OpenShift Container Platform 3.9

Upgrading Clusters

OpenShift Container Platform 3.9 Upgrading Clusters

Last Updated: 2019-12-17

OpenShift Container Platform 3.9 Upgrading Clusters

OpenShift Container Platform 3.9 Upgrading Clusters

Legal Notice

Abstract

Upgrade your OpenShift Container Platform 3.9 cluster with this guide

Table of Contents

CHAPTER 1. UPGRADE METHODS AND STRATEGIES

1.1. INTRODUCTION TO UPGRADING CLUSTERS

When new versions of OpenShift Container Platform 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 3.7 to 3.9, and applying asynchronous errata updates within a minor version (3.9.z releases). See the OpenShift Container Platform 3.9 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 Container Platform 3 and require a fresh installation.

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

Before upgrading, ensure that all OpenShift Container Platform services are running well.

In the event of a control plane upgrade failure, check the versions of your masters to ensure that all versions are the same. If your masters are all the same version, re-run the upgrade. If they differ, downgrade the masters to match the lower versioned master, then re-run the upgrade.

IMPORTANT

The OpenShift Container Platform 3.9 release includes a merge of features and fixes from Kubernetes 1.8 and 1.9. As a result, the upgrade process from OpenShift Container Platform 3.7 completes with the cluster fully upgraded to OpenShift Container Platform 3.9, seemingly "skipping" the 3.8 release. Technically, the OpenShift Container Platform 3.7 cluster is first upgraded to 3.8-versioned packages, and then the process immediately continues upgrading to OpenShift Container Platform 3.9 automatically. Your cluster should only remain at 3.8-versioned packages for as long as it takes to successfully complete the upgrade to OpenShift Container Platform 3.9.

1.2. UPGRADE METHODS

There are two methods available for performing OpenShift Container Platform cluster upgrades: automated or manual.

1.2.1. Automated Method

The automated upgrade method uses Ansible playbooks to automate the tasks needed to upgrade a OpenShift Container Platform cluster. You should use the inventory file that you used during initial installation or during the last time that the upgrade was successful to run the upgrade playbook. Using this method allows you to choose between either upgrade strategy: in-place upgrades or blue-green deployments.

1.2.2. Manual Method

The manual upgrade method breaks down the steps that happen during an automated Ansible-based upgrade and provides the equivalent commands to run manually. Using this method describes the in-place upgrade strategy.

1.3. UPGRADE STRATEGIES

When using the automated upgrade method, there are two strategies you can take for performing the OpenShift Container Platform cluster upgrade: in-place upgrades or blue-green deployments. When using the manual upgrade method, an in-place upgrade is described.

1.3.1. In-place Upgrades

With in-place upgrades, the cluster upgrade is performed on all hosts in a single, running cluster: first masters and then nodes. Pods are evacuated off of nodes and recreated on other running nodes before a node upgrade begins; this helps reduce downtime of user applications.

If you installed using the quick or advanced installation, you can perform an automated in-place upgrade. Alternatively, you can upgrade in-place manually.

NOTE

Quick Installation is now deprecated in OpenShift Container Platform 3.9 and will be completely removed in a future release.

Quick installation will only be capable of installing 3.9. It will not be able to upgrade from 3.7 or 3.8 to 3.9.

1.3.2. Blue-green Deployments

The blue-green deployment upgrade method follows a similar flow to the in-place method: masters and etcd servers are still upgraded first, however a parallel environment is created for new nodes instead of upgrading them in-place.

This method allows administrators to switch traffic from the old set of nodes (e.g., the "blue" deployment) to the new set (e.g., the "green" deployment) after the new deployment has been verified. If a problem is detected, it is also then easy to rollback to the old deployment quickly.

CHAPTER 2. PERFORMING AUTOMATED IN-PLACE CLUSTER UPGRADES

2.1. OVERVIEW

If you installed using the advanced installation and the inventory file that was used is available, you can use upgrade playbooks to automate the OpenShift cluster upgrade process.

The OpenShift Container Platform 3.9 release includes a merge of features and fixes from Kubernetes 1.8 and 1.9. As a result, the upgrade process from OpenShift Container Platform 3.7 completes with the cluster fully upgraded to OpenShift Container Platform 3.9, seemingly "skipping" the 3.8 release. Technically, the OpenShift Container Platform 3.7 cluster is first upgraded to 3.8-versioned packages, and then the process immediately continues upgrading to OpenShift Container Platform 3.9 automatically. Your cluster should only remain at 3.8-versioned packages for as long as it takes to successfully complete the upgrade to OpenShift Container Platform 3.9.

IMPORTANT

As of OpenShift Container Platform 3.9, the quick installation method is deprecated. In a future release, it will be removed completely. In addition, using the quick installer to upgrade from version 3.7 to 3.9 is not supported.

The automated 3.7 to 3.9 control plane upgrade performs the following steps for you:

- A backup of all etcd data is taken for recovery purposes.
- The API and controllers are updated from 3.7 to 3.8.
- Internal data structures are updated to 3.8.
- A second backup of all etcd data is taken for recovery purposes.
- The API and controllers are updated from 3.8 to 3.9.
- Internal data structures are updated to 3.9.
- The default router, if one exists, is updated from 3.7 to 3.9.
- The default registry, if one exists, is updated from 3.7 to 3.9.
- The default image streams and InstantApp templates are updated.

The automated 3.7 to 3.9 node upgrade performs a rolling update of nodes, which:

- Marks a subset of nodes unschedulable and drains them of pods.
- Updates node components from 3.7 to 3.9 (including openvswitch and container runtime).
- Returns those nodes to service.

IMPORTANT

- Ensure that you have met all prerequisites before proceeding with an upgrade. Failure to do so can result in a failed upgrade.
- If you are using GlusterFS, see Special Considerations When Using Containerized GlusterFS before proceeding.
- If you are using GCE Persistent Disk (gcePD), see Special Considerations When Using gcePD before proceeding.
- The day before the upgrade, validate OpenShift Container Platform storage migration to ensure potential issues are resolved prior to the outage window:

```
$ oc adm migrate storage --include=* --loglevel=2 --confirm --config
/etc/origin/master/admin.kubeconfig
```

Automated upgrade playbooks are run via Ansible directly using the **ansible-playbook** command with an inventory file, similar to the advanced installation method. The same *v3_9* upgrade playbooks can be used for either of the following scenarios:

- Upgrading existing OpenShift Container Platform 3.7 clusters to 3.9
- Upgrading existing OpenShift Container Platform 3.9 clusters to the latest asynchronous errata updates

IMPORTANT

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

2.2. PREPARING FOR AN AUTOMATED UPGRADE

IMPORTANT

Before upgrading Upgrading your cluster to OpenShift Container Platform 3.9 , the cluster must be already upgraded to the latest asynchronous release of version 3.7 . If your cluster is at a version earlier than 3.7, you must first upgrade incrementally (e.g., 3.5 to 3.6, then 3.6 to 3.7).

NOTE

Before attempting the upgrade, follow the guidance in Environment health checks to verify the cluster's health. This will confirm that nodes are in the **Ready** state, running the expected starting version, and will ensure that there are no diagnostic errors or warnings.

If you are completing a large-scale upgrade, which involves at least 10 worker nodes and thousands of projects and pods, review Special considerations for large-scale upgrades to prevent upgrade failures.

To prepare for an automated upgrade:

1. Pull the latest subscription data from RHSM:

```
# subscription-manager refresh
```

2. If you are upgrading from OpenShift Container Platform 3.7 to 3.9:

    a. Manually disable the 3.7 repository and enable *both* the 3.8 and 3.9 repositories on each master and node host. You must also enable the **rhel-7-server-ansible-2.4-rpms** repository, which is a new requirement starting with OpenShift Container Platform 3.9:

    ```
    # subscription-manager repos --disable="rhel-7-server-ose-3.7-rpms" \
        --enable="rhel-7-server-ose-3.8-rpms" \
        --enable="rhel-7-server-ose-3.9-rpms" \
        --enable="rhel-7-server-rpms" \
        --enable="rhel-7-server-extras-rpms" \
        --enable="rhel-7-server-ansible-2.4-rpms" \
        --enable="rhel-7-fast-datapath-rpms"
    # yum clean all
    ```

    b. Remove the **openshift_schedulable** parameter and its value from the entry for each master host in the Ansible inventory file.
    During upgrade, if a value is not provided for the **openshift_schedulable** parameter, it is set to **true**. If a value is provided for **openshift_schedulable**, it is not changed.

    In previous versions of OpenShift Container Platform, master hosts were marked unschedulable by default by the installer, meaning that no pods could be placed on the hosts. Starting with OpenShift Container Platform 3.9, however, masters must be marked schedulable.

    During upgrade, masters are automatically labeled with the **master** node role, and non-master, non-infrastructure nodes are labeled with the **compute** node role.

    See the OpenShift Container Platform 3.9 Release Notes on the these notable technical changes and ensure that you understand their impact on your cluster:

    - Masters Marked as Schedulable Nodes by Default

    - Default Node Selector Set By Default and Automatic Node Labeling

    c. You must disable swap memory in your cluster before upgrading to OpenShift Container Platform 3.9, otherwise the upgrade will fail. Whether swap memory was enabled using **openshift_disable_swap=false** in your Ansible inventory file or enabled manually per host, see Disabling Swap Memory in the Cluster Administration guide to disable it on each host.

3. For any upgrade path, always ensure that you have the latest version of the **atomic-openshift-utils** package on each RHEL 7 system, which also updates the **openshift-ansible-*** packages:

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

4. If you have applied manual configuration changes to your master or node configuration files since your last Ansible playbook run (whether that was initial installation or your most recent cluster upgrade), and you have not yet made the equivalent changes to your inventory file, review Configuring Ansible Inventory Files. For any variables that are relevant to the manual changes you made, apply the equivalent appropriate changes to your inventory files before running the upgrade. Otherwise, your manual changes may be overwritten by default values during the upgrade, which could cause pods to not run properly or other cluster stability issues. In particular, if you made any changes to **admissionConfig** settings in your master

configuration files, review the **openshift_master_admission_plugin_config** variable in Configuring Ansible Inventory Files. Failure to do so could cause pods to get stuck in **Pending** state if you had **ClusterResourceOverride** settings manually configured previously (as described in Configuring Masters for Overcommitment).

After satisfying these steps, you can review the following sections for more information on how the upgrade process works and make decisions on additional upgrade customization options if you so choose. When you are prepared to run the upgrade, you can continue to Upgrading to the Latest OpenShift Container Platform 3.9 Release.

2.2.1. Upgrading the Control Plane and Nodes in Separate Phases

An OpenShift Container Platform cluster can be upgraded in one or more phases. You can choose whether to upgrade all hosts in one phase by running a single Ansible playbook, or upgrade the *control plane* (master components) and nodes in multiple phases using separate playbooks.

Instructions on the full upgrade process and when to call these playbooks are described in Upgrading to the Latest OpenShift Container Platform 3.9 Release.

NOTE

If your OpenShift Container Platform cluster uses GlusterFS pods, you must perform the upgrade in multiple phases. See Special Considerations When Using Containerized GlusterFS for details on how to upgrade with GlusterFS.

