OpenShift Enterprise 3.1

Installation and Configuration

OpenShift Enterprise 3.1 Installation and Configuration
Abstract

OpenShift Installation and Configuration topics cover the basics of installing and configuring OpenShift in your environment. Use these topics for the one-time tasks required to get OpenShift up and running.
Table of Contents

CHAPTER 1. OVERVIEW ................................................................. 12

CHAPTER 2. INSTALLING .............................................................. 13
  2.1. OVERVIEW ...................................................................... 13
  2.2. PREREQUISITES .............................................................. 13
    2.2.1. Overview ................................................................ 13
    2.2.2. System Requirements .............................................. 13
      2.2.2.1. Host Recommendations ...................................... 14
      2.2.2.2. Configuring Core Usage ...................................... 15
      2.2.2.3. Security Warning .............................................. 15
    2.2.3. Environment Requirements ........................................ 15
      2.2.3.1. DNS .................................................................. 15
      2.2.3.2. Network Access ............................................... 16
      2.2.3.3. Git Access ......................................................... 18
      2.2.3.4. Persistent Storage .............................................. 19
      2.2.3.5. SELinux ............................................................ 19
      2.2.3.6. Cloud Provider Considerations ............................ 19
    2.2.4. Host Preparation ...................................................... 22
      2.2.4.1. Software Prerequisites ........................................ 22
      2.2.4.2. Configuring Docker Storage ................................. 25
    2.2.5. Ensuring Host Access ................................................ 29
    2.2.6. Setting Global Proxy Values ...................................... 29
    2.2.7. What’s Next? ............................................................ 30
  2.3. RPM VS CONTAINERIZED .................................................... 30
    2.3.1. Overview .................................................................. 30
    2.3.2. Required Images ...................................................... 31
    2.3.3. CLI Wrappers .......................................................... 31
    2.3.4. Starting and Stopping Containers ............................... 32
    2.3.5. File Paths ............................................................... 32
    2.3.6. Storage Requirements .............................................. 32
    2.3.7. Open vSwitch SDN Initialization ................................. 33
  2.4. QUICK INSTALLATION .......................................................... 33
    2.4.1. Overview .................................................................. 33
    2.4.2. Before You Begin ...................................................... 33
    2.4.3. Running an Interactive Installation ............................. 34
    2.4.4. Defining an Installation Configuration File ..................... 34
    2.4.5. Running an Unattended Installation .............................. 36
    2.4.6. Verifying the Installation .......................................... 36
    2.4.7. Adding Nodes or Reinstalling the Cluster ..................... 37
    2.4.8. Uninstalling OpenShift Enterprise .............................. 38
    2.4.9. What’s Next? ............................................................ 38
  2.5. ADVANCED INSTALLATION .................................................... 38
    2.5.1. Overview .................................................................. 38
    2.5.2. Before You Begin ...................................................... 38
    2.5.3. Configuring Ansible .................................................. 39
      2.5.3.1. Configuring Host Variables ................................. 39
      2.5.3.2. Configuring Cluster Variables .............................. 40
      2.5.3.3. Configuring Node Host Labels ............................... 43
      2.5.3.4. Marking Masters as Unschedulable Nodes ............... 43
      2.5.3.5. Configuring Session Options ................................. 44
      2.5.3.6. Configuring Custom Certificates ............................ 44
2.8. DEPLOYING A ROUTER

2.8.1. Overview
2.8.2. The Router Service Account
2.8.3. Deploying the Default HAProxy Router
  2.8.3.1. High Availability
  2.8.3.2. Customizing the Default Routing Subdomain
  2.8.3.3. Using Wildcard Certificates
  2.8.3.4. Using Secured Routes
  2.8.3.5. Using the Container Network Stack
  2.8.3.6. Exposing Router metrics
2.8.4. Deploying a Customized HAProxy Router
  2.8.4.1. Using Stick Tables
  2.8.4.2. Rebuilding Your Router
2.8.5. Deploying the F5 Router
2.8.6. What’s Next?

CHAPTER 3. UPGRADING

3.1. OVERVIEW
3.2. PERFORMING AUTOMATED CLUSTER UPGRADES
  3.2.1. Overview
  3.2.2. Preparing for an Automated Upgrade
  3.2.3. Using the Installer to Upgrade
  3.2.4. Running the Upgrade Playbook Directly
    3.2.4.1. Upgrading to OpenShift Enterprise 3.1.0
    3.2.4.2. Upgrading to OpenShift Enterprise 3.1 Asynchronous Releases
  3.2.5. Updating Master and Node Certificates
    3.2.5.1. Node Certificates
      3.2.5.1.1. Checking the Node’s Certificate
      3.2.5.1.2. Generating a New Node Certificate
      3.2.5.1.3. Replace Node Serving Certificates
    3.2.5.2. Master Certificates
      3.2.5.2.1. Checking the Master’s Certificate
      3.2.5.2.2. Generating a New Master Certificate
  3.2.6. Upgrading the EFK Logging Stack
  3.2.7. Verifying the Upgrade
3.3. PERFORMING MANUAL CLUSTER UPGRADES
  3.3.1. Overview
  3.3.2. Preparing for a Manual Upgrade
  3.3.3. Upgrading Masters
  3.3.4. Updating Policy Definitions
  3.3.5. Upgrading Nodes
  3.3.6. Upgrading the Router
  3.3.7. Upgrading the Registry
  3.3.8. Updating the Default Image Streams and Templates
  3.3.9. Importing the Latest Images
  3.3.10. Updating Master and Node Certificates
    3.3.10.1. Node Certificates
      3.3.10.1.1. Checking the Node’s Certificate
      3.3.10.1.2. Generating a New Node Certificate
      3.3.10.1.3. Replace Node Serving Certificates
    3.3.10.2. Master Certificates
      3.3.10.2.1. Checking the Master’s Certificate
      3.3.10.2.2. Generating a New Master Certificate
3.3.11. Upgrading the EFK Logging Stack
3.3.12. Additional Manual Steps Per Release
  3.3.12.1. OpenShift Enterprise 3.1.0
  3.3.12.2. OpenShift Enterprise 3.1.1
  3.3.12.3. OpenShift Enterprise 3.1.1.11
3.3.13. Verifying the Upgrade
3.4. UPGRADING FROM PACEMAKER TO NATIVE HA
  3.4.1. Overview
  3.4.2. Using Ansible Playbooks
    3.4.2.1. Modifying the Ansible Inventory
    3.4.2.1.1. Destroying the Pacemaker Cluster
    3.4.2.2. Updating DNS
    3.4.2.3. Running the Ansible Playbook
  3.4.3. Manually Upgrading
    3.4.3.1. Creating Unit and System Configuration for New Services
    3.4.3.2. Destroying the Pacemaker Cluster
    3.4.3.3. Updating DNS
    3.4.3.4. Modifying Master and Node Configuration
      3.4.3.4.1. Starting the API Service
      3.4.3.4.2. Starting the Controller Service
    3.4.3.5. Modifying the Ansible Inventory

CHAPTER 4. DOWNGRADING OPENSIFHT .................................................. 124.
  4.1. OVERVIEW .............................................................................. 124
  4.2. VERIFYING BACKUPS .............................................................. 124
  4.3. SHUTTING DOWN THE CLUSTER .............................................. 124
  4.4. REMOVING RPMS ................................................................. 124
  4.5. REINSTALLING RPMS ............................................................... 125
  4.6. RESTORING ETCD ................................................................. 125
    4.6.1. Embedded etcd ............................................................... 125
    4.6.2. External etcd ................................................................. 126
      4.6.2.1. Adding Additional etcd Members .................................. 128
    4.7. BRINGING OPENSIFHT SERVICES BACK ONLINE .................. 129

CHAPTER 5. MASTER AND NODE CONFIGURATION .................................... 131.
  5.1. OVERVIEW .............................................................................. 131
  5.2. CREATING NEW CONFIGURATION FILES .................................. 131
  5.3. LAUNCHING SERVERS USING CONFIGURATION FILES ............... 131
  5.4. MASTER CONFIGURATION FILES .......................................... 132
  5.5. NODE CONFIGURATION FILES ............................................... 134

  6.1. OVERVIEW .............................................................................. 136
  6.2. PREREQUISITES ...................................................................... 137
  6.3. CREATING IMAGE STREAMS FOR OPENSIFHT IMAGES ............... 137
  6.4. CREATING IMAGE STREAMS FOR XPAAS MIDDLEWARE IMAGES .... 138
  6.5. CREATING DATABASE SERVICE TEMPLATES ............................. 138
  6.6. CREATING INSTANT APP AND QUICKSTART TEMPLATES ............ 138
  6.7. WHAT’S NEXT? ....................................................................... 139

CHAPTER 7. CONFIGURING CUSTOM CERTIFICATES .................................... 140.
  7.1. OVERVIEW .............................................................................. 140
  7.2. CONFIGURING CUSTOM CERTIFICATES .................................... 140
10.4.4. Configuring Apache
10.4.5. Configuring OpenShift Enterprise
10.4.6. Debugging Notes

CHAPTER 11. CONFIGURING THE SDN ............................................. 185.
11.1. OVERVIEW ........................................................................ 185
11.2. CONFIGURING THE POD NETWORK ON MASTERS .......... 185
11.3. CONFIGURING THE POD NETWORK ON NODES ............ 185
11.4. MIGRATING BETWEEN SDN PLUG-INS ......................... 186
11.5. EXTERNAL ACCESS TO THE CLUSTER NETWORK ........ 186

CHAPTER 12. CONFIGURING FOR AWS ....................................... 187.
12.1. OVERVIEW ........................................................................ 187
12.2. CONFIGURING AWS VARIABLES .................................... 187
12.3. CONFIGURING MASTERS .................................................. 187
12.4. CONFIGURING NODES ...................................................... 187
12.5. SETTING KEY VALUE ACCESS PAIRS ......................... 187
12.6. APPLYING CONFIGURATION CHANGES ....................... 188

CHAPTER 13. CONFIGURING FOR OPENSTACK ......................... 189.
13.1. OVERVIEW ........................................................................ 189
13.2. CONFIGURING OPENSTACK VARIABLES ..................... 189
13.3. CONFIGURING MASTERS .................................................. 189
13.4. CONFIGURING NODES ...................................................... 189

CHAPTER 14. CONFIGURING FOR GCE ........................................ 191.
14.1. OVERVIEW ........................................................................ 191
14.2. CONFIGURING MASTERS .................................................. 191
14.3. CONFIGURING NODES ...................................................... 191

CHAPTER 15. CONFIGURING PERSISTENT STORAGE .................. 192.
15.1. OVERVIEW ........................................................................ 192
15.2. PERSISTENT STORAGE USING NFS ......................... 192
  15.2.1. Overview ................................................................. 192
  15.2.2. Provisioning ............................................................ 192
  15.2.3. Enforcing Disk Quotas ........................................... 194
  15.2.4. NFS Volume Security .............................................. 194
    15.2.4.1. Group IDs ....................................................... 195
    15.2.4.2. User IDs ......................................................... 195
    15.2.4.3. SELinux .......................................................... 196
    15.2.4.4. Export Settings ............................................... 197
  15.2.5. Reclaiming Resources ............................................. 197
  15.2.6. Automation ............................................................. 198
  15.2.7. Additional Configuration and Troubleshooting ......... 198
15.3. PERSISTENT STORAGE USING GLUSTERFS .............. 199
  15.3.1. Overview ................................................................. 199
  15.3.2. Provisioning ............................................................ 199
    15.3.2.1. Creating Gluster Endpoints ............................... 200
    15.3.2.2. Creating the Persistent Volume ......................... 201
    15.3.2.3. Creating the Persistent Volume Claim ............... 202
  15.3.3. Gluster Volume Security .......................................... 203
    15.3.3.1. Group IDs ....................................................... 203
    15.3.3.2. User IDs ......................................................... 204
    15.3.3.3. SELinux .......................................................... 205
15.4. PERSISTENT STORAGE USING OPENSTACK CINDER
  15.4.1. Overview 205
  15.4.2. Provisioning 205
    15.4.2.1. Creating the Persistent Volume 206
    15.4.2.2. Volume Format 207
15.5. PERSISTENT STORAGE USING CEPH RADOS BLOCK DEVICE (RBD)
  15.5.1. Overview 207
  15.5.2. Provisioning 207
    15.5.2.1. Creating the Ceph Secret 208
    15.5.2.2. Creating the Persistent Volume 208
  15.5.3. Ceph Volume Security 210
15.6. PERSISTENT STORAGE USING AWS ELASTIC BLOCK STORE
  15.6.1. Overview 211
  15.6.2. Provisioning 211
    15.6.2.1. Creating the Persistent Volume 211
    15.6.2.2. Volume Format 212
15.7. PERSISTENT STORAGE USING GCE PERSISTENT DISK
  15.7.1. Overview 213
  15.7.2. Provisioning 213
    15.7.2.1. Creating the Persistent Volume 213
    15.7.2.2. Volume Format 214
15.8. PERSISTENT STORAGE USING iSCSI
  15.8.1. Overview 214
  15.8.2. Provisioning 215
    15.8.2.1. Enforcing Disk Quotas 215
    15.8.2.2. iSCSI Volume Security 215
15.9. PERSISTENT STORAGE USING FIBRE CHANNEL
  15.9.1. Overview 215
  15.9.2. Provisioning 216
    15.9.2.1. Enforcing Disk Quotas 217
    15.9.2.2. Fibre Channel Volume Security 217
15.10. DYNAMICALLY PROVISIONING PERSISTENT VOLUMES
  15.10.1. Overview 217
  15.10.2. Enabling Provisioner Plug-ins 217
  15.10.3. Requesting Dynamically Provisioned Storage 218
  15.10.4. Volume Recycling 219
15.11. VOLUME SECURITY
  15.11.1. Overview 219
  15.11.2. SCCs, Defaults, and Allowed Ranges 220
  15.11.3. Supplemental Groups 223
  15.11.4. fsGroup 226
  15.11.5. User IDs 228
  15.11.6. SELinux Options 230

CHAPTER 16. PERSISTENT STORAGE EXAMPLES .................................................. 232
16.1. OVERVIEW 232
16.2. SHARING AN NFS PERSISTENT VOLUME (PV) ACROSS TWO PODS
  16.2.1. Overview 232
  16.2.2. Creating the Persistent Volume 232
  16.2.3. Creating the Persistent Volume Claim 233
  16.2.4. Ensuring NFS Volume Access 234
  16.2.5. Creating the Pod 235
  16.2.6. Creating an Additional Pod to Reference the Same PVC 239
16.3. COMPLETE EXAMPLE USING CEPH RBD
  16.3.1. Overview
  16.3.2. Installing the ceph-common Package
  16.3.3. Creating the Ceph Secret
  16.3.4. Creating the Persistent Volume
  16.3.5. Creating the Persistent Volume Claim
  16.3.6. Creating the Pod
  16.3.7. Defining Group and Owner IDs (Optional)

16.4. COMPLETE EXAMPLE USING GLUSTERFS
  16.4.1. Overview
  16.4.2. Installing the glusterfs-fuse Package
  16.4.3. Creating the Gluster Endpoints
  16.4.4. Creating the Persistent Volume
  16.4.5. Creating the Persistent Volume Claim
  16.4.6. Defining GlusterFS Volume Access
  16.4.7. Creating the Pod

16.5. BACKING DOCKER REGISTRY WITH GLUSTERFS STORAGE
  16.5.1. Overview
  16.5.2. Prerequisites
  16.5.3. Create the Gluster Persistent Volume
  16.5.4. Attach the PVC to the Docker Registry
  16.5.5. Known Issues
    16.5.5.1. Pod Cannot Resolve the Volume Host

16.6. MOUNTING VOLUMES ON PRIVILEGED PODS
  16.6.1. Overview
  16.6.2. Prerequisites
  16.6.3. Creating the Persistent Volume
  16.6.4. Creating a Regular User
  16.6.5. Creating the Persistent Volume Claim
  16.6.6. Verifying the Setup
    16.6.6.1. Checking the Pod SCC
    16.6.6.2. Verifying the Mount

CHAPTER 17. WORKING WITH HTTP PROXIES
  17.1. OVERVIEW
  17.2. CONFIGURING HOSTS FOR PROXIES
  17.3. PROXYING DOCKER PULL
  17.4. USING MAVEN BEHIND A PROXY
  17.5. CONFIGURING S2I BUILDS FOR PROXIES
  17.6. CONFIGURING DEFAULT TEMPLATES FOR PROXIES
  17.7. SETTING PROXY ENVIRONMENT VARIABLES IN PODS
  17.8. GIT REPOSITORY ACCESS

CHAPTER 18. NATIVE CONTAINER ROUTING
  18.1. OVERVIEW
  18.2. NETWORK LAYOUT
  18.3. NETWORK OVERVIEW
  18.4. NODE SETUP
  18.5. ROUTER SETUP

CHAPTER 19. ROUTING FROM EDGE LOAD BALANCERS
  19.1. OVERVIEW
  19.2. INCLUDING THE LOAD BALANCER IN THE SDN
  19.3. ESTABLISHING A TUNNEL USING A RAMP NODE
CHAPTER 1. OVERVIEW

OpenShift Enterprise Installation and Configuration topics cover the basics of installing and configuring OpenShift Enterprise in your environment. Configuration, management, and logging are also covered. Use these topics for the one-time tasks required quickly set up your OpenShift Enterprise environment and configure it based on your organizational needs.

For day to day cluster administrator tasks, see Cluster Administration.
CHAPTER 2. INSTALLING

2.1. OVERVIEW

The quick installation method allows you to use an interactive CLI utility to install OpenShift across a set of hosts. This installer is a self-contained wrapper intended for usage on a Red Hat Enterprise Linux 7 host.

For production environments, a reference configuration implemented using Ansible playbooks is available as the advanced installation method.

NOTE
Before beginning either installation method, start with the Prerequisites topic.

2.2. PREREQUISITES

2.2.1. Overview

OpenShift infrastructure components can be installed across multiple hosts. The following sections outline the system requirements and instructions for preparing your environment and hosts before installing OpenShift.

2.2.2. System Requirements

You must have an active OpenShift Enterprise subscription on your Red Hat account to proceed. If you do not, contact your sales representative for more information.

IMPORTANT
OpenShift Enterprise (OSE) 3.x supports Red Hat Enterprise Linux (RHEL) 7.1 or later. Starting in OSE 3.1.1, RHEL Atomic Host 7.1.6 or later is also supported, as it requires the containerized method introduced in OSE 3.1.1.

IMPORTANT
OSE 3.1.x requires Docker 1.8.2, however Docker 1.9 is currently not supported due to performance issues. See the Red Hat Knowledgebase for details, including steps on how to configure yum so that later Docker versions are not installed via yum update. Otherwise, running yum update after installing OpenShift Enterprise will upgrade Docker and put your cluster in an unsupported configuration. Follow this topic to ensure you have the correct version of Docker installed on your hosts before installing or upgrading to OSE 3.1.

The system requirements vary per host type:
## 2.2.2.1. Host Recommendations

The following apply to production environments. Test or sample environments will function with the minimum requirements.

### Master Hosts

In a highly-available OpenShift Enterprise cluster with external etcd, a master host should have 1 CPU core and 2.5 GB of memory, on top of the defaults in the table above, for each 1000 pods. Therefore, the recommended size of master host in an OpenShift Enterprise cluster of 2000 pods would be 2 CPU cores and 5 GB of RAM, as well as the minimum requirements for a master host of 2 CPU cores and 8 GB of RAM.

When planning an environment with multiple masters, a minimum of three etcd hosts as well as a load-balancer between the master hosts, is required.

### Node Hosts

The size of a node host depends on the expected size of its workload. As an OpenShift Enterprise cluster administrator, you will need to calculate the expected workload, then add about 10% for...
overhead. For production environments, allocate enough resources so that node host failure does not affect your maximum capacity.

Use the above with the following table to plan the maximum loads for nodes and pods:

<table>
<thead>
<tr>
<th>Host</th>
<th>Sizing Recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum nodes per cluster</td>
<td>300</td>
</tr>
<tr>
<td>Maximum pods per nodes</td>
<td>110</td>
</tr>
</tbody>
</table>

**IMPORTANT**

Oversubscribing the physical resources on a node affects resource guarantees the Kubernetes scheduler makes during pod placement. Learn what measures you can take to avoid memory swapping.

### 2.2.2.2. Configuring Core Usage

By default, OpenShift masters and nodes use all available cores in the system they run on. You can choose the number of cores you want OpenShift to use by setting the `GOMAXPROCS` environment variable.

For example, run the following before starting the server to make OpenShift only run on one core:

```
# export GOMAXPROCS=1
```

### 2.2.2.3. Security Warning

OpenShift runs Docker containers on your hosts, and in some cases, such as build operations and the registry service, it does so using privileged containers. Furthermore, those containers access your host’s Docker daemon and perform `docker build` and `docker push` operations. As such, you should be aware of the inherent security risks associated with performing `docker run` operations on arbitrary images as they effectively have root access.

For more information, see these articles:

- [http://opensource.com/business/14/7/docker-security-selinux](http://opensource.com/business/14/7/docker-security-selinux)
- [https://docs.docker.com/articles/security/](https://docs.docker.com/articles/security/)

To address these risks, OpenShift uses security context constraints that control the actions that pods can perform and what it has the ability to access.

### 2.2.3. Environment Requirements

The following must be set up in your environment before OpenShift can be installed.

#### 2.2.3.1. DNS

A wildcard for a DNS zone must ultimately resolve to the IP address of the OpenShift router.
For example, create a wildcard DNS entry for \texttt{cloudapps}, or something similar, that has a low TTL and points to the public IP address of the host where the router will be deployed:

\begin{verbatim}
*.cloudapps.example.com. 300 IN A 192.168.133.2
\end{verbatim}

In almost all cases, when referencing VMs you must use host names, and the host names that you use must match the output of the \texttt{hostname -f} command on each node.

\begin{quote}
\textbf{WARNING}

In your \texttt{/etc/resolv.conf} file on each node host, ensure that the DNS server that has the wildcard entry is not listed as a nameserver or that the wildcard domain is not listed in the search list. Otherwise, containers managed by OpenShift may fail to resolve host names properly.
\end{quote}

\subsection*{2.2.3.2. Network Access}

A shared network must exist between the master and node hosts. If you plan to configure \textit{multiple masters for high-availability} using the \textit{advanced installation method}, you must also select an IP to be configured as your \textit{virtual IP} (VIP) during the installation process. The IP that you select must be routable between all of your nodes, and if you configure using a FQDN it should resolve on all nodes.

\textbf{Required Ports}

OpenShift infrastructure components communicate with each other using ports, which are communication endpoints that are identifiable for specific processes or services. Ensure the following ports required by OpenShift are open between hosts, for example if you have a firewall in your environment. Some ports are optional depending on your configuration and usage.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|l|}
\hline
\textbf{Table 2.1. Node to Node} & & \\
\hline
4789 & UDP & Required for SDN communication between pods on separate hosts. \\
\hline
\end{tabular}
\end{table}

\begin{table}[h]
\centering
\begin{tabular}{|c|c|l|}
\hline
\textbf{Table 2.2. Nodes to Master} & & \\
\hline
53 or 8053 & TCP/UDP & Required for DNS resolution of cluster services (SkyDNS). Installations prior to 3.2 or environments upgraded to 3.2 use port 53. New installations will use 8053 by default so that \texttt{dnsmasq} may be configured. \\
4789 & UDP & Required for SDN communication between pods on separate hosts. \\
443 or 8443 & TCP & Required for node hosts to communicate to the master API, for the node hosts to post back status, to receive tasks, and so on. \\
\hline
\end{tabular}
\end{table}

\begin{table}[h]
\centering
\begin{tabular}{|c|c|l|}
\hline
\textbf{Table 2.3. Master to Node} & & \\
\hline
4789 & UDP & Required for SDN communication between pods on separate hosts. \\
\hline
\end{tabular}
\end{table}
The master proxies to node hosts via the Kubelet for `oc` commands.

**NOTE**

In the following table, (L) indicates the marked port is also used in *loopback mode*, enabling the master to communicate with itself.

In a single-master cluster:

- Ports marked with (L) must be open.
- Ports not marked with (L) need not be open.

In a multiple-master cluster, all the listed ports must be open.

### Table 2.4. Master to Master

<table>
<thead>
<tr>
<th>Port</th>
<th>Protocol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>53 (L) or 8053 (L)</td>
<td>TCP/UDP</td>
<td>Required for DNS resolution of cluster services (SkyDNS). Installations prior to 3.2 or environments upgraded to 3.2 use port 53. New installations of 3.2 or later use 8053 by default so that <code>dnsmasq</code> may be configured.</td>
</tr>
<tr>
<td>2049 (L)</td>
<td>TCP/UDP</td>
<td>Required when provisioning an NFS host as part of the installer.</td>
</tr>
<tr>
<td>2379</td>
<td>TCP</td>
<td>Used for standalone etcd (clustered) to accept changes in state.</td>
</tr>
<tr>
<td>2380</td>
<td>TCP</td>
<td>etcd requires this port be open between masters for leader election and peering connections when using standalone etcd (clustered).</td>
</tr>
<tr>
<td>4001 (L)</td>
<td>TCP</td>
<td>Used for embedded etcd (non-clustered) to accept changes in state.</td>
</tr>
<tr>
<td>4789 (L)</td>
<td>UDP</td>
<td>Required for SDN communication between pods on separate hosts.</td>
</tr>
</tbody>
</table>

### Table 2.5. External to Load Balancer

<table>
<thead>
<tr>
<th>Port</th>
<th>Protocol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9000</td>
<td>TCP</td>
<td>If you choose the native HA method, optional to allow access to the HAProxy statistics page.</td>
</tr>
</tbody>
</table>

### Table 2.6. External to Master

<table>
<thead>
<tr>
<th>Port</th>
<th>Protocol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>443 or 8443</td>
<td>TCP</td>
<td>Required for node hosts to communicate to the master API, for node hosts to post back status, to receive tasks, and so on.</td>
</tr>
</tbody>
</table>

### Table 2.7. IaaS Deployments

<table>
<thead>
<tr>
<th>Port</th>
<th>Protocol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>TCP</td>
<td>Required for SSH by the installer or system administrator.</td>
</tr>
<tr>
<td>Port(s)</td>
<td>Protocol</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>53 or 8053</td>
<td>TCP/UDP</td>
<td>Required for DNS resolution of cluster services (SkyDNS). Installations prior to 3.2 or environments upgraded to 3.2 use port 53. New installations will use 8053 by default so that <code>dnsmasq</code> may be configured. Only required to be internally open on master hosts.</td>
</tr>
<tr>
<td>80 or 443</td>
<td>TCP</td>
<td>For HTTP/HTTPS use for the router. Required to be externally open on node hosts, especially on nodes running the router.</td>
</tr>
<tr>
<td>1936</td>
<td>TCP</td>
<td>For router statistics use. Required to be open when running the template router to access statistics, and can be open externally or internally to connections depending on if you want the statistics to be expressed publicly.</td>
</tr>
<tr>
<td>4001</td>
<td>TCP</td>
<td>For embedded etcd (non-clustered) use. Only required to be internally open on the master host. 4001 is for server-client connections.</td>
</tr>
<tr>
<td>2379 and 2380</td>
<td>TCP</td>
<td>For standalone etcd use. Only required to be internally open on the master host. 2379 is for server-client connections. 2380 is for server-server connections, and is only required if you have clustered etcd.</td>
</tr>
<tr>
<td>4789</td>
<td>UDP</td>
<td>For VxLAN use (OpenShift Enterprise SDN). Required only internally on node hosts.</td>
</tr>
<tr>
<td>8443</td>
<td>TCP</td>
<td>For use by the OpenShift Enterprise web console, shared with the API server.</td>
</tr>
<tr>
<td>10250</td>
<td>TCP</td>
<td>For use by the Kubelet. Required to be externally open on nodes.</td>
</tr>
<tr>
<td>24224</td>
<td>TCP/UDP</td>
<td>For use by Fluentd. Required to be open on master hosts for internal connections to node hosts.</td>
</tr>
</tbody>
</table>

**Notes**

- In the above examples, port **4789** is used for User Datagram Protocol (UDP).
- When deployments are using the SDN, the pod network is accessed via a service proxy, unless it is accessing the registry from the same node the registry is deployed on.
- OpenShift internal DNS cannot be received over SDN. Depending on the detected values of `openshift_facts`, or if the `openshift_ip` and `openshift_public_ip` values are overridden, it will be the computed value of `openshift_ip`. For non-cloud deployments, this will default to the IP address associated with the default route on the master host. For cloud deployments, it will default to the IP address associated with the first internal interface as defined by the cloud metadata.
- The master host uses port **10250** to reach the nodes and does not go over SDN. It depends on the target host of the deployment and uses the computed values of `openshift_hostname` and `openshift_public_hostname`.

**2.2.3.3. Git Access**

You must have either Internet access and a GitHub account, or read and write access to an internal, HTTP-based Git server.
### 2.2.3.4. Persistent Storage

The Kubernetes persistent volume framework allows you to provision an OpenShift cluster with persistent storage using networked storage available in your environment. This can be done after completing the initial OpenShift installation depending on your application needs, giving users a way to request those resources without having any knowledge of the underlying infrastructure.

The Installation and Configuration Guide provides instructions for cluster administrators on provisioning an OpenShift cluster with persistent storage using NFS, GlusterFS, Ceph RBD, OpenStack Cinder, AWS Elastic Block Store (EBS), GCE Persistent Disks, and iSCSI.

### 2.2.3.5. SELinux

Security-Enhanced Linux (SELinux) must be enabled on all of the servers before installing OpenShift or the installer will fail. Also, configure `SELINUXTYPE=targeted` in the `/etc/selinux/config` file:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of these three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

### 2.2.3.6. Cloud Provider Considerations

#### Set up the Security Group

When installing on AWS or OpenStack, ensure that you set up the appropriate security groups. These are some ports that you should have in your security groups, without which the installation will fail. You may need more depending on the cluster configuration you want to install. For more information and to adjust your security groups accordingly, see Required Ports for more information.

<table>
<thead>
<tr>
<th>All OpenShift Hosts</th>
<th>tcp/22 from host running the installer/Ansible</th>
</tr>
</thead>
<tbody>
<tr>
<td>etcd Security Group</td>
<td>◆ tcp/2379 from masters</td>
</tr>
<tr>
<td></td>
<td>◆ tcp/2380 from etcd hosts</td>
</tr>
<tr>
<td>Master Security Group</td>
<td>◆ tcp/8443 from 0.0.0.0/0</td>
</tr>
<tr>
<td></td>
<td>◆ tcp/53 from all OpenShift hosts</td>
</tr>
<tr>
<td></td>
<td>◆ udp/53 from all OpenShift hosts</td>
</tr>
</tbody>
</table>
Node Security Group

- tcp/10250 from masters
- tcp/4789 from nodes

Infrastructure Nodes (ones that can host the `openshift-router`)

- tcp/443 from 0.0.0.0/0
- tcp/80 from 0.0.0.0/0

If configuring ELBs for load balancing the masters and/or routers, you also need to configure Ingress and Egress security groups for the ELBs appropriately.

**Override Detected IP Addresses and Host Names**

Some deployments require that the user override the detected host names and IP addresses for the hosts. To see the default values, run the `openshift_facts` playbook:

```
# ansible-playbook playbooks/byo/openshift_facts.yml
```

Now, verify the detected common settings. If they are not what you expect them to be, you can override them.

The [Advanced Installation](#) topic discusses the available Ansible variables in greater detail.

<table>
<thead>
<tr>
<th>hostname</th>
<th>Should resolve to the internal IP from the instances themselves.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>* <code>openshift_hostname</code> overrides.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ip</th>
<th>Should be the internal IP of the instance.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>* <code>openshift_ip</code> will overrides.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public_hostname</th>
<th>Should resolve to the external IP from hosts outside of the cloud.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>* Provider <code>openshift_public_hostname</code> overrides.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public_ip</th>
<th>Should be the externally accessible IP associated with the instance.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>* <code>openshift_public_ip</code> overrides.</td>
</tr>
</tbody>
</table>
### use_openshift_sdn

- Should be true unless the cloud is GCE.
- `openshift_use_openshift_sdn` overrides.

#### WARNING

If `openshift_hostname` is set to a value other than the metadata-provided `private-dns-name` value, the native cloud integration for those providers will no longer work.

In AWS, situations that require overriding the variables include:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>hostname</code></td>
<td>The user is installing in a VPC that is not configured for both DNS hostnames and DNS resolution.</td>
</tr>
<tr>
<td><code>ip</code></td>
<td>Possibly if they have multiple network interfaces configured and they want to use one other than the default. You must first set <code>openshift_node_set_node_ip</code> to <code>True</code>. Otherwise, the SDN would attempt to use the <code>hostname</code> setting or try to resolve the hostname for the IP.</td>
</tr>
</tbody>
</table>
| `public_hostname` | - A master instance where the VPC subnet is not configured for Auto-assign Public IP. For external access to this master, you need to have an ELB or other load balancer configured that would provide the external access needed, or you need to connect over a VPN connection to the internal name of the host.  
  - A master instance where metadata is disabled.  
  - This value is not actually used by the nodes. |
| `public_ip`       | - A master instance where the VPC subnet is not configured for Auto-assign Public IP.  
  - A master instance where metadata is disabled.  
  - This value is not actually used by the nodes. |

If setting `openshift_hostname` to something other than the metadata-provided `private-dns-name` value, the native cloud integration for those providers will no longer work.

For EC2 hosts in particular, they must be deployed in a VPC that has both DNS host names and DNS resolution enabled, and `openshift_hostname` should not be overridden.
Post-Installation Configuration for Cloud Providers

Following the installation process, you can configure OpenShift for **AWS**, **OpenStack**, or **GCE**.

### 2.2.4. Host Preparation

Before installing OpenShift, you must first prepare each host per the following.

#### 2.2.4.1. Software Prerequisites

**Installing an Operating System**

A base installation of RHEL 7.1 or later or RHEL Atomic Host 7.1.6 or later is required for master and node hosts. See the following documentation for the respective installation instructions, if required:

- [Red Hat Enterprise Linux 7 Installation Guide](#)
- [Red Hat Enterprise Linux Atomic Host 7 Installation and Configuration Guide](#)

**Registering the Hosts**

Each host must be registered using Red Hat Subscription Manager (RHSM) and have an active OpenShift Enterprise subscription attached to access the required packages.

1. On each host, register with RHSM:

   ```bash
   # subscription-manager register --username=<user_name> --password=<password>
   ```

2. List the available subscriptions:

   ```bash
   # subscription-manager list --available
   ```

3. In the output for the previous command, find the pool ID for an OpenShift Enterprise subscription and attach it:

   ```bash
   # subscription-manager attach --pool=<pool_id>
   ```

   **NOTE**

   When finding the pool ID, the related subscription name might include either "OpenShift Enterprise" or "OpenShift Container Platform", due to the product name change introduced with version 3.3.

If you plan to configure multiple masters with the **advanced installation** using the **pacemaker** HA method, you must also attach a subscription for **High Availability Add-on for Red Hat Enterprise Linux**:

```bash
# subscription-manager attach --pool=<pool_id_for_rhel_ha>
```
NOTE

The High Availability Add-on for Red Hat Enterprise Linux subscription is provided separately from the OpenShift Enterprise subscription.

4. Disable all repositories and enable only the required ones:

```
# subscription-manager repos --disable="*"
# subscription-manager repos \
   --enable="rhel-7-server-rpms" \
   --enable="rhel-7-server-extras-rpms" \
   --enable="rhel-7-server-ose-3.1-rpms"
```

If you plan to use the pacemaker HA method, enable the following repository as well:

```
# subscription-manager repos \
   --enable="rhel-ha-for-rhel-7-server-rpms"
```

Managing Packages

For RHEL 7 systems:

1. Install the following base packages:

```
# yum install wget git net-tools bind-utils iptables-services bridge-utils bash-completion
```

2. Update the system to the latest packages:

```
# yum update
```

3. Install the following package, which provides OpenShift utilities and pulls in other tools required by the quick and advanced installation methods, such as Ansible and related configuration files:

```
# yum install atomic-openshift-utils
```

4. Install the following *-excluder packages on each RHEL 7 system, which helps ensure your systems stay on the correct versions of atomic-openshift and docker packages when you are not trying to upgrade, according to the OpenShift Enterprise version:

```
# yum install atomic-openshift-excluder atomic-openshift-docker-excluder
```

5. The *-excluder packages add entries to the exclude directive in the host’s /etc/yum.conf file when installed. Run the following command on each host to remove the atomic-openshift packages from the list for the duration of the installation.

```
# atomic-openshift-excluder unexclude
```

For RHEL Atomic Host 7 systems:

1. Ensure the host is up to date by upgrading to the latest Atomic tree if one is available:
2. After the upgrade is completed and prepared for the next boot, reboot the host:

   # systemctl reboot

**Installing Docker**

At this point, you should install Docker on all master and node hosts. This allows you to configure your Docker storage options before installing OpenShift.

### IMPORTANT

Docker 1.9 is currently not supported due to performance issues. See the Red Hat Knowledgebase for details, including steps on how to configure `yum` so that later Docker versions are not installed via `yum update`. Otherwise, running `yum update` after installing OpenShift Enterprise will upgrade Docker and put your cluster in an unsupported configuration.

1. For RHEL 7 systems, install Docker 1.8.

   **NOTE**

   Docker should already be installed, configured, and running by default on RHEL Atomic Host 7 systems.

   The `atomic-openshift-docker-excluder` package that was installed in Software Prerequisites should ensure that the correct version of Docker is installed in this step:

   # yum install docker

   After the package installation is complete, verify that version 1.8.2 was installed:

   # docker version

2. Edit the `/etc/sysconfig/docker` file and add `--insecure-registry 172.30.0.0/16` to the `OPTIONS` parameter. For example:

   OPTIONS='--selinux-enabled --insecure-registry 172.30.0.0/16'

   The `--insecure-registry` option instructs the Docker daemon to trust any Docker registry on the indicated subnet, rather than requiring a certificate.

### IMPORTANT

172.30.0.0/16 is the default value of the `servicesSubnet` variable in the `master-config.yaml` file. If this has changed, then the `--insecure-registry` value in the above step should be adjusted to match, as it is indicating the subnet for the registry to use. Note that the `openshift_master_portal_net` variable can be set in the Ansible inventory file and used during the advanced installation method to modify the `servicesSubnet` variable.
NOTE

After the initial OpenShift installation is complete, you can choose to secure the integrated Docker registry, which involves adjusting the `--insecure-registry` option accordingly.

2.2.4.2. Configuring Docker Storage

Docker containers and the images they are created from are stored in Docker’s storage back end. This storage is ephemeral and separate from any persistent storage allocated to meet the needs of your applications.

For RHEL Atomic Host

The default storage back end for Docker on RHEL Atomic Host is a thin pool logical volume, which is supported for production environments. You must ensure that enough space is allocated for this volume per the Docker storage requirements mentioned in System Requirements.

If you do not have enough allocated, see Managing Storage with Docker Formatted Containers for details on using `docker-storage-setup` and basic instructions on storage management in RHEL Atomic Host.

For RHEL

The default storage back end for Docker on RHEL 7 is a thin pool on loopback devices, which is not supported for production use and only appropriate for proof of concept environments. For production environments, you must create a thin pool logical volume and re-configure Docker to use that volume.

You can use the `docker-storage-setup` script included with Docker to create a thin pool device and configure Docker’s storage driver. This can be done after installing Docker and should be done before creating images or containers. The script reads configuration options from the `/etc/sysconfig/docker-storage-setup` file and supports three options for creating the logical volume:

- **Option A** Use an additional block device.
- **Option B** Use an existing, specified volume group.
- **Option C** Use the remaining free space from the volume group where your root file system is located.

Option A is the most robust option, however it requires adding an additional block device to your host before configuring Docker storage. Options B and C both require leaving free space available when provisioning your host.

1. Create the `docker-pool` volume using one of the following three options:

   - **Option A** Use an additional block device.
     
     In `/etc/sysconfig/docker-storage-setup`, set `DEVS` to the path of the block device you wish to use. Set `VG` to the volume group name you wish to create; `docker-vg` is a reasonable choice. For example:

     ```
     # cat <<EOF > /etc/sysconfig/docker-storage-setup
     DEVS=/dev/vdc
     VG=docker-vg
     EOF
     ```
Then run `docker-storage-setup` and review the output to ensure the `docker-pool` volume was created:

```bash
# docker-storage-setup
[5/1868] 0
Checking that no-one is using this disk right now ... OK

Disk /dev/vdc: 31207 cylinders, 16 heads, 63 sectors/track
sfdisk: /dev/vdc: unrecognized partition table type

Old situation:
sfdisk: No partitions found

New situation:
Units: sectors of 512 bytes, counting from 0

<table>
<thead>
<tr>
<th>Device</th>
<th>Boot</th>
<th>Start</th>
<th>End</th>
<th>#sectors</th>
<th>Id</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dev/vdc1</td>
<td></td>
<td>2048</td>
<td>31457279</td>
<td>31455232</td>
<td>8e</td>
<td>Linux LVM</td>
</tr>
<tr>
<td>/dev/vdc2</td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
<td>Empty</td>
</tr>
<tr>
<td>/dev/vdc3</td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
<td>Empty</td>
</tr>
<tr>
<td>/dev/vdc4</td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
<td>Empty</td>
</tr>
</tbody>
</table>

Warning: partition 1 does not start at a cylinder boundary
Warning: partition 1 does not end at a cylinder boundary
Warning: no primary partition is marked bootable (active)
This does not matter for LILO, but the DOS MBR will not boot this disk.
Successfully wrote the new partition table

Re-reading the partition table ...

If you created or changed a DOS partition, /dev/foo7, say, then use `dd(1)` to zero the first 512 bytes: `dd if=/dev/zero of=/dev/foo7 bs=512 count=1`
(See `fdisk(8)`.)
Physical volume "/dev/vdc1" successfully created
Volume group "docker-vg" successfully created
Rounding up size to full physical extent 16.00 MiB
Logical volume "docker-poolmeta" created.
Logical volume "docker-pool" created.
WARNING: Converting logical volume docker-vg/docker-pool and docker-vg/docker-poolmeta to pool's data and metadata volumes.
THIS WILL DESTROY CONTENT OF LOGICAL VOLUME (filesystem etc.)
Converted docker-vg/docker-pool to thin pool.
Logical volume "docker-pool" changed.

- Option B) Use an existing, specified volume group.
  In `/etc/sysconfig/docker-storage-setup`, set `VG` to the desired volume group. For example:

```bash
# cat <<EOF > /etc/sysconfig/docker-storage-setup
VG=docker-vg
EOF
```
Then run `docker-storage-setup` and review the output to ensure the `docker-pool` volume was created:

```bash
# docker-storage-setup
Rounding up size to full physical extent 16.00 MiB
Logical volume "docker-poolmeta" created.
Logical volume "docker-pool" created.
WARNING: Converting logical volume docker-vg/docker-pool and docker-vg/docker-poolmeta to pool's data and metadata volumes.
THIS WILL DESTROY CONTENT OF LOGICAL VOLUME (filesystem etc.)
Converted docker-vg/docker-pool to thin pool.
Logical volume "docker-pool" changed.
```

- Option C) Use the remaining free space from the volume group where your root file system is located.

Verify that the volume group where your root file system resides has the desired free space, then run `docker-storage-setup` and review the output to ensure the `docker-pool` volume was created:

```bash
# docker-storage-setup
Rounding up size to full physical extent 32.00 MiB
Logical volume "docker-poolmeta" created.
Logical volume "docker-pool" created.
WARNING: Converting logical volume rhel/docker-pool and rhel/docker-poolmeta to pool's data and metadata volumes.
THIS WILL DESTROY CONTENT OF LOGICAL VOLUME (filesystem etc.)
Converted rhel/docker-pool to thin pool.
Logical volume "docker-pool" changed.
```

2. Verify your configuration. You should have a `dm.thinpooldev` value in the `/etc/sysconfig/docker-storage` file and a `docker-pool` logical volume:

```bash
# cat /etc/sysconfig/docker-storage
DOCKER_STORAGE_OPTIONS=--storage-opt dm.fs=xfs --storage-opt dm.thinpooldev=/dev/mapper/docker--vg-docker--pool
```

```bash
# lvs
LV          VG   Attr       LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
docker-pool rhel twi-a-t---  9.29g             0.00   0.12
```

**IMPORTANT**

Before using Docker or OpenShift, verify that the `docker-pool` logical volume is large enough to meet your needs. The `docker-pool` volume should be 60% of the available volume group and will grow to fill the volume group via LVM monitoring.

3. Check if Docker is running:

```bash
# systemctl is-active docker
```

4. If Docker has not yet been started on the host, enable and start the service:
# systemctl enable docker
# systemctl start docker

If Docker is already running, re-initialize Docker:

```
# systemctl stop docker
# rm -rf /var/lib/docker/*
# systemctl restart docker
```

If there is any content in `/var/lib/docker/`, it must be deleted. Files will be present if Docker has been used prior to the installation of OpenShift.

### Reconfiguring Docker Storage

Should you need to reconfigure Docker storage after having created the `docker-pool`, you should first remove the `docker-pool` logical volume. If you are using a dedicated volume group, you should also remove the volume group and any associated physical volumes before reconfiguring `docker-storage-setup` according to the instructions above.

See [Logical Volume Manager Administration](#) for more detailed information on LVM management.

### Managing Docker Container Logs

Sometimes a container’s log file (the `/var/lib/docker/containers/<hash>/<hash>-json.log` file on the node where the container is running) can increase to a problematic size. You can manage this by configuring Docker’s `json-file` logging driver to restrict the size and number of log files.

<table>
<thead>
<tr>
<th>Option</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--log-opt max-size</code></td>
<td>Sets the size at which a new log file is created.</td>
</tr>
<tr>
<td><code>--log-opt max-file</code></td>
<td>Sets the file on each host to configure the options.</td>
</tr>
</tbody>
</table>

For example, to set the maximum file size to 1MB and always keep the last three log files, edit the `/etc/sysconfig/docker` file to configure `max-size=1M` and `max-file=3`:

```
OPTIONS='--insecure-registry=172.30.0.0/16 --selinux-enabled --log-opt max-size=1M --log-opt max-file=3'
```

Next, restart the Docker service:

```
# systemctl restart docker
```
Viewing Available Container Logs

Container logs are stored in the `/var/lib/docker/containers/<hash>/` directory on the node where the container is running. For example:

```
# ls -lh
/var/lib/docker/containers/f088349cceac173305d3e2c2e4790051799efe363842fda
b5732f51f5b001fd8/
total 2.6M
-rw-r--r--. 1 root root 5.6K Nov 24 00:12 config.json
-rw-r--r--. 1 root root 649K Nov 24 00:15 f088349cceac173305d3e2c2e4790051799efe363842fdb5732f51f5b001fd8-json.log
-rw-r--r--. 1 root root 977K Nov 24 00:15 f088349cceac173305d3e2c2e4790051799efe363842fdb5732f51f5b001fd8-json.log.1
-rw-r--r--. 1 root root 977K Nov 24 00:15 f088349cceac173305d3e2c2e4790051799efe363842fdb5732f51f5b001fd8-json.log.2
-rw-r--r--. 1 root root 1.3K Nov 24 00:12 hostconfig.json
drwx------. 2 root root    6 Nov 24 00:12 secrets
```

See Docker's documentation for additional information on how to Configure Logging Drivers.

2.2.5. Ensuring Host Access

The quick and advanced installation methods require a user that has access to all hosts. If you want to run the installer as a non-root user, passwordless `sudo` rights must be configured on each destination host.

For example, you can generate an SSH key on the host where you will invoke the installation process:

```
# ssh-keygen
```

Do not use a password.

An easy way to distribute your SSH keys is by using a bash loop:

```
# for host in master.example.com 
   node1.example.com 
   node2.example.com; 
   do ssh-copy-id -i ~/.ssh/id_rsa.pub $host; 
   done
```

Modify the host names in the above command according to your configuration.

2.2.6. Setting Global Proxy Values

The OpenShift Enterprise installer uses the proxy settings in the `/etc/environment` file.

Ensure the following domain suffixes and IP addresses are in the `/etc/environment` file in the `no_proxy` parameter:

- Master and node host names (domain suffix).
- Other internal host names (domain suffix).
- Etcd IP addresses (must be IP addresses and not host names, as etcd access is done by IP address).
- Docker registry IP address.
- Kubernetes IP address, by default 172.30.0.1. Must be the value set in the openshift_portal_net parameter in the Ansible inventory file, by default /etc/ansible/hosts.
- Kubernetes internal domain suffix: cluster.local.
- Kubernetes internal domain suffix: .svc.

The following example assumes http_proxy and https_proxy values are set:

```bash
no_proxy=internal.example.com,10.0.0.1,10.0.0.2,10.0.0.3,cluster.local,cluster.svc,localhost,127.0.0.1,172.30.0.1
```

**NOTE**

Because noproxy does not support CIDR, you can use domain suffixes.

### 2.2.7. What’s Next?

If you are interested in installing OpenShift using the containerized method (optional for RHEL but required for RHEL Atomic Host), see [RPM vs Containerized](#) to ensure that you understand the differences between these methods.

When you are ready to proceed, you can install OpenShift Enterprise using the quick installation or advanced installation method.

### 2.3. RPM VS CONTAINERIZED

#### 2.3.1. Overview

The default method for installing OpenShift on Red Hat Enterprise Linux (RHEL) uses RPMs. Alternatively, you can use the containerized method, which deploys containerized OpenShift master and node components. When targeting a RHEL Atomic Host system, the containerized method is the only available option, and is automatically selected for you based on the detection of the /run/ostree-booted file.

You can easily deploy environments mixing containerized and RPM based installations. For the advanced installation method, you can set the Ansible variable containerized=true in an inventory file on a cluster-wide or per host basis. For the quick installation method, you can choose between the RPM or containerized method on a per host basis during the interactive installation, or set the values manually in an installation configuration file.
NOTE

Containerized installations are supported starting in OpenShift Enterprise 3.1.1. When installing an environment with multiple masters, the load balancer cannot be deployed by the installation process as a container. See Advanced Installation for load balancer requirements using either the native HA or Pacemaker methods.

The following sections detail the differences between the RPM and containerized methods.

2.3.2. Required Images

Containerized installations make use of the following images:

- openshift3/ose
- openshift3/node
- openshift3/openvswitch
- registry.access.redhat.com/rhel7/etcd

By default, all of the above images are pulled from the Red Hat Registry at registry.access.redhat.com.

If you need to use a private registry to pull these images during the installation, you can specify the registry information ahead of time. For the advanced installation method, you can set the following Ansible variables in your inventory file, as required:

```bash
cli_docker_additional_registries=<registry_hostname>
cli_docker_insecure_registries=<registry_hostname>
cli_docker_blocked_registries=<registry_hostname>
```

For the quick installation method, you can export the following environment variables on each target host:

```bash
# export OO_INSTALL_ADDITIONAL_REGISTRIES=<registry_hostname>
# export OO_INSTALL_INSECURE_REGISTRIES=<registry_hostname>
```

Blocked Docker registries cannot currently be specified using the quick installation method.

The configuration of additional, insecure, and blocked Docker registries occurs at the beginning of the installation process to ensure that these settings are applied before attempting to pull any of the required images.

2.3.3. CLI Wrappers

When using containerized installations, a CLI wrapper script is deployed on each master at /usr/local/bin/openshift. The following set of symbolic links are also provided to ease administrative tasks:

<table>
<thead>
<tr>
<th>Symbolic Link</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>/usr/local/bin/oc</td>
<td>Developer CLI</td>
</tr>
<tr>
<td>Symbolic Link</td>
<td>Usage</td>
</tr>
<tr>
<td>-------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>/usr/local/bin/oadm</td>
<td>Administrative CLI</td>
</tr>
<tr>
<td>/usr/local/bin/kubectl</td>
<td>Kubernetes CLI</td>
</tr>
</tbody>
</table>

The wrapper spawns a new container on each invocation, so you may notice it run slightly slower than native CLI operations.

The wrapper scripts mount a limited subset of paths:

- ~/.kube
- /etc/origin/
- /tmp/

Be mindful of this when passing in files to be processed by the `oc` or `oadm` commands. You may find it easier to redirect the input, for example:

```
# oc create -f - < my-file.json
```

**NOTE**

The wrapper is intended only to be used to bootstrap an environment. You should install the CLI tools on another host after you have granted `cluster-admin` privileges to a user. See [Managing Role Bindings](#) and [Get Started with the CLI](#) for more information.

### 2.3.4. Starting and Stopping Containers

The installation process creates relevant `systemd` units which can be used to start, stop, and poll services using normal `systemctl` commands. For containerized installations, these unit names match those of an RPM installation, with the exception of the `etcd` service which is named `etcd_container`.

This change is necessary as currently RHEL Atomic Host ships with the `etcd` package installed as part of the operating system, so a containerized version is used for the OpenShift installation instead. The installation process disables the default `etcd` service. The `etcd` package is slated to be removed from RHEL Atomic Host in the future.

### 2.3.5. File Paths

All OpenShift configuration files are placed in the same locations during containerized installation as RPM based installations and will survive `os-tree` upgrades.

However, the default image stream and template files are installed at `/etc/origin/examples/` for containerized installations rather than the standard `/usr/share/openshift/examples/`, because that directory is read-only on RHEL Atomic Host.

### 2.3.6. Storage Requirements
RHEL Atomic Host installations normally have a very small root file system. However, the etcd, master, and node containers persist data in the /var/lib/ directory. Ensure that you have enough space on the root file system before installing OpenShift; see the System Requirements section for details.

### 2.3.7. Open vSwitch SDN Initialization

OpenShift SDN initialization requires that the Docker bridge be reconfigured and that Docker is restarted. This complicates the situation when the node is running within a container. When using the Open vSwitch (OVS) SDN, you will see the node start, reconfigure Docker, restart Docker (which restarts all containers), and finally start successfully.

In this case, the node service may fail to start and be restarted a few times because the master services are also restarted along with Docker. The current implementation uses a workaround which relies on setting the `Restart=always` parameter in the Docker based `systemd` units.

### 2.4. QUICK INSTALLATION

#### 2.4.1. Overview

The quick installation method allows you to use an interactive CLI utility, the `atomic-openshift-installer` command, to install OpenShift across a set of hosts. This installer can deploy OpenShift components on targeted hosts by either installing RPMs or running containerized services.

This installation method is provided to make the installation experience easier by interactively gathering the data needed to run on each host. The installer is a self-contained wrapper intended for usage on a Red Hat Enterprise Linux (RHEL) 7 system. While RHEL Atomic Host is supported for running containerized OpenShift services, the installer is provided by an RPM not available by default in RHEL Atomic Host, and must therefore be run from a RHEL 7 system. The host initiating the installation does not need to be intended for inclusion in the OpenShift cluster, but it can be.

In addition to running interactive installations from scratch, the `atomic-openshift-installer` command can also be run or re-run using a predefined installation configuration file. This file can be used with the installer to:

- run an unattended installation,
- add nodes to an existing cluster,
- upgrade your cluster, or
- reinstall the OpenShift cluster completely.

Alternatively, you can use the advanced installation method for more complex environments.

#### 2.4.2. Before You Begin

The installer allows you to install OpenShift master and node components on a defined set of hosts.
NOTE

By default, any hosts you designate as masters during the installation process are automatically also configured as nodes so that the masters are configured as part of the OpenShift SDN. The node component on the masters, however, are marked unschedulable, which blocks pods from being scheduled on it. After the installation, you can mark them schedulable if you want.

Before installing OpenShift, you must first satisfy the prerequisites on your hosts, which includes verifying system and environment requirements and properly installing and configuring Docker. You must also be prepared to provide or validate the following information for each of your targeted hosts during the course of the installation:

- User name on the target host that should run the Ansible-based installation (can be root or non-root)
- Host name
- Whether to install components for master, node, or both
- Whether to use the RPM or containerized method
- Internal and external IP addresses

If you are interested in installing OpenShift using the containerized method (optional for RHEL but required for RHEL Atomic Host), see RPM vs Containerized to ensure that you understand the differences between these methods, then return to this topic to continue.

After following the instructions in the Prerequisites topic and deciding between the RPM and containerized methods, you can continue to running an interactive or unattended installation.

2.4.3. Running an Interactive Installation

NOTE

Ensure you have read through Before You Begin.

You can start the interactive installation by running:

```
$ atomic-openshift-installer install
```

Then follow the on-screen instructions to install a new OpenShift Enterprise cluster.

After it has finished, ensure that you back up the `~/config/openshift/installer.cfg.yml` installation configuration file that is created, as it is required if you later want to re-run the installation, add hosts to the cluster, or upgrade your cluster. Then, verify the installation.

2.4.4. Defining an Installation Configuration File

The installer can use a predefined installation configuration file, which contains information about your installation, individual hosts, and cluster. When running an interactive installation, an installation configuration file based on your answers is created for you in `~/config/openshift/installer.cfg.yml`. The
file is created if you are instructed to exit the installation to manually modify the configuration or when the installation completes. You can also create the configuration file manually from scratch to perform an unattended installation.

Example 2.1. Installation Configuration File Specification

```
version: v1
variant: openshift-enterprise
variant_version: 3.1
ansible_ssh_user: root
ansible_log_path: /tmp/ansible.log
hosts:
  - ip: 10.0.0.1
    hostname: master-private.example.com
    public_ip: 24.222.0.1
    public_hostname: master.example.com
    master: true
    node: true
    containerized: true
    connect_to: 24.222.0.1
  - ip: 10.0.0.2
    hostname: node1-private.example.com
    public_ip: 24.222.0.2
    public_hostname: node1.example.com
    node: true
    connect_to: 10.0.0.2
  - ip: 10.0.0.3
    hostname: node2-private.example.com
    public_ip: 24.222.0.3
    public_hostname: node2.example.com
    node: true
    connect_to: 10.0.0.3
```

1. The version of this installation configuration file. As of OpenShift Enterprise (OSE) 3.1, the only valid version here is v1.
2. The OpenShift variant to install. For OSE, set this to openshift-enterprise.
3. A valid version your selected variant. If not specified, this defaults to the newest version for the specified variant. For example: 3.1 or 3.0.
4. Defines which user Ansible uses to SSH in to remote systems for gathering facts and for the installation. By default, this is the root user, but you can set it to any user that has sudo privileges.
5. Defines where the Ansible logs are stored. By default, this is the /tmp/ansible.log file.
6. Defines a list of the hosts onto which you want to install the OpenShift master and node components.
7. Required. Allows the installer to connect to the system and gather facts before proceeding with the install.
Required for unattended installations. If these details are not specified, then this information is pulled from the facts gathered by the installer, and you are asked to confirm the details. If

Determines the type of services that are installed. At least one of these must be set to `true` for the configuration file to be considered valid.

If set to `true`, containerized OpenShift services are run on target master and node hosts instead of installed using RPM packages. If set to `false` or unset, the default RPM method is used. RHEL Atomic Host requires the containerized method, and is automatically selected for you based on the detection of the `/run/ostree-booted` file. See [RPM vs Containerized](#) for more details. Containerized installations are supported starting in OSE 3.1.1.

The IP address that Ansible attempts to connect to when installing, upgrading, or uninstalling the systems. If the configuration file was auto-generated, then this is the value you first enter for the host during that interactive install process.

### 2.4.5. Running an Unattended Installation

**NOTE**

Ensure you have read through the Before You Begin.

Unattended installations allow you to define your hosts and cluster configuration in an installation configuration file before running the installer so that you do not have to go through all of the interactive installation questions and answers. It also allows you to resume an interactive installation you may have left unfinished, and quickly get back to where you left off.

To run an unattended installation, first define an installation configuration file at `~/.config/openshift/installer.cfg.yml`. Then, run the installer with the `-u` flag:

```
$ atomic-openshift-installer -u install
```

By default in interactive or unattended mode, the installer uses the configuration file located at `~/.config/openshift/installer.cfg.yml` if the file exists. If it does not exist, attempting to start an unattended installation fails.

Alternatively, you can specify a different location for the configuration file using the `-c` option, but doing so will require you to specify the file location every time you run the installation:

```
$ atomic-openshift-installer -u -c </path/to/file> install
```

After the unattended installation finishes, ensure that you back up the `~/.config/openshift/installer.cfg.yml` file that was used, as it is required if you later want to re-run the installation, add hosts to the cluster, or upgrade your cluster. Then, verify the installation.

### 2.4.6. Verifying the Installation

After the installation completes:

1. Verify that the master is started and nodes are registered and reporting in Ready status. On the master host, run the following as root:
# oc get nodes

<table>
<thead>
<tr>
<th>NAME</th>
<th>LABELS</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>master.example.com</td>
<td>kubernetes.io/hostname=master.example.com,region=infra,zone=default</td>
<td>Ready, SchedulingDisabled</td>
</tr>
<tr>
<td>node1.example.com</td>
<td>kubernetes.io/hostname=node1.example.com,region=primary,zone=east</td>
<td>Ready</td>
</tr>
<tr>
<td>node2.example.com</td>
<td>kubernetes.io/hostname=node2.example.com,region=primary,zone=west</td>
<td>Ready</td>
</tr>
</tbody>
</table>

2. To verify that the web console is installed correctly, use the master host name and the console port number to access the console with a web browser. For example, for a master host with a hostname of `master.openshift.com` and using the default port of `8443`, the web console would be found at:

```
https://master.openshift.com:8443/console
```

3. Now that the install has been verified, run the following command on each master and node host to add the `atomic-openshift` packages back to the list of yum excludes on the host:

```
# atomic-openshift-excluder exclude
```

Then, see What’s Next for the next steps on configuring your OpenShift cluster.

### 2.4.7. Adding Nodes or Reinstalling the Cluster

You can use the installer to add nodes to your existing cluster, or to reinstall the cluster entirely.

If you installed OpenShift using the installer in either interactive or unattended mode, you can re-run the installation as long as you have an installation configuration file at `~/.config/openshift/installer.cfg.yml` (or specify a different location with the `-c` option).

If you installed using the advanced installation method and therefore do not have an installation configuration file, you can either try creating your own based on your cluster’s current configuration, or see the advanced installation method on how to run the playbook for adding new nodes directly.

To add nodes or reinstall the cluster:

1. Re-run the installer with the `install` subcommand in interactive or unattended mode:

   ```
   $ atomic-openshift-installer [-u] [-c </path/to/file>] install
   ```

2. The installer will detect your installed environment and allow you to either add an additional node or perform a clean install:

   Gathering information from hosts...
   
   Installed environment detected.
   
   By default the installer only adds new nodes to an installed
environment.
Do you want to (1) only add additional nodes or (2) perform a clean install?:

Choose one of the options and follow the on-screen instructions to complete your desired task.

2.4.8. Uninstalling OpenShift Enterprise

You can uninstall OpenShift Enterprise on all hosts in your cluster using the installer by running:

$ atomic-openshift-installer uninstall

See the advanced installation method for more options.

2.4.9. What’s Next?

Now that you have a working OpenShift Enterprise instance, you can:

- Configure authentication; by default, authentication is set to Deny All.
- Deploy an integrated Docker registry.
- Deploy a router.

2.5. ADVANCED INSTALLATION

2.5.1. Overview

For production environments, a reference configuration implemented using Ansible playbooks is available as the advanced installation method for installing OpenShift hosts. Familiarity with Ansible is assumed, however you can use this configuration as a reference to create your own implementation using the configuration management tool of your choosing.

While RHEL Atomic Host is supported for running containerized OpenShift services, the advanced installation method utilizes Ansible, which is not available in RHEL Atomic Host, and must therefore be run from a RHEL 7 system. The host initiating the installation does not need to be intended for inclusion in the OpenShift cluster, but it can be.

Alternatively, you can use the quick installation method if you prefer an interactive installation experience.

IMPORTANT

Running Ansible playbooks with the --tags or --check options is not supported by Red Hat.

2.5.2. Before You Begin

Before installing OpenShift, you must first see the Prerequisites topic to prepare your hosts, which includes verifying system and environment requirements per component type and properly installing and configuring Docker. It also includes installing Ansible version 1.8.4 or later, as the advanced installation method is based on Ansible playbooks and as such requires directly invoking Ansible.
If you are interested in installing OpenShift using the containerized method (optional for RHEL but required for RHEL Atomic Host), see RPM vs Containerized to ensure that you understand the differences between these methods, then return to this topic to continue.

After following the instructions in the Prerequisites topic and deciding between the RPM and containerized methods, you can continue in this topic to Configuring Ansible.

2.5.3. Configuring Ansible

The `/etc/ansible/hosts` file is Ansible’s inventory file for the playbook to use during the installation. The inventory file describes the configuration for your OpenShift cluster. You must replace the default contents of the file with your desired configuration.

The following sections describe commonly-used variables to set in your inventory file during an advanced installation, followed by example inventory files you can use as a starting point for your installation. The examples describe various environment topographies, including using multiple masters for high availability. You can choose an example that matches your requirements, modify it to match your own environment, and use it as your inventory file when running the advanced installation.

2.5.3.1. Configuring Host Variables

To assign environment variables to hosts during the Ansible installation, indicate the desired variables in the `/etc/ansible/hosts` file after the host entry in the `[masters]` or `[nodes]` sections. For example:

```
[masters]
ec2-52-6-179-239.compute-1.amazonaws.com openshift_public_hostname=ose3-master.public.example.com
```

The following table describes variables for use with the Ansible installer that can be assigned to individual host entries:

Table 2.8. Host Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>openshift_hostname</td>
<td>This variable overrides the internal cluster host name for the system. Use this when the system's default IP address does not resolve to the system host name.</td>
</tr>
<tr>
<td>openshift_public_hostname</td>
<td>This variable overrides the system's public host name. Use this for cloud installations, or for hosts on networks using a network address translation (NAT).</td>
</tr>
<tr>
<td>openshift_ip</td>
<td>This variable overrides the cluster internal IP address for the system. Use this when using an interface that is not configured with the default route.</td>
</tr>
<tr>
<td>openshift_public_ip</td>
<td>This variable overrides the system's public IP address. Use this for cloud installations, or for hosts on networks using a network address translation (NAT).</td>
</tr>
</tbody>
</table>
### Variable

<table>
<thead>
<tr>
<th>Variable</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>containerized</strong></td>
<td>If set to true, containerized OpenShift services are run on target master and node hosts instead of installed using RPM packages. If set to false or unset, the default RPM method is used. RHEL Atomic Host requires the containerized method, and is automatically selected for you based on the detection of the /run/ostree-booted file. See RPM vs Containerized for more details. Containerized installations are supported starting in OSE 3.1.1.</td>
</tr>
<tr>
<td><strong>openshift_node_labels</strong></td>
<td>This variable adds labels to nodes during installation. See Configuring Node Host Labels for more details.</td>
</tr>
<tr>
<td><strong>openshift_node_kubelet_args</strong></td>
<td>This variable is used to configure kubeletArguments on nodes, such as arguments used in container and image garbage collection, and to specify resources per node. kubeletArguments are key value pairs that are passed directly to the Kubelet that match the Kubelet's command line arguments. kubeletArguments are not migrated or validated and may become invalid if used. These values override other settings in node configuration which may cause invalid configurations. Example usage: {'image-gc-high-threshold': [90], 'image-gc-low-threshold': [80]}.</td>
</tr>
<tr>
<td><strong>openshift_docker_options</strong></td>
<td>This variable configures additional Docker options within /etc/sysconfig/docker, such as options used in Managing Docker Container Logs. Example usage: &quot;--log-driver json-file --log-opt max-size=1M --log-opt max-file=3&quot;.</td>
</tr>
</tbody>
</table>

### 2.5.3.2. Configuring Cluster Variables

To assign environment variables during the Ansible install that apply more globally to your OpenShift cluster overall, indicate the desired variables in the /etc/ansible/hosts file on separate, single lines within the [OSEv3:vars] section. For example:

```
[OSEv3:vars]

openshift_master_identity_providers=[["name": 'htpasswd_auth', 'login': 'true', 'challenge': 'true', 'kind': 'HTPasswordPasswordIdentityProvider', 'filename': '/etc/origin/master/htpasswd']]

osm_default_subdomain=apps.test.example.com
```
The following table describes variables for use with the Ansible installer that can be assigned cluster-wide:

**Table 2.9. Cluster Variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ansible_ssh_user</code></td>
<td>This variable sets the SSH user for the installer to use and defaults to <code>root</code>. This user should allow SSH-based authentication <em>without requiring a password</em>. If using SSH key-based authentication, then the key should be managed by an SSH agent.</td>
</tr>
<tr>
<td><code>ansible_sudo</code></td>
<td>If <code>ansible_ssh_user</code> is not <code>root</code>, this variable must be set to <code>true</code> and the user must be configured for passwordless <code>sudo</code>.</td>
</tr>
<tr>
<td><code>containerized</code></td>
<td>If set to <code>true</code>, containerized OpenShift services are run on all target master and node hosts in the cluster instead of installed using RPM packages. If set to <code>false</code> or unset, the default RPM method is used. RHEL Atomic Host requires the containerized method, and is automatically selected for you based on the detection of the <code>/run/ostree-booted</code> file. See RPM vs Containerized for more details. Containerized installations are supported starting in OSE 3.1.1.</td>
</tr>
<tr>
<td><code>openshift_master_cluster_hostname</code></td>
<td>This variable overrides the host name for the cluster, which defaults to the host name of the master.</td>
</tr>
<tr>
<td><code>openshift_master_cluster_public_hostname</code></td>
<td>This variable overrides the public host name for the cluster, which defaults to the host name of the master. If you use an external load balancer, specify the address of the external load balancer. For example: <code>---- openshift_master_cluster_public_hostname=openshift-ansible.public.example.com ----</code></td>
</tr>
<tr>
<td><code>openshift_master_cluster_method</code></td>
<td>Optional. This variable defines the HA method when deploying multiple masters. Can be either <code>native</code> or <code>pacemaker</code>. See Multiple Masters for more information.</td>
</tr>
<tr>
<td><code>openshift_master_cluster_password</code></td>
<td>These variables are only required when using the <code>pacemaker</code> HA method.</td>
</tr>
<tr>
<td><code>openshift_master_cluster_vip</code></td>
<td>For <code>openshift_master_cluster_vip</code>, the virtual IP (VIP) is assigned to the active master automatically, so the IP must be available in the cluster network. This IP should be in the same network and able to communicate with any other master, <code>etcd</code>, and node hosts' IP. See Multiple Masters for more information.</td>
</tr>
<tr>
<td><code>openshift_master_cluster_public_vip</code></td>
<td></td>
</tr>
<tr>
<td>Variable</td>
<td>Purpose</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>openshift_rolling_restart_mode</code></td>
<td>This variable enables rolling restarts of HA masters (i.e., masters are taken down one at a time) when running the upgrade playbook directly. It defaults to <code>services</code>, which allows rolling restarts of services on the masters. It can instead be set to <code>system</code>, which enables rolling, full system restarts and also works for single master clusters.</td>
</tr>
<tr>
<td><code>os_sdn_network_plugin_name</code></td>
<td>This variable configures which OpenShift SDN plug-in to use for the pod network, which defaults to <code>redhat/openshift-ovs-subnet</code> for the standard SDN plug-in. Set the variable to <code>redhat/openshift-ovs-multitenant</code> to use the multitenant plug-in.</td>
</tr>
<tr>
<td><code>openshift_master_identity_providers</code></td>
<td>This variable overrides the identity provider, which defaults to <code>Deny All</code>.</td>
</tr>
<tr>
<td><code>openshift_master_named_certificates</code></td>
<td>These variables are used to configure custom certificates which are deployed as part of the installation. See Configuring Custom Certificates for more information.</td>
</tr>
<tr>
<td><code>openshift_master_overwrite_named_certificates</code></td>
<td>These variables override defaults for session options in the OAuth configuration. See Configuring Session Options for more information.</td>
</tr>
<tr>
<td><code>openshift_master_session_name</code></td>
<td></td>
</tr>
<tr>
<td><code>openshift_master_session_max_seconds</code></td>
<td></td>
</tr>
<tr>
<td><code>openshift_master_session_auth_secrets</code></td>
<td></td>
</tr>
<tr>
<td><code>openshift_master_session_encryption_secrets</code></td>
<td></td>
</tr>
<tr>
<td><code>openshift_master_portal_net</code></td>
<td>This variable configures the subnet in which services will be created within the OpenShift Enterprise SDN. This network block should be private and must not conflict with any existing network blocks in your infrastructure to which pods, nodes, or the master may require access to, or the installation will fail. Defaults to <code>172.30.0.0/16</code>, and cannot be re-configured after deployment. If changing from the default, avoid <code>172.16.0.0/16</code>, which the <code>docker0</code> network bridge uses by default, or modify the <code>docker0</code> network.</td>
</tr>
<tr>
<td><code>openshift_hosted_router_selector</code></td>
<td>Default node selector for automatically deploying router pods. See Configuring Node Host Labels for details.</td>
</tr>
<tr>
<td><code>openshift_registry_selector</code></td>
<td>Default node selector for automatically deploying registry pods. See Configuring Node Host Labels for details.</td>
</tr>
<tr>
<td>Variable</td>
<td>Purpose</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>osm_default_subdomain</td>
<td>This variable overrides the default subdomain to use for exposed routes.</td>
</tr>
<tr>
<td>osm_default_node_selector</td>
<td>This variable overrides the node selector that projects will use by default when placing pods.</td>
</tr>
<tr>
<td>osm_cluster_network_cidr</td>
<td>This variable overrides the SDN cluster network CIDR block. This is the network from which pod IPs are assigned. This network block should be a private block and must not conflict with existing network blocks in your infrastructure to which pods, nodes, or the master may require access. Defaults to 10.1.0.0/16 and cannot be arbitrarily re-configured after deployment, although certain changes to it can be made in the SDN master configuration.</td>
</tr>
<tr>
<td>osm_host_subnet_length</td>
<td>This variable specifies the size of the per host subnet allocated for pod IPs by OpenShift SDN. Defaults to 8 which means that a subnet of size /24 is allocated to each host; for example, given the default 10.1.0.0/16 cluster network, this will allocate 10.1.0.0/24, 10.1.1.0/24, 10.1.2.0/24, and so on. This cannot be re-configured after deployment.</td>
</tr>
</tbody>
</table>

## 2.5.3.3. Configuring Node Host Labels

You can assign labels to node hosts during the Ansible install by configuring the `/etc/ansible/hosts` file. Labels are useful for determining the placement of pods onto nodes using the scheduler. Other than `region=infra` (discussed below), the actual label names and values are arbitrary and can be assigned however you see fit per your cluster’s requirements.

To assign labels to a node host during an Ansible install, use the `openshift_node_labels` variable with the desired labels added to the desired node host entry in the `[nodes]` section. In the following example, labels are set for a region called `primary` and a zone called `east`:

```
[nodes]
node1.example.com openshift_node_labels="{'region': 'primary', 'zone': 'east'}"
```

The `openshift_router_selector` and `openshift_registry_selector` Ansible settings are set to `region=infra` by default:

```
# default selectors for router and registry services
# openshift_router_selector='region=infra'
# openshift_registry_selector='region=infra'
```

The default router and registry will be automatically deployed if nodes exist that match the selector settings above. For example:

```
[nodes]
node1.example.com openshift_node_labels="
{'region':'infra','zone':'default'}"
```

## 2.5.3.4. Marking Masters as Unschedulable Nodes
Any hosts you designate as masters during the installation process should also be configured as nodes by adding them to the [nodes] section so that the masters are configured as part of the OpenShift SDN.

However, in order to ensure that your masters are not burdened with running pods, you can make them unschedulable by adding the openshift_scheduleable=false option any node that is also a master. For example:

```
[nodes]
master.example.com  openshift_node_labels="
{"region":"infra","zone":"default"}"  openshift_scheduleable=false
```

### 2.5.3.5. Configuring Session Options

Session options in the OAuth configuration are configurable in the inventory file. By default, Ansible populates a sessionSecretsFile with generated authentication and encryption secrets so that sessions generated by one master can be decoded by the others. The default location is 
/etc/origin/master/session-secrets.yaml, and this file will only be re-created if deleted on all masters.

You can set the session name and maximum number of seconds with openshift_master_session_name and openshift_master_session_max_seconds:

```
openshift_master_session_name=ssn
openshift_master_session_max_seconds=3600
```

If provided, openshift_master_session_auth_secrets and openshift_master_encryption_secrets must be equal length.

For openshift_master_session_auth_secrets, used to authenticate sessions using HMAC, it is recommended to use secrets with 32 or 64 bytes:

```
openshift_master_session_auth_secrets=['DONT+USE+THIS+SECRET+b4NV+pmZNS0']
```

For openshift_master_encryption_secrets, used to encrypt sessions, secrets must be 16, 24, or 32 characters long, to select AES-128, AES-192, or AES-256:

```
openshift_master_session_encryption_secrets=
['DONT+USE+THIS+SECRET+b4NV+pmZNS0']
```

### 2.5.3.6. Configuring Custom Certificates

Custom serving certificates for the public host names of the OpenShift API and web console can be deployed during an advanced installation and are configurable in the inventory file.

**NOTE**

Custom certificates should only be configured for the host name associated with the publicMasterURL which can be set using openshift_master_cluster_public_hostname. Using a custom serving certificate for the host name associated with the masterURL (openshift_master_cluster_hostname) will result in TLS errors as infrastructure components will attempt to contact the master API using the internal masterURL host.
Certificate and key file paths can be configured using the `openshift_master_named_certificates` cluster variable:

```
openshift_master_named_certificates=\[
    \{
        "certfile": "/path/to/custom1.crt",
        "keyfile": "/path/to/custom1.key"
    \}
```

File paths must be local to the system where Ansible will be run. Certificates are copied to master hosts and are deployed within the `/etc/origin/master/named_certificates/` directory.

Ansible detects a certificate’s Common Name and Subject Alternative Names. Detected names can be overridden by providing the "names" key when setting `openshift_master_named_certificates`:

```
openshift_master_named_certificates=\[
    \{
        "certfile": "/path/to/custom1.crt",
        "keyfile": "/path/to/custom1.key",
        "names": ["public-master-host.com"]
    \}
```

Certificates configured using `openshift_master_named_certificates` are cached on masters, meaning that each additional Ansible run with a different set of certificates results in all previously deployed certificates remaining in place on master hosts and within the master configuration file.

If you would like `openshift_master_named_certificates` to be overwritten with the provided value (or no value), specify the `openshift_master_overwrite_named_certificates` cluster variable:

```
openshift_master_overwrite_named_certificates=true
```

For a more complete example, consider the following cluster variables in an inventory file:

```
openshift_master_cluster_method=native
openshift_master_cluster_hostname=lb.openshift.com
openshift_master_cluster_public_hostname=custom.openshift.com
```

To overwrite the certificates on a subsequent Ansible run, you could set the following:

```
openshift_master_named_certificates=\[
    \{
        "certfile": "/root/STAR.openshift.com.crt",
        "keyfile": "/root/STAR.openshift.com.key",
        "names": ["custom.openshift.com"]
    \}

openshift_master_overwrite_named_certificates=true
```

### 2.5.4. Single Master Examples

You can configure an environment with a single master and multiple nodes, and either a single embedded `etcd` or multiple external `etcd` hosts.

**NOTE**

Moving from a single master cluster to multiple masters after installation is not supported.

**Single Master and Multiple Nodes**

The following table describes an example environment for a single master (with embedded `etcd`) and two nodes:
### Example 2.2. Single Master and Multiple Nodes Inventory File

```bash
# Create an OSEv3 group that contains the masters and nodes groups
[OSEv3:children]
masters
nodes

# Set variables common for all OSEv3 hosts
[OSEv3:vars]
# SSH user, this user should allow ssh based auth without requiring a password
ansible_ssh_user=root

# If ansible_ssh_user is not root, ansible_sudo must be set to true
#ansible_sudo=true

deployment_type=openshift-enterprise

# uncomment the following to enable htpasswd authentication; defaults to DenyAllPasswordIdentityProvider
#openshift_master_identity_providers=[{
#  'name': 'htpasswd_auth',
#  'login': 'true',
#  'challenge': 'true',
#  'kind': 'HTPasswdPasswordIdentityProvider',
#  'filename': '/etc/origin/master/htpasswd'}]

# host group for masters
[masters]
master.example.com

# host group for nodes, includes region info
[nodes]
master.example.com openshift_node_labels="{"region": "infra", "zone": "default"}"
node1.example.com openshift_node_labels="{"region": "primary", "zone": "east"}"
node2.example.com openshift_node_labels="{"region": "primary", "zone": "west"}"
```

To use this example, modify the file to match your environment and specifications, and save it as `/etc/ansible/hosts`.

**Single Master, Multiple etcd, and Multiple Nodes**
The following table describes an example environment for a single **master**, three **etcd** hosts, and two **nodes**:

<table>
<thead>
<tr>
<th>Host Name</th>
<th>Infrastructure Component to Install</th>
</tr>
</thead>
<tbody>
<tr>
<td>master.example.com</td>
<td>Master and node</td>
</tr>
<tr>
<td>etcd1.example.com</td>
<td>etcd</td>
</tr>
<tr>
<td>etcd2.example.com</td>
<td></td>
</tr>
<tr>
<td>etcd3.example.com</td>
<td></td>
</tr>
<tr>
<td>node1.example.com</td>
<td>Node</td>
</tr>
<tr>
<td>node2.example.com</td>
<td></td>
</tr>
</tbody>
</table>

**NOTE**

When specifying multiple **etcd** hosts, external **etcd** is installed and configured. Clustering of OpenShift’s embedded **etcd** is not supported.

You can see these example hosts present in the **[masters]**, **[nodes]**, and **[etcd]** sections of the following example inventory file:

**Example 2.3. Single Master, Multiple etcd, and Multiple Nodes Inventory File**

```plaintext
# Create an OSEv3 group that contains the masters, nodes, and etcd groups
[OSEv3:children]
masters
nodes
etcd

# Set variables common for all OSEv3 hosts
[OSEv3:vars]
ansible_ssh_user=root
deployment_type=openshift-enterprise

# uncomment the following to enable htpasswd authentication; defaults to DenyAllPasswordIdentityProvider
#openshift_master_identity_providers=[{'name': 'htpasswd_auth', 'login': 'true', 'challenge': 'true', 'kind': 'HTPasswdPasswordIdentityProvider', 'filename': '/etc/origin/master/htpasswd'}]

# host group for masters
[masters]
master.example.com

# host group for etcd
[etcd]
etcd1.example.com
```
etcd2.example.com
etcd3.example.com

# host group for nodes, includes region info
[nodes]
master.example.com openshift_node_labels="{'region': 'infra', 'zone': 'default'}"
node1.example.com openshift_node_labels="{'region': 'primary', 'zone': 'east'}"
node2.example.com openshift_node_labels="{'region': 'primary', 'zone': 'west'}"

To use this example, modify the file to match your environment and specifications, and save it as `/etc/ansible/hosts`.

2.5.5. Multiple Masters Examples

You can configure an environment with multiple masters, multiple etcd hosts, and multiple nodes. Configuring multiple masters for high availability (HA) ensures that the cluster has no single point of failure.

NOTE
Moving from a single master cluster to multiple masters after installation is not supported.

When configuring multiple masters, the advanced installation supports two high availability (HA) methods:

Table 2.10. HA Master Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>native</td>
<td>Leverages the native HA master capabilities built into OpenShift and can be combined with any load balancing solution. If a host is defined in the [lb] section of the inventory file, Ansible installs and configures HAProxy automatically as the load balancing solution. If no host is defined, it is assumed you have pre-configured a load balancing solution of your choice to balance the master API (port 8443) on all master hosts.</td>
</tr>
<tr>
<td>pacemaker</td>
<td>Configures Pacemaker as the load balancer for multiple masters. Requires a High Availability Add-on for Red Hat Enterprise Linux subscription, which is provided separately from the OpenShift Enterprise subscription.</td>
</tr>
</tbody>
</table>

NOTE
For more on these methods and the high availability master architecture, see Kubernetes Infrastructure.

To configure multiple masters, choose one of the above HA methods, and refer to the relevant example section that follows.

Multiple Masters Using Native HA

The following describes an example environment for three masters, one HAProxy load balancer, three etcd hosts, and two nodes using the native HA method:
<table>
<thead>
<tr>
<th>Host Name</th>
<th>Infrastructure Component to Install</th>
</tr>
</thead>
<tbody>
<tr>
<td>master1.example.com</td>
<td>Master (clustered using native HA) and node</td>
</tr>
<tr>
<td>master2.example.com</td>
<td></td>
</tr>
<tr>
<td>master3.example.com</td>
<td></td>
</tr>
<tr>
<td>lb.example.com</td>
<td>HAProy to load balance API master endpoints</td>
</tr>
<tr>
<td>etcd1.example.com</td>
<td>etcd</td>
</tr>
<tr>
<td>etcd2.example.com</td>
<td></td>
</tr>
<tr>
<td>etcd3.example.com</td>
<td></td>
</tr>
<tr>
<td>node1.example.com</td>
<td>Node</td>
</tr>
<tr>
<td>node2.example.com</td>
<td></td>
</tr>
</tbody>
</table>

**NOTE**

When specifying multiple etcd hosts, external etcd is installed and configured. Clustering of OpenShift’s embedded etcd is not supported.

You can see these example hosts present in the [masters], [etcd], [lb], and [nodes] sections of the following example inventory file:

```yaml
Example 2.4. Multiple Masters Using HAProy Inventory File

# Create an OSEv3 group that contains the master, nodes, etcd, and lb groups.
# The lb group lets Ansible configure HAProy as the load balancing solution.
# Comment lb out if your load balancer is pre-configured.
[OSEv3:children]
masters
nodes
etcd
lb

[OSEv3:vars]
ansible_ssh_user=root
deployment_type=openshift-enterprise

# Uncomment the following to enable htpasswd authentication; defaults to
# DenyAllPasswordIdentityProvider.
#openshift_master_identity_providers=[$'name': 'htpasswd_auth', 'login': 'true', 'challenge': 'true', 'kind': 'HTPasswdPasswordIdentityProvider',
Native high availability cluster method with optional load balancer.
If no lb group is defined installer assumes that a load balancer has
been preconfigured. For installation the value of
openshift_master_cluster_hostname must resolve to the load balancer
or to one or all of the masters defined in the inventory if no load
balancer is present.

```
openshift_master_cluster_method=native
openshift_master_cluster_hostname=openshift-cluster.example.com
openshift_master_cluster_public_hostname=openshift-cluster.example.com
```

# override the default controller lease ttl
# osm_controller_lease_ttl=30

# enable ntp on masters to ensure proper failover
openshift_clock_enabled=true

# host group for masters
[masters]
master1.example.com
master2.example.com
master3.example.com

# host group for etcd
[etcd]
etcd1.example.com
etcd2.example.com
etcd3.example.com

# Specify load balancer host
[lb]
lb.example.com

# host group for nodes, includes region info
[nodes]
master[1:3].example.com openshift_node_labels="{"region": "infra", "zone": "default"}"
node1.example.com openshift_node_labels="{"region": "primary", "zone": "east"}"
node2.example.com openshift_node_labels="{"region": "primary", "zone": "west"}"

To use this example, modify the file to match your environment and specifications, and save it as /etc/ansible/hosts.

Note the following when using the native HA method:

- The advanced installation method does not currently support multiple HAProxy load balancers in an active-passive setup. See the Load Balancer Administration documentation for post-installation amendments, or use the pacemaker method if you require this capability.

- In a HAProxy setup, controller manager servers run as standalone processes. They elect their active leader with a lease stored in etcd. The lease expires after 30 seconds by default. If a failure happens on an active controller server, it will take up to this number of seconds to elect
another leader. The interval can be configured with the `osm_controller_lease_ttl` variable.

### Multiple Masters Using Pacemaker

The following describes an example environment for three masters, three etcd hosts, and two nodes using the pacemaker HA method:

<table>
<thead>
<tr>
<th>Host Name</th>
<th>Infrastructure Component to Install</th>
</tr>
</thead>
<tbody>
<tr>
<td>master1.example.com</td>
<td>Master (clustered using Pacemaker) and node</td>
</tr>
<tr>
<td>master2.example.com</td>
<td></td>
</tr>
<tr>
<td>master3.example.com</td>
<td></td>
</tr>
<tr>
<td>etcd1.example.com</td>
<td>etcd</td>
</tr>
<tr>
<td>etcd2.example.com</td>
<td></td>
</tr>
<tr>
<td>etcd3.example.com</td>
<td></td>
</tr>
<tr>
<td>node1.example.com</td>
<td>Node</td>
</tr>
<tr>
<td>node2.example.com</td>
<td></td>
</tr>
</tbody>
</table>

**NOTE**

When specifying multiple etcd hosts, external etcd is installed and configured. Clustering of OpenShift’s embedded etcd is not supported.

You can see these example hosts present in the [masters], [nodes], and [etcd] sections of the following example inventory file:

```bash
Example 2.5. Multiple Masters Using Pacemaker Inventory File

# Create an OSEv3 group that contains the masters, nodes, and etcd groups
[OSEv3:children]
masters
nodes
etcd

# Set variables common for all OSEv3 hosts
[OSEv3:vars]
anansible_ssh_user=root
deployment_type=openshift-enterprise

# Uncomment the following to enable htpasswd authentication; defaults to
# DenyAllPasswordIdentityProvider.
#openshift_master_identity_providers=[{"name": 'htpasswd_auth', 'login':
```
# Pacemaker high availability cluster method.
# Pacemaker HA environment must be able to self provision the
# configured VIP. For installation openshift_master_cluster_hostname
# must resolve to the configured VIP.
openshift_master_cluster_method=pacemaker
openshift_master_cluster_password=openshift_cluster
openshift_master_cluster_vip=192.168.133.25
openshift_master_cluster_public_vip=192.168.133.25
openshift_master_cluster_hostname=openshift-cluster.example.com
openshift_master_cluster_public_hostname=openshift-cluster.example.com

# override the default controller lease ttl
#osm_controller_lease_ttl=30

# host group for masters
[masters]
master1.example.com
master2.example.com
master3.example.com

# host group for etcd
[etcd]
etcd1.example.com
etcd2.example.com
etcd3.example.com

# host group for nodes, includes region info
[nodes]
master[1:3].example.com openshift_node_labels="{'region': 'infra',
'zone': 'default'}"
node1.example.com openshift_node_labels="{'region': 'primary', 'zone':
'east'}"
node2.example.com openshift_node_labels="{'region': 'primary', 'zone':
'west'}"

To use this example, modify the file to match your environment and specifications, and save it as
/etc/ansible/hosts.

Note the following when using this configuration:

- Installing multiple masters with Pacemaker requires that you configure a fencing device after running the installer.
- When specifying multiple masters, the installer handles creating and starting the HA cluster. If during that process the pcs status command indicates that an HA cluster already exists, the installer skips HA cluster configuration.

2.5.6. Running the Advanced Installation

After you have configuredAnsible by defining an inventory file in /etc/ansible/hosts, you can run the advanced installation using the following playbook:
# ansible-playbook /usr/share/ansible/openshift-ansible/playbooks/byo/config.yml

If for any reason the installation fails, before re-running the installer, see Known Issues to check for any specific instructions or workarounds.

## WARNING

The installer caches playbook configuration values for 10 minutes, by default. If you change any system, network, or inventory configuration, and then re-run the installer within that 10 minute period, the new values are not used, and the previous values are used instead. You can delete the contents of the cache, which is defined by the fact_caching_connection value in the /etc/ansible/ansible.cfg file.

## NOTE

Due to a known issue, after running the installation, if NFS volumes are provisioned for any component, the following directories might be created whether their components are being deployed to NFS volumes or not:

- /exports/logging-es
- /exports/logging-es-ops/
- /exports/metrics/
- /exports/prometheus
- /exports/prometheus-alertbuffer/
- /exports/prometheus-alertmanager/

You can delete these directories after installation, as needed.

### 2.5.7. Configuring Fencing

If you installed OpenShift using a configuration for multiple masters with Pacemaker as a load balancer, you must configure a fencing device. See Fencing: Configuring STONITH in the High Availability Add-on for Red Hat Enterprise Linux documentation for instructions, then continue to Verifying the Installation.

### 2.5.8. Verifying the Installation

After the installation completes:

1. Verify that the master is started and nodes are registered and reporting in Ready status. On the master host, run the following as root:

   # oc get nodes

<table>
<thead>
<tr>
<th>NAME</th>
<th>LABELS</th>
</tr>
</thead>
</table>


2. To verify that the web console is installed correctly, use the master host name and the console port number to access the console with a web browser.
For example, for a master host with a hostname of `master.openshift.com` and using the default port of 8443, the web console would be found at:

https://master.openshift.com:8443/console

3. Now that the install has been verified, run the following command on each master and node host to add the `atomic-openshift` packages back to the list of yum excludes on the host:

```
# atomic-openshift-excluder exclude
```

Multiple etcd Hosts

If you installed multiple etcd hosts:

1. On a etcd host, verify the etcd cluster health, substituting for the FQDNs of your etcd hosts in the following:

```
```

2. Also verify the member list is correct:

```
```

Multiple Masters Using Pacemaker

If you installed multiple masters using Pacemaker as a load balancer:

1. On a master host, determine which host is currently running as the active master:
# pcs status

2. After determining the active master, put the specified host into standby mode:

   ```bash
   # pcs cluster standby <host1_name>
   ```

3. Verify the master is now running on another host:

   ```bash
   # pcs status
   ```

4. After verifying the master is running on another node, re-enable the host on standby for normal operation by running:

   ```bash
   # pcs cluster unstandby <host1_name>
   ```

Red Hat recommends that you also verify your installation by consulting the High Availability Add-on for Red Hat Enterprise Linux documentation.

**Multiple Masters Using HAProxy**

If you installed multiple masters using HAProxy as a load balancer, browse to the following URL according to your `[lb]` section definition and check HAProxy’s status:

```
http://<lb_hostname>:9000
```

You can verify your installation by consulting the HAProxy Configuration documentation.

### 2.5.9. Adding Nodes to an Existing Cluster

After your cluster is installed, you can install additional nodes (including masters) and add them to your cluster by running the `scaleup.yml` playbook. This playbook queries the master, generates and distributes new certificates for the new nodes, then runs the configuration playbooks on the new nodes only.

This process is similar to re-running the installer in the quick installation method to add nodes, however you have more configuration options available when using the advanced method and running the playbooks directly.

You must have an existing inventory file (for example, `/etc/ansible/hosts`) that is representative of your current cluster configuration in order to run the `scaleup.yml` playbook. If you previously used the `atomic-openshift-installer` command to run your installation, you can check `~/.config/openshift/.ansible/hosts` for the last inventory file that the installer generated and use or modify that as needed as your inventory file. You must then specify the file location with `-i` when calling `ansible-playbook` later.

To add nodes to an existing cluster:

1. Ensure you have the latest playbooks by updating the `atomic-openshift-utils` package:

   ```bash
   # yum update atomic-openshift-utils
   ```

2. Edit your `/etc/ansible/hosts` file and add `new_nodes` to the `[OSEv3:children]` section:
3. Then, create a [new_nodes] section much like the existing [nodes] section, specifying host information for any new nodes you want to add. For example:

```
[nodes]
master[1:3].example.com openshift_node_labels="{'region': 'infra', 'zone': 'default'}"
node1.example.com openshift_node_labels="{'region': 'primary', 'zone': 'east'}"
node2.example.com openshift_node_labels="{'region': 'primary', 'zone': 'west'}"

[new_nodes]
node3.example.com openshift_node_labels="{'region': 'primary', 'zone': 'west'}"
```

See Configuring Host Variables for more options.

4. Now run the `scaleup.yml` playbook. If your inventory file is located somewhere other than the default `/etc/ansible/hosts`, specify the location with the `-i` option:

For additional nodes:

```bash
# ansible-playbook [-i /path/to/file] \
/usr/share/ansible/openshift-ansible/playbooks/byo/openshift-node/scaleup.yml
```

For additional masters:

```bash
# ansible-playbook [-i /path/to/file] \
/usr/share/ansible/openshift-ansible/playbooks/byo/openshift-master/scaleup.yml
```

5. After the playbook completes successfully, verify the installation.

6. Finally, move any hosts you had defined in the [new_nodes] section up into the [nodes] section (but leave the [new_nodes] section definition itself in place) so that subsequent runs using this inventory file are aware of the nodes but do not handle them as new nodes. For example:

```
[nodes]
master[1:3].example.com openshift_node_labels="{'region': 'infra', 'zone': 'default'}"
node1.example.com openshift_node_labels="{'region': 'primary', 'zone': 'east'}"
node2.example.com openshift_node_labels="{'region': 'primary', 'zone': 'west'}"
node3.example.com openshift_node_labels="{'region': 'primary', 'zone': 'west'}"
```

```
[new_nodes]
```
2.5.10. Uninstalling OpenShift Enterprise

You can uninstall OpenShift Enterprise hosts in your cluster by running the `uninstall.yml` playbook. This playbook deletes OpenShift Enterprise content installed by Ansible, including:

- Configuration
- Containers
- Default templates and image streams
- Images
- RPM packages

The playbook will delete content for any hosts defined in the inventory file that you specify when running the playbook. If you want to uninstall OpenShift Enterprise across all hosts in your cluster, run the playbook using the inventory file you used when installing OpenShift Enterprise initially or ran most recently:

```
# ansible-playbook [-i /path/to/file] \
/usr/share/ansible/openshift-ansible/playbooks/adhoc/uninstall.yml
```

2.5.10.1. Uninstalling Nodes

You can also uninstall node components from specific hosts using the `uninstall.yml` playbook while leaving the remaining hosts and cluster alone:

WARNING

This method should only be used when attempting to uninstall specific node hosts and not for specific masters or etcd hosts, which would require further configuration changes within the cluster.

1. First follow the steps in Deleting Nodes to remove the node object from the cluster, then continue with the remaining steps in this procedure.

2. Create a different inventory file that only references those hosts. For example, to only delete content from one node:

```
[OSEv3:children]
nodes

[OSEv3:vars]
ansible_ssh_user=root
deployment_type=openshift-enterprise

[nodes]
ode3.example.com openshift_node_labels="{'region': 'primary', 'zone': 'west'}"
```
1. Only include the sections that pertain to the hosts you are interested in uninstalling.

2. Only include hosts that you want to uninstall.

3. Specify that new inventory file using the `-i` option when running the `uninstall.yml` playbook:

```bash
# ansible-playbook -i /path/to/new/file \
/usr/share/ansible/openshift-ansible/playbooks/adhoc/uninstall.yml
```

When the playbook completes, all OpenShift Enterprise content should be removed from any specified hosts.

### 2.5.11. Known Issues

The following are known issues for specified installation configurations.

#### Multiple Masters

- On failover, it is possible for the controller manager to overcorrect, which causes the system to run more pods than what was intended. However, this is a transient event and the system does correct itself over time. See [https://github.com/GoogleCloudPlatform/kubernetes/issues/10030](https://github.com/GoogleCloudPlatform/kubernetes/issues/10030) for details.

- On failure of the Ansible installer, you must start from a clean operating system installation. If you are using virtual machines, start from a fresh image. If you are use bare metal machines:
  1. Run the following on a master host with Pacemaker:
     ```bash
     # pcs cluster destroy --all
     ```
  2. Then, run the following on all node hosts:
     ```bash
     # yum -y remove openshift openshift-* etcd docker
     # rm -rf /etc/origin /var/lib/openshift /etc/etcd /var/lib/etcd /etc/sysconfig/atomic-openshift* /etc/sysconfig/docker* /root/.kube/config /etc/ansible/facts.d /usr/share/openshift
     ```

### 2.5.12. What’s Next?

Now that you have a working OpenShift instance, you can:

- Configure authentication; by default, authentication is set to Deny All.
- Deploy an integrated Docker registry.
- Deploy a router.

### 2.6. DISCONNECTED INSTALLATION

#### 2.6.1. Overview
Frequently, portions of a datacenter may not have access to the Internet, even via proxy servers. Installing OpenShift Enterprise in these environments is considered a disconnected installation.

An OpenShift Enterprise disconnected installation differs from a regular installation in two primary ways:

- The OpenShift Enterprise software channels and repositories are not available via Red Hat’s content distribution network.
- OpenShift Enterprise uses several containerized components. Normally, these images are pulled directly from Red Hat’s Docker registry. In a disconnected environment, this is not possible.

A disconnected installation ensures the OpenShift Enterprise software is made available to the relevant servers, then follows the same installation process as a standard connected installation. This topic additionally details how to manually download the Docker images and transport them onto the relevant servers.

Once installed, in order to use OpenShift Enterprise, you will need source code in a source control repository (for example, Git). This topic assumes that an internal Git repository is available that can host source code and this repository is accessible from the OpenShift Enterprise nodes. Installing the source control repository is outside the scope of this document.

Also, when building applications in OpenShift Enterprise, your build may have some external dependencies, such as a Maven Repository or Gem files for Ruby applications. For this reason, and because they might require certain tags, many of the Quickstart templates offered by OpenShift Enterprise may not work on a disconnected environment. However, while Red Hat Docker images try to reach out to external repositories by default, you can configure OpenShift Enterprise to use your own internal repositories. For the purposes of this document, we assume that such internal repositories already exist and are accessible from the OpenShift Enterprise nodes hosts. Installing such repositories is outside the scope of this document.

**NOTE**

You can also have a Red Hat Satellite server that provides access to Red Hat content via an intranet or LAN. For environments with Satellite, you can synchronize the OpenShift Enterprise software onto the Satellite for use with the OpenShift Enterprise servers.

Red Hat Satellite 6.1 also introduces the ability to act as a Docker registry, and it can be used to host the OpenShift Enterprise containerized components. Doing so is outside of the scope of this document.

### 2.6.2. Prerequisites

This document assumes that you understand OpenShift’s overall architecture and that you have already planned out what the topology of your environment will look like.

### 2.6.3. Required Software and Components

In order to pull down the required software repositories and Docker images, you will need a Red Hat Enterprise Linux (RHEL) 7 server with access to the Internet and at least 100GB of additional free space. All steps in this section should be performed on the Internet-connected server as the root system user.

#### 2.6.3.1. Syncing Repositories

Before you sync with the required repositories, you may need to import the appropriate GPG key:
# rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release

If the key is not imported, the indicated package is deleted after syncing the repository.

To sync the required repositories:

1. Register the server with the Red Hat Customer Portal. You must use the login and password associated with the account that has access to the OpenShift Enterprise subscriptions:

   # subscription-manager register

2. Attach to a subscription that provides OpenShift Enterprise channels. You can find the list of available subscriptions using:

   # subscription-manager list --available

   Then, find the pool ID for the subscription that provides OpenShift Enterprise, and attach it:

   # subscription-manager attach --pool=<pool_id>
   # subscription-manager repos --disable="**"
   # subscription-manager repos \      --enable="rhel-7-server-rpms" \      --enable="rhel-7-server-extras-rpms" \      --enable="rhel-7-server-ose-3.1-rpms"

3. The **yum-utils** command provides the **reposync** utility, which lets you mirror yum repositories, and **createrepo** can create a usable **yum** repository from a directory:

   # yum -y install yum-utils createrepo docker git

   You will need up to 110GB of free space in order to sync the software. Depending on how restrictive your organization’s policies are, you could re-connect this server to the disconnected LAN and use it as the repository server. You could use USB-connected storage and transport the software to another server that will act as the repository server. This topic covers these options.

4. Make a path to where you want to sync the software (either locally or on your USB or other device):

   # mkdir -p </path/to/repos>

5. Sync the packages and create the repository for each of them. You will need to modify the command for the appropriate path you created above:

   # for repo in \      rhel-7-server-rpms rhel-7-server-extras-rpms \      rhel-7-server-ose-3.1-rpms \      do \        reposync --gpgcheck -lm --repoid=${repo} --download_path=/path/to/repos \        createrepo -v </path/to/repos/>${repo} -o </path/to/repos/>${repo} \      done
2.6.3.2. Syncing Images

To sync the Docker images:

1. Start the Docker daemon:

   ```
   # systemctl start docker
   ```

2. Pull all of the required OpenShift Enterprise containerized components:

   ```
   # docker pull registry.access.redhat.com/openshift3/ose-haproxy-router:v3.1.1.11
   # docker pull registry.access.redhat.com/openshift3/ose-deployer:v3.1.1.11
   # docker pull registry.access.redhat.com/openshift3/ose-sti-builder:v3.1.1.11
   # docker pull registry.access.redhat.com/openshift3/ose-docker-builder:v3.1.1.11
   # docker pull registry.access.redhat.com/openshift3/ose-pod:v3.1.1.11
   # docker pull registry.access.redhat.com/openshift3/ose-docker-registry:v3.1.1.11
   ```

3. Pull all of the required OpenShift Enterprise containerized components for the additional centralized log aggregation and metrics aggregation components:

   ```
   # docker pull registry.access.redhat.com/openshift3/logging-deployment
   # docker pull registry.access.redhat.com/openshift3/logging-elasticsearch
   # docker pull registry.access.redhat.com/openshift3/logging-kibana
   # docker pull registry.access.redhat.com/openshift3/logging-fluentd
   # docker pull registry.access.redhat.com/openshift3/logging-auth-proxy
   # docker pull registry.access.redhat.com/openshift3/metrics-deployer
   # docker pull registry.access.redhat.com/openshift3/metrics-hawkular-metrics
   # docker pull registry.access.redhat.com/openshift3/metrics-cassandra
   # docker pull registry.access.redhat.com/openshift3/metrics-heapster
   ```

4. Pull the Red Hat-certified Source-to-Image (S2I) builder images that you intend to use in your OpenShift environment. You can pull the following images:

   - jboss-eap70-openshift
   - jboss-amq-62
   - jboss-datagrid65-openshift
   - jboss-decisionserver62-openshift
   - jboss-eap64-openshift
   - jboss-eap70-openshift
- jboss-webserver30-tomcat7-openshift
- jboss-webserver30-tomcat8-openshift
- mongodb
- mysql
- nodejs
- perl
- php
- postgresql
- python
- redhat-sso70-openshift
- ruby

Make sure to indicate the correct tag specifying the desired version number. For example, to pull both the previous and latest version of the Tomcat image:

```
# docker pull \
registry.access.redhat.com/jboss-webserver-3/webserver30-tomcat7-\openshift:latest
# docker pull \
registry.access.redhat.com/jboss-webserver-3/webserver30-tomcat7-\openshift:1.1
```

### 2.6.3.3. Preparing Images for Export

Docker images can be exported from a system by first saving them to a tarball and then transporting them:

1. Make and change into a repository home directory:

   ```
   # mkdir </path/to/repos/images>
   # cd </path/to/repos/images>
   ```

2. Export the OpenShift Enterprise containerized components:

   ```
   ```

3. If you synchronized the metrics and log aggregation images, export:

   ```
   # docker save -o ose3-logging-metrics-images.tar \\n   registry.access.redhat.com/openshift3/logging-deployment \\n   ```
4. Export the S2I builder images that you synced in the previous section. For example, if you synced only the Tomcat image:

```
# docker save -o ose3-builder-images.tar \
   registry.access.redhat.com/jboss-webserver-3/webserver30-tomcat7-openshift:latest \
   registry.access.redhat.com/jboss-webserver-3/webserver30-tomcat7-openshift:1.1
```

### 2.6.4. Repository Server

During the installation (and for later updates, should you so choose), you will need a webserver to host the repositories. RHEL 7 can provide the Apache webserver.

**Option 1: Re-configuring as a Web server**

If you can re-connect the server where you synchronized the software and images to your LAN, then you can simply install Apache on the server:

```
# yum install httpd
```

Skip to Placing the Software.

**Option 2: Building a Repository Server**

If you need to build a separate server to act as the repository server, install a new RHEL 7 system with at least 110GB of space. On this repository server during the installation, make sure you select the Basic Web Server option.

#### 2.6.4.1. Placing the Software

1. If necessary, attach the external storage, and then copy the repository files into Apache's root folder. Note that the below copy step (`cp -a`) should be substituted with move (`mv`) if you are repurposing the server you used to sync:

   ```
   # cp -a /path/to/repos /var/www/html/
   # chmod -R +r /var/www/html/repos
   # restorecon -vR /var/www/html
   ```

2. Add the firewall rules:

   ```
   # firewall-cmd --permanent --add-service=http
   # firewall-cmd --reload
   ```

3. Enable and start Apache for the changes to take effect:
2.6.5. OpenShift Enterprise Systems

2.6.5.1. Building Your Hosts

At this point you can perform the initial creation of the hosts that will be part of the OpenShift Enterprise environment. It is recommended to use the latest version of RHEL 7 and to perform a minimal installation. You will also want to pay attention to the other OpenShift Enterprise-specific prerequisites.

Once the hosts are initially built, the repositories can be set up.

2.6.5.2. Connecting the Repositories

On all of the relevant systems that will need OpenShift Enterprise software components, create the required repository definitions. Place the following text in the `/etc/yum.repos.d/ose.repo` file, replacing `<server_IP>` with the IP or host name of the Apache server hosting the software repositories:

```
[rhel-7-server-rpms]
name=rhel-7-server-rpms
baseurl=http://<server_IP>/repos/rhel-7-server-rpms
enabled=1
gpgcheck=0
[rhel-7-server-extras-rpms]
name=rhel-7-server-extras-rpms
baseurl=http://<server_IP>/repos/rhel-7-server-extras-rpms
enabled=1
gpgcheck=0
[rhel-7-server-ose-3.1-rpms]
name=rhel-7-server-ose-3.1-rpms
baseurl=http://<server_IP>/repos/rhel-7-server-ose-3.1-rpms
enabled=1
gpgcheck=0
```

2.6.5.3. Host Preparation

At this point, the systems are ready to continue to be prepared following the OpenShift Enterprise documentation.

Skip the section titled Registering the Hosts and start with Managing Packages.

2.6.6. Installing OpenShift Enterprise

2.6.6.1. Importing OpenShift Enterprise Containerized Components

To import the relevant components, securely copy the images from the connected host to the individual OpenShift Enterprise hosts:

```
# scp /var/www/html/repos/images/ose3-images.tar root@<openshift_host_name>: 
# ssh root@<openshift_host_name> "docker load -i ose3-images.tar"
```
If you prefer, you could use `wget` on each OpenShift Enterprise host to fetch the tar file, and then perform the Docker import command locally. Perform the same steps for the metrics and logging images, if you synchronized them.

On the host that will act as an OpenShift Enterprise master, copy and import the builder images:

```bash
# scp /var/www/html/images/ose3-builder-images.tar root@<openshift_master_host_name>:
# ssh root@<openshift_master_host_name> "docker load -i ose3-builder-images.tar"
```

### 2.6.6.2. Running the OpenShift Enterprise Installer

You can now choose to follow the quick or advanced OpenShift Enterprise installation instructions in the documentation.

### 2.6.6.3. Creating the Internal Docker Registry

You now need to create the internal Docker registry.

### 2.6.7. Post-Installation Changes

In one of the previous steps, the S2I images were imported into the Docker daemon running on one of the OpenShift Enterprise master hosts. In a connected installation, these images would be pulled from Red Hat's registry on demand. Since the Internet is not available to do this, the images must be made available in another Docker registry.

OpenShift Enterprise provides an internal registry for storing the images that are built as a result of the S2I process, but it can also be used to hold the S2I builder images. The following steps assume you did not customize the service IP subnet (172.30.0.0/16) or the Docker registry port (5000).

#### 2.6.7.1. Re-tagging S2I Builder Images

1. On the master host where you imported the S2I builder images, obtain the service address of your Docker registry that you installed on the master:

   ```bash
   # export REGISTRY=$(oc get service docker-registry -t '{{{.spec.clusterIP}}}{{"\n"}}')
   ```

2. Next, tag all of the builder images that you synced and exported before pushing them into the OpenShift Enterprise Docker registry. For example, if you synced and exported only the Tomcat image:

   ```bash
   # docker tag \
   registry.access.redhat.com/jboss-webserver-3/webserver30-tomcat7-openshift:1.1 \n   $REGISTRY:5000/openshift/webserver30-tomcat7-openshift:1.1 \
   # docker tag \
   registry.access.redhat.com/jboss-webserver-3/webserver30-tomcat7-openshift:latest \n   $REGISTRY:5000/openshift/webserver30-tomcat7-openshift:latest \
   # docker tag \
   ```
2.6.7.2. Creating an Administrative User

Pushing the Docker images into OpenShift Enterprise’s Docker registry requires a user with `cluster-admin` privileges. Because the default OpenShift system administrator does not have a standard authorization token, they cannot be used to log in to the Docker registry.

To create an administrative user:

1. Create a new user account in the authentication system you are using with OpenShift Enterprise. For example, if you are using local `htpasswd`-based authentication:

   ```bash
   # htpasswd -b /etc/openshift/openshift-passwd <admin_username> <password>
   ```

2. The external authentication system now has a user account, but a user must log in to OpenShift Enterprise before an account is created in the internal database. Log in to OpenShift Enterprise for this account to be created. This assumes you are using the self-signed certificates generated by OpenShift Enterprise during the installation:

   ```bash
   # oc login --certificate-authority=/etc/origin/master/ca.crt \ 
   -u <admin_username> https://<openshift_master_host>:8443
   ```

3. Get the user’s authentication token:

   ```bash
   # MYTOKEN=$(oc whoami -t)
   # echo $MYTOKEN
   iwo7hc4XilD2KOLL4V1O55ExH2V1PmLD-W2-JOd6Fko
   ```

2.6.7.3. Modifying the Security Policies

1. Using `oc login` switches to the new user. Switch back to the OpenShift Enterprise system administrator in order to make policy changes:

   ```bash
   # oc login -u system:admin
   ```

2. In order to push images into the OpenShift Enterprise Docker registry, an account must have the `image-builder` security role. Add this to your OpenShift Enterprise administrative user:

   ```bash
   # oadm policy add-role-to-user system:image-builder <admin_username>
   ```

3. Next, add the administrative role to the user in the `openshift` project. This allows the administrative user to edit the `openshift` project, and, in this case, push the Docker images:

   ```bash
   # oadm policy add-role-to-user admin <admin_username> -n openshift
   ```

2.6.7.4. Editing the Image Stream Definitions

The `openshift` project is where all of the image streams for builder images are created by the installer.
They are loaded by the installer from the /usr/share/openshift/examples directory. Change all of the definitions by deleting the image streams which had been loaded into OpenShift Enterprise’s database, then re-create them:

1. Delete the existing image streams:
   
   ```bash
   # oc delete is -n openshift --all
   ```

2. Make a backup of the files in /usr/share/openshift/examples/ if you desire. Next, edit the file image-streams-rhel7.json in the /usr/share/openshift/examples/image-streams folder. You will find an image stream section for each of the builder images. Edit the spec stanza to point to your internal Docker registry. For example, change:

   ```json
   "spec": {
     "dockerImageRepository": "registry.access.redhat.com/rhscl/mongodb-26-rhel7",
   }
   ```

   to:

   ```json
   "spec": {
     "dockerImageRepository": "172.30.69.44:5000/openshift/mongodb-26-rhel7",
   }
   ```

   In the above, the repository name was changed from rhscl to openshift. You will need to ensure the change, regardless of whether the repository is rhscl, openshift3, or another directory. Every definition should have the following format:

   `<registry_ip>:5000/openshift/<image_name>`

   Repeat this change for every image stream in the file. Ensure you use the correct IP address that you determined earlier. When you are finished, save and exit. Repeat the same process for the JBoss image streams in the /usr/share/openshift/examples/xpaas-streams/jboss-image-streams.json file.

3. Load the updated image stream definitions:

   ```bash
   # oc create -f /usr/share/openshift/examples/image-streams/image-streams-rhel7.json -n openshift
   # oc create -f /usr/share/openshift/examples/xpaas-streams/jboss-image-streams.json -n openshift
   ```

### 2.6.7.5. Loading the Docker Images

At this point the system is ready to load the Docker images.

1. Log in to the Docker registry using the token and registry service IP obtained earlier:

   ```bash
   # docker login -u adminuser -e mailto:adminuser@abc.com \
   -p $MYTOKEN $REGISTRY:5000
   ```

2. Push the Docker images:
3. Verify that all the image streams now have the tags populated:

```
# oc get imagestreams -n openshift
NAME                                 DOCKER REPO
TAGS                                     UPDATED
jboss-webserver30-tomcat7-openshift  $REGISTRY/jboss-webserver-
3/webserver30-jboss-tomcat7-openshift  1.1,1.1-2,1.1-6 + 2 more...
2 weeks ago
...
```

### 2.6.8. Installing a Router

At this point, the OpenShift Enterprise environment is almost ready for use. It is likely that you will want to install and configure a router.

### 2.7. DEPLOYING A DOCKER REGISTRY

#### 2.7.1. Overview

OpenShift can build Docker images from your source code, deploy them, and manage their lifecycle. To enable this, OpenShift provides an internal, integrated Docker registry that can be deployed in your OpenShift environment to locally manage images.

#### 2.7.2. Deploying the Registry

To deploy the integrated Docker registry, use the `oadm registry` command as a user with cluster administrator privileges. For example:

```
$ oadm registry --config=/etc/origin/master/admin.kubeconfig \ 1
    --credentials=/etc/origin/master/openshift-registry.kubeconfig \ 2
    --images='registry.access.redhat.com/openshift3/ose-
    ${component}:${version}' 3
```

1. **--config** is the path to the CLI configuration file for the cluster administrator.
2. **--credentials** is the path to the CLI configuration file for the openshift-registry. Required to pull the correct image for OpenShift Enterprise.
3. **--images** is the path to the CLI configuration file for the openshift-registry.

This creates a service and a deployment configuration, both called `docker-registry`. Once deployed successfully, a pod is created with a name similar to `docker-registry-1-ctpy9`.

Use **--selector** to deploy the registry to any node(s) that match a specified node label:
$ oadm registry <registry_name> --replicas=<number> --selector=<label> \
   --service-account=registry

For example, if you want to create a registry named registry and have it placed on a node labeled with region=infra:

$ oadm registry registry --replicas=1 --selector='region=infra' \
   --service-account=registry

To see a full list of options that you can specify when creating the registry:

$ oadm registry --help

2.7.2.1. Storage for the Registry

The registry stores Docker images and metadata. If you simply deploy a pod with the registry, it uses an ephemeral volume that is destroyed if the pod exits. Any images anyone has built or pushed into the registry would disappear.

2.7.2.1.1. Production Use

For production use, attach a remote volume or define and use the persistent storage method of your choice.

For example, to use an existing persistent volume claim:

$ oc volume deploymentconfigs/docker-registry --add --name=registry-storage -t pvc \
   --claim-name=<pvc_name> --overwrite

Or, to attach an existing NFS volume to the registry:

$ oc volume deploymentconfigs/docker-registry \
   --add --overwrite --name=registry-storage --mount-path=/registry \n   --source='{"nfs": { "server": "<fqdn>", "path": "/path/to/export"}}'

NOTE

See Known Issues if using a scaled registry with a shared NFS volume.

2.7.2.1.2. Non-Production Use

For non-production use, you can use the --mount-host=<path> option to specify a directory for the registry to use for persistent storage. The registry volume is then created as a host-mount at the specified <path>.
The **--mount-host** option mounts a directory from the node on which the registry container lives. If you scale up the docker-registry deployment configuration, it is possible that your registry pods and containers will run on different nodes, which can result in two or more registry containers, each with its own local storage. This will lead to unpredictable behavior, as subsequent requests to pull the same image repeatedly may not always succeed, depending on which container the request ultimately goes to.

The **--mount-host** option requires that the registry container run in privileged mode. This is automatically enabled when you specify **--mount-host**. However, not all pods are allowed to run privileged containers by default. If you still want to use this option, create the registry and specify that it use the registry service account that was created during installation:

```bash
$ oadm registry --service-account=registry \
   --config=/etc/origin/master/admin.kubeconfig \
   --credentials=/etc/origin/master/openshift-registry.kubeconfig \
   --images='registry.access.redhat.com/openshift3/ose-\n   ${component}:${version}' \
   --mount-host=<path>
```

The Docker registry pod runs as user **1001**. This user must be able to write to the host directory. You may need to change directory ownership to user ID **1001** with this command:

```bash
$ sudo chown 1001:root <path>
```

### 2.7.2.2. Maintaining the Registry IP Address

OpenShift refers to the integrated registry by its service IP address, so if you decide to delete and recreate the docker-registry service, you can ensure a completely transparent transition by arranging to re-use the old IP address in the new service. If a new IP address cannot be avoided, you can minimize cluster disruption by rebooting only the masters.

#### Re-using the Address

To re-use the IP address, you must save the IP address of the old docker-registry service prior to deleting it, and arrange to replace the newly assigned IP address with the saved one in the new docker-registry service.

1. Make a note of the **ClusterIP** for the service:

   ```bash
   $ oc get svc/docker-registry -o yaml | grep clusterIP:
   ```

2. Delete the service:

   ```bash
   $ oc delete svc/docker-registry dc/docker-registry
   ```

3. Create the registry definition in **registry.yaml**, replacing **<options>** with, for example, those used in step 3 of the instructions in the **Non-Production Use** section:
$ oadm registry <options> -o yaml > registry.yaml

4. Edit `registry.yaml`, find the Service there, and change its ClusterIP to the address noted in step 1.

5. Create the registry using the modified `registry.yaml`:

   $ oc create -f registry.yaml

Rebooting the Masters

If you are unable to re-use the IP address, any operation that uses a pull specification that includes the old IP address will fail. To minimize cluster disruption, you must reboot the masters:

   # systemctl restart atomic-openshift-master

This ensures that the old registry URL, which includes the old IP address, is cleared from the cache.

NOTE

We recommend against rebooting the entire cluster because that incurs unnecessary downtime for pods and does not actually clear the cache.

2.7.3. Viewing Logs

To view the logs for the Docker registry, run the `oc logs` indicating the desired pod:

   $ oc logs docker-registry-1-da73t


2.7.4. File Storage

Tag and image metadata is stored in OpenShift, but the registry stores layer and signature data in a volume that is mounted into the registry container at `/registry`. As `oc exec` does not work on privileged containers, to view a registry’s contents you must manually SSH into the node housing the registry pod’s container, then run `docker exec` on the container itself:

1. List the current pods to find the pod name of your Docker registry:

   # oc get pods
Then, use `oc describe` to find the host name for the node running the container:

```
# oc describe pod <pod_name>
```

2. Log into the desired node:

```
# ssh node.example.com
```

3. List the running containers on the node host and identify the container ID for the Docker registry:

```
# docker ps | grep ose-docker-registry
```

4. List the registry contents using the `docker exec` command:

```
# docker exec -it 4c01db0b339c find /registry
```

```
/registry/docker
/registry/docker/registry
/registry/docker/registry/v2
/registry/docker/registry/v2/blobs
/registry/docker/registry/v2/blobs/sha256
/registry/docker/registry/v2/blobs/sha256/ed
/registry/docker/registry/v2/blobs/sha256/ed/ede17b139a271d6b1331ca3d83c648c24f92c83e6f89d95ac6c34e751111810
/registry/docker/registry/v2/blobs/sha256/ed/ede17b139a271d6b1331ca3d83c648c24f92c83e6f89d95ac6c34e751111810/data
/registry/docker/registry/v2/blobs/sha256/a3
/registry/docker/registry/v2/blobs/sha256/a3/a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4
/registry/docker/registry/v2/blobs/sha256/a3/a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4/data
/registry/docker/registry/v2/repositories
/registry/docker/registry/v2/repositories/p1
/registry/docker/registry/v2/repositories/p1/pause
/registry/docker/registry/v2/repositories/p1/pause/_manifests
/registry/docker/registry/v2/repositories/p1/pause/_manifests/revisions
/registry/docker/registry/v2/repositories/p1/pause/_manifests/revisions/sha256
/registry/docker/registry/v2/repositories/p1/pause/_manifests/revisions/sha256/e9a2ac6418981897b399d3709f1b4a6d2723cd38a4909215ce2752a5c068b1cf
/registry/docker/registry/v2/repositories/p1/pause/_manifests/revisions/sha256/e9a2ac6418981897b399d3709f1b4a6d2723cd38a4909215ce2752a5c068b1cf/signatures
/registry/docker/registry/v2/repositories/p1/pause/_manifests/revisions/sha256/e9a2ac6418981897b399d3709f1b4a6d2723cd38a4909215ce2752a5c068b1cf/signatures/sha256
/registry/docker/registry/v2/repositories/p1/pause/_manifests/revisions/sha256/e9a2ac6418981897b399d3709f1b4a6d2723cd38a4909215ce2752a5c068b1cf
/registry/docker/registry/v2/repositories/p1/pause/_manifests/revisions/sha256/e9a2ac6418981897b399d3709f1b4a6d2723cd38a4909215ce2752a5c068b1cf/signatures/sha256
/registry/docker/registry/v2/repositories/p1/pause/_manifests/revisions/sha256/e9a2ac6418981897b399d3709f1b4a6d2723cd38a4909215ce2752a5c068b1cf
/registry/docker/registry/v2/repositories/p1/pause/_manifests/revisions/sha256/e9a2ac6418981897b399d3709f1b4a6d2723cd38a4909215ce2752a5c
```
This directory stores all layers and signatures as blobs.

This file contains the blob’s contents.

This directory stores all the image repositories.

This directory is for a single image repository `p1/pause`.

This directory contains signatures for a particular image manifest revision.

This file contains a reference back to a blob (which contains the signature data).

This directory contains any layers that are currently being uploaded and staged for the given repository.

This directory contains links to all the layers this repository references.

This file contains a reference to a specific layer that has been linked into this repository via an image.

### 2.7.5. Accessing the Registry Directly

For advanced usage, you can access the registry directly to invoke `docker` commands. This allows you to push images to or pull them from the integrated registry directly using operations like `docker push` or `docker pull`. To do so, you must be logged in to the registry using the `docker login` command. The operations you can perform depend on your user permissions, as described in the following sections.

#### 2.7.5.1. User Prerequisites

To access the registry directly, the user that you use must satisfy the following, depending on your intended usage:

- For any direct access, you must have a regular user, if one does not already exist, for your preferred identity provider. A regular user can generate an access token required for logging in.
to the registry. **System users**, such as **system:admin**, cannot obtain access tokens and, therefore, cannot access the registry directly.

For example, if you are using **HTPASSwd** authentication, you can create one using the following command:

```
# htpasswd /etc/origin/openshift-htpasswd <user_name>
```

- The user must have the **system:registry** role. To add this role:

```
# oadm policy add-role-to-user system:registry <user_name>
```

- Have the **admin** role for the project associated with the Docker operation. For example, if accessing images in the global **openshift** project:

```
$ oadm policy add-role-to-user admin <user_name> -n openshift
```

- For writing or pushing images, for example when using the **docker push** command, the user must have the **system:image-builder** role. To add this role:

```
$ oadm policy add-role-to-user system:image-builder <user_name>
```

For more information on user permissions, see **Managing Role Bindings**.

### 2.7.5.2. Logging in to the Registry

**NOTE**

Ensure your user satisfies the prerequisites for accessing the registry directly.

To log in to the registry directly:

1. Ensure you are logged in to OpenShift as a **regular user**:

```
$ oc login
```

2. Get your access token:

```
$ oc whoami -t
```

3. Log in to the Docker registry:

```
$ docker login -u <username> -e <any_email_address> -p <token_value> <registry_ip>:<port>
```

### 2.7.5.3. Pushing and Pulling Images

After **logging in to the registry**, you can perform **docker pull** and **docker push** operations against your registry.
IMPORTANT

You can pull arbitrary images, but if you have the system:registry role added, you can only push images to the registry in your project.

In the following examples, we use:

<table>
<thead>
<tr>
<th>Component</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;registry_ip&gt;</td>
<td>172.30.124.220</td>
</tr>
<tr>
<td>&lt;port&gt;</td>
<td>5000</td>
</tr>
<tr>
<td>&lt;project&gt;</td>
<td>openshift</td>
</tr>
<tr>
<td>&lt;image&gt;</td>
<td>busybox</td>
</tr>
<tr>
<td>&lt;tag&gt;</td>
<td>omitted (defaults to latest)</td>
</tr>
</tbody>
</table>

1. Pull an arbitrary image:

   ```
   $ docker pull docker.io/busybox
   ```

2. Tag the new image with the form `<registry_ip>:<port>/<project>/<image>`. The project name must appear in this pull specification for OpenShift to correctly place and later access the image in the registry.

   ```
   $ docker tag docker.io/busybox 172.30.124.220:5000/openshift/busybox
   ```

    **NOTE**

   Your regular user must have the system:image-builder role for the specified project, which allows the user to write or push an image. Otherwise, the docker push in the next step will fail. To test, you can create a new project to push the busybox image.

3. Push the newly-tagged image to your registry:

   ```
   $ docker push 172.30.124.220:5000/openshift/busybox
   ...  
cf2616975b4a: Image successfully pushed
   Digest: sha256:3662dd821983bc4326bee12caec61367e7fb6f6a3ee547cbbaf98f77403ca b55
   ```

### 2.7.6. Securing the Registry

Optionally, you can secure the registry so that it serves traffic via TLS:

1. Deploy the registry.
2. Fetch the service IP and port of the registry:

```
$ oc get svc/docker-registry
NAME            LABELS
SELECTOR         IP(S)            PORT(S)
docker-registry  docker-registry=default                   docker-
registry=default   172.30.124.220   5000/TCP
```

3. You can use an existing server certificate, or create a key and server certificate valid for specified IPs and host names, signed by a specified CA. To create a server certificate for the registry service IP and the `docker-registry.default.svc.cluster.local` host name:

```
$ oadm ca create-server-cert \
    --signer-cert=/etc/origin/master/ca.crt \
    --signer-key=/etc/origin/master/ca.key \
    --signer-serial=/etc/origin/master/ca.serial.txt \
    --hostnames='docker-
    registry.default.svc.cluster.local,172.30.124.220' \
    --cert=/etc/secrets/registry.crt \
    --key=/etc/secrets/registry.key
```

4. Create the secret for the registry certificates:

```
$ oc secrets new registry-secret \
    /etc/secrets/registry.crt \
    /etc/secrets/registry.key
```

5. Add the secret to the registry pod’s service accounts (including the `default` service account):

```
$ oc secrets add serviceaccounts/registry secrets/registry-secret
$ oc secrets add serviceaccounts/default secrets/registry-secret
```

6. Add the secret volume to the registry deployment configuration:

```
$ oc volume dc/docker-registry --add --type=secret \
    --secret-name=registry-secret -m /etc/secrets
```

7. Enable TLS by adding the following environment variables to the registry deployment configuration:

```
$ oc env dc/docker-registry \
    REGISTRY_HTTP_TLS_CERTIFICATE=/etc/secrets/registry.crt \
    REGISTRY_HTTP_TLS_KEY=/etc/secrets/registry.key
```

See more details on [overriding registry options](#).

8. Update the scheme used for the registry’s liveness probe from HTTP to HTTPS:

```
$ oc patch dc/docker-registry --api-version=v1 -p '{"spec": 
    {"template": {"spec": {"containers": [{
        "name":"registry",
        "livenessProbe": {"httpGet": {"scheme":"HTTPS"}}
    ]}}}}}'
```
9. Validate the registry is running in TLS mode. Wait until the `docker-registry` pod status changes to `Running` and verify the Docker logs for the registry container. You should find an entry for listening on :5000, tls.

