



# **OpenShift Container Platform 3.3**

## **Release Notes**



# OpenShift Container Platform 3.3 Release Notes

---

## Legal Notice

Copyright © 2018 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

## Abstract

## Table of Contents

<b>CHAPTER 1. OVERVIEW</b> .....	<b>6</b>
1.1. VERSIONING POLICY	6
<b>CHAPTER 2. OPENSIFT CONTAINER PLATFORM 3.3 RELEASE NOTES</b> .....	<b>7</b>
2.1. OVERVIEW	7
2.2. ABOUT THIS RELEASE	7
2.3. SAME PRODUCT, NEW NAME: OPENSIFT CONTAINER PLATFORM	7
2.4. NEW FEATURES AND ENHANCEMENTS	7
2.4.1. Enterprise Container Registry	7
2.4.1.1. Registry User Interface	7
2.4.1.2. Unauthenticated Image Pull (Anonymous Access)	11
2.4.1.3. Google Cloud Storage as the Registry Storage Back End	11
2.4.1.4. Support for Docker Distribution 2.4	11
2.4.1.5. Allow Image "Pull-Through" from a Remote Registry	12
2.4.2. Developer Experience	12
2.4.2.1. OpenShift Pipelines (Technology Preview)	12
2.4.2.2. Jenkins Plug-in Enhancements	13
2.4.2.3. Easy and Quick Development Cluster Setup	13
2.4.2.4. Serialized Build Execution	13
2.4.2.5. Enhanced Source Code Synchronization	13
2.4.2.6. Build Trigger Cause Tracking	14
2.4.2.7. Webhook Improvements	16
2.4.2.8. Self-tuning Images	16
2.4.3. Web Console	16
2.4.3.1. Usability Improvements: Project Overview	16
2.4.3.2. Usability Improvements: Project Navigation	17
2.4.3.3. New Concept: OpenShift Pipelines	18
2.4.3.4. New Concept: A/B Routing	19
2.4.3.5. Deploy Image	19
2.4.3.6. Import YAML / JSON	20
2.4.3.7. Other Resources	21
2.4.3.8. Monitoring	22
2.4.3.9. Debugging	22
2.4.3.10. Image Details	24
2.4.4. Networking	25
2.4.4.1. Controllable Source IP	25
2.4.4.2. Router Sharding	26
2.4.4.3. Non-Standard Ports	26
2.4.4.4. A/B Service Annotation	27
2.4.5. Security	27
2.4.5.1. SCC Profiles for seccomp	27
2.4.5.2. Kerberos Support in oc client for Linux	27
2.4.5.3. Certificate Maintenance	27
2.4.6. Cluster Longevity	28
2.4.6.1. Pod Eviction	28
2.4.6.2. Scale	28
2.4.6.3. Idling and Unidling	28
2.4.6.4. Storage Labels	28
2.4.7. Framework Services	29
2.4.7.1. Logging Enhancements	29
2.4.7.2. Metrics Installation Enhancement	29

2.5. NOTABLE TECHNICAL CHANGES	29
2.6. BUG FIXES	33
Authentication	33
Builds	33
Command Line Interface	34
Images	34
Image Registry	34
Installer	34
Kubernetes	35
Logging	35
Web Console	36
Metrics	37
Networking	37
Quick Starts	37
REST API	37
Routing	37
Storage	38
Upgrades	38
2.7. TECHNOLOGY PREVIEW FEATURES	39
2.8. KNOWN ISSUES	39
2.9. ASYNCHRONOUS ERRATA UPDATES	39
2.9.1. RHBA-2016:1988 - OpenShift Container Platform 3.3.0.34 Bug Fix Update	40
2.9.1.1. Upgrading	40
2.9.2. RHBA-2016:2084 - OpenShift Container Platform 3.3.1.3 Bug Fix and Enhancement Update	40
2.9.2.1. Upgrading	40
2.9.2.2. Technology Preview Features	40
2.9.2.3. Bug Fixes	41
2.9.3. RHBA-2016:2122 - atomic-openshift-utils Bug Fix Update	43
2.9.3.1. Upgrading	43
2.9.3.2. Bug Fixes	43
2.9.3.3. Known Issues	45
2.9.4. RHSA-2016:2696 - OpenShift Container Platform 3.3.1.4 Security and Bug Fix Update	46
2.9.4.1. Upgrading	46
2.9.5. RHBA-2016:2801 - OpenShift Container Platform 3.3.1.5 Bug Fix Update	46
2.9.5.1. Upgrading	46
2.9.6. RHSA-2016:2915 - OpenShift Container Platform 3.3.1.7 Security and Bug Fix Update	46
2.9.6.1. Upgrading	46
2.9.7. RHBA-2017:0199 - OpenShift Container Platform 3.3.1.11 Bug Fix Update	46
2.9.7.1. Upgrading	47
2.9.7.2. Bug Fixes	47
2.9.8. RHBA-2017:0289 - OpenShift Container Platform 3.3.1.14 Bug Fix Update	48
2.9.8.1. Upgrading	48
2.9.8.2. Bug Fixes	48
2.9.9. RHSA-2017:0448 - ansible and openshift-ansible Security and Bug Fix Update	49
2.9.9.1. Upgrading	49
2.9.9.2. Bug Fixes	49
Installer	50
Upgrades	50
2.9.10. RHBA-2017:0512 - OpenShift Container Platform 3.3.1.17 Bug Fix Update	50
2.9.10.1. Upgrading	50
2.9.10.2. Bug Fixes	50
Metrics	50
Storage	51

2.9.11. RHBA-2017:0865 - OpenShift Container Platform 3.3.1.17-4 Bug Fix Update	51
2.9.11.1. Upgrading	51
2.9.12. RHBA-2017:0989 - OpenShift Container Platform 3.3.1.19 Bug Fix Update	51
2.9.12.1. Upgrading	51
2.9.13. RHBA-2017:1129 - OpenShift Container Platform 3.3.1.20 Bug Fix Update	51
2.9.13.1. Upgrading	52
2.9.14. RHBA-2017:1235 - OpenShift Container Platform 3.3.1.25 Bug Fix Update	52
2.9.14.1. Upgrading	52
2.9.15. RHBA-2017:1235 - OpenShift Container Platform 3.3.1.35 Bug Fix Update	52
2.9.15.1. Upgrading	52
2.9.15.2. Bug Fixes	52
2.9.15.3. Images	53
2.9.16. RHBA-2017:1492 - OpenShift Container Platform 3.3.1.38 Bug Fix Update	53
2.9.16.1. Upgrading	53
2.9.17. RHBA-2017:1666 - atomic-openshift-utils Bug Fix and Enhancement Update	53
2.9.17.1. Upgrading	54
2.9.17.2. Bug Fixes	54
2.9.17.3. Enhancements	54
2.9.18. RHBA-2017:1828 - OpenShift Container Platform 3.3.1.46 Bug Fix Update	55
2.9.18.1. Images	55
2.9.18.2. Upgrading	55
2.9.19. RHSA-2018:3754 - OpenShift Container Platform 3.3.1.46.45 Security and Bug Fix Update	55
2.9.19.1. Upgrading	56
<b>CHAPTER 3. XPAAS RELEASE NOTES</b>	<b>57</b>
<b>CHAPTER 4. COMPARING WITH OPENSIFT ENTERPRISE 2</b>	<b>58</b>
4.1. OVERVIEW	58
4.2. ARCHITECTURE CHANGES	58
4.3. APPLICATIONS	58
4.4. CARTRIDGES VS IMAGES	59
4.5. BROKER VS MASTER	60
4.6. DOMAIN VS PROJECT	60
<b>CHAPTER 5. REVISION HISTORY: RELEASE NOTES</b>	<b>61</b>
5.1. THU AUG 31 2017	61
5.2. WED JUL 12 2017	61
5.3. THU JUN 29 2017	61
5.4. THU JUN 22 2017	61
5.5. WED JUNE 14 2017	61
5.6. THU MAY 18 2017	61
5.7. TUE APR 25 2017	62
5.8. THU APR 06 2017	62
5.9. WED MAR 15 2017	62
5.10. TUE MAR 07 2017	62
5.11. WED FEB 22 2017	62
5.12. THU JAN 26 2017	63
5.13. WED DEC 07 2016	63
5.14. MON NOV 21 2016	63
5.15. THU NOV 17 2016	63
5.16. TUE NOV 15 2016	63
5.17. WED NOV 02 2016	64
5.18. THU OCT 27 2016	64
5.19. WED OCT 05 2016	64

5.20. TUE OCT 04 2016

64

5.21. TUE SEP 27 2016

64





## CHAPTER 1. OVERVIEW

The following release notes for OpenShift Container Platform 3.3 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 3.5 cluster, all masters must be 3.5 and all nodes must be 3.5. However, OpenShift Container Platform will continue to support older **oc** clients against newer servers. For example, a 3.4 **oc** will work against 3.3, 3.4, and 3.5 servers.

Changes of APIs for non-security related reasons will involve, at minimum, two minor releases (3.1 to 3.2 to 3.3, 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 <sup>[a]</sup> (oc Client)
X.Y (Server)	<b>1</b>	<b>3</b>
X.Y+N <sup>[a]</sup> (Server)	<b>2</b>	<b>1</b>
[a] Where <b>N</b> 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. OPENSIFT CONTAINER PLATFORM 3.3 RELEASE NOTES

## 2.1. OVERVIEW

Red Hat OpenShift Container Platform 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 Container Platform supports a wide selection of programming languages and frameworks, such as Java, Ruby, and PHP.

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

## 2.2. ABOUT THIS RELEASE

Red Hat OpenShift Container Platform version 3.3 ([RHBA-2016:1933](#)) is now available. This release is based on [OpenShift Origin 1.3](#). New features, changes, bug fixes, and known issues that pertain to OpenShift Container Platform 3.3 are included in this topic.

For initial installations, see the [Installing a Cluster](#) topics in the [Installation and Configuration](#) documentation.

To upgrade to this release from a previous version, see the [Upgrading a Cluster](#) topics in the [Installation and Configuration](#) documentation.

## 2.3. SAME PRODUCT, NEW NAME: OPENSIFT CONTAINER PLATFORM

OpenShift Enterprise is now officially known as OpenShift Container Platform starting with version 3.3. It is the same product in every way; only the name has been changed. The product itself (code, CLI, web console) has been updated starting with version 3.3 to reflect this new name, as has the documentation.

While OpenShift Container Platform is now the official name of the product, the name change is not being backported to previous minor releases of OpenShift Container Platform 3. This means the product and documentation for versions 3.0, 3.1, and 3.2 will remain showing the OpenShift Enterprise name. However, all version 3.3 and later releases of the product and documentation will continue to use the OpenShift Container Platform name.

## 2.4. NEW FEATURES AND ENHANCEMENTS

### 2.4.1. Enterprise Container Registry

This release adds the following improvements to the registry and its user experience.

