MODULE: rubygem-openshift-origin-auth-mongo
INFO: NAME_MODULE: rubygem-openshift-origin-dns-bind
INFO: Broker package is: openshift-origin-broker
INFO: checking packages
INFO: checking package ruby
INFO: checking package rubygems
INFO: checking package rubygem-rails
INFO: checking package rubygem-passenger
INFO: checking package rubygem-openshift-origin-common
INFO: checking package rubygem-openshift-origin-controller
INFO: checking package openshift-origin-broker
INFO: checking ruby requirements
INFO: checking ruby requirements for openshift-origin-controller
INFO: checking ruby requirements for config/application
INFO: checking firewall settings
INFO: checking services
INFO: checking datastore
INFO: checking cloud user authentication
INFO: auth plugin =
/var/www/openshift/broker/config/initializers/broker.rb:2: uninitialized
constant ApplicationObserver (NameError) from -:6
INFO: checking dynamic dns plugin
INFO: checking messaging configuration
PASS
```

9.1.2. oo-accept-systems

This command checks that node host **PUBLIC_HOSTNAME** and **PUBLIC_IP** configuration settings are globally valid and unique. It also checks the cartridges installed on the nodes and the status of the broker's cache. It is run without options on the broker host.

If there are no errors, the command displays **PASS** and exits with return code 0. With the **-v** option added, it displays the current checks that are being performed.

If there are errors, they are displayed, and the return code is the number of errors.

Example 9.2. Checking For Errors With `oo-accept-systems`

```
# oo-accept-systems -v

INFO: checking that each public_hostname resolves to external IP
INFO: PUBLIC_HOSTNAME node1.example.com for node2.example.com resolves to
10.4.59.136
INFO: PUBLIC_HOSTNAME node2.example.com for node1.example.com resolves to
10.4.59.133
INFO: checking that each public_hostname is unique
INFO: checking that public_ip has been set for all nodes
INFO: PUBLIC_IP 10.4.59.136 for node1.example.com
INFO: PUBLIC_IP 10.4.59.133 for node2.example.com
INFO: checking that public_ip is unique for all nodes
INFO: checking that all node hosts have cartridges installed
INFO: cartridges for node1.example.com: cron-1.4|ruby-1.9|perl-
5.10|jenkins-client-1.4|diy-0.1|jenkins-1.4|php-5.3|haproxy-
1.4|abstract|abstract-jboss|jbosseap-6.0|mysql-5.1|postgresql-8.4|ruby-
1.8|jbossews-1.0|python-2.6|abstract-httpd
INFO: cartridges for node2.example.com: diy-0.1|jenkins-client-1.4|cron-
1.4|jbossews-1.0|php-5.3|abstract-httpd|ruby-1.9|python-2.6|jbosseap-
6.0|perl-5.10|abstract|postgresql-8.4|abstract-jboss|ruby-1.8|jenkins-
1.4|haproxy-1.4|mysql-5.1
INFO: checking that same cartridges are installed on all node hosts
INFO: checking that broker's cache is not stale
INFO: API reports carts: diy-0.1|jbossews-1.0|php-5.3|ruby-1.9|python-
2.6|jbosseap-6.0|perl-5.10|ruby-1.8|jenkins-1.4|jenkins-client-1.4|cron-
1.4|postgresql-8.4|haproxy-1.4|mysql-5.1
PASS
```

9.1.3. `oo-admin-chk`

This command checks that application records in the MongoDB datastore are consistent with the gears that are present on the node hosts. With the **-v** option added, it displays the current checks that are being performed.

Example 9.3. Checking For MongoDB Consistency with `oo-admin-chk`

```
# oo-admin-chk -v

Started at: 2013-05-03 03:36:28 +1000
Time to fetch mongo data: 0.005s
Total gears found in mongo: 3
Time to get all gears from nodes: 20.298s
Total gears found on the nodes: 3
Total nodes that responded : 1
Checking application gears and ssh keys on corresponding nodes:
```

```

51816f026892dfec74000004 : String... OK
518174556892dfec74000040 : String... OK
518176826892dfec74000059 : String... OK
Checking node gears in application database:
51816f026892dfec74000004... OK
518174556892dfec74000040... OK
518176826892dfec74000059... OK
Success
Total time: 20.303s
Finished at: 2013-05-03 03:36:49 +1000

```

With the **-l** option added, additional levels of checks can be included:

```
# oo-admin-chk -l 1 -v
```

9.1.4. oo-admin-clear-pending-ops

The **oo-admin-clear-pending-ops** removes stuck user operations from the application queue, so that they no longer hold up the queue preventing other operations from proceeding on that application.

