OpenShift Container Platform 3.10

Release Notes

Last Updated: 2018-10-31

Legal Notice

Abstract

Table of Contents

CHAPTER 1. OVERVIEW

The following release notes for OpenShift Container Platform 3.10 summarize all new features, major corrections from the previous version, and any known bugs upon general availability.

1.1. VERSIONING POLICY

OpenShift Container Platform provides strict backwards compatibility guarantees for all supported APIs, excluding alpha APIs (which may be changed without notice) and beta APIs (which may occasionally be changed in a non-backwards compatible manner).

The OpenShift Container Platform version must match between master and node hosts, excluding temporary mismatches during cluster upgrades. For example, in a 3.10 cluster, all masters must be 3.10 and all nodes must be 3.10. However, OpenShift Container Platform will continue to support older **oc** clients against newer servers. For example, a 3.5 **oc** will work against 3.4, 3.5, and 3.6 servers.

Changes of APIs for non-security related reasons will involve, at minimum, two minor releases (3.4 to 3.5 to 3.6, for example) to allow older **oc** to update. Using new capabilities may require newer **oc**. A 3.2 server may have additional capabilities that a 3.1 **oc** cannot use and a 3.2 **oc** may have additional capabilities that are not supported by a 3.1 server.

Table 1.1. Compatibility Matrix

	X.Y (**oc** Client)	**X.Y+N** [a] (**oc** Client)
X.Y (Server)	1	3
X.Y+N [a] (Server)	2	1

[a] Where **N** is a number greater than 1.

1 Fully compatible.

2 **oc** client may not be able to access server features.

3 **oc** client may provide options and features that may not be compatible with the accessed server.

CHAPTER 2. OPENSHIFT CONTAINER PLATFORM 3.10 RELEASE NOTES

2.1. OVERVIEW

Red Hat OpenShift Container Platform provides developers and IT organizations with a hybrid cloud application platform for deploying both new and existing applications on secure, scalable resources with minimal configuration and management overhead. OpenShift Container Platform supports a wide selection of programming languages and frameworks, such as Java, Javascript, Python, Ruby, and PHP.

Built on Red Hat Enterprise Linux and Kubernetes, OpenShift Container Platform provides a secure and scalable multi-tenant operating system for today's enterprise-class applications, while providing integrated application runtimes and libraries. OpenShift Container Platform enables organizations to meet security, privacy, compliance, and governance requirements.

2.2. ABOUT THIS RELEASE

Red Hat OpenShift Container Platform version 3.10 (RHBA-2018:1816) is now available. This release is based on OpenShift Origin 3.10 and it uses Kubernetes 1.10. New features, changes, bug fixes, and known issues that pertain to OpenShift Container Platform 3.10 are included in this topic.

OpenShift Container Platform 3.10 is supported on RHEL 7.4 and 7.5 with the latest packages from Extras, including Docker 1.13. It is also supported on Atomic Host 7.4.5 and newer. The **docker-latest** package is now deprecated.

TLSV1.2 is the only supported security version in OpenShift Container Platform version 3.4 and later. You must update if you are using TLSV1.0 or TLSV1.1.

For initial installations, see the Installing Clusters documentation.

To upgrade to this release from a previous version, see the Upgrading Clusters documentation.

2.3. NEW FEATURES AND ENHANCEMENTS

This release adds improvements related to the following components and concepts.

2.3.1. Installation

2.3.1.1. Important Changes

A number of important changes have been made to the installation process in OpenShift Container Platform 3.10.

- The control plane components (etcd, API server, and controllers) are now run as static pods by the kubelet on the masters. This is now the only supported way they can be run. The upgrade will change over to this automatically. You can see the pods once the control plane comes up with **`oc get pods -n kube-system`** and execute, log, and inspect them using normal **`oc`** CLI commands.
- The containerized mode (starting containers directly from docker) for OpenShift Container Platform is no longer supported.
- openshift-sdn and openvswitch will be run in a daemonset.

- The ***/etc/sysconfig/(origin|atomic-openshift)-(api|controllers)*** files will no longer be used. A new ***/etc/origin/master/master.env*** file can be used to set environment variables for the static pods. The set of configurations available as environment variables are limited. Future versions of OpenShift Container Platform will remove this configuration in favor of control plane files, so consider the ***master.env*** file a last resort and deprecated. To bridge the gap for host-level debugging of static pods, a set of shims are installed to ***/usr/local/bin***:
  - ***/usr/local/bin/master-restart (etcd|api|controllers)***
    - This causes the kubelet to restart the entire static pod for the named component.
    - This is *not* a blocking operation and is deliberately limited to prevent complex scripting.
    - You can also trigger a restart by performing:

      ```
      $ oc annotate -n kube-system POD_NAME gen=N --overwrite
      ```

  - ***/usr/local/bin/master-logs (etcd etcd|api api|controllers controllers)***
    - Print the most recent container logs for the given component (the component name and container name match).
    - Extra options are passed to the container runtime.
    - use **`oc logs -n kube-system POD_NAME`** instead.
  - ***/usr/local/bin/master-exec (etcd etcd|api api|controllers controllers)***
    - Execute a command inside the container to support the backup of etcd. Use these sparingly.

See Containerized Installation Method Support Changes and Control Plane as Static Pods for more information.

Debugging Tips

Use **`oc get nodes`** to see if nodes are ready. If a node is not ready, it is usually:

- Waiting for certificate approval. Run **`oc get csr`** to see all certificates being requested. Run **`oc get csr -o name | xargs oc adm certificate approve`** to approve all certificates. When you approve certificates, it can take a few seconds for the node to report that it is ready again.
- The SDN pod is not running on the node. You should usually be able to delete the SDN or OVS pods on a given node to trigger a reset and processes should continue.
- If your API or controller container is not running (**`docker ps | grep api`**) use **`master-logs api`** to see why. Typically, the cause is a failed configuration.
- There is one known Kubelet wedge state that will be fixed in the 1.10 rebase where the Kubelet will display messages like **system:anonymous cannot access resource foo**. This means that the certificates expired before the kubelet could refresh them. If restarting the kubelet does not fix the issue, delete the contents of ***/etc/origin/node/certificates/***, and then restart the kubelet.
- If you see any component that does not converge, meaning that it stays in a crashlooping state, open a bug with any logs from that pod. This is most commonly an openshift-sdn / OVS issue where the networking in the container is lost, but the kubelet or SDN does not realize it.

Atomic Host Deprecation
Atomic Host is now deprecated. Atomic Host will continue to be supported in OpenShift Container Platform 3.11 and will be removed in OpenShift Container Platform 4.0.

2.3.2. OpenShift Automation Broker

2.3.2.1. The OpenShift Automation Broker Now Uses CRDs Instead of Local etcd

The OpenShift Automation Broker will now use custom resource definitions (CRDs) instead of a local etcd instance.

There is now a migration path from etcd to CRD for **`openshift-ansible`**

2.3.2.2. mediawiki-abp Examples Updated

Examples of **mediawiki-apb** Ansible playbook bundles (APB) are updated to use version 1.27.

2.3.3. Storage

2.3.3.1. Persistent Volume (PV) Provisioning Using OpenStack Manila (Technology Preview)

Persistent volume (PV) provisioning using OpenStack Manila is currently in Technology Preview and not for production workloads.

OpenShift Container Platform is capable of provisioning PVs using the OpenStack Manila shared file system service.

See Persistent Storage Using OpenStack Manila for more information.

2.3.3.2. PV Resize (Technology Preview)

Persistent volume (PV) resize is currently in Technology Preview and not for production workloads.

You can expand persistent volume claims online from OpenShift Container Platform for glusterFS.

1. Create a storage class with **`allowVolumeExpansion=true`**.
2. The PVC uses the storage class and submits a claim.
3. The PVC specifies a new increased size.
4. The underlying PV is resized.

See Expanding Persistent Volumes for more information.

2.3.3.3. Container Storage Interface (Technology Preview)

Container Storage Interface (CSI) is currently in Technology Preview and not for production workloads.

CSI allows OpenShift Container Platform to consume storage from storage backends that implement the CSI interface as persistent storage.

See Persistent Storage Using Container Storage Interface (CSI) for more information.

2.3.3.4. Protection of Local Ephemeral Storage (Technology Preview)

Protection of Local Ephemeral Storage is currently in Technology Preview and not for production workloads.

You can now control the use of the local ephemeral storage feature on your nodes in order to prevent users from exhausting node local storage with their pods and other pods that happen to be on the same node.

This feature is disabled by default. If enabled, the OpenShift Container Platform cluster uses ephemeral storage to store information that does not need to persist after the cluster is destroyed.

See Configuring Ephemeral Storage for more information.

2.3.3.5. Tenant-driven Storage Snapshotting (Technology Preview)

Tenant-driven storage snapshotting is currently in Technology Preview and not for production workloads.

Tenants now have the ability to leverage the underlying storage technology backing the persistent volume (PV) assigned to them to make a snapshot of their application data. Tenants can also now restore a given snapshot from the past to their current application.

An external provisioner is used to access the EBS, GCE pDisk, and HostPath. This Technology Preview feature has tested EBS and HostPath. The tenant must stop the pods and start them manually.

1. The administrator runs an external provisioner for the cluster. These are images from the Red Hat Container Catalog.

2. The tenant made a PVC and owns a PV from one of the supported storage solutions.The administrator must create a new **`StorageClass`** in the cluster with:

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: snapshot-promoter
provisioner: volumesnapshot.external-storage.k8s.io/snapshot-
promoter
```

3. The tenant can create a snapshot of a PVC named **`gce-pvc`** and the resulting snapshot will be called **`snapshot-demo`**.

```
$ oc create -f snapshot.yaml

apiVersion: volumesnapshot.external-storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: snapshot-demo
  namespace: myns
spec:
  persistentVolumeClaimName: gce-pvc
```

4. Now, they can restore their pod to that snapshot.

