OpenShift Enterprise 3.0 What’s New?

What's New in This Version of OpenShift Enterprise 3.0

Red Hat OpenShift Documentation Team

What's New in This Version of OpenShift Enterprise 3.0

Legal Notice

Abstract

OpenShift Enterprise 3.0 brings many architectural changes and introduces various new components. It is built around the central idea of applications running in Docker containers, scheduling and management support provided by the Kubernetes project, and augmented deployment, orchestration, and routing functionality built on top. These topics provide information on the new architecture and components to help you understand what is new in OpenShift Enterprise 3.0. In the following sections, "OpenShift v2" and "OpenShift v3" refer to the second and third architectural revisions of OpenShift Enterprise, respectively.

Table of Contents

CHAPTER 1. OVERVIEW

OpenShift v3 brings many architectural changes and introduces new concepts and components. It is built around the applications running in Docker containers, scheduling and management support provided by the Kubernetes project, and augmented deployment, orchestration, and routing functionality on top.

The most significant changes surround the container model and how they are monitored and interconnected. Kubernetes pods are a group of containers that act like a single VM: they have a single IP address, they can share a file system, and they typically have similar security settings. Grouping containers together vastly increases the number of applications that can be brought to OpenShift. Rather than focusing on a microservice model to the exclusion of all other patterns, pods enable developers to port existing applications that need to share local resources while still benefiting from a containerized model.

Second, OpenShift containers are expected to be **immutable**: the image contains a particular snapshot of application code and its dependent libraries, while any configuration, secrets, or persistent data is attached to the container at runtime. This allows administrators and integrators to separate code and patches from configuration and data. While it is still possible to mutate your containers, the higher level concepts of build and deployments leverage immutable containers to provide higher level guarantees about what code is running where.

The third important change is in the core system design: OpenShift and Kubernetes are built as sets of microservices working together through common REST APIs to change the system. Those same APIs are available to system integrators, and those same core components can be disabled to allow alternative implementations. A great example of this are the **watch** APIs: clients can connect and receive a stream of changes to pods (or other objects) and take action on them as the pods become available (to signal errors or log changes to the system). OpenShift exposes fine-grained access control on the REST APIs to enable this integration, which means there are no actions in the system that cannot also be done by an integrator.

The following topics provide more information on the new architecture and concepts to help you understand what is new in OpenShift v3.

Note

In the following sections, "OpenShift v2" and "OpenShift v3" refer to the second and third architectural revisions of OpenShift, respectively.

CHAPTER 2. OPENSHIFT ENTERPRISE 3.0 RELEASE NOTES

2.1. OVERVIEW

OpenShift Enterprise by Red Hat is a Platform as a Service (PaaS) that provides developers and IT organizations with a cloud application platform for deploying new applications on secure, scalable resources with minimal configuration and management overhead. OpenShift Enterprise supports a wide selection of programming languages and frameworks, such as Java, Ruby, and PHP.

Built on Red Hat Enterprise Linux and Google Kubernetes, OpenShift Enterprise provides a secure and scalable multi-tenant operating system for today's enterprise-class applications while providing integrated application runtimes and libraries. OpenShift Enterprise brings the OpenShift PaaS platform to customer data centers, enabling organizations to implement a private PaaS that meets security, privacy, compliance, and governance requirements.

2.2. NEW FEATURES

Standard Containers API

Built on a core of standard and portable Linux containers in the Docker format.

Web-scale Container Orchestration and Management

Leverages powerful, web-scale container orchestration and management with Google Kubernetes.

Container-optimized Operating System

Integrates Red Hat Enterprise Linux 7, optimized for running containers at scale.

Runtimes and Services

Provides the largest collection of supported programming languages, frameworks, and services.

Robust Tools and User Experience for Developers and Operations

Includes a rich web console, CLI interfaces, multi-user collaboration features, build automation and Source-to-Image, integration with CI, and deployment automation.

Industry Standard, Web-scale Distributed Application Platform

Includes a new model for container networking, support for remote storage volumes, and simplified installation and administration.

2.3. INSTALLING OPENSHIFT ENTERPRISE 3.0

See the Administrator Guide for available installation methods.

2.4. MIGRATING APPLICATIONS TO OPENSHIFT ENTERPRISE 3.0

ENTERPRISE 3.0

Instructions for migrating applications to OpenShift Enterprise 3.0 from previous versions will be provided in future releases.