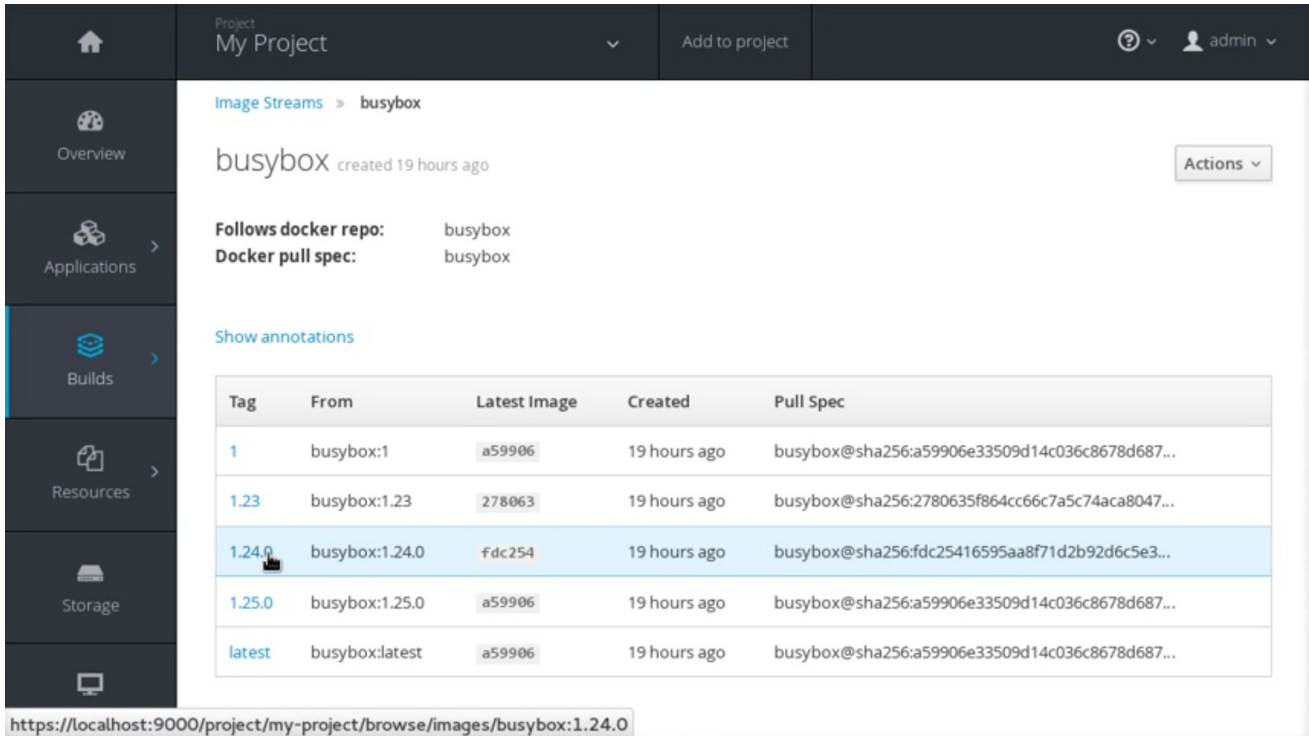
#### 2.4.1.1. Registry User Interface

The OpenShift Container Platform provides a registry to manage container images. The most noticeable improvements to the registry in 3.3 are in the user interface and its deployment options. The registry can be deployed one of two ways: as an integrated registry in the OpenShift Container Platform web console or as a stand-alone registry.

## Integrated Registry User Interface

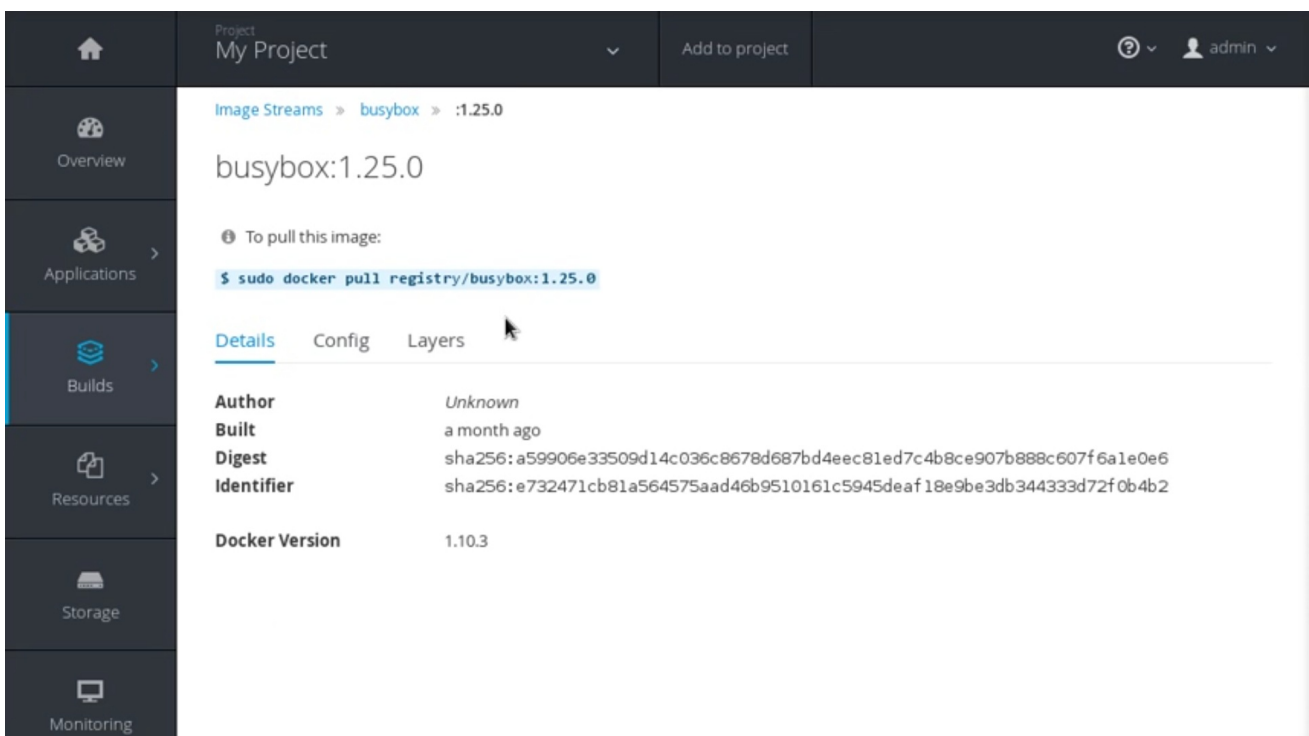
The updated OpenShift Container Platform integrated registry provides details about the container images it manages and their tagged versions from within the OpenShift Container Platform web console. Each tag of an image stream, from the **Builds** → **Images** view, is now a hyperlink that leads to a multi-tabbed page with more information about the selected image.

Figure 2.1. Integrated Registry User Interface



The **Details** tab shows the image author, built-on date, digest hash ID, labels, annotations, and docker version for compatibility comparisons. The **Config** tab shows how the container image was built and its set of defined metadata labels. The **Layers** tab shows the size and digest ID for each layer of the image.

Figure 2.2. Integrated Registry User Interface: Details Tab



## Stand-Alone Registry User Interface

The OpenShift Container Platform registry can alternatively be installed as a [stand-alone container image registry](#) to run on-premise or in the cloud, deployed as either an all-in-one cluster (running the master, node, etcd and registry components) or in a highly-available configuration (three hosts running all components on each, with the masters configured for native high-availability).

The web console of the stand-alone registry is based on the [Cockpit project](#), with full API and CLI access. Traffic to/from the stand-alone registry is secured by default with TLS.

**Figure 2.3. Stand-alone Registry User Interface: Details Tab**

The screenshot displays the 'Details Tab' of the Stand-alone Registry User Interface. On the left, a dark sidebar contains navigation options: 'Overview', 'Images', and 'Projects'. The main content area is split into two panels. The top-left panel, 'Images by project', lists three projects: 'david', 'redhat', and 'secret', each with a lock icon. The top-right panel, 'Images pushed recently', features a dropdown menu set to 'All Projects' and shows two image streams: 'redhat/rhev-guest-agent' (pushed 9 minutes ago) and 'redhat/rhel7' (pushed 10 minutes ago). Below these panels are buttons for 'New project' and 'New image stream'. The bottom section, 'Docker commands', provides instructions on how to log into the registry, push an image, and pull an image.

**Docker commands**

**Log into the registry:**

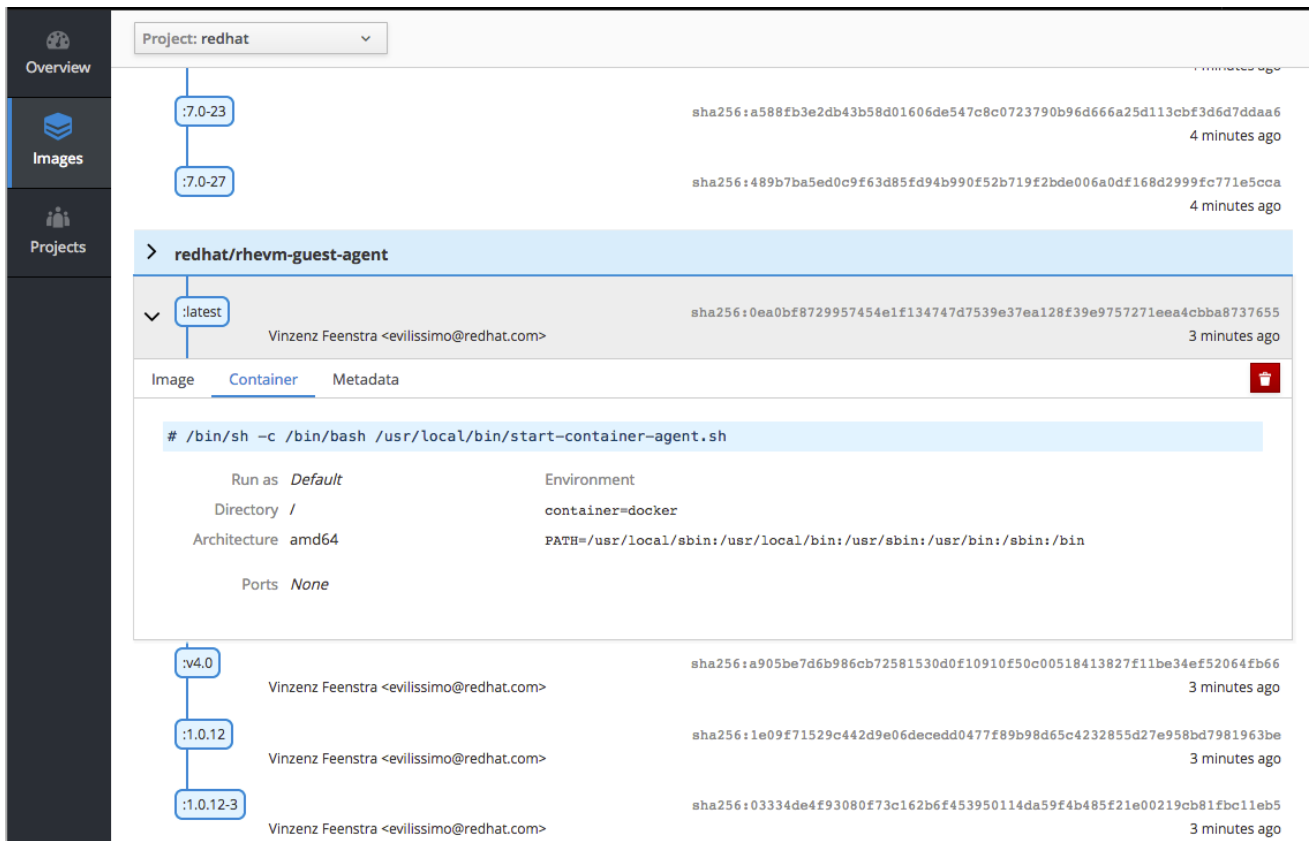
```
$ sudo docker login -p poTxgocJC795FK5QAciLHaC1J8mgzB_hJz7R888ChZs -e unused -u unused 10.3.11.254:5000
```

**Push an image:**

```
$ sudo docker tag myimage 10.3.11.254:5000/project/name:tag
$ sudo docker push 10.3.11.254:5000/project/name
```