```
oo-admin-clear-pending-ops [options]
```

The available options are:

Option	Description
-t, --time <i>n</i>	Deletes pending operations older than <i>n</i> hours. (Default: 1)
-u, --uuid <i>uuid</i>	Prunes only applications with the given UUID.



Note

In most scenarios, you should not need to use the **oo-admin-clear-pending-ops** command directly. It is most commonly run automatically by the **ose-upgrade** tool as part of the upgrade process described in the [OpenShift Enterprise Deployment Guide](#). This ensures the database is in a consistent state before data migrations happen.

9.1.5. oo-admin-console-cache

The **oo-admin-console-cache** command manages the Management Console Rails application's cache.

```
oo-admin-console-cache [-c | --clear] [-q | --quiet]
```

The available options are:

Option	Description
-c, --clear	Removes all entries from the Management Console Rails application's cache.
-q, --quiet	Shows as little output as possible.

See Also:

- [Section 5.4, “Adding QuickStarts to the Management Console”](#)
- [Section 5.6, “Disabling Obsolete Cartridges”](#)
- [Section 5.2, “Installing and Removing Custom and Community Cartridges”](#)

9.1.6. oo-admin-broker-auth

This command recreates broker authentication keys. If **AUTH_SALT** is changed in **/etc/openshift/broker.conf**, restart the broker service and run the **oo-admin-broker-auth** command to recreate authentication tokens for all applicable gears.

```
# oo-admin-broker-auth
```

9.1.7. oo-admin-broker-cache

This command clears the broker Rails application cache.

```
# oo-admin-broker-cache --clear
```

9.1.8. oo-admin-ctl-app

This command provides administration command options for applications.

```
# oo-admin-ctl-app
```

Modifying the HAProxy Multiplier

The HAProxy multiplier sets the ratio of how many HAProxy cartridges are enabled for application scaling. Setting the multiplier number to **2** means that for every two gears, one will have HAProxy enabled. Alternatively, you can set the minimum and maximum number of HAProxy cartridges allowed in scaling.

Modify the number of HAProxy multiplier using the **oo-admin-ctl-app** command with the **--multiplier** option.

```
# oo-admin-ctl-app -l username -a appname --cartridge haproxy-1.4 -c set-multiplier --multiplier 2
```

9.1.9. oo-admin-ctl-authorization

Use this command to either delete only expired authorization tokens for a user, or to delete both valid and expired authorization tokens for a user.

On the broker host, delete all expired authorization tokens for a user with:

```
# oo-admin-ctl-authorization -c expire
```

On the broker host, delete all valid and expired authorization tokens for a user with:

```
# oo-admin-ctl-authorization -c revoke_all
```

9.1.10. oo-admin-ctl-district

9.1.10. oo-admin-ctl-district

This command performs district operations, such as creating or removing districts and adding or removing nodes from districts. It can also be used to tag nodes in districts with a region and zone.

See Also:

- [Section 6.3, “Managing Districts”](#)
- [Section 6.4, “Managing Regions and Zones”](#)

9.1.11. oo-admin-ctl-domain

This command is used to query and control a user's domain. It produces detailed output in YAML format.