2.5. 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

The following features are in Technology Preview:

- Enabling clustering for database images.
- Using the JVM Console.
- Using persistent volume plug-ins other than the supported NFS plug-in, such as AWS Elastic Block Stores (EBS), GCE Persistent Disks, GlusterFS, iSCSI, and RADOS (Ceph).

2.6. KNOWN ISSUES

- BZ#1233540: Persistent Volume does not get recycled when the persistentVolumeReclaimPolicy is "Recycle"
  - Fixed in OpenShift Enterprise 3.0.0.1

2.7. XPAAS IMAGES

These are the release notes for xPaaS images on OpenShift Enterprise 3.0. For the release notes for OpenShift Enterprise 3.0, please refer here.

xPaaS images for OpenShift Enterprise 3.0 are provided for Red Hat JBoss Enterprise Application Platform, Red Hat JBoss Web Server and Red Hat JBoss A-MQ.

2.7.1. xPaaS Image for Red Hat JBoss EAP

Red Hat JBoss EAP is available as a containerized xPaaS image that is designed for use with OpenShift Enterprise 3.0.

However, there are significant differences in supported configurations and functionality in the JBoss EAP xPaaS image compared to the regular release of JBoss EAP. To learn about these differences, please read this documentation. Documentation for other JBoss EAP functionality not specific to the JBoss EAP xPaaS image can be found in the JBoss EAP documentation on the Red Hat Customer Portal.

xPaaS Image for Red Hat JBoss EAP 6.4.x

These are the release notes for the **`jboss-eap-6/eap64-openshift`** image.

Image ID	**c12d6b01a2fbd3036df445aa03ae6e9210801bd3245daf4e4fc23af08eb20c21**
Tags	**1.1-2**, **1.1**, **latest**
Build date	2015-09-18
Release date	2015-11-06
OpenShift Enterprise version	**3.0.2**+

Bug fixes

- **mvn clean** is no longer run as part of S2I build process (and so it won't fail)
- S2I builds force use of IPv4 by default

Enhancements

- **DNS_PING** protocol used for clustering is now replaced with **KUBE_PING**

> **Warning**
>
> For clustering to work, you must add the **view** role to the service account.

Example Policy commands

Using the default service account in the myproject namespace:

```
oc policy add-role-to-user view system:serviceaccount:myproject:default
-n myproject
```

Using the eap-service-account in the myproject namespace:

```
oc policy add-role-to-user view system:serviceaccount:myproject:eap-
service-account -n myproject
```

- Can now be used as a standalone builder image, e.g. oc new-app image~source
- JBoss EAP was updated to **6.4.3.GA**
- Image version information is now printed on boot

- **JAVA_OPTS_APPEND** environment variable is supported and appended to the **JAVA_OPTS** variable upon container start-up
- Clean shutdown is now possible by sending the **TERM** signal
- S2I scripts have moved to **/usr/local/s2i** from **/usr/local/sti**

Source to Image (S2I) Enhancements

For full details on using these features, See the S2I section of the image documentation.

- A Maven mirror/repository manager can be configured via environment variables.
- A maven HTTP mirror can be configured, including optional authentication and host exclusions

2.7.2. xPaaS Image for Red Hat JWS

With this release, the Apache Tomcat 7 and Apache Tomcat 8 components of Red Hat JBoss Web Server 3 are available as containerized xPaaS images that are designed for use with OpenShift Enterprise 3.0.

However, there are significant differences in the functionality between the JBoss Web Server xPaaS images and the regular release of JBoss Web Server. To learn about these differences, please read this documentation. Documentation for other JBoss Web Server functionality not specific to the JBoss Web Server xPaaS images can be found in the JBoss Web Server documentation on the Red Hat Customer Portal.

2.7.3. xPaaS Image for Red Hat JBoss A-MQ

Red Hat JBoss A-MQ is available as a containerized xPaaS image that is designed for use with OpenShift Enterprise 3.0. It allows developers to quickly deploy an A-MQ message broker in a hybrid cloud environment.

However, there are significant differences in supported configurations and functionality in the JBoss A-MQ image compared to the regular release of JBoss A-MQ. Documentation for other JBoss A-MQ functionality not specific to the JBoss A-MQ xPaaS image can be found in the JBoss A-MQ documentation on the Red Hat Customer Portal.

xPaaS Image for Red Hat JBoss A-MQ 6.2

Image ID	**1ac9fd48694766bf7f17a7f71f29033012769971bd0e95158d9ad8f1501f8d25**
Tags	**1.1-2**, **1.1**, **latest**
Build date	2015-09-18

Release date	2015-11-06
OpenShift Enterprise version	**3.0.2**+

Enhancements

- Image version information is now printed on boot
- Added support for using Kubernetes REST API to resolve service endpoints for mesh

> **Warning**
>
> For clustering to work, you must add the **view** role to the service account.