When upgrading in separate phases, the control plane phase includes upgrading:

- master components
- node services running on masters
- Docker running on masters
- Docker running on any stand-alone etcd hosts

When upgrading only the nodes, the control plane must already be upgraded. The node phase includes upgrading:

- node services running on stand-alone nodes
- Docker running on stand-alone nodes

NOTE

Nodes running master components are not included during the node upgrade phase, even though they have node services and Docker running on them. Instead, they are upgraded as part of the control plane upgrade phase. This ensures node services and Docker on masters are not upgraded twice (once during the control plane phase and again during the node phase).

2.2.2. Customizing Node Upgrades

Whether upgrading in a single or multiple phases, you can customize how the node portion of the upgrade progresses by passing certain Ansible variables to an upgrade playbook using the **-e** option.

NOTE

Instructions on the full upgrade process and when to call these playbooks are described in Upgrading to the Latest OpenShift Container Platform 3.7 Release .

The **openshift_upgrade_nodes_serial** variable can be set to an integer or percentage to control how many node hosts are upgraded at the same time. The default is **1**, upgrading nodes one at a time.

For example, to upgrade 20 percent of the total number of detected nodes at a time:

```
$ ansible-playbook -i <path/to/inventory/file> \
    </path/to/upgrade/playbook> \
    -e openshift_upgrade_nodes_serial="20%"
```

The **openshift_upgrade_nodes_label** variable allows you to specify that only nodes with a certain label are upgraded. This can also be combined with the **openshift_upgrade_nodes_serial** variable.

For example, to only upgrade nodes in the **group1** region, two at a time:

```
$ ansible-playbook -i <path/to/inventory/file> \
    </path/to/upgrade/playbook> \
    -e openshift_upgrade_nodes_serial="2" \
    -e openshift_upgrade_nodes_label="region=group1"
```

NOTE

See Managing Nodes for more on node labels.

The **openshift_upgrade_nodes_max_fail_percentage** variable allows you to specify how many nodes may fail in each batch. The percentage of failure must exceed your value before the playbook aborts the upgrade.

The **openshift_upgrade_nodes_drain_timeout** variable allows you to specify the length of time to wait before giving up.

In this example, 10 nodes are upgraded at a time, the upgrade will abort if more than 20 percent of the nodes fail, and there is a 600-second wait to drain the node:

```
$ ansible-playbook -i <path/to/inventory/file> \
    </path/to/upgrade/playbook> \
    -e openshift_upgrade_nodes_serial=10 \
    -e openshift_upgrade_nodes_max_fail_percentage=20 \
    -e openshift_upgrade_nodes_drain_timeout=600
```

2.2.3. Customizing Upgrades With Ansible Hooks

When upgrading OpenShift Container Platform, you can execute custom tasks during specific operations through a system called *hooks*. Hooks allow cluster administrators to provide files defining tasks to execute before and/or after specific areas during upgrades. This can be very helpful to validate or modify custom infrastructure when upgrading OpenShift Container Platform.

It is important to remember that when a hook fails, the operation fails. This means a good hook can run multiple times and provide the same results. A great hook is idempotent.

2.2.3.1. Limitations

- Hooks have no defined or versioned interface. They can use internal **openshift-ansible** variables, but there is no guarantee these will remain in future releases. In the future, hooks may be versioned, giving you advance warning that your hook needs to be updated to work with the latest **openshift-ansible**.
- Hooks have no error handling, so an error in a hook will halt the upgrade process. The problem will need to be addressed and the upgrade re-run.

2.2.3.2. Using Hooks

Hooks are defined in the ***hosts*** inventory file under the **OSEv3:vars** section.

Each hook must point to a YAML file which defines Ansible tasks. This file will be used as an *include*, meaning that the file cannot be a playbook, but a set of tasks. Best practice suggests using absolute paths to the hook file to avoid any ambiguity.

Example Hook Definitions in an Inventory File

```
[OSEv3:vars]
openshift_master_upgrade_pre_hook=/usr/share/custom/pre_master.yml
openshift_master_upgrade_hook=/usr/share/custom/master.yml
openshift_master_upgrade_post_hook=/usr/share/custom/post_master.yml
```

Example *pre_master.yml* Task

```
---
# Trivial example forcing an operator to ack the start of an upgrade
# file=/usr/share/custom/pre_master.yml

- name: note the start of a master upgrade
  debug:
      msg: "Master upgrade of {{ inventory_hostname }} is about to start"

- name: require an operator agree to start an upgrade
  pause:
      prompt: "Hit enter to start the master upgrade"
```

2.2.3.3. Available Upgrade Hooks

openshift_master_upgrade_pre_hook

- Runs *before* each master is upgraded.
- This hook runs against *each master* in serial.
- If a task must run against a different host, said task must use **delegate_to** or **local_action**.

openshift_master_upgrade_hook

- Runs *after* each master is upgraded, but *before* its service or system restart.
- This hook runs against **each master** in serial.

- If a task must run against a different host, said task must use **delegate_to** or **local_action**.

openshift_master_upgrade_post_hook

- Runs *after* each master is upgraded and has had its service or system restart.
- This hook runs against *each master* in serial.
- If a task must run against a different host, said task must use **delegate_to** or **local_action**.

2.3. UPGRADING TO THE LATEST OPENSHIFT CONTAINER PLATFORM 3.9 RELEASE

To upgrade an existing OpenShift Container Platform 3.7 or 3.9 cluster to the latest 3.9 release:

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

2. Ensure the **openshift_deployment_type** parameter in your inventory file is set to **openshift-enterprise**.

3. Starting with OpenShift Container Platform 3.9, the OpenShift Container Platform web console is deployed as a pod on masters during upgrade, and the **openshift_web_console_prefix** is introduced to deploy the web console with a customized image prefix. The **template_service_broker_prefix** is updated to match other components. If you use a customized **docker-registry** for your installation instead of *registry.access.redhat.com*, you must explicitly specify **openshift_web_console_prefix** and **template_service_broker_prefix** to point to the correct image prefix during upgrade:

```
openshift_web_console_prefix=<registry_ip>:<port>/openshift3/ose-
template_service_broker_prefix=<registry_ip>:<port>/openshift3/ose-
```

4. If you 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.

5. At this point, you can choose to run the upgrade in a single or multiple phases. See Upgrading the Control Plane and Nodes in Separate Phases for more details which components are upgraded in each phase.
If your inventory file is located somewhere other than the default */etc/ansible/hosts*, add the **-i** flag to specify its location. If you previously used the **atomic-openshift-installer** command to run your installation, you can check *~/.config/openshift/hosts* for the last inventory file that was used, if needed.

    - **Option A)** Upgrade control plane and nodes in a single phase.
    Run the ***upgrade.yml*** playbook to upgrade the cluster in a single phase using one playbook; the control plane is still upgraded first, then nodes in-place:

    ```
    # ansible-playbook -i </path/to/inventory/file> \
        /usr/share/ansible/openshift-ansible/playbooks/byo/openshift-cluster/upgrades/v3_9/upgrade.yml
    ```

    - **Option B)** Upgrade the control plane and nodes in separate phases.

a. To upgrade only the control plane, run the ***upgrade_control_plane.yaml*** playbook:

```
# ansible-playbook -i </path/to/inventory/file> \
    /usr/share/ansible/openshift-ansible/playbooks/byo/openshift-
cluster/upgrades/v3_9/upgrade_control_plane.yml
```

b. To upgrade only the nodes, run the ***upgrade_nodes.yaml*** playbook:

```
# ansible-playbook -i </path/to/inventory/file> \
    /usr/share/ansible/openshift-ansible/playbooks/byo/openshift-
cluster/upgrades/v3_9/upgrade_nodes.yml \
    [-e <customized_node_upgrade_variables>] 1
```

1 See Customizing Node Upgrades for any desired **<customized_node_upgrade_variables>**.

If you are upgrading the nodes in groups as described in Customizing Node Upgrades, continue invoking the ***upgrade_nodes.yml*** playbook until all nodes have been successfully upgraded.

6. After all master and node upgrades have completed, reboot all hosts. After rebooting, if there are no additional features enabled, you can verify the upgrade. Otherwise, the next step depends on what additional features you have previously enabled.

Feature	Next Step
Aggregated Logging	Upgrade the EFK logging stack.
Cluster Metrics	Upgrade cluster metrics.

2.4. UPGRADING THE EFK LOGGING STACK

To upgrade an existing EFK logging stack deployment, you must use the provided ***/usr/share/ansible/openshift-ansible/playbooks/openshift-logging/config.yml*** Ansible playbook. This is the playbook to use if you were deploying logging for the first time on an existing cluster, but is also used to upgrade existing logging deployments.

1. If you have any Elasticsearch SearchGuard indices in the following naming format, you need to delete and reseed the indices because the naming format has changed. Elastic search might not work as expected unless you update the indices:

```
.searchguard.logging-es-*
```

a. Run the following command to delete the SearchGuard indices:

```
# oc exec -c elasticsearch <pod> -- es_util --query=.searchguard* -XDELETE
```

b. Run the following command to reseed the SearchGuard indices:

```
# for pod in $(oc get pods -l component=es -o jsonpath={.items[*].metadata.name}); do
oc exec -c elasticsearch $pod -- es_seed_acl; done
```

2. If you have not already done so, see Specifying Logging Ansible Variables in the Aggregating Container Logs topic and update your Ansible inventory file to at least set the following required variable within the **[OSEv3:vars]** section:

```
[OSEv3:vars]

openshift_logging_install_logging=true 1
openshift_logging_image_version=<tag> 2
```

1 Enables the ability to upgrade the logging stack.

2 Replace **<tag>** with **v3.9.102** for the latest version.

3. Add any other **openshift_logging_*** variables that you want to specify to override the defaults, as described in Specifying Logging Ansible Variables .

4. When you have finished updating your inventory file, follow the instructions in Deploying the EFK Stack to run the *openshift-logging/config.yml* playbook and complete the logging deployment upgrade.

NOTE

If your Fluentd DeploymentConfig and DaemonSet for the EFK components are already set with:

```
    image: <image_name>:<vX.Y>
    imagePullPolicy: IfNotPresent
```

The latest version <image_name> might not be pulled if there is already one with the same <image_name:vX.Y> stored locally on the node where the pod is being re-deployed. If so, manually change the DeploymentConfig and DaemonSet to **imagePullPolicy: Always** to make sure it is re-pulled.

2.5. UPGRADING CLUSTER METRICS

To upgrade an existing cluster metrics deployment, you must use the provided */usr/share/ansible/openshift-ansible/playbooks/openshift-metrics/config.yml* Ansible playbook. This is the playbook to use if you were deploying metrics for the first time on an existing cluster, but is also used to upgrade existing metrics deployments.

1. If you have not already done so, see Specifying Metrics Ansible Variables in the Enabling Cluster Metrics topic and update your Ansible inventory file to at least set the following required variables within the **[OSEv3:vars]** section:

```
[OSEv3:vars]

openshift_metrics_install_metrics=true 1
openshift_metrics_image_version=<tag> 2
openshift_metrics_hawkular_hostname=<fqdn> 3
openshift_metrics_cassandra_storage_type=(emptydir|pv|dynamic) 4
```

1 Enables the ability to upgrade the metrics deployment.

2 Replace **<tag>** with **v3.9.102** for the latest version.

3 Used for the Hawkular Metrics route. Should correspond to a fully qualified domain name.

4 Choose a type that is consistent with the previous deployment.

2. Add any other **openshift_metrics_*** variables that you want to specify to override the defaults, as described in Specifying Metrics Ansible Variables .