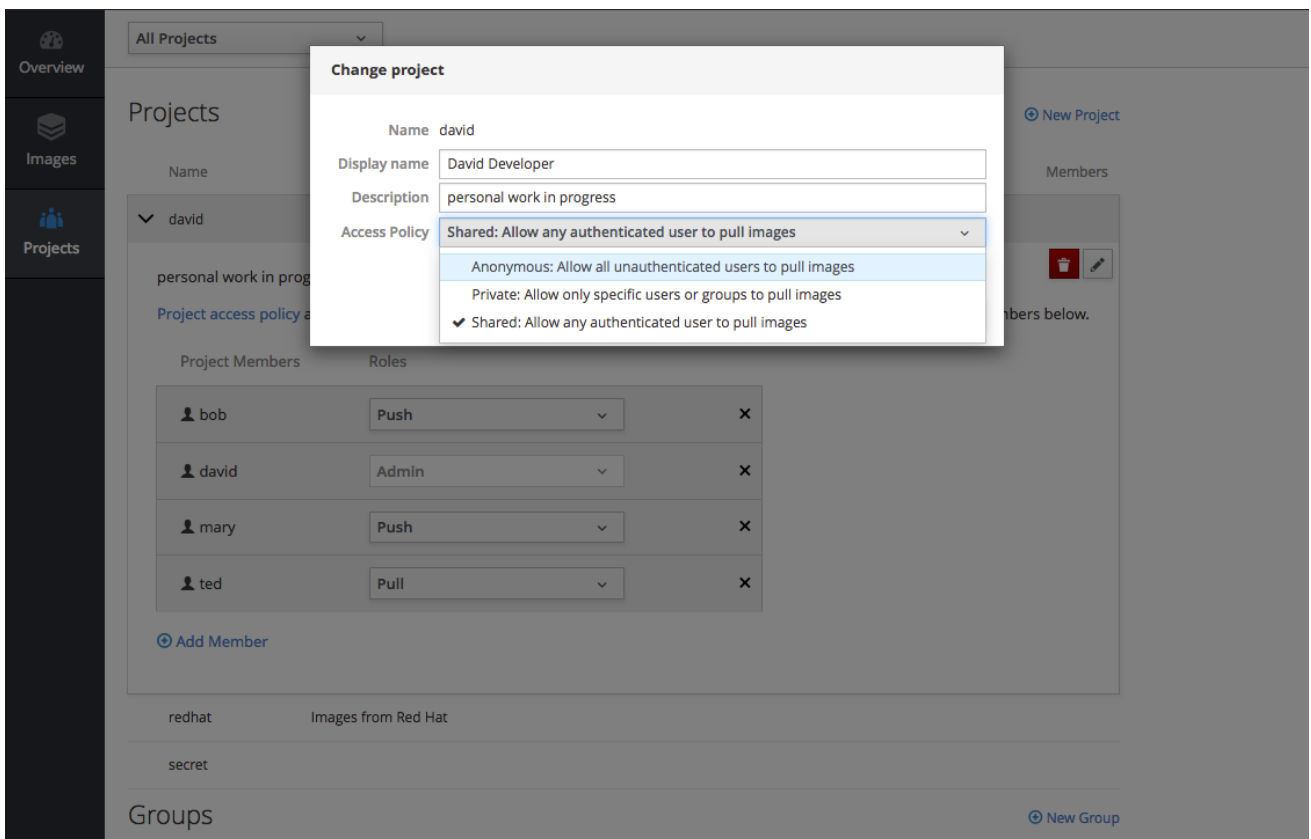
**Pull an image:**

Figure 2.4. Stand-alone Registry User Interface: Image Metadata



This container image registry viewer provides a more "hub-like" direct access experience with registry images, and allows users to manage role-based access control (RBAC) to authorize image delivery on a per user and per project basis for **oc policy** and **oc projects** functionality right from within the user interface.

Figure 2.5. Stand-alone Registry User Interface: Project RBAC



It has its own built-in OAuth server for a single sign-on (SSO) user experience and is easily integrated with common enterprise and cloud identity providers. The stand-alone registry management interface also has flexible options for persistent storage of the images it manages and their metadata.

### 2.4.1.2. Unauthenticated Image Pull (Anonymous Access)

This new feature provides the ability to pull images from the OpenShift Container Platform integrated registry without a docker login, to facilitate automation and users who want the ability to simply pull an image.

To enable this, the project administrator (a user with the **registry-admin** role) can assign the **registry-viewer** role with the following command:

```
$ oc policy add-role-to-group registry-viewer system:unauthenticated
```

### 2.4.1.3. Google Cloud Storage as the Registry Storage Back End

OpenShift Container Platform 3.3 adds a Google Cloud Storage (GCS) driver to enable its use as the storage back end for the registry's container images. Prior to GCS driver initialization, the OpenShift Container Platform admin must set a [bucket parameter](#) to define the name of the GCS bucket to store objects in.

### 2.4.1.4. Support for Docker Distribution 2.4

The OpenShift Container Platform 3.3 registry is based on Docker Distribution registry 2.4, and its features will be backported to OpenShift Container Platform 3.2. Version 2.4 of the registry includes a variety of performance and usability enhancements, notably:

#### Cross-repo Mounting When Pushing Images That Already Exist in the Registry

When a client wishes to push a blob to a target repository from a primary source, and knows that the blob already exists in a secondary source repository on the same server as the target, this feature gives the user the ability to optimize the push by requesting the server cross-mount the blob from the secondary source repository, speeding up push time.

Of course, the client must have proper authorizations (pull and push on the target repository, pull on the secondary source repository). If the client is not authorized to pull from the secondary source repository, the blob push will proceed, unoptimized, and the client will push the entire blob to the target repository from the primary source repository without assistance from the secondary source repository.

#### Support for the New schema 2 Storage Format for Images

The image manifest version 2, **schema 2**, introduces the ability to hash an image's configuration and thus reduce manifest size, to create an ID for the image, and provide content-addressable information about the image.

Support for consuming **schema 2** from external resources is enabled implicitly. However, images pushed to the internal registry will be converted to **schema 1** for compatibility with older docker versions. For clusters not in need of compatibility preservation, accepting of **schema 2** during pushes can be manually enabled:

```
$ oc login -u system:admin
$ oc set env dc/docker-registry -n default
REGISTRY_MIDDLEWARE_REPOSITORY_OPENSIFT_ACCEPTSCHEMA2=true
```

Manifest lists, introduced by **schema 2** to provide support for multi-architecture images (e.g., amd64 versus ppc64le), are not yet supported in OpenShift Container Platform 3.3.

#### 2.4.1.5. Allow Image "Pull-Through" from a Remote Registry

The OpenShift Container Platform integrated registry allows remote public and private images to be tagged into an image stream and "pulled-through" it, as if the image were already pushed to the OpenShift Container Platform registry. Authentication credentials required for private images to create the image stream are re-used by the integrated registry for subsequent pull-through requests to the remote registry.

The content-offload optimization configuration is still honored by pull-through requests. If the pull-through request points to a remote registry configured with both a storage back end (for example, GCS, S3, or Swift storage) and content-offload enabled, a redirect URL that points directly to the blobs on the remote back end storage will be passed through the local registry to the local docker daemon, creating a direct connection to the remote storage for the blobs.

To optimize image and blob lookups for pull-through requests, a small cache is kept in the registry to track which image streams have the manifest for the requested blobs, avoiding a potentially costly multi-server search.

### 2.4.2. Developer Experience

This release adds the following improvements to the developer workflow when developing and testing applications on OpenShift Container Platform.

#### 2.4.2.1. OpenShift Pipelines (Technology Preview)

Previously with CI/CD, it was possible to define small pipeline-like workflows, such as triggering deployments after a new image was built or building an image when upstream source code changed. OpenShift Pipelines (currently in [Technology Preview](#)) now expose a true first class pipeline execution capability. OpenShift Pipelines are based on the [Jenkins Pipeline plug-in](#). By integrating Jenkins Pipelines into OpenShift, you can now leverage the full power and flexibility of the Jenkins ecosystem while managing your workflow from within OpenShift.



#### NOTE

See [New Features and Enhancements: Web Console](#) for more details on the new pipelines user interface.

Pipelines are defined as a new build strategy within OpenShift Container Platform, meaning you can start, cancel, and view your pipelines in the same way as any other build. Because your pipeline is executed by Jenkins, you can also use the Jenkins console to view and manage your pipeline.

Finally, your pipelines can utilize the [OpenShift Pipeline plug-in](#) to easily perform first class actions in your OpenShift Container Platform cluster, such as triggering builds and deployments, tagging images, or verifying application status.

To keep the system fully integrated, the Jenkins server executing your pipeline can run within your cluster, launch Jenkins slaves on that same cluster, and OpenShift Container Platform can even automatically deploy a Jenkins server if one does not already exist when you first declare a new pipeline build configuration.

See the following for more on pipelines:



- [Pipeline Concept](#)
- [Configuring Pipeline Execution](#)
- [Pipeline Strategy Option](#)

#### 2.4.2.2. Jenkins Plug-in Enhancements

The Jenkins plug-in now provides full integration with the Jenkins Pipeline, exposing the same OpenShift Container Platform build steps available in the classic, "freestyle" jobs as Jenkins Pipeline DSL methods (replacing the Java language invocations previously available from the Jenkins Pipeline Groovy scripts).

Several user requested features have also been introduced, including:

- Exposing "Scale OpenShift Deployments" as a post-build action
- Additional configuration available at the specific step level for triggering builds and deployments
- Embeddable use of job parameters for configuration of specific step fields

#### 2.4.2.3. Easy and Quick Development Cluster Setup

Often a developer will want to have a stand-alone OpenShift Container Platform instance running on their desktop to enable evaluation of various features or developer and testing locally of their containerized applications containers. Launching a local instance of OpenShift Container Platform for application development is now as easy as downloading the latest client tools and running:

```
$ oc cluster up
```

This provides a running cluster using your local **docker** daemon or Docker Machine. All the basic infrastructure of the cluster is automatically configured for you: a registry, router, image streams for standard images, and sample templates.

It also creates a normal user and system administrator accounts for managing the cluster.

#### 2.4.2.4. Serialized Build Execution

Prior to OpenShift Container Platform 3.3, if multiple builds were created for a given build configuration, they all ran in parallel. This resulted in a race to the finish, with the last build to push an application image to the registry winning. This also lead to higher resource utilization peaks when multiple builds ran at the same time.

Now with OpenShift Container Platform 3.3, builds run serially by default. It is still possible to revert to the parallel build policy if desired. In addition, the new **SerialLatestOnly** policy runs builds in serial, but skips intermediary builds. In other words, if build 1 is running and builds 2, 3, 4, and 5 are in the queue, when build 1 completes the system will cancel builds 2 through 4 and immediately run build 5. This allows you to optimize your build system around building the latest code and not waste time building intermediate commits.

For more information, see [Build Run Policy](#).

#### 2.4.2.5. Enhanced Source Code Synchronization

The **oc rsync** command was added previously, allowing synchronizing of a local file system to a running container. This is a very useful tool for copying files into a container in general, but in particular it

can be used to synchronize local source code into a running application framework. For frameworks that support hot deployment when files change, this enables an extremely responsive "code, save, debug" workflow with source on the developer's machine using the their IDE of choice, while the application runs in the cloud with access to any service it depends on, such as databases.

This sync flow is made even easier with this release by coupling it with a file system watch. Instead of manually syncing changes, developers can now run `oc rsync --watch`, which launches a long running process that monitors the local file system for changes and continuously syncs them to the target container. Assuming the target container is running a framework that supports hot reload of source code, the development workflow is now: "save file in IDE, reload application page in browser, see changes."

For more information, see [Continuous Syncing on File Change](#).

#### 2.4.2.6. Build Trigger Cause Tracking

While OpenShift Container Platform has always automatically run a build of your application when source changes or an upstream image that your application is built on top of has been updated, prior to OpenShift Container Platform 3.3 it was not easy to know why your application had been rebuilt. With OpenShift Container Platform 3.3, builds now include information explaining what triggered the build (manual, image change, webhook, etc.) as well as details about the change, such as the image or commit ID associated with the change.