Example: Policy commands

Using the default service account in the myproject namespace:

```
oc policy add-role-to-user view system:serviceaccount:myproject:default
-n myproject
```

Using the amq-service-account in the myproject namespace:

```
oc policy add-role-to-user view system:serviceaccount:myproject:amq-
service-account -n myproject
```

- S2I scripts have moved to **/usr/local/s2i** from **/usr/local/sti**

2.8. KNOWN ISSUES FOR XPAAS IMAGES

These are the current known issues along with any known workarounds.

JWS

- https://issues.jboss.org/browse/CLOUD-57: Tomcat's access log valve logs to file in container instead of stdout

  Due to this issue, the logging data is not available for the central logging facility. To work around this issue, use the **oc exec** command to get the contents of the log file.

- https://issues.jboss.org/browse/CLOUD-153: **mvn clean** in JWS STI can fail

  It is not possible to clean up after a build in JWS STI because the Maven command **mvn clean** fails. This command fails due to Maven not being able to build the object model during startup.

To work around this issue, add Red Hat and JBoss repositories into the ***pom.xml*** file of the application if the application uses dependencies from there.

- https://issues.jboss.org/browse/CLOUD-156: Datasource realm configuration is incorrect for JWS

  It is not possible to do correct JNDI lookup for datasources in the current JWS image if an invalid combination of datasource and realm properties is defined. If a datasource is configured in the ***context.xml*** file and a realm in the***server.xml*** file, then the ***server.xml*** file's **`localDataSource`** property should be set to**true**.

EAP

- https://issues.jboss.org/browse/CLOUD-61: JPA application fails to start when the database is not available

  JPA applications fail to deploy in the EAP OpenShift Enterprise 3.0 image if an underlying database instance that the EAP instance relies on is not available at the start of the deployment. The EAP application tries to contact the database for initialization, but because it is not available, the server starts but the application fails to deploy.

  There are no known workarounds available at this stage for this issue.

- https://issues.jboss.org/browse/CLOUD-158: Continuous HornetQ errors after scale down "Failed to create netty connection"

  In the EAP image, an application not using messaging complains about messaging errors related to HornetQ when being scaled.

  Since there are no configuration options to disable messaging to work around this issue, simply include the ***standalone-openshift.xml*** file within the source of the image and remove or alter the following lines related to messaging:

  ```
  Line 18:

  <!-- ##MESSAGING_EXTENSION## -->

  Line 318:

  <!-- ##MESSAGING_SUBSYSTEM## -->
  ```

- https://issues.jboss.org/browse/CLOUD-161: EAP pod serving requests before it joins cluster, some sessions reset after failure

  In a distributed web application deployed on an EAP image, a new container starts serving requests before it joins the cluster.

  There are no known workarounds available at this stage for this issue.

EAP and JWS

- https://issues.jboss.org/browse/CLOUD-159: Database pool configurations should contain validation SQL setting

  In both the EAP and JWS images, when restarting a crashed database instance, the connection pools contain stale connections.

  To work around this issue, restart all instances in case of a database failure.

A-MQ

There are no known issues in the A-MQ image.

2.9. ASYNCHRONOUS ERRATA UPDATES

Security, bug fix, and enhancement updates for OpenShift Enterprise 3.0 are released as asynchronous errata through the Red Hat Network. All OpenShift Enterprise 3.0 errata is available on the Red Hat Customer Portal. See the OpenShift Enterprise 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 Enterprise entitlements for OpenShift Enterprise errata notification emails to generate.

The following sections provide notes on enhancements and bug fixes for each release.

Important

For any release, always review the Administrator Guide for instructions on upgrading your OpenShift cluster properly, including any additional steps that may be required for a specific release.

2.9.1. OpenShift Enterprise 3.0.1.0

OpenShift Enterprise release 3.0.1.0 (RHBA-2015:1540) is now available. Ensure that you follow the instructions on upgrading your OpenShift cluster properly, including steps specific to this release.