```
# oo-admin-ctl-domain
```

See Also:

- [Section 3.3, “Removing User Data”](#)
- [Section 3.4, “Removing a User”](#)

9.1.12. oo-admin-ctl-region

The **oo-admin-ctl-region** command is used to create, list, or destroy regions and add or remove zones within a given region.

See Also:

- [Section 6.4, “Managing Regions and Zones”](#)

9.1.13. oo-admin-ctl-team

The **oo-admin-ctl-team** tool manages global teams and is invoked with a set of commands using the **-c** or **--command** option:

```
oo-admin-ctl-team -c command [options]
```

The available commands are:

Command	Description
list	Lists all teams.

Command	Description
create	Creates a new team. Requires either both the --name and --maps-to options, or both the --groups and --config-file options. For example: <div style="border: 1px solid black; padding: 5px; margin: 10px 0;"> <pre># oo-admin-ctl-team create --name team_name --maps-to group_name</pre> </div> Alternatively: <div style="border: 1px solid black; padding: 5px; margin: 10px 0;"> <pre># oo-admin-ctl-team create --groups group_name_1,group_name_2 --config-file file_name</pre> </div>
update	Updates an existing team LDAP correspondance. Requires both the --name and --maps-to options.
delete	Deletes a team. Requires the --name option.
show	Displays a team and its members. Requires either the --name or --maps-to option.
sync	Syncs global teams with the LDAP groups. Requires the --config-file option.
sync-to-file	Generates a sync file for review. No changes are made to the teams and their members. Requires the --out-file and --config-file options.
sync-from-file	Syncs from a file. Requires the --in-file option.

Other options are:

Option	Description
--broker path	Specifies the path to the broker.
--create-new-users	Creates new users in OpenShift if they do not exist.
--remove-old-users	Removes members from a team that are no longer in the group.

See Also:

- » [Section 4.2, “Creating Global Teams and Synchronizing with LDAP Groups”](#)

9.1.14. oo-admin-ctl-usage

The **oo-admin-ctl-usage** displays usage data for all users. The output includes user names, gears, usage type and duration.

```
oo-admin-ctl-usage --list [--enable-logger]
```

The following options are available for OpenShift Enterprise:

Option	Description
--list	List usage data.
--enable-logger	Print error and warning messages to the log file instead of to the terminal.

The following table provides more information on the output of the **--list** option.

Field	Description
User	User names the user accumulating the resource usage.
Gear	Gear indicates the UUID of the relevant gear.
UsageType	GEAR_USAGE is related to how long a gear has been in use. ADDTL_FS_GB is related to how long additional storage has been in use on a gear.
Usage	Usage lists the duration of the gear (in hours).

See Also:

- [Section 8.7.2, “Viewing Accumulated Usage Data”](#)

9.1.15. oo-admin-ctl-user

This command administers users on the system. Some features are disabled for user accounts by default, such as the ability to add additional storage to gears or add private SSL certificates to aliases, and require this tool in order to enable them or set an explicit allowed value for the user.

Option	Description
-l, --login <i>Username</i>	Login name for an OpenShift Enterprise user account. Required unless -f is used.
-f, --logins-file <i>File_Name</i>	File containing one login name per line. Required unless -l is used.
-c, --create	Create user account(s) for the specified login name(s) if they do not already exist.
--setmaxdomains <i>Number</i>	Set the maximum number of domains a user is allowed to use.
--setmaxgears <i>Number</i>	Set the maximum number of gears a user is allowed to use.
--setmaxteams <i>Number</i>	Set the maximum number of teams a user is allowed to create.
--allowviewglobalteams true false	Add or remove the capability for a user to search and view any global team.
--allowprivatessslcertificates true false	Add or remove the capability for a user to add private SSL certificates.
--addgearsizes <i>Gear_Size</i>	Add a gear size to the capabilities for a user.
--removegearsizes <i>Gear_Size</i>	Remove a gear size from the capabilities for a user.
--allowha true false	Allow or disallow the high-availability applications capability for a user.
-q, --quiet	Suppress non-error output.
-h, --help	Show usage information.

Many common administrative tasks make use of the **oo-admin-ctl-user** command:

- See [Section 2.4, “Enabling Support for High-Availability Applications”](#) for more information on the high-availability applications capability.
- See [Section 3.1, “Creating a User”](#) for more information on creating users.

- See [Section 4.1, “Setting the Maximum Number of Teams for Specific Users”](#) and [Section 4.2.2, “Enabling Global Team Visibility”](#) for more information on team options.
- See [Section 6.1, “Adding or Modifying Gear Profiles”](#) for more information on modifying gear size capabilities.
- See [Section 6.7, “Setting Gear Quotas and Sizes for Specific Users”](#) for more information on setting gear quotas.
- See [Section 8.7.1, “Setting Tracked and Untracked Storage”](#) for more information on setting maximum tracked and untracked storage per gear.

9.1.16. oo-admin-move

This command moves a gear from one node to another. Note that moving gears requires the `rsync_id_rsa` private key in the broker host's `/etc/openshift/` directory and the public key in each node host's `/root/.ssh/authorized_keys` file as explained in the *OpenShift Enterprise Deployment Guide* at <https://access.redhat.com/site/documentation>. A gear retains its UNIX UID when moved, therefore cross-district moves are only allowed when the destination district has the same gear UID available.

To move a gear from one node to another, use the `oo-admin-move` command on the broker host, specifying the desired gear's UUID and the node host you wish to move the gear to:

Example 9.4. Moving a Gear From One Node to Another

