OpenShift Container Platform 3.11

Release Notes

Last Updated: 2018-10-12

Legal Notice

Abstract

Table of Contents

CHAPTER 1. OVERVIEW

The following release notes for OpenShift Container Platform 3.11 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 3.11 cluster, all masters must be 3.11 and all nodes must be 3.11. 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 3.11 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 more secure and scalable multi-tenant operating system for today's enterprise-class applications, while delivering 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 3.11 (RHBA-2018:2652) is now available. This release is based on OKD 3.11, and it uses Kubernetes 1.11. New features, changes, bug fixes, and known issues that pertain to OpenShift Container Platform 3.11 are included in this topic.

OpenShift Container Platform 3.11 is supported on Red Hat Enterprise Linux 7.4 and 7.5 with the latest packages from Extras, including CRI-O 1.11 and Docker 1.13. It is also supported on Atomic Host 7.5 and newer.

For initial installations, see the Installing Clusters documentation.

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

WARNING

In the initial release of OpenShift Container Platform version 3.11, downgrading does not completely restore your cluster to version 3.10. Do not downgrade.

This function will be restored in a future z-stream release of version 3.11 with **BZ#1638004**.

2.2.1. Major Changes Coming in Version 4.0

OpenShift Container Platform 3.11 is the last release in the 3.x stream. Large changes to the underlying architecture and installation process are coming in version 4.0, and many features will be deprecated.

Table 2.1. Features Deprecated in Version 4.0

Feature	Justification
Hawkular	Replaced by Prometheus monitoring.

Feature	Justification
Cassandra	Replaced by Prometheus monitoring.
Heapster	Replaced by Metrics-Server or Prometheus metrics adapter.
Atomic Host	Replaced by Red Hat CoreOs.
System containers	Replaced by Red Hat CoreOs.
projectatomic/docker-1.13 additional search registries	CRI-O is the default container runtime for 4.x on RHCOS and Red Hat Enterprise Linux.
`oc adm diagnostics`	Operator-based diagnostics.
`oc adm registry`	Replaced by the registry operator.
Custom Docker Build Strategy on Builder Pods	If you want to continue using custom builds, you must replace your Docker invocations with Podman and Buildah. The custom build strategy will not be removed, but the functionality will change significantly in OpenShift Container Platform 4.0.
Cockpit	Replaced by Quay.
Standalone Registry Installations	Quay is our enterprise container image registry.
DNSmasq	CoreDNS will be the default.
External etcd nodes	For 4.0, etcd is on the cluster always.
CFME OpenShift Provider and Podified CFME	Replaced by Prometheus.
Volume Provisioning via installer	Replaced by dynamic volumes or, if NFS is required, NFS provisioner.
blue-green-installation method	Ease of upgrade is a core value of 4.0.

Because of the extent of the changes in OpenShift Container Platform 4.0, the product documentation will also undergo significant changes, including the deprecation of large amounts of content. New content will be released based on the architectural changes and updated use cases.

2.3. NEW FEATURES AND ENHANCEMENTS

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

2.3.1. Operators

2.3.1.1. Operator Lifecycle Manager (OLM) (Technology Preview)

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

The OLM aids cluster administrators in installing, upgrading, and granting access to Operators running on their cluster:

- Includes a catalog of curated Operators, with the ability to load other Operators into the cluster
- Handles rolling updates of all Operators to new versions
- Supports role-based access control (RBAC) for certain teams to use certain Operators

See Installing the Operator Framework for more information.

2.3.1.2. Operator SDK

The Operator SDK is a development tool to jump-start building an Operator with generated code and a CLI to aid in building, testing, and publishing your Operator. The Operator SDK:

- Provides tools to get started quickly embedding application business logic into an Operator
- Saves you from doing the work to set up scaffolding to communicate with the Kubernetes API
- Helps run end-to-end tests of your logic on a local or remote cluster
- Is used by Couchbase, MongoDB, Redis and more

See Getting started with the Operator SDK in OKD documentation for more information and walkthroughs.

2.3.2. Brokers

Brokers mediate service requests in the Service Catalog. The goal is for you to initiate the request and for the system to fulfill the request in an automated fashion.

2.3.2.1. OpenShift Container Platform Automation Broker Integration with Ansible Galaxy

The Automation Broker manages applications defined in Ansible Playbook Bundles (APB). OpenShift Container Platform 3.11 includes support for discovering and running APB sources published to Ansible Galaxy from the OpenShift Container Platform Automation Broker.

See OpenShift Automation Broker for more information.

2.3.2.2. Broker Support for Authenticated Registries

The Red Hat Container Catalog is moving from **`registry.access.redhat.com`** to **`registry.redhat.io`**. **`registry.redhat.io`** requires authentication for access to images and hosted content on OpenShift Container Platform.

OpenShift Container Platform 3.11 adds support for authenticated registries. The broker uses **`cluster-wide`** as the default setting for registry authentication credentials. You can define **`oreg_auth_user`** and **`oreg_auth_password`** in the inventory file to configure the credentials.

2.3.2.3. Service Catalog Namespaced Brokers

The Service Catalog added support for namespaced brokers in addition to the previous cluster scoped behavior. This means you can register the broker with the service catalog as either a cluster-scoped **`ClusterServiceBroker`** or a namespace-scoped **`ServiceBroker`** kind. Depending on the broker's scope, its services and plans are available to the entire cluster or scoped to a specific namespace. When installing the broker, you can set the **`kind`** argument as **`ServiceBroker`** (namespace-specific) or **`ClusterServiceBroker`** (cluster-wide).

2.3.3. Installation and Upgrade

2.3.3.1. Checks for Expiring Certificates During Upgrade

In OpenShift Container Platform 3.11, **`openshift_certificate_expiry_warning_days`**, which indicates the amount of time the auto-generated certificates must be valid for an upgrade to proceed, is added.

Additionally, **`openshift_certificate_expiry_fail_on_warn`** is added, which determines whether the upgrade fails if the auto-generated certificates are not valid for the period specified by the **`openshift_certificate_expiry_warning_days`** parameter.

See Configuring Your Inventory File for more information.

2.3.3.2. Support for Ansible 2.6

`openshift-ansible` now requires Ansible 2.6 for both installation of OpenShift Container Platform 3.11 and upgrading from version 3.10.