This release includes the following enhancements and bug fixes:

Backwards Compatibility

API

- The **pods/exec** endpoint is being moved to **POST** instead of **GET**. For backwards compatibility, **GET** continues to be supported. Clients will try to use **POST**, and if that fails, will try to use **GET**. If you have an existing deployment, the default policy will need to be updated prior to 1.1.0. See Issue #3717 for more information.
- The **hostDir** volume type has been renamed **hostPath** in all **Pod** and **PodTemplate** objects.
- **Pod**: The **serviceAccount** field changed to **serviceAccountName**. OpenShift will continue to accept and output both fields; **serviceAccountName** takes precedence.

- **Pod**: The **host** field changed to **nodeName**. OpenShift will continue to accept and output both fields; **nodeName** takes precedence.
- **Service**: The **portalIP** field changed to **clusterIP**. OpenShift will continue to accept and output both fields; **clusterIP** takes precedence.
- **Service**: The protocol for a port under a **Service**, **Endpoint**, or **Container** must be uppercased: **TCP** instead of **tcp**, and **UDP** instead of **udp**. OpenShift will continue to accept all case variations.

Stored Objects

- Build pods previously inherited the labels of the build. This resulted in pods from builds being accidentally being included in deployments that had similar labels. It was never intended that build pods should share labels with existing components, so this behavior has been removed. Queries that attempt to retrieve build pods by label will no longer work.

Enhancements

For Administrators

- Kubernetes was updated to v1.0.0.
- To make it easier to upgrade your cluster, the **oadm reconcile-cluster-roles** command has been added to update your cluster roles to match the internal default. Use this command to verify the cluster infrastructure users have the appropriate permissions.
- A new LDAP authentication identity provider has been added, allowing administrators to configure OpenShift to verify passwords and users against an LDAP server directly.
- The master's CA certificate can be made available as a secret inside pods, making it easier to manage secure TLS inside the cluster. To enable this in an existing configuration, set the **masterCA** field in the master configuration file.
- The current version of the master is now shown on startup, and startup logging has been cleaned up.
- The ability to use host ports and the **hostNetwork** option is now properly secured by security context constraints (SCCs), and only restricted or higher users can use them.
- The **RunAsNonRoot** option for pod SCCs has been added. It is now possible to restrict users to running pods that are non-root (i.e., pods that have an explicit **USER** numeric value set in their Docker image or have specified the user ID on their pod SCC).

For Developers

- Output for **oc status** has been improved to make it easier to see the types of objects being presented.
- You can now search for images, templates, and other inputs for the **oc new-app** command using the **--search** and **--list** flags. For example, to find all of the images or templates that include PHP:

```
$ oc new-app --search php
```

- The **`oc new-app`** command now always add an **`app=<name>`** label on the created objects when you do not specify labels with **`--labels`**. The name is inferred from **`--name`** or the name of the first component passed to the command. For example:

```
$ oc new-app php
```

  adds a label **`app=php`** to all of objects it creates. You can then easily delete all of those components using:

```
$ oc delete all -l app=php
```

- The **`oc rsh <pod>`** command has been added, which is a shortcut for:

```
$ oc exec -itp POD -- bash
```

  The new command makes it easier to get a remote shell into your pods.

- Rolling updates can now be done by percentage: you can specify the percentage of pods to update by a negative or positive amount that adjusts the amount of replicas in chunks. If negative, old deployments are scaled down first. If positive, extra pods are created first. The rolling update works to keep the desired amount of pods running (100% of the old deployment size when a positive percentage or 100%-**`UpdatePercent`** when negative) as it goes.

Bug Fixes

For Cluster Administrators

- The **`openshift start --print-ip`** command was added, which reports the IP that the master will use if no **`--master`** address is provided, then exits.
- OpenShift performance when idling has been improved by removing an inefficient timer loop.
- The router and internal registry now default to using the **`RollingUpdate`** strategy deployment. Red Hat recommends updating any existing router or registry installations if you plan on scaling them up to multiple pods.
- The **`oadm policy who-can`** command now shows additional information.
- Master startup no longer has a chance to generate certificates with duplicate serial numbers, which previously rendered them unusable.

For Developers

- For the **`oc new-app`** command:
  - Scala Git repositories are now detected.
  - A bug was fixed where explicit tags were being set on new image streams, which confused builds.
  - Ports are now exposed that were defined in the source ***Dockerfile*** when creating an application from a Git repository.
  - The **FROM** instruction in a ***Dockerfile*** can now point to an image stream or invalid image.