```
# oo-admin-move --gear_uuid 3baf79139b0b449d90303464dfa8dd6f -i
node2.example.com
```

9.1.17. oo-admin-repair

This command checks for and fixes various inconsistencies in the MongoDB datastore on the broker. For example, because a mismatch in SSH keys can be a potential security risk, the tool fixes any mismatches found by taking information from the broker datastore and pushing it to the gear. See the `--help` output for additional uses.

```
# oo-admin-repair
```

9.1.18. oo-admin-upgrade

This command upgrades custom and community cartridges on a gear to the latest available version and applies gear-level changes that affect cartridges. See [Section 5.3, “Upgrading Custom and Community Cartridges”](#) for more information on the upgrade process and `oo-admin-upgrade` command usage.



Important

The `oo-admin-upgrade` tool is also often required when applying asynchronous errata updates provided by Red Hat for OpenShift Enterprise. See the latest *OpenShift Enterprise Deployment Guide* at <https://access.redhat.com/site/documentation> for usage instructions as it applies to these types of updates.

9.1.19. oo-admin-usage

The **oo-admin-usage** command displays a resource usage report for a particular user, or aggregated usage data of all users. The output includes usage type, gear ID, and duration.

```
oo-admin-usage [-l username] [options]
```

If **-l *username*** is omitted, the command displays aggregated data on all users.

Other options further restrict the output:

Option	Description
-a, --app <i>application_name</i>	Filters usage data by the given application name.
-g, --gear <i>gear_id</i>	Filters usage data by the given gear ID.
-s, --start <i>start_date</i>	Filters usage data by the given start date, expressed as ISO dates (YYYY-MM-DD).
-e, --end <i>end_date</i>	Filters usage data by the given end date, expressed as ISO dates (YYYY-MM-DD).

The following table provides more information on the output of this command:

Field	Description
Usage Type	GEAR_USAGE is related to how long a gear has been in use with the gear size in parentheses. ADDTL_FS_GB is related to how long additional storage has been in use on a gear with the number of GBs in parentheses.
Gear ID	Gear ID indicates the UUID of the relevant gear with the associated application name in parentheses.
Duration	Duration indicates the start and end time of the gear (or start time and PRESENT if still in use).

See Also:

- ✦ [Section 8.7.2, “Viewing Accumulated Usage Data”](#)

9.1.20. oo-admin-ctl-cartridge

In OpenShift Enterprise 2.1, the **oo-admin-ctl-cartridge** command facilitates cartridge management on the broker, including importing cartridge manifests from nodes and activating or deactivating cartridges. This command must be used to ensure that newly installed or updated cartridges can be used in applications.



Note

This command is not used for cartridge management in OpenShift Enterprise 2.0.

See Also:

- ✦ [Section 5.1, “Managing Cartridges on Broker Hosts”](#)

9.1.21. oo-register-dns

9.1.2.1. oo-register-dns

This command updates DNS A records in BIND by wrapping an **nsupdate** command. Normally this command is used for broker or node hosts, although it can be used for other infrastructure hosts. Do not use this command to change DNS records for applications and gears, because these are CNAME records.