```
$ oc get pods
POD                       IP           CONTAINER(S)   IMAGE(S)
HOST                           LABELS
STATUS    CREATED    MESSAGE
docker-registry-1-da73t   172.17.0.1   openshiftdev.local/127.0.0.1   deployment=docker-registry-
4,deploymentconfig=docker-registry,docker-registry=default   Running
38 hours
```

```
$ oc log docker-registry-1-da73t | grep tls
time="2015-05-27T05:05:53Z" level=info msg="listening on :5000, tls"
instance.id=deeba528-c478-41f5-b751-dc48e4935fc2
```

10. Copy the CA certificate to the Docker certificates directory. This must be done on all nodes in the cluster:

```
$ dcertsdir=/etc/docker/certs.d
$ destdir_addr=$dcertsdir/172.30.124.220:5000
$ destdir_name=$dcertsdir/docker-registry.default.svc.cluster.local:5000

$ sudo mkdir -p $destdir_addr $destdir_name
$ sudo cp ca.crt $destdir_addr
$ sudo cp ca.crt $destdir_name
```

The `ca.crt` file is a copy of `/etc/origin/master/ca.crt` on the master.

11. Remove the `--insecure-registry` option only for this particular registry in the `/etc/sysconfig/docker` file. Then, reload the daemon and restart the `docker` service to reflect this configuration change:

```
$ sudo systemctl daemon-reload
$ sudo systemctl restart docker
```

12. Validate the `docker` client connection. Running `docker push` to the registry or `docker pull` from the registry should succeed. Make sure you have logged into the registry.

```
$ docker tag|push <registry/image> <internal_registry/project/image>
```

For example:

```
$ docker pull busybox
$ docker tag docker.io/busybox 172.30.124.220:5000/openshift/busybox
$ docker push 172.30.124.220:5000/openshift/busybox
...
cf2616975b4a: Image successfully pushed
Digest: sha256:3662dd821983bc4326bee12caee61367e7fb6f6a3ee547cba9f8f77403cab55
```
2.7.7. Advanced: Overriding the Registry Configuration

You can override the integrated registry’s default configuration, found by default at /config.yml in a running registry’s container, with your own custom configuration. See the upstream registry documentation’s Registry Configuration Reference for the full list of available options.

To enable managing the registry configuration file directly, it is recommended that the configuration file be mounted as a secret volume:

1. Deploy the registry.
2. Edit the registry configuration file locally as needed. The initial YAML file deployed on the registry is provided below:

   ```yaml
   version: 0.1
   log:
     level: debug
   http:
     addr: :5000
   storage:
     cache:
       layerinfo: inmemory
     filesystem:
       rootdirectory: /registry
   delete:
     enabled: true
   auth:
     openshift:
       realm: openshift
   middleware:
     repository:
       - name: openshift
         options:
           pullthrough: true
   ```

   See Registry Configuration Reference for more options.
3. Create a new secret called registry-config from your custom registry configuration file you edited locally:

   ```bash
   $ oc secrets new registry-config config.yml=/path/to/custom/registry/config.yml
   ```
4. Add the registry-config secret as a volume to the registry’s deployment configuration to mount the custom configuration file at /etc/docker/registry/:

   ```bash
   $ oc volume dc/docker-registry --add --type=secret --secret-name=registry-config -m /etc/docker/registry/
   ```
5. Update the registry to reference the configuration path from the previous step by adding the following environment variable to the registry’s deployment configuration:

   ```bash
   $ oc env dc/docker-registry
   REGISTRY_CONFIGURATION_PATH=/etc/docker/registry/config.yml
   ```
This may be performed as an iterative process to achieve the desired configuration. For example, during troubleshooting, the configuration may be temporarily updated to put it in debug mode.

To update an existing configuration:

1. Edit the local registry configuration file, config.yml.

2. Delete the registry-config secret:

   ```
   $ oc delete secret registry-config
   ```

3. Recreate the secret to reference the updated configuration file the first step:

   ```
   $ oc secrets new registry-config config.yml=</path/to/custom/registry/config.yml>
   ```

4. Redeploy the registry to read the updated configuration:

   ```
   $ oc deploy docker-registry --latest
   ```

**TIP**

It is recommended that configuration files be maintained in a source control repository.

### 2.7.8. Whitelisting Docker Registries

You can specify a whitelist of docker registries, allowing you to curate a set of images and templates that are available for download by OpenShift users. This curated set can be placed in one or more docker registries, and then added to the whitelist. When using a whitelist, only the specified registries are accessible within OpenShift, and all other registries are denied access by default.

To configure a whitelist:

1. Edit the /etc/sysconfig/docker file to block all registries:

   ```
   BLOCK_REGISTRY='--block-registry=all'
   ```

   You may need to uncomment the BLOCK_REGISTRY line.

2. In the same file, add registries to which you want to allow access:

   ```
   ADD_REGISTRY='--add-registry=<registry1> --add-registry=<registry2>'
   ```

**Example 2.6. Allowing Access to Registries**
ADD_REGISTRY='--add-registry=registry.access.redhat.com'

This example would restrict access to images available on the Red Hat Customer Portal.

Once the whitelist is configured, if a user tries to pull from a docker registry that is not on the whitelist, they will receive an error message stating that this registry is not allowed.

2.7.9. Exposing the Registry

To expose your internal registry externally, it is recommended that you run a secure registry. To expose the registry you must first have deployed a router.

1. **Deploy the registry.**
2. **Secure the registry.**
3. **Deploy a router.**
4. Create your passthrough route with `oc create -f <filename>.json`. The passthrough route will point to the registry service that you have created.

```yaml
apiVersion: v1
kind: Route
metadata:
  name: registry
spec:
  host: <host>  # 1
  to:
    kind: Service
    name: docker-registry  # 2
  tls:
    termination: passthrough  # 3

# 1 The host for your route. You must be able to resolve this name externally via DNS to the router’s IP address.
# 2 The service name for your registry.
# 3 Specify this route as a passthrough route.
```

**NOTE**

Passthrough is currently the only type of route supported for exposing the secure registry.

5. Next, you must trust the certificates being used for the registry on your host system. The certificates referenced were created when you secured your registry.

```
$ sudo mkdir -p /etc/docker/certs.d/<host>
$ sudo cp <ca certificate file> /etc/docker/certs.d/<host>
$ sudo systemctl restart docker
```
6. Log in to the registry using the information from securing the registry. However, this time point to the host name used in the route rather than your service IP. You should now be able to tag and push images using the route host.

```
$ oc get imagestreams -n test
NAME    DOCKER REPO                       TAGS      UPDATED
busybox 172.30.11.215:5000/test/busybox   latest    2 seconds ago
```

```$ docker pull busybox
$ docker tag busybox <host>/test/busybox
$ docker push <host>/test/busybox
```

The push refers to a repository [<host>/test/busybox] (len: 1)
8c2e06607696: Image already exists
6ce2e90b0bc7: Image successfully pushed
cf2616975b4a: Image successfully pushed
Digest:
sha256:6c7e676d76921031532d7d9c0394d0da7c2906f4cb4c049904c4031147d8c
a31

```
$ docker pull <host>/test/busybox
latest: Pulling from <host>/test/busybox
cf2616975b4a: Already exists
6ce2e90b0bc7: Already exists
8c2e06607696: Already exists
Digest:
sha256:6c7e676d76921031532d7d9c0394d0da7c2906f4cb4c049904c4031147d8c
a31
Status: Image is up to date for <host>/test/busybox:latest
```

```
$ oc get imagestreams -n test
NAME    DOCKER REPO                       TAGS      UPDATED
busybox 172.30.11.215:5000/test/busybox   latest    2 seconds ago
```

NOTE
Your image streams will have the IP address and port of the registry service, not the route name and port. See `oc get imagestreams` for details.

NOTE
In the `<host>/test/busybox` example above, `test` refers to the project name.

### 2.7.10. Known Issues

The following are the known issues when deploying or using the integrated registry.

**Image Push Errors with Scaled Registry Using Shared NFS Volume**

When using a scaled registry with a shared NFS volume, you may see one of the following errors during the push of an image:

- `digest invalid`: provided digest did not match uploaded content
- `blob upload unknown`
- `blob upload invalid`
These errors are returned by an internal registry service when Docker attempts to push the image. Its cause originates in the synchronization of file attributes across nodes. Factors such as NFS client side caching, network latency, and layer size can all contribute to potential errors that might occur when pushing an image using the default round-robin load balancing configuration.

You can perform the following steps to minimize the probability of such a failure:

1. Ensure that the `sessionAffinity` of your `docker-registry` service is set to `ClientIP`:

   ```
   $ oc get svc/docker-registry --template='{{.spec.sessionAffinity}}'
   ```

   This should return `ClientIP`, which is the default in recent OpenShift Enterprise versions. If not, change it:

   ```
   $ oc get -o yaml svc/docker-registry | \
   sed 's/(sessionAffinity:\s*).*/\1ClientIP/' | \
   oc replace -f -
   ```

2. Ensure that the NFS export line of your registry volume on your NFS server has the `no_wdelay` options listed. See Export Settings in the Persistent Storage Using NFS topic for details.

### 2.7.11. What’s Next?

After you have a registry deployed, you can:

- Configure authentication; by default, authentication is set to Deny All.
- Deploy a router.

### 2.8. DEPLOYING A ROUTER

#### 2.8.1. Overview

The OpenShift router is the ingress point for all external traffic destined for services in your OpenShift installation. OpenShift provides and supports the following two router plug-ins:

- The HAProxy template router is the default plug-in. It uses the `openshift3/ose-haproxy-router` image to run an HAProxy instance alongside the template router plug-in inside a container on OpenShift. It currently supports HTTP(S) traffic and TLS-enabled traffic via SNI. The router’s container listens on the host network interface, unlike most containers that listen only on private IPs. The router proxies external requests for route names to the IPs of actual pods identified by the service associated with the route.

- The F5 router integrates with an existing F5 BIG-IP® system in your environment to synchronize routes. F5 BIG-IP® version 11.4 or newer is required in order to have the F5 iControl REST API.

   **NOTE**

   The F5 router plug-in is available starting in OpenShift Enterprise 3.0.2.

#### 2.8.2. The Router Service Account

Before deploying an OpenShift cluster, you must have a service account for the router. Starting in
OpenShift Enterprise 3.1, a router service account is automatically created during a quick or advanced installation (previously, this required manual creation). This service account has permissions to a security context constraint (SCC) that allows it to specify host ports.

2.8.3. Deploying the Default HAProxy Router

The `oadm router` command is provided with the administrator CLI to simplify the tasks of setting up routers in a new installation. Just about every form of communication between OpenShift components is secured by TLS and uses various certificates and authentication methods. Use the `--credentials` option to specify what credentials the router should use to contact the master.

**IMPORTANT**

Routers directly attach to port 80 and 443 on all interfaces on a host. Restrict routers to hosts where port 80/443 is available and not being consumed by another service, and set this using node selectors and the scheduler configuration. As an example, you can achieve this by dedicating infrastructure nodes to run services such as routers.

**IMPORTANT**

It is recommended to use separate distinct `openshift-router` credentials with your router. The credentials can be provided using the `--credentials` flag to the `oadm router` command. Alternatively, the default cluster administrator credentials can be used from the `$KUBECONFIG` environment variable.

```bash
$ oadm router --dry-run --service-account=router \   
   --credentials='/etc/origin/master/openshift-router.kubeconfig'
```

---

1. `--credentials` is the path to the CLI configuration file for the `openshift-router`.

**IMPORTANT**

Router pods created using `oadm router` have default resource requests that a node must satisfy for the router pod to be deployed. In an effort to increase the reliability of infrastructure components, the default resource requests are used to increase the QoS tier of the router pods above pods without resource requests. The default values represent the observed minimum resources required for a basic router to be deployed and can be edited in the routers deployment configuration and you may want to increase them based on the load of the router.

The default router service account, named router, is automatically created during quick and advanced installations. To verify that this account already exists:

```bash
$ oadm router --dry-run \   
   --credentials='/etc/origin/master/openshift-router.kubeconfig' \   
   --service-account=router
```

To see what the default router would look like if created:

```bash
$ oadm router -o yaml \   
   --credentials='/etc/origin/master/openshift-router.kubeconfig' \   
   --service-account=router
```
To create a router if it does not exist:

```bash
$ oadm router <router_name> --replicas=<number> \ 
  --credentials='/etc/origin/master/openshift-router.kubeconfig' \ 
  --service-account=router
```

Multiple instances are created on different hosts according to the scheduler policy.

To use a different router image and view the router configuration that would be used:

```bash
$ oadm router <router_name> -o <format> --images=<image> \ 
  --credentials='/etc/origin/master/openshift-router.kubeconfig' \ 
  --service-account=router
```

For example:

```bash
$ oadm router region-west -o yaml --images=myrepo/somerouter:mytag \ 
  --credentials='/etc/origin/master/openshift-router.kubeconfig' \ 
  --service-account=router
```

To deploy the router to any node(s) that match a specified node label:

```bash
$ oadm router <router_name> --replicas=<number> --selector=<label> \ 
  --service-account=router
```

For example, if you want to create a router named `router` and have it placed on a node labeled with `region=infra`:

```bash
$ oadm router router --replicas=1 --selector='region=infra' \ 
  --service-account=router
```

### 2.8.3.1. High Availability

You can set up a highly-available router on your OpenShift Enterprise cluster using IP failover.

### 2.8.3.2. Customizing the Default Routing Subdomain

You can customize the suffix used as the default routing subdomain for your environment using the master configuration file (the `/etc/origin/master/master-config.yaml` file by default). The following example shows how you can set the configured suffix to `v3.openshift.test`:

```yaml
Example 2.7. Master Configuration Snippet

```routingConfig:
  subdomain: v3.openshift.test```

**NOTE**

This change requires a restart of the master if it is running.
CHAPTER 2. INSTALLING

With the OpenShift master(s) running the above configuration, the generated host name for the example
of a host added to a namespace mynamespace would be:
Example 2.8. Generated Host Name
myroute-mynamespace.v3.openshift.test

2.8.3.3. Using Wildcard Certificates
A TLS-enabled route that does not include a certificate uses the router’s default certificate instead. In
most cases, this certificate should be provided by a trusted certificate authority, but for convenience you
can use the OpenShift CA to create the certificate. For example:
$ CA=/etc/origin/master
$ oadm ca create-server-cert --signer-cert=$CA/ca.crt \
--signer-key=$CA/ca.key --signer-serial=$CA/ca.serial.txt \
--hostnames='*.cloudapps.example.com' \
--cert=cloudapps.crt --key=cloudapps.key
The router expects the certificate and key to be in PEM format in a single file:
$ cat cloudapps.crt cloudapps.key $CA/ca.crt > cloudapps.router.pem
From there you can use the --default-cert flag:
$ oadm router --default-cert=cloudapps.router.pem --service-account=router
\
--credentials=${ROUTER_KUBECONFIG:-"$KUBECONFIG"}

NOTE
Browsers only consider wildcards valid for subdomains one level deep. So in this
example, the certificate would be valid for a.cloudapps.example.com but not for
a.b.cloudapps.example.com.

2.8.3.4. Using Secured Routes
Currently, password protected key files are not supported. HAProxy prompts for a password upon
starting and does not have a way to automate this process. To remove a passphrase from a keyfile, you
can run:
# openssl rsa -in <passwordProtectedKey.key> -out <new.key>
Here is an example of how to use a secure edge terminated route with TLS termination occurring on the
router before traffic is proxied to the destination. The secure edge terminated route specifies the TLS
certificate and key information. The TLS certificate is served by the router front end.
First, start up a router instance:
# oadm router --replicas=1 --service-account=router \
--credentials=${ROUTER_KUBECONFIG:-"$KUBECONFIG"}

85


Next, create a private key, csr and certificate for our edge secured route. The instructions on how to do that would be specific to your certificate authority and provider. For a simple self-signed certificate for a domain named \texttt{www.example.test}, see the example shown below:

```bash
# sudo openssl genrsa -out example-test.key 2048
#
# sudo openssl req -new -key example-test.key -out example-test.csr \  
#   -subj "/C=US/ST=CA/L=Mountain View/O=OS3/OU=Eng/CN=www.example.test"
#
# sudo openssl x509 -req -days 366 -in example-test.csr \  
#   -signkey example-test.key -out example-test.crt
```

Generate a route configuration file using the above certificate and key. Make sure to replace \texttt{servicename} with the name of your service.

```bash
# servicename="my-service"
# echo "
apiVersion: v1
kind: Route
metadata:
  name: secured-edge-route
spec:
  host: www.example.test
  to:
    kind: Service
    name: $servicename
  tls:
    termination: edge
    key: |
    $(openssl rsa -in example-test.key | sed 's/^/      /')
    certificate: |
    $(openssl x509 -in example-test.crt | sed 's/^/      /')
"
> example-test-route.yaml
```

Finally add the route to OpenShift (and the router) via:

```bash
# oc create -f example-test-route.yaml
```

Make sure your DNS entry for \texttt{www.example.test} points to your router instance(s) and the route to your domain should be available. The example below uses curl along with a local resolver to simulate the DNS lookup:

```bash
# routerip="4.1.1.1"  # replace with IP address of one of your router
# curl -k --resolve www.example.test:443:$routerip
https://www.example.test/
```

### 2.8.3.5. Using the Container Network Stack

The OpenShift router runs inside a Docker container and the default behavior is to use the network stack of the host (i.e., the node where the router container runs). This default behavior benefits performance because network traffic from remote clients does not need to take multiple hops through user space to
reach the target service and container.

Additionally, this default behavior enables the router to get the actual source IP address of the remote connection rather than getting the node’s IP address. This is useful for defining ingress rules based on the originating IP, supporting sticky sessions, and monitoring traffic, among other uses.

This host network behavior is controlled by the `--host-network` router command line option, and the default behaviour is the equivalent of using `--host-network=true`. If you wish to run the router with the container network stack, use the `--host-network=false` option when creating the router. For example:

```
$ oadm router \
  --credentials='/etc/origin/master/openshift-router.kubeconfig' \
  --service-account=router \
  --host-network=false
```

Internally, this means the router container must publish the 80 and 443 ports in order for the external network to communicate with the router.

**NOTE**

Running with the container network stack means that the router sees the source IP address of a connection to be the NATed IP address of the node, rather than the actual remote IP address.

**NOTE**

On OpenShift clusters using multi-tenant network isolation, routers on a non-default namespace with the `--host-network=false` option will load all routes in the cluster, but routes across the namespaces will not be reachable due to network isolation. With the `--host-network=true` option, routes bypass the container network and it can access any pod in the cluster. If isolation is needed in this case, then do not add routes across the namespaces.

### 2.8.3.6. Exposing Router metrics

Using the `--metrics-image` and `--expose-metrics` options, you can configure the OpenShift Enterprise router to run a sidecar container that exposes or publishes router metrics for consumption by external metrics collection and aggregation systems (e.g. Prometheus, statsd).

Depending on your router implementation, the image is appropriately set up and the metrics sidecar container is started when the router is deployed. For example, the HAProxy-based router implementation defaults to using the `prom/haproxy-exporter` image to run as a sidecar container, which can then be used as a metrics datasource by the Prometheus server.

**NOTE**

The `--metrics-image` option overrides the defaults for HAProxy-based router implementations and, in the case of custom implementations, enables the image to use for a custom metrics exporter or publisher.

1. Grab the HAProxy Prometheus exporter image from the Docker registry:
$ sudo docker pull prom/haproxy-exporter

2. Create the OpenShift Enterprise router:

```bash
$ oadm router \
  --credentials='/etc/origin/master/openshift-router.kubeconfig' \
  --service-account=router --expose-metrics
```

Or, optionally, use the `--metrics-image` option to override the HAProxy defaults:

```bash
$ oadm router \
  --credentials='/etc/origin/master/openshift-router.kubeconfig' \
  --service-account=router --expose-metrics \ 
  --metrics-image=prom/haproxy-exporter
```

3. Once the haproxy-exporter containers (and your HAProxy router) have started, point Prometheus to the sidecar container on port 9101 on the node where the haproxy-exporter container is running:

```bash
$ haproxy_exporter_ip="<enter-ip-address-or-hostname>"
$ cat > haproxy-scraper.yml  <<CFGEOF
---
global:
  scrape_interval: "60s"
  scrape_timeout:  "10s"
  # external_labels:
  # source: openshift-router

scrape_configs:
  - job_name:  "haproxy"
    target_groups:
      - targets: 
        - "${haproxy_exporter_ip}:9101"
CFGEOF
```

$ # And start prometheus as you would normally using the above
config file.

```bash
$ echo "  - Example:  prometheus -config.file=haproxy-scraper.yml "
$ echo "          or you can start it as a container on
OpenShift!!

$ echo "  - Once the prometheus server is up, view the OpenShift
HAProxy "
$ echo "          router metrics at:
http://<ip>:9090/consoles/haproxy.html "
```

### 2.8.4. Deploying a Customized HAProxy Router

The HAProxy router is based on a [golang template](#) that generates the HAProxy configuration file from a list of routes. If you want a customized template router to meet your needs, you can customize the template file, build a new Docker image, and run a customized router.
One common case for this might be implementing new features within the application back ends. For example, it might be desirable in a highly-available setup to \textit{use stick-tables} that synchronizes between peers. The router plug-in provides all the facilities necessary to make this customization.

You can obtain a new \texttt{haproxy-config.template} file from the latest router image by running:

```
# docker run --rm --interactive=true --tty --entrypoint=cat \ 
   registry.access.redhat.com/openshift3/ose-haproxy-router:v3.0.2.0 
   haproxy-config.template
```

Save this content to a file for use as the basis of your customized template.

### 2.8.4.1. Using Stick Tables

The following example customization can be used in a highly-available routing setup to use stick-tables that synchronize between peers.

**Adding a Peer Section**

In order to synchronize stick-tables amongst peers you must define a peers section in your HAProxy configuration. This section determines how HAProxy will identify and connect to peers. The plug-in provides data to the template under the \texttt{.PeerEndpoints} variable to allow you to easily identify members of the router service. You may add a peer section to the \texttt{haproxy-config.template} file inside the router image by adding:

```plaintext
{{ if (len .PeerEndpoints) gt 0 }}
    peers openshift_peers
    {{ range $endpointID, $endpoint := .PeerEndpoints }}
        peer {{($endpoint.TargetName)}} {{($endpoint.IP)}:1937
    {{ end }}
{{ end }}
```

**Changing the Reload Script**

When using stick-tables, you have the option of telling HAProxy what it should consider the name of the local host in the peer section. When creating endpoints, the plug-in attempts to set the \texttt{TargetName} to the value of the endpoint's \texttt{TargetRef.Name}. If \texttt{TargetRef} is not set, it will set the \texttt{TargetName} to the IP address. The \texttt{TargetRef.Name} corresponds with the Kubernetes host name, therefore you can add the \texttt{-L} option to the \texttt{reload-haproxy} script to identify the local host in the peer section.

```plaintext
peer_name=$HOSTNAME

if [ -n "$old_pid" ]; then
    /usr/sbin/haproxy -f $config_file -p $pid_file -L $peer_name -sf
    $old_pid
else
    /usr/sbin/haproxy -f $config_file -p $pid_file -L $peer_name
fi
```

1. Must match an endpoint target name that is used in the peer section.

**Modifying Back Ends**
Finally, to use the stick-tables within back ends, you can modify the HAProxy configuration to use the stick-tables and peer set. The following is an example of changing the existing back end for TCP connections to use stick-tables:

```plaintext
{{ if eq $cfg.TLSTermination "passthrough" }}
backend be_tcp_{{$cfgIdx}}
  balance leastconn
  timeout check 5000ms
  stick-table type ip size 1m expire 5m
  peers openshift_peers {{ end }}
  stick on src
    {{ range $endpointID, $endpoint := $serviceUnit.EndpointTable }}
      server {{$endpointID}} {{$endpoint.IP}}:{{$endpoint.Port}} check inter 5000ms
    {{ end }}
{{ end }}
```

After this modification, you can rebuild your router.

### 2.8.4.2. Rebuilding Your Router

After you have made any desired modifications to the template, such as the example stick tables customization, you must rebuild your router for your changes to go in effect:

1. Rebuild the Docker image to include your customized template.
2. Push the resulting image to your repository.
3. Create the router specifying your new image, either:
   a. in the pod's object definition directly, or
   b. by adding the `--images=<repo>/<image>:<tag>` flag to the `oadm router` command when creating a highly-available routing service.

### 2.8.5. Deploying the F5 Router

**NOTE**

The F5 router plug-in is available starting in OpenShift Enterprise 3.0.2.

The F5 router plug-in is provided as a Docker image and run as a pod, just like the default HAProxy router. Deploying the F5 router is done similarly as well, using the `oadm router` command but providing additional flags (or environment variables) to specify the following parameters for the F5 BIG-IP® host:

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--type=f5-router</code></td>
<td>Specifies that an F5 router should be launched (the default <code>--type</code> is <code>haproxy-router</code>).</td>
</tr>
</tbody>
</table>
As with the HAProxy router, the `oadm router` command creates the service and deployment configuration objects, and thus the replication controllers and pod(s) in which the F5 router itself runs. The replication controller restarts the F5 router in case of crashes. Because the F5 router is only watching routes and endpoints and configuring F5 BIG-IP® accordingly, running the F5 router in this way along with an appropriately configured F5 BIG-IP® deployment should satisfy high-availability requirements.

To deploy the F5 router:

1. First, establish a tunnel using a ramp node, which allows for the routing of traffic to pods through the OpenShift SDN.

2. Run the `oadm router` command with the appropriate flags. For example:

```bash
$ oadm router \
   --type=f5-router \
   --external-host=10.0.0.2 \
   --external-host-username=admin \
   --external-host-password=mypassword \
   --external-host-http-vserver=ose-vserver \
   --external-host-https-vserver=https-ose-vserver \
   --external-host-private-key=/path/to/key \
```

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--external-host</td>
<td>Specifies the F5 BIG-IP® host's management interface's host name or IP address.</td>
</tr>
<tr>
<td>--external-host-username</td>
<td>Specifies the F5 BIG-IP® user name (typically admin).</td>
</tr>
<tr>
<td>--external-host-password</td>
<td>Specifies the F5 BIG-IP® password.</td>
</tr>
<tr>
<td>--external-host-http-vserver</td>
<td>Specifies the name of the F5 virtual server for HTTP connections.</td>
</tr>
<tr>
<td>--external-host-https-vserver</td>
<td>Specifies the name of the F5 virtual server for HTTPS connections.</td>
</tr>
<tr>
<td>--external-host-private-key</td>
<td>Specifies the path to the SSH private key file for the F5 BIG-IP® host. Required to upload and delete key and certificate files for routes.</td>
</tr>
<tr>
<td>--external-host-insecure</td>
<td>A Boolean flag that indicates that the F5 router should skip strict certificate verification with the F5 BIG-IP® host.</td>
</tr>
</tbody>
</table>
--credentials='/etc/origin/master/openshift-router.kubeconfig' \
--service-account=router

--credentials is the path to the CLI configuration file for the openshift-router. It is recommended using an openshift-router specific profile with appropriate permissions.

2.8.6. What’s Next?

If you deployed an HAProxy router, you can learn more about monitoring the router.

If you have not yet done so, you can:

- Configure authentication; by default, authentication is set to Deny All.
- Deploy an integrated Docker registry.
CHAPTER 3. UPGRADING

3.1. OVERVIEW

When new versions of OpenShift are released, you can upgrade your existing cluster to apply the latest enhancements and bug fixes. This includes upgrading from previous minor versions, such as release 3.0 to 3.1, and applying asynchronous errata updates within a minor version (3.1.z releases). See the OpenShift Enterprise 3.1 Release Notes to review the latest changes.

NOTE

Due to the core architectural changes between the major versions, OpenShift Enterprise 2 environments cannot be upgraded to OpenShift Enterprise 3 and require a fresh installation.

Unless noted otherwise, node and masters within a major version are forward and backward compatible, so upgrading your cluster should go smoothly. However, you should not run mismatched versions longer than necessary to upgrade the entire cluster.

If you installed using the quick or advanced installation and the ~/.config/openshift/installer.cfg.yml or inventory file that was used is available, you can perform an automated upgrade. Alternatively, you can upgrade OpenShift manually.

NOTE

The Upgrading topics pertains to RPM-based installations only (i.e., the quick and advanced installation methods) and does not currently cover container-based installations.

If you are using the Pacemaker method for high availability (HA) masters, you can upgrade to the native HA method either using Ansible playbooks or manually.

3.2. PERFORMING AUTOMATED CLUSTER UPGRADES

3.2.1. Overview

Starting with OpenShift 3.0.2, if you installed using the advanced installation and the inventory file that was used is available, you can use the upgrade playbook to automate the OpenShift cluster upgrade process. If you installed using the quick installation method and a ~/.config/openshift/installer.cfg.yml file is available, you can use the installer to perform the automated upgrade.

The automated upgrade performs the following steps for you:

- Applies the latest configuration.
- Upgrades and restart master services.
- Upgrades and restart node services.
- Applies the latest cluster policies.
- Updates the default router if one exists.
Updates the default registry if one exists.

Updates default image streams and InstantApp templates.

**IMPORTANT**

Running Ansible playbooks with the **--tags** or **--check** options is not supported by Red Hat.

### 3.2.2. Preparing for an Automated Upgrade

1. If you are upgrading from OpenShift Enterprise 3.0 to 3.1, on each master and node host you must manually disable the 3.0 channel and enable the 3.1 channel:

   ```
   # subscription-manager repos --disable="rhel-7-server-ose-3.0-rpms"
   --enable="rhel-7-server-ose-3.1-rpms"
   --enable="rhel-7-server-rpms"
   ```

2. For any upgrade path, always ensure that you have the latest version of the `atomic-openshift-utils` package, which should also update the `openshift-ansible-*` packages:

   ```
   # yum update atomic-openshift-utils
   ```

3. Install or update to the following latest available `-excluder` packages on each RHEL 7 system, which helps ensure your systems stay on the correct versions of `atomic-openshift` and `docker` packages when you are not trying to upgrade, according to the OpenShift Enterprise version:

   ```
   # yum install atomic-openshift-excluder atomic-openshift-docker-excluder
   ```

   These packages add entries to the `exclude` directive in the host’s `/etc/yum.conf` file.

4. You must be logged in as a cluster administrative user on the master host for the upgrade to succeed:

   ```
   $ oc login
   ```

   There are two methods for running the automated upgrade: using the installer or running the upgrade playbook directly. Choose and follow one method.

### 3.2.3. Using the Installer to Upgrade

If you installed OpenShift using the **quick installation** method, you should have an installation configuration file located at `~/.config/openshift/installer.cfg.yml`. The installer requires this file to start an upgrade.

**NOTE**

The installer currently only supports upgrading from OpenShift Enterprise 3.0 to 3.1. See [Upgrading to OpenShift Enterprise 3.1 Asynchronous Releases](#) for instructions on using Ansible directly.
If you have an older format installation configuration file in `~/.config/openshift/installer.cfg.yml` from an existing OpenShift Enterprise 3.0 installation, the installer will attempt to upgrade the file to the new supported format. If you do not have an installation configuration file of any format, you can create one manually.

To start the upgrade, run the installer with the `upgrade` subcommand:

1. Satisfy the steps in Preparing for an Automated Upgrade to ensure you are using the latest upgrade playbooks.

2. Run the following command on each host to remove the `atomic-openshift` packages from the list of yum excludes on the host:
   ```
   # atomic-openshift-excluder unexclude
   ```

3. Run the installer with the `upgrade` subcommand:
   ```
   # atomic-openshift-installer upgrade
   ```

4. Follow the on-screen instructions to upgrade to the latest release.

5. After all master and node upgrades have completed, a recommendation will be printed to reboot all hosts. Before rebooting, run the following command on each master and node host to add the `atomic-openshift` packages back to the list of yum excludes on the host:
   ```
   # atomic-openshift-excluder exclude
   ```

Then reboot all hosts.

6. After rebooting, continue to Updating Master and Node Certificates.

### 3.2.4. Running the Upgrade Playbook Directly

Alternatively, you can run the upgrade playbook with Ansible directly, similar to the advanced installation method, if you have an inventory file.

#### 3.2.4.1. Upgrading to OpenShift Enterprise 3.1.0

Before running the upgrade, first update your inventory file to change the `deployment_type` parameter from `enterprise` to `openshift-enterprise`; this is required when upgrading from OpenShift Enterprise 3.0 to 3.1:

Before running the upgrade, first ensure the `deployment_type` parameter in your inventory file is set to `openshift-enterprise`.

If you have multiple masters configured and want to enable rolling, full system restarts of the hosts, you can set the `openshift_rolling_restart_mode` parameter in your inventory file to `system`. Otherwise, the default value `services` performs rolling service restarts on HA masters, but does not reboot the systems. See Configuring Cluster Variables for details.

Then, run the `v3_0_to_v3_1` upgrade playbook. If your inventory file is located somewhere other than the default `etc/ansible/hosts`, add the `-i` flag to specify the location. If you previously used the `atomic-openshift-installer` command to run your installation, you can check `~/.config/openshift/.ansible/hosts` for the last inventory file that was used, if needed.
When the upgrade finishes, a recommendation will be printed to reboot all hosts. After rebooting, continue to Updating Master and Node Certificates.

3.2.4.2. Upgrading to OpenShift Enterprise 3.1 Asynchronous Releases

To apply asynchronous errata updates to an existing OpenShift Enterprise 3.1 cluster, first upgrade the `atomic-openshift-utils` package on the Red Hat Enterprise Linux 7 system where you will be running Ansible:

```
# yum update atomic-openshift-utils
```

Then, run the `v3_1_minor` upgrade playbook. If your inventory file is located somewhere other than the default `/etc/ansible/hosts`, add the `-i` flag to specify the location. If you previously used the `atomic-openshift-installer` command to run your installation, you can check `~/.config/openshift/.ansible/hosts` for the last inventory file that was used, if needed.

```
# ansible-playbook [-i </path/to/inventory/file>] \
/usr/share/ansible/openshift-ansible/playbooks/byo/openshift-cluster/upgrades/v3_1_minor/upgrade.yml
```

When the upgrade finishes, a recommendation will be printed to reboot all hosts. After rebooting, continue to Verifying the Upgrade.

3.2.5. Updating Master and Node Certificates

The following steps may be required for any OpenShift cluster that was originally installed prior to the OpenShift Enterprise 3.1 release. This may include any and all updates from that version.

3.2.5.1. Node Certificates

With the 3.1 release, certificates for each of the kubelet nodes were updated to include the IP address of the node. Any node certificates generated before the 3.1 release may not contain the IP address of the node.

If a node is missing the IP address as part of its certificate, clients may refuse to connect to the kubelet endpoint. Usually this will result in errors regarding the certificate not containing an IP SAN.

In order to remedy this situation, you may need to manually update the certificates for your node.

3.2.5.1.1. Checking the Node’s Certificate

The following command can be used to determine which Subject Alternative Names (SANs) are present in the node’s serving certificate. In this example, the Subject Alternative Names are `mynode`, `mynode.mydomain.com`, and `1.2.3.4`:

```
# openssl x509 -in /etc/origin/node/server.crt -text -noout | grep -A 1 "Subject Alternative Name"
```

```
X509v3 Subject Alternative Name:
  DNS:mynode, DNS:mynode.mydomain.com, IP: 1.2.3.4
```
Ensure that the `nodeIP` value set in the `/etc/origin/node/node-config.yaml` file is present in the IP values from the Subject Alternative Names listed in the node's serving certificate. If the `nodeIP` is not present, then it will need to be added to the node's certificate.

If the `nodeIP` value is already contained within the Subject Alternative Names, then no further steps are required.

You will need to know the Subject Alternative Names and `nodeIP` value for the following steps.

### 3.2.5.1.2. Generating a New Node Certificate

If your current node certificate does not contain the proper IP address, then you must regenerate a new certificate for your node.

**IMPORTANT**

Node certificates will be regenerated on the master (or first master) and are then copied into place on node systems.

1. Create a temporary directory in which to perform the following steps:

   ```sh
   # mkdir /tmp/node_certificate_update
   # cd /tmp/node_certificate_update
   ```

2. Export the signing options:

   ```sh
   # export signing_opts="--signer-cert=/etc/origin/master/ca.crt \
   --signer-key=/etc/origin/master/ca.key \
   --signer-serial=/etc/origin/master/ca.serial.txt"
   ```

3. Generate the new certificate:

   ```sh
   # oadm ca create-server-cert --cert=server.crt \
   --key=server.key $signing_opts \
   --hostnames=<existing_SANs>,<nodeIP>
   ```

   For example, if the Subject Alternative Names from before were `mynode`, `mynode.mydomain.com`, and `1.2.3.4`, and the `nodeIP` was `10.10.10.1`, then you would need to run the following command:

   ```sh
   # oadm ca create-server-cert --cert=server.crt \
   --key=server.key $signing_opts \
   --hostnames=mynode,mynode.mydomain.com,1.2.3.4,10.10.10.1
   ```

### 3.2.5.1.3. Replace Node Serving Certificates

Back up the existing `/etc/origin/node/server.crt` and `/etc/origin/node/server.key` files for your node:

```sh
# mv /etc/origin/node/server.crt /etc/origin/node/server.crt.bak
# mv /etc/origin/node/server.key /etc/origin/node/server.key.bak
```

You must now copy the new `server.crt` and `server.key` created in the temporary directory during the previous step:
# mv /tmp/node_certificate_update/server.crt /etc/origin/node/server.crt
# mv /tmp/node_certificate_update/server.key /etc/origin/node/server.key

After you have replaced the node's certificate, restart the node service:

    # systemctl restart atomic-openshift-node

### 3.2.5.2. Master Certificates

With the 3.1 release, certificates for each of the masters were updated to include all names that pods may use to communicate with masters. Any master certificates generated before the 3.1 release may not contain these additional service names.

#### 3.2.5.2.1. Checking the Master's Certificate

The following command can be used to determine which Subject Alternative Names (SANs) are present in the master's serving certificate. In this example, the Subject Alternative Names are `mymaster`, `mymaster.mydomain.com`, and `1.2.3.4`:

    # openssl x509 -in /etc/origin/master/master.server.crt -text -noout | grep -A 1 "Subject Alternative Name"
    X509v3 Subject Alternative Name:
    DNS:mymaster, DNS:mymaster.mydomain.com, IP: 1.2.3.4

Ensure that the following entries are present in the Subject Alternative Names for the master's serving certificate:

<table>
<thead>
<tr>
<th>Entry</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kubernetes service IP address</td>
<td>172.30.0.1</td>
</tr>
<tr>
<td>All master host names</td>
<td><code>master1.example.com</code></td>
</tr>
<tr>
<td>All master IP addresses</td>
<td>192.168.122.1</td>
</tr>
<tr>
<td>Public master host name in clustered environments</td>
<td><code>public-master.example.com</code></td>
</tr>
<tr>
<td>kubernetes</td>
<td></td>
</tr>
<tr>
<td>kubernetes.default</td>
<td></td>
</tr>
<tr>
<td>kubernetes.default.svc</td>
<td></td>
</tr>
<tr>
<td>kubernetes.default.svc.cluster.local</td>
<td></td>
</tr>
<tr>
<td>openshift</td>
<td></td>
</tr>
<tr>
<td>openshift.default</td>
<td></td>
</tr>
</tbody>
</table>
If these names are already contained within the Subject Alternative Names, then no further steps are required.

### 3.2.5.2.2. Generating a New Master Certificate

If your current master certificate does not contain all names from the list above, then you must generate a new certificate for your master:

1. Back up the existing `/etc/origin/master/master.server.crt` and `/etc/origin/master/master.server.key` files for your master:

   ```bash
   # mv /etc/origin/master/master.server.crt /etc/origin/master/master.server.crt.bak
   # mv /etc/origin/master/master.server.key /etc/origin/master/master.server.key.bak
   ```

2. Export the service names. These names will be used when generating the new certificate:

   ```bash
   # export service_names="kubernetes,kubernetes.default,kubernetes.default.svc,
   kubernetes.default.svc.cluster.local,openshift,openshift.default,openshift.default.svc,openshift.default.svc.cluster.local"
   ```

3. You will need the first IP in the services subnet (the `kubernetes` service IP) as well as the values of `masterIP`, `masterURL` and `publicMasterURL` contained in the `/etc/origin/master/master-config.yaml` file for the following steps. The `kubernetes` service IP can be obtained with:

   ```bash
   # oc get svc/kubernetes --template='{{.spec.clusterIP}}'
   ```

4. Generate the new certificate:

   ```bash
   # oadm ca create-master-certs \
   --hostnames=<master_hostnames>,<master_IP_addresses>,<kubernetes_service_IP>,$service_names \  
   --master=<internal_master_address> \  
   --public-master=<public_master_address> \  
   --cert-dir=/etc/origin/master/ \  
   --overwrite=false
   ```

   **1.** Adjust `<master_hostnames>` to match your master host name. In a clustered environment, add all master host names.

   **2.** Adjust `<master_IP_addresses>` to match the value of `masterIP`. In a clustered environment, add all master IP addresses.
3. Adjust `<kubernetes_service_IP>` to the first IP in the `kubernetes` services subnet.

4. Adjust `<internal_master_address>` to match the value of `masterURL`.

5. Adjust `<public_master_address>` to match the value of `masterPublicURL`.

5. Restart master services. For single master deployments:

   ```
   # systemctl restart atomic-openshift-master
   
   For native HA multiple master deployments:
   # systemctl restart atomic-openshift-master-api
   # systemctl restart atomic-openshift-master-controllers
   
   For Pacemaker HA multiple master deployments:
   # pcs resource restart master
   
   After the service restarts, the certificate update is complete.
   
3.2.6. Upgrading the EFK Logging Stack

If you have previously deployed the EFK logging stack and want to upgrade to the latest logging component images, the steps must be performed manually as shown in Manual Upgrades.

3.2.7. Verifying the Upgrade

To verify the upgrade, first check that all nodes are marked as `Ready`:

```
# oc get nodes
NAME            LABELS                             STATUS
master.example.com kubernetes.io/hostname=master.example.com,region=infra,zone=default Ready
node1.example.com kubernetes.io/hostname=node1.example.com,region=primary,zone=east Ready
```

Then, verify that you are running the expected versions of the `docker-registry` and `router` images, if deployed:

```
# oc get -n default dc/docker-registry -o json | grep "image"
   "image": "openshift3/ose-docker-registry:v3.1.1.11",
# oc get -n default dc/router -o json | grep "image"
   "image": "openshift3/ose-haproxy-router:v3.1.1.11",
```

If you upgraded from OSE 3.0 to OSE 3.1, verify in your old `/etc/sysconfig/openshift-master` and `/etc/sysconfig/openshift-node` files that any custom configuration is added to your new `/etc/sysconfig/atomic-openshift-master` and `/etc/sysconfig/atomic-openshift-node` files.
After upgrading, you can use the experimental diagnostics tool to look for common issues:

```
# openshift ex diagnostics
...
[Note] Summary of diagnostics execution:
[Note] Completed with no errors or warnings seen.
```

3.3. PERFORMING MANUAL CLUSTER UPGRADES

3.3.1. Overview

As an alternative to performing an automated upgrade, you can manually upgrade your OpenShift cluster. To manually upgrade without disruption, it is important to upgrade each component as documented in this topic.

**IMPORTANT**

Before you begin your upgrade, familiarize yourself now with the entire procedure. *Specific releases may require additional steps* to be performed at key points before or during the standard upgrade process.

3.3.2. Preparing for a Manual Upgrade

1. If you are upgrading from OpenShift Enterprise 3.0 to 3.1, perform the following steps:
   a. On each master and node host, manually disable the 3.0 channel and enable the 3.1 channel:

      ```
      # subscription-manager repos --disable="rhel-7-server-ose-3.0-rpms"
      --enable="rhel-7-server-ose-3.1-rpms"
      ```
   b. Create an etcd backup on each master:

      ```
      # yum install etcd
      # etcdctl backup --data-dir /var/lib/openshift/openshift.local.etcd
      --backup-dir /var/lib/openshift/openshift.local.etcd.bak
      ```
   c. Remove support for the v1beta3 API. Update the `/etc/openshift/master/master-config.yml` file on each master, and remove v1beta3 from the `apiLevels` and `kubernetesMasterConfig.apiLevels` parameters.
   d. During this upgrade, some directories are renamed from openshift to origin, so create the following symlinks on each host:

      ```
      # ln -s /var/lib/openshift /var/lib/origin
      # ln -s /etc/openshift /etc/origin
      ```

2. Install or update to the following latest available `-excluder` packages on each RHEL 7 system, which helps ensure your systems stay on the correct versions of atomic-openshift and docker packages when you are not trying to upgrade, according to the OpenShift Enterprise version:
# yum install atomic-openshift-excluder atomic-openshift-docker-excluder

These packages add entries to the `exclude` directive in the host's `/etc/yum.conf` file.

3. If you are already running OpenShift Enterprise 3.1 or later, create an `etcd` backup by running:

   ```bash
   # yum install etcd
   # etcdctl backup --data-dir /var/lib/origin/openshift.local.etcd
   --backup-dir /var/lib/origin/openshift.local.etcd.bak
   ```

4. For any upgrade path, always ensure that you are running the latest kernel:

   ```bash
   # yum update kernel
   ```

3.3.3. Upgrading Masters

Upgrade your master hosts first:

1. Run the following command on each master to remove the `atomic-openshift` packages from the list of yum excludes on the host:

   ```bash
   # atomic-openshift-excluder unexclude
   ```

2. Upgrade the `atomic-openshift` packages or related images.

   ```bash
   # yum upgrade atomic-openshift-master
   ```

3. If you are upgrading from OpenShift Enterprise 3.0 to 3.1:
   a. Create the following master proxy client certificates:

      ```bash
      # cd /etc/origin/master/
      # oadm ca create-master-certs --cert-dir=/etc/origin/master/ 
      --master=https://<internal-master-fqdn>:8443 
      --public-master=https://<external-master-fqdn>:8443 
      --hostnames=<external-master-fqdn>,<internal-master-fqdn>,localhost,127.0.0.1,\nmaster-ip-address>,kubernetes.default.local 
      --overwrite=false
      ```

      This creates files at `/etc/origin/master/master.proxy-client.{crt,key}`.

   b. Then, add the master proxy client certificates to the `/etc/origin/master/master-config.yml` file on each master:

      ```yaml
      kubernetesMasterConfig:
      proxyClientInfo:
      certFile: master.proxy-client.crt
      keyFile: master.proxy-client.key
      ```

   c. Enable the following renamed service on master hosts:
# systemctl enable atomic-openshift-master

4. For any upgrade path, now restart the `atomic-openshift-master` service and review its logs to ensure services have been restarted successfully:

```bash
# systemctl restart atomic-openshift-master
# journalctl -r -u atomic-openshift-master
```

5. Run the following command on each master to add the `atomic-openshift` packages back to the list of yum excludes on the host:

```bash
# atomic-openshift-excluder exclude
```

### 3.3.4. Updating Policy Definitions

After a cluster upgrade, the recommended default cluster roles may have been updated. To check if an update is recommended for your environment, you can run:

```bash
# oadm policy reconcile-cluster-roles
```

This command outputs a list of roles that are out of date and their new proposed values. For example:

```bash
# oadm policy reconcile-cluster-roles
apiVersion: v1
items:
  - apiVersion: v1
    kind: ClusterRole
    metadata:
      creationTimestamp: null
      name: admin
    rules:
      - attributeRestrictions: null
        resources:
          - builds/custom
```

**NOTE**

Your output will vary based on the OpenShift version and any local customizations you have made. Review the proposed policy carefully.

You can either modify this output to re-apply any local policy changes you have made, or you can automatically apply the new policy using the following process:

1. Reconcile the cluster roles:

   ```bash
   # oadm policy reconcile-cluster-roles --confirm
   ```

2. Restart the master service:

   ```bash
   # systemctl restart atomic-openshift-master
   ```
3. Reconcile security context constraints:

   ```
   # oadm policy reconcile-sccs \
   --additive-only=true \ 
   --confirm
   ```

### 3.3.5. Upgrading Nodes

After upgrading your masters, you can upgrade your nodes. When restarting the `atomic-openshift-node` service, there will be a brief disruption of outbound network connectivity from running pods to services while the `service proxy` is restarted. The length of this disruption should be very short and scales based on the number of services in the entire cluster.

To upgrade nodes:

1. Run the following command on each node to remove the `atomic-openshift` packages from the list of yum excludes on the host:

   ```
   # atomic-openshift-excluder unexclude
   ```

2. As a user with `cluster-admin` privileges, disable scheduling for the node:

   ```
   # oadm manage-node <node> --schedulable=false
   ```

3. On each node host, upgrade all `atomic-openshift` packages:

   ```
   # yum upgrade atomic-openshift\*
   ```

4. If you are upgrading from OpenShift Enterprise 3.0 to 3.1, enable the following renamed service on node hosts:

   ```
   # systemctl enable atomic-openshift-node
   ```

5. For any upgrade path, now restart the `atomic-openshift-node` service:

   ```
   # systemctl restart atomic-openshift-node
   ```

6. Enable scheduling again for any non-master nodes that you disabled:

   ```
   # oadm manage-node <node> --schedulable=true
   ```

7. Run the following command on the node to add the `atomic-openshift` packages back to the list of yum excludes on the host:

   ```
   # atomic-openshift-excluder exclude
   ```

8. Repeat these steps on the next node, and continue repeating these steps until all nodes have been upgraded.

9. After all nodes have been upgraded, as a user with `cluster-admin` privileges, verify that all nodes are showing as `Ready`:

   ```
   # oc get nodes
   ```
<table>
<thead>
<tr>
<th>NAME</th>
<th>LABELS</th>
</tr>
</thead>
<tbody>
<tr>
<td>master.example.com</td>
<td>kubernetes.io/hostname=master.example.com</td>
</tr>
<tr>
<td>Ready,SchedulingDisabled</td>
<td></td>
</tr>
<tr>
<td>node1.example.com</td>
<td>kubernetes.io/hostname=node1.example.com</td>
</tr>
<tr>
<td>Ready</td>
<td></td>
</tr>
<tr>
<td>node2.example.com</td>
<td>kubernetes.io/hostname=node2.example.com</td>
</tr>
<tr>
<td>Ready</td>
<td></td>
</tr>
</tbody>
</table>

### 3.3.6. Upgrading the Router

If you have previously deployed a router, the router deployment configuration must be upgraded to apply updates contained in the router image. To upgrade your router without disrupting services, you must have previously deployed a highly-available routing service.

Edit your router’s deployment configuration. For example, if it has the default `router` name:

```bash
# oc edit dc/router
```

Apply the following changes:

```yaml
...  
spec:  
  template:  
    spec:  
      containers:  
        - env:  
          ...  
        image: registry.access.redhat.com/openshift3/ose-haproxy-router:v3.1.1.11  
        imagePullPolicy: IfNotPresent  
    ...  
```

1. Adjust the image version to match the version you are upgrading to.

You should see one router pod updated and then the next.

### 3.3.7. Upgrading the Registry

The registry must also be upgraded for changes to take effect in the registry image. If you have used a `PersistentVolumeClaim` or a host mount point, you may restart the registry without losing the contents of your registry. Deploying a Docker Registry details how to configure persistent storage for the registry.

Edit your registry’s deployment configuration:

```bash
# oc edit dc/docker-registry
```

Apply the following changes:

```yaml
...  
spec:  
  template:  
```

---

105
Adjust the image version to match the version you are upgrading to.

**IMPORTANT**

Images that are being pushed or pulled from the internal registry at the time of upgrade will fail and should be restarted automatically. This will not disrupt pods that are already running.

### 3.3.8. Updating the Default Image Streams and Templates

By default, the quick and advanced installation methods automatically create default image streams, InstantApp templates, and database service templates in the openshift project, which is a default project to which all users have view access. These objects were created during installation from the JSON files located under the `/usr/share/ansible/openshift-ansible/roles/openshift_examples/files/examples/` directory.

**NOTE**

Because RHEL Atomic Host 7 cannot use yum to update packages, the following steps must take place on a RHEL 7 system.

1. Update the packages that provide the example JSON files. On a subscribed Red Hat Enterprise Linux 7 system where you can run the CLI as a user with cluster-admin permissions, install or update to the latest version of the atomic-openshift-utils package, which should also update the openshift-ansible- packages:

   ```bash
   # yum update atomic-openshift-utils
   ```

   The openshift-ansible-roles package provides the latest example JSON files.

2. Update the global openshift project by running the following commands. Receiving warnings about items that already exist is expected.

   ```bash
   # oc create -n openshift -f /usr/share/openshift/examples/image-streams/image-streams-rhel7.json
   # oc create -n openshift -f /usr/share/openshift/examples/db-templates
   # oc create -n openshift -f /usr/share/openshift/examples/quickstart-templates
   # oc create -n openshift -f /usr/share/openshift/examples/xpaas-streams
   # oc create -n openshift -f /usr/share/openshift/examples/xpaas-templates
   # oc replace -n openshift -f /usr/share/openshift/examples/image-
streams/image-streams-rhel7.json
# oc replace -n openshift -f /usr/share/openshift/examples/db-templates
# oc replace -n openshift -f
/usr/share/openshift/examples/quickstart-templates
# oc replace -n openshift -f /usr/share/openshift/examples/xpaas-streams
# oc replace -n openshift -f /usr/share/openshift/examples/xpaas-templates

3. After a manual upgrade, get the latest templates from `openshift-ansible-roles`:

```
rpm -ql openshift-ansible-roles | grep examples | grep v1.2
```

In this example, `/usr/share/ansible/openshift-ansible/roles/openshift_examples/files/examples/v1.2/image-streams/image-streams-rhel7.json` is the latest file that you want in the latest `openshift-ansible-roles` package.

`/usr/share/openshift/examples/image-streams/image-streams-rhel7.json` is not owned by a package, but is updated by Ansible. If you are upgrading outside of Ansible, you need to get the latest `.json` files on the system where you are running `oc`, which can run anywhere that has access to the master.

4. Install `atomic-openshift-utils` and its dependencies to install the new content into `/usr/share/ansible/openshift-ansible/roles/openshift_examples/files/examples/v1.2/`

```
$ oc create -n openshift -f /usr/share/ansible/openshift-ansible/roles/openshift_examples/files/examples/v1.2/image-streams/image-streams-rhel7.json
$ oc replace -n openshift -f /usr/share/ansible/openshift-ansible/roles/openshift_examples/files/examples/v1.2/image-streams/image-streams-rhel7.json
```

5. Update the templates:

```
$ oc create -n openshift -f /usr/share/ansible/openshift-ansible/roles/openshift_examples/files/examples/v1.2/quickstart-templates/
$ oc create -n openshift -f /usr/share/ansible/openshift-ansible/roles/openshift_examples/files/examples/v1.2/db-templates/
$ oc create -n openshift -f /usr/share/ansible/openshift-ansible/roles/openshift_examples/files/examples/v1.2/infrastructure-templates/
$ oc create -n openshift -f /usr/share/ansible/openshift-ansible/roles/openshift_examples/files/examples/v1.2/xpaas-templates/
$ oc create -n openshift -f /usr/share/ansible/openshift-ansible/roles/openshift_examples/files/examples/v1.2/xpaas-streams/
$ oc replace -n openshift -f /usr/share/ansible/openshift-ansible/roles/openshift_examples/files/examples/v1.2/quickstart-templates/
$ oc replace -n openshift -f /usr/share/ansible/openshift-ansible/roles/openshift_examples/files/examples/v1.2/db-templates/
$ oc replace -n openshift -f /usr/share/ansible/openshift-ansible/roles/openshift_examples/files/examples/v1.2/infrastructure-templates/
```
Errors are generated for items that already exist. This is expected behavior:

```
# oc create -n openshift -f /usr/share/ansible/openshift-ansible/roles/openshift_examples/files/examples/v1.2/quickstart-templates/
Error from server: error when creating
"/usr/share/ansible/openshift-ansible/roles/openshift_examples/files/examples/v1.2/quickstart-templates/cakephp-mysql.json": templates "cakephp-mysql-example" already exists
Error from server: error when creating
"/usr/share/ansible/openshift-ansible/roles/openshift_examples/files/examples/v1.2/quickstart-templates/cakephp.json": templates "cakephp-example" already exists
Error from server: error when creating
"/usr/share/ansible/openshift-ansible/roles/openshift_examples/files/examples/v1.2/quickstart-templates/dancer-mysql.json": templates "dancer-mysql-example" already exists
Error from server: error when creating
"/usr/share/ansible/openshift-ansible/roles/openshift_examples/files/examples/v1.2/quickstart-templates/dancer.json": templates "dancer-example" already exists
Error from server: error when creating
"/usr/share/ansible/openshift-ansible/roles/openshift_examples/files/examples/v1.2/quickstart-templates/django-postgresql.json": templates "django-psql-example" already exists
```

Now, content can be updated. Without running the automated upgrade playbooks, the content is not updated in `/usr/share/openshift/`.

### 3.3.9. Importing the Latest Images

After updating the default image streams, you may also want to ensure that the images within those streams are updated. For each image stream in the default `openshift` project, you can run:

```
# oc import-image -n openshift <imagestream>
```

For example, get the list of all image streams in the default `openshift` project:

```
# oc get is -n openshift
NAME                  DOCKER REPO                                      TAGS                   UPDATED
mongodb               registry.access.redhat.com/openshift3/mongodb-24-rhel7 2.4,latest,v3.1       16 hours ago
mysql                 registry.access.redhat.com/openshift3/mysql-55-rhel7
```
Update each image stream one at a time:

```bash
# oc import-image -n openshift nodejs
Waiting for the import to complete, CTRL+C to stop waiting.
The import completed successfully.
```

```
Name:                   nodejs
Created:                16 hours ago
Labels:                 <none>
Annotations:            openshift.io/image.dockerRepositoryCheck=2015-07-21T13:17:00Z
Docker Pull Spec:       registry.access.redhat.com/openshift3/nodejs-010-rhel7

<table>
<thead>
<tr>
<th>Tag</th>
<th>Spec</th>
<th>Created</th>
<th>PullSpec</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.10</td>
<td>latest</td>
<td>16 hours ago</td>
<td>registry.access.redhat.com/openshift3/nodejs-010-rhel7:latest</td>
</tr>
<tr>
<td></td>
<td>registry.access.redhat.com/openshift3/nodejs-010-rhel7:latest</td>
<td>16 hours ago</td>
<td>66d92cebc0e48e4e4be3a93d0f9bd54f21af7928ceaa384d20800f6e6fcf669f</td>
</tr>
<tr>
<td>v3.1</td>
<td>&lt;pushed&gt;</td>
<td>16 hours ago</td>
<td>registry.access.redhat.com/openshift3/nodejs-010-rhel7:v3.1</td>
</tr>
<tr>
<td></td>
<td>registry.access.redhat.com/openshift3/nodejs-010-rhel7:v3.1</td>
<td>16 hours ago</td>
<td>66d92cebc0e48e4e4be3a93d0f9bd54f21af7928ceaa384d20800f6e6fcf669f</td>
</tr>
</tbody>
</table>
```

**IMPORTANT**

In order to update your S2I-based applications, you must manually trigger a new build of those applications after importing the new images using `oc start-build <app-name>`.

### 3.3.10. Updating Master and Node Certificates

The following steps may be required for any OpenShift cluster that was originally installed prior to the OpenShift Enterprise 3.1 release. This may include any and all updates from that version.

#### 3.3.10.1. Node Certificates

With the 3.1 release, certificates for each of the kubelet nodes were updated to include the IP address of the node. Any node certificates generated before the 3.1 release may not contain the IP address of the node.

If a node is missing the IP address as part of its certificate, clients may refuse to connect to the kubelet endpoint. Usually this will result in errors regarding the certificate not containing an IP SAN.

In order to remedy this situation, you may need to manually update the certificates for your node.

#### 3.3.10.1.1. Checking the Node's Certificate
The following command can be used to determine which Subject Alternative Names (SANs) are present in the node’s serving certificate. In this example, the Subject Alternative Names are `mynode`, `mynode.mydomain.com`, and `1.2.3.4`:

```bash
# openssl x509 -in /etc/origin/node/server.crt -text -noout | grep -A 1 "Subject Alternative Name"
X509v3 Subject Alternative Name:
  DNS:mynode, DNS:mynode.mydomain.com, IP: 1.2.3.4
```

Ensure that the `nodeIP` value set in the `/etc/origin/node/node-config.yaml` file is present in the IP values from the Subject Alternative Names listed in the node’s serving certificate. If the `nodeIP` is not present, then it will need to be added to the node’s certificate.

If the `nodeIP` value is already contained within the Subject Alternative Names, then no further steps are required.

You will need to know the Subject Alternative Names and `nodeIP` value for the following steps.

### 3.3.10.1.2. Generating a New Node Certificate

If your current node certificate does not contain the proper IP address, then you must regenerate a new certificate for your node.

**IMPORTANT**

Node certificates will be regenerated on the master (or first master) and are then copied into place on node systems.

1. Create a temporary directory in which to perform the following steps:

   ```bash
   # mkdir /tmp/node_certificate_update
   # cd /tmp/node_certificate_update
   
   # export signing_opts="--signer-cert=/etc/origin/master/ca.crt
   #     --signer-key=/etc/origin/master/ca.key
   #     --signer-serial=/etc/origin/master/ca.serial.txt"
   
   # oadm ca create-server-cert --cert=server.crt
   #     --key=server.key $signing_opts
   #     --hostnames=<existing_SANs>,<nodeIP>
   
   For example, if the Subject Alternative Names from before were `mynode`, `mynode.mydomain.com`, and `1.2.3.4`, and the `nodeIP` was `10.10.10.1`, then you would need to run the following command:

   ```bash
   # oadm ca create-server-cert --cert=server.crt
   #     --key=server.key $signing_opts
   #     --hostnames=mynode,mynode.mydomain.com,1.2.3.4,10.10.10.1
   ```
3.3.10.1.3. Replace Node Serving Certificates

Back up the existing `/etc/origin/node/server.crt` and `/etc/origin/node/server.key` files for your node:

```bash
# mv /etc/origin/node/server.crt /etc/origin/node/server.crt.bak
# mv /etc/origin/node/server.key /etc/origin/node/server.key.bak
```

You must now copy the new `server.crt` and `server.key` created in the temporary directory during the previous step:

```bash
# mv /tmp/node_certificate_update/server.crt /etc/origin/node/server.crt
# mv /tmp/node_certificate_update/server.key /etc/origin/node/server.key
```

After you have replaced the node's certificate, restart the node service:

```bash
# systemctl restart atomic-openshift-node
```

3.3.10.2. Master Certificates

With the 3.1 release, certificates for each of the masters were updated to include all names that pods may use to communicate with masters. Any master certificates generated before the 3.1 release may not contain these additional service names.