  - For any image that has volumes, **emptyDir** volumes are now created and the user is informed.
  - All ports defined on the image can now be exposed on the generated service.
  - The **--name** argument now also changes the name of the image stream.
  - Labels passed with **--labels** are now properly set onto the pod template and selector for the deployment.
- The **oc status** command now shows standalone replication controllers and a number of other warnings about issues.
- The timeout for log sessions and the **oc exec** and **oc portforward** commands has been increased from 5 minutes to 1 hour.
- Cleanup and improvements were made to the **Browse** pages in the web console, including a better layout at smaller resolutions.
- OpenShift now avoids writing excessive log errors on initial deployments when the image is not yet available.
- Quay.io registries are now supported by using cookies when importing images.
- Docker images of the form **<registry>/<name>** are now properly handled by the **oc new-app** command and the image import functionality.
- Secret volumes are now unique for push and pull secrets during builds.
- The **oc secret** commands now provide better usage errors.
- Builds are now filtered by completion time in the **Overview** page of the web console.
- A race condition was fixed when service accounts with ***.dockercfg*** files (for pull secrets) were deleted.
- When generating and adding secrets to a service account, the **oc secrets add** command now allows the user to specify which type of secret is being added: **mount** or **pull**.
- The custom builder build type now allows image output to be disabled instead of requiring it on input.
- WebSocket errors in the web console are now handled more effectively.
- The **http_proxy** and **HTTP_PROXY** environment variables can now be passed to builds.
- Routes now default to using the route name when creating a virtual host, not the service name.
- The **oc expose** command no longer defaults to creating routes, except when a service is exposed.
- More detail is now shown on the image streams page in the web console.
- Source code revision information is now shown in the **oc describe build** output.
- TLS termination output is now shown in **oc describe route** output.

- Image importing now works with registries that do not implement the whole Docker Registry API (e.g., Pulp read-only registries).
- Deployment configurations now trigger deployment when the **metadata** field of the pod template is changed, not just when the **spec** is changed.
- The project request template now allows Kubernetes objects as well as OpenShift objects.
- The **oc volume** command can now change the volume type when the**--mount-path** is unambiguous.
- Builds now properly cancel when the user requests them, rather than running to completion.
- The **oc export** command no longer fails when exporting image streams that do not have tags under their **spec**.
- Attempting to use the default PostgreSQL database service templates after using a default MySQL template failed with errors reporting "mysql" already exists. This was due to an incorrect value in the PostgreSQL templates, which has now been fixed. (BZ#1245559)
- Previously when creating from templates in the web console, the creation would fail if the template contained certain API object types, including persistent volume claims, secrets, and service accounts. This was due to the web console missing these types from its API type map. The type map has now been updated to include these missing types, and the web console also now gracefully handles unrecognized object types, reporting a relevant error message. (BZ#1244254)
- The web console previously produced errors when users attempted to create from templates that had a Custom build strategy. The errors obscured the template parameters from being shown or managed. The web console has now been updated to properly handle Custom and Docker build strategies in templates. As a result, the errors no longer occur, and template parameters can be viewed and managed. (BZ#1242312)
- If the **nfs-utils** package was not installed on a node host, when a user tried to add an NFS volume to an application, the mount operation would fail for the pod. This bug fix adds the **nfs-utils** package as a dependency for the**openshift-node** package so it is installed on all node hosts by default. (BZ#1238565)
- Previously, pruning images associated with an image stream that had been removed would fail. This bug fix updates layer pruning to always delete blobs from the registry, even if the image stream(s) that referenced the layer no longer exists. In the event that there are no longer any image streams referencing the layer, the blob can still be deleted, but not the registry image repository layer link files. (BZ#1237271)
- The **Documentation** link in the header of the web console previously linked to the latest OpenShift Origin documentation. It has been updated to now point to the OpenShift Enterprise documentation for the current version. (BZ#1233772)
- The **Create** page in the web console has been updated to make it more obvious that there are two options rather than one. The headings have been modified to "Create Using Your Code" and "Create Using a Template", and a separator has been added between the two options. (BZ#1233488)

- Previously using the CLI, labels could be set to empty values, and setting labels to invalid values produced an unfriendly error. This bug fix updates the CLI to no longer allow setting labels to empty values, and setting labels to invalid values produces a better error message. (BZ#1230581)

2.9.2. OpenShift Enterprise 3.0.2.0

OpenShift Enterprise release 3.0.2.0 (RHBA-2015:1835) is now available. Ensure that you follow the instructions on upgrading your OpenShift cluster properly, including steps specific to this release.