The minimum version of Ansible required for OpenShift Container Platform 3.11 to run playbooks is now 2.6.x. On both master and node, use **`subscription-manager`** to enable the repositories that are necessary to install OpenShift Container Platform using Ansible 2.6. For example:

```
$ subscription-manager repos --enable="rhel-7-server-rpms" \
    --enable="rhel-7-server-extras-rpms" \
    --enable="rhel-7-server-ose-3.11-rpms" \
    --enable="rhel-7-server-ansible-2.6-rpms"
```

Ansible 2.7 is not yet supported.

2.3.3.3. Registry Auth Credentials Are Now Required

Registry auth credentials are now required for OpenShift Container Platform so that images and metadata can be pulled from an authenticated registry, registry.redhat.io.

Registry auth credentials are required prior to installing and upgrading when:

- **`openshift_deployment_type`** == **`'openshift-enterprise'`**
- **`oreg_url`** == **`'registry.redhat.io'`** or undefined

To configure authentication, **`oreg_auth_user`** and **`oreg_auth_password`** must be defined in the inventory file.

Pods can also be allowed to reference images from other secure registries.

See Importing Images from Private Registries for more information.

2.3.3.4. Customer Installations Are Now logged

Ansible configuration is now updated to ensure OpenShift Container Platform installations are logged by default.

The Ansible configuration parameter **`log_path`** is now defined. Users must be in the ***/usr/share/ansible/openshift-ansible*** directory prior to running any playbooks.

2.3.4. Storage

2.3.4.1. Container Storage Interface (Technology Preview)

This feature 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.

2.3.4.2. Protection of Local Ephemeral Storage (Technology Preview)

This feature 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. This helps 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.

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

This feature 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.

2.3.4.4. Persistent Volume (PV) Resize (Technology Preview)

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

You can expand PV claims online from OpenShift Container Platform for GlusterFS by creating a storage class with **`allowVolumeExpansion`** set to **`true`**, which causes the following to happen:

1. The PVC uses the storage class and submits a claim.
2. The PVC specifies a new increased size.
3. The underlying PV is resized.

Block storage volume types such as GCE-PD, AWS-EBS, Azure Disk, Cinder, and Ceph RBD typically require a file system expansion before the additional space of an expanded volume is usable by pods. Kubernetes takes care of this automatically whenever the pod or pods referencing your volume are restarted.

Network attached file systems, such as GlusterFS and Azure File, can be expanded without having to restart the referencing pod, as these systems do not require unique file system expansion.

See Expanding Persistent Volumes for more information.

2.3.4.5. Tenant-driven Storage Snapshotting (Technology Preview)

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

Tenants can now leverage the underlying storage technology backing the 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.

You can use an external provisioner to access EBS, GCE pDisk, and hostPath. This Technology Preview feature has tested EBS and hostPath. The tenant must stop the pods and start them manually.

To use the external provisioner to access EBS and hostPath:

1. The administrator runs an external provisioner for the cluster. These are images from the Red Hat Container Catalog.
2. The tenant creates a PV claim and owns a PV from one of the supported storage solutions.
3. The administrator must create a new **`StorageClass`** in the cluster, for example:

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

4. The tenant creates a snapshot of a PV claim named **`gce-pvc`**, and the resulting snapshot is **`snapshot-demo`**, for example:

```
$ oc create -f snapshot.yaml

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

5. The pod is restored to that snapshot, for example:

```
$ 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
```

2.3.5. Scale

2.3.5.1. Cluster Limits

Updated guidance around Cluster Limits for OpenShift Container Platform 3.11 is now available.

New recommended guidance for master

For large or dense clusters, the API server might get overloaded because of the default queries per second (QPS) limits. Edit ***/etc/origin/master/master-config.yaml*** and double or quadruple the QPS limits.

See Recommended Practices for OpenShift Container Platform Master Hosts for more information.

2.3.5.2. Scaling the Cluster Monitoring Operator

OpenShift Container Platform exposes metrics that can be collected and stored in backends by the **cluster-monitoring-operator**. As an OpenShift Container Platform administrator, you can view system resources, containers, and component's metrics in one dashboard interface, Grafana.

In OpenShift Container Platform 3.11, the cluster monitoring operator installation is enabled by default as **`node-role.kubernetes.io/infra=true`** in your cluster. You can update this by setting **`openshift_cluster_monitoring_operator_node_selector`** in the inventory file of your customized node selector.Ensure there is an available node in your cluster to avoid unexpected failures.

See Scaling Cluster Monitoring Operator for capacity planning details.

2.3.6. Metrics and Logging

2.3.6.1. Prometheus Cluster Monitoring

Prometheus cluster monitoring is now fully supported in OpenShift Container Platform and deployed by default into an OpenShift Container Platform cluster.

- Query and plot cluster metrics collected by Prometheus.
- Receive notifications from pre-packaged alerts, enabling owners to take corrective actions and start troubleshooting problems.
- View pre-packaged Grafana dashboards for etcd, cluster state, and many other aspects of cluster health.

See Configuring Prometheus Cluster Monitoring for more information.

2.3.6.2. syslog Output Plug-in for fluentd (Technology Preview)

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

You can send system and container logs from OpenShift Container Platform nodes to external endpoints using the syslog protocol. The fluentd syslog output plug-in supports this.

IMPORTANT

Logs sent via syslog are not encrypted and, therefore, insecure.

See Sending Logs to an External Syslog Server for more information.

2.3.6.3. Elasticsearch 5 and Kibana 5

Elasticsearch 5 and Kibana 5 are now available. Kibana dashboards can be saved and shared between users. Elasticsearch 5 introduces better resource usage and performance and better resiliency.

Additionally, new numeric types, **`half_float`** and **`scaled_float`** are now added. There are now instant aggregations in Kibana 5, making it faster. There is also a new API that returns an explanation of why Elasticsearch shards are unassigned.

2.3.7. Developer Experience

2.3.7.1. CLI Plug-ins (Technology Preview)

This feature is 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.

2.3.7.2. Configure a Build Trigger Behavior without Triggering a Build Immediately

You can pause an image change trigger to allow multiple changes on the referenced image stream before a build is started. You can also set the **`paused`** attribute to **`true`** when initially adding an **`ImageChangeTrigger`** to a **`BuildConfig`** to prevent a build from being immediately triggered.

See Triggering Builds for more information.

2.3.7.3. More Flexibility in Providing Configuration Options to Builds Using ConfigMaps

In some scenarios, build operations require credentials or other configuration data to access dependent resources, but it is undesirable for that information to be placed in source control. You can define *input secrets* and *input ConfigMaps* for this purpose.

See Build Inputs for additional details.

2.3.7.4. kubectl

OpenShift Container Platform always shipped kubectl for Linux on the master's file system, but it is now available in the **`oc`** client downloads.

2.3.8. Registry

2.3.8.1. Accessing and Configuring the Red Hat Registry

All container images available through the Red Hat Container Catalog are hosted on an image registry, **`registry.access.redhat.com`**. The Red Hat Container Catalog is moving from **`registry.access.redhat.com`** to **`registry.redhat.io`**. The new registry, **`registry.redhat.io`**, requires authentication for access to images and hosted content on OpenShift Container Platform. Following the move to the new registry, the existing registry will be available for a period of time.

See Authentication Enabled Red Hat Registry for more information.

2.3.9. Quay

2.3.9.1. Red Hat Quay Registries

If you need an enterprise quality container image registry, Red Hat Quay is available both as a hosted service and as software you can install in your own data center or cloud environment. Advanced registry features in Red Hat Quay include geo-replication, image scanning, and the ability to roll back images. Visit the Quay.io site to set up your own hosted Quay registry account.

See Container Registry for more information.

2.3.10. Networking

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

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

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

2.3.10.2. Router (HAProxy) Enhancements

The OpenShift Container Platform router is the most common way to get traffic into the cluster. The table below lists the OpenShift Container Platform router (HAProxy) enhancements for 3.11.

Table 2.2. Router (HAProxy) enhancements

Feature	Feature enhancements	Command syntax
HTTP/2	Implements HAProxy router HTTP/2 support (terminating at the router).	**`$ oc set env dc/router ROUTER_ENABLE_HTTP2=true`**

Feature	Feature enhancements	Command syntax

Performance	Increases the number of threads that can be used by HAProxy to serve more routes.	1. Scale down the default router and create a new router using two threads:
		`$ oc scale dc/router --replicas=0`
		`$ oc adm router myrouter --threads=2 --images='openshift3/ose-haproxy-router:v3.x'`
		2. Set a new thread count (for, example **7**) for the HAProxy router:
		`$ oc set env dc/myrouter ROUTER_THREADS=7`

Dynamic changes	Implements changes to the HAProxy router without requiring a full router reload.	**`$ oc set env dc/router ROUTER_HAPROXY_CONFIG_MANAGER=true`**
Client SSL/TLS cert validation	Enables mTLS for route support of older clients/services that do not support SNI, but where certificate verification is a requirement.	**`$ oc adm router myrouter --mutual-tls-auth=optional --mutual-tls-auth-ca=/root/ca.pem --images="$image"`**

Feature	Feature enhancements	Command syntax

Logs captured by aggregated logging/EFK	Collects access logs so that Operators can see them.	1. Create a router with an rsyslog container:
		`$ oc adm router myrouter --extended-logging --images='xxxx'`
		2. Set the log level:
		`$ oc set env dc/myrouter ROUTER_LOG_LEVEL=debug`
		3. Check the access logs in the rsyslog container:
		`$ oc logs -f myrouter-x-xxxxx -c syslog`

2.3.10.3. HA Namespace-wide Egress IP

Adding basic active/backup HA for project/namespace egress IPs now allows a namespace to have multiple egress IPs hosted on different cluster nodes.

To add basic active/backup HA to an existing project/namepace:

1. Add two or more egress IPs to its **`netnamespace`**:

```
$ oc patch netnamespace myproject -p '{"egressIPs":
["10.0.0.1","10.0.0.2"]}'
```

2. Add the first egress IP to a node in the cluster:

```
# oc patch hostsubnet node1 -p '{"egressIPs":["10.0.0.1"]}'
```

3. Add the second egress IP to a different node in the cluster:

```
# oc patch hostsubnet node2 -p '{"egressIPs":["10.0.0.2"]}'
```

The project/namespace uses the first listed egress IP by default (if available) until that node stops responding, upon which other nodes switch to using the next listed egress IP, and so on. This solution requires greater than or equal to two IPs.

If the original IP eventually comes back, the nodes switch back to using the original egress IP.

See Enabling Static IPs for External Project Traffic for more information.

2.3.10.4. Fully-automatic Namespace-wide Egress IP

A fully-automatic HA option is now available. Projects/namespaces are automatically allocated a single egress IP on a node in the cluster, and that IP is automatically migrated from a failed node to a healthy node.

To enable the fully-automatic HA option:

1. Patch one of the cluster nodes with the **`egressCIDRs`**:

```
# oc patch hostsubnet node1 -p '{"egressCIDRs":["10.0.0.0/24"]}'
```

2. Create a project/namespace and add a single egress IP to its **`netnamespace`**:

```
# oc patch netnamespace myproject -p '{"egressIPs":["10.0.0.1"]}'
```

2.3.10.5. Configurable VXLAN Port

The OpenShift Container Platform SDN overlay VXLAN port is now configurable (default is **4789**). VMware modified the VXLAN port used in the VMware NSX SDN (≥v6.2.3) from **8472** to **4789** to adhere to RFC 7348.

When running the OpenShift Container Platform SDN overlay on top of VMware's NSX SDN underlay, there is a port conflict since both use the same VXLAN port (**4789**). With a configurable VXLAN port, users can choose the port configuration of the two products, used in combination, for their particular environment.

To configure the VXLAN port:

1. Modify the VXLAN port in ***master-config.yaml*** with the new port number (for example, **4889** instead of **4789**):

```
vxlanPort: 4889
```

2. Delete **`clusternetwork`** and restart the master API and controller:

```
$ oc delete clusternetwork default
$ master-restart api controller
```

3. Restart all SDN pods in the **`openshift-sdn`** project:

```
$ oc delete pod -n openshift-sdn -l app=sdn
```

4. Allow the new port on the firewall on all nodes:

```
# iptables -i OS_FIREWALL_ALLOW -p udp -m state --state NEW -m udp -
-dport 4889 -j ACCEPT
```

2.3.11. Master

2.3.11.1. Pod Priority and Preemption

You can enable pod priority and preemption in your cluster. Pod priority indicates the importance of a pod relative to other pods and queues the pods based on that priority. Pod preemption allows the cluster to evict, or preempt, lower-priority pods so that higher-priority pods can be scheduled if there is no available space on a suitable node. Pod priority also affects the scheduling order of pods and out-of-resource eviction ordering on the node.

See Pod Priority and Preemption for more information.

2.3.11.2. The Descheduler (Technology Preview)

This feature is currently in Technology Preview and 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.

2.3.11.3. Podman (Technology Preview)

This feature is currently in Technology Preview and 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.

2.3.11.4. Node Problem Detector (Technology Preview)

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

The Node Problem Detector monitors the health of your nodes by finding specific 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`.

The three problem daemons are:

- Kernel Monitor, which monitors the kernel log via journald and reports problems according to regex patterns.
- AbrtAdaptor, which monitors the node for kernel problems and application crashes from journald.
- CustomerPluginMonitor, which 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.

2.3.11.5. Cluster Autoscaling (AWS Only)

You can configure an auto-scaler on your OpenShift Container Platform cluster in Amazon Web Services (AWS) to provide elasticity for your application workload. The auto-scaler ensures that enough nodes are active to run your pods and that the number of active nodes is proportional to current demand.

See Configuring the cluster auto-scaler in AWS for more information.

2.3.12. Web Console

2.3.12.1. Cluster Administrator Console

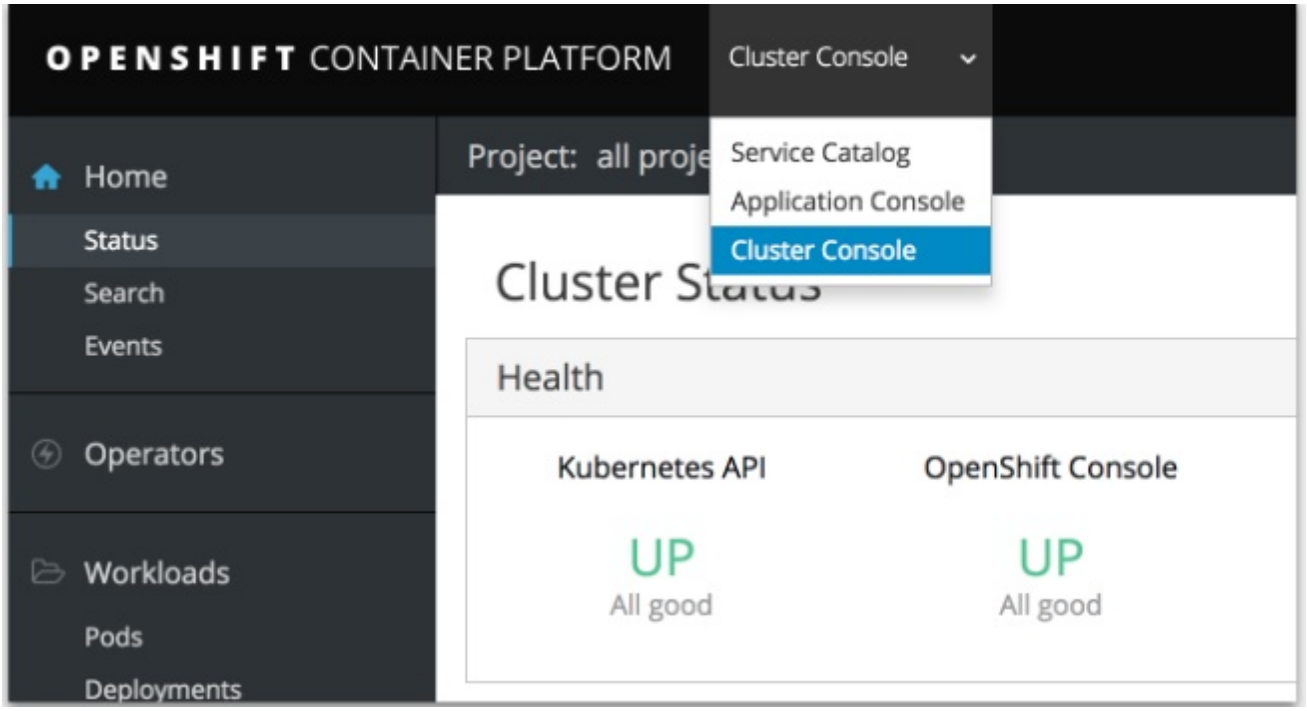
OpenShift Container Platform 3.11 introduces a cluster administrator console tailored toward application development and cluster administrator personas.

Users have a choice of experience based on their role or technical abilities, including:

- An administrator with Containers as a Service (CaaS) experience and with heavy exposure to Kubernetes.
- An application developer with Platform as a Service (PaaS) experience and standard OpenShift Container Platform UX.

Sessions are not shared across the consoles, but credentials are.

See Configuring Your Inventory File for details on configuring the cluster console.



2.3.12.2. Visibility into Nodes

OpenShift Container Platform now has an expanded ability to manage and troubleshoot cluster nodes, for example:

- Node status events are extremely helpful in diagnosing resource pressure and other failures.
- Runs **node-exporter** as a DaemonSet on all nodes, with a default set of scraped metrics from the **kube-state-metrics** project.
- Metrics are protected by RBAC.
- Those with **cluster-reader** access and above can view metrics.

2.3.12.3. Containers as a Service

You can view, edit, and delete the following Kubernetes objects:

- Networking
  - Routes and ingress
- Storage
  - PVs and PV claims
  - Storage classes
- Admin
  - Projects and namespaces
  - Nodes
  - Roles and RoleBindings

- CustomResourceDefinition (CRD)

2.3.12.4. Access Control Management

OpenShift Container Platform 3.11 includes visual management of the cluster's RBAC roles and RoleBindings, which allows you to:

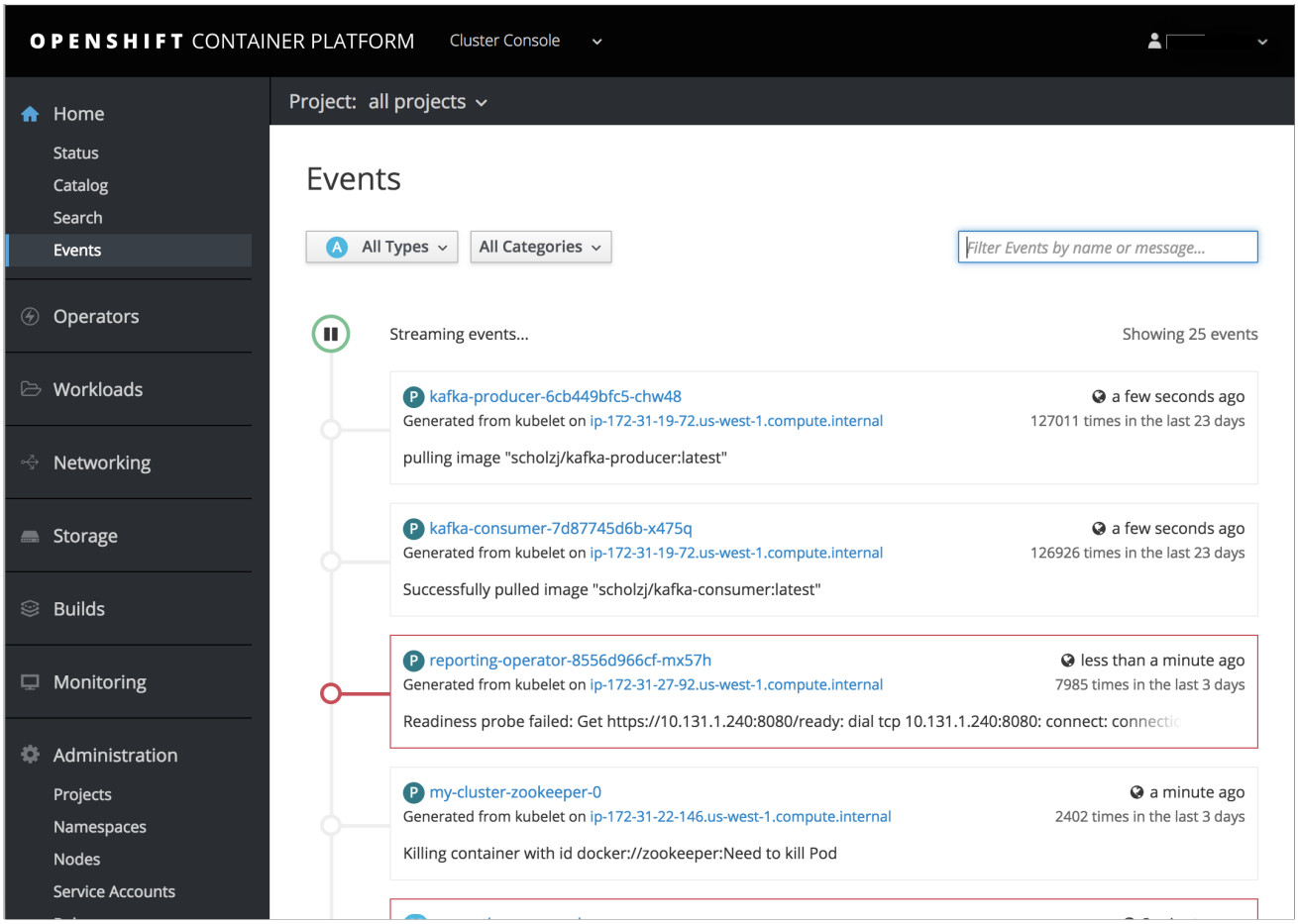
- Find users and service accounts with a specific role.
- View cluster-wide or namespaced bindings.
- Visually audit a role's verbs and objects.

Project administrators can self-manage roles and bindings scoped to their namespace.

2.3.12.5. Cluster-wide Event Stream

The cluster-wide event stream provides the following ways to help debug events:

- All namespaces are accessible by anyone who can list the namespaces and events.
- Per-namespace is accessible for all project viewers.
- There is an option to filter by category and object type.



2.3.13. Security

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

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

You can 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, for example:

- 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 visible to all other processes.
- Any **`kill all`** semantics used within the process are broken.
- Any **`exec`** processes from other containers show up.

See Expanding Persistent Volumes for more information.

2.3.13.2. GitHub Enterprise Added as Auth Provider

GitHub Enterprise is now an auth provider. OAuth facilitates a token exchange flow between OpenShift Container Platform and GitHub or GitHub Enterprise. You can use the GitHub integration to connect to either GitHub or GitHub Enterprise. For GitHub Enterprise integrations, you must provide the **`hostname`** of your instance and can optionally provide a **`ca`** certificate bundle to use in requests to the server.

See Configuring Authentication and User Agent for more information.

2.3.13.3. SSPI Connection Support on Microsoft Windows (Technology Preview)

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

oc now supports the Security Support Provider Interface (SSPI) to allow for single sign-on (SSO) flows on Windows. If you use the request header identity provider with a GSSAPI-enabled proxy to connect an Active Directory server to OpenShift Container Platform, users can automatically authenticate to OpenShift Container Platform using the **oc** command line interface from a domain-joined Windows computer.