```
$ oc create -f restore.yaml
apiVersion: v1
```

```
kind: PersistentVolumeClaim
metadata:
  name: snapshot-pv-provisioning-demo
  annotations:
    snapshot.alpha.kubernetes.io/snapshot: snapshot-demo
spec:
  storageClassName: snapshot-promoter
```

2.3.4. Scale

2.3.4.1. Cluster Limits

Updated guidance around Cluster Limits for OpenShift Container Platform 3.10 is now available.

2.3.4.2. Device Plug-ins

Device Plug-ins are now moved out of Technology Preview and generally available in OpenShift Container Platform 3.10. OpenShift Container Platform supports the device plug-in API, but the device plug-in containers are supported by individual vendors.

Device plug-ins allow you to use a particular device type (GPU, InfiniBand, or other similar computing resources that require vendor-specific initialization and setup) in your OpenShift Container Platform pod without needing to write custom code. The device plug-in provides a consistent and portable solution to consume hardware devices across clusters. The device plug-in provides support for these devices through an extension mechanism, which makes these devices available to containers, provides health checks of these devices, and securely shares them.

A device plug-in is a gRPC service running on the nodes (external to **`atomic-openshift-node.service`**) that is responsible for managing specific hardware resources.

See the Developer Guide for further conceptual information about Device Plug-ins.

2.3.4.3. CPU Manager

CPU Manager is now moved out of Technology Preview and generally available in OpenShift Container Platform 3.10.

CPU Manager manages groups of CPUs and constrains workloads to specific CPUs.

CPU Manager is useful for workloads that have some of these attributes:

- Require as much CPU time as possible.
- Are sensitive to processor cache misses.
- Are low-latency network applications.
- Coordinate with other processes and benefit from sharing a single processor cache.

See Using CPU Manager for more information.

2.3.4.4. Device Manager

Device Manager is now moved out of Technology Preview and generally available in OpenShift Container Platform 3.10. OpenShift Container Platform supports the device plug-in API, but the device plug-in containers are supported by individual vendors.

Some users want to set resource limits for hardware devices within their pod definition and have the scheduler find the node in the cluster with those resources. While at the same time, Kubernetes needed a way for hardware vendors to advertise their resources to the kubelet without forcing them to change core code within Kubernetes

The kubelet now houses a device manager that is extensible through plug-ins. You load the driver support at the node level. Then, you or the vendor writes a plug-in that listens for requests to stop/start/attach/assign the requested hardware resources seen by the drivers. This plug-in is deployed to all the nodes via a daemonSet.

See Using Device Manager for more information.

2.3.4.5. Huge Pages

Huge pages are now moved out of Technology Preview and generally available in OpenShift Container Platform 3.10.

Memory is managed in blocks known as pages. On most systems, a page is 4Ki. 1Mi of memory is equal to 256 pages; 1Gi of memory is 256,000 pages, and so on. CPUs have a built-in memory management unit that manages a list of these pages in hardware. The Translation Lookaside Buffer (TLB) is a small hardware cache of virtual-to-physical page mappings. If the virtual address passed in a hardware instruction can be found in the TLB, the mapping can be determined quickly. If not, a TLB miss occurs, and the system falls back to slower, software-based address translation, resulting in performance issues. Since the size of the TLB is fixed, the only way to reduce the chance of a TLB miss is to increase the page size.

A huge page is a memory page that is larger than 4Ki. On x86_64 architectures, there are two common huge page sizes: 2Mi and 1Gi. Sizes vary on other architectures. In order to use huge pages, code must be written so that applications are aware of them. Transparent Huge Pages (THP) attempt to automate the management of huge pages without application knowledge, but they have limitations. In particular, they are limited to 2Mi page sizes. THP can lead to performance degradation on nodes with high memory utilization or fragmentation due to defragmenting efforts of THP, which can lock memory pages. For this reason, some applications may be designed to use or recommend usage of pre-allocated huge pages instead of THP.

In OpenShift Container Platform, applications in a pod can allocate and consume pre-allocated huge pages.

See Managing Huge Pages for more information.

2.3.5. Networking

2.3.5.1. Route Annotation Limits Concurrent Connections

The route annotation **`haproxy.router.openshift.io/pod-concurrent-connections`** limits concurrent connections.

See Route-specific Annotations for more information.

2.3.5.2. Support for Kubernetes Ingress Objects

The Kubernetes ingress object is a configuration object determining how inbound connections reach internal services. OpenShift Container Platform has support for these objects, starting in OpenShift Container Platform 3.10, using a ingress controller configuration file.

See Support for Kubernetes ingress objects for more information.

2.3.5.3. IP failover Management Limited to 254 Groups of VIP Addresses

IP failover management is limited to 254 groups of VIP addresses. By default, OpenShift Container Platform assigns one IP address to each group. You can use the **`virtual-ip-groups`** option to change this so multiple IP addresses are in each group and define the number of VIP groups available for each VRRP instance when configuring IP failover.

See High Availability for more information.

2.3.5.4. Allow DNS Names for Egress Routers

You can now set the egress router to refer to an external service, with a potentially unstable IP address, by its host name.

See Deploying an Egress Router DNS Proxy Pod for more information.

2.3.5.5. Expanding the serviceNetwork

You can now grow the service network address range in a multi-node environment to a larger address space. This does not cover migration to a different range, just the increase of an existing range.

See Expanding the Service Network for more information.

2.3.5.6. Improved OpenShift Container Platform and Red Hat OpenStack Integration with Kuryr (Technology Preview)

This feature is currently in Technology Preview and is not for production workloads.

See Kuryr SDN Administration and Configuring Kuryr SDN for best practices in OpenShift Container Platform and Red Hat OpenStack integration.

2.3.6. Master

2.3.6.1. The Descheduler (Technology Preview)

The Descheduler is currently in Technology Preview and is not for production workloads.

The descheduler moves pods from less desirable nodes to new nodes. Pods can be moved for various reasons, such as:

- Some nodes are under- or over-utilized.
- The original scheduling decision does not hold true any more, as taints or labels are added to or removed from nodes, pod/node affinity requirements are not satisfied any more.
- Some nodes failed and their pods moved to other nodes.
- New nodes are added to clusters.

See Descheduling for more information.

2.3.6.2. Node Problem Detector (Technology Preview)

The Node Problem Detector is currently in Technology Preview and is not for production workloads.

The Node Problem Detector monitors the health of your nodes by finding certain problems and reporting these problems to the API server, where external controllers could take action. The Node Problem Detector is a daemon that runs on each node as a daemonSet. The daemon tries to make the cluster aware of node level faults that should make the node not schedulable. When you start the Node Problem Detector, you tell it a port over which it should broadcast the issues it finds. The detector allows you to load sub-daemons to do the data collection. There are three as of today. Issues found by the problem daemon can be classified as **`NodeCondition`**.

Problem daemons:

- Kernel Monitor: Monitors kernel log via journald and reports problems according to regex patterns.
- AbrtAdaptor: Monitors the node for kernel problems and application crashes from journald.
- CustomerPluginMonitor: Allows you to test for any condition and exit on a **`0`** or **`1`** should your condition not be met.

See Node Problem Detector for more information.

2.3.6.3. System Services Now Hosted on Pods

Each of the system services, API, controllers, and etcd, used to run as system services on the master. These services now run on static pods in the cluster. As a result, there are new commands to restart these services: **`master-restart api`**, **`master-restart controllers`**, and **`master-restart etcd`**. To view log information on these services, use **`master-logs api api`**, **`master-logs controllers controllers`**, and **`master-logs etcd etcd`**.

See Important Changes for more information.

2.3.6.4. New Node Configuration Process

You can modify existing nodes through a configuration map rather than the ***node-config.yaml***. The installation creates three node configuration groups: **node-config-master**, **node-config-infra**, and **node-config-compute** and creates a configuration map for each group. A sync pod watches for changes to these configuration maps. When a change is detected, the sync pod updates the ***node-config.yaml*** file on all of the nodes.

2.3.6.5. LDAP Group Pruning

To prune groups records from an external provider, administrators can run the following command:

```
$ oc adm prune groups --sync-config=path/to/sync/config [<options>]
```

See Pruning groups for more information.

2.3.6.6. Podman (Technology Preview)

Podman is currently in Technology Preview and is not for production workloads.

Podman is a daemon-less CLI/API for running, managing, and debugging OCI containers and pods. It:

- Is fast and lightweight.
- Leverages runC.
- Provides a syntax for working with containers.
- Has remote management API via Varlink.
- Provides systemd integration and advanced namespace isolation.

For more information, see Crictl Vs Podman.

2.3.7. Metrics and Logging

2.3.7.1. Prometheus (Technology Preview)

Prometheus remains in Technology Preview and is not for production workloads. Prometheus, AlertManager, and AlertBuffer versions are now updated and node-exporter is now included:

- prometheus 2.2.1
- Alertmanager 0.14.0
- AlertBuffer 0.2
- node_exporter 0.15.2

You can deploy Prometheus on an OpenShift Container Platform cluster, collect Kubernetes and infrastructure metrics, and get alerts. You can see and query metrics and alerts on the Prometheus web dashboard. Alternatively, you can bring your own Grafana and hook it up to Prometheus.

See Prometheus on OpenShift for more information.

2.3.8. Developer Experience

2.3.8.1. Service Catalog command-line interface (CLI)

The Service Catalog command-line interface (CLI) allows you to provision and bind services from the command line. You can use a full set of commands to list, describe, provision, deprovision, bind, and unbind.

The Service Catalog CLI utility called **`svcat`** is available for easier interaction with Service Catalog resources. **`svcat`** communicates with the Service Catalog API by using the aggregated API endpoint on an OpenShift cluster.

See Service catalog command-line interface (CLI) for more information.

2.3.8.2. New ignore-volume-az Configuration Option

A new configuration option, **`ignore-volume-az`**, is now available in the ***cloud.conf*** file for Red Hat OpenStack. This is added to let OpenShift Container Platform not create labels with zones for persistent

volumes. OpenStack Cinder and OpenStack Nova can have different topology zones. OpenShift Container Platform works exclusively with Nova zones, ignoring Cinder topology. Therefore, it makes no sense to set the label with a Cinder zone name into PVs, in case it is different than Nova zones. A pod that uses such a PV would be unschedulable by OpenShift Container Platform. Cluster administrators can now turn off labeling of Cinder PVs and make their pods schedulable. (**BZ#1500776**)

2.3.8.3. CLI Plug-ins (Technology Preview)

CLI plug-ins are currently in Technology Preview and not for production workloads.

Usually called *plug-ins* or *binary extensions*, this feature allows you to extend the default set of **oc** commands available and, therefore, allows you to perform new tasks.

See Extending the CLI for information on how to install and write extensions for the CLI.

2.3.8.4. Jenkins Updates

There is now synchronized removal of build jobs, which allows for the cleanup of old, stale jobs.

Jenkins is now updated to 2.107.3-1.1 and Jenkins build agent (slave) images are now updated:

- Node.js 8
- Maven 3.5

The following images are deprecated in OpenShift Container Platform 3.10:

```
jenkins-slave-maven-*
jenkins-slave-nodejs-*
```

The images still exist in the interim so you can migrate your applications to the newer images:

```
jenkins-agent-maven-*
jenkins-agent-nodejs-*
```

For more information, see Jenkins Agents.

2.3.9. Registry

2.3.9.1. Expose Registry Metrics with OpenShift Authentication

The OpenShift Container Platform 3.10 registry metrics endpoint is now protected by built-in OpenShift Container Platform authentication. You can use a ClusterRole to access registry metrics.

See Accessing Registry Metrics for more information.

2.3.10. Web Console

2.3.10.1. Improved Service Catalog Search

There is now an improved search algorithm for the service catalog UI. Weighting is based on where the match is found and factors include the title, description, and tagging.

2.3.10.2. Improved Way to Show and Choose Routes for Applications

There is now an improved way to show and choose routes for an application. There is now indication that there are multiple routes available. Annotate the route that you would like to be primary:

```
console.alpha.openshift.io/overview-app-route: ‘true’
```

2.3.10.3. Create Generic secrets

You can create generic secrets in the web console (secrets with any key / value pairs). You can already create secrets, but now you can create opaque secrets. This behaves like creating ConfigMaps.

2.3.10.4. Miscellaneous Changes

- The ***xterm.js*** dependency for pod terminal was updated with greatly improved performance.
- You can now create image pull secrets directly from the deploy image dialog.

2.3.11. Security

2.3.11.1. Specify TLS Cipher Suite for etcd

You can set TLS cipher suites for use with etcd in order to meet security policies.

For more information, see Specifying TLS ciphers for etcd

2.3.11.2. Control Sharing the PID Namespace Between Containers (Technology Preview)

Control Sharing the PID Namespace Between Containers is currently in Technology Preview and not for production workloads.

Use this feature to configure cooperating containers in a pod, such as a log handler sidecar container, or to troubleshoot container images that do not include debugging utilities like a shell.

- The feature gate **`PodShareProcessNamespace`** is set to **`false`** by default.
- Set **`feature-gates=PodShareProcessNamespace=true`** in the API server, controllers, and kubelet.
- Restart the API server, controller, and node service.
- Create a pod with the specification of **`shareProcessNamespace: true`**.
- Run **`oc create -f <pod spec file>`**.

Caveats

When the PID namespace is shared between containers:

- Sidecar containers are not isolated.
- Environment variables are now visible to all other processes.
- Any **kill all** semantics used within the process are now broken.

- Any **exec** processes from other containers will now show up.

See Expanding Persistent Volumes for more information.

2.3.11.3. Router Service Account No Longer Needs Access to Secrets

The router service account no longer needs permission to read all secrets. This improves security. Previously, if the router was compromised it could read all of the most sensitive data in the cluster.

Now, when you create an ingress object, a corresponding route object is created. If an ingress object is modified, a changed secret should take effect soon after. If an ingress object is deleted, a route that was created for it will be deleted.

2.3.12. Documentation

2.3.12.1. Removed Quick Installation

In OpenShift Container Platform 3.10, the Quick Installation method and the corresponding documentation is now removed.

2.3.12.2. Removed Manual Upgrade

In OpenShift Container Platform 3.10, the Manual Upgrade method and the corresponding documentation is now removed.

2.3.12.3. Installation and Configuration Guidance Now Separated

The Installation and Configuration Guide is now separated into Installing Clusters and Configuring Clusters for increased readability.

2.4. NOTABLE TECHNICAL CHANGES

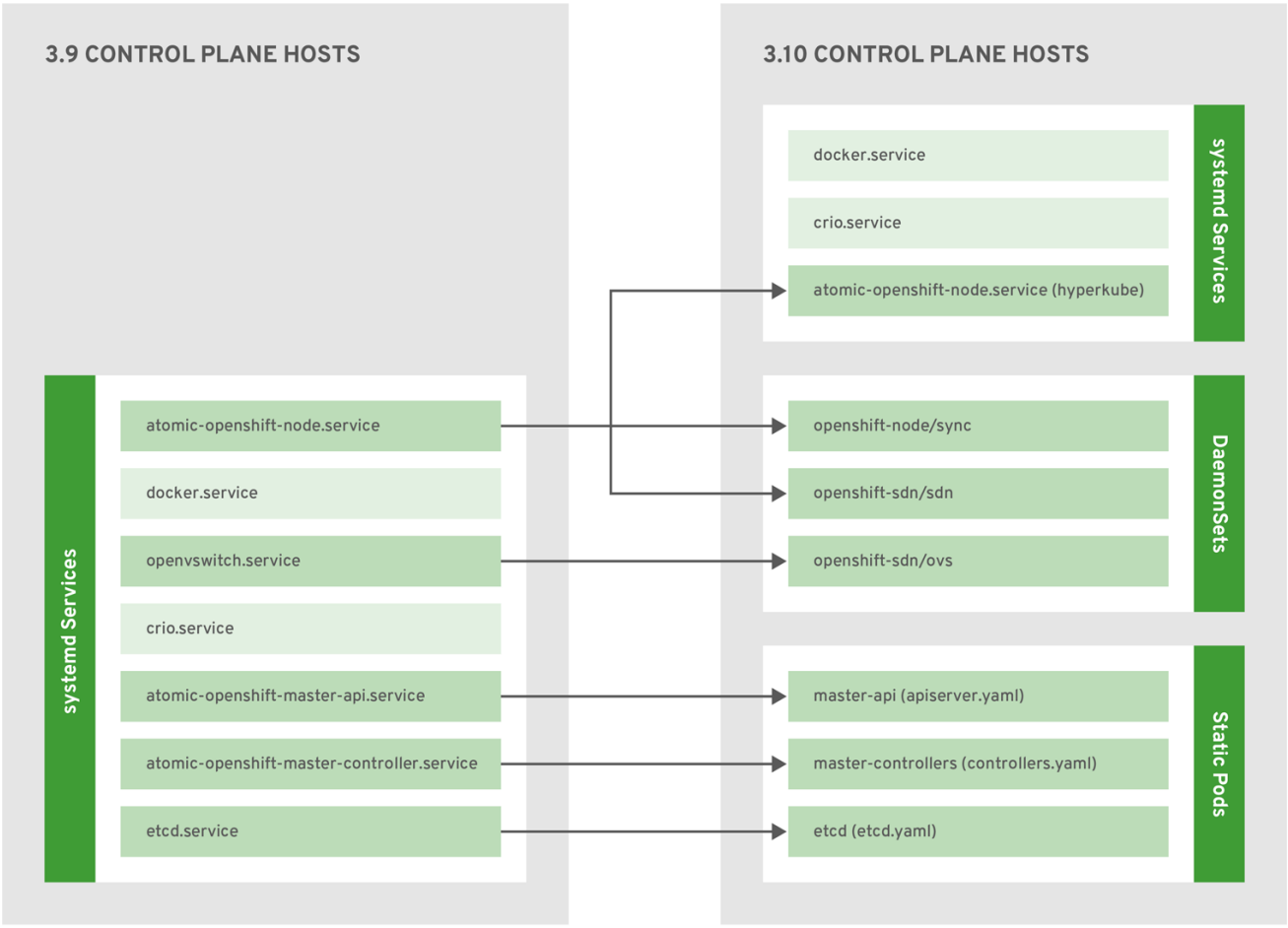
OpenShift Container Platform 3.10 introduces the following notable technical changes.

Major Changes to Cluster Architecture
OpenShift Container Platform 3.10 introduces major architecture changes in how control plane and node components are deployed, affecting new installations and upgrades from OpenShift Container Platform 3.9.

The following sections highlight the most significant changes, with more detail provided in the Architecture Guide.

Control Plane as Static Pods
While previously run as **systemd** services or system containers, the control plane components (apiserver, controllers, and etcd when co-located with a master) are now run as static pods by the kubelet on master hosts. The node components **openshift-sdn** and **openvswitch** are also now run using a DaemonSet instead of a **systemd** service.

Figure 2.1. Control plane host architecture changes



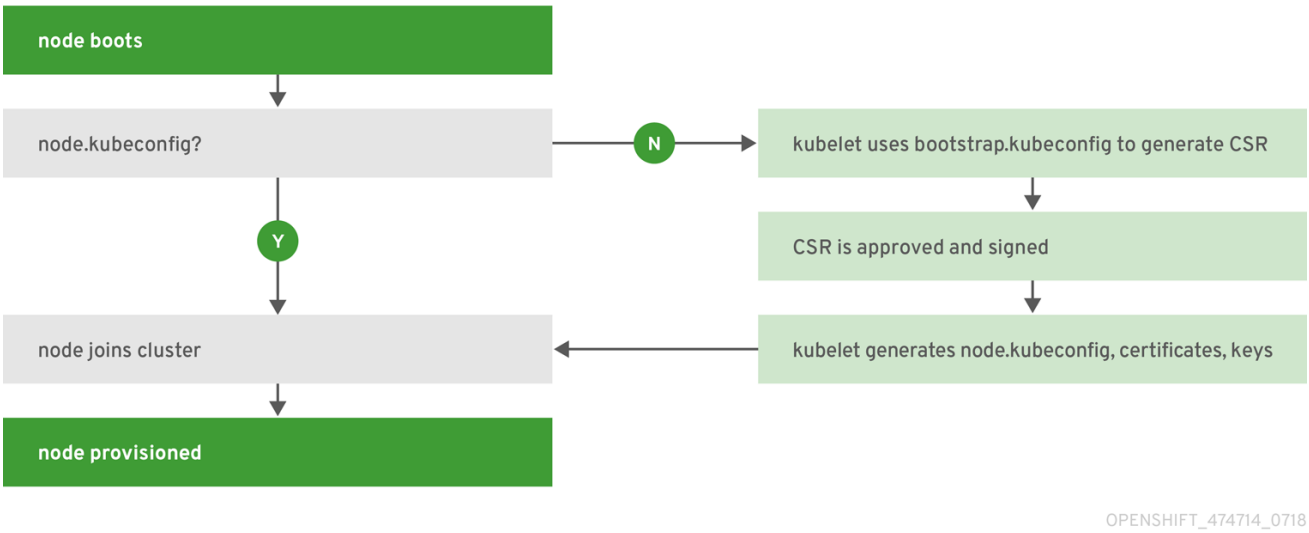
This is now the only supported way they can be run; system containers are no longer supported, (sans the kublet) with the exception of the node service RHEL Atomic Host. The upgrade will change over to the new architecture automatically. Control plane components continue to read configurations from the */etc/origin/master/* and */etc/etcd/* directories.

You can see the pods after the control plane starts using **`oc get pods -n kube-system`**, and **`exec`**, **`log`**, and **`inspect`** them using normal **`oc`** CLI commands.

Why?
Static pods are managed directly by the kubelet daemon on a specific node, without the API server having to observe it. With this simplified architecture, master and node static pods do not have an associated replication controller, and the kubelet daemon itself watches and restarts them if they crash. Static pods are always bound to one kubelet daemon and always run on the same node with it.

Nodes Bootstrapped from the Master
Nodes are now bootstrapped from the master by default, which means nodes will pull their pre-defined configuration, client and server certificates from the master. The 3.10 upgrade will automatically transform your nodes to use this new mode.

Figure 2.2. Node bootstrapping workflow overview



Why?
The goal for bootstrapping is to allow faster node start-up by reducing the differences between nodes, as well as centralizing more configuration and letting the cluster converge on the desired state. This enables certificate rotation and centralized certificate management by default (use `oc get csr` to see pending certificates).

Containerized Installation Method Support Changes
Documentation for previous versions of OpenShift Container Platform referred to the "containerized installation method", where OpenShift Container Platform components ran as standard container images. Starting in OpenShift Container Platform 3.10, support for containerized components has changed.

The OpenShift Container Platform 3.10 upgrade:

- Migrates RHEL Server hosts to the RPM-based installation method for the kubelet
- Migrates the container runtime and RHEL Atomic Hosts to the system container-based installation method for the kubelet only (because the container runtime is part of RHEL Atomic Host)

If you upgrade from OpenShift Container Platform 3.9 to 3.10 and standalone etcd was run as containerized on RHEL, then the installation will remain containerized after the upgrade.

These containerized installation methods are now the only supported methods for their respective RHEL variants, and the former method (where OpenShift Container Platform components run as standard container images) has been removed and is no longer supported starting in 3.10.

Why?
This reduces the number of installation and upgrade paths, and aligns better with features to be introduced in future releases.

Configuration Files
To upgrade from OpenShift Container Platform 3.9 to 3.10, you must first create a configuration file that maps your previous master and node configurations to the new ConfigMap usage, and supply the mapping when initiating your cluster upgrade. This ensures that the upgrade does not begin without this critical information, and that you are not left at the end of the upgrade with hosts using the previous style deployment.

In addition, the ***/etc/sysconfig/(origin|atomic-openshift)-(api|controllers)*** files will no longer be used. A new ***/etc/origin/master/master.env*** file can be used to set environment variables for the static pods. The set of configuration available as environment variables is limited (proxy and log levels). Future versions of OpenShift Container Platform will remove this configuration in favor of control plane files, so consider the ***master.env*** file a last resort and deprecated.

Updates to Static Pod Images

The following images are removed:

```
openshift3/ose-*
openshift3/container-engine-*
openshift3/node-*
openshift3/openvswitch-*
```

These images are replaced with:

```
openshift3/ose-node-*
openshift3/ose-control-plane-*
```

The image **`openshift3/metrics-schema-installer-container`** is also added.

The image **`openshift3/ose-sti-builder`** is now replaced by **`openshift3/ose-docker-builder`**, which already existed.

See Syncing Images for more information.

Pod Flag Removed for oc port-forward

The deprecated **`-p <POD>`** flag for **`oc port-forward`** is removed. Use **`oc port-forward pod/<POD>`** instead.

Specify the API Group and Version without the API prefix

When enabling or disabling API groups with the **`--runtime-config`** flag in **`kubernetesMasterConfig.apiServerArguments`**, specify **`<group>/<version>`** without the API prefix. In future releases, the API prefix will be disallowed. For example:

```
kubernetesMasterConfig:
  apiServerArguments:
    runtime-config:
    - apps.k8s.io/v1beta1=false
    - apps.k8s.io/v1beta2=false
...
```

Output of -o name Now Includes API Group

The output format of **`-o name`** now includes the API group and singular kind. For example:

```
$ oc get imagestream/my-image-stream -o name
imagestream.image.openshift.io/my-image-stream
```

Deprecated Web Console Support for Internet Explorer 11

Web console support for Internet Explorer (IE) 11 is now deprecated. This will be removed in a future version of OpenShift Container Platform. Microsoft Edge is still a supported browser.

Local Provisioner Configuration Changes

Adding a new device is semi-automatic. The provisioner periodically checks for new mounts in the configured directories. The administrator needs to create a new subdirectory there, mount a device there, and allow the pods to use the device by applying the SELinux label.

See Configuring for Local Volume for more information.

OpenStack Configuration Updates

When configuring the Red Hat OpenStack cloud provider, the node's host name must match the instance name in OpenStack to ensure that the registered name conforms to DNS-1123 specification.

Deprecated openshift-namespace Flag Now Removed

The deprecated **`openshift-namespace`** flag is now removed from the **`oc adm create-bootstrap-policy-file`** command.

Use of openshift_set_node_ip and openshift_ip Are No Longer Supported

In OpenShift Container Platform 3.10, the use of **`openshift_set_node_ip`** and **`openshift_ip`** are no longer supported.

You Can No Longer Configure dnsIP

It is no longer possible to configure the **`dnsIP`** value of the node, which could previously be set via **`openshift_dns_ip`**.

Removed openshift_hostname Variable

The **`openshift_hostname`** variable is now removed.

Use of openshift_docker_additional_registries Discouraged

Do not use or rely on **`openshift_docker_additional_registries`**.

openshift-infra Reserved for System Components

The **`openshift-infra`** namespace is reserved for system components. It does not run OpenShift Container Platform admission plug-ins for Kubernetes resources. SCC admission will not run for pods in the **`openshift-infra`** namespace. This can cause pods to fail, especially if they make use of persistent volume claims and rely on SCC-assigned **`uid`**/**`fsGroup`**/**`supplementalGroup`**/**`seLinux`** settings.

oc edit Respects Kube_EDITOR

The **`oc edit`** command now respects **`KUBE_EDITOR`**. **`OC_EDITOR`** support will be removed in a future release, so it is recommended that you switch to **`KUBE_EDITOR`**.

batch/v2alpha1 API Version No Longer Served by Default

The **`batch/v2alpha1`** API version is no longer served by default. If required, it can be re-enabled in the ***master-config.yaml*** file with this configuration:

```
kubernetesMasterConfig:
  apiServerArguments:
    ...
    runtime-config:
    - apis/batch/v2alpha1=true
```

New openshift_additional_ca Option

There is a new option in the OpenShift Ansible installer, **`openshift_additional_ca`**, which points to a file containing the load balancer CA certificate. If the cluster is using a load balancer which requires a difference CA than the one generated by the installer for the the master node, then the user will need to add this additional CA certificate to the ***/etc/origin/master/ca-bundle.crt*** file. This will make it available to pods in the cluster.

Namespace-scoped Requests

`subjectaccessreviews.authorization.openshift.io` and **`resourceaccessreviews.authorization.openshift.io`** will be cluster-scoped only in a future release. Use **`localsubjectaccessreviews.authorization.openshift.io`** and **`localresourceaccessreviews.authorization.openshift.io`** if you need namespace-scoped requests.

Default Image Streams Now Use Pullthrough

The default image streams now use pullthrough. This means that the internal registry will pull these images on behalf of the user. If you modify the upstream location of the images in the image stream, the registry will pull from that location. This means the registry must be able to trust the upstream location. If your upstream location uses a certificate that is not part of the standard system trust store, pulls will fail. You will need to mount the appropriate trust store into the docker-registry pod to provide appropriate certificates in this case, in the ***/etc/tls*** directory path.

The image import process now runs inside a pod (the apiserver pod). Image import needs to trust registries it is importing from. If the source registry uses a certicate that is not signed by a CA that is in the standard system store, you will need to provide appropriate trust store information to the apiserver pod. This can be done by mounting content into to the pod's ***/etc/tls*** directory.

Use a Local Flag to Avoid Contacting the Server

In a future release, when invoking **oc** commands against a local file, you must use a **`--local`** flag when you do not want the client to contact the server.

Deprecated GitLab Versions

The use of self-hosted versions of GitLab with a version less than v11.1.0 is now deprecated. Users of self hosted versions should upgrade their GitLab installation as soon as possible. No action is required if the hosted version at gitlab.com is used, as that environment is always running the latest version.

Flexvolume Plug-in Updates

When using flexvolume for performing **`attach`**/**`detach`**, the flex binary must not have external dependencies and should be self contained. Flexvolume plug-in path on atomic hosts has been changed to ***/etc/origin/kubelet-plugins***, which applies to both master and compute nodes.

Deprecated oc rollout latest … --output=revision

In OpenShift Container Platform 3.10, **`oc rollout latest … --output=revision`** is deprecated. Use **`oc rollout latest … --output jsonpath={.status.latestVersion}`** or **`oc rollout latest … --output go-template={{.status.latestVersion}}`** instead.

CNS Is Now Red Hat OpenShift Container Storage (RHOCS)

Container Native Storage (CNS) is now called Red Hat OpenShift Container Storage (RHOCS). Previously, there was confusion between CNS and CRS terminology.

Builder Image Replaced

In OpenShift Container Platform 3.10, the **Atomic OpenShift Docker Builder**, **`registry.access.redhat.com/openshift3/ose-docker-builder`**, replaced the **Atomic OpenShift S2I Builder**, **`registry.access.redhat.com/openshift3/ose-sti-builder`**.

Previously, the **Atomic OpenShift Docker Builder** was responsible for executing Docker image builds. It now executes source-to-image (s2i) image builds as well.

2.5. BUG FIXES

This release fixes bugs for the following components:

Builds

- Some build container environment variables were modified when redacted in the container log. As a result, URL proxy settings (such as HTTP/S proxies) were modified, breaking these settings. A copy of these environment variables are made prior to redaction in the logs. (**BZ#11571349**)

- Streaming of build logs failed due to a server-side timeout waiting for the build pod to start. Therefore, **`oc start-build`** could hang if the **`--wait`** and **`--follow`** flags were set. With this bug fix:

  - Server-side timeout for a build pod to start was increased from 10 to 30 seconds.

  - If the **`--follow`** flag is specified and the log streaming fails, return an error message to the user.

  - If **`--follow`** and **`--wait`** is specified, retry log streaming.
    As a result:

  - Log stream failures due to build pod wait timeouts are less likely to occur.

  - If **`--follow`** fails, the user is presented with the message **Failed to stream the build logs - to view the logs, run oc logs build/<build-name>**.

  - If **`--follow`** and **`--wait`** flags are set, **`oc start-build`** will retry fetching the build logs until successful.
    (**BZ#1575990**)

- The build watch maintained by the **openshift jenkins sync** plug-in would no longer function, even while watchers on other API object types still functioned. The finding of a build would then fall upon the background build list thread, which by default runs at 5-minute intervals. This bug fix adds better logging around unexpected closure of the **openshift jenkins sync** plug-in watches, adds reconnect when those closures occur, and adds the ability for customers to configure the relist interval. Now, customers do not have to wait up to 5 minutes for the pipeline strategy builds to start. (**BZ#1554902**)

- The build controller was susceptible to incorrectly failing builds when time was not synchronized accurately between multiple masters. The controller logic is now improved to not depend on time synchronization. (**BZ#1547551**)

- The webhook payload can contain an empty commit array, which results in an array indexing error when processed by the APIserver. As a result, the API server crashes. Check for an empty array before attempting to index into it. With this bug fix, empty commit payloads are handled without crashing the API server. (**BZ#1585663**)

Containers

- An invalid SELinux context for the Docker engine prevented **`docker exec`** to work. With this bug fix, the issue is resolved. (**BZ#1517212**)

Image

- Jenkins would fail to parse certificates with the **Bag Attributes** preceding the **`BEGIN CERTIFICATE`** line and fail to start since the **openshift jenkins** image adds such a certificate to the Kubernetes cloud configuration. With this bug fix, remove the **Bag Attributes** preceding the

`BEGIN CERTIFICATE` line in the certificates mounted into the pod; generally validate the certificate for proper format. Jenkins can now start when such certificates are introduced. (BZ#1548619)

- A new value in the **`Reference`** field was not considered as a change. Therefore, the status field was not updated. This bug fix updates detection of changes. You can now set **`Reference: true`** and get any image reference in an image stream tag. (BZ#1555149)

- Additional certificate name constraints prevented valid certificates from being processed, resulting in an error of "tls: failed to parse certificate from server: x509: unhandled critical extension". As a result, valid certificates were unusable. By moving to newer golang libraries that fixed the constraint. certificates that previously failed can now be used. (BZ#1518583)

- Previously, PhantomJS would not install on **`jenkins-slave-base-rhel7`** image. this was because PhantomJS is packaged as **`tar.bz2`** archive and **`jenkins-slave-base-rhel7`** did not contain the bzip2 binary. The OpenShift Container Platform version 3.10, includes newer Jenkins image with bzip2 binary. (BZ#1544693)

Installer

- Due to a compatibility issue in earlier versions, the **`networkPluginName`** entry was listed twice in the **`node-config.yaml`**. The duplicate entry is no longer needed and has been removed. (BZ#1567970)

- Due to a change in the installer, if using images from a registry other than the default, you need to configure the registry using the **`oreg_url`** parameter in the ***/etc/ansible/hosts*** file for all components and images. Previously, you needed to configure the **`oreg_url`**, **`openshift_docker_additional_registries`**, and **`openshift_docker_insecure_registries`** parameters. (BZ#1516534)

- Environments where the Azure cloud provider is enabled now provision a default storage class for use with Azure storage. (BZ#1537479)

- You can now uninstall the Service Catalog using the Ansible Playbook if the **openshift-ansible-serivce-broker** project is not present. Previously, the uninstall playbook would fail if the project is not present. (BZ#1561485)

- Because NFS storage cannot provide the file system capabilities required by OpenShift registry, logging, and metrics components, a check has been added to the installer that will not allow NFS storage for these components. To use NFS storage for these components, you must opt-in by setting the cluster variable **`openshift_enable_unsupported_configurations`** to **`true`**, otherwise the procedure will fail. The use of NFS storage for registry, metrics, and logging components is only supported for proof of concept environments and not for production environments. (BZ#1416639)

- Ansible playbooks were taking too long to execute and could result in certificate errors from hosts that are not relevant to the task being performed. The playbooks have been modified to check only relevant hosts. (BZ#1516526)

- Ansible installer playbooks were creating persistent volumes before creating storage classes causing the playbooks to be run twice. The playbooks were changed to create the storage classes before any persistent volumes. (BZ#1564170)

- Because the way the OpenShift prefix and version were set for the console, the version reported by the console was different than the version displayed by other components. Control plane upgrade now ensures that the console version matches the version of other control plane

components. (BZ#1540427)

- Because of the Ansible installation playbooks, you needed to manually configure storage classes after installation in order to create PVCs. You can now configure storage classes at installation time by setting the following parameters in your inventory file:

```
openshift_storageclass_name=test-1
openshift_storageclass_provisioner=rbd
openshift_storageclass_parameters={'fstype': 'ext4', 'iopsPerGB':
'10', 'foo': 'bar'}
```

(BZ#1471718)

- The certificate expiration playbook, ***easy-mode.yaml***, was not checking all certificate files for expiration information. As a result, expired files were not being discovered, which could result in errors. The Ansible playbook has been updated. (BZ#1520971)

- Previously, dnsmasq was configured to listen on a specific IP address in an effort to avoid binding to **`127.0.0.1:53`**, which is where the node service runs its DNS service. This update configures dnsmasq to bind to all interfaces except the loopback, which ensures that dnsmasq works properly on hosts with multiple interfaces. (BZ#1481366)

- In rare cases, the router or registry **`registryurl`** variables may need to be set to values other than the first **`master registry_url`** value. This fix allows the **`openshift_hosted_router_registryurl`** and **`openshift_hosted_registry_registryurl`** variables to be set in the inventory. (BZ#1509853)

- A recent change in SELinux policy requires that an additional SEBoolean is set when running any pods with systemd which includes CFME. (BZ#1587825)

Logging

- The **`kube-`** and **`openshift-`** prefixes are preserved for internal use cases. to avoid name conflict, it is better to use the preserved prefix as default logging project. This fix uses the preserved prefix as the default logging project. This fits the pattern used by other infrastructure applications and allows the EFK stack to participate with other services that assume the infrastructure is deployed to namespaces with a known pattern (for example, **`openshift-`**). (BZ#1535300)

- A utility ***logging-dump.sh*** dumps the ElasticSearch logs as part of useful information for troubleshooting. In OpenShift Container Platform 3.10, the log location of ElasticSearch has been moved from ***/elasticsearch/logging-es[-ops]/logs*** to ***/elasticsearch/persistent/logging-es[-ops]/logs***. ***logging-dump.sh*** fails to dump ElasticSearch logs with a an error of **Unable to get ES logs from pod <ES_POD_NAME>**. In addition to ***/elasticsearch/logging-es[-ops]/logs***, check the new path ***/elasticsearch/persistent/logging-es[-ops]/logs*** for the logs files. With this bug fix, ***logging-dump.sh*** successfully dumps ElasticSearch logs. (BZ#1588416)

Web Console

- Previously, if a pod took more than five minutes to become ready, the web console would warn you, regardless of **`timeoutSeconds`** specified in the deployment configuration. For some applications, this period was too short. This fix removes this warning from the web console. (BZ#1550138)

- Prior to this release, the copy and paste operation in the web console container terminal did not work properly on Firefox and Internet Explorer. This fix updates `xterm.js` to `v3.1.0`. You can now copy and paste from the context menu or using keyboard shortcuts. (BZ#1278733)

- When the “No results match” result occurred in the Console or Catalog page, there were two links for clearing the search keys, “Clear Filters” and “Clear All Filters”. With this fix, all occurrences of “Clear Filters” were changed to “Clear All Filters”. Now there is one option to clear filters. (BZ#1549450)

- Different BuildConfig Webhook URLs were obtained by the CLI and Web Console. This caused the CLI to use the the correct **build.openshift.io** API group, while the Web Console did not use an API group. This fix updated the Webhook filter to use the correct **build.openshift.io** API group for the Web Console, and as a result the the correct URL for the BuildConfig Webhook is provided. (BZ#1559325)

- Manually typing a URL with a non-existing image, such as `/console/project/pro1/browse/images/non-existent-image`, caused the loading screen to freeze even though the process was finished and the alert, “The image stream details could not be loaded”, to be displayed. With this fix, the loaded scope variable is set when the image is or is not loaded and is used in the view to hide the loading screen. As a result, following the attempt to load the image data, the screen will not freeze on loading. (BZ#1550797)

- Previously, the web console did not support deploying an application with private repository image on the **Deploy Image** page. This is fixed and users can now deploy an app with a private repository image. (BZ#1489374)

Master

- Previously, DaemonSet nodes were restricted with project default node selector, causing the creation and deletion of DaemonSet pods in a loop on those nodes. This fix patched upstream DaemonSet logic to be aware of project default node selector. As a result, creation and deletion loop of DaemonSet pods on the nodes that got restricted by project default node selector is resolved. (BZ#1501514)

- Previously, the client was not able to read full discovery but was stuck on the first aggregated server which was temporarily unavailable. This led to not having the proper information about all the resources that were available. This fix introduced a default timeout for discovery actions. As a result, in case of a failure on an aggregated server the client will continue discovering resources on other servers and allow users to work with the ones that are available. (BZ#1525014)

- Previously, when pods that used DeploymentConfigs with the recreate strategy were evicted, a new pod did not come online until the timeout interval elapsed. Now the the recreate strategy creates a new pod even if evicted pods are present. (BZ#1549931)

Metrics

- Previously, the `auto_snapshot` parameter was set to `true` in the ***cassandra.yaml*** file, and because of changes to Hawkular Metrics introduced in OpenShift Container Platform 3.7, so many snapshots were generated that the disk might fill up. Now `auto_snapshot` is disabled by default, and snapshots are generated only if you set the `openshift_metrics_cassandra_take_snapshot` property to `true` in the Ansible inventory file. (BZ#1567251)

- Previously, you could not distribute multiple CA certificates to pods in the cluster. This limitation caused issues with load balancer configurations that required a different CA certificate than the one generated by the installer for the the master node. Now you can define the location of the

load balancer certificate in the **`openshift_additional_ca`** parameter during installation. The certificate is added to the ***/etc/origin/master/ca-bundle.crt*** file, which is made available to pods in the cluster. (**BZ#1535585**)

- In version 3.9, the Prometheus service account did not have the required permissions to access the metrics endpoint of the router, so Prometheus could not obtain the router's metrics. Now the Prometheus service account has the necessary additional role to access the metrics endpoint and can obtain metrics from the router. (**BZ#1565095**)

Networking

- Previously, the service controller sent a request to the cloud provider every time a service was created. This request checked whether the cloud provider had a load balancer for the service, even for non-LoadBalancer services. In clusters where many services were created, the extra requests dominated some cloud provider API usage. The service controller no longer sends this request to the cloud provider when a non-LoadBalancer service is created, which reduces the cloud provider API usage. (**BZ#1571940**)

- Previously, the egress router configuration prevented egress router pods from connecting to the public IP address of the nodes that host them. If an egress pod was configured to use its node as a name server in the ***/etc/resolv.conf*** file, DNS resolution failed. Traffic from an egress router pod to its node is now routed via the SDN tunnel instead of through the egress interface. Egress routers can now connect to their node's IP, and egress router DNS works. (**BZ#1552738**)

- If two nodes swapped IP addresses after you rebooted them, other nodes were sometimes unable to send traffic to pods on one or both of those nodes. Now, the OVS flow correctly manage node IP address reassignment, and pod-to-pod traffic continues even if nodes swap IP addresses. (**BZ#1538220**)

- Previously, changing an EgressIP of a NetNamespace while its existing EgressIP is active, assigned duplicate EgressIPs to the NetNamespaces of the same HostSubnets, resulting in egress IPs to stop working if an egress IP is moved from one project or node to another. Additionally, if the same egress IP is assigned to two different projects, or two different nodes, then it may not work correctly even after the duplicate assignment is removed. The EgressIPs field on a NetNamespace have been fixed to change while the egress IP is active. This results in static per-project egress IPs should work more reliably. (**BZ#1551028**)

- The kube-proxy and kubelet parts of the OpenShift node process were being given different default values for the configuration options describing how to interact with iptables. This resulted in OpenShift periodically add a false iptables rule that would cause some per-project static egress IPs to not be used for some length of time, until the false rule was removed again. While the bogus rule was present, traffic from those projects would use the node IP address of the node hosting the egress IP, rather than the egress IP itself. The inconsistent configuration was resolved, causing the false iptables rule to no longer be added, and projects now consistently use their static egress IPs. (**BZ#1552869**)

- Previously, OpenShift's default network plug-in did not contain the newest NetworkPolicy features introduced upstream in Kubernetes. These included policies for controlling egress, and policies based on IP addresses rather than pods or namespaces. This meant that in version 3.9, creating a NetworkPolicy with an **`ipBlock`** stanza would cause nodes to crash, and creating a NetworkPolicy that contained only "egress" rules would erroneously cause ingress traffic to be blocked. Now, OpenShift Container Platform is aware of the unsupported NetworkPolicy features, though it does not yet implement them, and if a NetworkPolicy contains **`ipBlock`** rules, those rules are ignored. This may cause the policy to be treated as "deny all" if the **`ipBlock`** rule was the only rule in the policy. If a NetworkPolicy contains only "egress" rules, it is ignored completely and does not affect ingress. (**BZ#1583255**)