This release includes the following enhancements and bug fixes:

Backwards Compatibility

API

- If a deployment configuration is created without specifying the **`triggers`** field, the deployment now defaults to having a configuration change trigger.
- The new **`subjects`** field (a list of object references) is now available when creating role bindings. You can pass object references to **`User`**, **`SystemUser`**, **`Group`**, **`SystemGroup`**, or **`ServiceAccount`** when defining the binding. Passing a reference to a service account resolves the correct name, making it easier to grant access to service accounts in the current namespace. If users or groups are also specified, they take priority over values set in **`subjects`**.
- Template parameters now support **`displayName`**, which is an optional field to use from user interfaces when your template is shown.
- Secrets can now be added to custom builds and mounted at user-specified locations.
- Validation of the **`container.ports.name`** field was changed:
  - If specified, the value must be no more than 15 characters, have at least one letter [a-z] and contain only [a-z0-9-]. Hyphens cannot be leading or trailing characters, or adjacent to each other.
  - Resources generated by the web console, command line tooling, or templates were all updated to conform to the updated syntax.
  - Existing data may require manual modification to either remove the **`container.ports.name`** or update its value to conform to the new validation rules. Failure to do so may result in an inability to create pod resources that specify the invalid syntax.
  - A symptom of an invalid configuration would be the production of a large number of **Event** resources whose **Event.Reason** is **failedCreate** and whose **Event.Message** includes the string **must be an IANA_SVC_NAME**. Operators must edit the **Event.InvolvedObject** to address the invalid configuration by doing an **`oc edit`** command.
- Pending removal:
  - Support for **`v1beta3`** from the API and from client commands will be removed in OpenShift Enterprise 3.1.
  - Builds marked only with the **build** label will no longer be considered part of their parent build configuration in OpenShift Enterprise 3.1. You can see a list of

affected builds by running:

```
$ oc get builds --all-namespaces
```

Then look for builds that only have the **build** label and not **openshift.io/build**. See Issue #3502 for more information.

- The **spec.rollingParams.updatePercent** field on deployment configurations will no longer be recognized in OpenShift Enterprise 3.1. Use **maxUnavailable** and **maxSurge** instead.

Enhancements

Security

- Secrets were previously limited to only being available in pods when the service account referenced them. To make it easier to use secrets in templates, this is now disabled by default. Cluster administrators can override this by setting the **serviceAccountConfig.limitSecretReferences** variable to **true** in the master configuration file to force this for the whole platform. Project administrators can also set the **"kubernetes.io/enforce-mountable-secrets"** annotation to **"true"** on a particular service account to require that check.

Platform

- Groups of users are now supported. Cluster administrators can use the **oadm groups** command to manage them.
- Service accounts are now more easily bound to roles through the new **subjects** field, as described in **Backwards Compatibility** above.

Web Console

- You can now deploy, rollback, retry, and cancel deployments from the web console.
- You can now cancel running builds from the web console.
- Improvements have been made to layout and readability at mobile resolutions.
- You can now customize the web console and login page.

Networking/Routing

- The **--host-network** flag has been added to the **openshift router** command to allow the router to run with the container network stack when set to **false**.
- The default host name for a route has been changed to the form **<route-name>-<namespace>.<suffix>**. This allows TLS wildcards on **<suffix>** to work properly.
- A new **F5 BIG-IP®** router plug-in has been added, allowing F5 routers to be dynamically configured.
- The router can now be configured to serve a subset of the routes in your deployment:
  - Pass **--namespace** to the router command to select routes in a single namespace.
  - Pass **--labels=<selector>** or **--fields=<selector>** to select only routes with the provided labels or fields.

- Pass **--project-labels=*** to show routes in all labels that the router's service account is granted access to, **--project-labels=<selector>** to filter that list by label, or **--namespace-labels=<selector>** to filter all labels when the router service account has that permission.

Note

The label list is updated every 10 minutes or when the router restarts, so new projects may not instantly get served.

- Both the F5 and HAProxy routers now allow only the first route (by creation timestamp) with a given **host** or generated host (when you omit the **Host** field) to claim that route name. If multiple routes with the same host but different paths are defined, all routes in the same namespace as the oldest route with that host will be included. If the oldest route is deleted, and the next oldest route is in a different namespace, only routes in that other namespace will be served.