```
# oo-register-dns
```

9.2. Node Administration Commands

These tools are installed on node hosts with the **openshift-origin-node-util** RPM.



Note

Node hosts do not have administrative access to other nodes or to brokers, so running the commands described in this section only affect the node on which they are run.

9.2.1. oo-accept-node

This command checks that your node setup is valid and functional and that its gears are in good condition. It is run without options on a node host.

If there are no errors, it displays **PASS** and exits with return code 0. With the **-v** option added, it displays the current checks that are being performed.

If there are errors, they are displayed, and the return code is the number of errors.

```
# oo-accept-node -v
```

9.2.2. oo-admin-ctl-gears

This command is used to control gears on a node host. It is used by the **openshift-gears** service at boot time to activate existing gears, and can be used manually by an administrator.

```
# oo-admin-ctl-gears
```

9.2.3. oo-idler-stats

This command displays basic statistics about gears on a node.

```
# oo-idler-stats
```

9.2.4. Idler Commands

The idler is a tool for shutting down gears that have not been used recently in order to reclaim their resources and overcommit the node host's resource usage.

9.2.4.1. oo-last-access

This command checks for the last web or Git access an application has had, then records it in the gear operations directory. Running this command regularly in a cron job allows automatic idling of stale gears.

Example 9.5. A Cron auto-idler Script