##### A build triggered by an image change

Output provided by CLI command `oc describe build`:

```
$ oc describe build ruby-sample-build-2
Name: ruby-sample-build-2
.....
Status: Running
Started: Fri, 09 Sep 2016 16:39:46 EDT
Duration: running for 10s
Build Config: ruby-sample-build
Build Pod: ruby-sample-build-2-build

Strategy: Source
URL: https://github.com/openshift/ruby-hello-world.git
From Image: DockerImage centos/ruby-23-
centos7@sha256:940584acbbfb0347272112d2eb95574625c0c60b4e2fdadb139de5859cf
754bf
Output to: ImageStreamTag origin-ruby-sample:latest
Post Commit Hook: ["", "bundle", "exec", "rake", "test"]
Push Secret: builder-dockercfg-awr0v

Build trigger cause:Image change
Image ID:centos/ruby-23-
centos7@sha256:940584acbbfb0347272112d2eb95574625c0c60b4e2fdadb139de5859cf
754bf
Image Name/Kind: ruby:latest / ImageStreamTag
```

Then, within the web console:

Figure 2.6. Build Triggered by Image Change

The screenshot shows the OpenShift web console interface. The top navigation bar includes a home icon, the project name 'p1', an 'Add to project' button, and a user profile 'user1'. The left sidebar contains navigation options: Overview, Applications, Builds (highlighted), Resources, Storage, and Monitoring. The main content area displays the details for a build named 'ruby-sample-build-2', which was created 5 minutes ago. The build status is 'Complete'. The 'Triggered by' field indicates it was triggered by an 'Image change for ruby:latest'. Below the status, the 'Configuration' section lists the following details:

- Build strategy:** Source
- Builder image:** centos/ruby-23-centos7@sha256:940584acbbfb0347272112d2eb95574625c0c60b4e2fdadb139de5859cf754bf
- Source type:** Git
- Source repo:** <https://github.com/openshift/ruby-hello-world.git>
- Output image:** p1/origin-ruby-sample:latest
- Push secret:** builder-dockercfg-awr0v

Buttons for 'Rebuild' and 'Actions' are visible in the top right of the build details view.

### A build triggered by a webhook

Output provided by CLI command `oc describe build`:

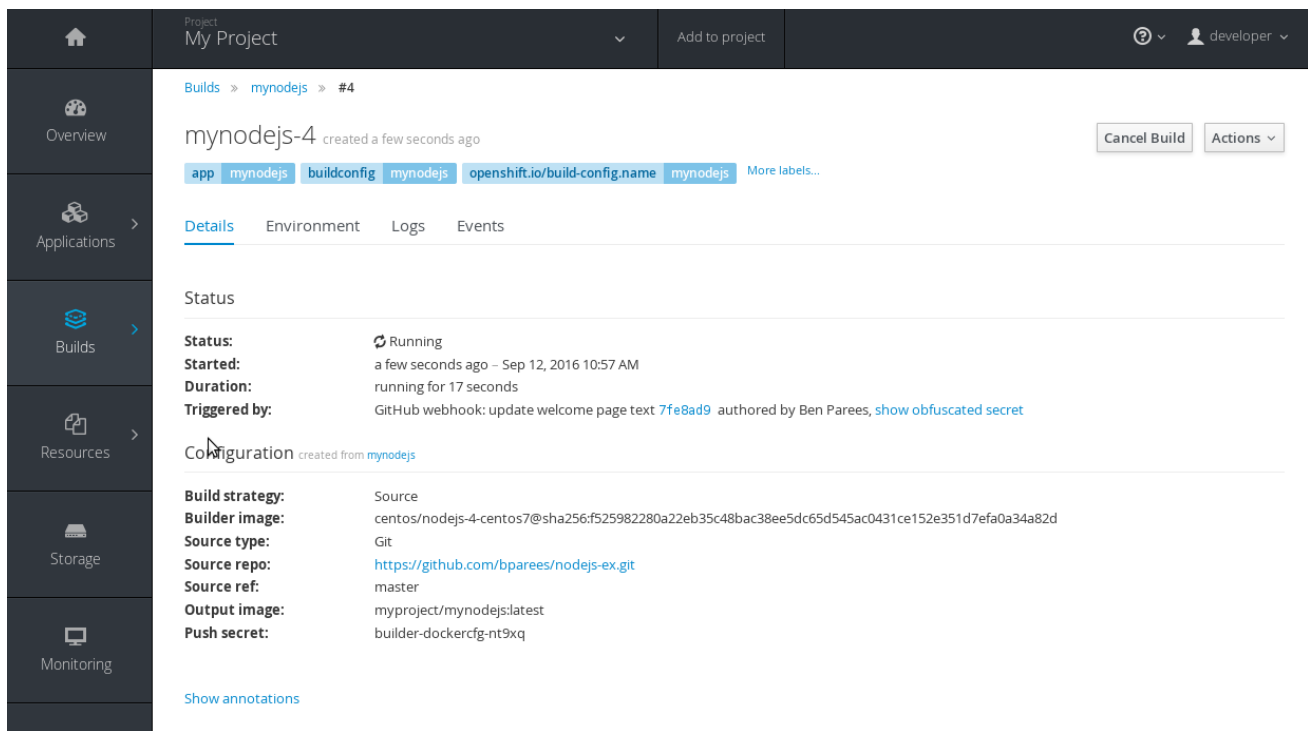
```
$ oc describe build mynodejs-4
Name: mynodejs-4
.....
Status: Complete
Started: Mon, 12 Sep 2016 04:57:44 EDT
Duration: 20s
Build Config: mynodejs
Build Pod: mynodejs-4-build

Strategy: Source
URL: https://github.com/bparees/nodejs-ex.git
Ref: master
Commit: 7fe8ad9 (update welcome page text)
Author/Committer: Ben Parees
From Image: DockerImage centos/nodejs-4-
centos7@sha256:f525982280a22eb35c48bac38ee5dc65d545ac0431ce152e351d7efa0a3
4a82d
Output to: ImageStreamTag mynodejs:latest
Push Secret: builder-dockercfg-nt9xq

Build trigger cause:GitHub WebHook
Commit:7fe8ad9 (update welcome page text)
Author/Committer:Ben Parees
Secret: 34c64fd2***
```

Then, within the web console:

Figure 2.7. Build Triggered by Webhook



### 2.4.2.7. Webhook Improvements

It is now possible to provide additional inputs to webhook triggered builds. Previously, the generic webhook simply started a new build with all the default values inherited from the build configuration. It is now possible to provide a payload to the webhook API.

The payload can provide Git information so that a specific commit or branch can be built. Environment variables can also be provided in the payload. Those environment variables are made available to the build in the same way as environment variables defined in the build configuration.

For examples of how to define a payload and invoke the webhook, see [Generic Webhooks](#).

### 2.4.2.8. Self-tuning Images

OpenShift Container Platform provides a number of framework images for working with Java, Ruby, PHP, Python, NodeJS, and Perl code. It also provides a few database images (MySQL, MongoDB, PostgreSQL) out of the box. For OpenShift Container Platform 3.3, these images are improved by making them self-tuning.

Based on the container memory limits specified when the images are deployed, these images will automatically configure parameters like heap sizes, cache sizes, number of worker threads, and more. All these automatically-tuned values can easily be overridden by environment variables, as well.

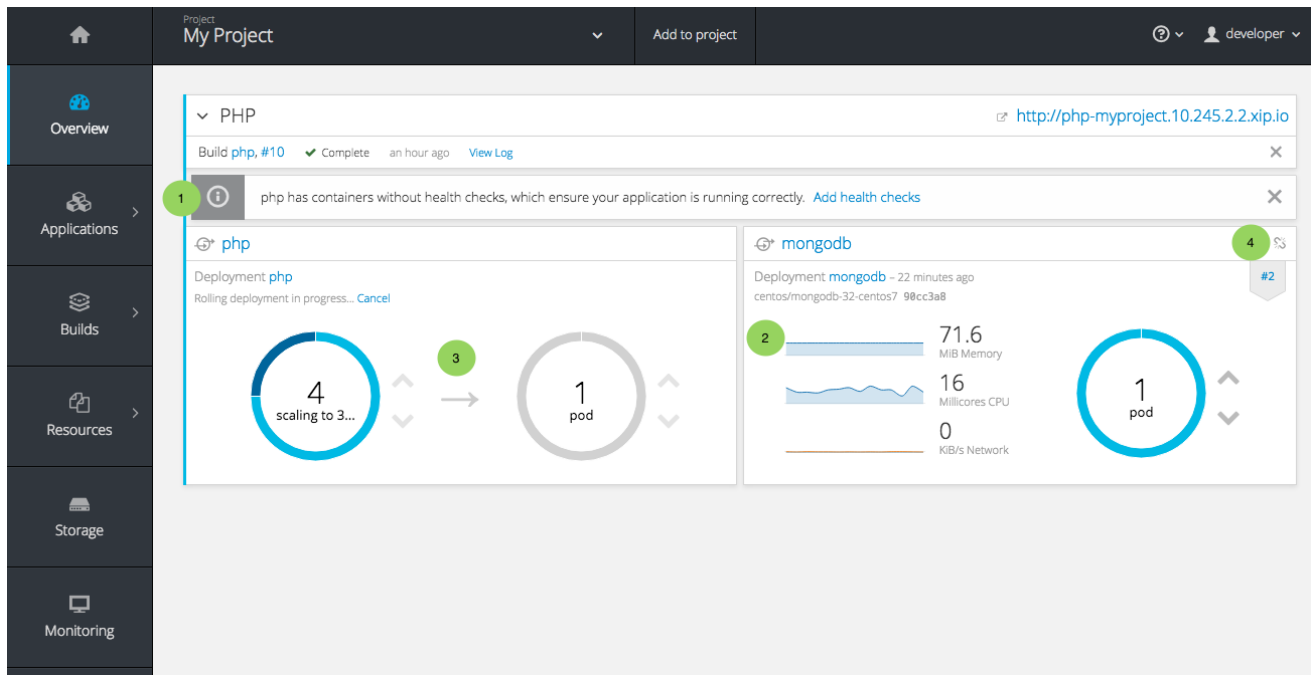
## 2.4.3. Web Console

This release adds the following improvements to the web console, including updates to existing features, usability overhauls, and a few brand new concepts.

### 2.4.3.1. Usability Improvements: Project Overview

The web console's **Overview** is the landing page for your project. At a glance, you should be able to see what is running in your project, how things are related, and what state they are in. To that end, the re-designed overview now includes the following:

**Figure 2.8. New Project Overview**



Warnings, suggestions, and other notifications in context

Metrics for a deployment or pod

Better awareness of deployment status (animation of rolling deployments, cancel in-progress deployments, and wake up idled deployments)

Grouping of related services

### 2.4.3.2. Usability Improvements: Project Navigation

Previously, most of the concepts in OpenShift Container Platform were hidden underneath a generic **Browse** menu. An exercise to define the information architecture resulted in the new left sidebar project navigation.

Overview	The dashboard for your project.
Applications	Everything that make up your running application. This means pods, things that create or replicate pods, and anything that controls the flow of network traffic to pods.
Builds	Builds, pipelines, and build artifacts, like images.
Resources	Resource restrictions like limit ranges, project quotas, and cluster quotas. Also, other advanced resources in your project that do not fit into one of the top level concepts.
Storage	View your existing persistent volume claims (PVCs) and request persistent storage.

Monitoring	A single page that gives you access to logs, metrics, and events.
------------	-------------------------------------------------------------------

### 2.4.3.3. New Concept: OpenShift Pipelines

A new set of pages have been added dedicated to the new [OpenShift Pipelines](#) feature (currently in [Technology Preview](#)) that allow you to visualize your pipeline's stages, edit the configuration, and manually kick off a build. Pipelines paused waiting for manual user intervention provide a link to the Jenkins pipeline interface.