Images

- Importing or creating a new app with an image from Docker v2 registries is now supported.
- The **oc import-image** command can now create image streams with the **--from** flag, specifying the image repository you want to import.
- When you tag an image with **oc tag** into an image stream that does not exist, an image stream can now be automatically created.

Storage

- The **oc volume** command now lists by default and shows you additional information about each volume type.
- Persistent volume claims now show whether they are provisioned or not, their size, and details about their bound persistent volume. The **oc volume** command can also now create a new persistent volume claim for you if you specify the **--claim-size** flag.

CLI

- The **--list** flag has been added to the **oc new-app** command to display list of available images and templates.
- The **--short** and **-q** flags have been added to the **oc project** command to only display the project name.

Builds

- Custom builds now allow a **forcePull** flag to indicate that the custom builder image must be pulled.
- Multiple image change triggers are now allowed in build configurations.
- The **--commit=<commit>** flag has been added to **oc start-build**, which triggers a build of the exact Git commit specified.

- The **--env** flag has been added to **oc new-build**, allowing you to set environment variables on your S2I builds.
- The **--wait** flag has been added to **oc start-build**, allowing you to wait for the build completion without viewing the logs.

Templates

- The **required** attribute has been added to template parameters. Templates now cannot be instantiated without supplying a value for all required parameters.

Remote Execution

- The **oc rsh** command now accepts commands and arguments after the pod is specified:

  ```
  oc rsh <pod> <command> [<arguments>]
  ```

  This behavior more closely mimics the **ssh** command. A TTY is automatically allocated if your current shell passes one, otherwise you can specify **-t** to force a TTY or **-T** to disable it.

- A number of stability and hanging issues have been resolved with **oc exec** and **oc rsh**. However, Docker 1.6.2 has a known issue with hangs to remote shells via **docker exec**, so Red Hat recommends upgrading Docker to a 1.7 or 1.8 build.

Bug Fixes

For Developers

- The web console now allows users to specify the Git reference (branch or tag) from which their build will be created. (BZ#1250153)
- Users may now specify the replica count by adjusting the scale of a deployment configuration. This is useful for setting the replica count before a replication controller has been created so that the value will be used for replication controllers created in the future. (BZ#1250652)
- Docker client libraries were updated, and OpenShift can now import images from authenticated Docker v2 registries.(BZ#1255502)
- Git URI parsing has been updated to account for *git://* style URIs, and as a result builds using these URIs no longer fail. (BZ#1261449)
- Project administrators can now change a project's display name and description by updating the project using **oc edit**.
- Updated the set of labels generated when creating a new application from source in the web console, just as in the CLI with **oc new-app**.
- Improved the display of builds in the web console.
- Builds in which a pod is not created are no longer marked as successful in the web console.
- S2I builds that may run as root are now prevented from starting, based on security context constraints on the builder service account.
- Remote shell access to builder containers is now prevented.

- Builds are now listed in the CLI according to creation timestamp.
- Builds from **`oc new-app`** are now started immediately with theconfiguration change trigger.
- The help text for **`oc get projects`** has been fixed.
- Hangs when using **`oc exec`** without a TTY no longer occur.
- The **`oc import-image`** command no longer panics when an error occurs.
- The **`--search`** and **`--list`** options are now suggested when calling**`oc new-app`** with no arguments.
- When running **`oc scale`** against a deployment configuration with no deployments, the replicas are now set directly.

For Cluster Administrators

- Administrators can now configure the IP address used for SDN traffic. Passing node IPs as a configuration option on the node allows it to be set distinct from the node host name for listening on other interfaces. (BZ#1253596)
- SDN node events are now triggered when a node IP changes.
- The Rolling deployment strategy is now used for router deployments.
- The **`mode http`** configuration has been added to the HAProxy front end SNI definition.
- Upgraded the integrated **etcd** to v2.1.2.
- Upgraded the internal Docker registry to v2.0.1.
- The Kubernetes master service address (the first address in the service CIDR range) has been added to the generated certificates to allow pods to verify TLS connections to the API.
- Permissions are now preserved during image builds.
- Panics in the API server are now recovered instead of allowing the server to crash.
- The OpenShift SDN MTU is now configurable.

CHAPTER 3. APPLICATIONS

Applications in OpenShift v2

Applications have always been the focal point within OpenShift. In OpenShift v2, an application was a single unit - it consisted of one web framework and no more than one of any given cartridge type. So an application could have one PHP and one MySQL, for example, but it could not have one Ruby, one PHP, and two MySQLs. It also could not be a MySQL cartridge by itself.