3.3.10.2.1. Checking the Master’s Certificate

The following command can be used to determine which Subject Alternative Names (SANs) are present in the master’s serving certificate. In this example, the Subject Alternative Names are `mymaster`, `mymaster.mydomain.com`, and `1.2.3.4`:

```bash
# openssl x509 -in /etc/origin/master/master.server.crt -text -noout | grep -A 1 "Subject Alternative Name"
X509v3 Subject Alternative Name:
   DNS:mymaster, DNS:mymaster.mydomain.com, IP: 1.2.3.4
```

Ensure that the following entries are present in the Subject Alternative Names for the master’s serving certificate:

<table>
<thead>
<tr>
<th>Entry</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kubernetes service IP address</td>
<td>172.30.0.1</td>
</tr>
<tr>
<td>All master host names</td>
<td><code>master1.example.com</code></td>
</tr>
<tr>
<td>All master IP addresses</td>
<td>192.168.122.1</td>
</tr>
<tr>
<td>Public master host name in clustered environments</td>
<td><code>public-master.example.com</code></td>
</tr>
<tr>
<td><code>kubernetes</code></td>
<td></td>
</tr>
<tr>
<td><code>kubernetes.default</code></td>
<td></td>
</tr>
</tbody>
</table>
### 3.3.10.2.2. Generating a New Master Certificate

If your current master certificate does not contain all names from the list above, then you must generate a new certificate for your master:

1. Back up the existing `/etc/origin/master/master.server.crt` and `/etc/origin/master/master.server.key` files for your master:

   ```bash
   # mv /etc/origin/master/master.server.crt /etc/origin/master/master.server.crt.bak
   # mv /etc/origin/master/master.server.key /etc/origin/master/master.server.key.bak
   ```

2. Export the service names. These names will be used when generating the new certificate:

   ```bash
   # export service_names="kubernetes,kubernetes.default,kubernetes.default.svc,
   kubernetes.default.svc.cluster.local,openshift,openshift.default,open
   shift.default.svc,openshift.default.svc.cluster.local"
   ```

3. You will need the first IP in the services subnet (the `kubernetes` service IP) as well as the values of `masterIP,masterURL` and `publicMasterURL` contained in the `/etc/origin/master/master-config.yaml` file for the following steps. The `kubernetes` service IP can be obtained with:

   ```bash
   # oc get svc/kubernetes --template='{{.spec.clusterIP}}'
   ```

4. Generate the new certificate:

   ```bash
   # oadm ca create-master-certs
   --hostnames=<master_hostnames>,<master_IP_addresses>,
   <kubernetes_service_IP>,$service_names 1 2 3
   --master=<internal_master_address> 4
   ```
Adjust `<master_hostnames>` to match your master host name. In a clustered environment, add all master host names.

Adjust `<master_IP_addresses>` to match the value of `masterIP`. In a clustered environment, add all master IP addresses.

Adjust `<kubernetes_service_IP>` to the first IP in the `kubernetes` services subnet.

Adjust `<internal_master_address>` to match the value of `masterURL`.

Adjust `<public_master_address>` to match the value of `masterPublicURL`.

5. Restart master services. For single master deployments:

```bash
# systemctl restart atomic-openshift-master
```

For native HA multiple master deployments:

```bash
# systemctl restart atomic-openshift-master-api
# systemctl restart atomic-openshift-master-controllers
```

For Pacemaker HA multiple master deployments:

```bash
# pcs resource restart master
```

After the service restarts, the certificate update is complete.

### 3.3.11. Upgrading the EFK Logging Stack

If you have previously deployed the EFK logging stack and want to upgrade to the latest logging component images, you must take the following steps to safely upgrade with minimal disruption to your log data.

**NOTE**

The following steps apply when you want to update to newer OpenShift Enterprise 3.1 logging images, but are not yet fully upgrading your cluster to a later minor or major release of OpenShift Enterprise. The `IMAGE_VERSION` variable is used in a later step to ensure that you do not accidentally pull the wrong images.

1. Ensure you are working in the project where the EFK stack was previously deployed, and stay in that project for the remainder of these steps. For example, if the project is named `logging`:

```bash
$ oc project logging
```

2. Scale down your Fluentd instances to 0:

```bash
$ oc scale dc/logging-fluentd --replicas=0
```
Wait until they have properly terminated. This helps prevent loss of data by giving them time to properly flush their current buffer and send any logs they were processing to Elasticsearch.

3. Scale down your Kibana instances:

   ```
   $ oc scale dc/logging-kibana --replicas=0
   ```

   If you have an operations deployment, also run:

   ```
   $ oc scale dc/logging-kibana-ops --replicas=0
   ```

4. Once your Fluentd and Kibana pods are confirmed to be terminated, you can safely scale down the Elasticsearch pods:

   ```
   $ oc scale dc/logging-es-<unique_name> --replicas=0
   ```

   If you have an operations deployment, also run:

   ```
   $ oc scale dc/logging-es-ops-<unique_name> --replicas=0
   ```

5. After your Elasticsearch pods are confirmed to be terminated, pull in the latest EFK images using the same procedure described in Importing the Latest Images, replacing the `openshift` project with the project where the EFK stack was previously deployed. For example, if the project is named `logging`:

   ```
   $ oc import-image -n logging <imagestream>
   ```

   The list of image streams are:

   ```
   logging-auth-proxy
   logging-elasticsearch
   logging-fluentd
   logging-kibana
   ```

6. With the latest images in your repository, you can now rerun the deployer to generate any missing or changed features.

   a. First, ensure that your OAuth client has been deleted:

      ```
      $ oc delete oauthclient --selector logging-infra=support
      ```

   b. Then, proceed to follow the same steps as done previously in Deploying the EFK Stack, but ensure that you add `IMAGE_VERSION` to the list of variables and set it to the appropriate version. For example, for the latest 3.1.1 image:

      ```
      $ oc process logging-deployer-template -n openshift -v
      IMAGE_VERSION=3.1.1,KIBANA_HOSTNAME=kibana.example.com,ES_CLUSTER_SIZE=1,PUBLIC_MASTER_URL=https://localhost:8443
      | oc create -f -
      ```

      See Deploying the EFK Stack for the full instructions. After the deployer completes, re-attach your persistent volumes you were using previously.
7. Next, scale Elasticsearch back up incrementally so that the cluster has time to rebuild.
   a. To begin, scale up to 1:
      
      ```
      $ oc scale dc/logging-es-<unique_name> --replicas=1
      ```
      
      Follow the logs of the resulting pod to ensure that it is able to recover its indices correctly and that there are no errors:
      
      ```
      $ oc logs -f <pod_name>
      ```
      
      If that is successful, you can then do the same for the operations cluster, if one was previously used.

   b. After all Elasticsearch nodes have recovered their indices, continue to scale it back up to the size it was prior to doing maintenance. Check the logs of the Elasticsearch members to verify that they have correctly joined the cluster and recovered.

8. Now scale Kibana and Fluentd back up to their previous state. Because Fluentd was shut down and allowed to push its remaining records to Elasticsearch in the previous steps, it can now pick back up from where it left off with no loss of logs, provided any unread log files are still available on the node.

9. In the latest version, Kibana will display indices differently now in order to prevent users from being able to access the logs of previously created projects that have been deleted. Due to this change, your old logs will not appear automatically. To migrate your old indices to the new format, rerun the deployer with `-v MODE=migrate` in addition to your prior flags. This should be run while your Elasticsearch cluster is running, as the script must connect to it to make changes.

   **NOTE**

   This only impacts non-operations logs. Operations logs will appear the same as in previous versions. There should be minimal performance impact to Elasticsearch while running this and it will not perform an install.

### 3.3.12. Additional Manual Steps Per Release

Some OpenShift releases may have additional instructions specific to that release that must be performed to fully apply the updates across the cluster. Read through the following sections carefully depending on your upgrade path, as you may be required to perform certain steps at key points during the standard upgrade process described earlier in this topic.

See the [OpenShift Enterprise 3.1 Release Notes](#) to review the latest release notes.

#### 3.3.12.1. OpenShift Enterprise 3.1.0

There are no additional manual steps for these releases that are not already mentioned inline during the standard manual upgrade process.

#### 3.3.12.2. OpenShift Enterprise 3.1.1

There was an issue with OpenShift Enterprise 3.1.1 where hosts with host names that resolved to IP addresses that were not local to the host would run into problems with liveness and readiness probes on newly-created HAProxy routers. This was resolved in RHBA-2016:0293 by configuring the probes to use [IP addresses](#) that were local to the host.
localhost as the hostname for pods with hostPort values.

If you created a router under the affected version, and your liveness or readiness probes unexpectedly fail for your router, then add host: localhost:

```
# oc edit dc/router
```

Apply the following changes:

```yaml
spec:
  template:
    spec:
      containers:
        ...
        livenessProbe:
          httpGet:
            host: localhost
            path: /healthz
            port: 1936
            scheme: HTTP
            initialDelaySeconds: 10
            timeoutSeconds: 1
        ...
        readinessProbe:
          httpGet:
            host: localhost
            path: /healthz
            port: 1936
            scheme: HTTP
            timeoutSeconds: 1
```

1. Add 'host: localhost' to your liveness probe.
2. Add 'host: localhost' to your readiness probe.

3.3.12.3. OpenShift Enterprise 3.1.1.11

There are no additional manual steps for the upgrade to OpenShift Enterprise 3.1.1.11 that are not already mentioned inline during the standard manual upgrade process.

3.3.13. Verifying the Upgrade

To verify the upgrade, first check that all nodes are marked as Ready:

```
# oc get nodes
NAME                   STATUS    LABELS                                                                                       
master.example.com     Ready     kubernetes.io/hostname=master.example.com,region=infra,zone=default
node1.example.com      Ready     kubernetes.io/hostname=node1.example.com,region=primary,zone=east
```
Then, verify that you are running the expected versions of the **docker-registry** and **router** images, if deployed:

```bash
# oc get -n default dc/docker-registry -o json | grep "image"  
  "image": "openshift3/ose-docker-registry:v3.1.1.11",  
# oc get -n default dc/router -o json | grep "image"  
  "image": "openshift3/ose-haproxy-router:v3.1.1.11",
```

If you upgraded from OSE 3.0 to OSE 3.1, verify in your old `/etc/sysconfig/openshift-master` and `/etc/sysconfig/openshift-node` files that any custom configuration is added to your new `/etc/sysconfig/atomic-openshift-master` and `/etc/sysconfig/atomic-openshift-node` files.

After upgrading, you can use the experimental diagnostics tool to look for common issues:

```bash
# openshift ex diagnostics  
...  
[Note] Summary of diagnostics execution:  
[Note] Completed with no errors or warnings seen.
```

### 3.4. UPGRADING FROM PACEMAKER TO NATIVE HA

#### 3.4.1. Overview

If you are using the Pacemaker method for high availability (HA) masters, you can upgrade to the native HA method either using Ansible playbooks or manually. Both methods are described in the following sections.

#### 3.4.2. Using Ansible Playbooks

These steps assume that cluster has been upgraded to OpenShift Enterprise 3.1 using either the [manual](https://example.com) or [automated](https://example.com) method.

**WARNING**

Playbooks used for the Pacemaker to native HA upgrade will re-run cluster configuration steps, therefore any settings that are not stored in your inventory file will be overwritten. Back up any configuration files that have been modified since installation before beginning this upgrade.

#### 3.4.2.1. Modifying the Ansible Inventory

Your original Ansible inventory file's Pacemaker configuration contains a VIP and a cluster host name which should resolve to this VIP. Native HA requires a cluster host name which resolves to the load balancer being used.

Consider the following example configuration:

```bash
# Pacemaker high availability cluster method.
```
# Pacemaker HA environment must be able to self provision the configured VIP. For installation `openshift_master_cluster_hostname` must resolve to the configured VIP.
# `openshift_master_cluster_method=pacemaker`
# `openshift_master_cluster_password=openshift_cluster`
# `openshift_master_cluster_vip=192.168.133.35`
# `openshift_master_cluster_public_vip=192.168.133.35`
# `openshift_master_cluster_hostname=openshift-cluster.example.com`
# `openshift_master_cluster_public_hostname=openshift-cluster.example.com`

Remove or comment the above section in your inventory file. Then, add the following section, modifying the host names to match your current cluster host names:

# Native high availability cluster method with optional load balancer.
# If no lb group is defined, the installer assumes that a load balancer has been preconfigured. For installation the value of `openshift_master_cluster_hostname` must resolve to the load balancer or to one or all of the masters defined in the inventory if no load balancer is present.
`openshift_master_cluster_method=native`
`openshift_master_cluster_hostname=openshift-cluster.example.com`
`openshift_master_cluster_public_hostname=openshift-cluster.example.com`

Native HA requires a load balancer to balance the master API (port 8443). When modifying your inventory file, specify an `[lb]` group and add `lb` to the `[OSEv3:children]` section if you would like the playbooks to configure an HAProxy instance as the load balancer. This instance must be on a separate host from the masters and nodes:

```
[OSEv3:children]
masters
nodes
lb
...
[lb]
lb1.example.com
```

1. Host name of the HAProxy load balancer.

Any external load balancer may be used in place of the default HAProxy host, but it must be preconfigured and allow API traffic to masters on port 8443.

### 3.4.2.1.1 Destroying the Pacemaker Cluster

On any master, run the following to destroy the Pacemaker cluster.
WARNING
After the Pacemaker cluster has been destroyed, the OpenShift cluster will be in outage until the remaining steps are completed.

# pcs cluster destroy --all

3.4.2.2. Updating DNS

Modify your cluster host name DNS entries such that the host name used resolves to the load balancer that will be used with native HA.

In the earlier example configuration, openshift-cluster.example.com resolves to 192.168.133.35. DNS must be modified such that openshift-cluster.example.com now resolves to the load balancer host or to the master API balancer in an alternative load balancing solution.

3.4.2.3. Running the Ansible Playbook

You can now run the following Ansible playbook:

# ansible-playbook /usr/share/ansible/openshift-ansible/playbooks/bYo/config.yml

After the playbook finishes successfully, your upgrade to the native HA method is complete. Restore any configuration files if you backed up any that had been modified since installation, and restart any relevant OpenShift services, if necessary.

3.4.3. Manually Upgrading

These steps assume that cluster has been upgraded to OpenShift Enterprise 3.1 using either the manual or automated method. They also assume that you are bringing your own load balancer which has been configured to balance the master API on port 8443.

3.4.3.1. Creating Unit and System Configuration for New Services

The Systemd unit files for the atomic-openshift-master-api and atomic-openshift-master-controllers services are not yet provided by packaging. Ansible creates unit and system configuration when installing with the native HA method.
Therefore, create the following files:

**Example 3.1. */usr/lib/systemd/system/atomic-openshift-master-api.service* File**

```ini
[Unit]
Description=Atomic OpenShift Master API
Documentation=https://github.com/openshift/origin
After=network.target
After=etcd.service
Before=atomic-openshift-node.service
Requires=network.target

[Service]
Type=notify
EnvironmentFile=/etc/sysconfig/atomic-openshift-master-api
Environment=GOTRACEBACK=crash
ExecStart=/usr/bin/openshift start master api --config=${CONFIG_FILE} $OPTIONS
LimitNOFILE=131072
LimitCORE=infinity
WorkingDirectory=/var/lib/origin/
SyslogIdentifier=atomic-openshift-master-api

[Install]
WantedBy=multi-user.target
WantedBy=atomic-openshift-node.service
```

**Example 3.2. */usr/lib/systemd/system/atomic-openshift-master-controllers.service* File**

```ini
[Unit]
Description=Atomic OpenShift Master Controllers
Documentation=https://github.com/openshift/origin
After=network.target
After=atomic-openshift-master-api.service
Before=atomic-openshift-node.service
Requires=network.target

[Service]
Type=notify
EnvironmentFile=/etc/sysconfig/atomic-openshift-master-controllers
Environment=GOTRACEBACK=crash
ExecStart=/usr/bin/openshift start master controllers --config=${CONFIG_FILE} $OPTIONS
LimitNOFILE=131072
LimitCORE=infinity
WorkingDirectory=/var/lib/origin/
SyslogIdentifier=atomic-openshift-master-controllers
Restart=on-failure

[Install]
WantedBy=multi-user.target
WantedBy=atomic-openshift-node.service
```
Example 3.3. `/etc/sysconfig/atomic-openshift-master-api` File

```bash
OPTIONS=--loglevel=2
CONFIG_FILE=/etc/origin/master/master-config.yaml

# Proxy configuration
# Origin uses standard HTTP_PROXY environment variables. Be sure to set
# NO_PROXY for your master
#NO_PROXY=master.example.com
#HTTP_PROXY=http://USER:PASSWORD@IPADDR:PORT
#HTTPS_PROXY=https://USER:PASSWORD@IPADDR:PORT
```

Example 3.4. `/etc/sysconfig/atomic-openshift-master-controllers` File

```bash
OPTIONS=--loglevel=2
CONFIG_FILE=/etc/origin/master/master-config.yaml

# Proxy configuration
# Origin uses standard HTTP_PROXY environment variables. Be sure to set
# NO_PROXY for your master
#NO_PROXY=master.example.com
#HTTP_PROXY=http://USER:PASSWORD@IPADDR:PORT
#HTTPS_PROXY=https://USER:PASSWORD@IPADDR:PORT
```

Then, reload Systemd to pick up your changes:

```
# systemctl daemon-reload
```

### 3.4.3.2. Destroying the Pacemaker Cluster

On any master, run the following to destroy the Pacemaker cluster.

```
# pcs cluster destroy --all
```

#### WARNING

After the Pacemaker cluster has been destroyed, the OpenShift cluster will be in outage until the remaining steps are completed.

```
# pcs cluster destroy --all
```

### 3.4.3.3. Updating DNS

Modify your cluster host name DNS entries such that the host name used resolves to the load balancer that will be used with native HA.
For example, if the cluster host name is `openshift-cluster.example.com` and it resolved to a VIP of 192.168.133.35, then DNS must be modified such that `openshift-cluster.example.com` now resolves to the master API balancer.

### 3.4.3.4. Modifying Master and Node Configuration

Edit the master configuration in the `/etc/origin/master/master-config.yaml` file and ensure that `kubernetesMasterConfig.masterCount` is updated to the total number of masters. Perform this step on all masters:

```
Example 3.5. /etc/origin/master/master-config.yaml File
...
kubernetesMasterConfig:
  apiServerArguments:
  controllerArguments:
    masterCount: 3
...
```

Update this value to the total number of masters.

Edit the node configuration in the `/etc/origin/node/node-config.yaml` file and remove the `dnsIP` setting. OpenShift will use the Kubernetes service IP as the `dnsIP` by default. Perform this step on all nodes:

```
Example 3.6. /etc/origin/node/node-config.yaml File
...
  allowDisabledDocker: false
  apiVersion: v1
  dnsDomain: cluster.local
  dnsIP: 10.6.102.3
  dockerConfig:
    execHandlerName: ""
...
```

Remove this line.

### 3.4.3.4.1. Starting the API Service

Start and enable the API service on all masters:

```
# systemctl start atomic-openshift-master-api
# systemctl enable atomic-openshift-master-api
```

### 3.4.3.4.2. Starting the Controller Service

Start and enable the controllers service on all masters:

```
# systemctl start atomic-openshift-master-controllers
```
# systemctl enable atomic-openshift-master-controllers

After the service restarts, your upgrade to the native HA method is complete.

3.4.3.5. Modifying the Ansible Inventory

Optionally, modify your Ansible inventory file for future runs per the instructions above in the playbooks method.
CHAPTER 4. DOWNGRADING OPENSSHIFT

4.1. OVERVIEW

Following an OpenShift Enterprise upgrade, it may be desirable in extreme cases to downgrade your cluster to a previous version. The following sections outline the required steps for each system in a cluster to perform such a downgrade, currently supported for the OpenShift Enterprise 3.1 to 3.0 downgrade path.

4.2. VERIFYING BACKUPS

The Ansible playbook used during the upgrade process should have created a backup of the master-config.yaml file and the etcd data directory. Ensure these exist on your masters and etcd members:

```
/etc/openshift/master/master-config.yaml.<timestamp>
/var/lib/openshift/etcd-backup-<timestamp>
```

If you use a separate etcd cluster instead of a single embedded etcd instance, the backup is likely created on all etcd members, though only one is required for the recovery process. You can run a separate etcd instance that is co-located with your master nodes.

The RPM downgrade process in a later step should create .rpmsave backups of the following files, but it may be a good idea to keep a separate copy regardless:

```
/etc/sysconfig/openshift-master
/etc/etcd/etcd.conf
```

1 Only required if using a separate etcd cluster.

4.3. SHUTTING DOWN THE CLUSTER

On all masters, nodes, and etcd members, if you use a separate etcd cluster that runs on different nodes, ensure the relevant services are stopped:

```
# systemctl stop atomic-openshift-master
# systemctl stop atomic-openshift-node
# systemctl stop etcd
```

1 Only required if using external etcd.

4.4. REMOVING RPMS

On all masters, nodes, and etcd members (if using an external etcd cluster), remove the following packages:

```
# yum remove atomic-openshift \ atomic-openshift-clients \ atomic-openshift-node \ atomic-openshift-master \
```
If you are using external etcd, also remove the `etcd` package:

```
# yum remove etcd
```

For embedded etcd, you can leave the `etcd` package installed, as the package is only required so that the `etcdctl` command is available to issue operations in later steps.

### 4.5. REINSTALLING RPMS

Disable the OpenShift Enterprise 3.1 repositories, and re-enable the 3.0 repositories:

```
# subscription-manager repos \
   --disable=rhel-7-server-ose-3.1-rpms \
   --enable=rhel-7-server-ose-3.0-rpms
```

On each master, install the following packages:

```
# yum install openshift \
   openshift-master \
   openshift-node \
   openshift-sdn-ovs
```

On each node, install the following packages:

```
# yum install openshift \
   openshift-node \
   openshift-sdn-ovs
```

If using a separate etcd cluster, install the following package on each etcd member:

```
# yum install etcd
```

### 4.6. RESTORING ETCD

Whether using embedded or external etcd, you must first restore the etcd backup by creating a new, single node etcd cluster. If using external etcd with multiple members, you must then also add any additional etcd members to the cluster one by one.

However, the details of the restoration process differ between `embedded` and `external` etcd. See the following section that matches your etcd configuration and follow the relevant steps before continuing to Bringing OpenShift Services Back Online.

#### 4.6.1. Embedded etcd

Restore your etcd backup and configuration:

1. Run the following on the master with the embedded etcd:
WARNING

The $ETCD_DIR$ location differs between external and embedded etcd.

2. Create the new, single node etcd cluster:

```
# etcd -data-dir=/var/lib/openshift/openshift.local.etcd -force-new-cluster
```

Verify etcd has started successfully by checking the output from the above command, which should look similar to the following at the end:

```
[...]  
2016/01/8 13:24:21 etcdserver: starting server... [version: 2.1.1, cluster version: 2.1.0]  
2016/01/8 13:24:22 raft: 5168c093630001 is starting a new election at term 13  
2016/01/8 13:24:22 raft: 5168c093630001 became candidate at term 14  
2016/01/8 13:24:22 raft: 5168c093630001 received vote from 5168c093630001 at term 14  
2016/01/8 13:24:22 raft: 5168c093630001 became leader at term 14  
2016/01/8 13:24:22 raft.node: 5168c093630001 elected leader 5168c093630001 at term 14  
```

3. Shut down the process by running the following from a separate terminal:

```
# pkill etcd
```

4. Continue to Bringing OpenShift Services Back Online.

4.6.2. External etcd

Choose a system to be the initial etcd member, and restore its etcd backup and configuration:

1. Run the following on the etcd host:

```
# ETCD_DIR=/var/lib/etcd/  
# mv $ETCD_DIR /var/lib/etcd.orig  
# cp -Rp /var/lib/openshift/etcd-backup-<timestamp>/ $ETCD_DIR  
# chcon -R --reference /var/lib/etcd.orig/ $ETCD_DIR  
# chown -R etcd:etcd $ETCD_DIR
```
# chcon -R --reference /var/lib/etcd.orig/ $ETCD_DIR
# chown -R etcd:etcd $ETCD_DIR

## WARNING

The **$ETCD_DIR** location differs between external and embedded etcd.

2. Restore your `/etc/etcd/etcd.conf` file from backup or `.rpmsave`.

3. Create the new single node cluster using etcd’s `--force-new-cluster` option. You can do this with a long complex command using the values from the `/etc/etcd/etcd.conf`, or you can temporarily modify the `systemd` file and start the service normally. To do so, edit the `/usr/lib/systemd/system/etcd.service` and add `--force-new-cluster`:

   ```bash
   # sed -i '/ExecStart/s/"$/  --force-new-cluster"/'
   /usr/lib/systemd/system/etcd.service
   # cat /usr/lib/systemd/system/etcd.service  | grep ExecStart
   ExecStart=/bin/bash -c "GOMAXPROCS=$(nproc) /usr/bin/etcd --force-new-cluster"
   ``

   Then restart the `etcd` service:

   ```bash
   # systemctl daemon-reload
   # systemctl start etcd
   ``

4. Verify the `etcd` service started correctly, then re-edit the `/usr/lib/systemd/system/etcd.service` file and remove the `--force-new-cluster` option:

   ```bash
   # sed -i '/ExecStart/s/ --force-new-cluster//'
   /usr/lib/systemd/system/etcd.service
   # cat /usr/lib/systemd/system/etcd.service  | grep ExecStart
   ExecStart=/bin/bash -c "GOMAXPROCS=$(nproc) /usr/bin/etcd"
   ``

5. Restart the `etcd` service, then verify the etcd cluster is running correctly and displays OpenShift’s configuration:

   ```bash
   # systemctl daemon-reload
   # systemctl restart etcd
   # etcdctl --cert-file=/etc/etcd/peer.crt \
          --key-file=/etc/etcd/peer.key \
          --ca-file=/etc/etcd/ca.crt \
          --peers="https://172.16.4.18:2379,https://172.16.4.27:2379" \
          ls /
   ```
6. If you have additional etcd members to add to your cluster, continue to **Adding Additional etcd Members**. Otherwise, if you only want a single node external etcd, continue to **Bringing OpenShift Services Back Online**.

### 4.6.2.1. Adding Additional etcd Members

To add additional etcd members to the cluster, you must first adjust the default **localhost peerURLs** for the first member:

1. Get the member ID for the first member using the `member list` command:

   ```
   # etcdctl --cert-file=/etc/etcd/peer.crt 
   --key-file=/etc/etcd/peer.key 
   --ca-file=/etc/etcd/ca.crt 
   member list
   ```

2. Update the **peerURLs**. In etcd 2.2 and beyond, this can be done with the `etcdctl member update` command. However, OpenShift Enterprise 3.1 uses etcd 2.1, so you must use `curl`:

   ```
   # curl --cacert /etc/etcd/ca.crt 
   --cert /etc/etcd/peer.crt 
   --key /etc/etcd/peer.key 
   https://172.18.1.18:2379/v2/members/511b7fb6cc0001 
   -XPUT -H "Content-Type: application/json" 
   -d '{"peerURLs":["https://172.18.1.18:2380"]}'
   ```

3. Re-run the `member list` command and ensure the **peerURLs** no longer points to **localhost**.

4. Now add each additional member to the cluster, one at a time.

   **WARNING**

   Each member must be fully added and brought online one at a time. When adding each additional member to the cluster, the **peerURLs** list must be correct for that point in time, so it will grow by one for each member added. The `etcdctl member add` command will output the values that need to be set in the **etcd.conf** file as you add each member, as described in the following instructions.

   a. For each member, add it to the cluster using the values that can be found in that system's **etcd.conf** file:

      ```
      # etcdctl --cert-file=/etc/etcd/peer.crt 
      --key-file=/etc/etcd/peer.key 
      --ca-file=/etc/etcd/ca.crt 
      ```
member add 10.3.9.222 https://172.16.4.27:2380

Added member named 10.3.9.222 with ID 4e1db163a21d7651 to cluster

ETCD_NAME="10.3.9.222"
ETCD_INITIAL_CLUSTER="10.3.9.221=https://172.16.4.18:2380,10.3.9.222=https://172.16.4.27:2380"
ETCD_INITIAL_CLUSTER_STATE="existing"

b. Using the environment variables provided in the output of the above `etcdctl member add` command, edit the `/etc/etcd/etc.conf` file on the member system itself and ensure these settings match.

c. Now start etcd on the new member:

```
# rm -rf /var/lib/etcd/member
# systemctl enable etcd
# systemctl start etcd
```

d. Ensure the service starts correctly and the etcd cluster is now healthy:

```
# etcdctl --cert-file=/etc/etcd/peer.crt \
--key-file=/etc/etcd/peer.key \
--ca-file=/etc/etcd/ca.crt \
--peers="https://172.16.4.18:2379,https://172.16.4.27:2379" \
member list

51251b34b80001: name=10.3.9.221 peerURLs=https://172.16.4.18:2380 
clientURLs=https://172.16.4.18:2379
d266df286a41a8a4: name=10.3.9.222 
peerURLs=https://172.16.4.27:2380 
clientURLs=https://172.16.4.27:2379

# etcdctl --cert-file=/etc/etcd/peer.crt \
--key-file=/etc/etcd/peer.key \
--ca-file=/etc/etcd/ca.crt \
--peers="https://172.16.4.18:2379,https://172.16.4.27:2379" \
cluster-health

cluster is healthy
member 51251b34b80001 is healthy 
member d266df286a41a8a4 is healthy
```

e. Now repeat this process for the next member to add to the cluster.

5. After all additional etcd members have been added, continue to Bringing OpenShift Services Back Online.

### 4.7. BRINGING OPENSHEET SERVICES BACK ONLINE

On each OpenShift master, restore your `openshift-master` configuration from backup and restart relevant services:

```
# cp /etc/sysconfig/openshift-master.rpmsave /etc/sysconfig/openshift-
```
master
# cp /etc/openshift/master/master-config.yaml.2015-11-20\@08\:36\:51~
/etc/openshift/master/master-config.yaml
# systemctl enable openshift-master
# systemctl enable openshift-node
# systemctl start openshift-master
# systemctl start openshift-node

On each OpenShift node, enable and restart the **openshift-node** service:

# systemctl enable openshift-node
# systemctl start openshift-node

Your OpenShift cluster should now be back online.
CHAPTER 5. MASTER AND NODE CONFIGURATION

5.1. OVERVIEW

The `openshift start` command is used to launch OpenShift servers. The command and its subcommands (master to launch a master server and node to launch a node server) all take a limited set of arguments that are sufficient for launching servers in a development or experimental environment.

However, these arguments are insufficient to describe and control the full set of configuration and security options that are necessary in a production environment. To provide those options, it is necessary to use the dedicated master and node configuration files.

Master configuration files and node configuration files are fully specified with no default values. Therefore, any empty value indicates that you want to start up with an empty value for that parameter. This makes it easy to reason about exactly what your configuration is, but it also makes it difficult to remember all of the options to specify. To make this easier, the configuration files can be created with the `--write-config` option and then used with the `--config` option.

5.2. CREATING NEW CONFIGURATION FILES

For masters, the `openshift start` command accepts options that indicate that it should simply write the configuration files that it would have used, then terminate. For nodes, a configuration file can be written using the `oadm create-node-config` command. Creating new configuration files is useful to get a starting point for defining your configuration.

The following commands write the relevant launch configuration file(s), certificate files, and any other necessary files to the specified `--write-config` or `--node-dir` directory.

To create configuration files for an all-in-one server (a master and a node on the same host) in the specified directory:

```
$ openshift start --write-config=/openshift.local.config
```

To create a master configuration file and other required files in the specified directory:

```
$ openshift start master --write-config=/openshift.local.config/master
```

To create a node configuration file and other related files in the specified directory:

```
$ oadm create-node-config --node-dir=/openshift.local.config/node-<node_hostname> --node=<node_hostname> --hostnames=<hostname>,<ip_address>
```

For the `--hostnames` option in the above command, use a comma-delimited list of every host name or IP address you want server certificates to be valid for. The above command also assumes that certificate files are located in an `openshift.local.config/master/` directory. If they are not, you can include options to specify their location. Run the command with the `-h` option to see details.

5.3. LAUNCHING SERVERS USING CONFIGURATION FILES

Once you have modified the master and/or node configuration files to your specifications, you can use them when launching servers by specifying them as an argument. Keep in mind that if you specify a configuration file, none of the other command line options you pass are respected.
To launch an all-in-one server using a master configuration and a node configuration file:

```bash
$ openshift start --master-config=/openshift.local.config/master/master-config.yaml --node-config=/openshift.local.config/node-<node_hostname>/node-config.yaml
```

To launch a master server using a master configuration file:

```bash
$ openshift start master --config=/openshift.local.config/master/master-config.yaml
```

To launch a node server using a node configuration file:

```bash
$ openshift start node --config=/openshift.local.config/node-<node_hostname>/node-config.yaml
```

### 5.4. MASTER CONFIGURATION FILES

The following `master-config.yaml` file is a sample master configuration file that was generated with the default values as of writing. You can create a new master configuration file to see the valid options for your installed version of OpenShift.

**Example 5.1. Sample Master Configuration File**

```yaml
apiLevels:
- v1beta3
- v1
apiServerArguments:
  event-ttl: 1
  "15m"
apiVersion: v1
assetConfig:
  logoutURL: ""
  masterPublicURL: https://10.0.2.15:8443
  publicURL: https://10.0.2.15:8443/console/
  servingInfo:
    bindAddress: 0.0.0.0:8443
    certFile: master.server.crt
    clientCA: ""
    keyFile: master.server.key
    maxRequestsInFlight: 0
    requestTimeoutSeconds: 0
  controllers: '*'
corsAllowedOrigins:
- 10.0.2.15:8443
- 127.0.0.1
- localhost
disabledFeatures: null
dnsConfig: allowRecursiveQueries: false
  bindAddress: 0.0.0.0:8053
  bindNetwork: tcp4
etcdClientInfo:
  ca: ca.crt
  certFile: master.etcd-client.crt
```

OpenShift Enterprise 3.1 Installation and Configuration
keyFile: master.etcd-client.key
urls:
- https://10.0.2.15:4001

etcdConfig:
  address: 10.0.2.15:4001
  peerAddress: 10.0.2.15:7001
  peerServingInfo:
    bindAddress: 0.0.0.0:7001
    certFile: etcd.server.crt
    clientCA: ca.crt
    keyFile: etcd.server.key
  servingInfo:
    bindAddress: 0.0.0.0:4001
    certFile: etcd.server.crt
    clientCA: ca.crt
    keyFile: etcd.server.key
  storageDirectory: /root/openshift.local.etcd

etcdStorageConfig:
  kubernetesStoragePrefix: kubernetes.io
  kubernetesStorageVersion: v1
  openShiftStoragePrefix: openshift.io
  openShiftStorageVersion: v1

imageConfig:
  format: openshift/origin-{$component}:${{$version}}
  latest: false

kind: MasterConfig
kubeletClientInfo:
  ca: ca.crt
  certFile: master.kubelet-client.crt
  keyFile: master.kubelet-client.key
  port: 10250

kubernetesMasterConfig:
  apiLevels:
    - v1beta3
    - v1
  apiServerArguments: null
  controllerArguments: null
  masterCount: 1
  masterIP: 10.0.2.15
  podEvictionTimeout: 5m
  schedulerConfigFile: ""
  servicesNodePortRange: 30000-32767
  servicesSubnet: 172.30.0.0/16
  staticNodeNames: []

masterClients:
  externalKubernetesKubeConfig: ""
  openshiftLoopbackKubeConfig: openshift-master.kubeconfig

masterPublicURL: https://10.0.2.15:8443

networkConfig:
  clusterNetworkCIDR: 10.1.0.0/16
  hostSubnetLength: 8
  networkPluginName: ""
  serviceNetworkCIDR: 172.30.0.0/16

oauthConfig:
  assetPublicURL: https://10.0.2.15:8443/console/
  grantConfig:
Prevents memory overload by defining a lower limit (for example, 15m) for stored events. This prevents excessive memory growth that can be the result of etcd storing too many events. By default, this is set to two hours.

5.5. NODE CONFIGURATION FILES

```yaml
method: auto
identityProviders:
- challenge: true
  login: true
  name: anypassword
  provider:
    apiVersion: v1
    kind: AllowAllPasswordIdentityProvider
masterPublicURL: https://10.0.2.15:8443
masterURL: https://10.0.2.15:8443
sessionConfig:
  sessionMaxAgeSeconds: 300
  sessionName: ssn
  sessionSecretsFile: ""
tokenConfig:
  accessTokenMaxAgeSeconds: 86400
  authorizeTokenMaxAgeSeconds: 300
policyConfig:
  bootstrapPolicyFile: policy.json
  openshiftInfrastructureNamespace: openshift-infra
  openshiftSharedResourcesNamespace: openshift
projectConfig:
  defaultNodeSelector: ""
  projectRequestMessage: ""
  projectRequestTemplate: ""
  securityAllocator:
    mcsAllocatorRange: s0:/2
    mcsLabelsPerProject: 5
    uidAllocatorRange: 1000000000-1999999999/10000
routingConfig:
  subdomain: router.default.svc.cluster.local
serviceAccountConfig:
  managedNames:
  - default
  - builder
  - deployer
  masterCA: ca.crt
  privateKeyFile: serviceaccounts.private.key
  publicKeyFiles:
  - serviceaccounts.public.key
servingInfo:
  bindAddress: 0.0.0.0:8443
  certFile: master.server.crt
  clientCA: ca.crt
  keyFile: master.server.key
  maxRequestsInFlight: 0
  requestTimeoutSeconds: 3600
```
The following `node-config.yaml` file is a sample node configuration file that was generated with the default values as of writing. You can create a new node configuration file to see the valid options for your installed version of OpenShift.

**Example 5.2. Sample Node Configuration File**

```yaml
allowDisabledDocker: true
apiVersion: v1
dnsDomain: cluster.local
dnsIP: 10.0.2.15
dockerConfig:
  execHandlerName: native
imageConfig:
  format: openshift/origin-${component}:${version}
  latest: false
kind: NodeConfig
masterKubeConfig: node.kubeconfig
networkConfig:
  mtu: 1450
  networkPluginName: ""
nodeIP: ""
nnodeName: node1.example.com
podManifestConfig: 1
  path: "/path/to/pod-manifest-file" 2
  fileCheckIntervalSeconds: 30 3
servingInfo:
  bindAddress: 0.0.0.0:10250
certFile: server.crt
clientCA: node-client-ca.crt
keyFile: server.key
volumeDirectory: /root/openshift.local.volumes
```

1. Allows pods to be placed directly on certain set of nodes, or on all nodes without going through the scheduler. You can then use pods to perform the same administrative tasks and support the same services on each node.

2. Specifies the path for the pod manifest file or directory. If it is a directory, then it is expected to contain one or more manifest files. This is used by the Kubelet to create pods on the node.

3. This is the interval (in seconds) for checking the manifest file for new data. The interval must be a positive value.
CHAPTER 6. LOADING THE DEFAULT IMAGE STREAMS AND TEMPLATES

6.1. OVERVIEW

Your OpenShift installation includes useful sets of Red Hat-provided image streams and templates to make it easy for developers to create new applications. By default, the quick and advanced installation methods automatically create these sets in the openshift project, which is a default global project to which all users have view access.

IMPORTANT

Before you consider using this topic, confirm if these image streams and templates are already registered in your OpenShift cluster by doing one of the following:

- Log into the web console and click Add to Project.
- List them for the openshift project using the CLI:

  ```
  $ oc get is -n openshift
  $ oc get templates -n openshift
  ```

If the default image streams and templates are ever removed or changed, you can follow this topic to create the default objects yourself. Otherwise, the following instructions are not necessary.

The core set of image streams and templates are provided and supported by Red Hat with an active OpenShift Enterprise subscription for the following technologies:

<table>
<thead>
<tr>
<th>Languages</th>
<th>Node.js</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Perl</td>
</tr>
<tr>
<td></td>
<td>PHP</td>
</tr>
<tr>
<td></td>
<td>Python</td>
</tr>
<tr>
<td></td>
<td>Ruby</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Database</th>
<th>MongoDB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MySQL</td>
</tr>
<tr>
<td></td>
<td>PostgreSQL</td>
</tr>
</tbody>
</table>

| Other Services  | Jenkins   |

If you also have the relevant xPaaS Middleware subscription active on your account, image streams and templates are also provided and supported by Red Hat for each of following middleware services:
6.2. PREREQUISITES

Before you can create the default image streams and templates:

- The integrated Docker registry service must be deployed in your OpenShift installation.
- You must be able to run the `oc create` command with `cluster-admin` privileges, because they operate on the default `openshift` project.
- You must have installed the `atomic-openshift-utils` RPM package. See Software Prerequisites for instructions.
- Define shell variables for the directories containing image streams and templates. This significantly shortens the commands in the following sections. To do this:

  ```bash
  $ IMAGESTREAMDIR="/usr/share/ansible/openshift-ansible/roles/openshift_examples/files/examples/v1.1/image-streams";
  \n  XPAASSTREAMDIR="/usr/share/ansible/openshift-ansible/roles/openshift_examples/files/examples/v1.1/xpaas-streams";
  \n  XPAATESTEMPLATES="/usr/share/ansible/openshift-ansible/roles/openshift_examples/files/examples/v1.1/xpaas-templates";
  \n  DBTEMPLATES="/usr/share/ansible/openshift-ansible/roles/openshift_examples/files/examples/v1.1/db-templates";
  \n  QSTEMPLATES="/usr/share/ansible/openshift-ansible/roles/openshift_examples/files/examples/v1.1/quickstart-templates"
  ```

6.3. CREATING IMAGE STREAMS FOR OPENSHIFT IMAGES

The core set of image streams provide images that can be used to build Node.js, Perl, PHP, Python, and Ruby applications. It also defines images for MongoDB, MySQL, and PostgreSQL to support data storage.

If your node hosts are subscribed using Red Hat Subscription Manager and you want to use the Red Hat Enterprise Linux (RHEL) 7 based images:

```bash
$ oc create -f $IMAGESTREAMDIR/image-streams-rhel7.json -n openshift
```
Alternatively, to create the core set of image streams that use the CentOS 7 based images:

```
$ oc create -f $IMAGESTREAMDIR/image-streams-centos7.json -n openshift
```

It is not possible to create both the CentOS and RHEL sets of image streams because they use the same names. If you desire to have both sets of image streams available to users, either create one set in a different project, or edit one of the files and modify the image stream names to make them unique.

### 6.4. CREATING IMAGE STREAMS FOR XPAAS MIDDLEWARE IMAGES

The xPaaS Middleware image streams provide images for JBoss EAP, JBoss JWS, JBoss A-MQ, JBoss Fuse Integration Services, Decision Server, and JBoss Data Grid. They can be used to build applications for those platforms using the provided templates.

To create the xPaaS Middleware set of image streams:

```
$ oc create -f $XPAASSTREAMDIR/jboss-image-streams.json -n openshift
```

**NOTE**

Access to the images referenced by these image streams requires the relevant xPaaS Middleware subscriptions.

### 6.5. CREATING DATABASE SERVICE TEMPLATES

The database service templates make it easy to run a database image which can be utilized by other components. For each database (MongoDB, MySQL, and PostgreSQL), two templates are defined.

One template uses ephemeral storage in the container which means data stored will be lost if the container is restarted, for example if the pod moves. This template should be used for demonstration purposes only.

The other template defines a persistent volume for storage, however it requires your OpenShift installation to have persistent volumes configured.

To create the core set of database templates:

```
$ oc create -f $DBTEMPLATES -n openshift
```

After creating the templates, users are able to easily instantiate the various templates, giving them quick access to a database deployment.

### 6.6. CREATING INSTANT APP AND QUICKSTART TEMPLATES

The Instant App and Quickstart templates define a full set of objects for a running application. These include:

- **Build configurations** to build the application from source located in a GitHub public repository
- **Deployment configurations** to deploy the application image after it is built.
- **Services** to provide load balancing for the application pods.

- **Routes** to provide external access to the application.

Some of the templates also define a database deployment and service so the application can perform database operations.

**NOTE**

The templates which define a database use ephemeral storage for the database content. These templates should be used for demonstration purposes only as all database data will be lost if the database pod restarts for any reason.

After creating the templates, users are able to easily instantiate full applications using the various language images provided with OpenShift. They can also customize the template parameters during instantiation so that it builds source from their own repository rather than the sample repository, so this provides a simple starting point for building new applications.

To create the core Instant App and Quickstart templates:

```bash
$ oc create -f $QSTEMPLATES -n openshift
```

There is also a set of templates for creating applications using various xPaaS Middleware products (**JBoss EAP**, **JBoss JWS**, **JBoss A-MQ**, **JBoss Fuse Integration Services**, **Decision Server**, and **JBoss Data Grid**), which can be registered by running:

```bash
$ oc create -f $XPAASTEMPLATES -n openshift
```

**NOTE**

The xPaaS Middleware templates require the xPaaS Middleware image streams, which in turn require the relevant xPaaS Middleware subscriptions.

**NOTE**

The templates which define a database use ephemeral storage for the database content. These templates should be used for demonstration purposes only as all database data will be lost if the database pod restarts for any reason.

### 6.7. WHAT’S NEXT?

With these artifacts created, developers can now log into the web console and follow the flow for creating from a template. Any of the database or application templates can be selected to create a running database service or application in the current project. Note that some of the application templates define their own database services as well.

The example applications are all built out of GitHub repositories which are referenced in the templates by default, as seen in the **SOURCE_REPOSITORY_URL** parameter value. Those repositories can be forked, and the fork can be provided as the **SOURCE_REPOSITORY_URL** parameter value when creating from the templates. This allows developers to experiment with creating their own applications.

You can direct your developers to the Using the Instant App and Quickstart Templates section in the Developer Guide for these instructions.
CHAPTER 7. CONFIGURING CUSTOM CERTIFICATES

7.1. OVERVIEW

Administrators can configure custom serving certificates for the public host names of the OpenShift API and web console. This can be done during an advanced installation or configured after installation.

7.2. CONFIGURING CUSTOM CERTIFICATES

The namedCertificates section may be listed in the servingInfo and assetConfig.servingInfo sections of the master configuration file or in the servingInfo section of the node configuration file. Multiple certificates can be configured this way and each certificate may be associated with multiple host names or wildcards.

A default certificate must be configured in the servingInfo.certFile and servingInfo.keyFile configuration sections in addition to namedCertificates.

NOTE

The namedCertificates section should only be configured for the host name associated with the masterPublicURL, assetConfig.publicURL, and oauthConfig.assetPublicURL settings. Using a custom serving certificate for the host name associated with the masterURL will result in TLS errors as infrastructure components will attempt to contact the master API using the internal masterURL host.

Example 7.1. Custom Certificates Configuration

```yaml
servingInfo:
  ...
  namedCertificates:
    - certFile: custom.crt
      keyFile: custom.key
      names:
        - "customhost.com"
        - "api.customhost.com"
        - "console.customhost.com"
    - certFile: wildcard.crt
      keyFile: wildcard.key
      names:
        - "*.wildcardhost.com"
  ...
```

Relative paths are resolved relative to the master configuration file. Restart the server to pick up the configuration changes.
CHAPTER 8. CONFIGURING AUTHENTICATION

8.1. OVERVIEW

The OpenShift master includes a built-in OAuth server. Developers and administrators obtain OAuth access tokens to authenticate themselves to the API.

As an administrator, you can configure OAuth using the master configuration file to specify an identity provider. If you installed OpenShift Enterprise using the Quick Installation or Advanced Installation method, the Deny All identity provider is used by default, which denies access for all user names and passwords. To allow access, you must choose a different identity provider and configure the master configuration file appropriately (located at /etc/origin/master/master-config.yaml by default).

When running a master without a configuration file, the Allow All identity provider is used by default, which allows any non-empty user name and password to log in. This is useful for testing purposes. To use other identity providers, or to modify any token, grant, or session options, you must run the master from a configuration file.

NOTE

Roles need to be assigned to administer the setup with an external user.

8.2. IDENTITY PROVIDERS

You can configure the master host for authentication using your desired identity provider by modifying the master configuration file. The following sections detail the identity providers supported by OpenShift.

There are four parameters common to all identity providers:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The provider name is prefixed to provider user names to form an identity name.</td>
</tr>
<tr>
<td>challenge</td>
<td>When true, unauthenticated token requests from non-web clients (like the CLI) are sent a WWW-Authenticate challenge header. Not supported by all identity providers. To prevent cross-site request forgery (CSRF) attacks against browser clients Basic authentication challenges are only sent if a X-CSRF-Token header is present on the request. Clients that expect to receive Basic WWW-Authenticate challenges should set this header to a non-empty value.</td>
</tr>
<tr>
<td>login</td>
<td>When true, unauthenticated token requests from web clients (like the web console) are redirected to a login page backed by this provider. Not supported by all identity providers.</td>
</tr>
<tr>
<td>mappingMethod</td>
<td>Defines how new identities are mapped to users when they login. See Mapping Identities to Users for more information.</td>
</tr>
</tbody>
</table>


NOTE

When adding or changing identity providers, you can map identities from the new provider to existing users by setting the `mappingMethod` parameter to `add`.

### 8.2.1. Mapping Identities to Users

Setting the `mappingMethod` parameter in a master configuration file determines how identities are mapped to users:

```yaml
... 
oauthConfig:
  identityProviders:
    - name: htpasswd_auth
      challenge: true
      login: false
      mappingMethod: "claim"
... 
```

When set to the default `claim` value, OAuth will fail if the identity is mapped to a previously-existing user name. The following table outlines the use cases for the available `mappingMethod` parameter values:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>claim</td>
<td>The default value. Provisions a user with the identity’s preferred user name. Fails if a user with that user name is already mapped to another identity.</td>
</tr>
<tr>
<td>lookup</td>
<td>Looks up an existing identity, user identity mapping, and user, but does not automatically provision users or identities. This allows cluster administrators to set up identities and users manually, or using an external process.</td>
</tr>
<tr>
<td>generate</td>
<td>Provisions a user with the identity’s preferred user name. If a user with the preferred user name is already mapped to an existing identity, a unique user name is generated. For example, <code>myuser2</code>. This method should not be used in combination with external processes that require exact matches between OpenShift user names and identity provider user names, such as LDAP group sync.</td>
</tr>
<tr>
<td>add</td>
<td>Provisions a user with the identity’s preferred user name. If a user with that user name already exists, the identity is mapped to the existing user, adding to any existing identity mappings for the user. Required when multiple identity providers are configured that identify the same set of users and map to the same user names.</td>
</tr>
</tbody>
</table>

### 8.2.2. Allow All

Set `AllowAllPasswordIdentityProvider` in the `identityProviders` stanza to allow any non-empty user name and password to log in. This is the default identity provider when running OpenShift without a master configuration file.

**Example 8.1. Master Configuration Using AllowAllPasswordIdentityProvider**

```yaml
oauthConfig:
  ... 
```
identityProviders:
- name: my_allow_provider
  challenge: true
  login: true
  mappingMethod: claim
  provider:
    apiVersion: v1
    kind: AllowAllPasswordIdentityProvider

1. This provider name is prefixed to provider user names to form an identity name.
2. When true, unauthenticated token requests from non-web clients (like the CLI) are sent a WWW-Authenticate challenge header for this provider.
3. When true, unauthenticated token requests from web clients (like the web console) are redirected to a login page backed by this provider.
4. Controls how mappings are established between this provider’s identities and user objects, as described above.

8.2.3. Deny All

Set DenyAllPasswordIdentityProvider in the identityProviders stanza to deny access for all user names and passwords.

Example 8.2. Master Configuration Using DenyAllPasswordIdentityProvider

oauthConfig:
  ...
  identityProviders:
    - name: my_deny_provider
      challenge: true
      login: true
      mappingMethod: claim
      provider:
        apiVersion: v1
        kind: DenyAllPasswordIdentityProvider

1. This provider name is prefixed to provider user names to form an identity name.
2. When true, unauthenticated token requests from non-web clients (like the CLI) are sent a WWW-Authenticate challenge header for this provider.
3. When true, unauthenticated token requests from web clients (like the web console) are redirected to a login page backed by this provider.
4. Controls how mappings are established between this provider’s identities and user objects, as described above.
8.2.4. HTPasswd

Set `HTPasswdPasswordIdentityProvider` in the `identityProviders` stanza to validate user names and passwords against a flat file generated using `htpasswd`.

**NOTE**

The `htpasswd` utility is in the `httpd-tools` package:

```
# yum install httpd-tools
```

Only MD5, bcrypt, and SHA encryption types are supported. MD5 encryption is recommended, and is the default for `htpasswd`. Plaintext and crypt hashes are not currently supported.

The flat file is re-read if its modification time changes, without requiring a server restart.

To create the file, run:

```
$ htpasswd -c </path/to/users.htpasswd> <user_name>
```

To add or update a login to the file, run:

```
$ htpasswd </path/to/users.htpasswd> <user_name>
```

To remove a login from the file, run:

```
$ htpasswd -D </path/to/users.htpasswd> <user_name>
```

**Example 8.3. Master Configuration Using HTPasswdPasswordIdentityProvider**

```
oauthConfig:
  ...
  identityProviders:
  - name: my_htpasswd_provider
    challenge: true
    login: true
    mappingMethod: claim
    provider:
      apiVersion: v1
      kind: HTPasswdPasswordIdentityProvider
      file: /path/to/users.htpasswd
```

1. This provider name is prefixed to provider user names to form an identity name.
2. When `true`, unauthenticated token requests from non-web clients (like the CLI) are sent a `WWW-Authenticate` challenge header for this provider.
3. When `true`, unauthenticated token requests from web clients (like the web console) are redirected to a login page backed by this provider.
4. Controls how mappings are established between this provider’s identities and user objects, as described above.
8.2.5. Keystone

Set `KeystonePasswordIdentityProvider` in the `identityProviders` stanza to validate user names and passwords against an OpenStack Keystone v3 server. This enables shared authentication with an OpenStack server configured to store users in an internal Keystone database.

Example 8.4. Master Configuration Using `KeystonePasswordIdentityProvider`

```yaml
oauthConfig:
  ...
  identityProviders:
  - name: my_keystone_provider
    challenge: true
    login: true
    mappingMethod: claim
    provider:
      apiVersion: v1
      kind: KeystonePasswordIdentityProvider
      domainName: default
      ca: ca.pem
      certFile: keystone.pem
      keyFile: keystonekey.pem
```

1. This provider name is prefixed to provider user names to form an identity name.
2. When `true`, unauthenticated token requests from non-web clients (like the CLI) are sent a `WWW-Authenticate` challenge header for this provider.
3. When `true`, unauthenticated token requests from web clients (like the web console) are redirected to a login page backed by this provider.
4. Controls how mappings are established between this provider’s identities and user objects, as described above.
5. Keystone domain name. In Keystone, usernames are domain-specific. Only a single domain is supported.
6. Optional: Certificate bundle to use to validate server certificates for the configured URL.
7. Optional: Client certificate to present when making requests to the configured URL.
8. Key for the client certificate. Required if `certFile` is specified.

8.2.6. LDAP Authentication

Set `LDAPPasswordIdentityProvider` in the `identityProviders` stanza to validate user names and passwords against an LDAPv3 server, using simple bind authentication.
During authentication, the LDAP directory is searched for an entry that matches the provided user name. If a single unique match is found, a simple bind is attempted using the distinguished name (DN) of the entry plus the provided password. Here are the steps taken:

1. Generate a search filter by combining the attribute and filter in the configured **url** with the user-provided user name.

2. Search the directory using the generated filter. If the search does not return exactly one entry, deny access.

3. Attempt to bind to the LDAP server using the DN of the entry retrieved from the search, and the user-provided password.

4. If the bind is unsuccessful, deny access.

5. If the bind is successful, build an identity using the configured attributes as the identity, email address, display name, and preferred user name.

The configured **url** is an RFC 2255 URL, which specifies the LDAP host and search parameters to use. The syntax of the URL is:

```
```

For the above example:

<table>
<thead>
<tr>
<th>URL Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ldap</strong></td>
<td>For regular LDAP, use the string ldap. For secure LDAP (LDAPS), use ldaps instead.</td>
</tr>
<tr>
<td><strong>host:port</strong></td>
<td>The name and port of the LDAP server. Defaults to localhost:389 for ldap and localhost:636 for LDAPS.</td>
</tr>
<tr>
<td><strong>basedn</strong></td>
<td>The DN of the branch of the directory where all searches should start from. At the very least, this must be the top of your directory tree, but it could also specify a subtree in the directory.</td>
</tr>
<tr>
<td><strong>attribute</strong></td>
<td>The attribute to search for. Although RFC 2255 allows a comma-separated list of attributes, only the first attribute will be used, no matter how many are provided. If no attributes are provided, the default is to use uid. It is recommended to choose an attribute that will be unique across all entries in the subtree you will be using.</td>
</tr>
<tr>
<td><strong>scope</strong></td>
<td>The scope of the search. Can be either either <strong>one</strong> or <strong>sub</strong>. If the scope is not provided, the default is to use scope of <strong>sub</strong>.</td>
</tr>
<tr>
<td><strong>filter</strong></td>
<td>A valid LDAP search filter. If not provided, defaults to (objectClass=*)</td>
</tr>
</tbody>
</table>

When doing searches, the attribute, filter, and provided user name are combined to create a search filter that looks like:

```
(&(filter)(attribute=username))
```
For example, consider a URL of:

```
ldap://ldap.example.com/o=Acme?cn?sub?(enabled=true)
```

When a client attempts to connect using a user name of `bob`, the resulting search filter will be `(& (enabled=true)(cn=bob))`.

If the LDAP directory requires authentication to search, specify a `bindDN` and `bindPassword` to use to perform the entry search.

**Example 8.5. Master Configuration Using LDAPPasswordIdentityProvider**

```yaml
oauthConfig:
  ...
  identityProviders:
  - name: "my_ldap_provider"
    challenge: true
    login: true
    mappingMethod: claim
    provider:
      apiVersion: v1
      kind: LDAPPasswordIdentityProvider
      attributes:
        id:
        - dn
        email:
        - mail
        name:
        - cn
        preferredUsername:
        - uid
      bindDN: ""
      bindPassword: ""
      ca: my-ldap-ca-bundle.crt
      insecure: false
      url: "ldap://ldap.example.com/ou=users,dc=acme,dc=com?uid"
```

1. This provider name is prefixed to the returned user ID to form an identity name.
2. When `true`, unauthenticated token requests from non-web clients (like the CLI) are sent a `WWW-Authenticate` challenge header for this provider.
3. When `true`, unauthenticated token requests from web clients (like the web console) are redirected to a login page backed by this provider.
4. Controls how mappings are established between this provider’s identities and user objects, as described above.
5. List of attributes to use as the identity. First non-empty attribute is used. At least one attribute is required. If none of the listed attribute have a value, authentication fails.
6. List of attributes to use as the email address. First non-empty attribute is used.
List of attributes to use as the display name. First non-empty attribute is used.

List of attributes to use as the preferred user name when provisioning a user for this identity. First non-empty attribute is used.

Optional DN to use to bind during the search phase.

Optional password to use to bind during the search phase.

Certificate bundle to use to validate server certificates for the configured URL. If empty, system trusted roots are used. Only applies if insecure: false.

When true, no TLS connection is made to the server. When false, ldaps:// URLs connect using TLS, and ldap:// URLs are upgraded to TLS.

An RFC 2255 URL which specifies the LDAP host and search parameters to use, as described above.

8.2.7. Basic Authentication (Remote)

Set BasicAuthPasswordIdentityProvider in the identityProviders stanza to validate user names and passwords against a remote server using a server-to-server Basic authentication request. User names and passwords are validated against a remote URL that is protected by Basic authentication and returns JSON.

A 401 response indicates failed authentication.

A non-200 status, or the presence of a non-empty "error" key, indicates an error:

```
{"error":"Error message"}
```

A 200 status with a sub (subject) key indicates success:

```
{"sub":"userid"}
```

1 The subject must be unique to the authenticated user and must not be able to be modified.

A successful response may optionally provide additional data, such as:

- A display name using the name key. For example:

  ```
  {"sub":"userid", "name": "User Name", ...}
  ```

- An email address using the email key. For example:

  ```
  {"sub":"userid", "email":"user@example.com", ...}
  ```

- A preferred user name using the preferred_username key. This is useful when the unique, unchangeable subject is a database key or UID, and a more human-readable name exists. This is used as a hint when provisioning the OpenShift user for the authenticated identity. For example:
Example 8.6. Master Configuration Using BasicAuthPasswordIdentityProvider