- When deleting a pod, some of the IP files were not deleted as intended. This was caused by the garbage collection picking up a dead container. The kubelet keeps the information from at least one container in the case of if a restart is needed. This bug fix ensures that a proper clean up happens only if the network plug-in returns success, but some other error happens after that before the runtime (eg. dockershim or CRI-O) returns to kubelet. (BZ#1532965)

- Previously, the **dnsmasq** service would randomly freeze and would need a manual restart to start the resolution. This caused no logs to be captured for the **dnsmasq** service on OpenShift Container Platform node hosts. This was caused by the interface connecting with **dnsmasq** changing between releases, overloading the service. The **`dns-forward-max`** and **`cache-size`** option limits have been increased to 10000, and the service now works as expected. (BZ#1560489)

- The updated egress policy needed to block outgoing traffic, patch OVS flows, and then re-enable traffic. However, the OVS flow generation for DNS names was slow. This resulted in a few seconds of egress traffic downtime. With this bug fix, egress policy handling is updated to pre-populate all new OVS flows before blocking the outgoing traffic. This reduces the downtime during egress policy updates. (BZ#1558484)

- Due to incorrect cleanup of the internal state, if you deleted a "static per-project egress IPs" from one project and then tried to reuse that IP for a different project, the OVS rules for the new project would be created incorrectly. The egress IP would not be used for the new project, and might start being used again for some traffic from the old project. The internal state is now cleaned up correctly when removing an egress IP and egress traffic works as expected. (BZ#1543786)

- When using per-namespace static egress IPs, all external traffic is routed through the egress IP. *External* means all traffic that is not directed to another pod, and so includes traffic from the pod to the pod's node. When pods are told to use the node's IP address for DNS, and the pod is using a static egress IP, then DNS traffic will be routed to the egress node first, and then back to the original node, which might be configured to not accept DNS requests from other hosts, causing the pod to be unable to resolve DNS. Pod-to-node DNS requests now bypass the egress IP and go directly to the node and DNS works. (BZ#1557924)

Pod

- Previously, errors and warning messages for the **`oc describe`** command were not clear. This issue is fixed now. (BZ#1523778)

- Previously, the garbage collector tried to delete images that were in use by stopped containers. Changes are made in the OpenShift Container Platform version 3.10, which prevents garbage collector from attempting to remove images in use by stopped containers. (BZ#1577739)

- The **`cpu-cfs-quota`** used to get applied, even if the **`node-config.yaml`** file had **`cpu-cfs-quota`** set to **`false`**. This happened because the container cgroup for **`cfs`** quota was unbound, but the pod level cgroup was bounded. This issue is now fixed, changes were made so that the pod level cgroups remain unbounded. Now if **`cpu-cfs-quota`** is set to **`false`**, it ignores any limits from being enforced. (BZ#1581409)

- The web console was incorrectly assigning **`extensions/v1beta1`** as the API version when creating HPA resources, regardless of the actual group of the scale target. This issue is fixed. (BZ#1543043)

Routing

- Previously, the HAProxy config failed to load causing router to not service any routes. This was

because the Headless service had **`service.Spec.ClusterIP=None`** field set, which was not getting ignored as part of un-idling. This is fixed, the HAProxy config ignore headless services during unidle handling and the router service routes as expected. (**BZ#1567532**)

- Path based routes did not work as expected for mixed TLS scenarios. Splitting up of the route types into separate map files caused this issue. Causing haproxy to match the wrong route. Maps are now merged automatically and they are searched appropriately to correctly match the incoming requests with the corresponding backends. (**BZ#1534816**)

- When upgrading the HAProxy Docker image, no logging of requests occurs by default. If logging was requested using the **`httplog`** option, a warning message was shown, as this option is not available on a TCP-only connection. In this situation, HAProxy will fall back to using the **`tcplog`** option instead. The warning message is therefore harmless and has been removed. (**BZ#1533346**)

Service Broker

- The **`type: openshift`** registry adapter does not support discovery of APB images. This means that users of this registry adapter must manually include a list of images to bootstrap. This enhancement introduces the use of a new registry adapter, **`type: partner_rhcc`**, which works with https://registry.connect.redhat.com, and supports image discovery without this manual requirement. (**BZ#1576881**)

- When attempting to deprovision a service instance, an error was occurring during the process, combined with an invalid response body, which was causing the deprovision process to fail. Changes have now been implemented to return the proper response body with the operation key, and to enhance the overall robustness of the deprovisioning workflow, which will increase the likelihood of successful deprovisioning. (**BZ#1562732**)

- According to the Open Source Broker (OSB) API documentation, if a binding exists, a status of **`200 OK`** must be returned from a binding call. An issue had been occurring where an incorrect response code (**`201`**) was being returned instead. This issue has been fixed by introducing support for asynchronous bindings. (**BZ#1563560**)

Service Catalog

- When a service class is removed from a provisioned service instance in the service broker's catalog, the service catalog marks the class as **`removedFromBrokerCatalog: true`**. This prevents the class from being used in new service plans or instances. An issue was preventing this status from being reset to **`removedFromBrokerCatalog: false`** if the service class is re-added to the broker catalog, and was preventing removed classes from being used again later. This issue is now resolved. (**BZ#1548122**)

- The Prometheus console did not previously allow access to service catalog metrics, which were only available using curl in the back end. The service catalog controller now exposes metrics for Prometheus to scrape, which enables monitoring of the service catalog. (**BZ#1549021**)

Storage

- The capacity of the local persistent storage volume (PV) was being reported in some cases as different to that reported by the **`df`** utility. This was due to a lack of propagation of newly mounted devices to the pods, resulting in an additional PV being created for the configured directory. The capacity of this newly created PV was equal to the root device. This propagation issue has now been fixed. (**BZ#1490722**)

- When the API call quota for in the AWS cloud was reached, certain AWS API calls returned

errors. These errors were not correctly handled when detaching AWS persistent Volumes, with some AWS volumes remaining attached to the nodes despite there being no pod using them. The volumes had to be detached manually, otherwise they became stuck forever. This bug fix updates the AWS API call error handling for dynamic volume detaching. As a result, even when the AWS API call quota is reached, the attach/detach controller re-tries to detach the volume until it succeeds, ensuring the volumes that should be detached are actually detached. (**BZ#1537236**)

Testing

- Previously, when **`masterConfig.ImagePolicyConfig.ExternalRegistryHostname`** was added for the ***master-config.yaml*** and the API and controller service was restarted, the API pod would recreate, but the controllers pod would error with **`CrashLoopBackOff`**. Having an **`m3.large`** instance in AWS resolves the issue. (**BZ#1593635**)

Upgrading

- You can now define a set of hooks to run arbitrary tasks during the node upgrade process. To implement these hooks set **`openshift_node_upgrade_pre_hook`**, **`openshift_node_upgrade_hook`**, or **`openshift_node_upgrade_post_hook`** to the path of the task file you wish to execute. The **`openshift_node_upgrade_pre_hook`** hook is executed after draining the node and before it has been upgraded. The **`openshift_node_upgrade_hook`** is executed after the node has been drained and packages updated but before it is marked schedulable again. The **`openshift_node_upgrade_post_hook`** hook is executed after the node has been marked schedulable immediately before moving on to other nodes. (**BZ#1559143**)

2.6. TECHNOLOGY PREVIEW FEATURES

Some features in this release are currently in Technology Preview. These experimental features are not intended for production use. Please note the following scope of support on the Red Hat Customer Portal for these features:

Technology Preview Features Support Scope

In the table below, features marked **TP** indicate *Technology Preview* and features marked **GA** indicate *General Availability*.

Table 2.1. Technology Preview Tracker

Feature	OCP 3.7	OCP 3.9	OCP 3.10
Prometheus Cluster Monitoring	TP	TP	TP
Local Storage Persistent Volumes	TP	TP	TP
CRI-O for runtime pods	TP	GA* [a]	GA
Tenant Driven Snapshotting	TP	TP	TP

Feature	OCP 3.7	OCP 3.9	OCP 3.10
oc CLI Plug-ins	TP	TP	TP
Service Catalog	GA	GA	GA
Template Service Broker	GA	GA	GA
OpenShift Automation Broker	GA	GA	GA
Network Policy	GA	GA	GA
Service Catalog Initial Experience	GA	GA	GA
New Add Project Flow	GA	GA	GA
Search Catalog	GA	GA	GA
CFME Installer	GA	GA	GA
Cron Jobs	TP	GA	GA
Kubernetes Deployments	TP	GA	GA
StatefulSets	TP	GA	GA
Explicit Quota	TP	GA	GA
Mount Options	TP	GA	GA
System Containers for docker, CRI-O	TP	Dropped	-
Installing from a system container	TP	GA	GA
Hawkular Agent	Dropped	-	-
Pod PreSets	Dropped	-	-
experimental-qos-reserved	TP	TP	TP
Pod sysctls	TP	TP	TP

Feature	OCP 3.7	OCP 3.9	OCP 3.10
Central Audit	TP	GA	GA
Static IPs for External Project Traffic	TP	GA	GA
Template Completion Detection	TP	GA	GA
`replicaSet`	TP	GA	GA
Mux	TP	TP	TP
Clustered MongoDB Template	Community	-	-
Clustered MySQL Template	Community	-	-
Image Streams with Kubernetes Resources	TP	GA	GA
Device Manager	-	TP	GA
Persistent Volume Resize	-	TP	TP
Huge Pages	-	TP	GA
CPU Manager	-	TP	GA
Device Plug-ins	-	TP	GA
syslog Output Plug-in for fluentd	-	GA	GA
Container Storage Interface (CSI)	-	-	TP
Persistent Volume (PV) Provisioning Using OpenStack Manila	-	-	TP