See Configuring Authentication and User Agent for more information.

2.3.14. Microservices

2.3.14.1. Red Hat OpenShift Service Mesh (Technology Preview)

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

Red Hat OpenShift Service Mesh is a platform that provides behavioral insights and operational control over the service mesh, providing a uniform way to connect, secure, and monitor microservice applications.

The term service mesh is often used to describe the network of microservices that make up applications based on a distributed microservice architecture and the interactions between those microservices. As a service mesh grows in size and complexity, it can become harder to understand and manage.

Based on the open source Istio project, Red Hat OpenShift Service Mesh layers transparently onto existing distributed applications, without requiring any changes in the service code.

See Installing Red Hat OpenShift Service Mesh for more information.

2.4. NOTABLE TECHNICAL CHANGES

OpenShift Container Platform 3.11 introduces the following notable technical changes.

subjectaccessreviews.authorization.openshift.io and resourceaccessreviews.authorization.openshift.io Are Cluster-scoped Only
subjectaccessreviews.authorization.openshift.io and ***resourceaccessreviews.authorization.openshift.io*** are now cluster-scoped only. If you need namespace-scoped requests, use ***localsubjectaccessreviews.authorization.openshift.io*** and ***localresourceaccessreviews.authorization.openshift.io***.

New SCC options
No new privs flag

Security Context Constraints have two new options to manage use of the (Docker) **`no_new_privs`** flag to prevent containers from gaining new privileges:

- The **`AllowPrivilegeEscalation`** flag gates whether or not a user is allowed to set the security context of a container.
- The **`DefaultAllowPrivilegeEscalation`** flag sets the default for the **`allowPrivilegeEscalation`** option.

For backward compatibility, the **`AllowPrivilegeEscalation`** flag defaults to **`allowed`**. If that behavior is not desired, this field can be used to default to **`disallow`**, while still permitting pods to request **`allowPrivilegeEscalation`** explicitly.

Forbidden and unsafe sysctls options

Security Context Constraints have two new options to control which sysctl options can be defined in a pod spec:

- The **`forbiddenSysctls`** option excludes specific sysctls.
- The **`allowedUnsafeSysctls`** option controls specific needs such as high performance or real-time application tuning.

All safe sysctls are enabled by default; all unsafe sysctls are disabled by default and must be manually allowed by the cluster administrator.

Removed oc deploy Command

The **`oc deploy`** command is deprecated in OpenShift Container Platform 3.7. The **`oc rollout`** command replaces this command.

Removed oc env and oc volume Commands
The deprecated **`oc env`** and **`oc volume`** commands are now removed. Use **`oc set env`** and **`oc set volume`** instead.

Removed the oc ex config patch Command
The **`oc ex config patch`** command will be removed in a future release, as the **`oc patch`** command replaces it.

oc export Now Deprecated
The **`oc export`** command is deprecated in OpenShift Container Platform 3.10. This command will be removed in a future release, as the **`oc get --export`** command replaces it.

oc types Now Deprecated
In OpenShift Container Platform 3.11, **`oc types`** is now deprecated. This command will be removed in a future release. Use the official documentation instead.

Pipeline Plug-in Is Deprecated
The OpenShift Container Platform Pipeline Plug-in is deprecated but continues to work with OpenShift Container Platform versions up to version 3.11. For later versions of OpenShift Container Platform, either use the **`oc`** binary directly from your Jenkins Pipelines or use the OpenShift Container Platform Client Plug-in.

Logging: Elasticsearch 5
Curator now works with Elasticsearch 5.

See Aggregating Container Logs for additional information.

Hawkular Now Deprecated
Hawkular is now deprecated and will be removed in a future release.

New Registry Source for Red Hat images
Instead of **`registry.access.redhat.com`**, OpenShift Container Platform now uses **`registry.redhat.io`** as the source of images for version 3.11. For access, **`registry.redhat.io`** requires credentials. See Authentication Enabled Red Hat Registry for more information.

New Storage Driver Recommendation
Red Hat strongly recommends using the overlayFS storage driver instead of Device Mapper. For better performance, use overlayfs2 for Docker engine or overlayFS for CRI-O. Previously, we recommended using Device Mapper.

2.5. BUG FIXES

This release fixes bugs for the following components:

Builds

- ConfigMap Build Sources allows you to use ConfigMaps as a build source, which is transparent and easier to maintain than secrets. ConfigMaps can be injected into any OpenShift build. (**BZ#1540978**)

- Information about out of memory (OOM) killed build pods is propagated to a build object. This information simplifies debugging and helps you discover what went wrong if appropriate failure reasons are described to the user. A build controller populates the status reason and message

correctly when a build pod is OOM killed. (BZ#1596440)