3. When you have finished updating your inventory file, follow the instructions in Deploying the Metrics Deployment to run the ***openshift-metrics/config.yml*** playbook and complete the metrics deployment upgrade.

2.6. SPECIAL CONSIDERATIONS FOR LARGE-SCALE UPGRADES

For large-scale cluster upgrades, which involve at least 10 worker nodes and thousands of projects and pods, the API object storage migration should be performed prior to running the upgrade playbooks, and then again after the upgrade has successfully completed. Otherwise, the upgrade process will fail.

Refer to the **Running the pre- and post- API server model object migration outside of the upgrade window** section of the Recommendations for large-scale OpenShift upgrades for further guidance.

2.7. SPECIAL CONSIDERATIONS FOR MIXED ENVIRONMENTS

Before you upgrade a mixed environment, such as one with Red Hat Enterprise Linux (RHEL) and RHEL Atomic Host, set values in the inventory file for both the **openshift_pkg_version** and **openshift_image_tag** parameters. Setting these values ensures that all nodes in your cluster run the same version of OpenShift Container Platform.

For example, to upgrade from OpenShift Container Platform 3.7 to OpenShift Container Platform 3.9, set the following parameters and values:

```
openshift_pkg_version=-3.9.74
openshift_image_tag=v3.9.74
```

NOTE

These parameters can also be present in other, non-mixed, environments.

2.8. SPECIAL CONSIDERATIONS WHEN USING CONTAINERIZED GLUSTERFS

When upgrading OpenShift Container Platform, you must upgrade the set of nodes where GlusterFS pods are running.

Use care when upgrading these nodes, as **drain** and **unschedule** will not terminate and evacuate the GlusterFS pods because they are running as part of a daemonset.

There is also the potential for someone to run an upgrade on multiple nodes at the same time. If this occurs, it would lead to data availability issues if more than one node was hosting GlusterFS pods.

Even if a serial upgrade is running, there is no guarantee sufficient time will be given for GlusterFS to complete all of its healing operations before GlusterFS on the next node is terminated. This could leave the cluster in a bad or unknown state. Therefore, the following procedure is recommended.

1. Upgrade the control plane (the master nodes and etcd nodes).
2. Upgrade standard **infra** nodes (router, registry, logging, and metrics).

   NOTE

   If any of the nodes in those groups are running GlusterFS, perform step 4 of this procedure at the same time. GlusterFS nodes must be upgraded along with other nodes in their class (**app** versus **infra**), one at a time.

3. Upgrade standard nodes running application containers.

   NOTE

   If any of the nodes in those groups are running GlusterFS, perform step 4 of this procedure at the same time. GlusterFS nodes must be upgraded along with other nodes in their class (**app** versus **infra**), one at a time.

4. Upgrade the OpenShift Container Platform nodes running GlusterFS one at a time.
   a. Run **oc get daemonset** to verify the label found under **NODE-SELECTOR**. The default value is **storagenode=glusterfs**.
   b. Add a label (for example, **type=upgrade**) to the node you want to upgrade.
   c. Using the ***upgrade_nodes.yml*** playbook on the node where you terminated GlusterFS, run **-e openshift_upgrade_nodes_label="type=upgrade"**.

      NOTE

      The ***upgrade_nodes.yml*** playbook is located here:

      ```
      # ansible-playbook -i </path/to/inventory/file> \
          /usr/share/ansible/openshift-ansible/playbooks/byo/openshift-cluster/upgrades/v3_9/upgrade_nodes.yml \
          [-e <customized_node_upgrade_variables>] 1
      ```

      1 See Customizing Node Upgrades for any desired **<customized_node_upgrade_variables>**.

   d. Wait for the GlusterFS pod to respawn and appear.
   e. **oc rsh** into the pod and verify all volumes are healed:

      ```
      $ oc rsh <GlusterFS_pod_name>
      $ for vol in `gluster volume list`; do gluster volume heal $vol info; done
      ```

      Ensure all of the volumes are healed and there are no outstanding tasks. The **heal info** command lists all pending entries for a given volume's heal process. A volume is considered healed when **Number of entries** for that volume is **0**.

f. Remove the upgrade label and go to the next GlusterFS node:

```
$ oc label node <node_name> type-
```

2.9. SPECIAL CONSIDERATIONS WHEN USING GCEPD

Because the default gcePD storage provider uses an RWO (Read-Write Only) access mode, you cannot perform a rolling upgrade on the registry or scale the registry to multiple pods. Therefore, when upgrading OpenShift Container Platform, you must specify the following environment variables in your Ansible inventory file:

```
[OSEv3:vars]

openshift_hosted_registry_storage_provider=gcs
openshift_hosted_registry_storage_gcs_bucket=bucket01
openshift_hosted_registry_storage_gcs_keyfile=test.key
openshift_hosted_registry_storage_gcs_rootdirectory=/registry
```

2.10. VERIFYING THE UPGRADE

Ensure that the:

- cluster is healthy,
- services (master, node, and etcd) are running well,
- the OpenShift Container Platform, **docker-registry**, and router versions are correct,
- the original applications are still available and the new application can be created, and
- running **oc adm diagnostics** produces no errors.

To verify the upgrade:

1. Check that all nodes are marked as **Ready**:

```
# oc get nodes
NAME                 STATUS    ROLES     AGE       VERSION
master.example.com     Ready     master    7h        v1.9.1+a0ce1bc657
node1.example.com      Ready     compute   7h        v1.9.1+a0ce1bc657
node2.example.com      Ready     compute   7h        v1.9.1+a0ce1bc657
```

2. Verify that you are running the expected versions of the **docker-registry** and **router** images, if deployed. Replace **<tag>** with **v3.9.102** for the latest version.

```
# oc get -n default dc/docker-registry -o json | grep \"image\"
    "image": "openshift3/ose-docker-registry:<tag>",
# oc get -n default dc/router -o json | grep \"image\"
    "image": "openshift3/ose-haproxy-router:<tag>",
```

3. Use the diagnostics tool on the master to look for common issues:

```
# oc adm diagnostics
```

```
...
[Note] Summary of diagnostics execution:
[Note] Completed with no errors or warnings seen.
```

CHAPTER 3. PERFORMING MANUAL IN-PLACE CLUSTER UPGRADES

3.1. OVERVIEW

IMPORTANT

As of OpenShift Container Platform 3.9, manual upgrades are not supported. In a future release, this process will be removed.

As an alternative to performing an automated upgrade, you can manually upgrade your OpenShift Container Platform cluster. To manually upgrade without disruption, it is important to upgrade each component as documented in this topic.

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.

IMPORTANT

Ensure that you have met all prerequisites before proceeding with an upgrade. Failure to do so can result in a failed upgrade.

If you are using GlusterFS, see Special Considerations When Using Containerized GlusterFS before proceeding.

The day before the upgrade, validate OpenShift Container Platform storage migration to ensure potential issues are resolved prior to the outage window:

```
$ oc adm migrate storage --include=* --loglevel=2 --confirm --config
/etc/origin/master/admin.kubeconfig
```

3.2. PREPARING FOR A MANUAL UPGRADE

NOTE

Before upgrading your cluster to OpenShift Container Platform 3.9, the cluster must be already upgraded to the latest asynchronous release of version 3.7 . If your cluster is at a version earlier than 3.7, you must first upgrade incrementally (e.g., 3.5 to 3.6, then 3.6 to 3.7).

To prepare for a manual upgrade, follow these steps:

1. Follow the steps in Verifying the Upgrade to verify the cluster’s health (if you are upgrading from 3.7 to 3.9, check the command outputs for the relevant 3.7 version instead of the stated 3.9). This will confirm that nodes are in the **Ready** state, running the expected starting version, and will ensure that there are no diagnostic errors or warnings.

2. Pull the latest subscription data from Red Hat Subscription Manager (RHSM):

   ```
   # subscription-manager refresh
   ```

3. If you are upgrading from OpenShift Container Platform 3.7 to 3.9:

    a. Manually disable the 3.7 repository and enable both the 3.8 and 3.9 repositories on each master and node host. You must also enable the **rhel-7-server-ansible-2.4-rpms** repository, which is a new requirement starting with OpenShift Container Platform 3.9:

    ```
    # subscription-manager repos --disable="rhel-7-server-ose-3.7-rpms" \
        --enable="rhel-7-server-ose-3.8-rpms" \
        --enable="rhel-7-server-ose-3.9-rpms" \
        --enable="rhel-7-server-extras-rpms" \
        --enable="rhel-7-server-ansible-2.4-rpms" \
        --enable="rhel-7-fast-datapath-rpms"
    # yum clean all
    ```

    b. You must disable swap memory in your cluster before upgrading to OpenShift Container Platform 3.9, otherwise the upgrade will fail. Whether swap memory was enabled using **openshift_disable_swap=false** in your Ansible inventory file or enabled manually per host, see Disabling Swap Memory in the Cluster Administration guide to disable it on each host.

4. Install or update to the latest available version of the **atomic-openshift-utils** package on each RHEL 7 system, which provides files that will be used in later sections:

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

5. 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 Container Platform 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.

6. Create an **etcd** backup on each etcd host.

7. For any upgrade path, ensure that you are running the latest kernel on each RHEL 7 system:

    ```
    # yum update kernel
    ```

3.3. UPGRADING MASTER COMPONENTS

Before upgrading any stand-alone nodes, upgrade the master components (which provide the *control plane* for the cluster).