**Figure 2.9. OpenShift Pipelines Details**

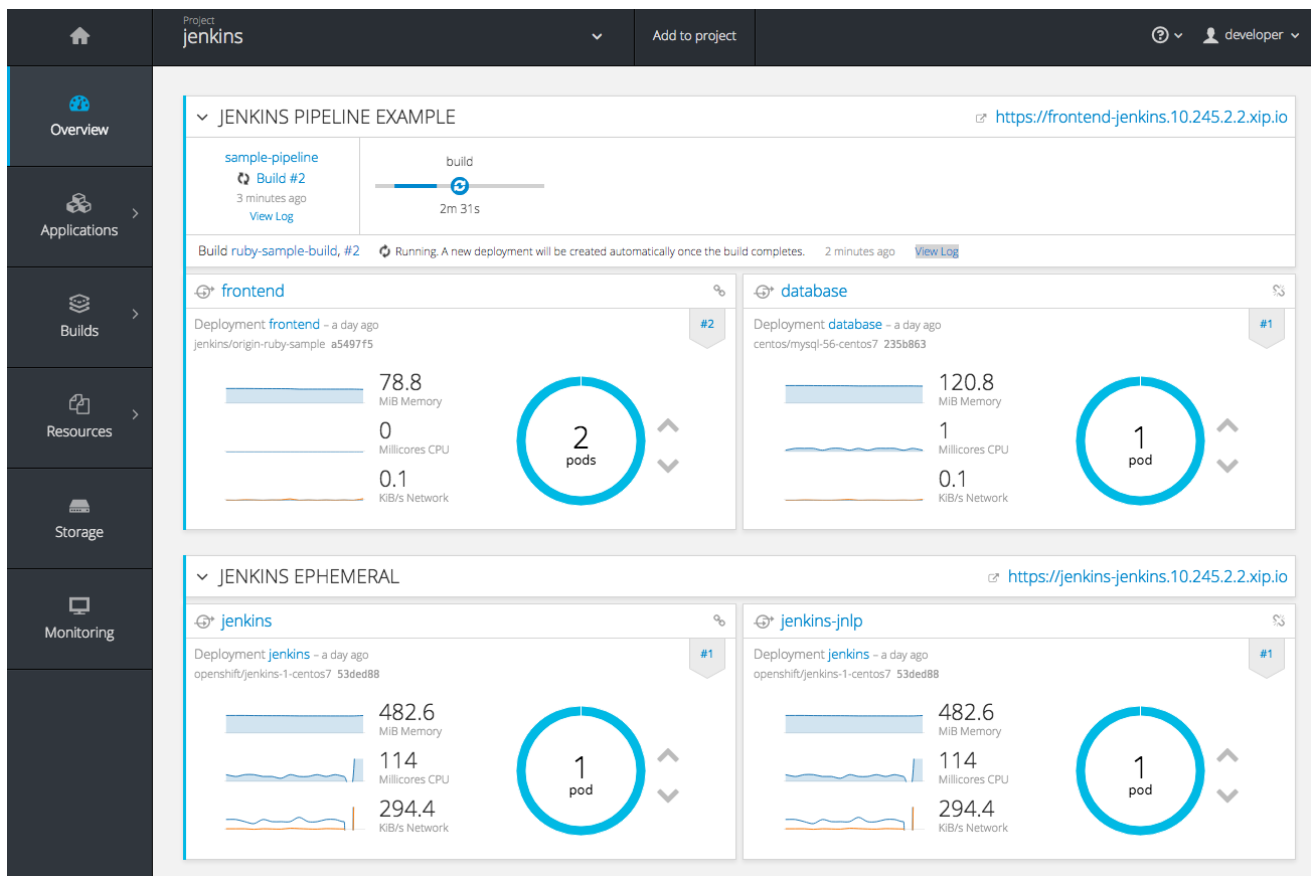
The screenshot shows the OpenShift Pipelines interface. On the left is a dark sidebar with navigation icons for Overview, Applications, Builds, Resources, Storage, and Monitoring. The main area is titled 'Pipelines' for 'Project Pipeline Example'. It displays a 'sample-pipeline' created 6 minutes ago with a 'Start Pipeline' button. Below this, 'Recent Runs' are shown in a table-like format:

Build	Age	Build Stage	Build Duration	Deploy Stage	Deploy Duration
Build #2	a minute ago	build	1m 21s	deploy	1s
Build #1	3 minutes ago	build	1m 32s	deploy	20s

At the bottom of the 'Recent Runs' section, there are links for 'View History' and 'Edit Pipeline'. The 'Average Duration' for the pipeline is noted as 1m 54s.

Running or recently completed pipeline builds also show up on the new **Overview** page if they are related to a deployment configuration.

Figure 2.10. Project Overview with Pipelines

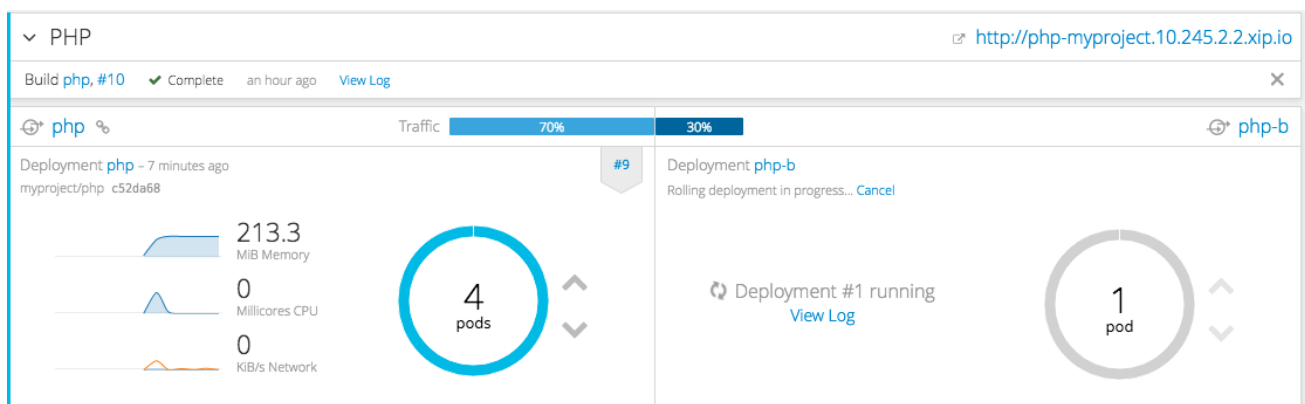


Because OpenShift Pipelines are currently in [Technology Preview](#), you must enable pipelines in the primary navigation of the web console to use this feature. See [Enabling Features in Technology Preview](#) for instructions.

#### 2.4.3.4. New Concept: A/B Routing

In OpenShift Container Platform 3.3, routes can now point to multiple back end services, commonly called [A/B deployments](#). Routes configured in this way will automatically group the related services and visualize the percentage of traffic configured to go to each one.

Figure 2.11. A/B Routes

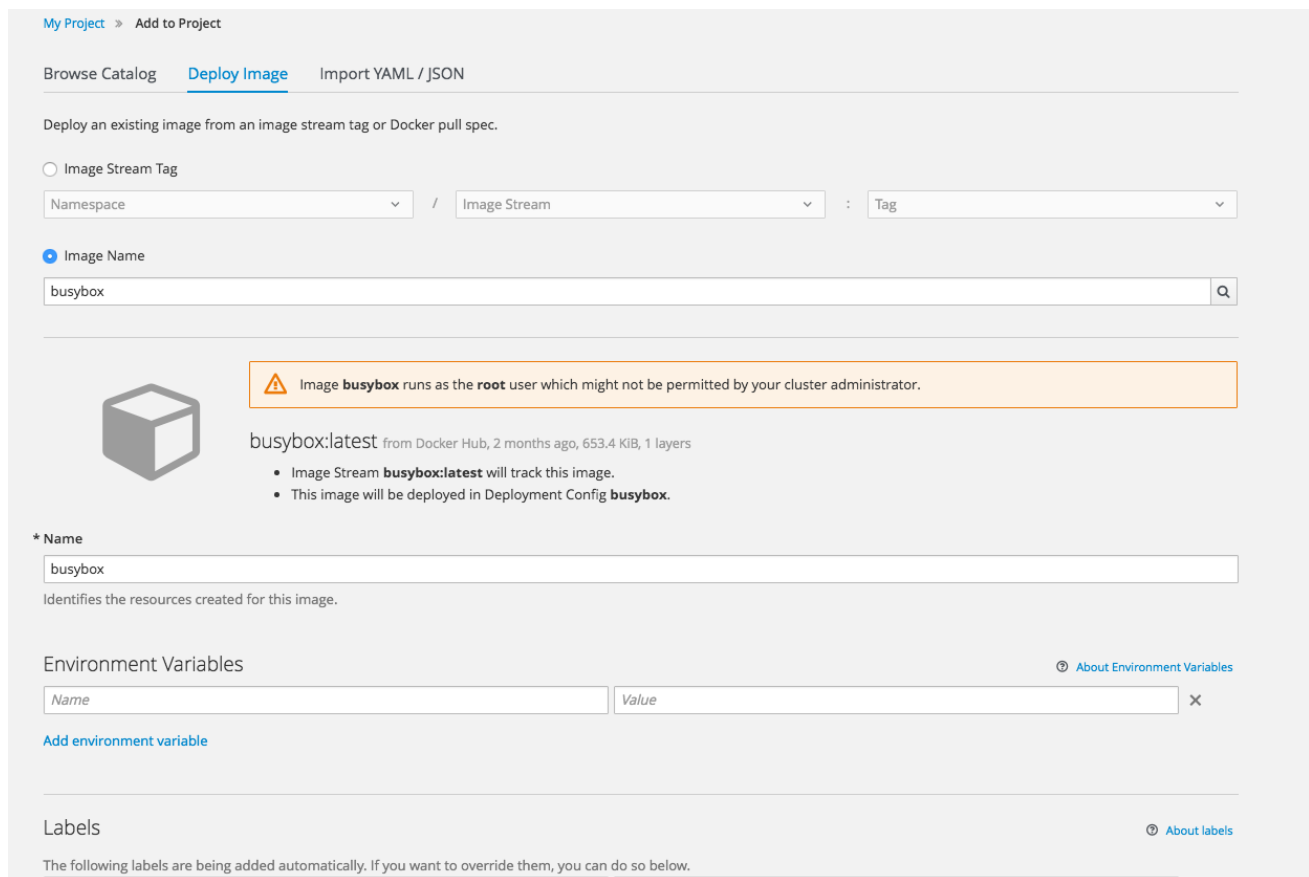


Modifying the route's back end services can be done in the new GUI editor, which also lets you change the route's target ports, path, and TLS settings.

#### 2.4.3.5. Deploy Image

The **Add to Project** page now a **Deploy Image** option. The behavior is similar to the `oc run` command, allowing you to pick any existing image or tag from an image stream, or to look for an image using a docker pull spec. After you have picked an image, it generates the service, deployment configuration, and an image stream if it is from a pull spec.

**Figure 2.12. Deploy Image**



My Project > Add to Project

Browse Catalog **Deploy Image** Import YAML / JSON


Deploy an existing image from an image stream tag or Docker pull spec.

Image Stream Tag

Namespace / Image Stream : Tag

Image Name

busybox

 **Image busybox runs as the root user which might not be permitted by your cluster administrator.**

busybox:latest from Docker Hub, 2 months ago, 653.4 KIB, 1 layers

- Image Stream **busybox:latest** will track this image.
- This image will be deployed in Deployment Config **busybox**.

\* Name

busybox

Identifies the resources created for this image.

Environment Variables [About Environment Variables](#)

Name	Value

[Add environment variable](#)

Labels [About labels](#)

The following labels are being added automatically. If you want to override them, you can do so below.