```yaml
oauthConfig:
  identityProviders:
    - name: my_remote_basic_auth_provider
      challenge: true
      login: true
      mappingMethod: claim
      provider:
        apiVersion: v1
        kind: BasicAuthPasswordIdentityProvider
        url: https://www.example.com/remote-idp
        ca: /path/to/ca.file
        certFile: /path/to/client.crt
        keyFile: /path/to/client.key
```

1. This provider name is prefixed to the returned user ID to form an identity name.
2. When `true`, unauthenticated token requests from non-web clients (like the CLI) are sent a `WWW-Authenticate` challenge header for this provider.
3. When `true`, unauthenticated token requests from web clients (like the web console) are redirected to a login page backed by this provider.
4. Controls how mappings are established between this provider's identities and user objects, as described above.
5. URL accepting credentials in Basic authentication headers.
6. Optional: Certificate bundle to use to validate server certificates for the configured URL.
7. Optional: Client certificate to present when making requests to the configured URL.
8. Key for the client certificate. Required if `certFile` is specified.

8.2.8. Request Header

Set `RequestHeaderIdentityProvider` in the `identityProviders` stanza to identify users from request header values, such as `X-Remote-User`. It is typically used in combination with an authenticating proxy, which sets the request header value. This is similar to how the remote user plug-in in OpenShift Enterprise 2 allowed administrators to provide Kerberos, LDAP, and many other forms of enterprise authentication.

For users to authenticate using this identity provider, they must access `<master>/oauth/authorize` via an authenticating proxy. You can either proxy the entire master API server so that all access goes through the proxy, or you can configure the OAuth server to redirect unauthenticated requests to the proxy.

To redirect unauthenticated requests from clients expecting login flows:
1. Set the `login` parameter to `true`.

2. Set the `provider.loginURL` parameter to the proxy URL to send those clients to.

To redirect unauthenticated requests from clients expecting `WWW-Authenticate` challenges:

1. Set the `challenge` parameter to `true`.

2. Set the `provider.challengeURL` parameter to the proxy URL to send those clients to.

The `provider.challengeURL` and `provider.loginURL` parameters can include the following tokens in the query portion of the URL:

- `$url` is replaced with the current URL, escaped to be safe in a query parameter. For example: `https://www.example.com/sso-login?then=${url}

- `$query` is replaced with the current query string, unescaped. For example: `https://www.example.com/auth-proxy/oauth/authorize?${query}

**WARNING**

If you expect unauthenticated requests to reach the OAuth server, a `clientCA` parameter should be set for this identity provider, so that incoming requests are checked for a valid client certificate before the request's headers are checked for a user name. Otherwise, any direct request to the OAuth server can impersonate any identity from this provider, merely by setting a request header.

---

**Example 8.7. Master Configuration Using RequestHeaderIdentityProvider**

```yaml
oauthConfig:
  ...
  identityProviders:
    - name: my_request_header_provider  
      challenge: true  
      login: true  
      mappingMethod: claim  
      provider:
        apiVersion: v1
        kind: RequestHeaderIdentityProvider
        challengeURL: "https://www.example.com/challenging-proxy/oauth/authorize?${query}"  
        loginURL: "https://www.example.com/login-proxy/oauth/authorize?${query}"  
        clientCA: /path/to/client-ca.file  
        headers:  
          - X-Remote-User  
          - SSO-User
```

This provider name is prefixed to the user name in the request header to form an identity name.
**RequestHeaderIdentityProvider** can only respond to clients that request **WWW-Authenticate** challenges by redirecting to a configured **challengeURL**. The configured URL should respond with a **WWW-Authenticate** challenge.

**RequestHeaderIdentityProvider** can only respond to clients requesting a login flow by redirecting to a configured **loginURL**. The configured URL should respond with a login flow.

Controls how mappings are established between this provider’s identities and user objects, as described above.

Optional: URL to redirect unauthenticated /oauth/authorize requests to, for clients which expect interactive logins. `$\{url\}` is replaced with the current URL, escaped to be safe in a query parameter. `$\{query\}` is replaced with the current query string.

Optional: URL to redirect unauthenticated /oauth/authorize requests to, for clients which expect **WWW-Authenticate** challenges. `$\{url\}` is replaced with the current URL, escaped to be safe in a query parameter. `$\{query\}` is replaced with the current query string.

Optional: PEM-encoded certificate bundle. If set, a valid client certificate must be presented and validated against the certificate authorities in the specified file before the request headers are checked for user names.

Header names to check, in order, for user names. The first header containing a value is used as the user name. Required, case-insensitive.

---

**Example 8.8. Apache Authentication Using RequestHeaderIdentityProvider**

This example configures an authentication proxy on the same host as the master. Having the proxy and master on the same host is merely a convenience and may not be suitable for your environment. For example, if you were already running a router on the master, port 443 would not be available.

It is also important to note that while this reference configuration uses Apache’s **mod_auth_form**, it is by no means required and other proxies can easily be used if the following requirements are met:

1. Block the **X-Remote-User** header from client requests to prevent spoofing.

2. Enforce client certificate authentication in the **RequestHeaderIdentityProvider** configuration.

3. Require the **X-Csrf-Token** header be set for all authentication request using the challenge flow.

4. Only the /oauth/authorize endpoint should be proxied, and redirects should not be rewritten to allow the backend server to send the client to the correct location.

**Installing the Prerequisites**

The **mod_auth_form** module is shipped as part of the **mod_session** package that is found in the Optional channel:

```
# yum install -y httpd mod_ssl mod_session apr-util openssl
```

Generate a CA for validating requests that submit the trusted header. This CA should be used as the file name for **clientCA** in the master’s identity provider configuration.
# oadm ca create-signer-cert \
--cert="/etc/origin/master/proxyca.crt" \
--key="/etc/origin/master/proxyca.key" \
--name='openshift-proxy-signer@1432232228' \
--serial="/etc/origin/master/proxyca.serial.txt"

Generate a client certificate for the proxy. This can be done using any x509 certificate tooling. For convenience, the `oadm` CLI can be used:

```
# oadm create-api-client-config \
--certificate-authority="/etc/origin/master/proxyca.crt" \
--client-dir="/etc/origin/master/proxy" \
--signer-cert="/etc/origin/master/proxyca.crt" \
--signer-key="/etc/origin/master/proxyca.key" \
--signer-serial="/etc/origin/master/proxyca.serial.txt" \
--user='system:proxy'
```

```
# pushd /etc/origin/master
# cp master.server.crt /etc/pki/tls/certs/localhost.crt
# cp master.server.key /etc/pki/tls/private/localhost.key
# cp ca.crt /etc/pki/CA/certs/ca.crt
# cat proxy/system\:proxy.crt \n  proxy/system\:proxy.key > \n  /etc/pki/tls/certs/authproxy.pem
# popd
```

1. The user name can be anything, however it is useful to give it a descriptive name as it will appear in logs.

2. When running the authentication proxy on a different host name than the master, it is important to generate a certificate that matches the host name instead of using the default master certificate as shown above. The value for `masterPublicURL` in the `/etc/origin/master/master-config.yaml` file must be included in the X509v3 **Subject Alternative Name** in the certificate that is specified for `SSLCertificateFile`. If a new certificate needs to be created, the `oadm ca create-server-cert` command can be used.

## Configuring Apache

Unlike OpenShift Enterprise 2, this proxy does not need to reside on the same host as the master. It uses a client certificate to connect to the master, which is configured to trust the `X-Remote-User` header.

Configure Apache per the following:

```
LoadModule auth_form_module modules/mod_auth_form.so
LoadModule session_module modules/mod_session.so
LoadModule request_module modules/mod_request.so

# Nothing needs to be served over HTTP. This virtual host simply redirects to
# HTTPS.
<VirtualHost *:80>
  DocumentRoot /var/www/html
  RewriteEngine On
```
RewriteRule     ^(.*)$     https://%{HTTP_HOST}$1 [R,L]
</VirtualHost>

<VirtualHost *:443>
    # This needs to match the certificates you generated. See the CN and
    # Subject Alternative Name in the output of:
    # openssl x509 -text -in /etc/pki/tls/certs/localhost.crt
    ServerName www.example.com
    DocumentRoot /var/www/html
    SSLEngine on
    SSLCertificateFile /etc/pki/tls/certs/localhost.crt
    SSLCertificateKeyFile /etc/pki/tls/private/localhost.key
    SSLCACertificateFile /etc/pki/CA/certs/ca.crt
    SSLProxyEngine on
    SSLProxyCACertificateFile /etc/pki/CA/certs/ca.crt
    # It's critical to enforce client certificates on the Master.
    Otherwise
    # requests could spoof the X-Remote-User header by accessing the
    # Master's
    # /oauth/authorize endpoint directly.
    SSLProxyMachineCertificateFile /etc/pki/tls/certs/authproxy.pem
    # Send all requests to the console
    RewriteEngine On
    RewriteRule     ^/console(.*)$     https://%{HTTP_HOST}:8443/console$1 [R,L]
    # In order to using the challenging-proxy an X-Csrf-Token must be
    # present.
    RewriteCond %{REQUEST_URI} ^/challenging-proxy
    RewriteCond %{HTTP:X-Csrf-Token} ^$ 
    RewriteRule ^.* - 
    # Insert your backend server name/ip here.
    ProxyPass https://[MASTER]:8443/oauth/authorize
    AuthType basic
</Location>

<Location /challenging-proxy/oauth/authorize>
# mod_auth_form providers are implemented by mod_authn_dbm,
mod_authn_file,
# mod_authn_dbd, mod_authnz_ldap and mod_authn_socache.
AuthFormProvider file
AuthType form
AuthName openshift
ErrorDocument 401 /login.html
</Location>

<ProxyMatch /oauth/authorize>
AuthUserFile /etc/origin/master/htpasswd
AuthName openshift
Require valid-user
RequestHeader set X-Remote-User %{REMOTE_USER}s

# For ldap:
# AuthBasicProvider ldap
# AuthLDAPURL "ldap://ldap.example.com:389/ou=People,dc=my-domain,dc=com?uid?sub?(objectClass=*)"

# It's possible to remove the mod_auth_form usage and replace it with
# something like mod_auth_kerb, mod_auth_gsspail or even
# mod_auth_mellon.
# The former would be able to support both the login and challenge flows
# from the Master. Mellon would likely only support the login flow.

# For Kerberos
# yum install mod_auth_gssapi
# AuthType GSSAPI
# GssapiCredStore keytab:/etc/httpd.keytab
</ProxyMatch>
</VirtualHost>

RequestHeader unset X-Remote-User

Additional mod_auth_form Requirements

A sample login page is available from the openshift_extras repository. This file should be placed in the DocumentRoot location (/var/www/html by default).

Creating Users

At this point, you can create the users in the system Apache is using to store accounts information. In this example, file-backed authentication is used:

```
# yum -y install httpd-tools
# touch /etc/origin/master/htpasswd
# htpasswd /etc/origin/master/htpasswd <user_name>
```

Configuring the Master

The identityProviders stanza in the /etc/origin/master/master-config.yaml file must be updated as well:

```
identityProviders:
- name: requestheader
  challenge: true
  login: true
  provider:
    apiVersion: v1
    kind: RequestHeaderIdentityProvider
    challengeURL: "https://[MASTER]/challenging-proxy/oauth/authorize?${query}"
loginURL: "https://[MASTER]/login-proxy/oauth/authorize?${query}"
clientCA: /etc/origin/master/proxyca.crt
headers:
  - X-Remote-User

Restarting Services

Finally, restart the following services:

```
# systemctl restart httpd
# systemctl restart atomic-openshift-master
```

Verifying the Configuration

1. Test by bypassing the proxy. You should be able to request a token if you supply the correct client certificate and header:

```
# curl -L -k -H "X-Remote-User: joe" \ 
  --cert /etc/pki/tls/certs/authproxy.pem \ 
  https://[MASTER]:8443/oauth/token/request
```

2. If you do not supply the client certificate, the request should be denied:

```
# curl -L -k -H "X-Remote-User: joe" \ 
  https://[MASTER]:8443/oauth/token/request
```

3. This should show a redirect to the configured challengeURL (with additional query parameters):

```
# curl -k -v -H 'X-Csrf-Token: 1' \ 
  '<masterPublicURL>/oauth/authorize?client_id=openshift-challenging-client&response_type=token'
```

4. This should show a 401 response with a WWW-Authenticate basic challenge:

```
# curl -k -v -H 'X-Csrf-Token: 1' \ 
  '<redirected challengeURL from step 3 +query>'
```

5. This should show a redirect with an access token:

```
# curl -k -v -u <your_user>:<your_password> \ 
  -H 'X-Csrf-Token: 1' '<redirected_challengeURL_from_step_3 +query>'
```

8.2.9. GitHub

Set GitHubIdentityProvider in the identityProviders stanza to use GitHub as an identity provider, using the OAuth integration.
NOTE

Using GitHub as an identity provider requires users to get a token using <master>/oauth/token/request to use with command-line tools.

Example 8.9. Master Configuration Using GitHubIdentityProvider

```yaml
oauthConfig:
  ...
  identityProviders:
    - name: github
      challenge: false
      login: true
      mappingMethod: claim
      provider:
        apiVersion: v1
        kind: GitHubIdentityProvider
        clientID: ...
        clientSecret: ...

1 This provider name is prefixed to the GitHub numeric user ID to form an identity name. It is also used to build the callback URL.
2 GitHubIdentityProvider cannot be used to send WWW-Authenticate challenges.
3 When true, unauthenticated token requests from web clients (like the web console) are redirected to GitHub to log in.
4 Controls how mappings are established between this provider’s identities and user objects, as described above.
5 The client ID of a registered GitHub OAuth application. The application must be configured with a callback URL of <master>/oauth2callback/<identityProviderName>.
6 The client secret issued by GitHub.

8.2.10. Google

Set GoogleIdentityProvider in the identityProviders stanza to use Google as an identity provider, using Google’s OpenID Connect integration.

NOTE

Using Google as an identity provider requires users to get a token using <master>/oauth/token/request to use with command-line tools.

Example 8.10. Master Configuration Using GoogleIdentityProvider

```yaml
oauthConfig:
  ...
  identityProviders:

```

```yaml
```
- name: google  
  challenge: false  
  login: true  
  mappingMethod: claim  
  provider:  
    apiVersion: v1  
    kind: GoogleIdentityProvider  
    clientID: ...  
    clientSecret: ...  
    hostedDomain: ""  

1. This provider name is prefixed to the Google numeric user ID to form an identity name. It is also used to build the redirect URL.

2. `GoogleIdentityProvider` cannot be used to send `WWW-Authenticate` challenges.

3. When `true`, unauthenticated token requests from web clients (like the web console) are redirected to Google to log in.

4. Controls how mappings are established between this provider’s identities and user objects, as described above.

5. The client ID of a registered Google project. The project must be configured with a redirect URI of `<master>/oauth2callback/<identityProviderName>`.

6. The client secret issued by Google.

7. Optional hosted domain to restrict sign-in accounts to. If empty, any Google account is allowed to authenticate.

8.2.11. OpenID Connect

Set `OpenIDIdentityProvider` in the `identityProviders` stanza to integrate with an OpenID Connect identity provider using an Authorization Code Flow.

**NOTE**

ID Token and UserInfo decryptions are not supported.

By default, the openid scope is requested. If required, extra scopes can be specified in the extraScopes field.

Claims are read from the JWT `id_token` returned from the OpenID identity provider and, if specified, from the JSON returned by the UserInfo URL.

At least one claim must be configured to use as the user’s identity. The standard identity claim is `sub`.

You can also indicate which claims to use as the user’s preferred user name, display name, and email address. If multiple claims are specified, the first one with a non-empty value is used. The standard claims are:
<table>
<thead>
<tr>
<th>sub</th>
<th>The user identity.</th>
</tr>
</thead>
<tbody>
<tr>
<td>preferred_username</td>
<td>The preferred user name when provisioning a user.</td>
</tr>
<tr>
<td>email</td>
<td>Email address.</td>
</tr>
<tr>
<td>name</td>
<td>Display name.</td>
</tr>
</tbody>
</table>

**NOTE**

Using an OpenID Connect identity provider requires users to get a token using `<master>/oauth/token/request` to use with command-line tools.

**Example 8.11. Standard Master Configuration Using OpenIDIdentityProvider**

```yaml
oauthConfig:
  ...  
  identityProviders:
    - name: my_openid_connect
      challenge: false
      login: true
      mappingMethod: claim
      provider:
        apiVersion: v1
        kind: OpenIDIdentityProvider
        clientID: ...
        clientSecret: ...
        claims:
          id:
            - sub
          preferredUsername:
            - preferred_username
          name:
            - name
          email:
            - email
          urls:
            authorize: https://myidp.example.com/oauth2/authorize
            token: https://myidp.example.com/oauth2/token
```

1. This provider name is prefixed to the value of the identity claim to form an identity name. It is also used to build the redirect URL.

2. **OpenIDIdentityProvider** cannot be used to send `WWW-Authenticate` challenges.

3. When `true`, unauthenticated token requests from web clients (like the web console) are redirected to the authorize URL to log in.

4. Controls how mappings are established between this provider’s identities and user objects, as described above.
The client ID of a client registered with the OpenID provider. The client must be allowed to redirect to `<master>/oauth2callback/<identityProviderName>`.

The client secret.

Use the value of the `sub` claim in the returned `id_token` as the user's identity.

Authorization Endpoint described in the OpenID spec. Must use `https`.

Token Endpoint described in the OpenID spec. Must use `https`.

A custom certificate bundle, extra scopes, extra authorization request parameters, and `userInfo` URL can also be specified:

**Example 8.12. Full Master Configuration Using OpenIDIdentityProvider**

```yaml
oauthConfig:
  identityProviders:
  - name: my_openid_connect
    challenge: false
    login: true
    mappingMethod: claim
    provider:
      apiVersion: v1
      kind: OpenIDIdentityProvider
      clientID: ...
      clientSecret: ...
      ca: my-openid-ca-bundle.crt
      extraScopes:
        - email
        - profile
      extraAuthorizeParameters:
        - include_granted_scopes: "true"
      claims:
        id:
          - custom_id_claim
          - sub
        preferredUsername:
          - preferred_username
          - email
        name:
          - nickname
          - given_name
          - name
        email:
          - custom_email_claim
          - email
      urls:
        authorize: https://myidp.example.com/oauth2/authorize
        token: https://myidp.example.com/oauth2/token
        userInfo: https://myidp.example.com/oauth2/userinfo
```
Certificate bundle to use to validate server certificates for the configured URLs. If empty, system trusted roots are used.

Optional list of scopes to request, in addition to the `openid` scope, during the authorization token request.

Optional map of extra parameters to add to the authorization token request.

List of claims to use as the identity. First non-empty claim is used. At least one claim is required. If none of the listed claims have a value, authentication fails.

List of claims to use as the preferred user name when provisioning a user for this identity. First non-empty claim is used.

List of claims to use as the display name. First non-empty claim is used.

List of claims to use as the email address. First non-empty claim is used.

`UserInfo Endpoint` described in the OpenID spec. Must use `https`.

### 8.3. TOKEN OPTIONS

The OAuth server generates two kinds of tokens:

<table>
<thead>
<tr>
<th>Access tokens</th>
<th>Longer-lived tokens that grant access to the API.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorize codes</td>
<td>Short-lived tokens whose only use is to be exchanged for an access token.</td>
</tr>
</tbody>
</table>

Use the `tokenConfig` stanza to set token options:

```yaml
oauthConfig:
  ...
  tokenConfig:
    accessTokenMaxAgeSeconds: 86400
    authorizeTokenMaxAgeSeconds: 300
```

1. Set `accessTokenMaxAgeSeconds` to control the lifetime of access tokens. The default lifetime is 24 hours.
2. Set `authorizeTokenMaxAgeSeconds` to control the lifetime of authorize codes. The default lifetime is five minutes.

### 8.4. GRANT OPTIONS
To configure how the OAuth server responds to token requests for a client the user has not previously granted permission, set the **method** value in the **grantConfig** stanza. Valid values for **method** are:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto</td>
<td>Auto-approve the grant and retry the request.</td>
</tr>
<tr>
<td>prompt</td>
<td>Prompt the user to approve or deny the grant.</td>
</tr>
<tr>
<td>deny</td>
<td>Auto-deny the grant and return a failure error to the client.</td>
</tr>
</tbody>
</table>

### Example 8.14. Master Configuration Grant Options

```yaml
oauthConfig:
  ...
  grantConfig:
    method: auto
```

### 8.5. SESSION OPTIONS

The OAuth server uses a signed and encrypted cookie-based session during login and redirect flows.

Use the **sessionConfig** stanza to set session options:

### Example 8.15. Master Configuration Session Options

```yaml
oauthConfig:
  ...
  sessionConfig:
    sessionMaxAgeSeconds: 300
    sessionName: ssn
    sessionSecretsFile: "...
```

1. Controls the maximum age of a session; sessions auto-expire once a token request is complete. If **auto-grant** is not enabled, sessions must last as long as the user is expected to take to approve or reject a client authorization request.
2. Name of the cookie used to store the session.
3. File name containing serialized **SessionSecrets** object. If empty, a random signing and encryption secret is generated at each server start.

If no **sessionSecretsFile** is specified, a random signing and encryption secret is generated at each start of the master server. This means that any logins in progress will have their sessions invalidated if the master is restarted. It also means that if multiple masters are configured, they will not be able to decode sessions generated by one of the other masters.

To specify the signing and encryption secret to use, specify a **sessionSecretsFile**. This allows you separate secret values from the configuration file and keep the configuration file distributable, for example for debugging purposes.
Multiple secrets can be specified in the `sessionSecretsFile` to enable rotation. New sessions are signed and encrypted using the first secret in the list. Existing sessions are decrypted and authenticated by each secret until one succeeds.

**Example 8.16. Session Secret Configuration:**

```yaml
apiVersion: v1
kind: SessionSecrets
secrets:
  - authentication: "...
    encryption: "...
  - authentication: "...
    encryption: "...
...
```

1. List of secrets used to authenticate and encrypt cookie sessions. At least one secret must be specified. Each secret must set an authentication and encryption secret.

2. Signing secret, used to authenticate sessions using HMAC. Recommended to use a secret with 32 or 64 bytes.

3. Encrypting secret, used to encrypt sessions. Must be 16, 24, or 32 characters long, to select AES-128, AES-192, or AES-256.
CHAPTER 9. SYNCING GROUPS WITH LDAP

9.1. OVERVIEW

As an OpenShift administrator, you can use groups to manage users, change their permissions, and enhance collaboration. Your organization may have already created user groups and stored them in an LDAP server. OpenShift can sync those LDAP records with internal OpenShift records, enabling you to manage your groups in one place. OpenShift currently supports group sync with LDAP servers using three common schemas for defining group membership: RFC 2307, Active Directory, and augmented Active Directory.

NOTE
You must have `cluster-admin` privileges to sync groups.

9.2. CONFIGURING LDAP SYNC

Before you can run LDAP sync, you need a sync configuration file. This file contains LDAP client configuration details:

- Configuration for connecting to your LDAP server.
- Sync configuration options that are dependent on the schema used in your LDAP server.

A sync configuration file can also contain an administrator-defined list of name mappings that maps OpenShift Group names to groups in your LDAP server.

9.2.1. LDAP Client Configuration

Example 9.1. LDAP Client Configuration

```
url: ldap://10.0.0.0:389
bindDN: cn=admin,dc=example,dc=com
bindPassword: password
insecure: true
ca: my-ldap-ca-bundle.crt
```

1. The connection protocol, IP address of the LDAP server hosting your database, and the port to connect to, formatted as `scheme://host:port`.
2. Optional distinguished name (DN) to use as the Bind DN. OpenShift uses this if elevated privilege is required to retrieve entries for the sync operation.
3. Optional password to use to bind. OpenShift uses this if elevated privilege is necessary to retrieve entries for the sync operation.
4. When `true`, no TLS connection is made to the server. When `false`, secure LDAP (ldaps://) URLs connect using TLS, and insecure LDAP (ldap://) URLs are upgraded to TLS.
5. The certificate bundle to use for validating server certificates for the configured URL. If empty, OpenShift uses system-trusted roots. This only applies if `insecure` is set to `false`.
9.2.2. LDAP Query Definition

Sync configurations consist of LDAP query definitions for the entries that are required for synchronization. The specific definition of an LDAP query depends on the schema used to store membership information in the LDAP server.

Example 9.2. LDAP Query Definition

```
baseDN: ou=users,dc=example,dc=com  
scope: sub  
derefAliases: never  
timeout: 0  
filter: (objectClass/inetOrgPerson)  
```

1. The distinguished name (DN) of the branch of the directory where all searches will start from. It is required that you specify the top of your directory tree, but you can also specify a subtree in the directory.

2. The scope of the search. Valid values are `base`, `one`, or `sub`. If this is left undefined, then a scope of `sub` is assumed. Descriptions of the scope options can be found in the table below.

3. The behavior of the search with respect to aliases in the LDAP tree. Valid values are `never`, `search`, `base`, or `always`. If this is left undefined, then the default is to `always` dereference aliases. Descriptions of the dereferencing behaviors can be found in the table below.

4. The time limit allowed for the search by the client, in seconds. A value of 0 imposes no client-side limit.

5. A valid LDAP search filter. If this is left undefined, then the default is `(objectClass=*)`.

Table 9.1. LDAP Search Scope Options

<table>
<thead>
<tr>
<th>LDAP Search Scope</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>base</code></td>
<td>Only consider the object specified by the base DN given for the query.</td>
</tr>
<tr>
<td><code>one</code></td>
<td>Consider all of the objects on the same level in the tree as the base DN for the query.</td>
</tr>
<tr>
<td><code>sub</code></td>
<td>Consider the entire subtree rooted at the base DN given for the query.</td>
</tr>
</tbody>
</table>

Table 9.2. LDAP Dereferencings Behaviors

<table>
<thead>
<tr>
<th>Dereferencing Behavior</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>never</code></td>
<td>Never dereference any aliases found in the LDAP tree.</td>
</tr>
</tbody>
</table>
### Dereferencing Behavior

<table>
<thead>
<tr>
<th>Behavior</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>search</td>
<td>Only dereference aliases found while searching.</td>
</tr>
<tr>
<td>base</td>
<td>Only dereference aliases while finding the base object.</td>
</tr>
<tr>
<td>always</td>
<td>Always dereference all aliases found in the LDAP tree.</td>
</tr>
</tbody>
</table>

### 9.2.3. User-Defined Name Mapping

A user-defined name mapping explicitly maps the names of OpenShift Groups to unique identifiers that find groups on your LDAP server. The mapping uses normal YAML syntax. A user-defined mapping can contain an entry for every group in your LDAP server or only a subset of those groups. If there are groups on the LDAP server that do not have a user-defined name mapping, the default behavior during sync is to use the attribute specified as the Group’s name.

#### Example 9.3. User-Defined Name Mapping

```yaml
groupUIDNameMapping:
  "cn=group1,ou=groups,dc=example,dc=com": firstgroup
  "cn=group2,ou=groups,dc=example,dc=com": secondgroup
  "cn=group3,ou=groups,dc=example,dc=com": thirdgroup
```

### 9.3. RUNNING LDAP SYNC

Once you have created a sync configuration file, then sync can begin. OpenShift allows administrators to perform a number of different sync types with the same server.

#### NOTE

By default, all group synchronization or pruning operations are dry-run, so you must set the `--confirm` flag on the `sync-groups` command in order to make changes to OpenShift Group records.

To sync all groups from the LDAP server with OpenShift:

```
$ oadm groups sync --sync-config=config.yaml --confirm
```

To sync all Groups already in OpenShift that correspond to groups in the LDAP server specified in the configuration file:

```
$ oadm groups sync --type=openshift --sync-config=config.yaml --confirm
```

To sync a subset of LDAP groups with OpenShift, you can use whitelist files, blacklist files, or both:
NOTE

Any combination of blacklist files, whitelist files, or whitelist literals will work; whitelist literals can be included directly in the command itself. This applies to groups found on LDAP servers, as well as Groups already present in OpenShift. Your files must contain one unique group identifier per line.

```bash
$ oadm groups sync --whitelist=<whitelist_file> \   
   --sync-config=config.yaml \   
   --confirm
$ oadm groups sync --blacklist=<blacklist_file> \   
   --sync-config=config.yaml \   
   --confirm
$ oadm groups sync <group_unique_identifier> \   
   --sync-config=config.yaml \   
   --confirm
$ oadm groups sync <group_unique_identifier> \   
   --whitelist=<whitelist_file> \   
   --blacklist=<blacklist_file> \   
   --sync-config=config.yaml \   
   --confirm
$ oadm groups sync --type=openshift \   
   --whitelist=<whitelist_file> \   
   --sync-config=config.yaml \   
   --confirm
```

9.4. RUNNING A GROUP PRUNING JOB

An administrator can also choose to remove groups from OpenShift records if the records on the LDAP server that created them are no longer present. The prune job will accept the same sync configuration file and white- or black-lists as used for the sync job.

For example, if groups had previously been synchronized from LDAP using some `config.yaml` file, and some of those groups no longer existed on the LDAP server, the following command would determine which Groups in OpenShift corresponded to the deleted groups in LDAP and then remove them from OpenShift:

```
$ oadm groups prune --sync-config=config.yaml --confirm
```

9.5. SYNC EXAMPLES

This section contains examples for the RFC 2307, Active Directory, and augmented Active Directory schemas. All of the following examples synchronize a group named `admins` that has two members: Jane and Jim. Each example explains:

- How the group and users are added to the LDAP server.
- What the LDAP sync configuration file looks like.
- What the resulting Group record in OpenShift will be after synchronization.

9.5.1. RFC 2307
In the RFC 2307 schema, both users (Jane and Jim) and groups exist on the LDAP server as first-class entries, and group membership is stored in attributes on the group. The following snippet of ldif defines the users and group for this schema:

```
Example 9.4. LDAP Entries Using RFC 2307 Schema: rfc2307.ldif

dn: ou=users,dc=example,dc=com
objectClass: organizationalUnit
ou: users

  dn: cn=Jane,ou=users,dc=example,dc=com
  objectClass: person
  objectClass: organizationalPerson
  objectClass: inetOrgPerson
  cn: Jane
  sn: Smith
  displayName: Jane Smith
  mail: jane.smith@example.com

  dn: cn=Jim,ou=users,dc=example,dc=com
  objectClass: person
  objectClass: organizationalPerson
  objectClass: inetOrgPerson
  cn: Jim
  sn: Adams
  displayName: Jim Adams
  mail: jim.adams@example.com

  dn: ou=groups,dc=example,dc=com
  objectClass: organizationalUnit
  ou: groups

  dn: cn=admins,ou=groups,dc=example,dc=com
  objectClass: groupOfNames
  cn: admins
  owner: cn=admin,dc=example,dc=com
  description: System Administrators
  member: cn=Jane,ou=users,dc=example,dc=com
  member: cn=Jim,ou=users,dc=example,dc=com

1 The group is a first-class entry in the LDAP server.
2 Members of a group are listed with an identifying reference as attributes on the group.
```

To sync this group, you must first create the configuration file. The RFC 2307 schema requires you to provide an LDAP query definition for both user and group entries, as well as the attributes with which to represent them in the internal OpenShift records.

For clarity, the Group you create in OpenShift should use attributes other than the distinguished name whenever possible for user- or administrator-facing fields. For example, identify the users of a Group by their e-mail, and use the name of the group as the common name. The following configuration file creates these relationships:
NOTE

If using user-defined name mappings, your configuration file will differ.

Example 9.5. LDAP Sync Configuration Using RFC 2307 Schema: rfc2307_config.yaml

```
kind: LDAPSyncConfig
apiVersion: v1
url: ldap://LDAP_SERVICE_IP:389
insecure: true
rfc2307:
  groupsQuery:
    baseDN: "ou=groups,dc=example,dc=com"
    scope: sub
    derefAliases: never
    filter: (objectclass=groupOfNames)
    groupUIDAttribute: dn
  groupNameAttributes: [ cn ]
  groupMembershipAttributes: [ member ]
usersQuery:
  baseDN: "ou=users,dc=example,dc=com"
  scope: sub
  derefAliases: never
  filter: (objectclass/inetOrgPerson)
  userUIDAttribute: dn
  userNameAttributes: [ mail ]
```

1. The IP address and host of the LDAP server where this group's record is stored.
2. When true, no TLS connection is made to the server. When false, secure LDAP (ldaps://) URLs connect using TLS, and insecure LDAP (ldap://) URLs are upgraded to TLS.
3. The attribute that uniquely identifies a group on the LDAP server.
4. The attribute to use as the name of the Group.
5. The attribute on the group that stores the membership information.
6. The attribute that uniquely identifies a user on the LDAP server.
7. The attribute to use as the name of the user in the OpenShift Group record.

To run sync with the rfc2307_config.yaml file:

```
$ oadm groups sync --sync-config=rfc2307_config.yaml --confirm
```

OpenShift creates the following Group record as a result of the above sync operation:

```
Example 9.6. OpenShift Group Created Using rfc2307_config.yaml

```

apiVersion: v1
kind: Group
metadata:
    annotations:
        openshift.io/ldap.sync-time: 2015-10-13T10:08:38-0400 1
        openshift.io/ldap.uid: cn=admins,ou=groups,dc=example,dc=com 2
        openshift.io/ldap.url: LDAP_SERVER_IP:389 3
    creationTimestamp:
        name: admins 4
    users: 5
        - jane.smith@example.com
        - jim.adams@example.com

1. The last time this Group was synchronized with the LDAP server, in ISO 6801 format.
2. The unique identifier for the group on the LDAP server.
3. The IP address and host of the LDAP server where this Group's record is stored.
4. The name of the Group as specified by the sync file.
5. The users that are members of the Group, named as specified by the sync file.

### 9.5.1.1. RFC2307 with User-Defined Name Mappings

When syncing groups with user-defined name mappings, the configuration file changes to contain these mappings as shown below.

**Example 9.7. LDAP Sync Configuration Using RFC 2307 Schema With User-Defined Name Mappings: rfc2307_config_user_defined.yaml**

```yaml
kind: LDAPSyncConfig
apiVersion: v1
groupNameNameMapping:
    "cn=admins,ou=groups,dc=example,dc=com": Administrators 1
rfc2307:
    groupsQuery:
        baseDN: "ou=groups,dc=example,dc=com"
        scope: sub
        derefAliases: never
        filter: (objectclass=groupOfNames)
        groupUIDAttribute: dn 2
    groupNameAttributes: [ cn ] 3
    groupMembershipAttributes: [ member ]
    usersQuery:
        baseDN: "ou=users,dc=example,dc=com"
        scope: sub
        derefAliases: never
        filter: (objectclass=inetOrgPerson)
        userUIDAttribute: dn
        userNameAttributes: [ mail ]
```

1. The user-defined name mapping.
The unique identifier attribute that is used for the keys in the user-defined name mapping.

The attribute to name OpenShift Groups with if their unique identifier is not in the user-defined name mapping.

To run sync with the `rfc2307_config_user_defined.yaml` file:

```bash
$ oadm groups sync --sync-config=rfc2307_config_user_defined.yaml --confirm
```

OpenShift creates the following Group record as a result of the above sync operation:

**Example 9.8. OpenShift Group Created Using `rfc2307_config_user_defined.yaml`**

```yaml
apiVersion: v1
kind: Group
metadata:
  annotations:
    openshift.io/ldap.uid: cn=admins,ou=groups,dc=example,dc=com
    openshift.io/ldap.url: LDAP_SERVER_IP:389
  creationTimestamp:
    name: Administrators
    users:
      - jane.smith@example.com
      - jim.adams@example.com
```

The name of the Group as specified by the user-defined name mapping.

**9.5.2. Active Directory**

In the Active Directory schema, both users (Jane and Jim) exist in the LDAP server as first-class entries, and group membership is stored in attributes on the user. The following snippet of `ldif` defines the users and group for this schema:

**Example 9.9. LDAP Entries Using Active Directory Schema: `active_directory.ldif`**

```ldap
dn: ou=users,dc=example,dc=com
objectClass: organizationalUnit
ou: users

dn: cn=Jane,ou=users,dc=example,dc=com
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: testPerson
cn: Jane
sn: Smith
displayName: Jane Smith
mail: jane.smith@example.com
```
The user’s group memberships are listed as attributes on the user, and the group does not exist as an entry on the server. The `testMemberOf` attribute cannot be a literal attribute on the user; it can be created during search and returned to the client but not committed to the database.

To sync this group, you must first create the configuration file. The Active Directory schema requires you to provide an LDAP query definition for user entries, as well as the attributes to represent them with in the internal OpenShift Group records.

For clarity, the Group you create in OpenShift should use attributes other than the distinguished name whenever possible for user- or administrator-facing fields. For example, identify the users of a Group by their e-mail, but define the name of the Group by the name of the group on the LDAP server. The following configuration file creates these relationships:

**Example 9.10. LDAP Sync Configuration Using Active Directory Schema:**

```
kind: LDAPSyncConfig
apiVersion: v1
url: ldap://LDAP_SERVICE_IP:389
insecure: true
activeDirectory:
  usersQuery:
    baseDN: "ou=users,dc=example,dc=com"
    scope: sub
derefAliases: never
    filter: (objectClass/inetOrgPerson)
    userNameAttributes: [ mail ]
  groupMembershipAttributes: [ testMemberOf ]
```

1. The attribute to use as the name of the user in the OpenShift Group record.
2. The attribute on the user that stores the membership information.

To run sync with the `active_directory_config.yaml` file:

```
$ oadm groups sync --sync-config=active_directory_config.yaml --confirm
```
OpenShift creates the following Group record as a result of the above sync operation:

Example 9.11. OpenShift Group Created Using active_directory_config.yaml

```
apiVersion: v1
kind: Group
metadata:
  annotations:
    openshift.io/ldap.uid: admins
    openshift.io/ldap.url: LDAP_SERVER_IP:389
  creationTimestamp:
    name: admins
users: 
  - jane.smith@example.com
  - jim.adams@example.com
```

1. The last time this Group was synchronized with the LDAP server, in ISO 6801 format.
2. The unique identifier for the group on the LDAP server.
3. The IP address and host of the LDAP server where this Group's record is stored.
4. The name of the group as listed in the LDAP server.
5. The users that are members of the Group, named as specified by the sync file.

9.5.3. Augmented Active Directory

In the augmented Active Directory schema, both users (Jane and Jim) and groups exist in the LDAP server as first-class entries, and group membership is stored in attributes on the user. The following snippet of ldif defines the users and group for this schema:

Example 9.12. LDAP Entries Using Augmented Active Directory Schema:

```
adn: ou=users,dc=example,dc=com
objectClass: organizationalUnit
ou: users

dn: cn=Jane,ou=users,dc=example,dc=com
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: testPerson
cn: Jane
sn: Smith
displayName: Jane Smith
mail: jane.smith@example.com
testMemberOf: cn=admins,ou=groups,dc=example,dc=com

dn: cn=Jim,ou=users,dc=example,dc=com
```

1. The users that are members of the Group, named as specified by the sync file.
The user's group memberships are listed as attributes on the user.

The group is a first-class entry on the LDAP server.

To sync this group, you must first create the configuration file. The augmented Active Directory schema requires you to provide an LDAP query definition for both user entries and group entries, as well as the attributes with which to represent them in the internal OpenShift Group records.

For clarity, the Group you create in OpenShift should use attributes other than the distinguished name whenever possible for user- or administrator-facing fields. For example, identify the users of a Group by their e-mail, and use the name of the Group as the common name. The following configuration file creates these relationships.

**Example 9.13. LDAP Sync Configuration Using Augmented Active Directory Schema:**

```
kind: LDAPSyncConfig
apiVersion: v1
url: ldap://LDAP_SERVICE_IP:389
insecure: true
augmentedActiveDirectory:
groupsQuery:
  baseDN: "ou=groups,dc=example,dc=com"
  scope: sub
  derefAliases: never
  filter: (objectclass=groupOfNames)
groupUIDAttribute: dn
groupNameAttributes: [ cn ]
usersQuery:
  baseDN: "ou=users,dc=example,dc=com"
  scope: sub
```
The attribute that uniquely identifies a group on the LDAP server.

2 The attribute to use as the name of the Group.

3 The attribute to use as the name of the user in the OpenShift Group record.

4 The attribute on the user that stores the membership information.

To run sync with the `augmented_active_directory_config.yaml` file:

```bash
$ oadm groups sync --sync-config=augmented_active_directory_config.yaml --confirm
```

OpenShift creates the following Group record as a result of the above sync operation:

**Example 9.14. OpenShift Group Created Using augmented_active_directory_config.yaml**

```yaml
apiVersion: v1
class: Group
metadata:
  annotations:
    openshift.io/ldap.sync-time: 2015-10-13T08:38:0400
    openshift.io/ldap.uid: cn=admins,ou=groups,dc=example,dc=com
    openshift.io/ldap.url: LDAP_SERVER_IP:389
  creationTimestamp: name: admins
users:
  - jane.smith@example.com
  - jim.adams@example.com
```

1 The last time this Group was synchronized with the LDAP server, in ISO 6801 format.

2 The unique identifier for the group on the LDAP server.

3 The IP address and host of the LDAP server where this Group’s record is stored.

4 The name of the Group as specified by the sync file.

5 The users that are members of the Group, named as specified by the sync file.
10.1. OVERVIEW

OpenShift Enterprise Advanced Lightweight Directory Access Protocol (LDAP) Configuration covers the following topics:

- Setting up SSSD for LDAP Failover
- Configuring Form-Based Authentication
- Configuring Extended LDAP Attributes

10.2. SETTING UP SSSD FOR LDAP FAILOVER

10.2.1. Overview

OpenShift Enterprise provides an authentication provider for use with Lightweight Directory Access Protocol (LDAP) setups, but it can only connect to a single LDAP server. This can be problematic if that LDAP server becomes unavailable. System Security Services Daemon (SSSD) can be used to solve the issue.

Originally designed to manage local and remote authentication to the host operating system, SSSD can now be configured to provide identity, authentication, and authorization services to web services like OpenShift Enterprise. SSSD provides advantages over the built-in LDAP provider, including the ability to connect to any number of failover LDAP servers, as well as the ability to cache authentication attempts in case it can no longer reach any of those servers.

The setup for this configuration is advanced and requires a separate authentication server (also called an authenticating proxy) for OpenShift Enterprise to communicate with. This topic describes how to do this setup on a dedicated physical or virtual machine (VM), but the concepts are also applicable to a setup in a container.

10.2.2. Prerequisites for Authenticating Proxy Setup

1. Before starting setup, you need to know the following information about your LDAP server.
   - Whether the directory server is powered by FreeIPA, Active Directory, or another LDAP solution.
   - The Uniform Resource Identifier (URI) for the LDAP server (for example, ldap.example.com).
   - The location of the CA certificate for the LDAP server.
   - Whether the LDAP server corresponds to RFC 2307 or RFC2307bis for user groups.

2. Prepare the VMs:
   - **proxy.example.com**: A VM to use as the authenticating proxy. This machine must have at least SSSD 1.12.0 available, which means a fairly recent operating system. This topic uses a Red Hat Enterprise Linux 7.2 server for its examples.
   - **openshift.example.com**: A VM to use to run OpenShift Enterprise.
NOTE

These VMs can be configured to run on the same system, but for the examples used in this topic they are kept separate.

10.2.3. Phase 1: Certificate Generation

1. To ensure that communication between the authenticating proxy and OpenShift Enterprise is trustworthy, create a set of Transport Layer Security (TLS) certificates to use during the other phases of this setup. In the OpenShift Enterprise system, start by using the auto-generated certificates created as part of running:

```bash
# openshift start --public-master=https://openshift.example.com:8443 --write-config=/etc/origin/
```

Among other things, this generates a `/etc/origin/master/ca.{cert|key}`. Use this signing certificate to generate keys to use on the authenticating proxy.

```bash
# mkdir -p /etc/origin/proxy/
# oadm ca create-server-cert --cert='/etc/origin/proxy/proxy.example.com.crt' --key='/etc/origin/proxy/proxy.example.com.key' --hostnames=proxy.example.com --signer-cert=/etc/origin/master/ca.crt --signer-key='/etc/origin/master/ca.key' --signer-serial='/etc/origin/master/ca.serial.txt'
```

IMPORTANT

Ensure that any host names and interface IP addresses that need to access the proxy are listed. Otherwise, the HTTPS connection will fail.

2. Generate the API client certificate that the authenticating proxy will use to prove its identity to OpenShift Enterprise. This is necessary and prevents malicious users from impersonating the proxy and sending fake identities.

3. Create a new CA to sign this client certificate:

```bash
# mkdir -p /etc/origin/proxy/
# oadm ca create-server-cert --cert='/etc/origin/proxy/proxy.example.com.crt' --key='/etc/origin/proxy/proxy.example.com.key' --hostnames=proxy.example.com,1.2.3.4 --signer-cert=/etc/origin/master/ca.crt --signer-key='/etc/origin/master/ca.key' --signer-serial='/etc/origin/master/ca.serial.txt'
```

Make `UNIQUESTRING` something unique:

```bash
# oadm ca create-signer-cert --cert='/etc/origin/proxy/proxyca.crt' --key='/etc/origin/proxy/proxyca.key'
```
10.2.4. Phase 2: Authenticating Proxy Setup

This section guides you through the steps to authenticate the proxy setup.

10.2.4.1. Step 1: Copy Certificates

From openshift.example.com, securely copy the necessary certificates to the proxy machine:

```bash
# scp /etc/origin/proxy/master/ca.crt  
    root@proxy.example.com:/etc/pki/CA/certs/
# scp /etc/origin/proxy/proxy.example.com.crt  
    /etc/origin/proxy/authproxy.pem  
    root@proxy.example.com:/etc/pki/tls/certs/
# scp /etc/origin/proxy/proxy.example.com.key  
    root@proxy.example.com:/etc/pki/tls/private/
```

10.2.4.2. Step 2: SSSD Configuration

1. Install a new VM with an operating system that includes 1.12.0 or later so that you can use the `mod_identity_lookup` module. The examples in this topic use a Red Hat Enterprise Linux 7.2 Server.

2. Install all of the necessary dependencies:

   ```bash
   # yum install -y sssd
      sssd-dbus
      realmd
      httpd
      mod_session
      mod_ssl
      mod_lookup_identity
      mod_authnz_pam
   ```

   This gives you the needed SSSD and the web server components.

3. Set up SSSD to authenticate this VM against the LDAP server. If the LDAP server is a FreeIPA or Active Directory environment, then `realmd` can be used to join this machine to the domain.

   ```bash
   # realm join ldap.example.com
   ```

   For more advanced case, see the System-Level Authentication Guide

   If you want to use SSSD to manage failover situations for LDAP, this can be configured by adding additional entries in `/etc/sssd/sssd.conf` on the `ldap_uri` line. Systems enrolled with FreeIPA can automatically handle failover using DNS SRV records.

4. Restart SSSD to ensure that all of the changes are applied properly:

   ```bash
   $ systemctl restart sssd.service
   ```
5. Test that the user information can be retrieved properly:

   $ getent passwd <username>
   username:*:12345:12345:Example User:/home/username:/usr/bin/bash

6. Attempt to log into the VM as an LDAP user and confirm that the authentication is properly set up. This can be done via the local console or a remote service such as SSH.

   **NOTE**
   If you do not want LDAP users to be able to log into this machine, it is recommended to modify `/etc/pam.d/system-auth` and `/etc/pam.d/password-auth` to remove the lines containing `pam_sss.so`.

10.2.4.3. Step 3: Apache Configuration

You need to set up Apache to communicate with SSSD. Create a PAM stack file for use with Apache. To do so:

1. Create the `/etc/pam.d/openshift` file and add the following contents:

   ```
   auth required pam_sss.so
   account required pam_sss.so
   ```

   This configuration enables PAM (the pluggable authentication module) to use `pam_sss.so` to determine authentication and access control when an authentication request is issued for the `openshift` stack.

2. Configure the Apache `httpd.conf`. The steps in this section focus on setting up the challenge authentication, which is useful for logging in with `oc login` and similar automated tools.

   **NOTE**
   Configuring Form-Based Authentication explains how to set up a graphical login using SSSD as well, but it requires the rest of this setup as a prerequisite.

3. Create the new file `openshift-proxy.conf` in `/etc/httpd/conf.d` (substituting the correct host names where indicated):

   ```
   LoadModule request_module modules/mod_request.so
   LoadModule lookup_identity_module modules/mod_lookup_identity.so
   # Nothing needs to be served over HTTP. This virtual host simply redirects to HTTPS.
   <VirtualHost *:80>
       DocumentRoot /var/www/html
       RewriteEngine On
       RewriteRule ^(.*)$ https://%{HTTP_HOST}$1 [R,L]
   </VirtualHost>
   
   <VirtualHost *:443>
       # This needs to match the certificates you generated. See the CN and X509v3
   ```
Subject Alternative Name in the output of:
openssl x509 -text -in /etc/pki/tls/certs/proxy.example.com.crt
ServerName proxy.example.com

DocumentRoot /var/www/html
SSLEngine on
SSLCertificateFile /etc/pki/tls/certs/proxy.example.com.crt
SSLCertificateKeyFile /etc/pki/tls/private/proxy.example.com.key
SSLCACertificateFile /etc/pki/CA/certs/ca.crt

# Send logs to a specific location to make them easier to find
ErrorLog logs/proxy_error_log
TransferLog logs/proxy_access_log
LogLevel warn
SSLProxyEngine on
SSLProxyCACertificateFile /etc/pki/CA/certs/ca.crt

# It's critical to enforce client certificates on the Master.
Otherwise
# requests could spoof the X-Remote-User header by accessing the Master's
# /oauth/authorize endpoint directly.
SSLProxyMachineCertificateFile /etc/pki/tls/certs/authproxy.pem

# Send all requests to the console
RewriteEngine On
RewriteRule ^/console(.*)$ https://%{HTTP_HOST}:8443/console$1 [R,L]

# In order to using the challenging-proxy an X-Csrf-Token must be present.
RewriteCond %{REQUEST_URI} ^/challenging-proxy
RewriteCond %{HTTP:X-Csrf-Token} ^$ [NC]
RewriteRule ^.* - [F,L]

<Location /challenging-proxy/oauth/authorize>
# Insert your backend server name/ip here.
ProxyPass https://openshift.example.com:8443/oauth/authorize
AuthType Basic
AuthBasicProvider PAM
AuthPAMService openshift
Require valid-user
</Location>

<ProxyMatch /oauth/authorize>
AuthName openshift
RequestHeader set X-Remote-User %{REMOTE_USER}s
</ProxyMatch>
</VirtualHost>

RequestHeader unset X-Remote-User

NOTE
Configuring Form-Based Authentication explains how to add the login-proxy block to support form authentication.
4. Set a boolean to tell SELinux that it is acceptable for Apache to contact the PAM subsystem:

```
# setsebool -P allow_httpd_mod_auth_pam on
```

5. Start up Apache:

```
# systemctl start httpd.service
```

### 10.2.5. Phase 3: OpenShift Enterprise Configuration

This section describes how to set up an OpenShift Enterprise server from scratch in an “all in one” configuration. **Master and Node Configuration** provides more information on alternate configurations.

Modify the default configuration to use the new identity provider just created. To do so:

1. Modify the `/etc/origin/master/master-config.yaml` file.

2. Scan through it and locate the `identityProviders` section and replace it with:

```
    identityProviders:
    - name: any_provider_name
      challenge: true
      login: false
      mappingMethod: claim
      provider:
        apiVersion: v1
        kind: RequestHeaderIdentityProvider
        challengeURL: "https://proxy.example.com/challenging-proxy/oauth/authorize?$\{query\}"
        clientCA: /etc/origin/master/proxy/proxyca.crt
        headers:
          - X-Remote-User
```

**NOTE**

*Configuring Form-Based Authentication* explains how to add the login URL to support web logins.

*Configuring Extended LDAP Attributes* explains how to add the email and full-name attributes. Note that the full-name attributes are only stored to the database on the first login.

3. Start OpenShift Enterprise with the updated configuration:

```
# openshift start \
   --public-master=https://openshift.example.com:8443 \
   --master-config=/etc/origin/master/master-config.yaml \
   --node-config=/etc/origin/node-node1.example.com/node-config.yaml
```

4. Test logins:

```
oc login https://openshift.example.com:8443
```
It should now be possible to log in with only valid LDAP credentials.

### 10.3. CONFIGURING FORM-BASED AUTHENTICATION

#### 10.3.1. Overview

This topic builds upon Setting up SSSD for LDAP Failover and describes how to set up form-based authentication for signing into the OpenShift Enterprise web console.

#### 10.3.2. Prepare a Login Page

The OpenShift Enterprise upstream repositories have a template for forms. Copy that to your authenticating proxy on `proxy.example.com`:

```bash
# curl -o /var/www/html/login.html \\
https://raw.githubusercontent.com/openshift/openshift-extras/master/misc/form_auth/login.html
```

Modify this .html file to change the logo icon and "Welcome" content for your environment.

#### 10.3.3. Install Another Apache Module

To intercept form-based authentication, install an Apache module:

```bash
# yum -y install mod_intercept_form_submit
```

#### 10.3.4. Apache Configuration

1. Modify `/etc/httpd/conf.modules.d/55-intercept_form_submit.conf` and uncomment the `LoadModule` line.

2. Add a new section to your `openshift-proxy.conf` file inside the `<VirtualHost *:443>` block.

```http
<Location /login-proxy/oauth/authorize>
    # Insert your backend server name/ip here.
    ProxyPass https://openshift.example.com:8443/oauth/authorize

    InterceptFormPAMService openshift
    InterceptFormLogin httpd_username
    InterceptFormPassword httpd_password

    RewriteCond %{REQUEST_METHOD} GET
    RewriteRule ^.*$ /login.html [L]
</Location>
```

This tells Apache to listen for POST requests on the `/login-proxy/oauth/authorize` and to pass the user name and password over to the `openshift` PAM service.

3. Restart the service and move back over to the OpenShift Enterprise configuration.

#### 10.3.5. OpenShift Enterprise Configuration
1. In the `master-config.yaml` file, update the `identityProviders` section:

   ```yaml
   identityProviders:
     - name: any_provider_name
       challenge: true
       login: true
       mappingMethod: claim
       provider:
         apiVersion: v1
         kind: RequestHeaderIdentityProvider
         challengeURL: "https://proxy.example.com/challenging-proxy/oauth/authorize?${query}"
         loginURL: "https://proxy.example.com/login-proxy/oauth/authorize?${query}"
         clientCA: /etc/origin/master/proxy/proxyca.crt
         headers:
           - X-Remote-User
   ``

   1. Note that `login` is set to `true`, not `false`.
   2. Newly added line.

2. Restart OpenShift Enterprise with the updated configuration.

    **NOTE**

    You should be able to browse to `https://openshift.example.com:8443` and use your LDAP credentials to sign in via the login form.

### 10.4. Configuring Extended LDAP Attributes

#### 10.4.1. Overview

This topic builds upon Setting up SSSD for LDAP Failover and Configuring Form-Based Authentication and focuses on configuring extended Lightweight Directory Access Protocol (LDAP) attributes.

#### 10.4.2. Prerequisites

- SSSD 1.12.0 or later. This is available on Red Hat Enterprise Linux 7.0 and later.

- `mod_lookup_identity` 0.9.4 or later.
  - The required version is not yet available on any version of Red Hat Enterprise Linux. However, compatible packages (RPMs) are available from upstream until they arrive in Red Hat Enterprise Linux.

#### 10.4.3. Configuring SSSD

You need to ask System Security Services Daemon (SSSD) to look up attributes in LDAP that it normally does not care about for simple system-login use-cases. In the case of OpenShift Enterprise, there is only one such attribute: email. So, you need to:
1. Modify the [domain/DOMAINNAME] section of `/etc/sssd/sssd.conf` on the authenticating proxy and add this attribute:

```bash
domain/example.com
...ldap_user_extra_attrs = mail
```

2. Tell SSSD that it is acceptable for this attribute to be retrieved by Apache. Add the following two lines to the [ifp] section of `etc/sssd/sssd.conf`:

```bash
[ifp]
user_attributes = +mail
allowed_uids = apache, root
```

3. Restart SSSD:

```bash
# systemctl restart sssd.service
```

4. Test this configuration.

### 10.4.4. Configuring Apache

Now that SSSD is set up and successfully serving extended attributes, configure the web server to ask for them and to insert them in the correct places.

1. Enable the module to be loaded by Apache. To do so, modify `/etc/httpd/conf.modules.d/55-lookup_identity.conf` and uncomment the line:

```bash
LoadModule lookup_identity_module modules/mod_lookup_identity.so
```

2. Set an SELinux boolean so that SElinux allows Apache to connect to SSSD over D-BUS:

```bash
# setsebool -P httpd_dbus_sssd on
```

3. Edit `/etc/httpd/conf.d/openshift-proxy.conf` and add the following lines inside the `<ProxyMatch /oauth/authorize>` section:

```bash
<ProxyMatch /oauth/authorize>
  AuthName openshift
  LookupOutput Headers
    LookupUserAttr mail X-Remote-User-Email
    LookupUserGECOS X-Remote-User-Display-Name
  RequestHeader set X-Remote-User %{REMOTE_USER}s env=REMOTE_USER
</ProxyMatch>
```

4. Restart Apache to pick up the changes:

```bash
# systemctl restart httpd.service
```
10.4.5. Configuring OpenShift Enterprise

Tell OpenShift Enterprise where to find these new attributes during login. To do so:

1. Edit the `/etc/origin/master/master-config.yaml` file and add the following lines to the `identityProviders` section:

   ```yaml
   identityProviders:
   - name: sssd
     challenge: true
     login: true
     mappingMethod: claim
     provider:
       apiVersion: v1
       kind: RequestHeaderIdentityProvider
       challengeURL: "https://proxy.example.com/challenging-proxy/oauth/authorize?${query}"
       loginURL: "https://proxy.example.com/login-proxy/oauth/authorize?${query}"
       clientCA: 
         /home/example/workspace/openshift/configs/openshift.example.com/proxy/proxyca.crt
       headers:
       - X-Remote-User
       emailHeaders:
       - X-Remote-User-Email
       nameHeaders:
       - X-Remote-User-Display-Name
   ```

2. Launch OpenShift Enterprise with this updated configuration and log in to the web as a new user.
   You should see their full name appear in the upper-right of the screen. You can also verify with `oc get identities -o yaml` that both email addresses and full names are available.

10.4.6. Debugging Notes

Currently, OpenShift Enterprise only saves these attributes to the user at the time of the first login and does not update them again after that. So, while you are testing (and only while testing), run `oc delete users,identities --all` to clear the identities out so you can log in again.
CHAPTER 11. CONFIGURING THE SDN

11.1. OVERVIEW

The OpenShift SDN enables communication between pods across the OpenShift cluster, establishing a pod network. Two SDN plug-ins are currently available (ovs-subnet and ovs-multitenant), which provide different methods for configuring the pod network.

For initial advanced installations, the ovs-subnet plug-in is installed and configured by default, though it can be overridden during installation using the os_sdn_network_plugin_name parameter.

For initial quick installations, the ovs-subnet plug-in is installed and configured by default as well, and can be reconfigured post-installation.

11.2. CONFIGURING THE POD NETWORK ON MASTERS

Cluster administrators can control pod network settings on masters by modifying parameters in the networkConfig section of the master configuration file (located at /etc/origin/master/master-config.yaml by default):

```
networkConfig:
  clusterNetworkCIDR: 10.1.0.0/16
  hostSubnetLength: 8
  networkPluginName: "redhat/openshift-ovs-subnet"
  serviceNetworkCIDR: 172.30.0.0/16
```

1. Cluster network for node IP allocation
2. Number of bits for pod IP allocation within a node
3. Set to redhat/openshift-ovs-subnet for the ovs-subnet plug-in or redhat/openshift-ovs-multitenant for the ovs-multitenant plug-in
4. Service IP allocation for the cluster

**IMPORTANT**

The serviceNetworkCIDR and hostSubnetLength values cannot be changed after the cluster is first created, and clusterNetworkCIDR can only be changed to be a larger network that still contains the original network. For example, given the default value of 10.1.0.0/16, you could change clusterNetworkCIDR to 10.0.0.0/15 (i.e., 10.0.0.0/16 plus 10.1.0.0/16) but not to 10.2.0.0/16, because that does not overlap the original value.

11.3. CONFIGURING THE POD NETWORK ON NODES

Cluster administrators can control pod network settings on nodes by modifying parameters in the networkConfig section of the node configuration file (located at /etc/origin/node/node-config.yaml by default):
networkConfig:
  mtu: 1450
  networkPluginName: "redhat/openshift-ovs-subnet"

1. Maximum transmission unit (MTU) for the pod overlay network
2. Set to redhat/openshift-ovs-subnet for the ovs-subnet plug-in or redhat/openshift-ovs-multitenant for the ovs-multitenant plug-in

11.4. MIGRATING BETWEEN SDN PLUG-INS

If you are already using one SDN plug-in and want to switch to another:

1. Change the networkPluginName parameter on all masters and nodes in their configuration files.
2. Restart the atomic-openshift-master service on masters and the atomic-openshift-node service on nodes.

When switching from the ovs-subnet to the ovs-multitenant plug-in, all the existing projects in the cluster will be fully isolated (assigned unique VNIDs). Cluster administrators can choose to modify the project networks using the administrator CLI.

11.5. EXTERNAL ACCESS TO THE CLUSTER NETWORK

If a host that is external to OpenShift requires access to the cluster network, you have two options:

1. Configure the host as an OpenShift node but mark it unschedulable so that the master does not schedule containers on it.
2. Create a tunnel between your host and a host that is on the cluster network.

Both options are presented as part of a practical use-case in the documentation for configuring routing from an edge load-balancer to containers within OpenShift SDN.
CHAPTER 12. CONFIGURING FOR AWS

12.1. OVERVIEW

OpenShift can be configured to access an AWS EC2 infrastructure, including using AWS volumes as persistent storage for application data. After AWS is configured properly, some additional configurations will need to be completed on the OpenShift hosts.

12.2. CONFIGURING AWS VARIABLES

To set the required AWS variables, create a `/etc/aws/aws.conf` file with the following contents on all of your OpenShift hosts, both masters and nodes:

```
[Global]
Zone = us-east-1c
```

This is the Availability Zone of your AWS Instance and where your EBS Volume resides; this information is obtained from the AWS Management Console.

12.3. CONFIGURING MASTERS

Edit or create the master configuration file on all masters (`/etc/origin/master/master-config.yaml` by default) and update the contents of the `apiServerArguments` and `controllerArguments` sections:

```
kubernetesMasterConfig:
  ...
  apiServerArguments:
    cloud-provider:
      - "aws"
    cloud-config:
      - "/etc/aws/aws.conf"
  controllerArguments:
    cloud-provider:
      - "aws"
    cloud-config:
      - "/etc/aws/aws.conf"
```

12.4. CONFIGURING NODES

Edit or create the node configuration file on all nodes (`/etc/origin/node/node-config.yaml` by default) and update the contents of the `kubeletArguments` section:

```
kubeletArguments:
  cloud-provider:
    - "aws"
  cloud-config:
    - "/etc/aws/aws.conf"
```

12.5. SETTING KEY VALUE ACCESS PAIRS
Make sure the following environment variables are set in the `/etc/sysconfig/atomic-openshift-master` file on masters and the `/etc/sysconfig/atomic-openshift-node` file on nodes:

```bash
AWS_ACCESS_KEY_ID=<key_ID>
AWS_SECRET_ACCESS_KEY=<secret_key>
```

**NOTE**
Access keys are obtained when setting up your AWS IAM user.

### 12.6. APPLYING CONFIGURATION CHANGES

Start or restart OpenShift services on all master and node hosts to apply your configuration changes:

```bash
# systemctl restart atomic-openshift-master
# systemctl restart atomic-openshift-node
```

Switching from not using a cloud provider to using a cloud provider produces an error message. Adding the cloud provider tries to delete the node because the node switches from using the `hostname` as the `externalID` (which would have been the case when no cloud provider was being used) to using the AWS `instance-id` (which is what the AWS cloud provider specifies). To resolve this issue:

1. Log in to the CLI as a cluster administrator.

2. Delete the nodes:

   ```bash
   $ oc delete node <node_name>
   ```

3. On each node host, restart the `atomic-openshift-node` service.

4. Add back any labels on each node that you previously had.
CHAPTER 13. CONFIGURING FOR OPENSTACK

13.1. OVERVIEW

When deployed on OpenStack, OpenShift can be configured to access OpenStack infrastructure, including using OpenStack Cinder volumes as persistent storage for application data.

13.2. CONFIGURING OPENSTACK VARIABLES

To set the required OpenStack variables, create a `/etc/cloud.conf` file with the following contents on all of your OpenShift hosts, both masters and nodes:

```
[Global]
auth-url = <OS_AUTH_URL>
username = <OS_USERNAME>
password = <password>
tenant-id = <OS_TENANT_ID>
region = <OS_REGION_NAME>

[LoadBalancer]
subnet-id = <UUID of the load balancer subnet>
```

Consult your OpenStack administrators for values of the `OS_` variables, which are commonly used in OpenStack configuration.