- The logic for updating the build status waited to update the log snippet containing the tail of the build log only ran after the build status changed to the failed state. The build would first transition to a failed state, then get updated again with the log snippet. This means code watching for the build to enter a failed state would not see the log snippet value populated initially. The code is now changed to populate the log snippet field when the build transitions to failed status, so the build update will contain both the failed state and the log snippet. Code that watches the build for a transition to the failed state will see the log snippet as part of the update that transitioned the build to failed, instead of seeing a subsequent update later. (BZ#1596449)

- If a job used the `JenkinsPipelineStrategy` build strategy, the prune settings were ignored. As a result, setting `successfulBuildsHistoryLimit` and `failedBuildsHistoryLimit` did not correctly prune older jobs. The code has been changed to prune jobs properly. (BZ#1543916)

Cloud Compute

- You can now configure NetworkManager for **dns=none** during installation. This configuration is commonly used when deploying OpenShift Container Platform on Microsoft Azure, but can also be useful in other scenarios. To configure this, set `openshift_node_dnsmasq_disable_network_manager_dns=true`. (BZ#1535340)

Image

- Previously, because of improper handling of empty image stream updates, updates to an image stream that did not result in a change in tags resulted in a request to the image import API that included no content to be imported, which was invalid and lead to errors in the controller. Now, updates to the image stream that result in no new or updated tags that need to be imported will not result in an import API call. With this fix, invalid requests do not go to the import API, and no errors occur in the controller. (BZ#1613979)

- Image pruning stopped on encountering any unexpected error while deleting blobs. In the case of an image deletion error, image pruning failed to remove any image object from etcd. Images are now being pruned concurrently in separated jobs. As a result, image pruning does not stop on a single unexpected blob deletion failure. (BZ#1567657)

Installer

- When deploying to AWS, the `build_ami` play failed to clean */var/lib/cloud*. An unclean */var/lib/cloud* directory causes cloud-init to skip execution. Skipping execution causes a newly deployed node to fail to bootstrap and auto-register to OpenShift Container Platform. This bug fix cleans the */var/lib/cloud* directory during `seal_ami` play. (BZ#1599354)

- The installer now enables the router's extended route validation by default. This validation performs additional validation and sanitation of routes' TLS configuration and certificates. Extended route validation was added to the router in OpenShift Container Platform 3.3 and enhanced with certificate sanitation in OpenShift Container Platform 3.6. However, the installer did not previously enable extended route validation. There was initial concern that the validation might be too strict and reject valid routes and certificates, so it was disabled by default. But it has been determined to be safe to enable by default on new installs. As a result, extended route validation is enabled by default on new clusters. It can be disabled using by setting `openshift_hosted_router_extended_validation=False` in the Ansible inventory. Upgrading an existing cluster does **not** enable extended route validation. (BZ#1542711)

- Without the fully defined *azure.conf* file when a load balancer service was requested through

OpenShift Container Platform, the load balancer would never fully register and provide the external IP address. Now the ***azure.conf***, with all the required variables, allows the load balancer to be deployed and provides the external IP address. (**BZ#1613546**)

- To facilitate using CRI-O as the container-runtime for OpenShift Container Platform, update the ***node-config.yaml*** file with the correct endpoint settings. The **`openshift_node_groups`** defaults have been extended to include CRI-O variants for each of the existing default node groups. To use the CRI-O runtime for a group of compute nodes, use the following inventory variables:
  - **`openshift_use_crio=True`**
  - **`openshift_node_group_name="node-config-compute-crio"`**
    Additionally, to deploy the Docker garbage collector, **`docker gc`**, the following variable must be set to **`True`**. This bug fix changes the previous variable default value from **`True`** to **`False`**:
  - **`openshift_crio_enable_docker_gc=True`** (**BZ#1615884**)

- The ***ansible.cfg*** file distributed with **`openshift-ansible`** now sets a default log path of ***~/openshift-ansible.log***. This ensures that logs are written in a predictable location by default. To use the distributed ***ansible.cfg*** file, you must first change directories to ***/usr/share/ansible/openshift-ansible*** before running Ansible playbooks. This ***ansible.cfg*** file also sets other options meant to increase the performance and reliability of **`openshift-ansible`**. (**BZ#1458018**)