1. Run the following command on each master to remove the **atomic-openshift** packages from the list of yum excludes on the host:

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

2. Upgrade **etcd** on all master hosts and any external etcd hosts.

    a. For RHEL 7 systems using the RPM-based method:

        i. Upgrade the **etcd** package:

```
# yum update etcd
```

ii. Restart the **etcd** service and review the logs to ensure it restarts successfully:

```
# systemctl restart etcd
# journalctl -r -u etcd
```

b. For RHEL Atomic Host 7 systems and RHEL 7 systems using the containerized method:

i. Pull the latest **rhel7/etcd** image:

```
# docker pull registry.access.redhat.com/rhel7/etcd
```

ii. Restart the **etcd_container** service and review the logs to ensure it restarts successfully:

```
# systemctl restart etcd_container
# journalctl -r -u etcd_container
```

3. Configure Dnsmasq. You must perform the following steps to configure dnsmasq as part of the upgrade.

a. Create a ***/etc/origin/node/node-dnsmasq.conf*** node configuration file.

```
server=/in-addr.arpa/127.0.0.1
server=/cluster.local/127.0.0.1
```

b. Edit the ***/etc/dnsmasq.d/origin-dns.conf*** file as follows:

```
no-resolv
domain-needed
no-negcache
max-cache-ttl=1
enable-dbus
dns-forward-max=5000
cache-size=5000
bind-dynamic
except-interface=lo
```

c. Edit the ***/etc/dnsmasq.d/origin-upstream-dns.conf*** file as follows:

```
server=<dns_server1_ip_address>
server=<dns_server2_ip_address>
```

d. Edit the ***/etc/origin/node/node-config.yaml*** as follows:

```
dnsBindAddress: 127.0.0.1:53
dnsRecursiveResolvConf: /etc/origin/node/resolv.conf
dnsDomain: cluster.local
dnsIP: <node_ip_address> 1
```

1 This is the IP address of the node host.

e. Update the ***/etc/systemd/system/atomic-openshift-node.service*** node systemd unit file:

```
[Unit]
Description=OpenShift Node
After=docker.service
Wants=openvswitch.service
After=ovsdb-server.service
After=ovs-vswitchd.service
Wants=docker.service
Documentation=https://github.com/openshift/origin
Requires=dnsmasq.service
After=dnsmasq.service

[Service]
Type=notify
EnvironmentFile=/etc/sysconfig/atomic-openshift-node
Environment=GOTRACEBACK=crash
ExecStartPre=/usr/bin/cp /etc/origin/node/node-dnsmasq.conf /etc/dnsmasq.d/
ExecStartPre=/usr/bin/dbus-send --system --dest=uk.org.thekelleys.dnsmasq
/uk/org/thekelleys/dnsmasq uk.org.thekelleys.SetDomainServers array:string:/in-
addr.arpa/127.0.0.1,/cluster.local/127.0.0.1
ExecStopPost=/usr/bin/rm /etc/dnsmasq.d/node-dnsmasq.conf
ExecStopPost=/usr/bin/dbus-send --system --dest=uk.org.thekelleys.dnsmasq
/uk/org/thekelleys/dnsmasq uk.org.thekelleys.SetDomainServers array:string:
ExecStart=/usr/bin/openshift start node --config=${CONFIG_FILE} $OPTIONS
LimitNOFILE=65536
LimitCORE=infinity
WorkingDirectory=/var/lib/origin/
SyslogIdentifier=atomic-openshift-node
Restart=always
RestartSec=5s
TimeoutStartSec=300
OOMScoreAdjust=-999

[Install]
WantedBy=multi-user.target
```

f. Reload systemd and restart node service.

```
# systemctl daemon-reload
# systemctl restart atomic-openshift-node dnsmaq
```

g. Restart dnsmasq:

```
# systemctl restart dnsmasq
```

4. On each master host, upgrade the **atomic-openshift** packages or related images.

a. For masters using the RPM-based method on a RHEL 7 system, upgrade all installed **atomic-openshift** packages and the **openvswitch** package:

```
# yum upgrade atomic-openshift\* openvswitch
```

b. For masters using the containerized method on a RHEL 7 or RHEL Atomic Host 7 system, set the **IMAGE_VERSION** parameter to the version you are upgrading to in the following files:

   - */etc/sysconfig/atomic-openshift-master-controllers*
   - */etc/sysconfig/atomic-openshift-master-api*
   - */etc/sysconfig/atomic-openshift-node*
   - */etc/sysconfig/atomic-openshift-openvswitch*

   For example:

   ```
   IMAGE_VERSION=<tag>
   ```

   Replace **<tag>** with **v3.9.102** for the latest version.

5. Modify host labels and parameters.
   In previous versions of OpenShift Container Platform, master hosts were marked unschedulable by default by the installer, meaning that no pods could be placed on the hosts. Starting with OpenShift Container Platform 3.9, however, the web console is removed from the master services and deployed as a pod on the masters, requiring that masters be marked schedulable. This change also requires hosts to have *node role* labels, either **master** or **compute**.

NOTE

You can see the OpenShift Container Platform 3.9 Release Notes for more information on how these notable technical changes relate in the context of supported automated upgrades:

- Masters Marked as Schedulable Nodes by Default
- Default Node Selector Set By Default and Automatic Node Labeling

If you are upgrading from OpenShift Container Platform 3.7 to 3.9:

a. Add the following label to each master host:

   ```
   $ oc label node <hostname> node-role.kubernetes.io/master=true
   ```

   This assigns them the **master** node role.

b. Add the following label to each non-master, non-infrastructure (e.g., **region=infra** labeled) host:

   ```
   $ oc label node <hostname> node-role.kubernetes.io/compute=true
   ```

   This assigns them the **compute** node role.

c. The default node selector must also now be set to ensure that only the web console pods will be scheduled to the masters. If you already have this set to a custom value via **projectConfig.defaultNodeSelector** in the */etc/origin/master/master-config.yaml* file, you can skip this step. If you have not previously set it, set it to the following:

```
projectConfig:
  defaultNodeSelector: node-role.kubernetes.io/compute=true
```

1. Restart the master service(s) on each master and review logs to ensure they restart successfully.

   ```
   # systemctl restart atomic-openshift-master-controllers
   # systemctl restart atomic-openshift-master-api
   # journalctl -r -u atomic-openshift-master-controllers
   # journalctl -r -u atomic-openshift-master-api
   ```

2. Because masters also have node components running on them in order to be configured as part of the OpenShift SDN, restart the **atomic-openshift-node** and **openvswitch** services:

   ```
   # systemctl restart openvswitch
   # systemctl restart atomic-openshift-node
   # journalctl -r -u openvswitch
   # journalctl -r -u atomic-openshift-node
   ```

3. If you are upgrading from OpenShift Container Platform 3.7 to 3.9, run the following for each master host to mark them schedulable:

   ```
   $ oc adm manage-node <hostname> --schedulable=true
   ```

4. If you are performing a cluster upgrade that requires updating Docker to version 1.13, you must also perform the following steps if you are not already on Docker 1.13:

d. Upgrade the **docker** package.

   i. For RHEL 7 systems:

   ```
   # yum update docker
   ```

   Then, restart the **docker** service and review the logs to ensure it restarts successfully:

   ```
   # systemctl restart docker
   # journalctl -r -u docker
   ```

   ii. For RHEL Atomic Host 7 systems, upgrade to the latest Atomic tree if one is available:

   ```
   # atomic host upgrade
   ```

e. After the upgrade is completed and prepared for the next boot, reboot the host and ensure the **docker** service starts successfully:

```
# systemctl reboot
# journalctl -r -u docker
```

f. Remove the following file, which is no longer required:

```
# rm /etc/systemd/system/docker.service.d/docker-sdn-ovs.conf
```

1. Run the following command on each master to add the **atomic-openshift** packages back to the list of yum excludes on the host:

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

NOTE

During the cluster upgrade, it can sometimes be useful to take a master out of rotation since some DNS client libraries will not properly to the other masters for cluster DNS. In addition to stopping the master and controller services, you can remove the EndPoint from the Kubernetes service's **subsets.addresses**.

```
$ oc edit ep/kubernetes -n default
```

When the master is restarted, the Kubernetes service will be automatically updated.

3.4. UPDATING POLICY DEFINITIONS

During a cluster upgrade, and on every restart of any master, the default cluster roles are automatically reconciled to restore any missing permissions.

1. If you customized default cluster roles and want to ensure a role reconciliation does not modify them, protect each role from reconciliation:

   ```
   $ oc annotate clusterrole.rbac <role_name> --overwrite
   rbac.authorization.kubernetes.io/autoupdate=false
   ```

   WARNING

   You must manually update the roles that contain this setting to include any new or required permissions after upgrading.

2. Generate a default bootstrap policy template file:

   ```
   $ oc adm create-bootstrap-policy-file --filename=policy.json
   ```

   NOTE

   The contents of the file vary based on the OpenShift Container Platform version, but the file contains only the default policies.

3. Update the ***policy.json*** file to include any cluster role customizations.

4. Use the policy file to automatically reconcile roles and role bindings that are not reconcile protected:

   ```
   $ oc process --local -f policy.json | oc auth reconcile -f policy.json
   ```

5. Reconcile security context constraints:

```
# oc adm policy reconcile-sccs \
    --additive-only=true \
    --confirm
```

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.

NOTE

You can alternatively use the blue-green deployment method at this point to create a parallel environment for new nodes instead of upgrading them in place.

One at at time for each node that is not also a master, you must disable scheduling and evacuate its pods to other nodes, then upgrade packages and restart services.

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:

```
# oc adm manage-node <node> --schedulable=false
```

3. Evacuate pods on the node to other nodes:

IMPORTANT

The **--force** option deletes any pods that are not backed by a replication controller.

```
# oc adm drain <node> --force --delete-local-data --ignore-daemonsets
```

4. Upgrade the node component packages or related images.

    a. For nodes using the RPM-based method on a RHEL 7 system, upgrade all installed **atomic-openshift** packages and the **openvswitch** package:

    ```
    # yum upgrade atomic-openshift\* openvswitch
    ```

    b. For nodes using the containerized method on a RHEL 7 or RHEL Atomic Host 7 system, set the **IMAGE_VERSION** parameter in the ***/etc/sysconfig/atomic-openshift-node*** and ***/etc/sysconfig/openvswitch*** files to the version you are upgrading to. For example:

    ```
    IMAGE_VERSION=<tag>
    ```

Replace **<tag>** with **v3.9.102** for the latest version.

5. Restart the **atomic-openshift-node** and **openvswitch** services and review the logs to ensure they restart successfully:

```
# systemctl restart openvswitch
# systemctl restart atomic-openshift-node
# journalctl -r -u atomic-openshift-node
# journalctl -r -u openvswitch
```

6. If you are performing a cluster upgrade that requires updating Docker to version 1.13, you must also perform the following steps if you are not already on Docker 1.13:

   a. Upgrade the **docker** package.

      i. For RHEL 7 systems:

```
# yum update docker
```

      Then, restart the **docker** service and review the logs to ensure it restarts successfully:

```
# systemctl restart docker
# journalctl -r -u docker
```

      After Docker is restarted, restart the **atomic-openshift-node** service again and review the logs to ensure it restarts successfully:

```
# systemctl restart atomic-openshift-node
# journalctl -r -u atomic-openshift-node
```

      ii. For RHEL Atomic Host 7 systems, upgrade to the latest Atomic tree if one is available:

```
# atomic host upgrade
```

      After the upgrade is completed and prepared for the next boot, reboot the host and ensure the **docker** service starts successfully:

```
# systemctl reboot
# journalctl -r -u docker
```

   b. Remove the following file, which is no longer required:

```
# rm /etc/systemd/system/docker.service.d/docker-sdn-ovs.conf
```

7. Re-enable scheduling for the node:

```
# oc adm manage-node <node> --schedulable
```

8. Run the following command on each node to add the **atomic-openshift** packages back to the list of yum excludes on the host:

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

9. Repeat the previous steps on the next node, and continue repeating these steps until all nodes have been upgraded.

10. After all nodes have been upgraded, as a user with **cluster-admin** privileges, verify that all nodes are showing as **Ready**:

```
# oc get nodes
NAME                 STATUS    ROLES     AGE       VERSION
master.example.com   Ready     master    7h        v1.9.1+a0ce1bc657
node1.example.com    Ready     compute   7h        v1.9.1+a0ce1bc657
node2.example.com    Ready     compute   7h        v1.9.1+a0ce1bc657
```

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:

```
# oc edit dc/router
```

Apply the following changes:

```
...
spec:
 template:
    spec:
      containers:
      - env:
        ...
        image: registry.access.redhat.com/openshift3/ose-haproxy-router:<tag> 1
        imagePullPolicy: IfNotPresent
        ...
```

1 Adjust **<tag>** to match the version you are upgrading to (use **v3.9.102** for the latest version).

You should see one router pod updated and then the next.