13.3. CONFIGURING MASTERS

Edit or create the master configuration file on all masters (`/etc/origin/master/master-config.yaml` by default) and update the contents of the `apiServerArguments` and `controllerArguments` sections:

```
kubernetesMasterConfig:
  ...
  apiServerArguments:
    cloud-provider:
      - "openstack"
    cloud-config:
      - "/etc/cloud.conf"
  controllerArguments:
    cloud-provider:
      - "openstack"
    cloud-config:
      - "/etc/cloud.conf"
```

13.4. CONFIGURING NODES

Edit or create the node configuration file on all nodes (`/etc/origin/node/node-config.yaml` by default) and update the contents of the `kubeletArguments` and `nodeName` sections:

```
nodeName:
  <instance_name> 1

kubeletArguments:
```

---

1

---
cloud-provider:
  - "openstack"
cloud-config:
  - "/etc/cloud.conf"

1 Name of the OpenStack instance where the node runs (i.e., name of the virtual machine)
CHAPTER 14. CONFIGURING FOR GCE

14.1. OVERVIEW

OpenShift can be configured to access an GCE infrastructure, including using GCE volumes as persistent storage for application data. After GCE is configured properly, some additional configurations will need to be completed on the OpenShift hosts.

14.2. CONFIGURING MASTERS

Edit or create the master configuration file on all masters (\texttt{/etc/origin/master/master-config.yaml} by default) and update the contents of the \texttt{apiServerArguments} and \texttt{controllerArguments} sections:

\begin{verbatim}
    kubernetesMasterConfig:
        ...
        apiServerArguments:
          cloud-provider:
            - "gce"
        controllerArguments:
          cloud-provider:
            - "gce"
\end{verbatim}

14.3. CONFIGURING NODES

Edit or create the node configuration file on all nodes (\texttt{/etc/origin/node/node-config.yaml} by default) and update the contents of the \texttt{kubeletArguments} section:

\begin{verbatim}
    kubeletArguments:
      cloud-provider:
        - "gce"
\end{verbatim}

Then, start or restart the OpenShift services on the master and all nodes.
CHAPTER 15. CONFIGURING PERSISTENT STORAGE

15.1. OVERVIEW

The Kubernetes persistent volume framework allows you to provision an OpenShift cluster with persistent storage using networked storage available in your environment. This can be done after completing the initial OpenShift installation depending on your application needs, giving users a way to request those resources without having any knowledge of the underlying infrastructure.

These topics show how to configure persistent volumes in OpenShift using the following supported volume plug-ins:

- NFS
- GlusterFS
- OpenStack Cinder
- Ceph RBD
- AWS Elastic Block Store (EBS)
- GCE Persistent Disk
- iSCSI
- Fibre Channel

15.2. PERSISTENT STORAGE USING NFS

15.2.1. Overview

OpenShift clusters can be provisioned with persistent storage using NFS. Persistent volumes (PVs) and persistent volume claims (PVCs) provide a convenient method for sharing a volume across a project. While the NFS-specific information contained in a PV definition could also be defined directly in a pod definition, doing so does not create the volume as a distinct cluster resource, making the volume more susceptible to conflicts.

This topic covers the specifics of using the NFS persistent storage type. Some familiarity with OpenShift and NFS is beneficial. See the Persistent Storage concept topic for details on the OpenShift persistent volume (PV) framework in general.

15.2.2. Provisioning

Storage must exist in the underlying infrastructure before it can be mounted as a volume in OpenShift. To provision NFS volumes, a list of NFS servers and export paths are all that is required.

You must first create an object definition for the PV:

---

Example 15.1. PV Object Definition Using NFS

```yaml
apiVersion: v1
type:PersistentVolume
metadata:
```

---

192
The name of the volume. This is the PV identity in various oc <command> pod commands.

The amount of storage allocated to this volume.

Though this appears to be related to controlling access to the volume, it is actually used similarly to labels and used to match a PVC to a PV. Currently, no access rules are enforced based on the `accessModes`.

The volume type being used, in this case the `nfs` plug-in.

The path that is exported by the NFS server.

The host name or IP address of the NFS server.

The reclaim policy for the PV. This defines what happens to a volume when released from its claim. Valid options are `Retain` (default) and `Recycle`. See Reclaiming Resources.

---

**NOTE**

Each NFS volume must be mountable by all schedulable nodes in the cluster.

Save the definition to a file, for example `nfs-pv.yaml`, and create the PV:

```bash
$ oc create -f nfs-pv.yaml
persistentvolume "pv0001" created
```

Verify that the PV was created:

```bash
# oc get pv
NAME   LABELS     CAPACITY        ACCESSMODES   STATUS
CLAIM  REASON     AGE
pv0001 <none>    5368709120   RWO           Available
```

The next step can be to create a PVC, which binds to the new PV:

**Example 15.2. PVC Object Definition**

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
```
15.2.3. Enforcing Disk Quotas

You can use disk partitions to enforce disk quotas and size constraints. Each partition can be its own export. Each export is one PV. OpenShift enforces unique names for PVs, but the uniqueness of the NFS volume’s server and path is up to the administrator.

Enforcing quotas in this way allows the developer to request persistent storage by a specific amount (for example, 10Gi) and be matched with a corresponding volume of equal or greater capacity.

15.2.4. NFS Volume Security

This section covers NFS volume security, including matching permissions and SELinux considerations. The user is expected to understand the basics of POSIX permissions, process UIDs, supplemental groups, and SELinux.

**NOTE**

See the full Volume Security topic before implementing NFS volumes.

Developers request NFS storage by referencing, in the **volumes** section of their pod definition, either a PVC by name or the NFS volume plug-in directly.

The `/etc/exports` file on the NFS server contains the accessible NFS directories. The target NFS directory has POSIX owner and group IDs. The OpenShift NFS plug-in mounts the container’s NFS directory with the same POSIX ownership and permissions found on the exported NFS directory. However, the container is not run with its effective UID equal to the owner of the NFS mount, which is the desired behavior.

As an example, if the target NFS directory appears on the NFS server as:

```
# ls -lZ /opt/nfs -d
drwxrws---. nfsnobody 5555 unconfined_u:object_r:usr_t:s0   /opt/nfs
# id nfsnobody
```

```yaml
metadata:
  name: nfs-claim1
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi

Save the definition to a file, for example `nfs-claim.yaml`, and create the PVC:

```
# oc create -f nfs-claim.yaml
```
uid=65534(nfsnobody) gid=65534(nfsnobody) groups=65534(nfsnobody)

Then the container must match SELinux labels, and either run with a UID of 65534 (nfsnobody owner) or with 5555 in its supplemental groups in order to access the directory.

NOTE

The owner ID of 65534 is used as an example. Even though NFS’s root_squash maps root (0) to nfsnobody (65534), NFS exports can have arbitrary owner IDs. Owner 65534 is not required for NFS exports.

15.2.4.1. Group IDs

The recommended way to handle NFS access (assuming it is not an option to change permissions on the NFS export) is to use supplemental groups. Supplemental groups in OpenShift are used for shared storage, of which NFS is an example. In contrast, block storage, such as Ceph RBD or iSCSI, use the fsGroup SCC strategy and the fsGroup value in the pod’s securityContext.

NOTE

It is generally preferable to use supplemental group IDs to gain access to persistent storage versus using user IDs. Supplemental groups are covered further in the full Volume Security topic.

Because the group ID on the example target NFS directory shown above is 5555, the pod can define that group ID using supplementalGroups under the pod-level securityContext definition. For example:

```
    securityContext:  
        supplementalGroups: [5555]  
```

1 securityContext must be defined at the pod level, not under a specific container.

2 An array of GIDs defined for the pod. In this case, there is one element in the array; additional GIDs would be comma-separated.

Assuming there are no custom SCCs that might satisfy the pod’s requirements, the pod likely matches the restricted SCC. This SCC has the supplementalGroups strategy set to RunAsAny, meaning that any supplied group ID is accepted without range checking.

As a result, the above pod passes admissions and is launched. However, if group ID range checking is desired, a custom SCC, as described in pod security and custom SCCs, is the preferred solution. A custom SCC can be created such that minimum and maximum group IDs are defined, group ID range checking is enforced, and a group ID of 5555 is allowed.

15.2.4.2. User IDs
User IDs can be defined in the container image or in the pod definition. The full Volume Security topic covers controlling storage access based on user IDs, and should be read prior to setting up NFS persistent storage.

**NOTE**

It is generally preferable to use supplemental group IDs to gain access to persistent storage versus using user IDs.

In the example target NFS directory shown above, the container needs its UID set to 65534 (ignoring group IDs for the moment), so the following can be added to the pod definition:

```yaml
spec:
  containers:
    - name: ...
      securityContext:
        runAsUser: 65534
```

1. Pods contain a `securityContext` specific to each container (shown here) and a pod-level `securityContext` which applies to all containers defined in the pod.

2. 65534 is the `nfsnobody` user.

Assuming the default project and the restricted SCC, the pod’s requested user ID of 65534 is not allowed, and therefore the pod fails. The pod fails for the following reasons:

- It requests 65534 as its user ID.
- All SCCs available to the pod are examined to see which SCC allows a user ID of 65534 (actually, all policies of the SCCs are checked but the focus here is on user ID).
- Because all available SCCs use `MustRunAsRange` for their `runAsUser` strategy, UID range checking is required.
- 65534 is not included in the SCC or project’s user ID range.

It is generally considered a good practice not to modify the predefined SCCs. The preferred way to fix this situation is to create a custom SCC, as described in the full Volume Security topic. A custom SCC can be created such that minimum and maximum user IDs are defined, UID range checking is still enforced, and the UID of 65534 is allowed.

15.2.4.3. SELinux

**NOTE**

See the full Volume Security topic for information on controlling storage access in conjunction with using SELinux.

By default, SELinux does not allow writing from a pod to a remote NFS server. The NFS volume mounts correctly, but is read-only.

To enable writing to NFS volumes with SELinux enforcing on each node, run:
# setsebool -P virt_use_nfs 1
# setsebool -P virt_sandbox_use_nfs 1

The -P option above makes the bool persistent between reboots.

Thevirt_use_nfs boolean is defined by the docker-selinuxpackage. If an error is seen indicating that this bool is not defined, ensure this package has been installed.

### 15.2.4.4. Export Settings

In order to enable arbitrary container users to read and write the volume, each exported volume on the NFS server should conform to the following conditions:

- Each export must be:
  
  /<example_fs> *(rw, root_squash, no_wdelay)

  The no_wdelay option prevents the server from delaying writes, which greatly improves read-after-write consistency.

- The firewall must be configured to allow traffic to the mount point. For NFSv4, the default port is 2049 (nfs). For NFSv3, there are three ports to configure: 2049 (nfs), 20048 (mountd), and 111 (portmapper).

**NFSv4**

```
# iptables -I INPUT 1 -p tcp --dport 2049 -j ACCEPT
```

**NFSv3**

```
# iptables -I INPUT 1 -p tcp --dport 2049 -j ACCEPT
# iptables -I INPUT 1 -p tcp --dport 20048 -j ACCEPT
# iptables -I INPUT 1 -p tcp --dport 111 -j ACCEPT
```

- The NFS export and directory must be set up so that it is accessible by the target pods. Either set the export to be owned by the container’s primary UID, or supply the pod group access using supplementalGroups, as shown in Group IDs above. See the full Volume Security topic for additional pod security information as well.

### 15.2.5. Reclaiming Resources

NFS implements the OpenShift Recyclable plug-in interface. Automatic processes handle reclamation tasks based on policies set on each persistent volume.

By default, PVs are set to Retain. NFS volumes which are set to Recycle are scrubbed (i.e., rm -rf is run on the volume) after being released from their claim (i.e., after the user’s PersistentVolumeClaim bound to the volume is deleted). Once recycled, the NFS volume can be bound to a new claim.

Once claim to a PV is released (that is, the PVC is deleted), the PV object should not be re-used. Instead, a new PV should be created with the same basic volume details as the original.

For example, the administrator creates a PV named nfs1:
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfs1
spec:
  capacity:
    storage: 1Mi
  accessModes:
  - ReadWriteMany
  nfs:
    server: 192.168.1.1
    path: "/

The user creates PVC1, which binds to nfs1. The user then deletes PVC1, releasing claim to nfs1, which causes nfs1 to be Released. If the administrator wishes to make the same NFS share available, they should create a new PV with the same NFS server details, but a different PV name:

apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfs2
spec:
  capacity:
    storage: 1Mi
  accessModes:
  - ReadWriteMany
  nfs:
    server: 192.168.1.1
    path: "/

Deleting the original PV and re-creating it with the same name is discouraged. Attempting to manually change the status of a PV from Released to Available causes errors and potential data loss.

NOTE

A PV with retention policy of Recycle scrubs (rm -rf) the data and marks it as Available for claim. The Recycle retention policy is deprecated starting in OpenShift Enterprise 3.6 and should be avoided. Anyone using recycler should use dynamic provision and volume deletion instead.

15.2.6. Automation

Clusters can be provisioned with persistent storage using NFS in the following ways:

- Enforce storage quotas using disk partitions.
- Enforce security by restricting volumes to the project that has a claim to them.
- Configure reclamation of discarded resources for each PV.

They are many ways that you can use scripts to automate the above tasks. You can use an example Ansible playbook to help you get started.

15.2.7. Additional Configuration and Troubleshooting
Depending on what version of NFS is being used and how it is configured, there may be additional configuration steps needed for proper export and security mapping. The following are some that may apply:

| NFSv4 mount incorrectly shows all files with ownership of nobody:nobody | - Could be attributed to the ID mapping settings (/etc/idmapd.conf) on your NFS
| | - See this Red Hat Solution.

### Disabling ID mapping on NFSv4

On both the NFS client and server, run:

```
# echo 'Y' > /sys/module/nfsd/parameters/nfs4_disable_idmapping
```

### 15.3. PERSISTENT STORAGE USING GLUSTERFS

#### 15.3.1. Overview

OpenShift Enterprise clusters can be provisioned with persistent storage using GlusterFS.

Persistent volumes (PVs) and persistent volume claims (PVCs) can share volumes across a single project. While the GlusterFS-specific information contained in a PV definition could also be defined directly in a pod definition, doing so does not create the volume as a distinct cluster resource, making the volume more susceptible to conflicts.

This topic presumes some familiarity with OpenShift Enterprise and GlusterFS. See the Persistent Storage topic for details on the OpenShift Enterprise PV framework in general.

#### IMPORTANT

High-availability of storage in the infrastructure is left to the underlying storage provider.

#### 15.3.2. Provisioning

To provision GlusterFS volumes the following are required:

- An existing storage device in your underlying infrastructure.
- A distinct list of servers (IP addresses) in the Gluster cluster, to be defined as endpoints.
- A service, to persist the endpoints (optional).
- An existing Gluster volume to be referenced in the persistent volume object.
- `glusterfs-fuse` installed on each schedulable OpenShift Enterprise node in your cluster:

```
# yum install glusterfs-fuse
```
NOTE

OpenShift Enterprise nodes can also host a gluster node (referred to as hyperconverged storage). However, performance may be less predictable and harder to manage.

15.3.2.1. Creating Gluster Endpoints

An endpoints definition defines the GlusterFS cluster as **Endpoints** and includes the IP addresses of your Gluster servers. The port value can be any numeric value within the accepted range of ports. Optionally, you can create a **service** that persists the endpoints.

1. Define the following service:

   **Example 15.3. Gluster Service Definition**

   ```yaml
   apiVersion: v1
   kind: Service
   metadata:
     name: glusterfs-cluster
   spec:
     ports:
       - port: 1
   ```

   This name must be defined in the endpoints definition to match the endpoints to this service.

2. Save the service definition to a file, for example `gluster-service.yaml`, then create the service:

   ```bash
   $ oc create -f gluster-service.yaml
   ```

3. Verify that the service was created:

   ```bash
   # oc get services
   NAME                       CLUSTER_IP       EXTERNAL_IP   PORT(S)
   SELECTOR        AGE
   glusterfs-cluster          172.30.205.34    <none>        1/TCP
   <none>          44s
   ```

4. Define the Gluster endpoints:

   **Example 15.4. Gluster Endpoints Definition**

   ```yaml
   apiVersion: v1
   kind: Endpoints
   metadata:
     name: glusterfs-cluster
   subsets:
     - addresses:
       - ip: 192.168.122.221
       ports:
         - port: 1
   ```
This name must match the service name from step 1.

The ip values must be the actual IP addresses of a Gluster server, not fully-qualified host names.

The port number is ignored.

5. Save the endpoints definition to a file, for example `gluster-endpoints.yaml`, then create the endpoints:

   ```bash
   $ oc create -f gluster-endpoints.yaml
   endpoints "glusterfs-cluster" created
   ```

6. Verify that the endpoints were created:

   ```bash
   $ oc get endpoints
   NAME                ENDPOINTS                             AGE
   docker-registry     10.1.0.3:5000                         4h
   glusterfs-cluster   192.168.122.221:1,192.168.122.222:1   11s
   kubernetes          172.16.35.3:8443                      4d
   ```

15.3.2.2. Creating the Persistent Volume

1. Next, define the PV in an object definition before creating it in OpenShift Enterprise:

   **Example 15.5. Persistent Volume Object Definition Using GlusterFS**

   ```yaml
   apiVersion: v1
   kind: PersistentVolume
   metadata:
     name: gluster-default-volume
   spec:
     capacity:
       storage: 2Gi
     accessModes:
       - ReadWriteMany
     glusterfs:
       endpoints: glusterfs-cluster
       path: myVol1
       readOnly: false
     persistentVolumeReclaimPolicy: Recycle
   ```

   The name of the volume. This is how it is identified via persistent volume claims or from pods.

   The amount of storage allocated to this volume.
accessModes are used as labels to match a PV and a PVC. They currently do not define any form of access control.

4 The volume type being used, in this case the glusterfs plug-in.

5 The endpoints name that defines the Gluster cluster created in Creating Gluster Endpoints.

6 The Gluster volume that will be accessed, as shown in the gluster volume status command.

2. Save the definition to a file, for example gluster-pv.yaml, and create the persistent volume:

```bash
# oc create -f gluster-pv.yaml
```

3. Verify that the persistent volume was created:

```bash
# oc get pv
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>LABELS</th>
<th>CAPACITY</th>
<th>ACCESSMODES</th>
</tr>
</thead>
<tbody>
<tr>
<td>gluster-default-volume</td>
<td>&lt;none&gt;</td>
<td>2147483648</td>
<td>RWX</td>
</tr>
<tr>
<td>Available</td>
<td></td>
<td>2s</td>
<td></td>
</tr>
</tbody>
</table>

15.3.2.3. Creating the Persistent Volume Claim

Developers request GlusterFS storage by referencing either a PVC or the Gluster volume plug-in directly in the volumes section of a pod spec. A PVC exists only in the user's project and can only be referenced by pods within that project. Any attempt to access a PV across a project causes the pod to fail.

1. Create a PVC that will bind to the new PV:

**Example 15.6. PVC Object Definition**

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gluster-claim
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
```

1 accessModes do not enforce security, but rather act as labels to match a PV to a PVC.

2 This claim will look for PVs offering 1Gi or greater capacity.

2. Save the definition to a file, for example gluster-claim.yaml, and create the PVC:

```bash
# oc create -f gluster-claim.yaml
```
PVs and PVCs make sharing a volume across a project simpler. The gluster-specific information contained in the PV definition can also be defined directly in a pod specification.

15.3.3. Gluster Volume Security

This section covers Gluster volume security, including matching permissions and SELinux considerations. Understanding the basics of POSIX permissions, process UIDs, supplemental groups, and SELinux is presumed.

NOTE

See the full Volume Security topic before implementing Gluster volumes.

As an example, assume that the target Gluster volume, HadoopVol is mounted under /mnt/glusterfs/, with the following POSIX permissions and SELinux labels:

```
# ls -lZ /mnt/glusterfs/
```

```
drwxrwx---. yarn hadoop system_u:object_r:fusefs_t:s0    HadoopVol
```

```
# id yarn
```

```
uid=592(yarn) gid=590(hadoop) groups=590(hadoop)
```

In order to access the HadoopVol volume, containers must match the SELinux label, and run with a UID of 592 or 590 in their supplemental groups. The OpenShift Enterprise GlusterFS plug-in mounts the volume in the container with the same POSIX ownership and permissions found on the target gluster mount, namely the owner will be 592 and group ID will be 590. However, the container is not run with its effective UID equal to 592, nor with its GID equal to 590, which is the desired behavior. Instead, a container’s UID and supplemental groups are determined by Security Context Constraints (SCCs) and the project defaults.

15.3.3.1. Group IDs

Configure Gluster volume access by using supplemental groups, assuming it is not an option to change permissions on the Gluster mount. Supplemental groups in OpenShift Enterprise are used for shared storage, such as GlusterFS. In contrast, block storage, such as Ceph RBD or iSCSI, use the fsGroup SCC strategy and the fsGroup value in the pod’s securityContext.

NOTE

Use supplemental group IDs instead of user IDs to gain access to persistent storage. Supplemental groups are covered further in the full Volume Security topic.

The group ID on the target Gluster mount example above is 590. Therefore, a pod can define that group ID using supplementalGroups under the pod-level securityContext definition. For example:

```
spec:
  containers:
    - name:
```

securityContext: 1
 supplementalGroups: [590] 2

1 securityContext must be defined at the pod level, not under a specific container.

2 An array of GIDs defined at the pod level.

Assuming there are no custom SCCs that satisfy the pod’s requirements, the pod matches the restricted SCC. This SCC has the supplementalGroups strategy set to RunAsAny, meaning that any supplied group IDs are accepted without range checking.

As a result, the above pod will pass admissions and can be launched. However, if group ID range checking is desired, use a custom SCC, as described in pod security and custom SCCs. A custom SCC can be created to define minimum and maximum group IDs, enforce group ID range checking, and allow a group ID of 590.

15.3.3.2. User IDs

User IDs can be defined in the container image or in the pod definition. The full Volume Security topic covers controlling storage access based on user IDs, and should be read prior to setting up NFS persistent storage.

NOTE
Use supplemental group IDs instead of user IDs to gain access to persistent storage.

In the target Gluster mount example above, the container needs a UID set to 592, so the following can be added to the pod definition:

```
spec:
  containers:
    - name: ...
      securityContext:
        runAsUser: 592
```

1 Pods contain a securityContext specific to each container and a pod-level securityContext, which applies to all containers defined in the pod.

2 The UID defined on the Gluster mount.

With the default project and the restricted SCC, a pod’s requested user ID of 592 will not be allowed, and the pod will fail. This is because:

- The pod requests 592 as its user ID.
- All SCCs available to the pod are examined to see which SCC will allow a user ID of 592.
- Because all available SCCs use MustRunAsRange for their runAsUser strategy, UID range checking is required.
Do not modify the predefined SCCs. Instead, create a custom SCC so that minimum and maximum user IDs are defined, UID range checking is still enforced, and the UID of 592 will be allowed.

### 15.3.3.3. SELinux

**NOTE**

See the full [Volume Security](#) topic for information on controlling storage access in conjunction with using SELinux.

By default, SELinux does not allow writing from a pod to a remote Gluster server.

To enable writing to GlusterFS volumes with SELinux enforcing on each node, run:

```bash
$ sudo setsebool -P virt_sandbox_use_fusefs on
```

**NOTE**

The `virt_sandbox_use_fusefs` boolean is defined by the `docker-selinux` package. If you get an error saying it is not defined, please ensure that this package is installed.

The `-P` option makes the bool persistent between reboots.

### 15.4. PERSISTENT STORAGE USING OPENSTACK CINDER

#### 15.4.1. Overview

You can provision your OpenShift cluster with [persistent storage](#) using OpenStack Cinder. Some familiarity with Kubernetes and OpenStack is assumed.

**IMPORTANT**

Before creating persistent volumes using Cinder, OpenShift must first be properly configured for OpenStack.

The Kubernetes [persistent volume](#) framework allows administrators to provision a cluster with persistent storage and gives users a way to request those resources without having any knowledge of the underlying infrastructure. Persistent volumes are not bound to a single project or namespace; they can be shared across the OpenShift cluster. [Persistent volume claims](#), however, are specific to a project or namespace and can be requested by users.

For a detailed example, see the guide for [WordPress and MySQL using persistent volumes](#).

**IMPORTANT**

High-availability of storage in the infrastructure is left to the underlying storage provider.

#### 15.4.2. Provisioning
Storage must exist in the underlying infrastructure before it can be mounted as a volume in OpenShift. After ensuring OpenShift is configured for OpenStack, all that is required for Cinder is a Cinder volume ID and the PersistentVolume API.

15.4.2.1. Creating the Persistent Volume

You must define your persistent volume in an object definition before creating it in OpenShift:

Example 15.7. Persistent Volume Object Definition Using Cinder

```yaml
apiVersion: "v1"
kind: "PersistentVolume"
metadata:
  name: "pv0001"
spec:
  capacity:
    storage: "5Gi"
  accessModes:
    - "ReadWriteOnce"
  cinder:
    fsType: "ext3"
    volumeID: "f37a03aa-6212-4c62-a805-9ce139fab180"
```

1. The name of the volume. This will be how it is identified via persistent volume claims or from pods.
2. The amount of storage allocated to this volume.
3. This defines the volume type being used, in this case the cinder plug-in.
4. File system type to mount.
5. This is the Cinder volume that will be used.

**IMPORTANT**

Changing the value of the fsType parameter after the volume has been formatted and provisioned can result in data loss and pod failure.

Save your definition to a file, for example `cinder-pv.yaml`, and create the persistent volume:

```
# oc create -f cinder-pv.yaml
persistentvolume "pv0001" created
```

Verify that the persistent volume was created:

```
# oc get pv
NAME     LABELS     CAPACITY  ACCESSModes  STATUS   CLAIM         REASON
pv0001   <none>     5Gi       RWO         Available
```

206
Users can then request storage using persistent volume claims, which can now utilize your new persistent volume.

**IMPORTANT**

Persistent volume claims only exist in the user's namespace and can only be referenced by a pod within that same namespace. Any attempt to access a persistent volume from a different namespace causes the pod to fail.

### 15.4.2.2. Volume Format

Before OpenShift mounts the volume and passes it to a container, it checks that it contains a file system as specified by the *fsType* parameter in the persistent volume definition. If the device is not formatted with the file system, all data from the device is erased and the device is automatically formatted with the given file system.

This allows using unformatted Cinder volumes as persistent volumes, because OpenShift Enterprise formats them before the first use.

### 15.5. PERSISTENT STORAGE USING CEPH RADOS BLOCK DEVICE (RBD)

#### 15.5.1. Overview

OpenShift Enterprise clusters can be provisioned with persistent storage using Ceph RBD.

Persistent volumes (PVs) and persistent volume claims (PVCs) can share volumes across a single project. While the Ceph RBD-specific information contained in a PV definition could also be defined directly in a pod definition, doing so does not create the volume as a distinct cluster resource, making the volume more susceptible to conflicts.

This topic presumes some familiarity with OpenShift Enterprise and Ceph RBD. See the Persistent Storage concept topic for details on the OpenShift Enterprise persistent volume (PV) framework in general.

**NOTE**

*Project* and *namespace* are used interchangeably throughout this document. See Projects and Users for details on the relationship.

**IMPORTANT**

High-availability of storage in the infrastructure is left to the underlying storage provider.

#### 15.5.2. Provisioning

To provision Ceph volumes, the following are required:

- An existing storage device in your underlying infrastructure.
- The Ceph key to be used in an OpenShift Enterprise secret object.
- The Ceph image name.
- The file system type on top of the block storage (e.g., ext4).

- `ceph-common` installed on each schedulable OpenShift Enterprise node in your cluster:

  ```bash
  # yum install ceph-common
  ```

### 15.5.2.1. Creating the Ceph Secret

Define the authorization key in a secret configuration, which is then converted to base64 for use by OpenShift Enterprise.

**NOTE**

In order to use Ceph storage to back a persistent volume, the secret must be created in the same project as the PVC and pod. The secret cannot simply be in the default project.

1. Run `ceph auth get-key` on a Ceph MON node to display the key value for the `client.admin` user:

   ```yaml
   apiVersion: v1
   kind: Secret
   metadata:
     name: ceph-secret
   data:
     key: QVFBOFF2SlZheUJQRVJBQWgvS2cwT1laQUhPQno3akZwekxxdGc9PQ==
   ```

2. Save the secret definition to a file, for example `ceph-secret.yaml`, then create the secret:

   ```bash
   $ oc create -f ceph-secret.yaml
   ```

3. Verify that the secret was created:

   ```bash
   # oc get secret ceph-secret
   NAME          TYPE      DATA      AGE
   ceph-secret   Opaque    1         23d
   ```

### 15.5.2.2. Creating the Persistent Volume

Developers request Ceph RBD storage by referencing either a PVC, or the Gluster volume plug-in directly in the `volumes` section of a pod specification. A PVC exists only in the user's namespace and can be referenced only by pods within that same namespace. Any attempt to access a PV from a different namespace causes the pod to fail.

1. Define the PV in an object definition before creating it in OpenShift Enterprise:

   **Example 15.8. Persistent Volume Object Definition Using Ceph RBD**

   ```yaml
   apiVersion: v1
   kind: PersistentVolume
   metadata:
     name: ceph-pv
   spec:
     capacity:
   ```
storage: 2Gi
accessModes:
- ReadWriteOnce
rbd:
  monitors:
  - 192.168.122.133:6789
pool: rbd
image: ceph-image
user: admin
secretRef:
  name: ceph-secret
fsType: ext4
readOnly: false
persistentVolumeReclaimPolicy: Recycle

1. The name of the PV that is referenced in pod definitions or displayed in various `oc` volume commands.
2. The amount of storage allocated to this volume.
3. `accessModes` are used as labels to match a PV and a PVC. They currently do not define any form of access control. All block storage is defined to be single user (non-shared storage).
4. The volume type being used, in this case the `rbd` plug-in.
5. An array of Ceph monitor IP addresses and ports.
6. The Ceph secret used to create a secure connection from OpenShift Enterprise to the Ceph server.
7. The file system type mounted on the Ceph RBD block device.

**IMPORTANT**

Changing the value of the `fstype` parameter after the volume has been formatted and provisioned can result in data loss and pod failure.

2. Save your definition to a file, for example `ceph-pv.yaml`, and create the PV:

    # oc create -f ceph-pv.yaml

3. Verify that the persistent volume was created:

    # oc get pv

<table>
<thead>
<tr>
<th>NAME</th>
<th>LABELS</th>
<th>CAPACITY</th>
<th>ACCESSMODES</th>
</tr>
</thead>
<tbody>
<tr>
<td>ceph-pv</td>
<td>&lt;none&gt;</td>
<td>2147483648</td>
<td>Rw0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2s</td>
<td></td>
</tr>
</tbody>
</table>

4. Create a PVC that will bind to the new PV:
Example 15.9. PVC Object Definition

```yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: ceph-claim
spec:
  accessModes:  
    - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
```

1. The `accessModes` do not enforce access right, but instead act as labels to match a PV to a PVC.
2. This claim looks for PVs offering 2Gi or greater capacity.

5. Save the definition to a file, for example `ceph-claim.yaml`, and create the PVC:

```
# oc create -f ceph-claim.yaml
```

15.5.3. Ceph Volume Security

**NOTE**

See the full Volume Security topic before implementing Ceph RBD volumes.

A significant difference between shared volumes (NFS and GlusterFS) and block volumes (Ceph RBD, iSCSI, and most cloud storage), is that the user and group IDs defined in the pod definition or docker image are applied to the target physical storage. This is referred to as managing ownership of the block device. For example, if the Ceph RBD mount has its owner set to 123 and its group ID set to 567, and if the pod defines its `runAsUser` set to 222 and its `fsGroup` to be 7777, then the Ceph RBD physical mount’s ownership will be changed to 222:7777.

**NOTE**

Even if the user and group IDs are not defined in the pod specification, the resulting pod may have defaults defined for these IDs based on its matching SCC, or its project. See the full Volume Security topic which covers storage aspects of SCCs and defaults in greater detail.

A pod defines the group ownership of a Ceph RBD volume using the `fsGroup` stanza under the pod’s `securityContext` definition:
The `securityContext` must be defined at the pod level, not under a specific container.

All containers in the pod will have the same fsGroup ID.

15.6. PERSISTENT STORAGE USING AWS ELASTIC BLOCK STORE

15.6.1. Overview

OpenShift supports AWS Elastic Block Store volumes (EBS). You can provision your OpenShift cluster with persistent storage using AWS EC2. Some familiarity with Kubernetes and AWS is assumed.

**IMPORTANT**

Before creating persistent volumes using AWS, OpenShift must first be properly configured for AWS ElasticBlockStore.

The Kubernetes persistent volume framework allows administrators to provision a cluster with persistent storage and gives users a way to request those resources without having any knowledge of the underlying infrastructure. Persistent volumes are not bound to a single project or namespace; they can be shared across the OpenShift cluster. Persistent volume claims, however, are specific to a project or namespace and can be requested by users.

**IMPORTANT**

High-availability of storage in the infrastructure is left to the underlying storage provider.

15.6.2. Provisioning

Storage must exist in the underlying infrastructure before it can be mounted as a volume in OpenShift. After ensuring OpenShift is configured for AWS Elastic Block Store, all that is required for OpenShift and AWS is an AWS EBS volume ID and the `PersistentVolume` API.

15.6.2.1. Creating the Persistent Volume

You must define your persistent volume in an object definition before creating it in OpenShift:

**Example 15.10. Persistent Volume Object Definition Using AWS**

```yaml
apiVersion: "v1"
kind: "PersistentVolume"
metadata:
  name: "pv0001"
spec:
  capacity:
    storage: "5Gi"
  accessModes:
    - "ReadWriteOnce"
```
The name of the volume. This will be how it is identified via persistent volume claims or from pods.

The amount of storage allocated to this volume.

This defines the volume type being used, in this case the `awsElasticBlockStore` plug-in.

File system type to mount.

This is the AWS volume that will be used.

**IMPORTANT**

Changing the value of the `fstype` parameter after the volume has been formatted and provisioned can result in data loss and pod failure.

Save your definition to a file, for example `aws-pv.yaml`, and create the persistent volume:

```
# oc create -f aws-pv.yaml
persistentvolume "pv0001" created
```

Verify that the persistent volume was created:

```
# oc get pv
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>LABELS</th>
<th>CAPACITY</th>
<th>ACCESSMODES</th>
<th>STATUS</th>
<th>CLAIM</th>
<th>REASON</th>
</tr>
</thead>
<tbody>
<tr>
<td>pv0001</td>
<td>&lt;none&gt;</td>
<td>5Gi</td>
<td>RWO</td>
<td>Available</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2s</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Users can then request storage using persistent volume claims, which can now utilize your new persistent volume.

**IMPORTANT**

Persistent volume claims only exist in the user’s namespace and can only be referenced by a pod within that same namespace. Any attempt to access a persistent volume from a different namespace causes the pod to fail.

### 15.6.2.2. Volume Format

Before OpenShift mounts the volume and passes it to a container, it checks that it contains a file system as specified by the `fsType` parameter in the persistent volume definition. If the device is not formatted with the file system, all data from the device is erased and the device is automatically formatted with the given file system.

This allows using unformatted AWS volumes as persistent volumes, because OpenShift Enterprise formats them before the first use.
15.7. PERSISTENT STORAGE USING GCE PERSISTENT DISK

15.7.1. Overview

OpenShift supports GCE Persistent Disk volumes (gcePD). You can provision your OpenShift cluster with persistent storage using GCE. Some familiarity with Kubernetes and GCE is assumed.

IMPORTANT

Before creating persistent volumes using GCE, OpenShift must first be properly configured for GCE Persistent Disk.

The Kubernetes persistent volume framework allows administrators to provision a cluster with persistent storage and gives users a way to request those resources without having any knowledge of the underlying infrastructure. Persistent volumes are not bound to a single project or namespace; they can be shared across the OpenShift cluster. Persistent volume claims, however, are specific to a project or namespace and can be requested by users.

IMPORTANT

High-availability of storage in the infrastructure is left to the underlying storage provider.

15.7.2. Provisioning

Storage must exist in the underlying infrastructure before it can be mounted as a volume in OpenShift. After ensuring OpenShift is configured for GCE PersistentDisk, all that is required for OpenShift and GCE is an GCE Persistent Disk volume ID and the PersistentVolume API.

15.7.2.1. Creating the Persistent Volume

You must define your persistent volume in an object definition before creating it in OpenShift:

Example 15.11. Persistent Volume Object Definition Using GCE

```yaml
apiVersion: "v1"
kind: "PersistentVolume"
metadata:
  name: "pv0001"

spec:
  capacity:
    storage: "5Gi"
  accessModes:
    - "ReadWriteOnce"
  gcePersistentDisk:
    fsType: "ext4"
    pdName: "pd-disk-1"
```

1. The name of the volume. This will be how it is identified via persistent volume claims or from pods.
2. The amount of storage allocated to this volume.
This defines the volume type being used, in this case the `gcePersistentDisk` plug-in.

File system type to mount.

This is the GCE Persistent Disk volume that will be used.

**IMPORTANT**

Changing the value of the `fstype` parameter after the volume has been formatted and provisioned can result in data loss and pod failure.

Save your definition to a file, for example `gce-pv.yaml`, and create the persistent volume:

```
# oc create -f gce-pv.yaml
persistentvolume "pv0001" created
```

Verify that the persistent volume was created:

```
# oc get pv
NAME      LABELS    CAPACITY   ACCESSMODES   STATUS      CLAIM     REASON
pv0001    <none>    5Gi        RWO           Available 2s
```

Users can then request storage using persistent volume claims, which can now utilize your new persistent volume.

**IMPORTANT**

Persistent volume claims only exist in the user's namespace and can only be referenced by a pod within that same namespace. Any attempt to access a persistent volume from a different namespace causes the pod to fail.

### 15.7.2.2. Volume Format

Before OpenShift mounts the volume and passes it to a container, it checks that it contains a file system as specified by the `fstype` parameter in the persistent volume definition. If the device is not formatted with the file system, all data from the device is erased and the device is automatically formatted with the given file system.

This allows using unformatted GCE volumes as persistent volumes, because OpenShift Enterprise formats them before the first use.

### 15.8. PERSISTENT STORAGE USING ISCSI

#### 15.8.1. Overview

You can provision your OpenShift cluster with persistent storage using iSCSI. Some familiarity with Kubernetes and iSCSI is assumed.
The Kubernetes **persistent volume** framework allows administrators to provision a cluster with persistent storage and gives users a way to request those resources without having any knowledge of the underlying infrastructure.

**IMPORTANT**

High-availability of storage in the infrastructure is left to the underlying storage provider.

### 15.8.2. Provisioning

Storage must exist in the underlying infrastructure before it can be mounted as a volume in OpenShift. All that is required for iSCSI is iSCSI target portal, valid iSCSI IQN, valid LUN number, and filesystem type, and the `PersistentVolume` API.

**Example 15.12. Persistent Volume Object Definition**

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: iscsi-pv
spec:
  capacity:
    storage: 1Gi
  accessModes:
  - ReadWriteOnce
  iscsi:
    targetPortal: 10.16.154.81
    iqn: iqn.2014-12.example.server:storage.target00
    lun: 0
    fsType: 'ext4'
    readOnly: false
```

### 15.8.2.1. Enforcing Disk Quotas

Use LUN partitions to enforce disk quotas and size constraints. Each LUN is one persistent volume. Kubernetes enforces unique names for persistent volumes.

Enforcing quotas in this way allows the end user to request persistent storage by a specific amount (e.g., 10Gi) and be matched with a corresponding volume of equal or greater capacity.

### 15.8.2.2. iSCSI Volume Security

Users request storage with a `PersistentVolumeClaim`. This claim only lives in the user’s namespace and can only be referenced by a pod within that same namespace. Any attempt to access a persistent volume across a namespace causes the pod to fail.

Each iSCSI LUN must be accessible by all nodes in the cluster.

### 15.9. PERSISTENT STORAGE USING FIBRE CHANNEL

#### 15.9.1. Overview
You can provision your OpenShift cluster with persistent storage using Fibre Channel. Some familiarity with Kubernetes and Fibre Channel is assumed.

The Kubernetes persistent volume framework allows administrators to provision a cluster with persistent storage and gives users a way to request those resources without having any knowledge of the underlying infrastructure.

**IMPORTANT**

High-availability of storage in the infrastructure is left to the underlying storage provider.

### 15.9.2. Provisioning

Storage must exist in the underlying infrastructure before it can be mounted as a volume in OpenShift. All that is required for Fibre Channel persistent storage is the targetWWNs (array of Fibre Channel target's World Wide Names), a valid LUN number, and filesystem type, and the `PersistentVolume` API. Note, the number of LUNs must correspond to the number of Persistent Volumes that are created. In the example below, we have LUN as 2, therefore we have created two Persistent Volume definitions.

**Example 15.13. Persistent Volumes Object Definition**

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv0001
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  fc:
    targetWWNs: ['500a0981891b8dc5', '500a0981991b8dc5']
    lun: 2
    fsType: ext4

apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv0002
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadOnlyMany
  fc:
    targetWWNs: ['500a0981891b8dc5', '500a0981991b8dc5']
    lun: 2
    fsType: ext4
```
IMPORTANT

Changing the value of the `fstype` parameter after the volume has been formatted and provisioned can result in data loss and pod failure.

15.9.2.1. Enforcing Disk Quotas

Use LUN partitions to enforce disk quotas and size constraints. Each LUN is one persistent volume. Kubernetes enforces unique names for persistent volumes.

Enforcing quotas in this way allows the end user to request persistent storage by a specific amount (e.g., 10Gi) and be matched with a corresponding volume of equal or greater capacity.

15.9.2.2. Fibre Channel Volume Security

Users request storage with a `PersistentVolumeClaim`. This claim only lives in the user’s namespace and can only be referenced by a pod within that same namespace. Any attempt to access a persistent volume across a namespace causes the pod to fail.

Each Fibre Channel LUN must be accessible by all nodes in the cluster.

15.10. DYNAMICALLY PROVISIONING PERSISTENT VOLUMES

15.10.1. Overview

You can provision your OpenShift cluster with storage dynamically when running in a cloud environment. The Kubernetes `persistent volume` framework allows administrators to provision a cluster with persistent storage and gives users a way to request those resources without having any knowledge of the underlying infrastructure.

Many storage types are available for use as persistent volumes in OpenShift. While all of them can be statically provisioned by an administrator, some types of storage can be created dynamically using an API. These types of storage can be provisioned in an OpenShift cluster using the new and experimental dynamic storage feature.

IMPORTANT

Dynamic provisioning of persistent volumes is currently a Technology Preview feature, introduced in OpenShift Enterprise 3.1.1. This feature is experimental and expected to change in the future as it matures and feedback is received from users. New ways to provision the cluster are planned and the means by which one accesses this feature is going to change. Backwards compatibility is not guaranteed.

15.10.2. Enabling Provisioner Plug-ins

OpenShift provides the following provisioner plug-ins, which have generic implementations for dynamic provisioning that use the cluster’s configured cloud provider’s API to create new storage resources:

<table>
<thead>
<tr>
<th>Storage Type</th>
<th>Provisioner Plug-in Name</th>
<th>Required Cloud Configuration</th>
<th>Notes</th>
</tr>
</thead>
</table>
## 15.10.3. Requesting Dynamically Provisioned Storage

Users can request dynamically provisioned storage by including a storage class annotation in their persistent volume claim:
Example 15.14. Persistent Volume Claim Requesting Dynamic Storage

```yaml
kind: "PersistentVolumeClaim"
apiVersion: "v1"
metadata:
  name: "claim1"
  annotations:
    volume.alpha.kubernetes.io/storage-class: "foo"
spec:
  accessModes:
    - "ReadWriteOnce"
  resources:
    requests:
      storage: "3Gi"
```

The value of the `volume.alpha.kubernetes.io/storage-class` annotation is not meaningful at this time. The presence of the annotation, with any arbitrary value, triggers provisioning using the single implied provisioner plug-in per cloud.

15.10.4. Volume Recycling

Volumes created dynamically by a provisioner have their `persistentVolumeReclaimPolicy` set to `Delete`. When a persistent volume claim is deleted, its backing persistent volume is considered released of its claim, and that resource can be reclaimed by the cluster. Dynamic provisioning utilizes the provider's API to delete the volume from the provider and then removes the persistent volume from the cluster.

15.11. VOLUME SECURITY

15.11.1. Overview

This topic provides a general guide on pod security as it relates to volume security. For information on pod-level security in general, see Managing Security Context Constraints (SCC) and the Security Context Constraint concept topic. For information on the OpenShift persistent volume (PV) framework in general, see the Persistent Storage concept topic.

Accessing persistent storage requires coordination between the cluster and/or storage administrator and the end developer. The cluster administrator creates PVs, which abstract the underlying physical storage. The developer creates pods and, optionally, PVCs, which bind to PVs, based on matching criteria, such as capacity.

Multiple persistent volume claims (PVCs) within the same project can bind to the same PV. However, once a PVC binds to a PV, that PV cannot be bound by a claim outside of the first claim's project. If the underlying storage needs to be accessed by multiple projects, then each project needs its own PV, which can point to the same physical storage. In this sense, a bound PV is tied to a project. For a detailed PV and PVC example, see the guide for WordPress and MySQL using NFS.

For the cluster administrator, granting pods access to PVs involves:

- knowing the group ID and/or user ID assigned to the actual storage,
- understanding SELinux considerations, and
• ensuring that these IDs are allowed in the range of legal IDs defined for the project and/or the SCC that matches the requirements of the pod.

Group IDs, the user ID, and SELinux values are defined in the SecurityContext section in a pod definition. Group IDs are global to the pod and apply to all containers defined in the pod. User IDs can also be global, or specific to each container. Four sections control access to volumes:

• supplementalGroups
• fsGroup
• runAsUser
• seLinuxOptions

15.11.2. SCCs, Defaults, and Allowed Ranges

SCCs influence whether or not a pod is given a default user ID, fsGroup ID, supplemental group ID, and SELinux label. They also influence whether or not IDs supplied in the pod definition (or in the image) will be validated against a range of allowable IDs. If validation is required and fails, then the pod will also fail.

SCCs define strategies, such as runAsUser, supplementalGroups, and fsGroup. These strategies help decide whether the pod is authorized. Strategy values set to RunAsAny are essentially stating that the pod can do what it wants regarding that strategy. Authorization is skipped for that strategy and no OpenShift default is produced based on that strategy. Therefore, IDs and SELinux labels in the resulting container are based on container defaults instead of OpenShift policies.

For a quick summary of RunAsAny:

• Any ID defined in the pod definition (or image) is allowed.
• Absence of an ID in the pod definition (and in the image) results in the container assigning an ID, which is root (0) for Docker.
• No SELinux labels are defined, so Docker will assign a unique label.

For these reasons, SCCs with RunAsAny for ID-related strategies should be protected so that ordinary developers do not have access to the SCC. On the other hand, SCC strategies set to MustRunAs or MustRunAsRange trigger ID validation (for ID-related strategies), and cause default values to be supplied by OpenShift to the container when those values are not supplied directly in the pod definition or image.

SCCs may define the range of allowed IDs (user or groups). If range checking is required (for example, using MustRunAs) and the allowable range is not defined in the SCC, then the project determines the ID range. Therefore, projects support ranges of allowable ID. However, unlike SCCs, projects do not define strategies, such as runAsUser.

Allowable ranges are helpful not only because they define the boundaries for container IDs, but also because the minimum value in the range becomes the default value for the ID in question. For example, if the SCC ID strategy value is MustRunAs, the minimum value of an ID range is 100, and the ID is absent from the pod definition, then 100 is provided as the default for this ID.

As part of pod admission, the SCCs available to a pod are examined (roughly, in priority order followed by most restrictive) to best match the requests of the pod. Setting a SCC’s strategy type to RunAsAny is less restrictive, whereas a type of MustRunAs is more restrictive. All of these strategies are evaluated. To see which SCC was assigned to a pod, use the oc get pod command:
Name of the SCC that the pod used (in this case, a custom SCC).

Name of the pod.

Name of the project. "Namespace" is interchangeable with "project" in OpenShift. See Projects and Users for details.

It may not be immediately obvious which SCC was matched by a pod, so the command above can be very useful in understanding the UID, supplemental groups, and SELinux relabeling in a live container.

Any SCC with a strategy set to RunAsAny allows specific values for that strategy to be defined in the pod definition (and/or image). When this applies to the user ID (runAsUser) it is prudent to restrict access to the SCC to prevent a container from being able to run as root.

Because pods often match the restricted SCC, it is worth knowing the security this entails. The restricted SCC has the following characteristics:

- User IDs are constrained due to the runAsUser strategy being set to MustRunAsRange. This forces user ID validation.
- Because a range of allowable user IDs is not defined in the SCC (see oc export scc restricted for more details), the project's openshift.io/sa.scc.uid-range range will be used for range checking and for a default ID, if needed.
- A default user ID is produced when a user ID is not specified in the pod definition due to runAsUser being set to MustRunAsRange.
- An SELinux label is required (seLinuxContext set to MustRunAs), which uses the project's default MCS label.
- Arbitrary supplemental group IDs are allowed because no range checking is required. This is a result of both the supplementalGroups and fsGroup strategies being set to RunAsAny.
- Default supplemental groups are not produced for the running pod due to RunAsAny for the two group strategies above. Therefore, if no groups are defined in the pod definition (or in the image), the container(s) will have no supplemental groups predefined.

The following shows the default project and a custom SCC (my-custom-scc), which summarizes the interactions of the SCC and the project:
$ oc get scc my-custom-scc -o yaml
...
fsGroup:
  type: MustRunAs
  ranges:
  - min: 5000
    max: 6000
runAsUser:
  type: MustRunAsRange
  uidRangeMin: 65534
  uidRangeMax: 65634
seLinuxContext:
  type: MustRunAs
  SELinuxOptions:
    user: <selinux-user-name>
    role: ...
    type: ...
    level: ...
supplementalGroups:
  type: MustRunAs
  ranges:
  - min: 5000
    max: 6000

**default** is the name of the project.

Default values are only produced when the corresponding SCC strategy is **not** **RunAsAny**.

SELinux default when not defined in the pod definition or in the SCC.

Range of allowable group IDs. ID validation only occurs when the SCC strategy is **RunAsAny**. There can be more than one range specified, separated by commas. See below for supported formats.

Same as <4> but for user IDs. Also, only a single range of user IDs is supported.

**MustRunAs** enforces group ID range checking and provides the container’s groups default. Based on this SCC definition, the default is 5000 (the minimum ID value). If the range was omitted from the SCC, then the default would be 1000000000 (derived from the project). The other supported type, **RunAsAny**, does not perform range checking, thus allowing any group ID, and produces no default groups.

**MustRunAsRange** enforces user ID range checking and provides a UID default. Based on this SCC, the default UID is 65534 (the minimum value). If the minimum *and maximum range were omitted from the SCC, the default user ID would be *1000000000 (derived from the project). *MustRunAsNonRoot and **RunAsAny** are *the other supported types. The range of allowed IDs can be defined to include *any user IDs required for the target storage.

When set to **MustRunAs**, the container is created with the SCC’s SELinux options, or the MCS default defined in the project. A type of **RunAsAny** indicates that SELinux context is not required, and, if not defined in the pod, is not set in the container.
The SELinux user name, role name, type, and labels can be defined here.

Two formats are supported for allowed ranges:

1. **M/N**, where **M** is the starting ID and **N** is the count, so the range becomes **M** through (and including) **M+N-1**.

2. **M-N**, where **M** is again the starting ID and **N** is the ending ID. The default group ID is the starting ID in the first range, which is **1000000000** in this project. If the SCC did not define a minimum group ID, then the project’s default ID is applied.

### 15.11.3. Supplemental Groups

**NOTE**

Read SCCs, Defaults, and Allowed Ranges before working with supplemental groups.

**TIP**

It is generally preferable to use group IDs (supplemental or fsGroup) to gain access to persistent storage versus using user IDs.

Supplemental groups are regular Linux groups. When a process runs in Linux, it has a UID, a GID, and one or more supplemental groups. These attributes can be set for a container’s main process. The **supplementalGroups** IDs are typically used for controlling access to shared storage, such as NFS and GlusterFS, whereas fsGroup is used for controlling access to block storage, such as Ceph RBD and iSCSI.

The OpenShift shared storage plug-ins mount volumes such that the POSIX permissions on the mount match the permissions on the target storage. For example, if the target storage’s owner ID is **1234** and its group ID is **5678**, then the mount on the host node and in the container will have those same IDs. Therefore, the container’s main process must match one or both of those IDs in order to access the volume.

For example, consider the following NFS export.

On an OpenShift node:

**NOTE**

**showmount** requires access to the ports used by **rpcbind** and **rpc.mount** on the NFS server

```bash
# showmount -e <nfs-server-ip-or-hostname>
Export list for f21-nfs.vm:
/opt/nfs  *
```

On the NFS server:

```bash
# cat /etc/exports
/opt/nfs *(rw,sync,no_root_squash)
...```
# ls -lZ /opt/nfs -d
drwxrws---. nfsnobody 5555 unconfined_u:object_r:usr_t:s0 /opt/nfs

# id nfsnobody
uid=65534(nfsnobody) gid=65534(nfsnobody) groups=65534(nfsnobody)

NOTE

In the above, the owner is 65534 (nfsnobody), but the suggestions and examples in this topic apply to any non-root owner.

The /opt/nfs/ export is accessible by UID 65534 and the group 5555. In general, containers should not run as root, so in this NFS example, containers which are not run as UID 65534 or are not members the group 5555 will not be able to access the NFS export.

Often, the SCC matching the pod does not allow a specific user ID to be specified, thus using supplemental groups is a more flexible way to grant storage access to a pod. For example, to grant NFS access to the export above, the group 5555 can be defined in the pod definition:

```yaml
apiVersion: v1
kind: Pod
...
spec:
  containers:
    - name: ...
      volumeMounts:
        - name: nfs
          mountPath: /usr/share/...
      securityContext:
        supplementalGroups: [5555]
  volumes:
    - name: nfs
      nfs:
        server: <nfs_server_ip_or_host>
        path: /opt/nfs
```

1. Name of the volume mount. Must match the name in the volumes section.
2. NFS export path as seen in the container.
3. Pod global security context. Applies to all containers in the pod. Each container can also define its securityContext, however group IDs are global to the pod and cannot be defined for individual containers.
4. Supplemental groups, which is an array of IDs, is set to 5555. This grants group access to the export.
5. Name of the volume. Must match the name in the volumeMounts section.
6. Actual NFS export path on the NFS server.

All containers in the above pod (assuming the matching SCC or project allows the group 5555) will be
members of the group 5555 and have access to the volume, regardless of the container's user ID. However, the assumption above is critical. Sometimes, the SCC does not define a range of allowable group IDs but requires group ID validation (due to supplementalGroups set to MustRunAs; note this is not the case for the restricted SCC). The project will not likely allow a group ID of 5555, unless the project has been customized for access to this NFS export. So in this scenario, the above pod will fail because its group ID of 5555 is not within the SCC's or the project's range of allowed group IDs.

**Supplemental Groups and Custom SCCs**

To remedy the situation in the previous example, a custom SCC can be created such that:

- a minimum and max group ID are defined,
- ID range checking is enforced, and
- the group ID of 5555 is allowed.

It is better to create new SCCs versus modifying a predefined SCC, or changing the range of allowed IDs in the predefined projects.

The easiest way to create a new SCC is to export an existing SCC and customize the YAML file to meet the requirements of the new SCC. For example:

1. Use the restricted SCC as a template for the new SCC:
   ```
   $ oc export scc restricted > new-scc.yaml
   ```
2. Edit the `new-scc.yaml` file to your desired specifications.
3. Create the new SCC:
   ```
   $ oc create -f new-scc.yaml
   ```

   **NOTE**

   The `oc edit scc` command can be used to modify an instantiated SCC.

Here is a fragment of a new SCC named `nfs-scc`:

```
$ oc export scc nfs-scc
allowHostDirVolumePlugin: false
...
kind: SecurityContextConstraints
metadata:
...
  name: nfs-scc
priority: 9
...
supplementalGroups:
  type: MustRunAs
ranges:
```
The allow* bools are the same as for the restricted SCC.

Name of the new SCC.

Numerically larger numbers have greater priority. Nil or omitted is the lowest priority. Higher priority
SCCs sort before lower priority SCCs and thus have a better chance of matching a new pod.

supplementalGroups is a strategy and it is set to MustRunAs, which means group ID checking
is required.

Multiple ranges are supported. The allowed group ID range here is 5000 through 5999, with the
default supplemental group being 5000.

When the same pod shown earlier runs against this new SCC (assuming, of course, the pod has access
to the new SCC), it will start because the group 5555, supplied in the pod definition, is now allowed by
the custom SCC.

15.11.4. fsGroup

NOTE
Read SCCs, Defaults, and Allowed Ranges before working with supplemental groups.

TIP
It is generally preferable to use group IDs (supplemental or fsGroup) to gain access to persistent
storage versus using user IDs.

fsGroup defines a pod’s “file system group” ID, which is added to the container’s supplemental groups.
The supplementalGroups ID applies to shared storage, whereas the fsGroup ID is used for block
storage.

Block storage, such as Ceph RBD, iSCSI, and various cloud storage, is typically dedicated to a single
pod which has requested the block storage volume, either directly or using a PVC. Unlike shared
storage, block storage is taken over by a pod, meaning that user and group IDs supplied in the pod
definition (or image) are applied to the actual, physical block device. Typically, block storage is not
shared.

A fsGroup definition is shown below in the following pod definition fragment:

```yaml
kind: Pod
...
spec:
    containers:
    - name: ...
      securityContext:
        fsGroup: 5555
```
As with `supplementalGroups`, `fsGroup` must be defined globally to the pod, not per container.

5555 will become the group ID for the volume’s group permissions and for all new files created in the volume.

As with `supplementalGroups`, all containers in the above pod (assuming the matching SCC or project allows the group 5555) will be members of the group 5555, and will have access to the block volume, regardless of the container’s user ID. If the pod matches the restricted SCC, whose `fsGroup` strategy is `RunAsAny`, then any `fsGroup` ID (including 5555) will be accepted. However, if the SCC has its `fsGroup` strategy set to `MustRunAs`, and 5555 is not in the allowable range of `fsGroup` IDs, then the pod will fail to run.

**fsGroups and Custom SCCs**

To remedy the situation in the previous example, a custom SCC can be created such that:

- a minimum and maximum group ID are defined,
- ID range checking is enforced, and
- the group ID of 5555 is allowed.

It is better to create new SCCs versus modifying a predefined SCC, or changing the range of allowed IDs in the predefined projects.

Consider the following fragment of a new SCC definition:

```yaml
# oc export scc new-scc
...
kind: SecurityContextConstraints
...
fsGroup:
  type: MustRunAs
  ranges:
  - max: 6000
    min: 5000
...
```

1. **MustRunAs** triggers group ID range checking, whereas **RunAsAny** does not require range checking.

2. The range of allowed group IDs is 5000 through, and including, 5999. Multiple ranges are supported. The allowed group ID range here is 5000 through 5999, with the default `fsGroup` being 5000.

3. The minimum value (or the entire range) can be omitted from the SCC, and thus range checking and generating a default value will defer to the project’s `openshift.io/sa.scc.supplemental-groups` range. `fsGroup` and `supplementalGroups` use the same group field in the project; there is not a separate range for `fsGroup`.

When the pod shown above runs against this new SCC (assuming, of course, the pod has access to the new SCC), it will start because the group 5555, supplied in the pod definition, is allowed by the custom SCC. Additionally, the pod will "take over" the block device, so when the block storage is viewed by a process outside of the pod, it will actually have 5555 as its group ID.
Currently the list of volumes which support block ownership (block) management include:

- AWS Elastic Block Store
- OpenStack Cinder
- Ceph RBD
- GCE Persistent Disk
- iSCSI
- emptyDir
- gitRepo

15.11.5. User IDs

NOTE
Read SCCs, Defaults, and Allowed Ranges before working with supplemental groups.

TIP
It is generally preferable to use group IDs (supplemental or fsGroup) to gain access to persistent storage versus using user IDs.

User IDs can be defined in the container image or in the pod definition. In the pod definition, a single user ID can be defined globally to all containers, or specific to individual containers (or both). A user ID is supplied as shown in the pod definition fragment below:

```yaml
spec:
  containers:
    - name: ...
      securityContext:
        runAsUser: 65534
```

ID 65534 in the above is container-specific and matches the owner ID on the export. If the NFS export’s owner ID was 54321, then that number would be used in the pod definition. Specifying `securityContext` outside of the container definition makes the ID global to all containers in the pod.

Similar to group IDs, user IDs may be validated according to policies set in the SCC and/or project. If the SCC’s `runAsUser` strategy is set to `RunAsAny`, then any user ID defined in the pod definition or in the image is allowed.

WARNING
This means even a UID of 0 (root) is allowed.
If, instead, the `runAsUser` strategy is set to `MustRunAsRange`, then a supplied user ID will be validated against a range of allowed IDs. If the pod supplies no user ID, then the default ID is the minimum value of the range of allowable user IDs.

Returning to the earlier NFS example, the container needs its UID set to 65534, which is shown in the pod fragment above. Assuming the default project and the restricted SCC, the pod’s requested user ID of 65534 will not be allowed, and therefore the pod will fail. The pod fails because:

- it requests 65534 as its user ID,

- all available SCCs use `MustRunAsRange` for their `runAsUser` strategy, so UID range checking is required, and

- 65534 is not included in the SCC or project’s user ID range.

To address this situation, the recommended path would be to create a new SCC with the appropriate user ID range. A new project could also be created with the appropriate user ID range defined. There are other, less-preferred options:

- The restricted SCC could be modified to include 65534 within its minimum and maximum user ID range. This is not recommended as you should avoid modifying the predefined SCCs if possible.

- The restricted SCC could be modified to use RunAsAny for the `runAsUser` value, thus eliminating ID range checking. This is strongly not recommended, as containers could run as root.

- The default project’s UID range could be changed to allow a user ID of 65534. This is not generally advisable because only a single range of user IDs can be specified.

**User IDs and Custom SCCs**

It is good practice to avoid modifying the predefined SCCs if possible. The preferred approach is to create a custom SCC that better fits an organization’s security needs, or create a new project that supports the desired user IDs.

To remedy the situation in the previous example, a custom SCC can be created such that:

- a minimum and maximum user ID is defined,

- UID range checking is still enforced, and

- the UID of 65534 will be allowed.

For example:

```bash
$ oc export scc nfs-scc
allowHostDirVolumePlugin: false ①
...
kind: SecurityContextConstraints
metadata:
...
  name: nfs-scc ②
priority: 9 ③
requiredDropCapabilities: null
```

---

CHAPTER 15. CONFIGURING PERSISTENT STORAGE
The `runAsUser` strategy is set to **MustRunAsRange**, which means UID range checking is enforced. The UID range is 65534 through 65534 (a range of one value).

Now, with `runAsUser: 65534` shown in the previous pod definition fragment, the pod matches the new `nfs-scc` and is able to run with a UID of 65534.

### 15.11.6. SELinux Options

All predefined SCCs, except for the `privileged` SCC, set the `seLinuxContext` to `MustRunAs`. So the SCCs most likely to match a pod’s requirements will force the pod to use an SELinux policy. The SELinux policy used by the pod can be defined in the pod itself, in the image, in the SCC, or in the project (which provides the default).

SELinux labels can be defined in a pod’s `securityContext.seLinuxOptions` section, and supports `user`, `role`, `type`, and `level`:

```
securityContext:
  seLinuxOptions:
    level: "s0:c123,c456"
```

**NOTE**

Level and MCS label are used interchangeably in this topic.

```
securityContext:
  seLinuxOptions:
    level: "s0:c123,c456"
```

**1** `level` can be defined globally for the entire pod, or individually for each container.

**2** SELinux level label.

Here are fragments from an SCC and from the `default` project:

```
$ oc export scc scc-name
...
seLinuxContext:
  type: MustRunAs
```

**1** SELinux context type.
MustRunAs causes volume relabeling.

If the label is not provided in the pod or in the SCC, then the default comes from the project.