- Installing Prometheus in a multi-zone or region cluster using dynamic storage provisioning causes the Prometheus pod to become unschedulable in some cases. The Prometheus pod requires three physical volumes: one for the Prometheus server, one for the Alertmanager, and one for the alert-buffer. In a multi-zone cluster with dynamic storage, it is possible that one or more of these volumes becomes allocated in a different zone than the others. This causes the Prometheus pod to become unschedulable due to each node in the cluster only able to access physical volumes in its own zone. Therefore, no node can run the Prometheus pod and access all three physical volumes. The recommended solution is to create a storage class which restricts volumes to a single zone using the **`zone:`** parameter, and assigning this storage class to the Prometheus volumes using the Ansible installer inventory variable, **`openshift_prometheus_<COMPONENT>_storage_class=<zone_restricted_storage_class>`**. With this workaround, all three volumes get created in the same zone or region, and the Prometheus pod is automatically scheduled to a node in the same zone. (**BZ#1554921**)

Logging

- Previously, the **`openshift-ansible installer`** only supported **`shared_ops`** and **`unique`** as Kibana index methods. This bug fix allows users in a non-ops EFK cluster to share the default index in Kibana, to share queries, dashboards, and so on. (**BZ#1608984**)

- As part of installing the ES5 stack, users need to create a ***sysctl*** file for the nodes that ES runs on. This bug fix evaluates which nodes/Ansible hosts to run the tasks against. (**BZ#1609138**)

- Additional memory is required to support Prometheus metrics and retry queues to avoid periodic restarts from out-of-the-box memory. This bug fix increases out-of-the-box memory for Fluentd. As a result, Fluentd pods avoid out-of-the-box memory restarts. (**BZ#1590920**)

- Fluentd will now reconnect to Elasticsearch every 100 operations by default. If one Elasticsearch starts before the others in the cluster, the load balancer in the Elasticsearch service will connect

to that one and that one only, and so will all of the Fluentd connecting to Elasticsearch. With this enhancement, by having Fluentd reconnect periodically, the load balancer will be able to spread the load evenly among all of the Elasticsearch in the cluster. (**BZ#1489533**)

- The rubygem ffi 1.9.25 reverted a patch, which allowed it to work on systems with SELinux **`deny_execmem=1`**. This cases Fluentd to crash. This bug fix reverts the patch reversion and, as a result, Fluentd does not crash when using SELinux **`deny_execmem=1`**. (**BZ#1628407**)

Management Console

- The log viewer was not accounting for multi-line or partial line responses. If a response contained a multi-line message, it was appended and treated as a single line, causing the line numbers to be incorrect. Similarly, if a partial line were received, it would be treated as a full line, causing longer log lines sometimes to be split into multiple lines, again making the line count incorrect. This bug fix adds logic in the log viewer to account for multi-line and partial line responses. As a result, line numbers are now accurate. (**BZ#1607305**)

Monitoring

- The **`9100`** port was blocked on all nodes by default. Prometheus could not scrape the **`node_exporter`** service running on the other nodes, which listens on port **`9100`**. This bug fix modifies the firewall configuration to allow incoming TCP traffic for the **`9000`** - **`1000`** port range. As a result, Prometheus can now scrape the **`node_exporter`** services. (**BZ#1563888**)

- **`node_exporter`** starts with the **`wifi`** collector enabled by default. The **`wifi`** collector requires SELinux permissions that are not enabled, which causes AVC denials though it does not stop **`node_exporter`**. This bug fix ensures **`node_exporter`** starts with the **`wifi`** collector being explicitly disabled. As a result, SELinux no longer reports AVC denials. (**BZ#1593211**)

- Uninstalling Prometheus currently deletes the entire **`openshift-metrics`** namespace. This has the potential to delete objects which have been created in the same namespace but are not part of the Prometheus installation. This bug fix changes the uninstall process to delete only the specific objects which were created by the Prometheus install and delete the namespace if there are no remaining objects, which allows Prometheus to be installed and uninstalled while sharing a namespace with other objects. (**BZ#1569400**)

Pod

- Previously, a Kubernetes bug caused **`kubectl drain`** to stop when pods returned an error. With the Kubernetes fix, the command no longer hangs if pods return an error. (**BZ#1586120**)

Routing

- Because dnsmasq was exhausting the available file descriptors after the OpenShift Extended Comformance Tests and the Node Vertical Test, dnsmasq was hanging and new pods were not being created. A change to the code increases the maximum number of open file descriptors so the node can pass the tests. (**BZ#1608571**)

- If 62 or more IP addresses are specified using an **`haproxy.router.openshift.io/ip_whitelist`** annotation on a route, the router will error due to exceeding the maximum parameters on the command (63). The router will not reload. The code was changed to use an overflow map if the there are too many IPs in the whitelist annotation and pass the map to the HA-proxy ACL. (**BZ#1598738**)

- By design, using a route with several services, when configuring a service with **`set route-backend`** set to **`0`**, the weight would drop all existing connections and associated end user

connections. With this bug fix, a value of `0` means the server will not participate in load-balancing but will still accept persistent connections. (BZ#1584701)

- Because the liveness and readiness probe could not differentiate between a pod that was alive and one that was ready, a router with `ROUTER_BIND_PORTS_AFTER_SYNC=true` was reported as failed. This bug fix splits the liveness and readiness probe into separate probes, one for readiness and one for liveness. As a result, a router pod can be alive but not yet ready. (BZ#1550007)