Node Problem Detector	-	-	TP
Protection of Local Ephemeral Storage	-	-	TP

Feature	OCP 3.7	OCP 3.9	OCP 3.10
Descheduler	-	-	TP
Podman	-	-	TP
Kuryr CNI Plug-in	-	-	TP
Sharing Control of the PID Namespace	-	-	TP

[a] Features marked with * indicate delivery in a z-stream patch.

2.7. KNOWN ISSUES

- There is one known Kubelet wedge state that will be fixed in the 1.10 rebase where the Kubelet will display messages like **system:anonymous cannot access resource foo**. This means that the certificates expired before the kubelet could refresh them. If restarting the kubelet does not fix the issue, delete the contents of ***/etc/origin/node/certificates/***, and then restart the kubelet.

- The blue-green node deployment method as documented in Upgrading Clusters should only be used for the initial upgrade path from OpenShift Container Platform 3.9 to 3.10. It will be further updated when the first asynchronous OpenShift Container Platform 3.10.z update is released.

- In the GA release of the Downgrading OpenShift documentation, an issue was found with the steps for restoring etcd. The document has since been updated and this is no longer an issue.

- OpenShift Container Platform 3.10 adds the ability for multiple instances of an APB to be invoked in the same namespace. This new ability requires relying on a globally unique identifier (GUID) for each instance. While instances deployed by an 3.9 version of an APB lack the GUID, 3.10 APBs require it.
  A 3.10 APB is unable to manage a 3.9 deployed service because it lacks the newly required GUID. This causes clusters upgraded from 3.9 to 3.10 to result in an error if an application that was previously deployed on 3.9 is then deprovisioned from 3.10 APB.

  There are two workarounds to this issue currently:

  - After the cluster upgrade to 3.10 has completed, delete the namespace of the application and recreate it. This will use the 3.10 version of the APB and function as expected.

  - Modify the configuration of the OpenShift Ansible broker to remain on the 3.9 version of APBs. This is not recommended, however, as it has the downside of the broker using 3.10 code while the APBs use the older 3.9 version:

    a. Follow the procedure in Modifying the OpenShift Ansible Broker Configuration to change the label to `v3.9`.

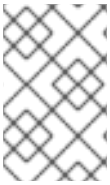
    b. Run the `apb bootstrap` command to bootstrap the broker and relist the catalog.

  (**BZ#1586108**)

2.8. ASYNCHRONOUS ERRATA UPDATES

Security, bug fix, and enhancement updates for OpenShift Container Platform 3.10 are released as asynchronous errata through the Red Hat Network. All OpenShift Container Platform 3.10 errata is available on the Red Hat Customer Portal. See the OpenShift Container Platform Life Cycle for more information about asynchronous errata.

Red Hat Customer Portal users can enable errata notifications in the account settings for Red Hat Subscription Management (RHSM). When errata notifications are enabled, users are notified via email whenever new errata relevant to their registered systems are released.



NOTE

Red Hat Customer Portal user accounts must have systems registered and consuming OpenShift Container Platform entitlements for OpenShift Container Platform errata notification emails to generate.

This section will continue to be updated over time to provide notes on enhancements and bug fixes for future asynchronous errata releases of OpenShift Container Platform 3.10. Versioned asynchronous releases, for example with the form OpenShift Container Platform 3.10.z, will be detailed in subsections. In addition, releases in which the errata text cannot fit in the space provided by the advisory will be detailed in subsections that follow.

IMPORTANT

For any OpenShift Container Platform release, always review the instructions on upgrading your cluster properly.

2.8.1. RHBA-2018:2376 - OpenShift Container Platform 3.10.34 Bug Fix and Enhancement Update

Issued: 2018-08-28

OpenShift Container Platform release 3.10.34 is now available. The list of packages and bug fixes included in the update are documented in the RHBA-2018:2376 advisory. The container images included in the update are provided by the RHBA-2018:2377 advisory.

Space precluded documenting all of the bug fixes and enhancements for this release in the advisory. See the following sections for notes on upgrading and details on the bug fixes and enhancements included in this release.

2.8.1.1. Bug Fixes

- When Mux is configured and it fails to find a project or namespace that a log belongs to, the log was indexed into **`project.mux-undefined`** where **`mux-undefined`** was a Mux default namespace. At the same time, the fluentd (without Mux configuration) puts such logs into the ***.orphaned.YYYY.MM.DD*** index. With this bug fix, such orphaned logs are also indexed into the ***.orphaned.YYYY.MM.DD*** index for the Mux case. (**BZ#1538560**)

- The installer was creating an incorrect **`spec`** attribute for CPU and memory. Additionally, it did not allow modifying the CPU limit. Therefore, the values were ignored. Conditionally patch in **`cpu_limit`** if it is defined and correct the attribute name used to specify CPU and memory requests. With this bug fix, the values are honored as expected. (**BZ#1575546**)

- The Ansible template did not quote the value in the selector, producing invalid JSON. The selector value is now quoted and a PVC can be created with the selector. (BZ#1597282)

- The 9100 port is blocked on all nodes by default. Prometheus can not scrape the **node_exporter** service running on the other nodes, which listens on port 9100. The firewall configuration is now modified to allow incoming TCP traffic for the 9000-1000 port range and Prometheus can scrape the **node_exporter** services. (BZ#1600562)

- Recently, `cloudResourceSyncManager` was implemented, which continuously fetched node addresses from cloud providers. Kubelet then received node addresses from the `cloudResourceSyncManager`. At the time of node registration or kubelet start, kubelet fetches node addresses in a blocking loop from `cloudResourceSyncManager`. The issue was that `cloudResourceSyncManager` was not started before kubelet had started fetching node addresses from it for the first time and, due to this, kubelet got stuck in the blocking loop and never returned. It caused node failures at network level, and no node could be registered. Also, as kubelet blocked early, the `cloudResourceSyncManager` never got a chance to start. `CloudResourceSyncManager` is now started early in the kubelet start up process so that kubelet does not get blocked on it and `cloudResourceSyncManager` is always started. (BZ#1603611)

- If a node selector was provided as a value of `true`, it was interpreted as a boolean and would cause daemonset deployment to fail. The template for creating the daemonset is now updated to quote the provided value to ensure it is interpreted as a string. (BZ#1609027)

- Groups associated with a user were not checked when performing access checks to look up the readiness of objects created by the templates. For objects the user could only access due to their group membership, objects would be created by the template, but could not be checked for readiness, resulting in a readiness failure at the template instance level. Pass the user's groups when performing the readiness check operation, not just when performing the object creation. Objects can now successfully be checked for readiness as long as the user's group membership permits the check. (BZ#1610994)

- There was a race condition when piping output from a tar stream extraction. Binary builds with large numbers of files could hang indefinitely. The tar streaming logic is now reverted to use a previous mechanism, which does not have a race condition. Binary builds with large numbers of files now complete normally. (BZ#1614493)

- By default, older versions of dnsmasq can use privileged, lower-numbered source ports for outbound DNS queries. Outbound DNS queries could be dropped; for example, firewall rules might drop queries coming from reserved ports. dnsmasq is now configured using its `min-port` setting to set the minimum port number for outbound queries to `1024`. DNS queries should no longer be dropped. (BZ#1614984)

- Ansible 2.6.0 will not evaluate undefined variables with `|bool` as `false`. You must define a `| default(false)` for `logging_elasticsearch_rollout_override`. With this bug fix, the playbook executes successfully. (BZ#1615194)

2.8.1.2. Enhancements

- The default fluentd memory is increased to `756m`. Performance and scaling testing demonstrated that fluentd requires more memory after recent improvements to avoid out of memory failures and container restarts. Fluentd is now less likely to run out of memory. (BZ#1600258)

- During an upgrade, a check is performed to see if the node is running CRI-O as a system container. If so, the CRI-O system container is uninstalled and the CRI-O RPM is installed. Running CRI-O as a system container is unsupported. In some cases during a OpenShift

Container Platform 3.9 installation, a node may inadvertently be installed with CRI-O as a system container. Nodes upgraded from OpenShift Container Platform 3.9 to 3.10 will be converted to a supported configuration with CRI-O running from an RPM. (**BZ#1618425**)

2.8.1.3. Upgrading

To upgrade an existing OpenShift Container Platform 3.9 or 3.10 cluster to this latest release, use the automated upgrade playbook. See Performing Automated In-place Cluster Upgrades for instructions.

2.8.2. RHBA-2018:2660 - OpenShift Container Platform 3.10.45 Bug Fix and Enhancement Update

Issued: 2018-09-24

OpenShift Container Platform release 3.10.45 is now available. The list of packages and bug fixes included in the update are documented in the RHBA-2018:2660 advisory. The container images included in the update are provided by the RHBA-2018:2661 advisory.

2.8.2.1. Upgrading

To upgrade an existing OpenShift Container Platform 3.9 or 3.10 cluster to this latest release, use the automated upgrade playbook. See Performing Automated In-place Cluster Upgrades for instructions.

2.8.3. RHBA-2018:2738 - OpenShift Container Platform 3.10.45 Images Update

Issued: 2018-09-24

The list of container images included in the update are documented in the RHBA-2018:2738 advisory.

The container images in this release have been updated using the latest base images.

2.8.3.1. Images

This release updates the Red Hat Container Registry (**`registry.access.redhat.com`**) with the following images:

```
openshift3/metrics-hawkular-openshift-agent:v3.10.45-5
openshift3/metrics-heapster:v3.10.45-5
```

2.8.4. RHBA-2018:2869 - OpenShift Container Platform PowerPC Packages and Images Update

Issued: 2018-10-03

The list of packages and bug fixes included in the update are documented in the RHBA-2018:2869 advisory. The container images included in the update are provided by the RHBA-2018:2708 advisory.

2.8.4.1. Images

This release updates the Red Hat Container Registry (**`registry.access.redhat.com`**) with the following images:

```
openshift3/apb-base:v3.10.45-6
```

```
openshift3/apb-tools:v3.10.45-3
openshift3/csi-attacher:v3.10.45-5
openshift3/csi-driver-registrar:v3.10.45-5
openshift3/csi-livenessprobe:v3.10.45-5
openshift3/csi-provisioner:v3.10.45-5
openshift3/image-inspector:v3.10.45-5
openshift3/jenkins-2-rhel7:v3.10.45-7
openshift3/jenkins-agent-maven-35-rhel7:v3.10.45-7
openshift3/jenkins-agent-nodejs-8-rhel7:v3.10.45-7
openshift3/jenkins-slave-base-rhel7:v3.10.45-9
openshift3/local-storage-provisioner:v3.10.45-5
openshift3/logging-curator:v3.10.45-5
openshift3/logging-elasticsearch:v3.10.45-5
openshift3/logging-eventrouter:v3.10.45-5
openshift3/logging-fluentd:v3.10.45-5
openshift3/logging-kibana:v3.10.45-7
openshift3/manila-provisioner:v3.10.45-5
openshift3/mariadb-apb:v3.10.45-5
openshift3/mediawiki-apb:v3.10.45-5
openshift3/mediawiki:v3.10.45-5
openshift3/metrics-hawkular-openshift-agent:v3.10.45-7
openshift3/metrics-heapster:v3.10.45-7
openshift3/mysql-apb:v3.10.45-5
openshift3/node:v3.10.45-7
openshift3/oauth-proxy:v3.10.45-5
openshift3/ose-ansible-service-broker:v3.10.45-4
openshift3/ose-ansible:v3.10.45-4
openshift3/ose-cli:v3.10.45-7
openshift3/ose-cluster-capacity:v3.10.45-6
openshift3/ose-deployer:v3.10.45-6
openshift3/ose-descheduler:v3.10.45-5
openshift3/ose-docker-builder:v3.10.45-5
openshift3/ose-docker-registry:v3.10.45-6
openshift3/ose-egress-dns-proxy:v3.10.45-6
openshift3/ose-egress-http-proxy:v3.10.45-6
openshift3/ose-egress-router:v3.10.45-6
openshift3/ose-f5-router:v3.10.45-6
openshift3/ose-haproxy-router:v3.10.45-6
openshift3/ose-hyperkube:v3.10.45-6
openshift3/ose-hypershift:v3.10.45-6
openshift3/ose-keepalived-ipfailover:v3.10.45-6
openshift3/ose-node-problem-detector:v3.10.45-5
openshift3/ose-pod:v3.10.45-5
openshift3/ose-recycler:v3.10.45-6
openshift3/ose-service-catalog:v3.10.45-6
openshift3/ose-template-service-broker:v3.10.45-5
openshift3/ose-web-console:v3.10.45-6
openshift3/ose:v3.10.45-7
openshift3/postgresql-apb:v3.10.45-5
openshift3/prometheus-alert-buffer:v3.10.45-5
openshift3/prometheus-alertmanager:v3.10.45-5
openshift3/prometheus-node-exporter:v3.10.45-5
openshift3/prometheus:v3.10.45-5
openshift3/registry-console:v3.10.45-5
openshift3/snapshot-controller:v3.10.45-5
openshift3/snapshot-provisioner:v3.10.45-5
```

2.8.4.2. Upgrading

To upgrade an existing OpenShift Container Platform 3.9 or 3.10 cluster to this latest release, use the automated upgrade playbook. See Performing Automated In-place Cluster Upgrades for instructions.

CHAPTER 3. XPAAS RELEASE NOTES

The release notes for xPaaS docs have migrated to their own book on the Red Hat customer portal.

CHAPTER 4. COMPARING WITH OPENSHIFT ENTERPRISE 2

4.1. OVERVIEW

OpenShift Container Platform 3 is based on the OpenShift version 3 (v3) architecture, which is very different product than OpenShift version 2 (v2). Many of the same terms from OpenShift v2 are used in v3, and the same functions are performed, but the terminology can be different, and behind the scenes things may be happening very differently. Still, OpenShift remains an application platform.

This topic discusses these differences in detail, in an effort to help OpenShift users in the transition from OpenShift v2 to OpenShift v3.

4.2. ARCHITECTURE CHANGES

Gears Versus Containers

Gears were a core component of OpenShift v2. Technologies such as kernel namespaces, cGroups, and SELinux helped deliver a highly-scalable, secure, containerized application platform to OpenShift users. Gears themselves were a form of container technology.

OpenShift v3 takes the gears idea to the next level. It uses Docker as the next evolution of the v2 container technology. This container architecture is at the core of OpenShift v3.

Kubernetes

As applications in OpenShift v2 typically used multiple gears, applications on OpenShift v3 will expectedly use multiple containers. In OpenShift v2, gear orchestration, scheduling, and placement was handled by the OpenShift broker host. OpenShift v3 integrates Kubernetes into the master host to drive container orchestration.

4.3. APPLICATIONS

Applications are still the focal point of OpenShift. In OpenShift v2, an application was a single unit, consisting of one web framework of no more than one cartridge type. For example, an application could have one PHP and one MySQL, but it could not have one Ruby, one PHP, and two MySQLs. It also could not be a database cartridge, such as MySQL, by itself.

This limited scoping for applications meant that OpenShift performed seamless linking for all components within an application using environment variables. For example, every web framework knew how to connect to MySQL using the **`OPENSHIFT_MYSQL_DB_HOST`** and **`OPENSHIFT_MYSQL_DB_PORT`** variables. However, this linking was limited to within an application, and only worked within cartridges designed to work together. There was nothing to help link across application components, such as sharing a MySQL instance across two applications.

While most other PaaSes limit themselves to web frameworks and rely on external services for other types of components, OpenShift v3 makes even more application topologies possible and manageable.

OpenShift v3 uses the term "application" as a concept that links services together. You can have as many components as you desire, contained and flexibly linked within a project, and, optionally, labeled to provide grouping or structure. This updated model allows for a standalone MySQL instance, or one shared between JBoss components.

Flexible linking means you can link any two arbitrary components together. As long as one component can export environment variables and the second component can consume values from those

environment variables, and with potential variable name transformation, you can link together any two components without having to change the images they are based on. So, the best containerized implementation of your desired database and web framework can be consumed directly rather than you having to fork them both and rework them to be compatible.

This means you can build anything on OpenShift. And that is OpenShift's primary aim: to be a container-based platform that lets you build entire applications in a repeatable lifecycle.

4.4. CARTRIDGES VERSUS IMAGES

In OpenShift v3, an image has replaced OpenShift v2's concept of a cartridge.

Cartridges in OpenShift v2 were the focal point for building applications. Each cartridge provided the required libraries, source code, build mechanisms, connection logic, and routing logic along with a preconfigured environment to run the components of your applications.

However, cartridges came with disadvantages. With cartridges, there was no clear distinction between the developer content and the cartridge content, and you did not have ownership of the home directory on each gear of your application. Also, cartridges were not the best distribution mechanism for large binaries. While you could use external dependencies from within cartridges, doing so would lose the benefits of encapsulation.

From a packaging perspective, an image performs more tasks than a cartridge, and provides better encapsulation and flexibility. However, cartridges also included logic for building, deploying, and routing, which do not exist in images. In OpenShift v3, these additional needs are met by Source-to-Image (S2I) and configuring the template.

Dependencies

In OpenShift v2, cartridge dependencies were defined with **`Configure-Order`** or **`Requires`** in a cartridge manifest. OpenShift v3 uses a declarative model where pods bring themselves in line with a predefined state. Explicit dependencies that are applied are done at runtime rather than just install time ordering.

For example, you might require another service to be available before you start. Such a dependency check is always applicable and not just when you create the two components. Thus, pushing dependency checks into runtime enables the system to stay healthy over time.

Collection

Whereas cartridges in OpenShift v2 were colocated within gears, images in OpenShift v3 are mapped 1:1 with containers, which use pods as their colocation mechanism.

Source Code

In OpenShift v2, applications were required to have at least one web framework with one Git repository. In OpenShift v3, you can choose which images are built from source and that source can be located outside of OpenShift itself. Because the source is disconnected from the images, the choice of image and source are distinct operations with source being optional.

Build

In OpenShift v2, builds occurred in application gears. This meant downtime for non-scaled applications due to resource constraints. In v3, builds happen in separate containers. Also, OpenShift v2 build results used rsync to synchronize gears. In v3, build results are first committed as an immutable image and

published to an internal registry. That image is then available to launch on any of the nodes in the cluster, or available to rollback to at a future date.

Routing

In OpenShift v2, you had to choose up front as to whether your application was scalable, and whether the routing layer for your application was enabled for high availability (HA). In OpenShift v3, routes are first-class objects that are HA-capable simply by scaling up your application component to two or more replicas. There is never a need to recreate your application or change its DNS entry.

The routes themselves are disconnected from images. Previously, cartridges defined a default set of routes and you could add additional aliases to your applications. With OpenShift v3, you can use templates to set up any number of routes for an image. These routes let you modify the scheme, host, and paths exposed as desired, with no distinction between system routes and user aliases.

4.5. BROKER VERSUS MASTER

A master in OpenShift v3 is similar to a broker host in OpenShift v2. However, the MongoDB and ActiveMQ layers used by the broker in OpenShift v2 are no longer necessary, because **etcd** is typically installed with each master host.

4.6. DOMAIN VERSUS PROJECT

A project is essentially a v2 domain.