The limited scoping for applications meant that OpenShift could perform seamless linking for all components within an application using well-defined environment variables. Every web framework knew how to connect to MySQL using the **`OPENSHIFT_MYSQL_DB_HOST`** and **`OPENSHIFT_MYSQL_DB_PORT`** variables, for example. But 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.

Applications in OpenShift v3

From OpenShift v2 it was clear that solving the problems of the entire application is essential. Most other PaaSes limit themselves to web frameworks and rely on external services for the other types of components. OpenShift v3 takes the next steps by making even more application topologies possible and making existing topologies more manageable.

The first step necessary to accomplish this is to remove "application" as a keyword since "application" can mean something different to everyone. Instead, you can have as many components as you desire, contained by a project, flexibly linked together, and optionally labeled to provide any groupings or structure. This new model allows for a standalone MySQL instance, or one shared between JBoss components, or really any combination of components you can imagine.

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

The result means you can build anything on OpenShift. And that is the problem OpenShift really aims to solve: a platform built on containers that lets you build entire applications in a repeatable lifecycle.

CHAPTER 4. CARTRIDGES VS IMAGES

4.1. OVERVIEW

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, with cartridges there was no clear distinction between the developer content and the cartridge content. For example, one limitation was that you did not have ownership of the home directory on each gear of your application. Another disadvantage was that 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.

4.2. DEPENDENCIES

Cartridge dependencies were defined with **`Configure-Order`** or **`Requires`** in the 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 is available before you start. Such a dependency check is always applicable and not just when you create the two components. Thus, pushing the dependency into a runtime dependency enables the system to stay healthy over time.

4.3. COLLOCATION

Whereas cartridges in OpenShift v2 could be collocated within gears, images in OpenShift v3 are mapped 1:1 with containers. Containers usepods as their collocation mechanism.

4.4. SOURCE CODE

In OpenShift v2, applications were required to have at least one web framework with one git repo. With 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.

4.5. BUILD

In OpenShift v2, builds happened in place as part of your application gears. This meant downtime for non scaled applications due to resource constraints. In v3, builds happen in separate containers.

Another change for builds is how those builds are transmitted between containers. In v2, build results were rsynced between 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 rollback to at a future date.

All of this means that builds are no longer part of the cartridges and are provided as distinct choices.

4.6. ROUTING

Routing is another area that has undergone a lot of formalization in OpenShift v3. In v2, you had to make choices up front as to whether your application was scalable, and a secondary choice as to whether the routing layer for your app was enabled for high availability. In OpenShift v3, routes are first class objects that are HA capable simply by scaling up your application component to 2 or more replicas. There is never a need to recreate your application or change its DNS entry.

The routes themselves have also been moved up a level and are disconnected from the images themselves. Previously, cartridges defined a default set of routes and you could add additional aliases to your applications. With v3, you can use templates to setup 0-N routes for any image. These routes let you modify the scheme, host, and paths exposed as desired, and there is no distinction between system routes and user aliases.

CHAPTER 5. TERMINOLOGY

5.1. OVERVIEW

Because of the architectural changes in OpenShift v3, a number of core terms used in OpenShift v2 have changed to better reflect the new model. The following sections highlight some of these important changes. See the Core Concepts topics for more detailed information on the concepts and objects in the new model.

Application

A specific *application* term or concept no longer exists in OpenShift v3. See the Applications topic for a more in-depth look at this change.

Cartridge vs Image

The easiest replacement term for *cartridge* in OpenShift v3 is *image*. An image does more than a cartridge from a packaging perspective, providing better encapsulation and flexibility. But the cartridge concept 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 templated configuration.

See the Cartridges vs Images topic for more detailed information on these changes.

Project vs Domain

Project is essentially a rename of *domain* from OpenShift v2. Projects do have several features that are not a part of domains in OpenShift v2.

Gear vs Container

The *gear* and *container* terms are interchangeable. Containers have a cleaner mapping of being one-to-one with images, whereas many cartridges could be added to a single gear. With containers, the collocation concept is satisfied by pods.

Master vs Broker

Masters in OpenShift v3 do the job of the *broker* layer in OpenShift v2. However, the MongoDB and ActiveMQ layers used by the broker in OpenShift v2 are no longer necessary because the key-value store **etcd** is typically installed with each master.

CHAPTER 6. REVISION HISTORY: WHAT’S NEW

6.1. TUE JUN 23 2015

OpenShift Enterprise 3.0 release.