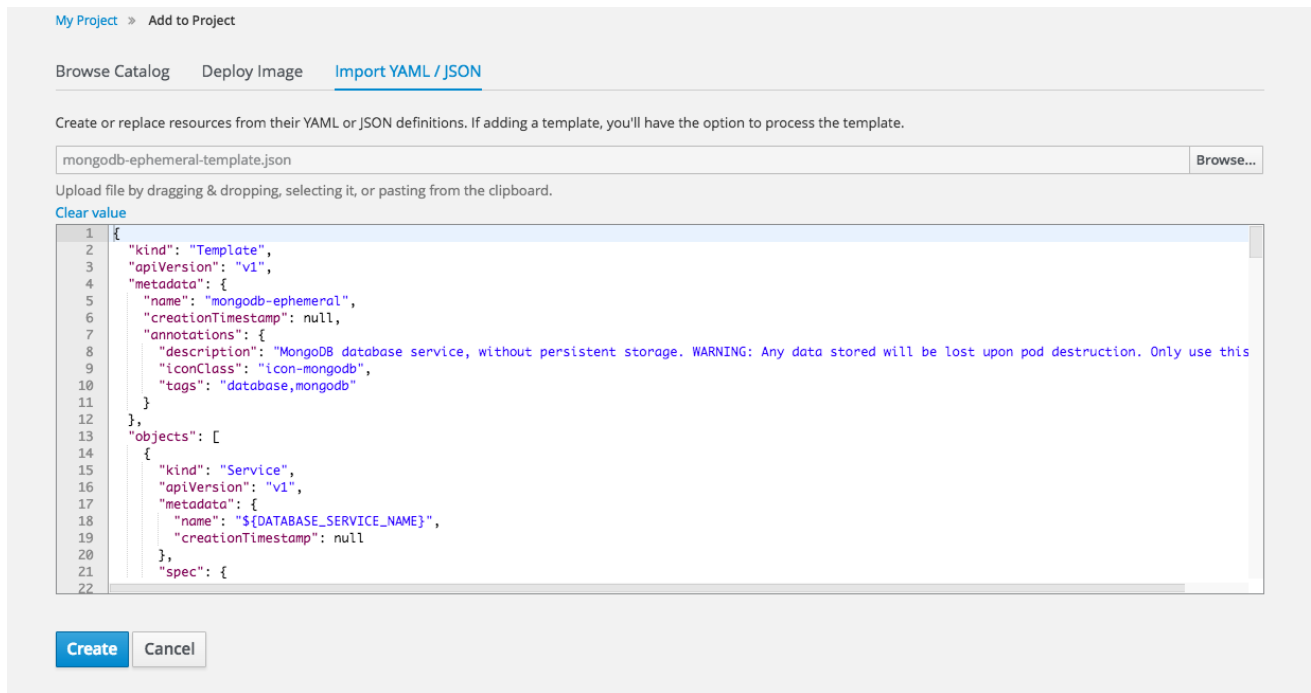
You can also take advantage of the new and improved key value editor for environment variables and labels.

### 2.4.3.6. Import YAML / JSON

The **Add to Project** page now has an **Import YAML / JSON** option, which behaves like the `oc create -f` command. You can paste, upload, or drag and drop your file, and even edit the YAML or JSON before submitting it. If your file contained a template resource, you can choose whether you want to create and/or process the template resource.



Figure 2.13. Import YAML / JSON

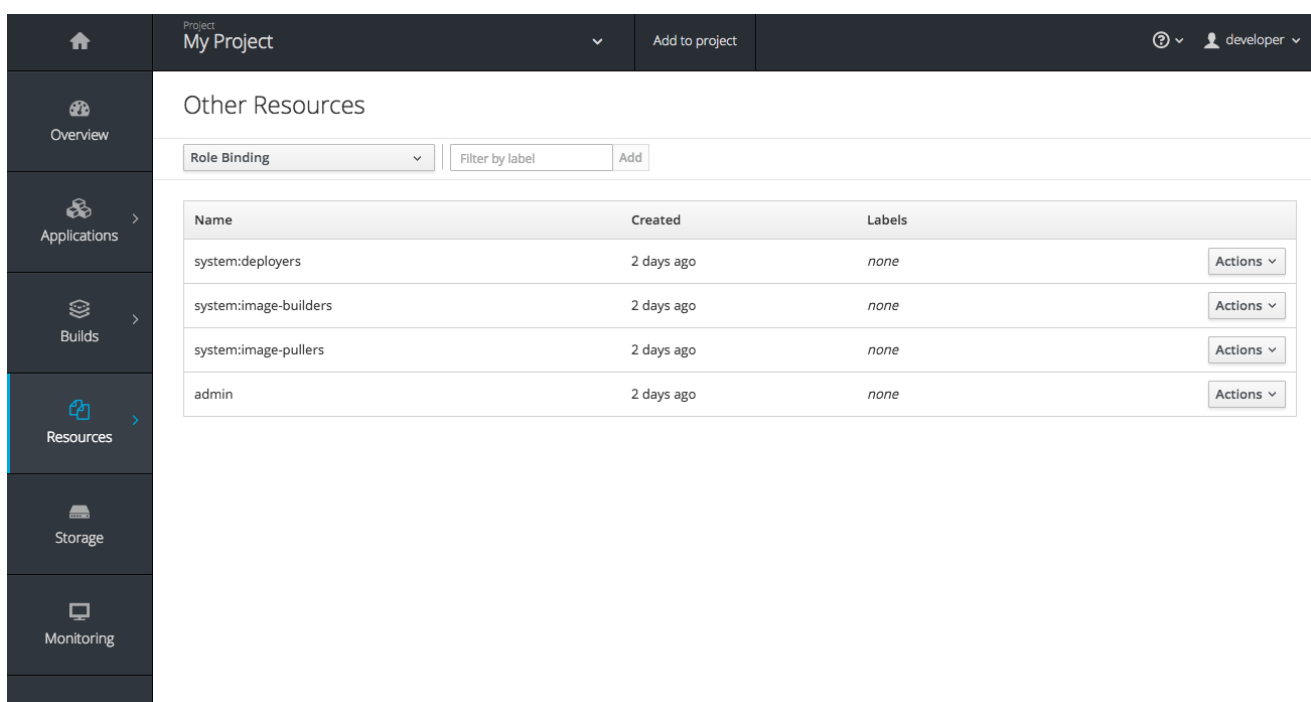


Processing a template goes to the existing experience for creating from a template, and now supports showing a message to the user on the next steps page. This message can be defined by the template author and can include generated parameters like passwords and other keys.

### 2.4.3.7. Other Resources

The **Other Resources** page gives you access to all the other content that exists in your project that do not have dedicated pages yet. You can select the type of resource you want to list and get actions to **Edit YAML** (similar to **oc edit**) and **Delete**. Due to a new feature that has been applied to the whole web console, only the resource types you have permission to list are shown, and only actions that you can actually perform.

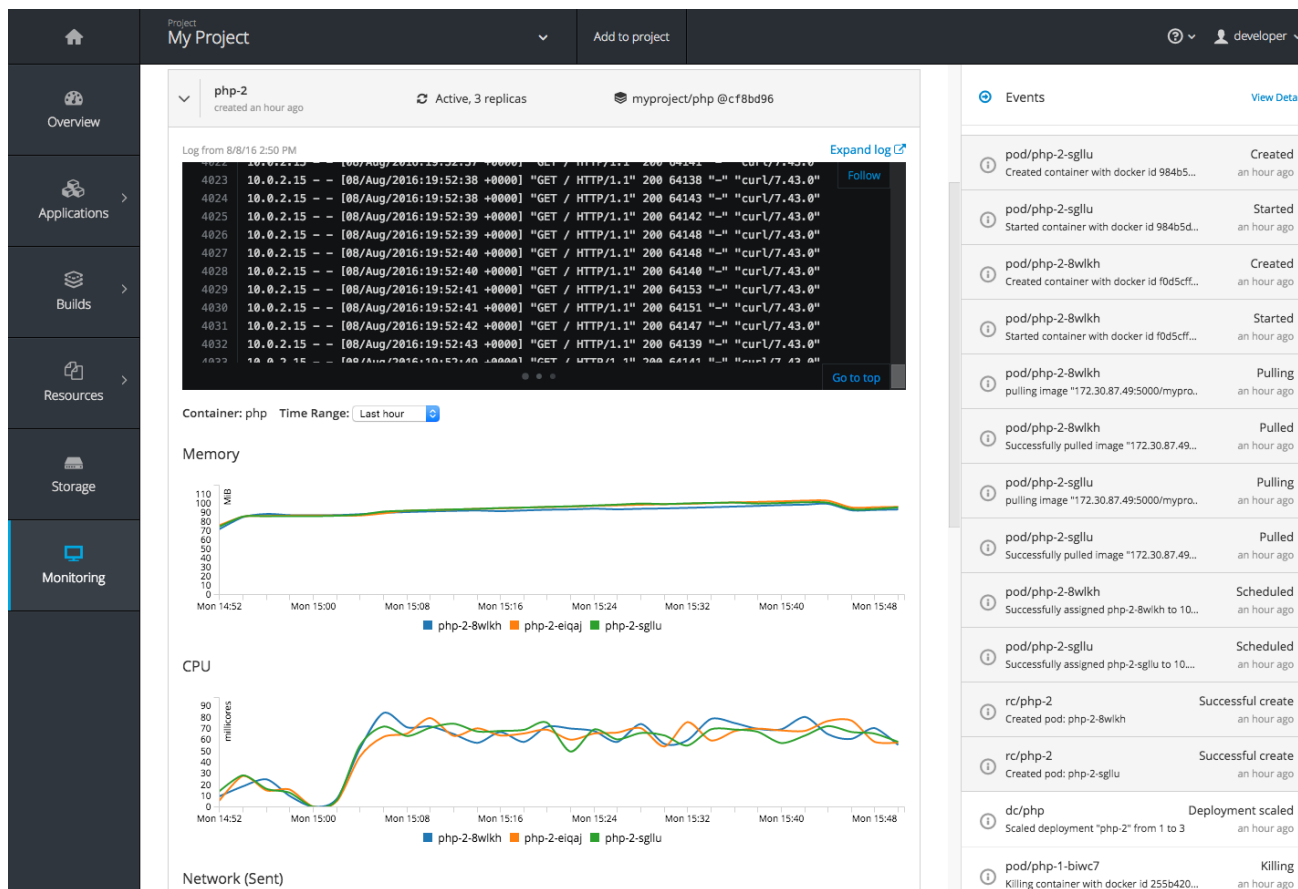
Figure 2.14. Other Resources



### 2.4.3.8. Monitoring

While the **Overview** provides some simple metrics and pod status, the new **Monitoring** page provides a deeper dive into the logs, metrics, and events happening in your project.

Figure 2.15. Monitoring



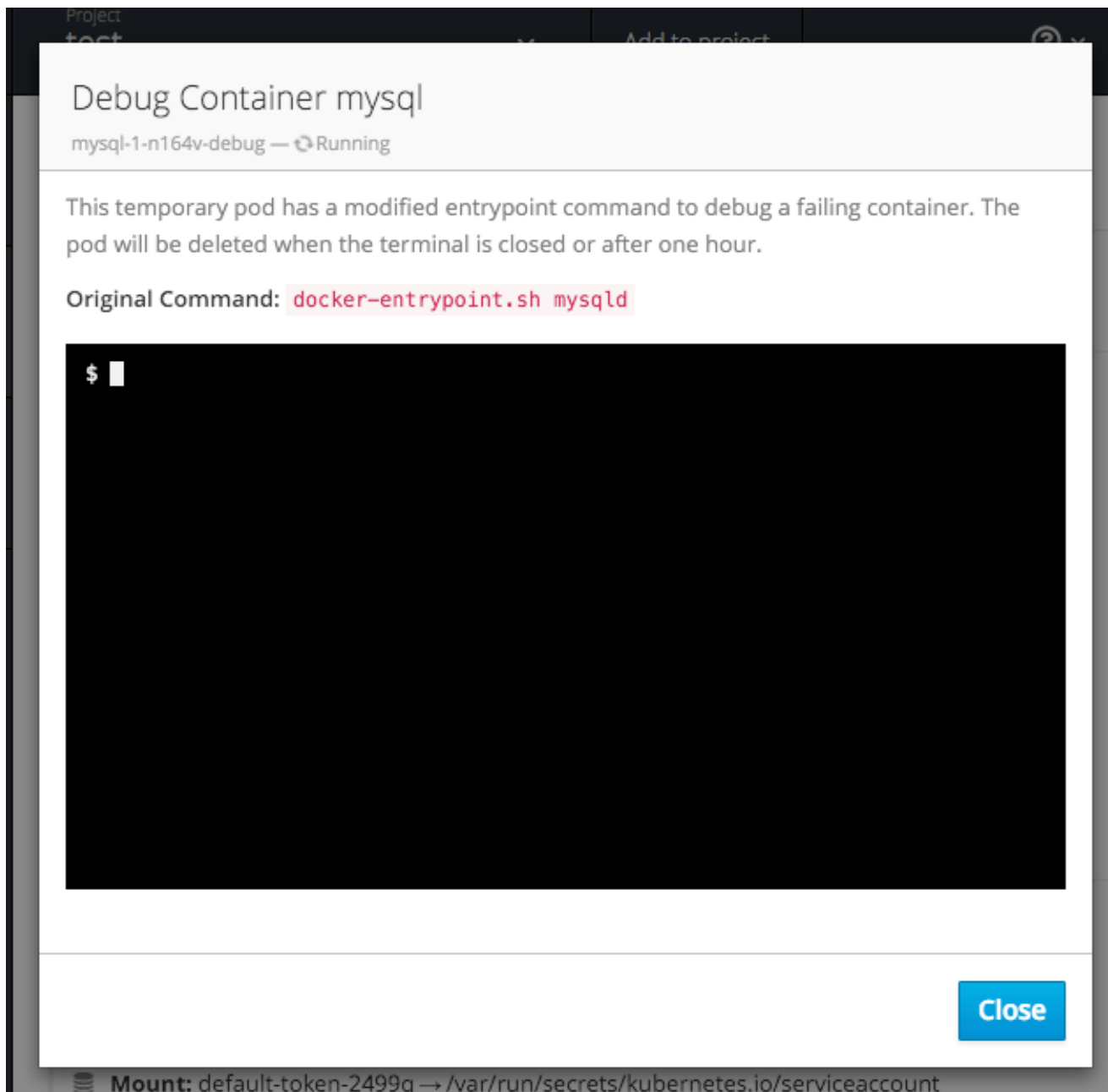
Metrics and logs both received some minor improvements including:

- Network sent and received metrics for deployments and pods
- Deployment metrics show a separate line for each pod
- Log viewer supports ANSI color codes and ANSI carriage returns (treated as new lines)
- Log viewer turns URLs into links

### 2.4.3.9. Debugging

When a pod's containers are not starting cleanly, a link is now shown on the pod details page to debug it in a terminal. This starts a pod with identical settings, but changes the container's entrypoint to `/bin/sh` instead, giving you access to the runtime environment of the container.

Figure 2.16. Debugging



A number of small improvements to the container terminal have also been added that create a smoother experience, including:

- Automatically focusing the keyboard input when the terminal connection is established
- Resizing based on the available space in the browser window
- Setting the **TERM** environment variable so common shell actions like **clear** behave the way you expect
- Better support for multi-container pods

Figure 2.17. Terminal

The screenshot shows the OpenShift web console interface. At the top, the project is identified as "My Project". The breadcrumb navigation shows "Pods > vim-go-2-qoyr9". The pod name "vim-go-2-qoyr9" is displayed, along with its creation time "7 minutes ago". Below the pod name, there are tabs for "app", "vim-go", "deployment", "vim-go-2", "deploymentconfig", and "vim-go". The "Terminal" tab is selected, showing a terminal window with the following JavaScript code:

```

7       return _.startsWith(path, prefix);
8     });
9   });
10  return {
11    restrict: 'E',
12    templateUrl: 'views/sidebar.html',
13    controller: function($scope) {
14      var path = $location.path().replace("/project/" + $scope.projectName, "");
15      $scope.activeSecondary;
16      $scope.navItems = Constants.PROJECT_NAVIGATION;
17      $scope.activePrimary = _.find($scope.navItems, function(primaryItem) {
18        if (itemMatchesPath(primaryItem, path) {
19          $scope.activeSecondary = null;
20          return true;
21        }
22      });
23      // Check if there is a secondary nav item that is active
24      return _.some(primaryItem.secondaryNavSections, function(secondarySection) {
25        var activeSecondary = _.find(secondarySection.items, function(secondaryItem) {
26          return itemMatchesPath(secondaryItem, path);
27        });
28        if (activeSecondary) {
29          $scope.activeSecondary = activeSecondary;
30          return true;
31        }
32        return false;
33      });
34    });
35  };
36  $scope.navURL = function(href) {
37    if (!href) {
38      return '';
39    }
40    if ($filter('isAbsoluteURL')(href)) {
41      return href;
42    }
43    return "project/" + $scope.projectName + href;
44  };
45  });
46  });
47  });
48  .directive('projectHeader', function($timeout, $location, $filter, DataService, projectOverviewURLFilter) {
49    // cache these to eliminate flicker
50    var projects = {};
51    var sortedProjects = [];
52  };
53  example.js [+]
```

### 2.4.3.10. Image Details

Before OpenShift Container Platform 3.3, there was no information in the web console about the images in your image streams, aside from the SHAs. This made it difficult to know the specifics of how your image was defined unless you used the CLI. Now, for any image stream tag you can see the metadata, configuration, and layers.

Figure 2.18. Image Stream Tag Details

Project: My Project

Image Streams > php > :latest

php:latest

To pull this image:

```
$ sudo docker pull registry/php:latest
```

Details Config Layers

<b>Name</b>	CentOS Base Image
<b>Author</b>	Unknown
<b>Built</b>	Invalid date
<b>Digest</b>	sha256:c52da6829ea3b1c8f4cdd3fa52f120001cc12f791342e064b4a67d845238db0
<b>Identifier</b>	
<b>Labels</b>	<pre>build-date=20160729 io.k8s.description=Platform for building and running PHP 5.6 applications io.k8s.display-name=myproject/php-10:a7e55d3d io.openshift.build.commit.author=Ben Parees &lt;bparees@users.noreply.github.com&gt; io.openshift.build.commit.date=Mon Aug 22 14:44:49 2016 -0400 io.openshift.build.commit.id=701d706b7f2b50ee972d0bf76990042f6c0cda5c io.openshift.build.commit.message=Merge pull request #42 from bparees/recreate io.openshift.build.commit.ref=master io.openshift.build.image=centos/php-56-centos7@sha256:a2251404ff06b636526aa042cb0878fb7411250a7f4b1c0af002cc284e226446 io.openshift.build.source-location=https://github.com/openshift/cakephp-ex.git io.openshift.builder-base-version=fa7ba54 io.openshift.builder-version=6fdd73c47e5e00927f0da67343e41adb870d78e5 io.openshift.expose-services=8080:http io.openshift.s2i.scripts-url=image:///usr/libexec/s2i io.openshift.tags=builder,php,php56,rh-php56 io.s2i.scripts-url=image:///usr/libexec/s2i license=GPLv2 vendor=CentOS</pre>
<b>Annotations</b>	openshift.io/image.managed: true
<b>Docker Version</b>	1.10.3

Figure 2.19. Image Stream Tag Configuration

Project: Pipeline Example

Image Streams > origin-ruby-sample > :latest

origin-ruby-sample:latest

To pull this image:

```
$ sudo docker pull registry/origin-ruby-sample:latest
```

Details Config Layers

<b>Command:</b>	\$ container-entrypoint /usr/libexec/s2i/run		
<b>Run as</b>	1001	<b>Environment</b>	RACK_ENV=production
<b>Directory</b>	/opt/app-root/src		OPENSIFT_BUILD_NAME=ruby-sample-build-3
<b>Architecture</b>	amd64		OPENSIFT_BUILD_NAMESPACE=pipeline
			OPENSIFT_BUILD_SOURCE=https://github.com/openshift/ruby-hello-world.git
			EXAMPLE=sample-app
			PATH=/opt/app-root/src/bin:/opt/app-root/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin
			STI_SCRIPTS_URL=image:///usr/libexec/s2i
			STI_SCRIPTS_PATH=/usr/libexec/s2i
			HOME=/opt/app-root/src
			BASH_ENV=/opt/app-root/etc/scl_enable
			ENV=/opt/app-root/etc/scl_enable
			PROMPT_COMMAND=. /opt/app-root/etc/scl_enable
			RUBY_VERSION=2.2
<b>Ports</b>	8080/tcp		

## 2.4.4. Networking

This release adds the following improvements to networking components.

### 2.4.4.1. Controllable Source IP

Platform administrators can now identify a node in the cluster and allocate a number of static IP addresses to the node at the host level. If a developer needs an unchanging source IP for their application service, they can request access to one during the process they use to ask for firewall access. Platform administrators can then deploy an egress router from the developer's project, leveraging a `nodeSelector` in the deployment configuration to ensure the pod lands on the host with the pre-allocated static IP address.

The egress pod's deployment declares one of the source IPs, the destination IP of the protected service, and a gateway IP to reach the destination. After the pod is deployed, the platform administrator can create a service to access the egress router pod. They then add that source IP to the corporate firewall and close out the ticket. The developer then has access information to the egress router service that was created in their project (e.g., `service.project.cluster.domainname.com`).

When the developer would like to reach the external, firewalled service, they can call out to the egress router pod's service (e.g., `service.project.cluster.domainname.com`) in their application (e.g., the JDBC connection information) rather than the actual protected service url.

See [Controlling Egress Traffic](#) for more details.

#### 2.4.4.2. Router Sharding

OpenShift Container Platform offers a [multi-tenant](#), docker-compliant platform. Thousands of tenants can be placed on the platform, some of which may be subsidiary corporations or have drastically different affiliations. With such diversity, often times business rules and regulatory requirements will dictate that tenants not flow through the same routing tier.

To solve this issue, OpenShift Container Platform 3.3 introduces [router sharding](#). With router sharding, a platform administrator can [group specific routes or namespaces into shards](#) and then assign those shards to routers that may be up and running on the platform or be external to the platform. This allows tenants to have separation of egress traffic at the routing tiers.

#### 2.4.4.3. Non-Standard Ports

OpenShift Container Platform has always been able to support non-standard TCP ports via SNI routing with SSL. As the internet of things (IoT) has exploded, so to has the need to speak to dumb devices or aggregation points without SNI routing. At the same time, with more and more people running data sources (such as databases) on OpenShift Container Platform, many more people want to expose ports other than 80 or 433 for their applications so that people outside of the platform can leverage their service.

Previously, the solution for this in Kubernetes was to leverage NodePorts or External IPs. The problem with NodePorts is that only one developer can have the port on all the nodes in the cluster. The problem with External IPs is that duplications can be common if the administrator is not carefully assigning them out.

OpenShift Container Platform 3.3 solves this problem through [the clever use of edge routers](#). Platform administrators can either select one or more of the nodes (more than one for high availability) in the cluster to become edge routers or they can just run additional pods on the HAProxy nodes.

For example, a platform administrator can run additional pods that are ipfailover pods. A pool of available Ingress IPs are specified that are routable to the nodes in the cluster and resolvable externally via the corporate DNS. This pool of IP addresses are served out to developers who want to use a port other than 80 and 433. In these use cases, there are services outside of the cluster trying to connect to services inside the cluster that are running on ports other than 80 or 433. This means they are coming into the

cluster (ingress) as opposed to leaving the cluster (egress). By resolving through the edge routers, the cluster can ensure each developers gets their desired port by pairing it with a Ingress IP from the available pool rather than giving them a random port.

In order to trigger this allocation of an Ingress IP, the developer declares a **LoadBalancer** type in their service definition for their application. Afterwards, they can use the `oc get <service_name>` command to see what Ingress IP was assigned to them. See [Getting Traffic into the Cluster](#) for details.

#### 2.4.4.4. A/B Service Annotation

OpenShift Container Platform 3.3 adds service lists to routes, making it easier to perform A/B testing. Each route can now have multiple services assigned to it, and those services can come from different applications or pods.

New automation enables HAProxy to be able to read weight annotations on the route for the services. This enables developers to declare traffic flow (for example, 70% to application A and 30% to application B) using the CLI or web console.



#### NOTE

See [New Features and Enhancements: Web Console](#) for more details on the new A/B routing user interface.

See [Load Balancing for A/B Testing](#) for more details.

### 2.4.5. Security

This release adds the following improvements to cluster security.