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. Storage for the Registry details how to configure persistent storage for the registry.

Edit your registry's deployment configuration:

```
# oc edit dc/docker-registry
```

Apply the following changes:

```
...
```

```
spec:
 template:
   spec:
     containers:
     - env:
       ...
       image: registry.access.redhat.com/openshift3/ose-docker-registry:<tag> 1
       imagePullPolicy: IfNotPresent
       ...
```

1 Adjust **<tag>** to match the version you are upgrading to (use **v3.9.102** for the latest version).

If the registry console is deployed, edit its deployment configuration:

```
# oc edit dc/registry-console
```

Apply the following changes:

```
...
spec:
 template:
   spec:
     containers:
     - env:
       ...
       image: registry.access.redhat.com/openshift3/registry-console:v3.9
       imagePullPolicy: IfNotPresent
       ...
```

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.

3.8. UPDATING THE DEFAULT IMAGE STREAMS AND TEMPLATES

By default, the advanced installation method automatically creates 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.

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:

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

To persist ***/usr/share/openshift/examples/*** on the first master:

```
$ scp -R /usr/share/ansible/openshift-ansible/roles/openshift_examples/files/examples/v3.9/*
user@master1:/usr/share/openshift/examples/
```

To persist ***/usr/share/openshift/examples/*** on all masters:

```
$ mkdir /usr/share/openshift/examples
$ scp -R /usr/share/ansible/openshift-ansible/roles/openshift_examples/files/examples/v3.9/*
user@masterx:/usr/share/openshift/examples
```

The **openshift-ansible-roles** package provides the latest example JSON files.

1. After a manual upgrade, get the latest templates from **openshift-ansible-roles**:

   ```
   # rpm -ql openshift-ansible-roles | grep examples | grep v3.9
   ```

   In this example, ***/usr/share/ansible/openshift-ansible/roles/openshift_examples/files/examples/v3.9/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.

2. Install **atomic-openshift-utils** and its dependencies to install the new content into ***/usr/share/ansible/openshift-ansible/roles/openshift_examples/files/examples/v3.9/***.:

   ```
   $ oc create -n openshift -f  /usr/share/ansible/openshift-
   ansible/roles/openshift_examples/files/examples/v3.9/image-streams/image-streams-
   rhel7.json
   $ oc create -n openshift -f  /usr/share/ansible/openshift-
   ansible/roles/openshift_examples/files/examples/v3.9/image-
   streams/dotnet_imagestreams.json
   $ oc replace -n openshift -f  /usr/share/ansible/openshift-
   ansible/roles/openshift_examples/files/examples/v3.9/image-streams/image-streams-
   rhel7.json
   $ oc replace -n openshift -f  /usr/share/ansible/openshift-
   ansible/roles/openshift_examples/files/examples/v3.9/image-
   streams/dotnet_imagestreams.json
   ```

3. Update the templates:

   ```
   $ oc create -n openshift -f /usr/share/ansible/openshift-
   ansible/roles/openshift_examples/files/examples/v3.9/quickstart-templates/
   $ oc create -n openshift -f /usr/share/ansible/openshift-
   ansible/roles/openshift_examples/files/examples/v3.9/db-templates/
   $ oc create -n openshift -f /usr/share/ansible/openshift-
   ansible/roles/openshift_examples/files/examples/v3.9/infrastructure-templates/
   $ oc create -n openshift -f /usr/share/ansible/openshift-
   ansible/roles/openshift_examples/files/examples/v3.9/xpaas-templates/
   ```

```
$ oc create -n openshift -f /usr/share/ansible/openshift-
ansible/roles/openshift_examples/files/examples/v3.9/xpaas-streams/
$ oc replace -n openshift -f /usr/share/ansible/openshift-
ansible/roles/openshift_examples/files/examples/v3.9/quickstart-templates/
$ oc replace -n openshift -f /usr/share/ansible/openshift-
ansible/roles/openshift_examples/files/examples/v3.9/db-templates/
$ oc replace -n openshift -f /usr/share/ansible/openshift-
ansible/roles/openshift_examples/files/examples/v3.9/infrastructure-templates/
$ oc replace -n openshift -f /usr/share/ansible/openshift-
ansible/roles/openshift_examples/files/examples/v3.9/xpaas-templates/
$ oc replace -n openshift -f /usr/share/ansible/openshift-
ansible/roles/openshift_examples/files/examples/v3.9/xpaas-streams/
```

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/v3.9/quickstart-templates/
Error from server: error when creating "/usr/share/ansible/openshift-
ansible/roles/openshift_examples/files/examples/v3.9/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/v3.9/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/v3.9/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/v3.9/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/v3.9/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/***.

3.9. 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,v3.1.1.6    16
hours ago
mysql    registry.access.redhat.com/openshift3/mysql-55-rhel7        5.5,latest,v3.1.1.6    16 hours
ago
nodejs   registry.access.redhat.com/openshift3/nodejs-010-rhel7      0.10,latest,v3.1.1.6   16 hours
ago
...
```

Update each image stream one at a time:

```
# oc import-image -n openshift nodejs
The import completed successfully.

Name:   nodejs
Created:  10 seconds ago
Labels:   <none>
Annotations:  openshift.io/image.dockerRepositoryCheck=2016-07-05T19:20:30Z
Docker Pull Spec: 172.30.204.22:5000/openshift/nodejs

Tag Spec        Created  PullSpec      Image
latest 4        9 seconds ago registry.access.redhat.com/rhscl/nodejs-4-rhel7:latest
570ad8ed927fd5c2c9554ef4d9534cef808dfa05df31ec491c0969c3bd372b05
4 registry.access.redhat.com/rhscl/nodejs-4-rhel7:latest  9 seconds ago <same>
570ad8ed927fd5c2c9554ef4d9534cef808dfa05df31ec491c0969c3bd372b05
0.10 registry.access.redhat.com/openshift3/nodejs-010-rhel7:latest 9 seconds ago <same>
a1ef33be788a28ec2bdd48a9a5d174ebcfbe11c8e986d2996b77f5bccaaa4774
```

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>**.

3.10. UPGRADING THE WEB CONSOLE

NOTE

Manual upgrade steps for the web console are not available.

Starting with OpenShift Container Platform 3.9, the web console is deployed as a separate component rather than as part of the master. The console will be installed by default as part of an automated upgrade unless you set the **openshift_web_console_install** variable to **false**.

Alternatively, you can install the web console separately if needed, for example as part of a manual upgrade:

1. See the following section in the Advanced Installation topic and update your inventory file accordingly.

2. Run the following playbook:

```
# ansible-playbook -i </path/to/inventory/file> \
    /usr/share/ansible/openshift-ansible/playbooks/openshift-web-console/config.yml
```

CAUTION

The way the web console loads extensions has changed in OpenShift Container Platform 3.9. If you are upgrading from an earlier version, you must host your extension scripts and stylesheets at a URL instead of on the master file system. See the Loading Extension Scripts and Stylesheets topic for details.

3.11. UPGRADING THE SERVICE CATALOG

NOTE

Manual upgrade steps for the service catalog and service brokers are not available.

To upgrade the service catalog:

1. See the following three sections in the Advanced Installation topic and update your inventory file accordingly:
   - Configuring the Service Catalog
   - Configuring the OpenShift Ansible Broker
   - Configuring the Template Service Broker
2. Run the following playbook:

```
# ansible-playbook -i </path/to/inventory/file> \
    /usr/share/ansible/openshift-ansible/playbooks/openshift-service-catalog/config.yml
```

3.12. UPGRADING THE EFK LOGGING STACK

NOTE

Manual upgrade steps for logging deployments are no longer available starting in OpenShift Container Platform 3.5.

To upgrade an existing EFK logging stack deployment, you must use the provided ***/usr/share/ansible/openshift-ansible/playbooks/openshift-logging/config.yml*** Ansible playbook. This is the playbook to use if you were deploying logging for the first time on an existing cluster, but is also used to upgrade existing logging deployments.

1. If you have any Elasticsearch SearchGuard indices in the following naming format, you need to delete and reseed the indices because the naming format has changed. Elastic search might not work as expected unless you update the indices:

   ```
   .searchguard.logging-es-*
   ```

   a. Run the following command to delete the SearchGuard indices:

   ```
   # oc exec -c elasticsearch <pod> -- es_util --query=.searchguard* -XDELETE
   ```

   b. Run the following command to reseed the SearchGuard indices:

   ```
   # for pod in $(oc get pods -l component=es -o jsonpath={.items[*].metadata.name}); do
   oc exec -c elasticsearch $pod -- es_seed_acl; done
   ```

2. If you have not already done so, see Specifying Logging Ansible Variables in the Aggregating Container Logs topic and update your Ansible inventory file to at least set the following required variable within the **[OSEv3:vars]** section:

   ```
   [OSEv3:vars]
   ```

```
openshift_logging_install_logging=true 1
openshift_logging_image_version=<tag> 2
```

1 Enables the ability to upgrade the logging stack.

2 Replace **<tag>** with **v3.9.102** for the latest version.

3. Add any other **openshift_logging_*** variables that you want to specify to override the defaults, as described in Specifying Logging Ansible Variables .

4. When you have finished updating your inventory file, follow the instructions in Deploying the EFK Stack to run the ***openshift-logging/config.yml*** playbook and complete the logging deployment upgrade.

NOTE

If your Fluentd DeploymentConfig and DaemonSet for the EFK components are already set with:

```
image: <image_name>:<vX.Y>
imagePullPolicy: IfNotPresent
```

The latest version <image_name> might not be pulled if there is already one with the same <image_name:vX.Y> stored locally on the node where the pod is being re-deployed. If so, manually change the DeploymentConfig and DaemonSet to **imagePullPolicy: Always** to make sure it is re-pulled.

3.13. UPGRADING CLUSTER METRICS

NOTE

Manual upgrade steps for metrics deployments are no longer available starting in OpenShift Container Platform 3.5.

To upgrade an existing cluster metrics deployment, you must use the provided ***/usr/share/ansible/openshift-ansible/playbooks/openshift-metrics/config.yml*** Ansible playbook. This is the playbook to use if you were deploying metrics for the first time on an existing cluster, but is also used to upgrade existing metrics deployments.

1. If you have not already done so, see Specifying Metrics Ansible Variables in the Enabling Cluster Metrics topic and update your Ansible inventory file to at least set the following required variables within the **[OSEv3:vars]** section:

```
[OSEv3:vars]

openshift_metrics_install_metrics=true 1
openshift_metrics_image_version=<tag> 2
openshift_metrics_hawkular_hostname=<fqdn> 3
openshift_metrics_cassandra_storage_type=(emptydir|pv|dynamic) 4
```

1 Enables the ability to upgrade the metrics deployment.

2 Replace **<tag>** with **v3.9.102** for the latest version.

3 Used for the Hawkular Metrics route. Should correspond to a fully qualified domain name.

4 Choose a type that is consistent with the previous deployment.

2. Add any other **openshift_metrics_*** variables that you want to specify to override the defaults, as described in Specifying Metrics Ansible Variables .

3. When you have finished updating your inventory file, follow the instructions in Deploying the Metrics Deployment to run the ***openshift-metrics/config.yml*** playbook and complete the metrics deployment upgrade.

3.14. ADDITIONAL MANUAL STEPS PER RELEASE

Some OpenShift Container Platform releases may have additional instructions specific to that release that must be performed to fully apply the updates across the cluster. This section will be updated over time as new asynchronous updates are released for OpenShift Container Platform 3.9.

See the OpenShift Container Platform 3.9 Release Notes to review the latest release notes.

3.15. VERIFYING THE UPGRADE

To verify the upgrade:

1. Check that all nodes are marked as **Ready**:

```
# oc get nodes
NAME                 STATUS    ROLES     AGE       VERSION
master.example.com   Ready     master    7h        v1.9.1+a0ce1bc657
node1.example.com    Ready     compute   7h        v1.9.1+a0ce1bc657
node2.example.com    Ready     compute   7h        v1.9.1+a0ce1bc657
```

2. Verify that you are running the expected versions of the **docker-registry** and **router** images, if deployed. Replace **<tag>** with **v3.9.102** for the latest version.