All predefined SCCs, except for the privileged SCC, set the seLinuxContext to MustRunAs. This forces pods to use MCS labels, which can be defined in the pod definition, the image, or provided as a default.

The SCC determines whether or not to require an SELinux label and can provide a default label. If the seLinuxContext strategy is set to MustRunAs and the pod (or image) does not define a label, OpenShift defaults to a label chosen from the SCC itself or from the project.

If seLinuxContext is set to RunAsAny, then no default labels are provided, and the container determines the final label. In the case of Docker, the container will use a unique MCS label, which will not likely match the labeling on existing storage mounts. Volumes which support SE Linux management will be relabeled so that they are accessible by the specified label and, depending on how exclusionary the label is, only that label.

This means two things for unprivileged containers:

- The volume will be given a type which is accessible by unprivileged containers. This type is usually svirt_sandbox_file_t.
- If a level is specified, the volume will be labeled with the given MCS label.

For a volume to be accessible by a pod, the pod must have both categories of the volume. So a pod with s0:c1,c2 will be able to access a volume with s0:c1,c2. A volume with s0 will be accessible by all pods.

If pods fail authorization, or if the storage mount is failing due to permissions errors, then there is a possibility that SE Linux enforcement is interfering. One way to check for this is to run:

```
# ausearch -m avc --start recent
```

This examines the log file for AVC (Access Vector Cache) errors.
CHAPTER 16. PERSISTENT STORAGE EXAMPLES

16.1. OVERVIEW

The following sections provide detailed, comprehensive instructions on setting up and configuring common storage use cases. These examples cover both the administration of persistent volumes and their security, and how to claim against the volumes as a user of the system.

- Sharing an NFS PV Across Two Pods
- Ceph-RBD Block Storage Volume
- Shared Storage Using a GlusterFS Volume
- Backing Docker Registry with GlusterFS Storage
- Mounting a PV to Privileged Pods

16.2. SHARING AN NFS PERSISTENT VOLUME (PV) ACROSS TWO PODS

16.2.1. Overview

The following use case describes how a cluster administrator wanting to leverage shared storage for use by two separate containers would configure the solution. This example highlights the use of NFS, but can easily be adapted to other shared storage types, such as GlusterFS. In addition, this example will show configuration of pod security as it relates to shared storage.

Persistent Storage Using NFS provides an explanation of persistent volumes (PVs), persistent volume claims (PVCs), and using NFS as persistent storage. This topic shows and end-to-end example of using an existing NFS cluster and OpenShift Enterprise persistent store, and assumes an existing NFS server and exports exist in your OpenShift Enterprise infrastructure.

**NOTE**

All `oc` commands are executed on the OpenShift Enterprise master host.

16.2.2. Creating the Persistent Volume

Before creating the PV object in OpenShift Enterprise, the persistent volume (PV) file is defined:

*Example 16.1. Persistent Volume Object Definition Using NFS*

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfs-pv
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteMany
```
The name of the PV, which is referenced in pod definitions or displayed in various `oc` volume commands.

The amount of storage allocated to this volume.

`accessModes` are used as labels to match a PV and a PVC. They currently do not define any form of access control.

A volume reclaim policy of `retain` indicates to preserve the volume after the pods.

This defines the volume type being used, in this case the `NFS` plug-in.

This is the NFS mount path.

This is the NFS server. This can also be specified by IP address.

Save the PV definition to a file, for example `nfs-pv.yaml`, and create the persistent volume:

```
# oc create -f nfs-pv.yaml
persistentvolume "nfs-pv" created
```

Verify that the persistent volume was created:

```
# oc get pv
NAME         LABELS    CAPACITY   ACCESSMODES   STATUS      CLAIM
REASON   AGE
nfs-pv       <none>    1Gi        RWX           Available
37s
```

### 16.2.3. Creating the Persistent Volume Claim

A persistent volume claim (PVC) specifies the desired access mode and storage capacity. Currently, based on only these two attributes, a PVC is bound to a single PV. Once a PV is bound to a PVC, that PV is essentially tied to the PVC’s project and cannot be bound to by another PVC. There is a one-to-one mapping of PVs and PVCs. However, multiple pods in the same project can use the same PVC. This is the use case we are highlighting in this example.

#### Example 16.2. PVC Object Definition

```yaml
apiVersion: v1
classKind: PersistentVolumeClaim
metadata:
  name: nfs-pvc
spec:
  accessModes:
```

The claim name is referenced by the pod under its `volumes` section.

As mentioned above for PVs, the `accessModes` do not enforce access right, but rather act as labels to match a PV to a PVC.

This claim will look for PVs offering 1Gi or greater capacity.

Save the PVC definition to a file, for example `nfs-pvc.yaml`, and create the PVC:

```bash
# oc create -f nfs-pvc.yaml
persistentvolumeclaim "nfs-pvc" created
```

Verify that the PVC was created and bound to the expected PV:

```bash
# oc get pvc
NAME            LABELS    STATUS    VOLUME       CAPACITY   ACCESSMODES
AGE
nfs-pvc         <none>    Bound     nfs-pv       1Gi        RWX
24s
```

The claim, `nfs-pvc`, was bound to the `nfs-pv` PV.

### 16.2.4. Ensuring NFS Volume Access

Access is necessary to a node in the NFS server. On this node, examine the NFS export mount:

```
[root@nfs nfs]# ls -lZ /opt/nfs/
total 8
-rw-r--r--. 1 root 100003  system_u:object_r:usr_t:s0     10 Oct 12 23:27
  test2b
```

1. the owner has ID 0.
2. the group has ID 100003.

In order to access the NFS mount, the container must match the SELinux label, and either run with a UID of 0, or with 100003 in its supplemental groups range. Gain access to the volume by matching the NFS mount’s groups, which will be defined in the pod definition below.

By default, SELinux does not allow writing from a pod to a remote NFS server. To enable writing to NFS volumes with SELinux enforcing on each node, run:

```yaml
- ReadWriteMany
  resources:
    requests:
      storage: 1Gi
```

234
# setsebool -P virt_sandbox_use_nfs on
# setsebool -P virt_use_nfs on

NOTE

The `virt_sandbox_use_nfs` boolean is defined by the `docker-selinux` package. If you get an error saying it is not defined, ensure that this package is installed.

16.2.5. Creating the Pod

A pod definition file or a template file can be used to define a pod. Below is a pod specification that creates a single container and mounts the NFS volume for read-write access:

Example 16.3. Pod Object Definition

```yaml
apiVersion: v1
class: Pod
description: Pod

metadata:
  name: nginx-nfs-pod
  labels:
    name: nginx-nfs-pod

spec:
  containers:
    - name: nginx-nfs-pod
      image: fedora/nginx
      ports:
        - name: web
          containerPort: 80
      volumeMounts:
        - name: nfsvol
          mountPath: /usr/share/nginx/html
      securityContext:
        supplementalGroups: [100003]
        privileged: false
      volumes:
        - name: nfsvol
          persistentVolumeClaim:
            claimName: nfs-pvc
```

1. The name of this pod as displayed by `oc get pod`.
2. The image run by this pod.
3. The name of the volume. This name must be the same in both the `containers` and `volumes` sections.
4. The mount path as seen in the container.
5. The group ID to be assigned to the container.
6. The PVC that was created in the previous step.
Save the pod definition to a file, for example `nfs.yaml`, and create the pod:

```bash
# oc create -f nfs.yaml
pod "nginx-nfs-pod" created
```

Verify that the pod was created:

```bash
# oc get pods
NAME                READY     STATUS    RESTARTS   AGE
nginx-nfs-pod       1/1       Running   0          4s
```

More details are shown in the `oc describe pod` command:

```
[root@ose70 nfs]# oc describe pod nginx-nfs-pod
Name:    nginx-nfs-pod
Namespace:   default
Image(s):   fedora/nginx
Node:    ose70.rh7/192.168.234.148
Start Time:   Mon, 21 Mar 2016 09:59:47 -0400
Labels:    name=nginx-nfs-pod
Status:    Running
Reason:
Message:
IP:    10.1.0.4
Replication Controllers: <none>
Containers:
    nginx-nfs-pod:
        Container ID: docker://a3292104d6c28d9cf49f440b2967a0fc5583540fc3b062db598557b93893bc6f
        Image:  fedora/nginx
        Image ID:
        QoS Tier:
            cpu:  BestEffort
            memory:  BestEffort
        State:  Running
            Started:  Mon, 21 Mar 2016 09:59:49 -0400
        Ready:  True
        Restart Count: 0
        Environment Variables:
Conditions:
    Type  Status
    Ready  True
Volumes:
    nfsvol:
        Type: PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same namespace)
            ClaimName: nfs-pvc
            ReadOnly: false
default-token-a06zb:
        Type: Secret (a secret that should populate this volume)
            SecretName: default-token-a06zb
Events:
    FirstSeen  LastSeen  Count  From   SubobjectPath   Reason   Message
```

236
The project (namespace) name.

The IP address of the OpenShift Enterprise node running the pod.

The PVC name used by the pod.

The list of events resulting in the pod being launched and the NFS volume being mounted. The container will not start correctly if the volume cannot mount.

There is more internal information, including the SCC used to authorize the pod, the pod’s user and group IDs, the SELinux label, and more, shown in the `oc get pod <name> -o yaml` command:

```
[root@ose70 nfs]# oc get pod nginx-nfs-pod -o yaml
apiVersion: v1
kind: Pod
metadata:
  annotations:
    openshift.io/scc: restricted
  creationTimestamp: 2016-03-21T13:59:47Z
  labels:
    name: nginx-nfs-pod
  name: nginx-nfs-pod
  namespace: default
  resourceVersion: "2814411"
  selflink: /api/v1/namespaces/default/pods/nginx-nfs-pod
  uid: 2c22d2ea-ef6d-11e5-adc7-000c2900f1e3
spec:
  containers:
  - image: fedora/nginx
    imagePullPolicy: IfNotPresent
    name: nginx-nfs-pod
    ports:
      - containerPort: 80
        name: web
        protocol: TCP
    resources: {}
    securityContext:
      privileged: false
    terminationMessagePath: /dev/termination-log
```
The SCC used by the pod.

The project (namespace) name.

The supplemental group ID for the pod (all containers).

The PVC name used by the pod.
16.2.6. Creating an Additional Pod to Reference the Same PVC

This pod definition, created in the same namespace, uses a different container. However, we can use the same backing storage by specifying the claim name in the volumes section below:

Example 16.4. Pod Object Definition

```
apiVersion: v1
kind: Pod
metadata:
  name: busybox-nfs-pod
labels:
  name: busybox-nfs-pod
spec:
  containers:
    - name: busybox-nfs-pod
      image: busybox
      command: ["sleep", "60000"]
      volumeMounts:
        - name: nfsvol-2
          mountPath: /usr/share/busybox
          readOnly: false
      securityContext:
        supplementalGroups: [100003]
        privileged: false
  volumes:
    - name: nfsvol-2
      persistentVolumeClaim:
       claimName: nfs-pvc
```

1. The name of this pod as displayed by `oc get pod`.
2. The image run by this pod.
3. The name of the volume. This name must be the same in both the `containers` and `volumes` sections.
4. The mount path as seen in the container.
5. The group ID to be assigned to the container.
6. The PVC that was created earlier and is also being used by a different container.

Save the pod definition to a file, for example `nfs-2.yaml`, and create the pod:

```
# oc create -f nfs-2.yaml
pod "busybox-nfs-pod" created
```

Verify that the pod was created:
# oc get pods

<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>busybox-nfs-pod</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>3s</td>
</tr>
</tbody>
</table>

More details are shown in the `oc describe pod` command:

```
[root@ose70 nfs]# oc describe pod busybox-nfs-pod
Name:    busybox-nfs-pod
Namespace:   default
Image(s):   busybox
Node:    ose70.rh7/192.168.234.148
Start Time:   Mon, 21 Mar 2016 10:19:46 -0400
Labels:    name=busybox-nfs-pod
Status:    Running
Reason:  
Message:  
IP:    10.1.0.5
Replication Controllers: <none>
Containers:
    busybox-nfs-pod:
        Container ID:
        docker://346d432e5a4824ebf5a47fceb4247e0568ecc64eadcc160e9bab481ae0b0594
            Image:  busybox
            Image ID:
                docker://17583c7dd0da6244203b8029733b7d717fcccbb2b5d93e2b24cf48b8bdf06e2
            QoS Tier:
                cpu:  BestEffort
                memory:  BestEffort
            State:  Running
                Started:  Mon, 21 Mar 2016 10:19:48 -0400
            Ready:  True
            Restart Count: 0
            Environment Variables:
Conditions:  
    Type  Status
        Ready  True
Volumes:
    nfsvol-2:
        Type: PersistentVolumeClaim (a reference to a PersistentVolumeClaim in
        the same namespace)
            ClaimName: nfs-pvc
            ReadOnly: false
            default-token-32d2z:  
                Type: Secret (a secret that should populate this volume)
                    SecretName: default-token-32d2z
Events:  
    FirstSeen  LastSeen  Count  From  SubobjectPath  Reason  Message
        4m   4m        1  {scheduler }  Scheduled  Successfully assigned busybox-nfs-pod to ose70.rh7
        4m   4m        1  {kubelet ose70.rh7}  implicitly required container POD Pulled
            Container image "openshift3/ose-pod:v3.1.0.4" already present on machine
        4m   4m        1  {kubelet ose70.rh7}  implicitly required container POD Created
            Created with docker id 249b7d7519b1
        4m   4m        1  {kubelet ose70.rh7}  implicitly required container POD Started
```
As you can see, both containers are using the same storage claim that is attached to the same NFS mount on the back end.

### 16.3. COMPLETE EXAMPLE USING CEPH RBD

#### 16.3.1. Overview

This topic provides an end-to-end example of using an existing Ceph cluster as an OpenShift Enterprise persistent store. It is assumed that a working Ceph cluster is already set up. If not, consult the [Overview of Red Hat Ceph Storage](Overview of Red Hat Ceph Storage).

Persistent Storage Using Ceph Rados Block Device provides an explanation of persistent volumes (PVs), persistent volume claims (PVCs), and using Ceph RBD as persistent storage.

**NOTE**

All `oc` ... commands are executed on the OpenShift Enterprise master host.

#### 16.3.2. Installing the ceph-common Package

The `ceph-common` library must be installed on **all schedulable** OpenShift Enterprise nodes:

```
# yum install -y ceph-common
```

#### 16.3.3. Creating the Ceph Secret

The `ceph auth get-key` command is run on a Ceph **MON** node to display the key value for the `client.admin` user:

**Example 16.5. Ceph Secret Definition**

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: ceph-secret
data:
  key: QVF8OFF2S1ZheUJQRVJBPwVr2cwT1laQUhPQno3akZwekxxdGc9PQ==
```

---

CHAPTER 16. PERSISTENT STORAGE EXAMPLES

241
This base64 key is generated on one of the Ceph MON nodes using the `ceph auth get-key client.admin | base64` command, then copying the output and pasting it as the secret key’s value.

Save the secret definition to a file, for example `ceph-secret.yaml`, then create the secret:

```
$ oc create -f ceph-secret.yaml
secret "ceph-secret" created
```

Verify that the secret was created:

```
# oc get secret ceph-secret
NAME          TYPE      DATA      AGE
ceph-secret   Opaque    1         23d
```

16.3.4. Creating the Persistent Volume

Next, before creating the PV object in OpenShift Enterprise, define the persistent volume file:

```
Example 16.6. Persistent Volume Object Definition Using Ceph RBD
```  
```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: ceph-pv
spec:
  capacity:
    storage: 2Gi
  accessModes:
    - ReadWriteOnce
  rbd:
    monitors:
      - 192.168.122.133:6789
    pool: rbd
    image: ceph-image
    user: admin
    secretRef:
      name: ceph-secret
    fsType: ext4
    readOnly: false
    persistentVolumeReclaimPolicy: Recycle
```

1. The name of the PV, which is referenced in pod definitions or displayed in various `oc` volume commands.

2. The amount of storage allocated to this volume.

3. `accessModes` are used as labels to match a PV and a PVC. They currently do not define any form of access control. All block storage is defined to be single user (non-shared storage).

4. This defines the volume type being used. In this case, the `rbd` plug-in is defined.
This is an array of Ceph monitor IP addresses and ports.

This is the Ceph secret, defined above. It is used to create a secure connection from OpenShift Enterprise to the Ceph server.

This is the file system type mounted on the Ceph RBD block device.

Save the PV definition to a file, for example `ceph-pv.yaml`, and create the persistent volume:

```
# oc create -f ceph-pv.yaml
persistentvolume "ceph-pv" created
```

Verify that the persistent volume was created:

```
# oc get pv
NAME   LABELS    CAPACITY     ACCESSMODES   STATUS
CLAIM  REASON    AGE
ceph-pv <none>    2147483648   RWO           Available
2s
```

16.3.5. Creating the Persistent Volume Claim

A persistent volume claim (PVC) specifies the desired access mode and storage capacity. Currently, based on only these two attributes, a PVC is bound to a single PV. Once a PV is bound to a PVC, that PV is essentially tied to the PVC’s project and cannot be bound to by another PVC. There is a one-to-one mapping of PVs and PVCs. However, multiple pods in the same project can use the same PVC.

Example 16.7. PVC Object Definition

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: ceph-claim
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
```

1 As mentioned above for PVs, the `accessModes` do not enforce access right, but rather act as labels to match a PV to a PVC.

2 This claim will look for PVs offering 2Gi or greater capacity.

Save the PVC definition to a file, for example `ceph-claim.yaml`, and create the PVC:

```
# oc create -f ceph-claim.yaml
persistentvolumeclaim "ceph-claim" created
```
and verify the PVC was created and bound to the expected PV:

```
# oc get pvc
NAME     LABELS    STATUS    VOLUME     CAPACITY  ACCESSMODES  AGE
ceph-claim <none>    Bound     ceph-pv    1Gi        RWX         21s
```

1 the claim was bound to the *ceph-pv* PV.

### 16.3.6. Creating the Pod

A pod definition file or a template file can be used to define a pod. Below is a pod specification that creates a single container and mounts the Ceph RBD volume for read-write access:

**Example 16.8. Pod Object Definition**

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: ceph-pod1
spec:
  containers:
  - name: ceph-busybox
    image: busybox
    command: ["sleep", "60000"
    volumeMounts:
    - name: ceph-vol1
      mountPath: /usr/share/busybox
      readOnly: false
  volumes:
  - name: ceph-vol1
    persistentVolumeClaim:
      claimName: ceph-claim
```

1 The name of this pod as displayed by `oc get pod`.

2 The image run by this pod. In this case, we are telling `busybox` to sleep.

3 5 The name of the volume. This name must be the same in both the `containers` and `volumes` sections.

4 The mount path as seen in the container.

6 The PVC that is bound to the Ceph RBD cluster.

Save the pod definition to a file, for example `ceph-pod1.yaml`, and create the pod:

```
# oc create -f ceph-pod1.yaml
pod "ceph-pod1" created
```

#verify pod was created
# oc get pod
NAME        READY     STATUS    RESTARTS   AGE
ceph-pod1   1/1       Running   0          2m  

After a minute or so, the pod will be in the **Running** state.

## 16.3.7. Defining Group and Owner IDs (Optional)

When using block storage, such as Ceph RBD, the physical block storage is **managed** by the pod. The group ID defined in the pod becomes the group ID of **both** the Ceph RBD mount inside the container, and the group ID of the actual storage itself. Thus, it is usually unnecessary to define a group ID in the pod specification. However, if a group ID is desired, it can be defined using `fsGroup`, as shown in the following pod definition fragment:

**Example 16.9. Group ID Pod Definition**

```yaml
...                      
spec:                      
  containers:              
    - name:                
      ...                  
    securityContext:       
      fsGroup: 7777         
...                      
```

**securityContext** must be defined at the pod level, not under a specific container.

**All containers in the pod will have the same **fsGroup** ID.**

## 16.4. COMPLETE EXAMPLE USING GLUSTERFS

### 16.4.1. Overview

This topic provides an end-to-end example of how to use an existing Gluster cluster as an OpenShift Enterprise persistent store. It is assumed that a working Gluster cluster is already set up. If not, consult the **Red Hat Gluster Storage Administration Guide**.

**Persistent Storage Using GlusterFS** provides an explanation of persistent volumes (PVs), persistent volume claims (PVCs), and using GlusterFS as persistent storage.

**NOTE**

All `oc` … commands are executed on the OpenShift Enterprise master host.

### 16.4.2. Installing the glusterfs-fuse Package

The `glusterfs-fuse` library must be installed on all **schedulable** OpenShift Enterprise nodes:
# yum install -y glusterfs-fuse

**NOTE**

The OpenShift Enterprise all-in-one host is often not used to run pod workloads and, thus, is not included as a schedulable node.

### 16.4.3. Creating the Gluster Endpoints

The named endpoints define each node in the Gluster-trusted storage pool:

**Example 16.10. GlusterFS Endpoint Definition**

```yaml
apiVersion: v1
kind: Endpoints
metadata:
  name: gluster-endpoints
subsets:
- addresses:
  - ip: 192.168.122.21
    ports:
      - port: 1
        protocol: TCP
  - ip: 192.168.122.22
    ports:
      - port: 1
        protocol: TCP
```

1. The name of the endpoints is used in the PV definition below.
2. An array of IP addresses for each node in the Gluster pool. Currently, host names are not supported.
3. The port numbers are ignored, but must be legal port numbers. The value 1 is commonly used.

Save the endpoints definition to a file, for example `gluster-endpoints.yaml`, then create the endpoints object:

```bash
# oc create -f gluster-endpoints.yaml
endpoints "gluster-endpoints" created
```

Verify that the endpoints were created:

```bash
# oc get endpoints gluster-endpoints
NAME          ENDPOINTS                           AGE
---           -------                           ----
gluster-endpoints   192.168.122.21:1,192.168.122.22:1   1m
```

### 16.4.4. Creating the Persistent Volume
Next, before creating the PV object, define the persistent volume in OpenShift Enterprise:

**Example 16.11. Persistent Volume Object Definition Using GlusterFS**

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: gluster-pv
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteMany
  glusterfs:
    endpoints: gluster-endpoints
    path: /HadoopVol
    readOnly: false
  persistentVolumeReclaimPolicy: Retain
```

1. The name of the PV, which is referenced in pod definitions or displayed in various `oc` volume commands.
2. The amount of storage allocated to this volume.
3. `accessModes` are used as labels to match a PV and a PVC. They currently do not define any form of access control.
4. This defines the volume type being used. In this case, the `glusterfs` plug-in is defined.
5. This references the endpoints named above.
6. This is the Gluster volume name, preceded by `/`.
7. A volume reclaim policy of `retain` indicates that the volume will be preserved after the pods accessing it terminate.

Save the PV definition to a file, for example `gluster-pv.yaml`, and create the persistent volume:

```
# oc create -f gluster-pv.yaml
persistentvolume "gluster-pv" created
```

Verify that the persistent volume was created:

```
# oc get pv
NAME          LABELS   CAPACITY  ACCESSMODES  STATUS    CLAIM
REASON        AGE
gluster-pv     <none>   1Gi      RWX         Available
37s
```

**16.4.5. Creating the Persistent Volume Claim**
A persistent volume claim (PVC) specifies the desired access mode and storage capacity. Currently, based on only these two attributes, a PVC is bound to a single PV. Once a PV is bound to a PVC, that PV is essentially tied to the PVC’s project and cannot be bound to by another PVC. There is a one-to-one mapping of PVs and PVCs. However, multiple pods in the same project can use the same PVC.

**Example 16.12. PVC Object Definition**

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gluster-claim
spec:
  accessModes:
  - ReadWriteMany
  requests:
    storage: 1Gi
```

1. The claim name is referenced by the pod under its `volumes` section.
2. As mentioned above for PVs, the `accessModes` do not enforce access rights, but rather act as labels to match a PV to a PVC.
3. This claim will look for PVs offering `1Gi` or greater capacity.

Save the PVC definition to a file, for example `gluster-claim.yaml`, and create the PVC:

```
# oc create -f gluster-claim.yaml
persistentvolumeclaim "gluster-claim" created
```

Verify the PVC was created and bound to the expected PV:

```
# oc get pvc
NAME            LABELS    STATUS    VOLUME       CAPACITY   ACCESSMODES
AGE
gluster-claim   <none>    Bound     gluster-pv   1Gi        RWX
24s
```

1. The claim was bound to the `gluster-pv` PV.

### 16.4.6. Defining GlusterFS Volume Access

Access is necessary to a node in the Gluster-trusted storage pool. On this node, examine the `glusterfs-fuse` mount:

```
# ls -lZ /mnt/glusterfs/
-rw-rw---.  yarn  hadoop  system_u:object_r:glusterfs_t:s0    HadoopVol

# id yarn
```
uid=592(yarn) gid=590(hadoop) groups=590(hadoop)

1. The owner has ID 592.
2. The group has ID 590.

In order to access the HadoopVol volume, the container must match the SELinux label, and either run with a UID of 592, or with 590 in its supplemental groups. It is recommended to gain access to the volume by matching the Gluster mount’s groups, which is defined in the pod definition below.

By default, SELinux does not allow writing from a pod to a remote Gluster server. To enable writing to GlusterFS volumes with SELinux enforcing on each node, run:

```
# setsebool -P virt_sandbox_use_fusefs on
```

**NOTE**

The `virt_sandbox_use_fusefs` boolean is defined by the `docker-selinux` package. If you get an error saying it is not defined, ensure that this package is installed.

### 16.4.7. Creating the Pod

A pod definition file or a template file can be used to define a pod. Below is a pod specification that creates a single container and mounts the Gluster volume for read-write access:

**Example 16.13. Pod Object Definition**

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: gluster-pod1
  labels:
    name: gluster-pod1
spec:
  containers:
  - name: gluster-pod1
    image: busybox
    command: ['sleep', '60000']
    volumeMounts:
    - name: gluster-vol1
      mountPath: /usr/share/busybox
      readOnly: false
    securityContext:
      supplementalGroups: [590]
      privileged: false
  volumes:
  - name: gluster-vol1
    persistentVolumeClaim:
      claimName: gluster-claim
```
The name of this pod as displayed by `oc get pod`.

The image run by this pod. In this case, we are telling `busybox` to sleep.

The name of the volume. This name must be the same in both the `containers` and `volumes` sections.

The mount path as seen in the container.

The group ID to be assigned to the container.

The PVC that is bound to the Gluster cluster.

Save the pod definition to a file, for example `gluster-pod1.yaml`, and create the pod:

```
# oc create -f gluster-pod1.yaml
pod "gluster-pod1" created
```

Verify the pod was created:

```
# oc get pod
NAME           READY     STATUS    RESTARTS   AGE
---            ------     ------     --------   -----
gluster-pod1   1/1       Running   0          31s
```

After a minute or so, the pod will be in the `Running` state.

More details are shown in the `oc describe pod` command:

```
# oc describe pod gluster-pod1
Name:    gluster-pod1
Namespace:   default
Image(s):   busybox
Node:    rhe17.2-dev/192.168.122.177
Start Time:   Tue, 22 Mar 2016 10:55:57 -0700
Labels:    name=gluster-pod1
Status:    Running
Reason:
Message:
IP:    10.1.0.2
Replication Controllers: <none>
Containers:
  gluster-pod1:
    Container ID: docker://acc0c80c28a5cd64b6e3f2848052ef30a21ee850d27ef5fe959d11da4e5a3f4f
    Image:  busybox
    Image ID: docker://964092b7f3e54185d3f425880be0b022bfc9a706701390e0ceab527c84dea3e3
    QoS Tier:
      cpu:  BestEffort
      memory:  BestEffort
```
The project (namespace) name.

The IP address of the OpenShift Enterprise node running the pod.

The PVC name used by the pod.

The list of events resulting in the pod being launched and the Gluster volume being mounted.

There is more internal information, including the SCC used to authorize the pod, the pod’s user and group IDs, the SELinux label, and more shown in the `oc get pod <name> -o yaml` command:

```bash
# oc get pod gluster-pod1 -o yaml
apiVersion: v1
kind: Pod
metadata:
  annotations:
    openshift.io/scc: restricted
  creationTimestamp: 2016-03-22T17:55:57Z
labels:
```
name: gluster-pod1
name: gluster-pod1
namespace: default
resourceVersion: "511908"
selflink: /api/v1/namespaces/default/pods/gluster-pod1
uid: 545068a3-f057-11e5-a8e5-5254008f071b
spec:
containers:
- command:
  - sleep
  - "60000"
image: busybox
imagePullPolicy: IfNotPresent
name: gluster-pod1
resources: {}
securityContext:
  privileged: false
  runAsUser: 1000000000
  seLinuxOptions:
    level: s0:c1,c0
terminationMessagePath: /dev/termination-log
volumeMounts:
- mountPath: /usr/share/busybox
  name: gluster-vol1
- mountPath: /var/run/secrets/kubernetes.io/serviceaccount
  name: default-token-rbi9o
  readOnly: true
dnsPolicy: ClusterFirst
host: rhel7.2-dev
imagePullSecrets:
- name: default-dockercfg-2g6go
nodeName: rhel7.2-dev
restartPolicy: Always
securityContext:
  seLinuxOptions:
    level: s0:c1,c0
  supplementalGroups:
    - 590
serviceAccount: default
serviceAccountName: default
terminationGracePeriodSeconds: 30
volumes:
- name: gluster-vol1
  persistentVolumeClaim:
    claimName: gluster-claim
- name: default-token-rbi9o
  secret:
    secretName: default-token-rbi9o
status:
conditions:
- lastProbeTime: null
  lastTransitionTime: 2016-03-22T17:56:00Z
  status: "True"
type: Ready
containerStatuses:
- containerID:
The SCC used by the pod.

The project (namespace) name.

The UID of the busybox container.

The SELinux label for the container, and the default SELinux label for the entire pod, which happen to be the same here.

The supplemental group ID for the pod (all containers).

The PVC name used by the pod.

16.5. BACKING DOCKER REGISTRY WITH GLUSTERFS STORAGE

16.5.1. Overview

This topic reviews how to attach a GlusterFS persistent volume to the Docker Registry.

It is assumed that the Docker registry service has already been started and the Gluster volume has been created.

16.5.2. Prerequisites

- The docker-registry was deployed without configuring storage.

- A Gluster volume exists and glusterfs-fuse is installed on schedulable nodes.

- Definitions written for GlusterFS endpoints and service, persistent volume (PV), and persistent volume claim (PVC).
  - For this guide, these will be:
    - `gluster-endpoints-service.yaml`
    - `gluster-endpoints.yaml`
    - `gluster-pv.yaml`
- **gluster-pvc.yaml**
  - A user with the `cluster-admin` role binding.
    - For this guide, that user is `admin`.

**NOTE**

All `oc` commands are executed on the master node as the `admin` user.

### 16.5.3. Create the Gluster Persistent Volume

First, make the Gluster volume available to the registry.

```
$ oc create -f gluster-endpoints-service.yaml
$ oc create -f gluster-endpoints.yaml
$ oc create -f gluster-pv.yaml
$ oc create -f gluster-pvc.yaml
```

Check to make sure the PV and PVC were created and bound successfully. The expected output should resemble the following. Note that the PVC status is **Bound**, indicating that it has bound to the PV.

```
$ oc get pv
NAME         LABELS    CAPACITY   ACCESSMODES   STATUS      CLAIM
REASON    AGE
gluster-pv   <none>    1Gi        RWX           Available
37s
$ oc get pvc
NAME            LABELS    STATUS    VOLUME       CAPACITY   ACCESSMODES
AGE
gluster-claim   <none>    Bound     gluster-pv   1Gi        RWX
24s
```

**NOTE**

If either the PVC or PV failed to create or the PVC failed to bind, refer back to the GlusterFS Persistent Storage guide. Do **not** proceed until they initialize and the PVC status is **Bound**.

### 16.5.4. Attach the PVC to the Docker Registry

Before moving forward, ensure that the `docker-registry` service is running.

```
$ oc get svc
NAME              CLUSTER_IP       EXTERNAL_IP   PORT(S)
SELECTOR                  AGE
docker-registry   172.30.167.194   <none>        5000/TCP
docker-registry=default   18m
```

**NOTE**

If either the `docker-registry` service or its associated pod is not running, refer back to the `docker-registry` setup instructions for troubleshooting before continuing.
Then, attach the PVC:

```
$ oc volume deploymentconfigs/docker-registry --add --name=v1 -t pvc \
   --claim-name=gluster-claim --overwrite
```

Deploying a Docker Registry provides more information on using the Docker registry.

### 16.5.5. Known Issues

#### 16.5.5.1. Pod Cannot Resolve the Volume Host

In non-production cases where the `dnsmasq` server is located on the same node as the OpenShift Enterprise master service, pods might not resolve to the host machines when mounting the volume, causing errors in the `docker-registry-1-deploy` pod. This can happen when `dnsmasq.service` fails to start because of a collision with OpenShift DNS on port 53. To run the DNS server on the master host, some configurations needs to be changed.

In `/etc/dnsmasq.conf`, add:

```
# Reverse DNS record for master
dnsConfig:
  host-record=master.example.com,<master-IP>
# Wildcard DNS for OpenShift Applications - Points to Router
  address=/apps.example.com/<master-IP>
# Forward .local queries to SkyDNS
  server=/local/127.0.0.1#8053
# Forward reverse queries for service network to SkyDNS.
  server=/17.30.172.in-addr.arpa/127.0.0.1#8053
```

With these settings, `dnsmasq` will pull from the `/etc/hosts` file on the master node.

Add the appropriate host names and IPs for all necessary hosts.

In `master-config.yaml`, change `bindAddress` to:

```
dnsConfig:
  bindAddress: 127.0.0.1:8053
```

When pods are created, they receive a copy of `/etc/resolv.conf`, which typically contains only the master DNS server so they can resolve external DNS requests. To enable internal DNS resolution, insert the `dnsmasq` server at the top of the server list. This way, `dnsmasq` will attempt to resolve requests internally first.

In `/etc/resolv.conf` all scheduled nodes:

```
nameserver 192.168.1.100
nameserver 192.168.1.1
```

1. Add the internal DNS server.
2. Pre-existing external DNS server.
Once the configurations are changed, restart the OpenShift Enterprise master and \texttt{dnsmasq} services.

\begin{verbatim}
$ systemctl restart atomic-openshift-master
$ systemctl restart dnsmasq
\end{verbatim}

### 16.6. MOUNTING VOLUMES ON PRIVILEGED PODS

#### 16.6.1. Overview

Persistent volumes can be mounted to pods with the \texttt{privileged} security context constraint (SCC) attached.

\begin{itemize}
  \item \textbf{NOTE}
  \begin{itemize}
    \item While this topic uses GlusterFS as a sample use-case for mounting volumes onto privileged pods, it can be adapted to use any \texttt{supported storage plug-in}.
  \end{itemize}
\end{itemize}

#### 16.6.2. Prerequisites

- An existing Gluster volume.
- \texttt{glusterfs-fuse} installed on all hosts.
- Definitions for GlusterFS:
  - Endpoints and services: \texttt{gluster-endpoints-service.yaml} and \texttt{gluster-endpoints.yaml}
  - Persistent volumes: \texttt{gluster-pv.yaml}
  - Persistent volume claims: \texttt{gluster-pvc.yaml}
  - Privileged pods: \texttt{gluster-nginx-pod.yaml}
- A user with the \texttt{cluster-admin} role binding. For this guide, that user is called \texttt{admin}.

#### 16.6.3. Creating the Persistent Volume

Creating the \texttt{PersistentVolume} makes the storage accessible to users, regardless of projects.

1. As the admin, create the service, endpoint object, and persistent volume:

\begin{verbatim}
$ oc create -f gluster-endpoints-service.yaml
$ oc create -f gluster-endpoints.yaml
$ oc create -f gluster-pv.yaml
\end{verbatim}

2. Verify that the objects were created:

\begin{verbatim}
$ oc get svc
NAME              CLUSTER_IP      EXTERNAL_IP   PORT(S)   SELECTOR
AGE   172.30.151.58   <none>        1/TCP     <none>
gluster-cluster   24s
\end{verbatim}
### 16.6.4. Creating a Regular User

Adding a regular user to the privileged SCC (or to a group given access to the SCC) allows them to run privileged pods:

1. As the admin, add a user to the SCC:

   ```
   $ oadm policy add-scc-to-user privileged <username>
   ```

2. Log in as the regular user:

   ```
   $ oc login -u <username> -p <password>
   ```

3. Then, create a new project:

   ```
   $ oc new-project <project_name>
   ```

### 16.6.5. Creating the Persistent Volume Claim

1. As a regular user, create the PersistentVolumeClaim to access the volume:

   ```
   $ oc create -f gluster-pvc.yaml -n <project_name>
   ```

2. Define your pod to access the claim:

   **Example 16.14. Pod Definition**

   ```yaml
   apiVersion: v1
   id: gluster-nginx-pvc
   kind: Pod
   metadata:
     name: gluster-nginx-priv
   spec:
     containers:
       - name: gluster-nginx-priv
         image: fedora/nginx
         volumeMounts:
           - mountPath: /mnt/gluster
             name: gluster-volume-claim
         securityContext:
           privileged: true
   volumes:
   ```

---

$ oc get ep
NAME                  ENDPOINTS                           AGE
---
---
gluster-cluster  192.168.59.102:1,192.168.59.103:1   2m
---

$ oc get pv
NAME                     LABELS    CAPACITY   ACCESSMODES   STATUS
---
---
claim     reason    age
---
gluster-default-volume   <none>    2Gi        RWX
Available                       2d
Volume mount within the pod.

The `gluster-claim` must reflect the name of the `PersistentVolume`.

3. Upon pod creation, the mount directory is created and the volume is attached to that mount point.
   As regular user, create a pod from the definition:

   ```
   $ oc create -f gluster-nginx-pod.yaml
   ```

4. Verify that the pod created successfully:

   ```
   $ oc get pods
   NAME                  READY     STATUS    RESTARTS   AGE
   gluster-nginx-pod     1/1       Running   0          36m
   ```
   It can take several minutes for the pod to create.

### 16.6.6. Verifying the Setup

#### 16.6.6.1. Checking the Pod SCC

1. Export the pod configuration:

   ```
   $ oc export pod <pod_name>
   ```

2. Examine the output. Check that `openshift.io/scc` has the value of `privileged`:

   **Example 16.15. Export Snippet**
   ```yaml
   metadata:
     annotations:
       openshift.io/scc: privileged
   ```

#### 16.6.6.2. Verifying the Mount

1. Access the pod and check that the volume is mounted:

   ```
   $ oc rsh <pod_name>
   [root@gluster-nginx-pvc /]# mount
   ```

2. Examine the output for the Gluster volume:

   **Example 16.16. Volume Mount**

258
192.168.59.102:gv0 on /mnt/gluster type fuse.gluster
(rw, relatime, user_id=0, group_id=0, default_permissions, allow_other,
max_read=131072)
CHAPTER 17. WORKING WITH HTTP PROXIES

17.1. OVERVIEW

Production environments can deny direct access to the Internet and instead have an HTTP or HTTPS proxy available. Configuring OpenShift to use these proxies can be as simple as setting standard environment variables in configuration or JSON files.

17.2. CONFIGURING HOSTS FOR PROXIES

1. Add the NO_PROXY, HTTP_PROXY, and HTTPS_PROXY environment variables to each host’s /etc/sysconfig/atomic-openshift-master file (for single master configuration), /etc/sysconfig/atomic-openshift-master-api, or /etc/sysconfig/atomic-openshift-master_controllers files (for multi-master configuration) and /etc/sysconfig/atomic-openshift-node file (for node configuration) depending on the type of host:

   HTTP_PROXY=http://USERNAME:PASSWORD@10.0.1.1:8080/
   HTTPS_PROXY=https://USERNAME:PASSWORD@10.0.0.1:8080/
   NO_PROXY=master.hostname.example.com

   NO_PROXY accepts a comma-separated list of hosts:

   For master hosts
   
   - Node hostname
   - Master IP or hostname
   - Service IP
   - Cluster IP

   For node hosts
   
   - Master IP or hostname
   - Service IP
   - Cluster IP

   For the Docker service
   
   - Registry service IP and hostname

   IMPORTANT

   Currently, using CIDR for IP addressing is not supported by NO_PROXY. You must add individual IP addresses for values, such as, the registry.
NOTE

The only wildcard **NO_PROXY** accepts is a single * character, which matches all hosts, and effectively disables the proxy. Each name in this list is matched as either a domain which contains the host name as a suffix, or the host name itself.

For instance, `example.com` would match `example.com`, `example.com:80`, and `www.example.com`.

2. Restart the master or node host as appropriate:

```
# systemctl restart atomic-openshift-master
# systemctl restart atomic-openshift-node
```

For multi-master installations:

```
# systemctl restart atomic-openshift-master_controllers
# systemctl restart atomic-openshift-master_api
```

OpenShift does not accept * as a wildcard attached to a domain suffix. For example, this works:

```
NO_PROXY=.example.com
```

However, this does not:

```
NO_PROXY=*.example.com
```

To deploy Hawkular Metrics on a proxied OpenShift Enterprise environment, include the following services in the **NO_PROXY** configuration:

- Hawkular Cassandra
- Hawkular Metrics
- Heapster
- Kubernetes
- Application
- OpenShift infra domain (added when using two DNS zones)

To obtain the service IPs, run:

```
$ oc get svc
```

NOTE

AutoScaling does not work on a proxied environment.

17.3. PROXYING DOCKER PULL
OpenShift node hosts need to perform push and pull operations to Docker registries. If you have a registry that does not need a proxy for nodes to access, include the `NO_PROXY` parameter with the registry's host name, the registry service's IP address, and service name. This blacklists that registry, leaving the external HTTP proxy as the only option.

1. Edit the `/etc/sysconfig/docker` file and add the variables in shell format:

   ```
   HTTP_PROXY=http://USERNAME:PASSWORD@10.0.1.1:8080/
   HTTPS_PROXY=https://USERNAME:PASSWORD@10.0.0.1:8080/
   NO_PROXY=master.hostname.example.com,172.30.123.45,docker-registry.default.svc.cluster.local
   ```

2. Restart the Docker service:

   ```
   # systemctl restart docker
   ```

17.4. USING MAVEN BEHIND A PROXY

There are three options for using Maven behind a proxy on OpenShift Enterprise:

- Generate the `settings.xml` file for the user by setting the `SHTTP_PROXY_HOST` and `SHTTP_PROXY_PORT` environment variables in the `.s2i/environment` file:

  ```
  HTTP_PROXY_HOST=<hostname>
  HTTP_PROXY_PORT=<port_number>
  ```

  Optionally, you can also set the `SHTTP_PROXY_USERNAME`, `HTTP_PROXY_PASSWORD`, and `HTTP_PROXY_NONPROXYHOSTS` variables:

  ```
  HTTP_PROXY_USERNAME=<user_name>
  HTTP_PROXY_PASSWORD=<password>
  HTTP_PROXY_NONPROXYHOSTS=<hostname>
  ```

- Move the `settings.xml` file into your application's local Git repository:

  ```
  $ mv settings.xml <git_repo>/configuration/settings.xml
  ```

- Point the `MAVEN_ARGS_APPEND` environment variable to the location of the `settings.xml` file:

  ```
  MAVEN_ARGS_APPEND=" -s path/to/file"
  ```

17.5. CONFIGURING S2I BUILDS FOR PROXIES

S2I builds fetch dependencies from various locations. You can use a `.sti/environment` file to specify simple shell variables and OpenShift will react accordingly when seeing build images.

The following are the supported proxy environment variables with example values:

```
HTTP_PROXY=http://USERNAME:PASSWORD@10.0.1.1:8080/
HTTPS_PROXY=https://USERNAME:PASSWORD@10.0.0.1:8080/
NO_PROXY=master.hostname.example.com
```
17.6. CONFIGURING DEFAULT TEMPLATES FOR PROXIES

The example templates available in OpenShift by default do not include settings for HTTP proxies. For existing applications based on these templates, modify the source section of the application’s build configuration and add proxy settings:

```yaml
...  
source:
  type: Git
  git:
    uri: https://github.com/openshift/ruby-hello-world
    httpProxy: http://proxy.example.com
    httpsProxy: https://proxy.example.com
...  
```

This is similar to the process for using proxies for Git cloning.

17.7. SETTING PROXY ENVIRONMENT VARIABLES IN PODS

You can set the NO_PROXY, HTTP_PROXY, and HTTPS_PROXY environment variables in the templates.spec.containers stanza in a deployment configuration to pass proxy connection information. The same can be done for configuring a Pod’s proxy at runtime:

```yaml
...  
containers:
  - env:
    - name: "HTTP_PROXY"
      value: "http://USER:PASSWORD@IPADDR:PORT"
...  
```

You can also use the oc env command to update an existing deployment configuration with a new environment variable:

```
$ oc env dc/frontend HTTP_PROXY=http://USER:PASSWORD@IPADDR:PORT
```

If you have a ConfigChange trigger set up in your OpenShift instance, the changes happen automatically. Otherwise, manually redeploy your application for the changes to take effect.

17.8. GIT REPOSITORY ACCESS

If your Git repository can only be accessed using a proxy, you can define the proxy to use in the source section of the BuildConfig. You can configure both a HTTP and HTTPS proxy to use. Both fields are optional.

**NOTE**

Your source URI must use the HTTP or HTTPS protocol for this to work.

```yaml
...  
source:
  type: Git
  git:  
```
uri: "https://github.com/openshift/ruby-hello-world"
httpProxy: http://proxy.example.com
httpsProxy: https://proxy.example.com
CHAPTER 18. NATIVE CONTAINER ROUTING

18.1. OVERVIEW

This topic describes how to set up container networking using existing switches and routers and the kernel networking stack in Linux. The setup requires that the network administrator or a script modifies the router or routers when new nodes are added to the cluster.

**NOTE**

The procedures outlined in this topic can be adapted to any type of router.

18.2. NETWORK LAYOUT

The following diagram shows the container networking setup described in this topic. It uses one Linux node with two network interface cards serving as a router, two switches, and three nodes connected to these switches.

18.3. NETWORK OVERVIEW

The following describes a general network setup:

- 11.11.0.0/16 is the container network.
- The 11.11.x.0/24 subnet is reserved for each node and assigned to the Docker Linux bridge.
- Each node has a route to the router for reaching anything in the 11.11.0.0/16 range, except the local subnet.
- The router has routes for each node, so it can be directed to the right node.
- Existing nodes do not need any changes when new nodes are added, unless the network topology is modified.
IP forwarding is enabled on each node.

18.4. NODE SETUP

1. Assign an unused 11.11.x.0/24 subnet IP address to the Linux bridge on the node:

   ```
   # brctl addbr lbr0
   # ip addr add 11.11.1.1/24 dev lbr0
   # ip link set dev lbr0 up
   ```

2. Modify the Docker startup script to use the new bridge. By default, the startup script is the `/etc/sysconfig/docker` file:

   ```
   # docker -d -b lbr0 --other-options
   ```

3. Add a route to the router for the 11.11.0.0/16 network:

   ```
   # ip route add 11.11.0.0/16 via 192.168.2.2 dev p3p1
   ```

4. Enable IP forwarding on the node:

   ```
   # sysctl -w net.ipv4.ip_forward=1
   ```

18.5. ROUTER SETUP

The following procedure assumes a Linux box with multiple NICs is used as a router. Modify the steps as required to use the syntax for a particular router:

1. Enable IP forwarding on the router:

   ```
   # sysctl -w net.ipv4.ip_forward=1
   ```

2. Add a route for each node added to the cluster:

   ```
   # ip route add <node_subnet> via <node_ip_address> dev <interface through which node is L2 accessible>
   # ip route add 11.11.1.0/24 via 192.168.2.1 dev p3p1
   # ip route add 11.11.2.0/24 via 192.168.3.3 dev p3p2
   # ip route add 11.11.3.0/24 via 192.168.3.4 dev p3p2
   ```
CHAPTER 19. ROUTING FROM EDGE LOAD BALANCERS

19.1. OVERVIEW

Pods inside of an OpenShift cluster are only reachable via their IP addresses on the cluster network. An edge load balancer can be used to accept traffic from outside networks and proxy the traffic to pods inside the OpenShift cluster. In cases where the load balancer is not part of the cluster network, routing becomes a hurdle as the internal cluster network is not accessible to the edge load balancer.

To solve this problem where the OpenShift cluster is using OpenShift SDN as the cluster networking solution, there are two ways to achieve network access to the pods.

19.2. INCLUDING THE LOAD BALANCER IN THE SDN

If possible, run an OpenShift node instance on the load balancer itself that uses OpenShift SDN as the networking plug-in. This way, the edge machine gets its own Open vSwitch bridge that the SDN automatically configures to provide access to the pods and nodes that reside in the cluster. The routing table is dynamically configured by the SDN as pods are created and deleted, and thus the routing software is able to reach the pods.

Mark the load balancer machine as an unschedulable node so that no pods end up on the load balancer itself:

```bash
$ oadm manage-node <load_balancer_hostname> --schedulable=false
```

If the load balancer comes packaged as a Docker container, then it is even easier to integrate with OpenShift: Simply run the load balancer as a pod with the host port exposed. The pre-packaged HAProxy router in OpenShift runs in precisely this fashion.

19.3. ESTABLISHING A TUNNEL USING A RAMP NODE

In some cases, the previous solution is not possible. For example, an F5 BIG-IP® host cannot run an OpenShift node instance or the OpenShift SDN because F5® uses a custom, incompatible Linux kernel and distribution.

Instead, to enable F5 BIG-IP® to reach pods, you can choose an existing node within the cluster network as a ramp node and establish a tunnel between the F5 BIG-IP® host and the designated ramp node. Because it is otherwise an ordinary OpenShift node, the ramp node has the necessary configuration to route traffic to any pod on any node in the cluster network. The ramp node thus assumes the role of a gateway through which the F5 BIG-IP® host has access to the entire cluster network.

Following is an example of establishing an ipip tunnel between an F5 BIG-IP® host and a designated ramp node.

**On the F5 BIG-IP® host:**

1. Set the following variables:

```bash
# F5_IP=10.3.89.66
# RAMP_IP=10.3.89.89
# TUNNEL_IP1=10.3.91.216
# CLUSTER_NETWORK=10.1.0.0/16
```
The F5_IP and RAMP_IP variables refer to the F5 BIG-IP® host’s and the ramp node’s IP addresses, respectively, on a shared, internal network.

An arbitrary, non-conflicting IP address for the F5® host’s end of the ipip tunnel.

The overlay network CIDR that the OpenShift SDN uses to assign addresses to pods.

2. Delete any old route, self, tunnel and SNAT pool:

   # tmsh delete net route $CLUSTER_NETWORK || true
   # tmsh delete net self SDN || true
   # tmsh delete net tunnels tunnel SDN || true
   # tmsh delete ltm snatpool SDN_snatpool || true

3. Create the new tunnel, self, route and SNAT pool and use the SNAT pool in the virtual servers:

   # tmsh create net tunnels tunnel SDN \
   \{ description "OpenShift SDN" local-address \n   $F5_IP profile ipip remote-address $RAMP_IP \}
   # tmsh create net self SDN \{ address \n   ${TUNNEL_IP1}/24 allow-service all vlan SDN \}
   # tmsh create net route $CLUSTER_NETWORK interface SDN
   # tmsh create ltm snatpool SDN_snatpool members add { $TUNNEL_IP1 }
   # tmsh modify ltm virtual ose-vserver source-address-translation { 
   type snat pool SDN_snatpool }
   # tmsh modify ltm virtual https-ose-vserver source-address-translation { 
   type snat pool SDN_snatpool }

On the ramp node:

1. Set the following variables:

   # F5_IP=10.3.89.66
   # TUNNEL_IP1=10.3.91.216
   # TUNNEL_IP2=10.3.91.217

   A second, arbitrary IP address for the ramp node’s end of the ipip tunnel.

2. Delete any old tunnel:

   # ip tunnel del tun1 || true

3. Create the ipip tunnel on the ramp node, using a suitable L2-connected interface (e.g., eth0):

   # ip tunnel add tun1 mode ipip \n   remote $F5_IP dev eth0
   # ip addr add $TUNNEL_IP2 dev tun1
   # ip link set tun1 up
   # ip route add $TUNNEL_IP1 dev tun1
   # ping -c 5 $TUNNEL_IP1

4. SNAT the tunnel IP with an unused IP from the SDN subnet:

   -
5. Assign this `RAMP_SDN_IP` as an additional address to `tun0` (the local SDN's gateway):

   ```
   # ip addr add ${RAMP_SDN_IP} dev tun0
   ```

6. Modify the OVS rules for SNAT:

   ```
   # set plugin based on the openshift-sdn plugin used in your cluster.
   # plugin="sdn"  # "multitenant"
   #
   # ipflowopts="cookie=0x999,ip"
   # [ "$plugin" == "multitenant" ] &&
   ipflowopts="cookie=0x999,table=1,priority=40000,ip"
   #
   # arpflowopts="cookie=0x999, table=0, arp"
   # [ "$plugin" == "multitenant" ] &&
   arpflowopts="cookie=0x999, table=1,priority=40000, arp"
   #
   # ovs-ofctl -O OpenFlow13 add-flow br0 \
   "${ipflowopts},nw_src=${TUNNEL_IP1},actions=mod_nw_src:${RAMP_SDN_IP},resubmit(,0)"
   # ovs-ofctl -O OpenFlow13 add-flow br0 \
   "${ipflowopts},nw_dst=${RAMP_SDN_IP},actions=mod_nw_dst:${TUNNEL_IP1},resubmit(,0)"
   # ovs-ofctl -O OpenFlow13 add-flow br0 \
   "$[arpflowopts], arp_tpa=${RAMP_SDN_IP}, actions=output:2"
   # if [ "$plugin" == "multitenant" ]; then
   #    ovs-ofctl -O OpenFlow13 add-flow br0 \
   #      "cookie=0x999, table=1,priority=40000,ip,nw_dst=${TUNNEL_IP1},actions=output:2"
   # fi
   ```

7. Optionally, if you do not plan on configuring the ramp node to be highly available, mark the ramp node as unschedulable. Skip this step if you do plan to follow the next section and plan on creating a highly available ramp node.

   ```
   $ oadm manage-node <ramp_node_hostname> --schedulable=false
   ```

---

**NOTE**

The F5 router plug-in integrates with F5 BIG-IP®.

### 19.3.1. Configuring a Highly Available Ramp Node
You can use OpenShift Enterprise’s ipfailover feature, which uses keepalived internally, to make the ramp node highly available from F5 BIG-IP®'s point of view. To do so, first bring up two nodes, for example called ramp-node-1 and ramp-node-2, on the same L2 subnet.

Then, choose some unassigned IP address from within the same subnet to use for your virtual IP, or VIP. This will be set as the RAMP_IP variable with which you will configure your tunnel on F5 BIG-IP®.

For example, suppose you are using the 10.20.30.0/24 subnet for your ramp nodes, and you have assigned 10.20.30.2 to ramp-node-1 and 10.20.30.3 to ramp-node-2. For your VIP, choose some unassigned address from the same 10.20.30.0/24 subnet, for example 10.20.30.4. Then, to configure ipfailover, mark both nodes with a label, such as f5rampnode:

```
$ oc label node ramp-node-1 f5rampnode=true
$ oc label node ramp-node-2 f5rampnode=true
```

Similar to instructions from the ipfailover documentation, you must now create a service account and add it to the privileged SCC. First, create the f5ipfailover service account:

```
$ echo '  { "kind": "ServiceAccount",  
    "apiVersion": "v1",  
    "metadata": { "name": "f5ipfailover" } }
' | oc create -f -
```

Next, you can manually edit the privileged SCC and add the f5ipfailover service account, or you can script editing the privileged SCC if you have jq installed. To manually edit the privileged SCC, run:

```
$ oc edit scc privileged
```

Then add the f5ipfailover service account in form system:serviceaccount:<project>:<name> to the users section:

```
...  
users:  
- system:serviceaccount:openshift-infra:build-controller  
- system:serviceaccount:default:router  
- system:serviceaccount:default:f5ipfailover
```

Alternatively, to script editing privileged SCC if you have jq installed, run:

```
$ oc get scc privileged -o json |  
jq '.users |= .+ ["system:serviceaccount:default:f5ipfailover"]' |  
oc replace scc -f -
```

Finally, configure ipfailover using your chosen VIP (the RAMP_IP variable) and the f5ipfailover service account, assigning the VIP to your two nodes using the f5rampnode label you set earlier:

```
# RAMP_IP=10.20.30.4  
# IFNAME=eth0  
# oadm ipfailover <name-tag> \  
  --virtual-ips=$RAMP_IP \  
  --interface=$IFNAME \
```
```
--watch-port=0 \
--replicas=2 \
--service-account=f5ipfailover \
--selector='f5rampnode=true'
```
CHAPTER 20. AGGREGATING CONTAINER LOGS

20.1. OVERVIEW

As an OpenShift Enterprise cluster administrator, you can deploy the EFK stack to aggregate logs for a range of OpenShift Enterprise services. Application developers can view the logs of the projects for which they have view access. The EFK stack aggregates logs from hosts and applications, whether coming from multiple containers or even deleted pods.

The EFK stack is a modified version of the **ELK stack** and is comprised of:

- **Elasticsearch**: An object store where all logs are stored.
- **Fluentd**: Gathers logs from nodes and feeds them to Elasticsearch.
- **Kibana**: A web UI for Elasticsearch.

Once deployed in a cluster, the stack aggregates logs from all nodes and projects into Elasticsearch, and provides a Kibana UI to view any logs. Cluster administrators can view all logs, but application developers can only view logs for projects they have permission to view. The stack components communicate securely.

**NOTE**

*Managing Docker Container Logs* discusses the use of `json-file` logging driver options to manage container logs and prevent filling node disks.

20.2. PRE-DEPLOYMENT CONFIGURATION

1. Ensure that you have deployed a router for the cluster.

2. Ensure that you have the necessary storage for Elasticsearch. Note that each Elasticsearch replica requires its own storage volume. See Elasticsearch for more information.

3. Ansible-based installs should create the `logging-deployer-template` template in the `openshift` project. Otherwise you can create it with the following command:

   ```bash
   $ oc create -n openshift -f /usr/share/openshift/examples/infrastructure-templates/enterprise/logging-deployer.yaml
   ```

4. Create a new project. Once implemented in a single project, the EFK stack collects logs for every project within your OpenShift Enterprise cluster. The examples in this topic use `logging` as an example project:

   ```bash
   $ oadm new-project logging --node-selector=""
   $ oc project logging
   ```

**NOTE**

Specifying a non-empty `node selector` on the project is not recommended, as this would restrict where Fluentd can be deployed. Instead, specify node selectors for the deployer to be applied to your other deployment configurations.
5. Create a secret to provide security-related files to the deployer. While the secret is necessary, the contents of the secret are optional, and will be generated for you if none are supplied. You can supply the following files when creating a new secret:

<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kibana.crt</td>
<td>A browser-facing certificate for the Kibana server.</td>
</tr>
<tr>
<td>kibana.key</td>
<td>A key to be used with the Kibana certificate.</td>
</tr>
<tr>
<td>kibana-ops.crt</td>
<td>A browser-facing certificate for the Ops Kibana server.</td>
</tr>
<tr>
<td>kibana-ops.key</td>
<td>A key to be used with the Ops Kibana certificate.</td>
</tr>
<tr>
<td>server-tls.json</td>
<td>JSON TLS options to override the Kibana server defaults. Refer to Node.JS docs for available options.</td>
</tr>
<tr>
<td>ca.crt</td>
<td>A certificate for a CA that will be used to sign all certificates generated by the deployer.</td>
</tr>
<tr>
<td>ca.key</td>
<td>A matching CA key.</td>
</tr>
</tbody>
</table>

For example:

```
$ oc secrets new logging-deployer \
   kibana.crt=/path/to/cert kibana.key=/path/to/key
```

If a certificate file is not passed as a secret, the deployer will generate a self-signed certificate instead. However, a secret is still required for the deployer to run. In this case, you can create a "dummy" secret that does not specify a certificate value:

```
$ oc secrets new logging-deployer nothing=/dev/null
```

6. Create the deployer service account:

```
$ oc create -f - <<API
apiVersion: v1
kind: ServiceAccount
metadata:
  name: logging-deployer
secrets:
- name: logging-deployer
API
```

7. Enable the Fluentd service account, which the deployer will create, that requires special privileges to operate Fluentd. Add the service account user to the security context:

```
$ oadm policy add-scc-to-user \
```
privileged system:serviceaccount:logging:aggregated-logging-fluentd

Use the new project you created earlier (e.g., logging) when specifying this service account.

Give the Fluentd service account permission to read labels from all pods:

```bash
$ oadm policy add-cluster-role-to-user cluster-reader \system:serviceaccount:logging:aggregated-logging-fluentd
```

Use the new project you created earlier (e.g., logging) when specifying this service account.

### 20.3. DEPLOYING THE EFK STACK

The EFK stack is deployed using a template.

1. Run the deployer, specifying at least the parameters in the following example (more are described in the table below):

```bash
```

Be sure to replace at least KIBANA_HOSTNAME and PUBLIC_MASTER_URL with values relevant to your deployment.

The available parameters are:

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUBLIC_MASTER_URL</td>
<td>(Required with the <code>oc process</code> command) The external URL for the master. For OAuth use.</td>
</tr>
<tr>
<td>ENABLE_OPS_CLUSTER</td>
<td>If set to true, configures a second Elasticsearch cluster and Kibana for operations logs. Fluentd splits logs between the main cluster and a cluster reserved for operations logs (which consists of /var/log/messages on nodes and the logs from the projects default, openshift, and openshift-infra). This means a second Elasticsearch and Kibana are deployed. The deployments are distinguishable by the -ops included in their names and have parallel deployment options listed below.</td>
</tr>
<tr>
<td>KIBANA_HOSTNAME, KIBANA_OPS_HOSTNAME</td>
<td>(Required with the <code>oc process</code> command) The external host name for web clients to reach Kibana.</td>
</tr>
<tr>
<td>Variable Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ES_CLUSTER_SIZE,</td>
<td>(Required with the <code>oc process</code> command) The number of instances of Elasticsearch to deploy. Redundancy requires at least three, and more can be used for scaling.</td>
</tr>
<tr>
<td>ES_OPS_CLUSTER_SIZE</td>
<td></td>
</tr>
<tr>
<td>ES_INSTANCE_RAM,</td>
<td>Amount of RAM to reserve per Elasticsearch instance. The default is 8G (for 8GB), and it must be at least 512M. Possible suffixes are G,g,M,m.</td>
</tr>
<tr>
<td>ES_OPS_INSTANCE_RAM</td>
<td></td>
</tr>
<tr>
<td>ES_NODE_QUORUM,</td>
<td>The quorum required to elect a new master. Should be more than half the intended cluster size.</td>
</tr>
<tr>
<td>ES_OPS_NODE_QUORUM</td>
<td></td>
</tr>
<tr>
<td>ES_RECOVER_AFTERSNODES,</td>
<td>When restarting the cluster, require this many nodes to be present before starting recovery. Defaults to one less than the cluster size to allow for one missing node.</td>
</tr>
<tr>
<td>ES_OPS_RECOVER_AFTERNODES</td>
<td></td>
</tr>
<tr>
<td>ES_RECOVER_EXPECTEDNODES,</td>
<td>When restarting the cluster, wait for this number of nodes to be present before starting recovery. By default, the same as the cluster size.</td>
</tr>
<tr>
<td>ES_OPS_RECOVER_EXPECTEDNODES</td>
<td></td>
</tr>
<tr>
<td>ES_RECOVER_AFTERTIME,</td>
<td>When restarting the cluster, this is a timeout for waiting for the expected number of nodes to be present. Defaults to &quot;5m&quot;.</td>
</tr>
<tr>
<td>ES_OPS_RECOVER_AFTERTIME</td>
<td></td>
</tr>
<tr>
<td>IMAGE_PREFIX</td>
<td>The prefix for logging component images. For example, setting the prefix to <code>registry.access.redhat.com/openshift3/ose</code> creates registry.access.redhat.com/openshift3/ose-logging-deployment:latest.</td>
</tr>
<tr>
<td>IMAGE_VERSION</td>
<td>The version for logging component images. For example, setting the version to 3.1.1 creates registry.access.redhat.com/openshift3/logging-deployment:3.1.1.</td>
</tr>
</tbody>
</table>

Running the deployer creates a deployer pod and prints its name.

2. Wait until the pod is running. It may take several minutes for OpenShift Enterprise to retrieve the deployer image from the registry.

**NOTE**

The logs for the `openshift` and `openshift-infra` projects are automatically aggregated and grouped into the `.operations` item in the Kibana interface.

The project where you have deployed the EFK stack (logging, as documented here) is not aggregated into `.operations` and is found under its ID.

You can watch its progress with:
$ oc get pod/<pod_name> -w

If it seems to be taking too long to start, you can retrieve more details about the pod and any associated events with:

$ oc describe pod/<pod_name>

When it runs, you can check the logs of the resulting pod to see if the deployment was successful:

$ oc logs -f <pod_name>

3. As a cluster administrator, deploy the **logging-support-template** template that the deployer created:

$ oc process logging-support-template | oc create -f -

---

**IMPORTANT**

Deployment of logging components should begin automatically. However, because deployment is triggered based on tags being imported into the ImageStreams created in this step, and not all tags are automatically imported, this mechanism has become unreliable as multiple versions are released. Therefore, manual importing may be necessary as follows.

For each ImageStream **logging-auth-proxy**, **logging-kibana**, **logging-elasticsearch**, and **logging-fluentd**, manually import the tag corresponding to the **IMAGE_VERSION** specified (or defaulted) for the deployer.

$ oc import-image <name>:<version> --from <prefix><name>:<tag>

For example:

```
$ oc import-image logging-auth-proxy:3.1.1 \ 
   --from registry.access.redhat.com/openshift3/logging-auth-proxy:3.1.1
$ oc import-image logging-kibana:3.1.1 \ 
   --from registry.access.redhat.com/openshift3/logging-kibana:3.1.1
$ oc import-image logging-elasticsearch:3.1.1 \ 
   --from registry.access.redhat.com/openshift3/logging-elasticsearch:3.1.1
$ oc import-image logging-fluentd:3.1.1 \ 
   --from registry.access.redhat.com/openshift3/logging-fluentd:3.1.1
```

---

**20.4. POST-DEPLOYMENT CONFIGURATION**
20.4.1. Elasticsearch

A highly-available environment requires at least three replicas of Elasticsearch; each on a different host. Elasticsearch replicas require their own storage, but an OpenShift Enterprise deployment configuration shares storage volumes between all its pods. So, when scaled up, the EFK deployer ensures each replica of Elasticsearch has its own deployment configuration.

Viewing all Elasticsearch Deployments

To view all current Elasticsearch deployments:

$ oc get dc --selector logging-infra=elasticsearch

Persistent Elasticsearch Storage

The deployer creates an ephemeral deployment in which all of a pod's data is lost upon restart. For production usage, add a persistent storage volume to each Elasticsearch deployment configuration.

The best-performing volumes are local disks, if it is possible to use them. Doing so requires some preparation as follows.

1. The relevant service account must be given the privilege to mount and edit a local volume, as follows:

   $ oadm policy add-scc-to-user privileged  \
       system:serviceaccount:logging:aggregated-logging-elasticsearch

   ı Use the new project you created earlier (e.g., logging) when specifying this service account.

2. Each Elasticsearch replica definition must be patched to claim that privilege, for example:

   $ for dc in $(oc get deploymentconfig --selector logging-infra=elasticsearch -o name); do
     oc scale $dc --replicas=0
     oc patch $dc \
       -p '"spec":{"template":{"spec":{"containers":
       [{"name":"elasticsearch","securityContext":{"privileged":true}}]}}}}'
     done

3. The Elasticsearch pods must be located on the correct nodes to use the local storage, and should not move around even if those nodes are taken down for a period of time. This requires giving each Elasticsearch replica a node selector that is unique to the node where an administrator has allocated storage for it. See below for directions on setting a node selector.

4. Once these steps are taken, a local host mount can be applied to each replica as in this example (where we assume storage is mounted at the same path on each node):

   $ for dc in $(oc get deploymentconfig --selector logging-infra=elasticsearch -o name); do
     oc volume $dc \
       --add --overwrite --name=elasticsearch-storage \

If using host mounts is impractical or undesirable, it may be necessary to attach block storage as a PersistentVolumeClaim as in the following example:

```bash
$ oc volume dc/logging-es-<unique> \
    --add --overwrite --name=elasticsearch-storage \
    --type=persistentVolumeClaim --claim-name=logging-es-1
```

**WARNING**

Using NFS storage directly or as a PersistentVolume (or via other NAS such as Gluster) is not supported for Elasticsearch storage, as Lucene relies on filesystem behavior that NFS does not supply. Data corruption and other problems can occur. If NFS storage is a requirement, you can allocate a large file on that storage to serve as a storage device and treat it as a host mount on each host. For example:

```bash
$ truncate -s 1T /nfs/storage/elasticsearch-1
$ mkfs.xfs /nfs/storage/elasticsearch-1
$ mount -o loop /nfs/storage/elasticsearch-1 /usr/local/es-storage
$ chown 1000:1000 /usr/local/es-storage
```

Then, use `/usr/local/es-storage` as a host-mount as described above. Performance under this solution is significantly worse than using actual local drives.

**Node Selector**

Because Elasticsearch can use a lot of resources, all members of a cluster should have low latency network connections to each other. Ensure this by directing the instances to dedicated nodes, or a dedicated region within your cluster, using a node selector.

To configure a node selector, edit each deployment configuration and add the `nodeSelector` parameter to specify the label of the desired nodes:

```yaml
apiVersion: v1
kind: DeploymentConfig
spec:
  template:
    spec:
      nodeSelector:
        nodeLabel: logging-es-node-1
```

Alternatively you can use the `oc patch` command:
$ oc patch dc/logging-es-<unique_name> \
   -p '{"spec":{"template":{"spec":{"nodeSelector":{"<label_name>":"<label_value>"}}}}}'

Changing the Scale of Elasticsearch

If you need to scale up the number of Elasticsearch instances your cluster uses, it is not as simple as changing the number of Elasticsearch cluster nodes. This is due to the nature of persistent volumes and how Elasticsearch is configured to store its data and recover the cluster. Instead, you must create a deployment configuration for each Elasticsearch cluster node.

During installation, the deployer creates templates with the Elasticsearch configurations provided to it: `logging-es-template` and `logging-es-ops-template` if the deployer was run with `ENABLE_OPS_CLUSTER=true`.

The node quorum and recovery settings were initially set based on the `CLUSTER_SIZE` value provided to the deployer. Since the cluster size is changing, those values need to be updated.

1. Prior to changing the number of Elasticsearch cluster nodes, the EFK stack should first be scaled down to preserve log data as described in `Upgrading the EFK Logging Stack`.

2. Edit the cluster template you are scaling up and change the parameters to the desired value:
   - `NODE_QUORUM` is the intended cluster size / 2 (rounded down) + 1. For an intended cluster size of 5, the quorum would be 3.
   - `RECOVER_EXPECTED_NODES` is the same as the intended cluster size.
   - `RECOVER_AFTER_NODES` is the intended cluster size - 1.

   ```bash
   $ oc edit template logging-es[-ops]-template
   ```

3. In addition to updating the template, all of the deployment configurations for that cluster also need to have the three environment variable values above updated. To edit each of the configurations for the cluster in series, you use the following.

   ```bash
   $ oc get dc -l component=es[-ops] -o name | xargs -r oc edit
   ```

4. Create an additional deployment configuration, run the following command against the Elasticsearch cluster you want to to scale up for (`logging-es-template` or `logging-es-ops-template`).

   ```bash
   $ oc new-app logging-es[-ops]-template
   ```

   These deployments will be named differently, but all will have the `logging-es` prefix. Be aware of the cluster parameters (described in the deployer parameters) based on cluster size that may need corresponding adjustment in the template, as well as existing deployments.

5. After the intended number of deployment configurations are created, scale up your cluster, starting with Elasticsearch as described in `Upgrading the EFK Logging Stack`. 

---

CHAPTER 20. AGGREGATING CONTAINER LOGS

279
NOTE

The `oc new-app logging-es[-ops]-template` command creates a deployment configuration with a persistent volume. If you want to create an Elasticsearch cluster node with a persistent volume attached to it, upon creation you can instead run the following command to create your deployment configuration with a persistent volume claim (PVC) attached.