```
# run the last-access compiler hourly
0 * * * * /usr/sbin/oo-last-access > /var/lib/openshift/last_access.log
2>&1
# run the auto-idler twice daily and idle anything stale for 24 hours
30 7,19 * * * /usr/sbin/oo-auto-idler idle --interval 12
```

9.2.4.2. oo-auto-idler

This command retrieves a list of gears that are not receiving any web traffic, then idles them. Red Hat recommends that this command be run regularly in a cron job.

Appendix A. Revision History

Revision 2.2-6	Wed Nov 23 2016	Ashley Hardin
BZ 1394396: Updated Chapter 5, Cartridge Management to add an Important box about <i>atomcat7</i> known issue.		
Revision 2.2-5	Thu Sep 08 2016	Ashley Hardin
BZ 1366393, BZ 1366397: Bug Fixes		
Revision 2.2-4	Wed Sep 09 2015	Brice Fallon-Freeman
BZ 1159287: Updated link in Section 8.8, "Enabling Syslog" .		
Revision 2.2-3	Fri Jun 05 2015	Vikram Goyal
BZ 1121699: No documentation for a number of new OSE 2.1 commands. New topics added: Section 9.1.4 "oo-admin-clear-pending-ops" , Section 9.1.5, "oo-admin-console-cache" , Section 9.1.13, "oo-admin-ctl-team" , Section 9.1.14, "oo-admin-ctl-usage" and Section 9.1.19, "oo-admin-usage" .		
Revision 2.2-2	Fri Apr 10 2015	Brice Fallon-Freeman
OpenShift Enterprise 2.2.5 release. BZ 1124840: Reworked Section 2.10, "Backup and Recovery" and updated subsections with more details for both host types. BZ 1178039: Updated table in Section 2.11, "Component Timeout Value Locations" to include Background Thread. BZ 1146254: Added section Section 4.2.1, "Encrypting an LDAP Global Team Connection" . BZ 1166654: Updated Section 8.4.2, "Verifying Functionality with Administration Commands" with more context and verification commands. BZ 1148083: Updated Section 8.5.4.2, "Supported Watchman Plug-ins" with Metrics logging information and an example.		
Revision 2.2-1	Wed Dec 10 2014	Timothy Poitras
OpenShift Enterprise 2.2.2 release. BZ 1159182: Updated Section 2.4, "Enabling Support for High-Availability Applications" with information on setting <code>HA_DNS_PREFIX</code> parameters consistently. Added Section 2.12, "Enabling Network Isolation for Gears" . Updated Section 3.1, "Creating a User" and Section 9.1.15, "oo-admin-ctl-user" to note <code>-f</code> option for the <code>oo admin-ctl-user</code> command. BZ 1167810: Updated Section 8.8.3, "Enabling Syslog for Cartridge Logs from Gears" for changes in <code>Rsyslog</code> packaging. BZ 1146147: Updated Section 8.5.4.2, "Supported Watchman Plug-ins" with information on configuring the Watchman throttler plug-in. BZ 1158691: Updated Section 9.2.4.1, "oo-last-access" to fix file paths.		
Revision 2.2-0	Tue Nov 4 2014	Brice Fallon-Freeman
OpenShift Enterprise 2.2 release. Updated Section 6.5, "Gear Placement Algorithm" to link to the <i>OpenShift Enterprise Deployment Guide</i> for information on the gear placement plug-in. Added Section 6.8, "Restricting Gear Sizes for Cartridges" . Added Section 6.4.4, "Disabling Region Selection" . Added Section 8.5.3, "Viewing Application Details" . BZ 1118766 and BZ 1146728: Updated Section 2.4, "Enabling Support for High-Availability Applications" to fix wording and procedure order and to add detail.		
Revision 2.1-8	Thu Oct 2 2014	Brice Fallon-Freeman

OpenShift Enterprise 2.1.7 release.

Updated `oo-admin-ctl-cartridge` commands to use `import-profile`.

BZ 1140732: Updated [Section 2.6, "Controlling Direct SSL Connections to Gears"](#) to fix a typo.

BZ 1134034: Updated [Section 8.8.3, "Enabling Syslog for Cartridge Logs from Gears"](#) to include the `multi` option for the `outputType` setting.

Revision 2.1-7	Thu Sep 11 2014	Alex Dellapenta
<p>OpenShift Enterprise 2.1.6 release.</p> <p>BZ 1133936: Updated Section 6.1, "Adding or Modifying Gear Profiles" to note new, additional example <code>resource_limits.conf</code> files.</p>		
Revision 2.1-6	Tue Aug 26 2014	Alex Dellapenta
<p>OpenShift Enterprise 2.1.5 release.</p> <p>Added Section 1.3, "Migrating from RHN Classic to RHSM".</p> <p>BZ 1083380: Added Section 2.2, "Enabling User Login Normalization".</p> <p>BZ 1133493: Updated Section 6.4.3, "Setting the Default Region For New Applications" with correct file location.</p> <p>BZ 1123949: Updated example in Section 9.1.3, "oo-admin-chk" with missing a value for the <code>-l</code> option.</p>		
Revision 2.1-5	Fri Aug 8 2014	Alex Dellapenta
<p>Fixed minor publication issue.</p>		
Revision 2.1-4	Tue Aug 5 2014	Brice Fallon-Freeman
<p>OpenShift Enterprise 2.1.4 release.</p> <p>Added Section 6.4.3, "Setting the Default Region For New Applications".</p> <p>Updated Section 8.5.4.1, "Enabling Watchman" and Section 8.5.4.2, "Supported Watchman Plug-ins" with new options.</p>		