```
# oc get -n default dc/docker-registry -o json | grep \"image\"
    "image": "openshift3/ose-docker-registry:<tag>",
# oc get -n default dc/router -o json | grep \"image\"
    "image": "openshift3/ose-haproxy-router:<tag>",
```

3. Use the diagnostics tool on the master to look for common issues:

```
# oc adm diagnostics
...
[Note] Summary of diagnostics execution:
[Note] Completed with no errors or warnings seen.
```

CHAPTER 4. BLUE-GREEN DEPLOYMENTS

4.1. OVERVIEW

NOTE

This topic serves as an alternative approach for node host upgrades to the in-place upgrade method.

The *blue-green deployment* upgrade method follows a similar flow to the in-place method: masters and etcd servers are still upgraded first, however a parallel environment is created for new node hosts instead of upgrading them in-place.

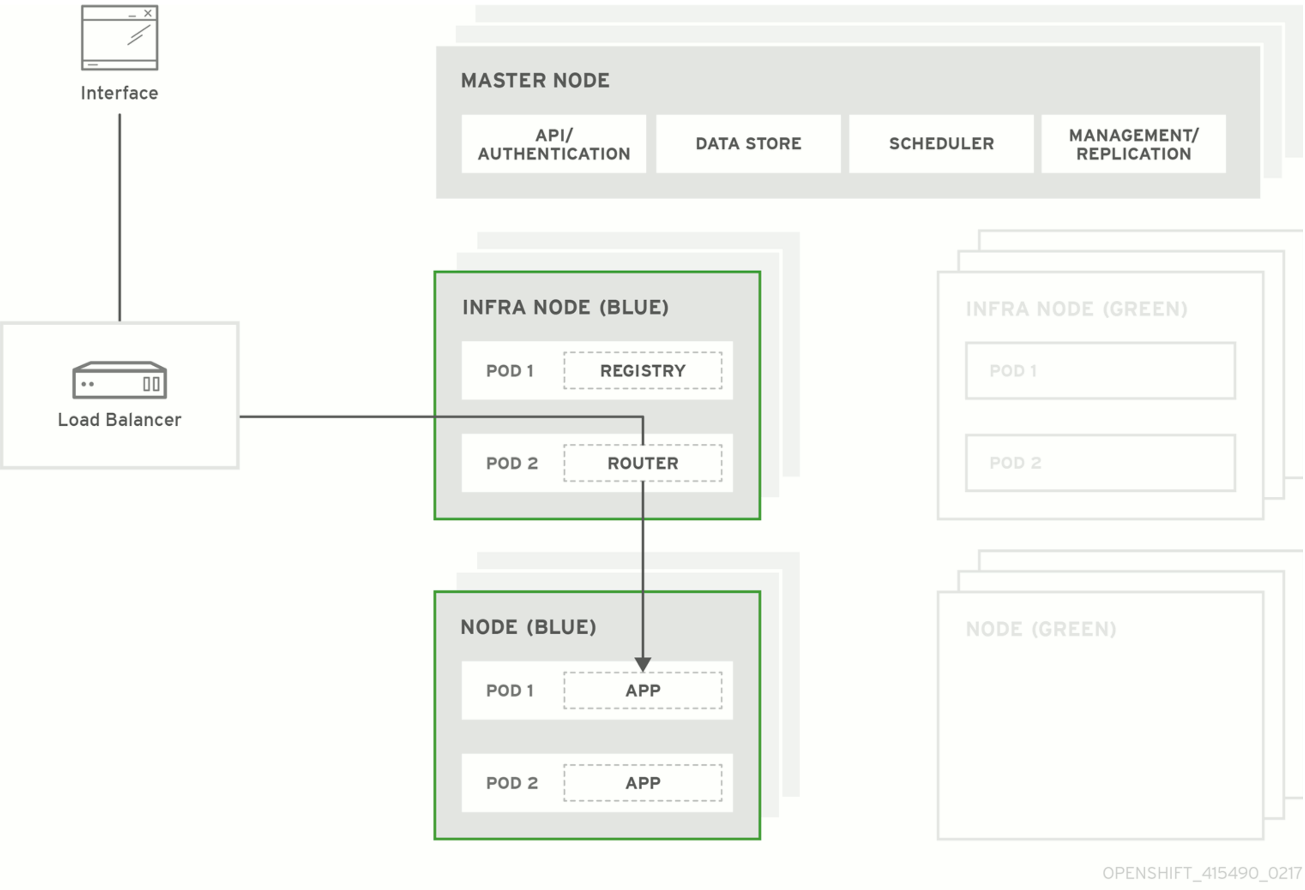
This method allows administrators to switch traffic from the old set of node hosts (e.g., the *blue* deployment) to the new set (e.g., the *green* deployment) after the new deployment has been verified. If a problem is detected, it is also then easy to rollback to the old deployment quickly.

While blue-green is a proven and valid strategy for deploying just about any software, there are always trade-offs. Not all environments have the same uptime requirements or the resources to properly perform blue-green deployments.

In an OpenShift Container Platform environment, the most suitable candidate for blue-green deployments are the node hosts. All user processes run on these systems and even critical pieces of OpenShift Container Platform infrastructure are self-hosted on these resources. Uptime is most important for these workloads and the additional complexity of blue-green deployments can be justified.

The exact implementation of this approach varies based on your requirements. Often the main challenge is having the excess capacity to facilitate such an approach.

Figure 4.1. Blue-Green Deployment



4.2. PREPARING FOR A BLUE-GREEN UPGRADE

After you have upgraded your master and etcd hosts using method described for In-place Upgrades, use the following sections to prepare your environment for a blue-green upgrade of the remaining node hosts.

4.2.1. Sharing Software Entitlements

Administrators must temporarily share the Red Hat software entitlements between the blue-green deployments or provide access to the installation content by means of a system such as Red Hat Satellite. This can be accomplished by sharing the consumer ID from the previous node host:

1. On each old node host that will be upgraded, note its **system identity** value, which is the consumer ID:

```
# subscription-manager identity | grep system
system identity: 6699375b-06db-48c4-941e-689efd6ce3aa
```

2. On each new RHEL 7 or RHEL Atomic Host 7 system that is going to replace an old node host, register using the respective consumer ID from the previous step:

```
# subscription-manager register --consumerid=6699375b-06db-48c4-941e-689efd6ce3aa
```

IMPORTANT

After a successful deployment, remember to unregister the old host with **subscription-manager clean** to prevent the environment from being out of compliance.

4.2.2. Labeling Blue Nodes

You must ensure that your current node hosts in production are labeled either **blue** or **green**. In this example, the current production environment will be **blue** and the new environment will be **green**.

1. Get the current list of node names known to the cluster:

   ```
   $ oc get nodes
   ```

2. Ensure that all hosts have appropriate node labels. All master hosts should be configured as schedulable node hosts (so that they are joined to the pod network and can run the web console pod). To improve cluster management, add a label to each host that describes its type, such as **type=master** or **type=node**.
   For example, to label node hosts that are also masters as **type=master**, run the following for each relevant **<node_name>**:

   ```
   $ oc label node <node_name> type=master
   ```

   To label non-master node hosts as **type=node**, run the following for each relevant **<node_name>**:

   ```
   $ oc label node <node_name> type=node
   ```

   Alternatively, if you have already finished labeling certain nodes with **type=master** and just want to label all remaining nodes as **type=node**, you can use the **--all** option and any hosts that already had a **type=** set will not be overwritten:

   ```
   $ oc label node --all type=node
   ```

3. Label all non-master node hosts in your current production environment to **color=blue**. For example, using the labels described in the previous step:

   ```
   $ oc label node -l type=node color=blue
   ```

   In the above command, the **-l** flag is used to match a subset of the environment using the selector **type=node**, and all matches are labeled with **color=blue**.

4.2.3. Creating and Labeling Green Nodes

Create the new green environment for any node hosts that are to be replaced by adding an equal number of new node hosts to the existing cluster. You can use the advanced install method as described in Adding Hosts to an Existing Cluster .

When adding these new nodes, use the following Ansible variables:

- Apply the **color=green** label automatically during the installation of these hosts by setting the **openshift_node_labels** variable for each node host. You can always adjust the labels after installation as well, if needed, using the **oc label node** command.

- In order to delay workload scheduling until the nodes are deemed healthy (which you will verify in later steps), set the **openshift_schedulable=false** variable for each node host to ensure they are unschedulable initially.

Example new_nodes Host Group

Add the following to your existing inventory. Everything that was in your inventory previously should remain.

```
[new_nodes]
node4.example.com openshift_node_labels="{'region': 'primary', 'color':'green'}"
openshift_schedulable=false
node5.example.com openshift_node_labels="{'region': 'primary', 'color':'green'}"
openshift_schedulable=false
node6.example.com openshift_node_labels="{'region': 'primary', 'color':'green'}"
openshift_schedulable=false
infra-node3.example.com openshift_node_labels="{'region': 'infra', 'color':'green'}"
openshift_schedulable=false
infra-node4.example.com openshift_node_labels="{'region': 'infra', 'color':'green'}"
openshift_schedulable=false
```

4.2.4. Verifying Green Nodes

Verify that your new green nodes are in a healthy state. Perform the following checklist:

1. Verify that new nodes are detected in the cluster and are in **Ready** state:

```
$ oc get nodes

ip-172-31-49-10.ec2.internal    Ready                    3d
```

2. Verify that the green nodes have proper labels:

```
$ oc get nodes --show-labels

ip-172-31-49-10.ec2.internal    Ready                    4d
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/instance-
type=m4.large,beta.kubernetes.io/os=linux,color=green,failure-
domain.beta.kubernetes.io/region=us-east-1,failure-domain.beta.kubernetes.io/zone=us-
east-1c,hostname=openshift-cluster-1d005,kubernetes.io/hostname=ip-172-31-49-
10.ec2.internal,region=us-east-1,type=infra
```

3. Perform a diagnostic check for the cluster:

```
$ oc adm diagnostics

[Note] Determining if client configuration exists for client/cluster diagnostics
Info:  Successfully read a client config file at '/root/.kube/config'
Info:  Using context for cluster-admin access: 'default/internal-api-upgradetest-openshift-
com:443/system:admin'
[Note] Performing systemd discovery

[Note] Running diagnostic: ConfigContexts[default/api-upgradetest-openshift-
com:443/system:admin]
       Description: Validate client config context is complete and has connectivity
...
         [Note] Running diagnostic: CheckExternalNetwork
              Description: Check that external network is accessible within a pod

       [Note] Running diagnostic: CheckNodeNetwork
```

```
        Description: Check that pods in the cluster can access its own node.