- When the HAproxy router contains a large number of routes (10,000 or more), the router will not pass the liveness and Readiness due to low performance, which kills the router repeatedly. The root cause of this issue is likely that a health check cannot be completed within the default readiness and liveness detection cycle. To prevent this problem, increase the interval of the probes. (BZ#1595513)

Service Broker

- The deprovision process for Ansible Service Broker was not deleting secrets from the **openshift-ansible-service-broker** project. With this bug fix, the code was changed to delete all associated secrets upon Ansible Service Broker deprovisioning. (BZ#1585951)

- Previously, the broker's reconciliation feature would delete its image references before getting the updated information from the registry, and there would be a period before the records appeared in the broker's data store while other jobs were still running. The reconciliation feature was redesigned to do an in-place update for items that have changed. For items removed from the registry, the broker deletes only those not already provisioned. It will also mark those items for deletion, which filters them out of the UI, preventing future provisions of those items. As a result, the broker's reconciliation feature makes provisioning and deprovisioning more resilient to registry changes. (BZ#1577810)

- Previously, users would see an error message when an item was not found, even if it is normal not to be found. As a result, successful jobs might have an error message logged, causing the user concern that there might be a problem when there was none. The logging level of the message has now been changed from `error` to `debug`, because the message is still useful for debugging purposes, but not useful for a production installation, which usually has the level set to `info` or higher. As a result, users will not see an error message when the instance is not found unless there was an actual problem. (BZ#1583587)

- If the cluster is not running or is not reachable, the `svcat version` command resulted in an error. The code has been changed to always report the client version, and if the server is reachable, it then reports the server version. (BZ#1585127)

- In some scenarios, using the `svcat deprovision <service-instance-name> --wait` command sometimes resulted in the `svcat` command terminating with a panic error. When this happened, the `deprovision` command got executed, and the program then encountered a code bug when attempting to wait for the instance to be fully deprovisioned. This issue is now resolved. (BZ#1595065)

Storage

- Previously, because the kubelet system containers could not write to the */var/lib/iscsi* directory, iSCSI volumes could not be attached. Now, you can mount the host */var/lib/iscsi* into the kubelet system container so that iSCSI volumes can be attached. (BZ#1598271)

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 2.3. Technology Preview Tracker

Feature	OCP 3.9	OCP 3.10	OCP 3.11
Prometheus Cluster Monitoring	TP	TP	GA
Local Storage Persistent Volumes	TP	TP	TP
CRI-O for runtime pods	GA	GA* [a]	GA
Tenant Driven Snapshotting	TP	TP	TP
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	GA	GA	GA
Kubernetes Deployments	GA	GA	GA
StatefulSets	GA	GA	GA

Feature	OCP 3.9	OCP 3.10	OCP 3.11
Explicit Quota	GA	GA	GA
Mount Options		GA	GA
System Containers for Docker, CRI-O	Dropped	-	-
Installing from a System Container	GA	GA	GA
Hawkular Agent	-	-	-
Pod PreSets	-	-	-
experimental-qos-reserved	TP	TP	TP
Pod sysctls	TP	TP	TP
Central Audit	GA	GA	GA
Static IPs for External Project Traffic	GA	GA	GA
Template Completion Detection	GA	GA	GA
`replicaSet`	GA	GA	GA
Mux	TP	TP	TP
Clustered MongoDB Template	-	-	-
Clustered MySQL Template	-	-	-
Image Streams with Kubernetes Resources	GA	GA	GA
Device Manager	TP	GA	GA
Persistent Volume Resize	TP	TP	TP
Huge Pages	TP	GA	GA

Feature	OCP 3.9	OCP 3.10	OCP 3.11
CPU Manager	TP	GA	GA
Device Plug-ins	TP	GA	GA
syslog Output Plug-in for fluentd	TP	TP	TP
Container Storage Interface (CSI)	-	TP	TP
Persistent Volume (PV) Provisioning Using OpenStack Manila	-	TP	TP
Node Problem Detector	-	TP	TP
Protection of Local Ephemeral Storage	-	TP	TP
Descheduler	-	TP	TP
Podman	-	TP	TP
Kuryr CNI Plug-in	-	TP	TP
Sharing Control of the PID Namespace	-	TP	TP
Cluster Administrator console	-	-	GA
Cluster Autoscaling (AWS Only)	-	-	GA
Operator Lifecycle Manager	-	-	TP
Red Hat OpenShift Service Mesh	-	-	TP

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

2.7. KNOWN ISSUES

- Due to a change in the authentication for the Kibana web console, you must log back into the console after an upgrade and every 168 hours after initial login. The Kibana console has migrated to **oauth-proxy**. (**BZ#1614255**)

2.8. ASYNCHRONOUS ERRATA UPDATES

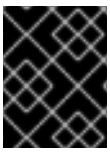
Security, bug fix, and enhancement updates for OpenShift Container Platform 3.11 are released as asynchronous errata through the Red Hat Network. All OpenShift Container Platform 3.11 errata is available on the Red Hat Customer Portal. See the OpenShift Container Platform Life Cycle for more information 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 3.11. Versioned asynchronous releases, for example with the form OpenShift Container Platform 3.11.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.

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.