Revision 2.1-3	Wed Jul 9 2014	Julie Wu
<p>BZ 1101768: Updated Chapter 5, Cartridge Management with more detailed information.</p> <p>BZ 1112822: Updated the second procedure in Section 8.8.3, "Enabling Syslog for Cartridge Logs from Gears".</p> <p>BZ 1116293: Updated Section 4.2, "Creating Global Teams and Synchronizing with LDAP Groups" with example file configurations.</p> <p>BZ 1084617: Updated Section 2.1, "Changing the Front-end HTTP Configuration for Existing Deployments" with correct Apache plug-in information.</p>		
Revision 2.1-2	Thu Jun 26 2014	Julie Wu
<p>Updated the guide to call out 2.1 features.</p> <p>BZ 1097764: Added Section 5.4, "Adding QuickStarts to the Management Console".</p> <p>BZ 1100883: Updated Section 3.1, "Creating a User".</p> <p>BZ 1110547: Updated Section 5.2, "Installing and Removing Custom and Community Cartridges".</p>		
Revision 2.1-1	Thu Jun 5 2014	Julie Wu
<p>OpenShift Enterprise 2.1.1 release.</p> <p>BZ 1104412: Updated Section 9.1.1, "oo-accept-broker".</p> <p>BZ 1100877: Updated Section 3.1, "Creating a User".</p> <p>BZ 1025747: Updated Section 6.6, "Setting Default Gear Quotas and Sizes".</p> <p>BZ 1090096: Updated Section 6.6, "Setting Default Gear Quotas and Sizes".</p>		
Revision 2.1-0	Fri May 16 2014	Brice Fallon-Freeman

OpenShift Enterprise 2.1 release.

BZ 1063859: Updated log file locations to `/var/log/openshift/node/ruby193-mcollective.log`

BZ 1027627: Added [Section 2.11, "Component Timeout Value Locations"](#).

BZ 1045239: Updated oo-auto-idler commands.

Added [Section 4.2.2, "Enabling Global Team Visibility"](#) and [Section 4.2, "Creating Global Teams and Synchronizing with LDAP Groups"](#).

Updated [Section 6.3.2, "Creating and Populating Districts"](#) and [Section 6.4.2, "Tagging a Node with a Region and Zone"](#) with information on how to simultaneously create a district and add any number of nodes to it.

Added [Section 2.3, "Allowing Multiple HAProxies on a Node Host"](#).

Added [Section 2.5, "Creating Environment Variables on Node Hosts"](#).

Added [Section 2.9, "Enabling Maintenance Mode"](#).

Added [Section 2.10.2, "Backing Up Node Host Files"](#).

BZ 1060815: Added [Section 5.1, "Managing Cartridges on Broker Hosts"](#).

Added information on activating cartridges.

Added [Section 6.4, "Managing Regions and Zones"](#) and subsections.

Added "Customizing the Gear Placement Algorithm".

Added [Section 8.8.2, "Enabling Syslog for Node Components"](#), [Section 8.5.2, "Enabling Application and Gear Context in Node Component Logs"](#), and [Section 8.8.3, "Enabling Syslog for Cartridge Logs from Gears"](#).

Added [Section 8.5.4.1, "Enabling Watchman"](#) and subsections.

Added [Section 8.6, "Monitoring Management Console Activity"](#) and subsections.

Added [Section 9.1.12, "oo-admin-ctl-region"](#).

Updated [Section 1.2, "Upgrading OpenShift Enterprise"](#).

Updated [Section 6.5, "Gear Placement Algorithm"](#).

Updated [Section 8.4.1, "Default Broker Log File Locations"](#) and [Section 8.5.1, "Default Node Log File Locations"](#).

Updated [Section 9.1.18, "oo-admin-upgrade"](#).

Updated various sections to note that districts are now required by default.

[BZ 1071443](#): Updated [Section 2.10.3, "Recovering Failed Node Hosts"](#) and added [Section 2.10.4, "Recreating /etc/passwd Entries"](#).

Added [Section 4.1, "Setting the Maximum Number of Teams for Specific Users"](#).

Revision 2.0-2	Tue Jan 28 2014	Julie Wu
----------------	-----------------	----------

OpenShift Enterprise 2.0.2 release.

Updated [Section 9.1.15, "oo-admin-ctl-user"](#) with more options.

Revision 2.0-1	Tue Jan 14 2014	Brice Fallon-Freeman
----------------	-----------------	----------------------

OpenShift Enterprise 2.0.1 release.

[BZ 1026990](#): Added [Section 3.6, "Setting Default Maximum Number of Domains per User"](#).

Revision 2.0-0	Mon Dec 9 2013	Alex Dellapenta
----------------	----------------	-----------------

OpenShift Enterprise 2.0 release.

Added "Disabling Obsolete Cartridges" section

Added "Adding a Kerberos Principal SSH Key" section.

Added "Disabling Downloadable Cartridges" section.

Added "Administration Console" sections.

Added "Capacity Planning and Districts" sections.

Added "Adding New Gear Profiles" section.

Update oo-admin-repair content.

Added "Enforcing Low Tenancy on Nodes" section.

[BZ 988576](#): Added "Controlling Direct SSL Connections to Gears" section.

[BZ 994783](#): Added more details to "Viewing Accumulated Usage Data" section.