    [Note] Running diagnostic: CheckPodNetwork
        Description: Check pod to pod communication in the cluster. In case of ovs-subnet
network plugin, all pods
should be able to communicate with each other and in case of multitenant network plugin,
pods in non-global projects
should be isolated and pods in global projects should be able to access any pod in the cluster
and vice versa.

    [Note] Running diagnostic: CheckServiceNetwork
        Description: Check pod to service communication in the cluster. In case of ovs-
subnet network plugin, all
pods should be able to communicate with all services and in case of multitenant network
plugin, services in non-global
projects should be isolated and pods in global projects should be able to access any service
in the cluster.
...
```

4.3. REGISTRY AND ROUTER CANARY DEPLOYMENTS

A common practice is to scale the registry and router pods until they are migrated to new (green) infrastructure node hosts. For these pods, a *canary* deployment approach is commonly used.

Scaling these pods up will make them immediately active on the new infrastructure nodes. Pointing their deployment configuration to the new image initiates a rolling update. However, because of node anti-affinity, and the fact that the blue nodes are still unschedulable, the deployments to the old nodes will fail.

At this point, the registry and router deployments can be scaled down to the original number of pods. At any given point, the original number of pods is still available so no capacity is lost and downtime should be avoided.

4.4. WARMING THE GREEN NODES

In order for pods to be migrated from the blue environment to the green, the required container images must be pulled. Network latency and load on the registry can cause delays if there is not sufficient capacity built in to the environment.

Often, the best way to minimize impact to the running system is to trigger new pod deployments that will land on the new nodes. Accomplish this by importing new image streams.

Major releases of OpenShift Container Platform (and sometimes asynchronous errata updates) introduce new image streams for builder images for users of Source-to-Image (S2I). Upon import, any builds or deployments configured with image change triggers are automatically created.

Another benefit of triggering the builds is that it does a fairly good job of fetching the majority of the ancillary images to all node hosts such as the various builder images, the pod infrastructure image, and deployers. The green nodes are then considered *warmed* (that is, ready for the expected load increase), and everything else can be migrated over using node evacuation in a later step, proceeding more quickly as a result.

When you are ready to continue with the upgrade process, follow these steps to warm the green nodes:

1. Set the green nodes to schedulable so that new pods only land on them:

```
$ oc adm manage-node --schedulable=true --selector=color=green
```

2. Disable the blue nodes so that no new pods are run on them by setting them unschedulable:

```
$ oc adm manage-node --schedulable=false --selector=color=blue
```

3. Update the default image streams and templates as described in Manual In-place Upgrades.

4. Import the latest images as described in Manual In-place Upgrades.
   It is important to realize that this process can trigger a large number of builds. The good news is that the builds are performed on the green nodes and, therefore, do not impact any traffic on the blue deployment.

5. To monitor build progress across all namespaces (projects) in the cluster:

```
$ oc get events -w --all-namespaces
```

   In large environments, builds rarely completely stop. However, you should see a large increase and decrease caused by the administrative image import.

4.5. EVACUATING AND DECOMMISSIONING BLUE NODES

For larger deployments, it is possible to have other labels that help determine how evacuation can be coordinated. The most conservative approach for avoiding downtime is to evacuate one node host at a time.

If services are composed of pods using zone anti-affinity, then an entire zone can be evacuated at once. It is important to ensure that the storage volumes used are available in the new zone as this detail can vary among cloud providers.

In OpenShift Container Platform 3.2 and later, a node host evacuation is triggered whenever the node service is stopped. Node labeling is very important and can cause issues if nodes are mislabeled or commands are run on nodes with generalized labels. Exercise caution if master hosts are also labeled with **color=blue**.

When you are ready to continue with the upgrade process, follow these steps.

1. Evacuate and delete all blue nodes by following one of the following options:

   a. **Option A** Manually evacuate then delete all the **color=blue** nodes with the following commands:

```
$ oc adm manage-node --selector=color=blue --evacuate
$ oc delete node --selector=color=blue
```

   b. **Option B** Filter out the masters before running the **delete** command:

      i. Verify the list of blue node hosts to delete by running the following command. The output of this command includes a list of all node hosts that have the **color=blue** label but do not have the **type=master** label. All of the hosts in your cluster must be assigned both the **color** and **type** labels. You can change the command to apply more filters if you need to further limit the list of nodes.

```
$ oc get nodes -o go-template='{{ range .items }}{{ if (eq .metadata.labels.color
"blue") and (ne .metadata.labels.type "master") }}{{ .metadata.name }}{{ "\n" }}{{end}}{{
end }}'
```

ii. After you confirm the list of blue nodes to delete, run this command to delete that list of nodes:

```
$ for i in $(oc get nodes -o \
    go-template='{{ range .items }}{{ if (eq .metadata.labels.color "blue") and (ne
.metadata.labels.type "master") }}{{ .metadata.name }}{{ "\n" }}{{end}}{{ end }}'); \
do
    oc delete node $i
done
```

2. After the blue node hosts no longer contain pods and have been removed from OpenShift Container Platform they are safe to power off. As a safety precaution, leaving the hosts around for a short period of time can prove beneficial if the upgrade has issues.

3. Ensure that any desired scripts or files are captured before terminating these hosts. After a determined time period and capacity is not an issue, remove these hosts.

CHAPTER 5. UPDATING OPERATING SYSTEMS

5.1. PURPOSE

Updating the operating system (OS) on a host, by either upgrading across major releases or updating the system software for a minor release, can impact the OpenShift Container Platform software running on those machines. In particular, these updates can affect the **iptables** rules or **ovs** flows that OpenShift Container Platform requires to operate.

5.2. UPDATING THE OPERATING SYSTEM ON A HOST

Use the following to safely upgrade the OS on a host:

1. Drain the node in preparation for maintenance:

   ```
   $ oc adm drain <node_name> --force --delete-local-data --ignore-daemonsets
   ```

2. In order to protect sensitive packages that do not need to be updated, apply the exclude rules to the host:

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

3. Update the host packages and reboot the host. A reboot ensures that the host is running the newest versions and means that the **docker** and OpenShift Container Platform processes have been restarted, which forces them to check that all of the rules in other services are correct.

   ```
   # yum update
   # reboot
   ```

   However, instead of rebooting a node host, you can restart the services that are affected or preserve the **iptables** state. Both processes are described in the OpenShift Container Platform iptables topic. The **ovs** flow rules do not need to be saved, but restarting the OpenShift Container Platform node software fixes the flow rules.

4. Configure the host to be schedulable again:

   ```
   $ oc adm uncordon <node_name>
   ```

CHAPTER 6. DOWNGRADING OPENSHIFT

6.1. OVERVIEW

Following an OpenShift Container Platform 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 for the OpenShift Container Platform 3.9 to 3.7 downgrade path.

NOTE

You can downgrade directly from 3.9 to 3.7, but you must restore from the etcd backup.

If the upgrade failed at the 3.8 step, then the same downgrade procedures apply.

WARNING

These steps are currently only supported for RPM-based installations of OpenShift Container Platform and assumes downtime of the entire cluster.

6.2. 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/origin/master/master-config.yaml.<timestamp>
/var/lib/etcd/openshift-backup-pre-upgrade-<timestamp>
/etc/origin/master/scheduler.json.<timestamp>
```

Also, back up the ***node-config.yaml*** file on each node (including masters, which have the node component on them) with a timestamp:

```
/etc/origin/node/node-config.yaml.<timestamp>
```

If you are using an external etcd cluster (versus the single embedded etcd), the backup is likely created on all etcd members, though only one is required for the recovery process.

Keep a copy of the ***.rpmsave*** backups of the following files:

```
/etc/sysconfig/atomic-openshift-master-api
/etc/sysconfig/atomic-openshift-master-controller
/etc/etcd/etcd.conf
```

NOTE

Restore from the first pre-upgrade backup taken on the day of the upgrade.

6.3. SHUTTING DOWN THE CLUSTER

On all masters, nodes, and etcd members (if using an external etcd cluster), ensure the relevant services are stopped.

```
# systemctl stop atomic-openshift-master-api atomic-openshift-master-controllers
```

On all master and node hosts:

```
# systemctl stop atomic-openshift-node
```

On any external etcd hosts:

```
# systemctl stop etcd
```

6.4. REMOVING RPMS

The ***-excluder** packages add entries to the exclude directive in the host's ***/etc/yum.conf*** file when installed.

1. On all masters, nodes, and etcd members (if using a dedicated etcd cluster), remove the following packages:

```
# yum remove atomic-openshift \
    atomic-openshift-clients \
    atomic-openshift-node \
    atomic-openshift-master-api \
    atomic-openshift-master-controllers \
    openvswitch \
    atomic-openshift-sdn-ovs \
    tuned-profiles-atomic-openshift-node\
    atomic-openshift-excluder\
    atomic-openshift-docker-excluder
```

2. If you are using external etcd, also remove the **etcd** package:

```
# yum remove etcd
```

6.5. DOWNGRADING DOCKER

OpenShift Container Platform 3.9 requires Docker 1.13, while OpenShift Container Platform 3.7 requires Docker 1.12.

On any host running Docker 1.13, downgrade to Docker 1.12 using the following steps:

1. Remove all local containers and images on the host. Any pods backed by a replication controller will be recreated.

WARNING

The following commands are destructive and should be used with caution.

Delete all containers:

```
# docker rm $(docker ps -a -q) -f
```

Delete all images:

```
# docker rmi $(docker images -q)
```

2. Use **yum swap** (instead of **yum downgrade**) to install Docker 1.12.6:

```
# yum swap docker-* docker-*1.12.6 -y
# sed -i 's/--storage-opt dm.use_deferred_deletion=true//' /etc/sysconfig/docker-storage
# systemctl restart docker
```

3. You should now have Docker 1.12.6 installed and running on the host. Verify with the following:

```
# docker version
# systemctl status docker
```

6.6. REINSTALLING RPMS

Disable the OpenShift Container Platform 3.8 and 3.9 repositories, and re-enable the 3.7 repositories:

```
# subscription-manager repos \
    --disable=rhel-7-server-ose-3.8-rpms \
    --disable=rhel-7-server-ose-3.9-rpms \
    --enable=rhel-7-server-ose-3.7-rpms
```

On each master, install the following packages:

```
# yum install atomic-openshift \
    atomic-openshift-clients \
    atomic-openshift-node \
    atomic-openshift-master-api \
    atomic-openshift-master-controllers \
    openvswitch \
    atomic-openshift-sdn-ovs \
    tuned-profiles-atomic-openshift-node \
    atomic-openshift-excluder \
    atomic-openshift-docker-excluder
```

On each node, install the following packages:

```
# yum install atomic-openshift \
    atomic-openshift-node \
    openvswitch \
    atomic-openshift-sdn-ovs \
    tuned-profiles-atomic-openshift-node \
    atomic-openshift-excluder \
    atomic-openshift-docker-excluder
```

If using an external etcd cluster, install the following package on each etcd member:

```
# yum install etcd
```

6.7. RESTORING ETCD

The restore procedure for etcd configuration files replaces the appropriate files, then restarts the service.