```
$ oc process logging-es-template | oc volume -f - \
   --add --overwrite --name=elasticsearch-storage \
   --type=persistentVolumeClaim --claim-name={your_pvc}
```

20.4.2. Fluentd

Once Elasticsearch is running, scale Fluentd to every node to feed logs into Elasticsearch. The following example is for an OpenShift Enterprise instance with three nodes:

```
$ oc scale dc/logging-fluentd --replicas=3
```

You will need to scale Fluentd if nodes are added or subtracted.

When you make changes to any part of the EFK stack, specifically Elasticsearch or Fluentd, you should first scale Elasticsearch down to zero and scale Fluentd so it does not match any other nodes. Then, make the changes and scale Elasticsearch and Fluentd back.

To scale Elasticsearch to zero:

```
$ oc scale --replicas=0 dc/<ELASTICSEARCH_DC>
```

Change nodeSelector in the daemonset configuration to match zero:

**Get the fluentd node selector:**

```
$ oc get ds logging-fluentd -o yaml |grep -A 1 Selector nodeSelector:
   logginginfra-fluentd: "true"
```

**Use the `oc patch` command to modify the daemonset nodeSelector:**

```
$ oc patch ds logging-fluentd -p '{"spec":{"template":{"spec":
   "nodeSelector":{"nonexistlabel":"true"}}}}'
```

**Get the fluentd node selector:**

```
$ oc get ds logging-fluentd -o yaml |grep -A 1 Selector nodeSelector:
   "nonexistlabel: "true"
```

Scale Elasticsearch back up from zero:
20.4.3. Kibana

To access the Kibana console from the OpenShift Enterprise web console, add the `loggingPublicURL` parameter in the `/etc/origin/master/master-config.yaml` file, with the URL of the Kibana console (the `KIBANA_HOSTNAME` parameter). The value must be an HTTPS URL:

```
... assetConfig:
    ... loggingPublicURL: "https://kibana.example.com"
...
```

Setting the `loggingPublicURL` parameter creates a View Archive button on the OpenShift Enterprise web console under the Browse → Pods → `<pod_name>` → Logs tab. This links to the Kibana console.

You can scale the Kibana deployment as usual for redundancy:

```
$ oc scale dc/logging-kibana --replicas=2
```

You can see the UI by visiting the site specified at the `KIBANA_HOSTNAME` variable.

See the Kibana documentation for more information on Kibana.

20.4.4. Cleanup

You can remove everything generated during the deployment while leaving other project contents intact:

```
$ oc delete all --selector logging-infra=kibana
$ oc delete all --selector logging-infra=fluentd
$ oc delete all --selector logging-infra=elasticsearch
$ oc delete all --selector logging-infra=curator
$ oc delete all,sa,oauthclient --selector logging-infra=support
$ oc delete secret logging-fluentd logging-elasticsearch \
    logging-es-proxy logging-kibana logging-kibana-proxy \
    logging-kibana-ops-proxy
```

20.5. UPGRADING

To upgrade the EFK logging stack, see Manual Upgrades.

20.6. TROUBLESHOOTING KIBANA
Using the Kibana console with OpenShift Enterprise can cause problems that are easily solved, but are not accompanied with useful error messages. Check the following troubleshooting sections if you are experiencing any problems when deploying Kibana on OpenShift Enterprise:

Login Loop

The OAuth2 proxy on the Kibana console must share a secret with the master host’s OAuth2 server. If the secret is not identical on both servers, it can cause a login loop where you are continuously redirected back to the Kibana login page.

To fix this issue, delete the current oauthclient, and create a new one, using the same template as before:

```
$ oc delete oauthclient/kibana-proxy
$ oc process logging-support-template | oc create -f -
```

Cryptic Error When Viewing the Console

When attempting to visit the Kibana console, you may instead receive a browser error:

```
"error":"invalid_request","error_description":"The request is missing a required parameter,
   includes an invalid parameter value, includes a parameter more than once, or is otherwise malformed."
```

This can be caused by a mismatch between the OAuth2 client and server. The return address for the client must be in a whitelist so the server can securely redirect back after logging in.

Fix this issue by replacing the OAuth client entry:

```
$ oc delete oauthclient/kibana-proxy
$ oc process logging-support-template | oc create -f -
```

If the problem persists, check that you are accessing Kibana at a URL listed in the OAuth client. This issue can be caused by accessing the URL at a forwarded port, such as 1443 instead of the standard 443 HTTPS port. You can adjust the server whitelist by editing the OAuth client:

```
$ oc edit oauthclient/kibana-proxy
```

503 Error When Viewing the Console

If you receive a proxy error when viewing the Kibana console, it could be caused by one of two issues.

First, Kibana may not be recognizing pods. If Elasticsearch is slow in starting up, Kibana may timeout trying to reach it. Check whether the relevant service has any endpoints:

```
$ oc describe service logging-kibana
Name:                   logging-kibana
[...]
Endpoints:              <none>
```

If any Kibana pods are live, endpoints will be listed. If they are not, check the state of the Kibana pods and deployment. You may need to scale the deployment down and back up again.
The second possible issue may be caused if the route for accessing the Kibana service is masked. This can happen if you perform a test deployment in one project, then deploy in a different project without completely removing the first deployment. When multiple routes are sent to the same destination, the default router will only route to the first created. Check the problematic route to see if it is defined in multiple places:

```
$ oc get route --all-namespaces --selector logging-infra=support
```

### 20.7. EXTERNAL ELASTICSEARCH INSTANCE WITH FLUENTD

It is possible to configure the Fluentd pod created with aggregated logging to connect to an externally hosted Elasticsearch instance.

Fluentd knows where to send its logs to based on the `ES_HOST`, `ES_PORT`, `OPS_HOST` and `OPS_PORT` environment variables. If you have an external Elasticsearch instance that will contain both application and operations logs, ensure that `ES_HOST` and `OPS_HOST` are the same and that `ES_PORT` and `OPS_PORT` are also the same. Fluentd is configured to send its application logs to the `ES_HOST` destination and all of its operations logs to `OPS_HOST`.

If your externally hosted Elasticsearch does not make use of TLS you will need to update the `*_CLIENT_CERT`, `*_CLIENT_KEY` and `*_CA` variables to be empty. If it uses TLS but not Mutual TLS, update the `*_CLIENT_CERT` and `*_CLIENT_KEY` variables to be empty and patch or recreate the `logging-fluentd` secret with the appropriate `*_CA` for communicating with your Elasticsearch. If it uses Mutual TLS as the provided Elasticsearch does, you will just need to patch or recreate the `logging-fluentd` secret with your client key, client cert, and CA.

You can use `oc edit dc/logging-fluentd` to update your Fluentd configuration. It is advised that you first scale down your number of replicas to 0 before editing the DeploymentConfig.

**NOTE**

If you are not using the provided Kibana and Elasticsearch images, you will not have the same multi-tenant capabilities and your data will not be restricted by user access to a particular project.
CHAPTER 21. ENABLING CLUSTER METRICS

21.1. OVERVIEW

The kubelet exposes metrics that can be collected and stored in back-ends by Heapster.

As an OpenShift Enterprise administrator, you can view a cluster’s metrics from all containers and components in one user interface. These metrics are also used by horizontal pod autoscalers in order to determine when and how to scale.

This topic describes using Hawkular Metrics as a metrics engine which stores the data persistently in a Cassandra database. When this is configured, CPU and memory-based metrics are viewable from the OpenShift Enterprise web console and are available for use by horizontal pod autoscalers.

Heapster retrieves a list of all nodes from the master server, then contacts each node individually through the /stats endpoint. From there, Heapster scrapes the metrics for CPU and memory usage, then exports them into Hawkular Metrics.

Browsing individual pods in the web console displays separate sparkline charts for memory and CPU. The time range displayed is selectable, and these charts automatically update every 30 seconds. If there are multiple containers on the pod, then you can select a specific container to display its metrics.

If you have resource limits defined for your project, then you can also see a donut chart for each pod. The donut chart displays usage against the resource limit. For example: 145 Available of 200 MiB, with the donut chart showing 55 MiB Used.

21.2. BEFORE YOU BEGIN

The components for cluster metrics must be deployed to the openshift-infra project. This allows horizontal pod autoscalers to discover the Heapster service and use it to retrieve metrics that can be used for autoscaling.

All of the following commands in this topic must be executed under the openshift-infra project. To switch to the openshift-infra project:

$ oc project openshift-infra

To enable cluster metrics, you must next configure the following:

- Service Accounts
- Metrics Data Storage
- Metrics Deployer

21.3. SERVICE ACCOUNTS

You must configure service accounts for:

- Metrics Deployer
- Heapster
21.3.1. Metrics Deployer Service Account

The Metrics Deployer will be discussed in a later step, but you must first set up a service account for it:

1. Create a `metrics-deployer` service account:

   ```bash
   $ oc create -f - <<API
   apiVersion: v1
   kind: ServiceAccount
   metadata:
     name: metrics-deployer
   secrets:
     - name: metrics-deployer
   API
   ```

2. Before it can deploy components, the `metrics-deployer` service account must also be granted the `edit` permission for the `openshift-infra` project:

   ```bash
   $ oadm policy add-role-to-user \
   edit system:serviceaccount:openshift-infra:metrics-deployer
   ```

21.3.2. Heapster Service Account

The Heapster component requires access to the master server to list all available nodes and access the `/stats` endpoint for each node. Before it can do this, the Heapster service account requires the `cluster-reader` permission:

```bash
$ oadm policy add-cluster-role-to-user \
   cluster-reader system:serviceaccount:openshift-infra:heapster
```

**NOTE**

The Heapster service account is created automatically during the Deploying the Metrics Components step.

21.4. METRICS DATA STORAGE

You can store the metrics data to either persistent storage or to a temporary pod volume.

21.4.1. Persistent Storage

Running OpenShift Enterprise cluster metrics with persistent storage means that your metrics will be stored to a persistent volume and be able to survive a pod being restarted or recreated. This is ideal if you require your metrics data to be guarded from data loss.

The size of the persisted volume can be specified with the `CASSANDRA_PV_SIZE` template parameter. By default it is set to 10 GB, which may or may not be sufficient for the size of the cluster you are using. If you require more space, for instance 100 GB, you could specify it with something like this:

```bash
$ oc process -f metrics-deployer.yaml -v \
   HAWKULAR_METRICS_HOSTNAME=hawkular- \
   metrics.example.com,CASSANDRA_PV_SIZE=100Gi \
   | oc create -f -
```
The size requirement of the Cassandra storage is dependent on the cluster size. It is the administrator’s responsibility to ensure that the size requirements are sufficient for their setup and to monitor usage to ensure that the disk does not become full.

![WARNING]

Data loss will result if the Cassandra persisted volume runs out of sufficient space.

For cluster metrics to work with persistent storage, ensure that the persistent volume has the **ReadWriteOnce** access mode. If not, the persistent volume claim will not be able to find the persistent volume, and Cassandra will fail to start.

To use persistent storage with the metric components, ensure that a **persistent volume** of sufficient size is available. The creation of **persistent volume claims** is handled by the **Metrics Deployer**.

### 21.4.2. Non-Persistent Storage

Running OpenShift Enterprise cluster metrics with non-persistent storage means that any stored metrics will be deleted when the pod is deleted. While it is much easier to run cluster metrics with non-persistent data, running with non-persistent data does come with the risk of permanent data loss. However, metrics can still survive a container being restarted.

In order to use non-persistent storage, you must set the **USE_PERSISTENT_STORAGE** template option to **false** for the Metrics Deployer.

### 21.5. METRICS DEPLOYER

The Metrics Deployer deploys and configures all of the metrics components. You can configure it by passing in information from **secrets** and by passing parameters to the Metrics Deployer’s **template**.

#### 21.5.1. Using Secrets

By default, the Metrics Deployer auto-generates self-signed certificates for use between components. Because these are self-signed certificates, they are not automatically trusted by a web browser. Therefore, it is recommended to use internal certificates for anything being accessed outside of the OpenShift Enterprise cluster, and then use the re-encrypting route to provide your own custom certificates. This is especially important for the Hawkular Metrics server as it must be accessible in a browser for the web console to function.

The Metrics Deployer requires that you manually create a **metrics-deployer** secret whether you are providing your own certificates or using generated self-signed certificates.

#### 21.5.1.1. Providing Your Own Certificates

To provide your own certificates and replace the internally used ones, you can pass these values as **secrets** to the Metrics Deployer.

The preferred metrics deployment method is to pass the metrics secret with no certificates:
Then, use the a re-encrypting route to pass your custom certificates to Heapster. This allows for greater control in modifying the certificates in the future.

**NOTE**

Using a re-encrypting route allows the self-signed certificates to remain in use internally while allowing your own certificates to be used for externally access. To use a re-encrypting route, do not set the certificates as a secret, but a secret named **metrics-deployer** must still exist before the Metrics Deployer can complete.

 Optionally, provide your own certificate that is configured to be trusted by your browser by pointing your secret to the certificate's `.pem` and certificate authority certificate files:

```
$ oc secrets new metrics-deployer \\n    hawkular-metrics.pem=/home/openshift/metrics/hm.pem \\n    hawkular-metrics-ca.cert=/home/openshift/metrics/hm-ca.cert
```

**WARNING**

Setting the value using secrets will replace the internally used certificates. Therefore, these certificates must be valid for both the externally used host names as well as the external host name. For **hawkular-metrics**, this means the certificate must have a value of the literal string **hawkular-metrics** as well as the value specified in **HAWKULAR_METRICS_HOSTNAME**.

If you are unable to add the internal host name to your certificate, then you can use the re-encrypting route method.

The following table contains more advanced configuration options, detailing all the secrets which can be used by the deployer:

<table>
<thead>
<tr>
<th>Secret Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>hawkular-metrics.pem</strong></td>
<td>The <code>.pem</code> file to use for the Hawkular Metrics certificate. This certificate must contain the literal string <strong>hawkular-metrics</strong> as a host name as well as the publicly available host name used by the route. This file is auto-generated if unspecified.</td>
</tr>
<tr>
<td><strong>hawkular-metrics-ca.cert</strong></td>
<td>The certificate for the CA used to sign the <strong>hawkular-metrics.pem</strong>. This option is ignored if the <strong>hawkular-metrics.pem</strong> option is not specified.</td>
</tr>
<tr>
<td><strong>hawkular-cassandra.pem</strong></td>
<td>The <code>.pem</code> file to use for the Cassandra certificate. This certificate must contain the <strong>hawkular-cassandra</strong> host name. This file is auto-generated if unspecified.</td>
</tr>
<tr>
<td>Secret Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>hawkular-cassandra-ca.cert</code></td>
<td>The certificate for the CA used to sign the <code>hawkular-cassandra.pem</code>. This option is ignored if the <code>hawkular-cassandra.pem</code> option is not specified.</td>
</tr>
<tr>
<td><code>heapster.cert</code></td>
<td>The certificate for Heapster to use. This is auto-generated if unspecified.</td>
</tr>
<tr>
<td><code>heapster.key</code></td>
<td>The key to use with the Heapster certificate. This is ignored if <code>heapster.cert</code> is not specified</td>
</tr>
<tr>
<td><code>heapster_client_ca.cert</code></td>
<td>The certificate that generates <code>heapster.cert</code>. This is required if <code>heapster.cert</code> is specified. Otherwise, the main CA for the OpenShift Enterprise installation is used. In order for horizontal pod autoscaling to function properly, this should not be overridden.</td>
</tr>
<tr>
<td><code>heapster_allowed_users</code></td>
<td>A file containing a comma-separated list of CN to accept from certificates signed with the specified CA. By default, this is set to allow the OpenShift Enterprise service proxy to connect. If you override this, make sure to add <code>system:master-proxy</code> to the list in order to allow horizontal pod autoscaling to function properly.</td>
</tr>
</tbody>
</table>

The Heapster component uses the service name DNS registry to connect to Hawkular Metrics. In the metrics code, the URL used by Heapster to connect to Hawkular Metrics is hard-coded. It attaches the search domain and resolves to the service IP.

**21.5.1.2. Using Generated Self-Signed Certificates**

The Metrics Deployer can accept multiple certificates using secrets. If a certificate is not passed as a secret, the deployer will generate a self-signed certificate to be used instead. For the deployer to generate certificates for you, a secret is still required before it can be deployed. In this case, create a "dummy" secret that does not specify a certificate value:

```
$ oc secrets new metrics-deployer nothing=/dev/null
```

**21.5.2. Modifying the Deployer Template**

The OpenShift Enterprise installer uses a template to deploy the metrics components. The default template can be found at the following path:

```
/usr/share/openshift/examples/infrastructure-templates/enterprise/metrics-deployer.yaml
```

In case you need to make any changes to this file, copy it to another directory with the file name `metrics-deployer.yaml` and refer to the new location when using it in the following sections.

**21.5.2.1. Deployer Template Parameters**
The deployer template parameter options and their defaults are listed in the default `metrics-deployer.yaml` file. If required, you can override these values when creating the Metrics Deployer.

### Table 21.1. Template Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>METRIC_DURATION</td>
<td>The number of days metrics should be stored.</td>
</tr>
<tr>
<td>CASSANDRA_PV_SIZE</td>
<td>The persistent volume size for each of the Cassandra nodes.</td>
</tr>
<tr>
<td>USE_PERSISTENT_STORAGE</td>
<td>Set to true for persistent storage; set to false to use non-persistent storage.</td>
</tr>
<tr>
<td>REDEPLOY</td>
<td>If set to true, the deployer will try to delete all the existing components before trying to redeploy.</td>
</tr>
<tr>
<td>HAWKULAR_METRICS_HOSTNAME</td>
<td>External host name where clients can reach Hawkular Metrics.</td>
</tr>
<tr>
<td>MASTER_URL</td>
<td>Internal URL for the master, for authentication retrieval.</td>
</tr>
<tr>
<td>IMAGE_VERSION</td>
<td>Specify version for metrics components. For example, for openshift/origin-metrics-deployer:latest, set version to latest.</td>
</tr>
<tr>
<td>IMAGE_PREFIX</td>
<td>Specify prefix for metrics components. For example, for openshift/origin-metrics-deployer:latest, set prefix to openshift/origin-.</td>
</tr>
</tbody>
</table>

The only required parameter is **HAWKULAR_METRICS_HOSTNAME**. This value is required when creating the deployer because it specifies the hostname for the Hawkular Metrics route. This value should correspond to a fully qualified domain name. You will need to know the value of **HAWKULAR_METRICS_HOSTNAME** when configuring the console for metrics access.

If you are using persistent storage with Cassandra, it is the administrator’s responsibility to set a sufficient disk size for the cluster using the **CASSANDRA_PV_SIZE** parameter. It is also the administrator’s responsibility to monitor disk usage to make sure that it does not become full.

**WARNING**

Data loss will result if the Cassandra persisted volume runs out of sufficient space.

All of the other parameters are optional and allow for greater customization. For instance, if you have a custom install in which the Kubernetes master is not available under
you can specify the value to use instead with the `HAWKULAR_METRICS_HOSTNAME` parameter. If you wish to deploy a specific version of the metrics components, you can do so with the `IMAGE_VERSION` parameter.

## 21.6. DEPLOYING THE METRIC COMPONENTS

Because deploying and configuring all the metric components is handled by the Metrics Deployer, you can simply deploy everything in one step.

The following examples show you how to deploy metrics with and without persistent storage using the default template parameters. Optionally, you can specify any of the template parameters when calling these commands.

**IMPORTANT**

In accordance with upstream Kubernetes rules, metrics can be collected only on the default interface of `eth0`.

### Example 21.1. Deploying with Persistent Storage

The following command sets the Hawkular Metrics route to use `hawkular-metrics.example.com` and is deployed using persistent storage.

You must have a persistent volume of sufficient size available.

```bash
$ oc process -f metrics-deployer.yaml -v 
   HAWKULAR_METRICS_HOSTNAME=hawkular-metrics.example.com 
   | oc create -f -
```

### Example 21.2. Deploying without Persistent Storage

The following command sets the Hawkular Metrics route to use `hawkular-metrics.example.com` and is deployed without persistent storage. Remember, this is being deployed without persistent storage, so metrics data loss can occur.

```bash
$ oc process -f metrics-deployer.yaml -v 
   HAWKULAR_METRICS_HOSTNAME=hawkular-metrics.example.com,USE_PERSISTENT_STORAGE=false 
   | oc create -f -
```

## 21.7. USING A RE-ENCRYPTING ROUTE

**NOTE**

The following section is not required if the `hawkular-metrics.pem` secret was specified as a deployer secret.

By default, the Hawkular Metrics server uses self-signed certificates, which are not trusted by a browser or other external services. If you want to provide your own trusted certificate to be used for external
access, you can do so using a route with a **re-encryption termination** after deploying the metrics components.

1. First, delete the default route that uses the self-signed certificates:

   ```bash
   $ oc delete route hawkular-metrics
   ```

2. Define a new route with a **re-encryption termination**:

   ```yaml
   apiVersion: v1
   kind: Route
   metadata:
     name: hawkular-metrics-reencrypt
   spec:
     host: hawkular-metrics.example.com
     port:
       targetPort: 8444
     to:
       kind: Service
       name: hawkular-metrics
   tls:
     termination: reencrypt
     key: |
       -----BEGIN PRIVATE KEY-----
       [...] 2
       -----END PRIVATE KEY-----
     certificate: |
       -----BEGIN CERTIFICATE-----
       [...] 3
       -----END CERTIFICATE-----
     caCertificate: |
       -----BEGIN CERTIFICATE-----
       [...] 4
       -----END CERTIFICATE-----
     destinationCACertificate: |
       -----BEGIN CERTIFICATE-----
       [...] 5
       -----END CERTIFICATE-----
   ```

   **1** The value specified in the `HAWKULAR_METRICS_HOSTNAME` template parameter.

   **2 3 4** These need to define the custom certificate you wish to provide.

   **5** This needs to correspond to the CA used to sign the internal Hawkular Metrics certificate

   The CA used to sign the internal Hawkular Metrics certificate can be found from the `hawkular-metrics-certificate` secret:

   ```bash
   $ base64 -d <<< "
   `oc get -o yaml secrets hawkular-metrics-certificate \
   | grep -i hawkular-metrics-ca.certificate | awk '{print $2}'`
   ```

3. Save your route definition to a file, for example `metrics-reencrypt.yaml`, and create it:
$ oc create -f metrics-reencrypt.yaml

21.8. CONFIGURING OPENSHIFT

The OpenShift Enterprise web console uses the data coming from the Hawkular Metrics service to display its graphs. The URL for accessing the Hawkular Metrics service must be configured via the `metricsPublicURL` option in the master configuration file (`/etc/origin/master/master-config.yaml`). This URL corresponds to the route created with the `HAWKULAR_METRICS_HOSTNAME` template parameter during the deployment of the metrics components.

**NOTE**

You must be able to resolve the `HAWKULAR_METRICS_HOSTNAME` from the browser accessing the console.

For example, if your `HAWKULAR_METRICS_HOSTNAME` corresponds to `hawkular-metrics.example.com`, then you must make the following change in the `master-config.yaml` file:

```yaml
assetConfig:
  ...  
  metricsPublicURL: "https://hawkular-metrics.example.com/hawkular/metrics"
```

Once you have updated and saved the `master-config.yaml` file, you must restart your OpenShift Enterprise instance.

When your OpenShift Enterprise server is back up and running, metrics will be displayed on the pod overview pages.

**CAUTION**

If you are using self-signed certificates, remember that the Hawkular Metrics service is hosted under a different hostname and uses different certificates than the console. You may need to explicitly open a browser tab to the value specified in `metricsPublicURL` and accept that certificate.

To avoid this issue, use certificates which are configured to be acceptable by your browser.

21.9. CLEANUP

You can remove everything deployed by the metrics deployer by performing the following steps:

```bash
$ oc delete all --selector="metrics-infra"
$ oc delete sa --selector="metrics-infra"
$ oc delete templates --selector="metrics-infra"
$ oc delete secrets --selector="metrics-infra"
$ oc delete pvc --selector="metrics-infra"
```

If you also wish to remove the deployer components themselves, you can do so by performing the following steps:

```bash
$ oc delete sa metrics-deployer
$ oc delete secret metrics-deployer
```
CHAPTER 22. CUSTOMIZING THE WEB CONSOLE

22.1. OVERVIEW

Administrators can customize the web console using extensions, which let you run scripts and load custom stylesheets when the web console loads. You can change the look and feel of nearly any aspect of the user interface in this way.

22.2. LOADING CUSTOM SCRIPTS AND STYLESHEETS

To add scripts and stylesheets, edit the master configuration file. The scripts and stylesheet files must exist on the Asset Server and are added with the following options:

```json
assetConfig:
  ...
  extensionScripts:
    - /path/to/script1.js
    - /path/to/script2.js
    - ...
  extensionStylesheets:
    - /path/to/stylesheet1.css
    - /path/to/stylesheet2.css
    - ...
```

Relative paths are resolved relative to the master configuration file. To pick up configuration changes, restart the server.

Custom scripts and stylesheets are read once at server start time. To make developing extensions easier, you can reload scripts and stylesheets on every request by enabling development mode with the following setting:

```json
assetConfig:
  ...
  extensionDevelopment: true
```

When set, the web console reloads any changes to existing extension script or stylesheet files when you refresh the page in your browser. You still must restart the server when adding new extension stylesheets or scripts, however. This setting is only recommended for testing changes and not for production.

The following examples show common ways you can customize the web console.

**Customizing the Logo**

The following style changes the logo in the web console header:

```css
#header-logo {
  background-image: url("https://www.example.com/images/logo.png");
  width: 160px;
  height: 10px;
}
```
Replace the `example.com` URL with a URL to an actual image, and adjust the width and height. The ideal height is 10px.

Save the style to a file, for example `logo.css`, and add it to the master configuration file:

```yaml
assetConfig:
  ...
  extensionStylesheets:
    - /path/to/logo.css
```

**Changing the Header Color**

The following style changes the header color to dark blue:

```css
.navbar-header {
  background-color: #2B3856;
}
```

Save the style to a file, for example `theme.css`, and add it to the master configuration file:

```yaml
assetConfig:
  ...
  extensionStylesheets:
    - /path/to/theme.css
```

**Adding a Link to the Header**

The following script adds a link into the web console header:

```javascript
$('".navbar-utility"').prepend('<li><a href="http://example.com/status/">System Status</a></li>');
```

Save this script to a file, for example `nav-link.js`, and add it to the master configuration file:

```yaml
assetConfig:
  ...
  extensionScripts:
    - /path/to/nav-link.js
```

### 22.3. SERVING STATIC FILES

You can serve other files from the Asset Server as well. For example, you might want to make the CLI executable available for download from the web console or add images to use in a custom stylesheet.

Add the directory with the files you want using the following configuration option:

```yaml
assetConfig:
  ...
  extensions:
    - name: images
      sourceDirectory: /path/to/my_images
```
The files under the `/path/to/my_images` directory will be available under the URL
`/<context>/extensions/images` in the web console.

To reference these files from a stylesheet, you should generally use a relative path. For example:

```
#header-logo {
  background-image: url("../extensions/images/my-logo.png");
}
```

### 22.3.1. Enabling HTML5 Mode

The web console has a special mode for supporting certain static web applications that use the HTML5 history API:

```
assetConfig:
  ...
  extensions:
    - name: my_extension
      sourceDirectory: /path/to/myExtension
      html5Mode: true
```

Setting `html5Mode` to `true` enables two behaviors:

1. Any request for a non-existent file under `/<context>/extensions/my_extension/` instead serves `/path/to/myExtension/index.html` rather than a "404 Not Found" page.

2. The element `<base href="/">` will be rewritten in `/path/to/myExtension/index.html` to use the actual base depending on the asset configuration; only this exact string is rewritten.

This is needed for JavaScript frameworks such as AngularJS that require `base` to be set in `index.html`.

### 22.4. CUSTOMIZING THE LOGIN PAGE

You can also change the login page for the web console. Run the following command to create a template you can modify:

```
$ oadm create-login-template > login-template.html
```

Edit the file to change the styles or add content, but be careful not to remove any required parameters inside curly braces.

To use your custom login page, set the following option in the master configuration file:

```
oauthConfig:
  ...
  templates:
    login: /path/to/login-template.html
```

Relative paths are resolved relative to the master configuration file. You must restart the server after changing this configuration.
When there are multiple login providers configured or when the `alwaysShowProviderSelection` option in the `master-config.yaml` file is set to `true`, each time a user’s token to OpenShift Enterprise expires, the user is presented with this custom page before they can proceed with other tasks.

### 22.4.1. Example Usage

Custom login pages can be used to create Terms of Service information. They can also be helpful if you use a third-party login provider, like GitHub or Google, to show users a branded page that they trust and expect before being redirected to the authentication provider.

### 22.5. CUSTOMIZING THE OAUTH ERROR PAGE

When errors occur during authentication, you can change the page shown.

1. Run the following command to create a template you can modify:

   ```bash
   $ oadm create-error-template > error-template.html
   ```

2. Edit the file to change the styles or add content.
   You can use the `Error` and `ErrorCode` variables in the template. To use your custom error page, set the following option in the master configuration file:

   ```yaml
   oauthConfig:
     ...
     templates:
       error: /path/to/error-template.html
   ```

   Relative paths are resolved relative to the master configuration file.

3. You must restart the server after changing this configuration.

### 22.6. CHANGING THE LOGOUT URL

You can change the location a console user is sent to when logging out of the console by modifying the `logoutURL` parameter in the `/etc/origin/master/master-config.yaml` file:

```yaml
...  
assetConfig:
  logoutURL: "http://www.example.com"
...  
```

This can be useful when authenticating with Request Header and OAuth or OpenID identity providers, which require visiting an external URL to destroy single sign-on sessions.
### 23.1. WED FEB 01 2017

<table>
<thead>
<tr>
<th>Affected Topic</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installing → Prerequisites</td>
<td>Added instructions for installing and using the <code>atomic-openshift-excluder</code> and <code>atomic-openshift-docker-excluder</code> scripts during cluster installations and upgrades.</td>
</tr>
<tr>
<td>Installing → Quick Installation</td>
<td></td>
</tr>
<tr>
<td>Installing → Advanced Installation</td>
<td></td>
</tr>
<tr>
<td>Upgrading → Manual Upgrades</td>
<td></td>
</tr>
<tr>
<td>Upgrading → Automated Upgrades</td>
<td></td>
</tr>
</tbody>
</table>

### 23.2. MON OCT 24 2016

<table>
<thead>
<tr>
<th>Affected Topic</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installing → Prerequisites</td>
<td>Added Note box to the <code>Software Prerequisites</code> section about subscription names.</td>
</tr>
</tbody>
</table>

### 23.3. MON OCT 17 2016

<table>
<thead>
<tr>
<th>Affected Topic</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installing → Advanced Installation</td>
<td>Added more information to the <code>openshift_master_portal_net</code> parameter description in the <code>Configuring Cluster Variables</code> section.</td>
</tr>
</tbody>
</table>

### 23.4. TUE OCT 04 2016

<table>
<thead>
<tr>
<th>Affected Topic</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuring Persistent Storage → Volume Security</td>
<td>Fixed formatting of the <code>oc get project default -o yaml</code> example output within the <code>SCCs, Defaults, and Allowed Ranges</code> section.</td>
</tr>
</tbody>
</table>
### 23.5. TUE SEP 13 2016

<table>
<thead>
<tr>
<th>Affected Topic</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installing → Advanced</td>
<td>Updated the Multiple Masters Using HAProxy Inventory File example with a line about enabling <code>ntp</code> on masters to ensure proper failover as part of HA configuration.</td>
</tr>
<tr>
<td>Installation</td>
<td></td>
</tr>
</tbody>
</table>

### 23.6. TUE SEP 06 2016

<table>
<thead>
<tr>
<th>Affected Topic</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Working with HTTP Proxies</td>
<td>Added more information about the <code>NO_PROXY</code> variable.</td>
</tr>
</tbody>
</table>

### 23.7. MON AUG 29 2016

<table>
<thead>
<tr>
<th>Affected Topic</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installing → Disconnected Install</td>
<td>Fixed the tag references of images to be more generic.</td>
</tr>
</tbody>
</table>

### 23.8. TUE AUG 23 2016

<table>
<thead>
<tr>
<th>Affected Topic</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregating Container Logs</td>
<td>Removed <code>oc set volume</code> references, as <code>oc volume</code> is the correct command until OpenShift Enterprise 3.2.</td>
</tr>
<tr>
<td>Enabling Cluster Metrics</td>
<td>Added clarifying details to the Providing Your Own Certificates section.</td>
</tr>
<tr>
<td>Upgrading → Performing</td>
<td>Added manual upgrade steps to get the latest templates from <code>openshift-ansible-roles</code>.</td>
</tr>
<tr>
<td>Manual Cluster Upgrades</td>
<td></td>
</tr>
</tbody>
</table>

### 23.9. MON AUG 15 2016

<table>
<thead>
<tr>
<th>Affected Topic</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregating Container Logs</td>
<td>Added information on log locations within Kibana to the Deploying the EFK Stack section.</td>
</tr>
</tbody>
</table>

### 23.10. MON AUG 08 2016
<table>
<thead>
<tr>
<th>Affected Topic</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installing → Prerequisites</td>
<td>Updated Important boxes in the System Requirements and Installing Docker sections with more specific details regarding Docker versions and <strong>yum update.</strong> <em>(BZ#1341142)</em></td>
</tr>
<tr>
<td>Aggregating Container Logs</td>
<td>Added that NFS is a not suitable for Lucene storage, NFS is not supported, and how to use local storage.</td>
</tr>
</tbody>
</table>

**23.11. MON AUG 01 2016**

<table>
<thead>
<tr>
<th>Affected Topic</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Routing from Edge Load Balancers</td>
<td>Added a link connecting F5 router and Routing from Edge Load Balancers topics within the Establishing a Tunnel Using a Ramp Node section.</td>
</tr>
<tr>
<td>Working With HTTP Proxies</td>
<td>Added <strong>Using Maven Behind a Proxy</strong> section.</td>
</tr>
</tbody>
</table>

**23.12. WED JUL 27 2016**

<table>
<thead>
<tr>
<th>Affected Topic</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installing → Prerequisites</td>
<td>Added TCP/UDP information to the xref:prereq-network-access[Network Access] tables.</td>
</tr>
<tr>
<td>Installing → Disconnected Installation</td>
<td>Fixed command in <strong>Syncing Repositories</strong> section.</td>
</tr>
<tr>
<td>Configuring Persistent Storage</td>
<td>Added important box about changing <strong>fstype</strong> field in a persistent volume configuration in several files.</td>
</tr>
</tbody>
</table>

**23.13. WED JUL 20 2016**

<table>
<thead>
<tr>
<th>Affected Topic</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installing → Prerequisites</td>
<td>Added an Important box to the <strong>System Requirements</strong> section.</td>
</tr>
<tr>
<td></td>
<td>Corrected information in the <strong>Host Recommendations</strong> section.</td>
</tr>
<tr>
<td></td>
<td>Described which <strong>required ports</strong> are necessary for master self-communication.</td>
</tr>
</tbody>
</table>
### Advanced LDAP Configuration

New set of topics for advanced LDAP configuration:

- Setting up SSSD for LDAP Failover
- Configuring Form-Based Authentication
- Configuring Extended LDAP Attributes

### Aggregating Container Logs

Rewording and clarifications.

Expanded documentation on scaling up Elasticsearch instances.

Added a section on sending logs to an external source.

### Enabling Cluster Metrics

Added a Template Parameters table to the Deployer Template Parameters section.

---

### 23.14. MON JUN 13 2016

<table>
<thead>
<tr>
<th>Affected Topic</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregating Container Logs</td>
<td>Specified the correct units for <strong>ES_INSTANCE_RAM</strong> and <strong>ES_OPS_INSTANCE_RAM</strong>.</td>
</tr>
<tr>
<td>Deploying a Router</td>
<td>Added an Important box regarding default resource requests for router pods.</td>
</tr>
<tr>
<td>Deploying a Docker Registry</td>
<td>Updated the example of using an existing persistent volume claim (PVC) to a matching configuration for docker-registry PVC.</td>
</tr>
</tbody>
</table>

### 23.15. FRI JUN 10 2016

<table>
<thead>
<tr>
<th>Affected Topic</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installing  →  Advanced Installation</td>
<td>Replaced the <strong>openshift_docker_log_options</strong> Ansible variable with <strong>openshift_docker_options</strong> in the Configuring Host Variables section.</td>
</tr>
<tr>
<td></td>
<td>Updated <strong>openshift_router_selector</strong> to its new name of <strong>openshift_hosted_router_selector</strong>.</td>
</tr>
<tr>
<td>Affected Topic</td>
<td>Description of Change</td>
</tr>
<tr>
<td>----------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Installing → Deploying a Docker Registry</td>
<td>Fixed examples in the Securing the Registry section to use consistent <code>--cert</code> and <code>--key</code> values. Also, clarify the origin of the <code>ca.crt</code> file that must be installed per-node.</td>
</tr>
<tr>
<td>Configuring Authentication</td>
<td>Added a note on how to obtain the <code>htpasswd</code> utility.</td>
</tr>
<tr>
<td>Customizing the Web Console</td>
<td>Added that each time a user’s token to OpenShift Enterprise expires, the user is presented with a custom page. Also, added use cases for custom login pages.</td>
</tr>
</tbody>
</table>

**23.16. FRI JUN 03 2016**

<table>
<thead>
<tr>
<th>Affected Topic</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installing → Advanced Installation</td>
<td>Updated the location of the <code>scaleup.yml</code> playbook in the Adding Nodes to an Existing Cluster section.</td>
</tr>
<tr>
<td>Installing → Deploying a Docker Registry</td>
<td>Removed support information for upstream registry configuration not relevant to OpenShift Enterprise.</td>
</tr>
</tbody>
</table>

**23.17. MON MAY 30 2016**

<table>
<thead>
<tr>
<th>Affected Topic</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installing → Advanced Installation</td>
<td>Updated the parameter name <code>docker_log_options</code> to <code>openshift_docker_log_options</code> in the Host Variables table.</td>
</tr>
<tr>
<td>Installing → Disconnected Installation</td>
<td>Fixed some outdated image names.</td>
</tr>
<tr>
<td>Installing → Prerequisites</td>
<td>Added an Important box to the Sizing Recommendations section advising that oversubscribing the physical resources on a node affects resource guarantees the Kubernetes scheduler makes during pod placement.</td>
</tr>
<tr>
<td>Installing → Deploying a Docker Registry</td>
<td>Added support information for upstream registry configuration.</td>
</tr>
<tr>
<td>Working with HTTP Proxies</td>
<td>Updated the example in the Configuring Default Templates for Proxies section to use <code>https</code> for GitHub access.</td>
</tr>
<tr>
<td>Persistent Storage Examples → Backing Docker Registry with GlusterFS Storage</td>
<td>New topic about how to attach a GlusterFS persistent volume to the Docker Registry.</td>
</tr>
</tbody>
</table>
## 23.18. TUE MAY 10 2016

<table>
<thead>
<tr>
<th>Affected Topic</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upgrading → Manual Upgrades</td>
<td>Added the Upgrading the EFK Logging Stack section.</td>
</tr>
<tr>
<td>Configuring Persistent Storage → Persistent Storage Using GlusterFS</td>
<td>Updated for clarity throughout. Enhanced the Volume Security section significantly.</td>
</tr>
<tr>
<td>Configuring Persistent Storage → Persistent Storage Using Ceph Rados Block Device (RBD)</td>
<td>Updated for clarity throughout. Add the Creating the Ceph Secret and Volume Security section.</td>
</tr>
<tr>
<td>Persistent Storage Examples</td>
<td>New topic category that includes the following topics:</td>
</tr>
<tr>
<td></td>
<td>- Sharing an NFS Persistent Volume (PV) Across Two Pods: Provides an end-to-end example of how to use an existing NFS cluster and OpenShift Enterprise persistent store.</td>
</tr>
<tr>
<td></td>
<td>- Complete Example Using GlusterFS: Provides an end-to-end example of how to use an existing Gluster cluster as an OpenShift Enterprise persistent store.</td>
</tr>
<tr>
<td></td>
<td>- Complete Example Using Ceph RBD: Provides an end-to-end example of using an existing Ceph cluster as an OpenShift Enterprise persistent store.</td>
</tr>
<tr>
<td>Enabling Cluster Metrics</td>
<td>Updated the port value for the re-encrypting port to 8444 for OpenShift Enterprise, which is different from the value for OpenShift Origin, which uses 8443.</td>
</tr>
<tr>
<td></td>
<td>Simplified steps in the Cleanup section.</td>
</tr>
<tr>
<td></td>
<td>Added extra warnings for Cassandra and its disk size in the Persistent Storage and Deployer Template Parameters sections.</td>
</tr>
</tbody>
</table>

## 23.19. WED APR 27 2016

<table>
<thead>
<tr>
<th>Affected Topic</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuring Persistent Storage → Persistent Storage Using NFS</td>
<td>Updated the Export Settings section to note the no_wdelay NFS export option.</td>
</tr>
<tr>
<td>Installing → Deploying a Docker Registry</td>
<td>Updated the Known Issues section to note the no_wdelay NFS export option.</td>
</tr>
</tbody>
</table>
### 23.20. MON APR 18 2016

<table>
<thead>
<tr>
<th>Affected Topic</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Working with HTTP Proxies</td>
<td>Added specific <code>/etc/sysconfig</code> files to the Configuring Hosts for Proxies section. Added information explaining that OpenShift does not accept an asterisk as a wildcard attached to a domain suffix.</td>
</tr>
<tr>
<td>Installing → Advanced Installation</td>
<td>Fixed syntax of examples in the Configuring Custom Certificates section to be in proper INI format. Added an Adding Nodes to an Existing Cluster section on using the <code>scaleup.yml</code> playbook. (BZ#1324571)</td>
</tr>
<tr>
<td>Installing → Disconnected Installation</td>
<td>New topic on disconnected installations, detailing how to install OpenShift Enterprise in datacenters that do not have access to the Internet.</td>
</tr>
</tbody>
</table>

### 23.21. WED APR 06 2016

<table>
<thead>
<tr>
<th>Affected Topic</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregating Container Logs</td>
<td>Removed references to non-existent roles in the Pre-deployment Configuration section. (BZ#1324571)</td>
</tr>
</tbody>
</table>

### 23.22. MON APR 04 2016

<table>
<thead>
<tr>
<th>Affected Topic</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installing → Prerequisites</td>
<td>Updated the System Requirements and Installing Docker sections to take into account the release of Docker 1.9. Added the Cloud Provider Considerations section and documented ports 2049, 5404, 5405, and 9000 in the Required Ports section.</td>
</tr>
<tr>
<td>Installing → Advanced Installation</td>
<td>Added information about <code>regionsinfra</code> to the Configuring Node Host Labels section and added <code>openshift_router_selector</code> and <code>openshift_registry_selector</code> to the Host Variables table.</td>
</tr>
<tr>
<td>Affected Topic</td>
<td>Description of Change</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>---------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Aggregating Container Logs</td>
<td>Updated significantly throughout to fix errors and recommended practices.</td>
</tr>
<tr>
<td>Enabling Cluster Metrics</td>
<td>Fixed typo of the <code>destinationCACertificate</code> parameter name.</td>
</tr>
</tbody>
</table>

### 23.23. TUE MAR 29 2016

<table>
<thead>
<tr>
<th>Affected Topic</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploying a Docker Registry</td>
<td>Added an Important box about writing to the host directory in the <code>Storage for the Registry</code> section.</td>
</tr>
<tr>
<td>Configuring Persistent Storage → Persistent Storage Using NFS</td>
<td>Updated for clarity throughout.</td>
</tr>
<tr>
<td></td>
<td>Enhanced the <code>Volume Security</code> section significantly.</td>
</tr>
<tr>
<td></td>
<td>Added the <code>Additional Configuration and Troubleshooting</code> section.</td>
</tr>
<tr>
<td>Configuring Persistent Storage → Volume Security</td>
<td>Updated significantly for clarity throughout.</td>
</tr>
</tbody>
</table>

### 23.24. MON MAR 21 2016

<table>
<thead>
<tr>
<th>Affected Topic</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installing</td>
<td>Fixed broken links.</td>
</tr>
</tbody>
</table>

### 23.25. THU MAR 17 2016

<table>
<thead>
<tr>
<th>Affected Topic</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loading the Default Image Streams and Templates</td>
<td>Moved and updated the &quot;First Steps&quot; topic to become the <code>Loading the Default Image Streams and Templates</code> topic</td>
</tr>
<tr>
<td>Upgrading → Manual Upgrades</td>
<td>Changed a known issue to a fix regarding liveness and readiness probes.</td>
</tr>
<tr>
<td>Deploying a Docker Registry</td>
<td>Changed command to update the liveness probe to use <code>oc patch</code> instead of <code>sed</code>.</td>
</tr>
<tr>
<td>Enabling Cluster Metrics</td>
<td>Added the <code>Using a Re-encrypting Route</code> section.</td>
</tr>
</tbody>
</table>
### Affected Topic | Description of Change
--- | ---
Advanced Installation | Combined duplicate `openshift_node_kubelet_args` descriptions and moved all of the content to the Host Variables table.
Aggregating Container Logs | Fixed some errors and added some extra information.

### 23.26. MON MAR 7 2016

<table>
<thead>
<tr>
<th>Affected Topic</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installing → Advanced Installation</td>
<td>Clarified in the Configuring Ansible section that the services and cluster networks also cannot overlap with networks to which the master and nodes need access, and not just networks to which the pods need access. Modifying the SDN-related Ansible cluster variables in the Configuring Ansible section to be more consistent with each other in general.</td>
</tr>
<tr>
<td>Installing → Deploying a Docker Registry</td>
<td>Mentioned default tag <code>latest</code>. Clarified importance of the project name in the pull specification. Added section Maintaining the Registry IP Address.</td>
</tr>
<tr>
<td>Upgrading</td>
<td>In the Automated Upgrades and Manual Upgrades sections, added guidance about verifying that custom configurations are added to the updated <code>/etc/sysconfig/</code> paths after upgrading from OSE 3.0 to 3.1. (BZ#1284504)</td>
</tr>
<tr>
<td>Configuring the SDN</td>
<td>Added an Important box to the Configuring the Pod Network on Masters section noting that <code>clusterNetworkCIDR</code> can now be changed under certain conditions.</td>
</tr>
<tr>
<td>Configuring for AWS</td>
<td>Added the Applying Configuration Changes section. (BZ#1314085)</td>
</tr>
<tr>
<td>Persistent Storage → Persistent Storage Using NFS</td>
<td>Updated the &quot;SELinux and NFS Export Settings&quot; section to distinguish between NFSv3 and NFSv4 port requirements.</td>
</tr>
<tr>
<td>Aggregating Container Logs</td>
<td>Added a Note box to the Pre-deployment Configuration section recommending use of node selectors. Fixed a service account name reference.</td>
</tr>
<tr>
<td>Enabling Cluster Metrics</td>
<td>Added a Note box about the cluster metrics template location.</td>
</tr>
</tbody>
</table>

### 23.27. MON FEB 29 2016
### Affected Topic | Description of Change
--- | ---
**Upgrading** | Converted the "Upgrading OpenShift" topic into its own **Upgrading** directory with separate topics for **Performing Automated Cluster Upgrades** and **Performing Manual Cluster Upgrades**.

**Upgrading from Pacemaker to Native HA** | New topic providing instructions on upgrading a multiple master cluster from Pacemaker to native HA.

**Enabling Cluster Metrics** | Removed the template in the "Creating the Deployer Template" section and fixed an incorrect file location.

**Aggregating Container Logs** | Added a step within the **Pre-deployment Configuration** section indicating that you must switch to your new project after creating it.

**Prerequisites** | Fixed the `/etc/selinux/config` file path in the **SELinux** section.

**Advanced Installation** | Added notes indicating that moving from a single master cluster to multiple masters after installation is not supported.

---

**23.28. MON FEB 22 2016**

### Affected Topic | Description of Change
--- | ---
**Configuring Custom Certificates** | In the **Configuring Custom Certificates** section, replaced `publicMasterURL` with `masterPublicURL`.

**Installing → Prerequisites** | Added an **SELinux** section to include guidance that SELinux must be enabled, or the installer will fail.

**Enabling Cluster Metrics** | Added the **Cleanup** section with instructions on how to remove a metrics deployment.

**Syncing Groups With LDAP** | Updated the **Running LDAP Sync** section with better example command formatting.

**Configuring Authentication** | Updated the "Apache Authentication Using RequestHeaderIdentityProvider" example to use the `/etc/origin/master/htpasswd` file path.

| | Added a section for the **Keystone identity provider**.

**Advanced Installation** | Updated example inventory files to show the `/etc/origin/master/htpasswd` file path.

| | Clarified in the **Verifying the Installation** section to run the `oc get nodes` command on the master host.
<table>
<thead>
<tr>
<th>Affected Topic</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Routing from Edge Load Balancers</td>
<td>Corrected the <code>/run/openshift-sdn/config.env</code> path in the Establishing a Tunnel Using a Ramp Node section.</td>
</tr>
<tr>
<td>Installing → Deploying a Docker Registry</td>
<td>Added the Advanced: Overriding the Registry Configuration section.</td>
</tr>
</tbody>
</table>

### 23.29. MON FEB 15 2016

<table>
<thead>
<tr>
<th>Affected Topic</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installing → Prerequisites</td>
<td>Added a new Managing Docker Container Logs section.</td>
</tr>
<tr>
<td></td>
<td>Updated to include guidance on how to check if Docker is running.</td>
</tr>
<tr>
<td>Installing → Advanced Installation</td>
<td>Listed <code>docker_log_options</code> as an host variable in the Configuring Ansible section.</td>
</tr>
<tr>
<td>Aggregating Container Logs</td>
<td>Added a Note box about <code>json-file</code> logging driver options.</td>
</tr>
</tbody>
</table>

### 23.30. MON FEB 08 2016

<table>
<thead>
<tr>
<th>Affected Topic</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installing → Prerequisites</td>
<td>Updated the System Requirements section to clarify that instances can be running on a private IaaS, not just a public one.</td>
</tr>
</tbody>
</table>

### 23.31. THU FEB 04 2016

<table>
<thead>
<tr>
<th>Affected Topic</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installing → Deploying a Docker Registry</td>
<td>Updated the Securing the Registry section to account for the liveness probe that is now added to new registries by default starting in OpenShift Enterprise 3.1.1. (BZ#1302956)</td>
</tr>
<tr>
<td>Configuring for AWS</td>
<td>Fixed the default node configuration file path.</td>
</tr>
<tr>
<td></td>
<td>Corrected instructions on setting access key environment variables.</td>
</tr>
<tr>
<td>Configuring for GCE</td>
<td>Fixed the default node configuration file path.</td>
</tr>
</tbody>
</table>
Configuring Persistent Storage → Dynamically Provisioning Persistent Volumes

New topic on the experimental feature for allowing users to request dynamically provisioned persistent storage based on the configured cloud provider. Available in Technology Preview starting in OpenShift Enterprise 3.1.1.

### 23.32. MON FEB 01 2016

<table>
<thead>
<tr>
<th>Affected Topic</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuring for OpenStack</td>
<td>Changed <code>&lt;instance_ID&gt;</code> to <code>&lt;instance_name&gt;</code> in the Configuring Nodes section for readability.</td>
</tr>
</tbody>
</table>

### 23.33. THU JAN 28 2016

OpenShift Enterprise 3.1.1 release.

<table>
<thead>
<tr>
<th>Affected Topic</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installing → Prerequisites</td>
<td>Updated to include support for RHEL Atomic Host.</td>
</tr>
<tr>
<td>Installing → RPM vs Containerized</td>
<td>New topic discussing differences between RPM and containerized installations.</td>
</tr>
<tr>
<td>Installing → Quick Installation</td>
<td>Updated to include support for RHEL Atomic Host and containerized installations.</td>
</tr>
<tr>
<td></td>
<td>The former &quot;Prerequisites&quot; section in this topic has been renamed to Before You Begin and enhanced to differentiate from the actual Prerequisites topic.</td>
</tr>
<tr>
<td>Installing → Advanced Installation</td>
<td>Updated to include support for RHEL Atomic Host and containerized installations.</td>
</tr>
<tr>
<td></td>
<td>The former &quot;Prerequisites&quot; section in this topic has been renamed to Before You Begin and enhanced to differentiate from the actual Prerequisites topic.</td>
</tr>
<tr>
<td>Upgrading</td>
<td>Added the Upgrading to OpenShift Enterprise 3.1 Asynchronous Releases section and various enhancements to support the OpenShift Enterprise 3.1.1 release.</td>
</tr>
<tr>
<td>Syncing Groups With LDAP</td>
<td>Updated to promote the <code>openshift ex sync-groups</code> command to <code>oadm groups sync</code> and added the Running a Group Pruning Job section.</td>
</tr>
</tbody>
</table>

### 23.34. TUE JAN 26 2016
<table>
<thead>
<tr>
<th>Affected Topic</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabling Cluster Metrics</td>
<td>Fixed the <code>metrics-deployer.yaml</code> file path.</td>
</tr>
<tr>
<td>Installing → Prerequisites</td>
<td>Added a Warning box about wildcards and DNS server entries in the <code>/etc/resolv.conf</code> file.</td>
</tr>
<tr>
<td>Configuring Persistent Storage → Persistent Storage Using Ceph Rados Block Device (RBD)</td>
<td>Fixed the <code>ceph-common</code> package name.</td>
</tr>
<tr>
<td>Configuring Persistent Storage → Persistent Storage Using NFS</td>
<td>Removed a contradictory Note box about NFS and SELinux.</td>
</tr>
</tbody>
</table>

23.35. MON JAN 19 2016

<table>
<thead>
<tr>
<th>Affected Topic</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installing → Advanced Installation</td>
<td>Added custom certificate parameters and added the Configuring Custom Certificates section.</td>
</tr>
<tr>
<td>Installing → Deploying a Docker Registry</td>
<td>Enhanced the Accessing the Registry Directly section, including organizing all user-related requirements under a User Prerequisites subsection. (<a href="#">BZ#1273412</a>)</td>
</tr>
<tr>
<td>Downgrading OpenShift</td>
<td>New topic for downgrading from OpenShift Enterprise 3.1 to 3.0.</td>
</tr>
<tr>
<td>Configuring Custom Certificates</td>
<td>New topic for configuring custom certificates after initial installation.</td>
</tr>
<tr>
<td>Configuring Authentication</td>
<td>Added the <code>mappingMethod</code> parameter to all examples.</td>
</tr>
<tr>
<td>Configuring for OpenStack</td>
<td>Added references to <code>nodeName</code> in the Configuring Nodes section.</td>
</tr>
<tr>
<td></td>
<td>Fixed the default node configuration file path.</td>
</tr>
<tr>
<td>Aggregating Container Logs</td>
<td>Fixed the path to the <code>logging-deployer.yaml</code> file.</td>
</tr>
<tr>
<td>Enabling Cluster Metrics</td>
<td>Added information about Metrics Deployer certificates and the <code>nothing=/dev/null</code> option.</td>
</tr>
<tr>
<td></td>
<td>Added clarification about required host names for the Hawkular Metrics certificate.</td>
</tr>
</tbody>
</table>
OpenShift Enterprise 3.1 release.

OpenShift Enterprise 3.1 Installation and Configuration