If an etcd host has become corrupted and the **/etc/etcd/etcd.conf** file is lost, restore it using:

```
$ ssh master-0
# cp /backup/yesterday/master-0-files/etcd.conf /etc/etcd/etcd.conf
# restorecon -Rv /etc/etcd/etcd.conf
# systemctl restart etcd.service
```

In this example, the backup file is stored in the **/backup/yesterday/master-0-files/etcd.conf** path where it can be used as an external NFS share, S3 bucket, or other storage solution.

6.7.1. Restoring etcd v2 & v3 data

The following process restores healthy data files and starts the etcd cluster as a single node, then adds the rest of the nodes if an etcd cluster is required.

Procedure

1. Stop all etcd services:

   ```
   # systemctl stop etcd.service
   ```

2. To ensure the proper backup is restored, delete the etcd directories:

   - To back up the current etcd data before you delete the directory, run the following command:

     ```
     # mv /var/lib/etcd /var/lib/etcd.old
     # mkdir /var/lib/etcd
     # chown -R etcd.etcd /var/lib/etcd/
     # restorecon -Rv /var/lib/etcd/
     ```

   - Or, to delete the directory and the etcd, data, run the following command:

     ```
     # rm -Rf /var/lib/etcd/*
     ```

NOTE

In an all-in-one cluster, the etcd data directory is located in the **/var/lib/origin/openshift.local.etcd** directory.

3. Restore a healthy backup data file to each of the etcd nodes. Perform this step on all etcd hosts, including master hosts collocated with etcd.

```
# cp -R /backup/etcd-xxx/* /var/lib/etcd/
# mv /var/lib/etcd/db /var/lib/etcd/member/snap/db
# chcon -R --reference /backup/etcd-xxx/* /var/lib/etcd/
# chown -R etcd:etcd /var/lib/etcd/R
```

4. Run the etcd service on each host, forcing a new cluster.
   This creates a custom file for the etcd service, which overwrites the execution command adding the **--force-new-cluster** option:

```
# mkdir -p /etc/systemd/system/etcd.service.d/
# echo "[Service]" > /etc/systemd/system/etcd.service.d/temp.conf
# echo "ExecStart=" >> /etc/systemd/system/etcd.service.d/temp.conf
# sed -n '/ExecStart/s/"$/ --force-new-cluster"/p' \
    /usr/lib/systemd/system/etcd.service \
    >> /etc/systemd/system/etcd.service.d/temp.conf

# systemctl daemon-reload
# systemctl restart etcd
```

5. Check for error messages:

```
$ journalctl -fu etcd.service
```

6. Check for health status:

```
# etcdctl2 cluster-health
member 5ee217d17301 is healthy: got healthy result from https://192.168.55.8:2379
cluster is healthy
```

7. Restart the etcd service in cluster mode:

```
# rm -f /etc/systemd/system/etcd.service.d/temp.conf
# systemctl daemon-reload
# systemctl restart etcd
```

8. Check for health status and member list:

```
# etcdctl2 cluster-health
member 5ee217d17301 is healthy: got healthy result from https://192.168.55.8:2379
cluster is healthy

# etcdctl2 member list
5ee217d17301: name=master-0.example.com peerURLs=http://localhost:2380
clientURLs=https://192.168.55.8:2379 isLeader=true
```

9. After the first instance is running, you can restore the rest of your etcd servers.

6.7.1.1. Fix the **peerURLS** parameter

After restoring the data and creating a new cluster, the **peerURLs** parameter shows **localhost** instead of the IP where etcd is listening for peer communication:

```
# etcdctl2 member list
5ee217d17301: name=master-0.example.com peerURLs=http://*localhost*:2380
clientURLs=https://192.168.55.8:2379 isLeader=true
```

6.7.1.1.1. Procedure

1. Get the member ID using **etcdctl member list**:

   ```
   `etcdctl member list`
   ```

2. Get the IP where etcd listens for peer communication:

   ```
   $ ss -l4n | grep 2380
   ```

3. Update the member information with that IP:

   ```
   # etcdctl2 member update 5ee217d17301 https://192.168.55.8:2380
   Updated member with ID 5ee217d17301 in cluster
   ```

4. To verify, check that the IP is in the member list:

   ```
   $ etcdctl2 member list
   5ee217d17301: name=master-0.example.com peerURLs=https://*192.168.55.8*:2380
   clientURLs=https://192.168.55.8:2379 isLeader=true
   ```

6.7.2. Restoring etcd for v3

The restore procedure for v3 data is similar to the restore procedure for the v2 data.

Snapshot integrity may be optionally verified at restore time. If the snapshot is taken with **etcdctl snapshot save**, it will have an integrity hash that is checked by **etcdctl snapshot restore**. If the snapshot is copied from the data directory, there is no integrity hash and it will only restore by using **--skip-hash-check**.

IMPORTANT

The procedure to restore only the v3 data must be performed on a single etcd host. You can then add the rest of the nodes to the cluster.

Procedure

1. Stop all etcd services:

   ```
   # systemctl stop etcd.service
   ```

2. Clear all old data, because **etcdctl** recreates it in the node where the restore procedure is going to be performed:

```
# rm -Rf /var/lib/etcd
```

3. Run the **snapshot restore** command, substituting the values from the **/etc/etcd/etcd.conf** file:

```
# etcdctl3 snapshot restore /backup/etcd-xxxxxx/backup.db \
  --data-dir /var/lib/etcd \
  --name master-0.example.com \
  --initial-cluster "master-0.example.com=https://192.168.55.8:2380" \ --initial-cluster-token
"etcd-cluster-1" \
  --initial-advertise-peer-urls https://192.168.55.8:2380

2017-10-03 08:55:32.440779 I | mvcc: restore compact to 1041269
2017-10-03 08:55:32.468244 I | etcdserver/membership: added member 40bef1f6c79b3163
[https://192.168.55.8:2380] to cluster 26841ebcf610583c
```

4. Restore permissions and **selinux** context to the restored files:

```
# chown -R etcd.etcd /var/lib/etcd/
# restorecon -Rv /var/lib/etcd
```

5. Start the etcd service:

```
# systemctl start etcd
```

6. Check for any error messages:

```
$ journalctl -fu etcd.service
```

6.8. BRINGING OPENSHIFT CONTAINER PLATFORM SERVICES BACK ONLINE

After you finish your changes, bring OpenShift Container Platform back online.

Procedure

1. On each OpenShift Container Platform master, restore your master and node configuration from backup and enable and restart all relevant services:

```
# cp ${MYBACKUPDIR}/etc/sysconfig/atomic-openshift-master-api /etc/sysconfig/atomic-
openshift-master-api
# cp ${MYBACKUPDIR}/etc/sysconfig/atomic-openshift-master-controllers
/etc/sysconfig/atomic-openshift-master-controllers
# cp ${MYBACKUPDIR}/etc/origin/master/master-config.yaml.<timestamp>
/etc/origin/master/master-config.yaml
# cp ${MYBACKUPDIR}/etc/origin/node/node-config.yaml.<timestamp>
/etc/origin/node/node-config.yaml
# cp ${MYBACKUPDIR}/etc/origin/master/scheduler.json.<timestamp>
/etc/origin/master/scheduler.json
# systemctl enable atomic-openshift-master-api
# systemctl enable atomic-openshift-master-controllers
# systemctl enable atomic-openshift-node
# systemctl start atomic-openshift-master-api
# systemctl start atomic-openshift-master-controllers
# systemctl start atomic-openshift-node
```

2. On each OpenShift Container Platform node, restore your ***node-config.yaml*** file from backup and enable and restart the **atomic-openshift-node** service:

```
# cp /etc/origin/node/node-config.yaml.<timestamp> /etc/origin/node/node-config.yaml
# systemctl enable atomic-openshift-node
# systemctl start atomic-openshift-node
```

6.9. VERIFYING THE DOWNGRADE

1. To verify the downgrade, first check that all nodes are marked as **Ready**:

```
# oc get nodes
NAME                 STATUS                     AGE
master.example.com   Ready,SchedulingDisabled   165d
node1.example.com    Ready                      165d
node2.example.com    Ready                      165d
```

2. 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:v3.7.23",
# oc get -n default dc/router -o json | grep \"image\"
    "image": "openshift3/ose-haproxy-router:v3.7.23",
```

3. You can use the diagnostics tool on the master to look for common issues and provide suggestions:

```
# oc adm diagnostics
...
[Note] Summary of diagnostics execution:
[Note] Completed with no errors or warnings seen.
```

CHAPTER 7. KNOWN ISSUES

7.1. OVERVIEW

When upgrading your OpenShift Container Platform cluster, there are two primary data migrations that take place. The first occurs while still running the current version of OpenShift Container Platform, and the second happens after the upgrade completes.

If the pre-upgrade migration fails, the upgrade will be halted and the cluster administrator must resolve any data consistency problems before attempting the upgrade again. If the post-upgrade migration fails, the upgrade will complete and the cluster administrator should investigate the data migration issues as soon as possible. This document catalogs known data consistency problems and how to resolve them.

7.2. DUPLICATE PORTS IN SERVICE DEFINITION

Service objects have more strict validation rules, one that does not allow duplicate **Ports** entries (considering **Protocol** and **Port** pair). (BZ#1563929)

Error Output

```
TASK [Upgrade all storage] ********************************************************************************************************************
**********************************************************
fatal: [test.linux.lan]: FAILED! => {
    "changed": true,
    "cmd": [
        "oc",
        "adm",
        "--config=/etc/origin/master/admin.kubeconfig",
        "migrate",
        "storage",
        "--include=*",
        "--confirm"
    ],
    "delta": "0:07:06.574833",
    "end": "2018-04-03 11:22:07.834827",
    "failed_when_result": true,
    "msg": "non-zero return code",
    "rc": 1,
    "start": "2018-04-03 11:15:01.259994",
}

STDOUT

error: Service "my-service" is invalid: spec.ports[5]: Duplicate value: api.ServicePort{Name:"",
Protocol:"TCP", Port:8500, TargetPort:intstr.IntOrString{Type:0, IntVal:0, StrVal:""}, NodePort:0}
summary: total=37768 errors=1 ignored=0 unchanged=37767 migrated=0
info: to rerun only failing resources, add --include=services
error: 1 resources failed to migrate
```

To resolve this issue:

1. Log in to the CLI as a cluster administrator and inspect the **Service** objects listed in the error.

2. Edit the **.spec.ports** of the objects removing duplicate ports keeping in mind that pair **.spec.ports[*].name** and **.spec.ports[*].port** must be unique.