#### 2.4.5.1. SCC Profiles for seccomp

The **seccomp** feature in Red Hat Enterprise Linux (RHEL) has been enabled for docker 1.10 or higher. This feature allows containers to define interactions with the kernel using **syscall** filtering. This reduces the risk of a malicious container exploiting a kernel vulnerability, thereby reducing the guest attack surface.

OpenShift Container Platform adds the ability to create **seccomp** policies with security context constraints (SCCs). This allows platform administrators to set SCC policies on developers that imposes a filter on their containers for Linux-level system calls.

See the [Authorization](#) concept for more details.

#### 2.4.5.2. Kerberos Support in oc client for Linux

The **oc** client on Linux can now recognize and handle the **kinit** process of generating a Kerberos ticket during developer interactions with the CLI. For example:

```
$ kinit <user>@<domain>
$ oc login <openshift_master>
```

#### 2.4.5.3. Certificate Maintenance

OpenShift Container Platform leverages TLS encryption and token-based authentication between its framework components. In order to accelerate and ease the installation of the product, certificates are self-signed during automated installation.

OpenShift Container Platform 3.3 adds the ability to update and change those certificates that govern the communication between framework components. This allows platform administrators to more easily maintain the life cycles of their OpenShift Container Platform installations.

See [Redeploying Certificates](#) for more details.

## 2.4.6. Cluster Longevity

This release adds the following improvements to cluster longevity.

### 2.4.6.1. Pod Eviction

OpenShift Container Platform 3.3 allows platform administrators more control over what happens over the lifecycle of the workload on the cluster after the process (container) is started. By leveraging limits and request setting at deployment time, the cluster can determine automatically how the developer wants their workload handled in terms of resources. [Three positions can be taken](#):

- If the developer declares no resource requirements (best effort), slack resources are offered on the cluster. More importantly, workloads are re-deployed first should an individual node become exhausted.
- If the developer sets minimum resource requirements but does not ask for a very specific range of consumption (burstable), their minimum is set while also giving them an ability to consume slack resources should any exist. This workload is considered more important than best effort in terms of re-deployment during a node eviction.
- If a developer sets the minimum and maximum resource requirements (guaranteed), a node with those resources is found and the workload is set as most important on the node. These workloads remain as the last survivor on a node should it go into a memory starvation situation.

The decision to [evict is a configurable setting](#). Platform administrators can turn on the ability to hand a pod (container) back to the scheduler for re-deployment on a different node should out of memory errors start to occur.

### 2.4.6.2. Scale

1000 nodes per cluster at 250 pods per node (with a recommendation of 10 pods per hyper-threaded core) are now supported. See [Sizing Considerations](#) for more details.

### 2.4.6.3. Idling and Unidling

OpenShift Container Platform 3.3 adds an API to idle an application's pods (containers). This allows for monitoring solutions to call the API when a threshold to a metric of interest is crossed. At the routing tier, the HAProxy holds the declared route URL that is connected to the service open and the pods are shut down. Should someone hit this application URL, the pods are re-launched on available resources in the cluster and connected to the existing route.

### 2.4.6.4. Storage Labels

OpenShift Container Platform already included the ability to offer remote persistence block and file based storage, and this release adds the ability for developers to select a storage provider on the cluster in a



more granular manner using storage labels. Storage labels help developers call out to a specific provider in a simple manner by adding a label request to their persistent volume claim (PVC).

See [Binding Persistent Volumes by Labels](#) for example usage.

## 2.4.7. Framework Services

OpenShift Container Platform provides resource usage metrics and log access to developers based on the Hawkular and Elasticsearch open source projects. This release adds the following improvements to these components.

### 2.4.7.1. Logging Enhancements

A new [log curator](#) utility helps platform administrators deal with the storage requirements of storing tenant logs over time.

Integration with existing ELK stacks you might already own or be invested in has also been enhanced by allowing logs to more easily be sent to multiple locations.

### 2.4.7.2. Metrics Installation Enhancement

This release adds network usage attributes to the core metrics tracked for tenants. Metrics deployment is also now a core installation feature instead of a post-installation activity. The OpenShift Container Platform installer now guides you through the Ansible playbooks required to successfully deploy metrics, thus driving more usage of the feature in the user interface and Red Hat CloudForms.

## 2.5. NOTABLE TECHNICAL CHANGES

OpenShift Container Platform 3.3 introduces the following notable technical changes.

### Updated Infrastructure Components

- Kubernetes has been updated to v1.3.0+52492b4.
- etcd has been updated to 2.3.0+git.
- OpenShift Container Platform 3.3 requires Docker 1.10.

### Routing Data Structure Changes

The underlying data structure that a router template can use has changed in OpenShift Container Platform 3.3. [Additional steps](#) may be needed for an upgrade from 3.2 to 3.3 if you previously customized your HAProxy routing template.

#### *Short Summary of Changes*

In the older model, the top level had one map of all services. To get to routes in the system, all services had to be iterated over to get to the routes that each service holds. In the new model, the top level contains two maps: one for all the routes and one for all the services. You can now get to any of them without repeated iteration.

#### *Understanding the New Model*

The new data structure defining the routing back ends consists of two structures representing services and routes and one top-level structure that contains a map to both.

- **ServiceUnit** ← → **Service**
- **ServiceAliasConfig** ← → **Route**

The top-level router template has two maps:

```
State          map[string]ServiceAliasConfig
ServiceUnits   map[string]ServiceUnit
```

In OpenShift Container Platform 3.3, a route can have many services and any service can be part of many routes. The **ServiceAliasConfig(Route)** holds a map of **ServiceUnitNames(Service)** with their corresponding weights. To get to the actual **service/ServiceUnit**, you must look up the top-level map **ServiceUnits**:

```
type ServiceAliasConfig {
  ..
  ..
  ServiceUnitNames map[string]int32
}
```

To quickly go through all the routes as an example:

1. Iterate over the **template.State** map, which gives all routes represented by **ServiceAliasConfig**.
2. Go over all services of a route along with their weights.
3. With each service name, look up the actual service from the **template.ServiceUnits** map.
4. Go over endpoints of the service with the **Endpoints** field in the **ServiceUnit** structure and use those endpoints with the associated weight for the service.

#### Example 2.1. Example Code

```
# get the routes/ServiceAliasConfigs from .State
{{ range $routeId, $route := .State }}
  # get the names of all services that this route has, with the
  corresponding weights
  {{ range $serviceName, $weight := $route.ServiceUnitNames }}
    # now look up the top level structure .ServiceUnits to get the
    actual service object
    {{ with $service := index $.ServiceUnits $serviceName }}
      # get endpoints from the service object
      {{ range $idx, $endpoint := endpointsForAlias $route
      $service }}
        # print the endpoint
        server {{$endpoint.IdHash}} {{$endpoint.IP}}:{{$endpoint.Port}}...
```

#### Comparing with the Older Model

To contrast with the older model, previously a service could be part of many routes, so there were two basic structures:

- **ServiceAliasConfig** corresponded to a **Route**.

- **ServiceUnit** corresponded to a **Service**, but also held how many **Routes** pointed to it.

**ServiceUnit** had one special field that contained all the **ServiceAliasConfigs** (routes) that it was part of:

```
type ServiceUnit {
  ..
  ..
  ServiceAliasConfigs map[string]ServiceAliasConfig
}
```

The top level template had a map of all services in the system. To iterate to routes, you previously had to iterate over services first to get the routes that it was part of. For example:

1. Iterate over all **ServiceUnits** (services)
2. Iterate over all **ServiceAliasConfigs** (routes) that this Service has.
3. Get the route information (header, TLS, etc.) and use the **Endpoints** field in the **ServiceUnit** to get to the actual back ends.

#### Example 2.2. Example Code

```
{{ range $id, $serviceUnit := .State }}
  {{ range $routeId, $route := $serviceUnit.ServiceAliasConfigs }}
    {{ range $idx, $endpoint := endpointsForAlias $route
      $serviceUnit }}
      server {{$endpoint.IdHash}} {{$endpoint.IP}}:{{$endpoint.Port}}
```

The older model could not accommodate the idea that a route could contain multiple services.

#### *Upgrade Requirements for Customized HAProxy Routing Templates*

If you are upgrading to OpenShift Container Platform 3.3 but you never changed the default HAProxy routing template that came with the image, then no action is required. Ensure that the new router image is used so that you can use the latest features for the release. If you ever need to change the template, consult this documentation.

If you previously customized your HAProxy routing template, then, depending on the changes, the following may be required:

- Re-apply the changes on the newer template. Or,
- Rewrite your existing template using the newer model:
  - Iterating over **.State** now gives **ServiceAliasConfigs** and not the **ServiceUnits**.
  - Each **ServiceAliasConfig** now has multiple **ServiceUnits** in it stored as keys of a map, where the value of each key is the weight associated with the service.
  - To get the actual service object, index over another top level object called **ServiceUnits**.
  - You can no longer get the list of routes that a service serves; this information was not found to be useful. If you use this information for any reason, you must construct your own map by iterating over all routes that contain a particular service.

It is recommended that the new template is taken as a base and modifications are re-applied on it. Then, rebuild the router image. The same applies if you use a **configMap** to supply the template to the router: you must use the new image or rebuild your image either way because the OpenShift Container Platform executable inside the image needs an upgrade, too.

### Manually-Created Endpoints Inside ClusterNetworkCIDR

In OpenShift Enterprise 3.2 and earlier, if the cluster was using the **redhat/openshift-ovs-multitenant** network plug-in, and a service endpoint was manually created pointing to a pod or service owned by another tenant, then that endpoint would be ignored. In OpenShift Container Platform 3.3, it is no longer possible for regular users to create such an endpoint ([openshift/origin#9383](#)). As a result, the plug-in now no longer filters them out ([openshift/origin#9982](#)).

However, previously-created illegal endpoints might still exist; if so, the old, pre-upgrade logs will show warnings like the following, indicating the illegal endpoints object:

```
Service 'foo' in namespace 'bob' has an Endpoint inside the service
network (172.30.99.99)
Service 'foo' in namespace 'bob' has an Endpoint pointing to non-existent
pod (10.130.0.8)
Service 'foo' in namespace 'bob' has an Endpoint pointing to pod
10.130.0.4 in namespace 'alice'
```

These log messages are the simplest way to find such illegal endpoints, but if you no longer have the pre-upgrade logs, you can try commands like the following to search for them.

To find endpoints pointing to the default **ServiceNetworkCIDR** (172.30.0.0/16):

```
$ oc get endpoints --all-namespaces --template \
  '{{ range .items }}{{ .metadata.namespace }}:{{ .metadata.name }} \
  {{ range .subsets }}{{ range .addresses }}{{ .ip }} \
  {{ end }}{{ end }}{{ "\n" }}{{ end }}' | awk '/ 172\.30\.\/ { print $1
}'
```

To find endpoints pointing to the default **ClusterNetworkCIDR** (10.128.0.0/14):

```
$ for ep in $(oc get services --all-namespaces --template \
  '{{ range .items}}{{ range .spec.selector }}{{ else }}{{
  .metadata.namespace }}:{{ .metadata.name }} \
  {{ end }}{{ end }}'); do \
  oc get endpoints --namespace $(echo $ep | sed -e 's/.*://') $(echo
  $ep | sed -e 's/.*://') \
  --template '{{ .metadata.namespace }}:{{ .metadata.name }} {{
  range .subsets }}{{ range \
  .addresses }}{{ .ip }} {{ end }}{{ end }}{{ "\n" }}' | awk '/ \
  10\.([12][8-9]|1[3-9][0-9]|2[0-5][0-9])\.\/ { print $1 }' \
done
```

### Pull Access When Tagging Image Streams

When tagging images across projects, for example:

```
$ oc tag <project_1>/<image_stream_a>:<tag_a>
<project_b>/<image_stream_b>:<tag_b>
```

a user must have pull permission on the source image stream ([openshift/origin#10109](#)). This means they must get access on the **imagestreams/layers** resource in the source project. The **admin**, **edit**, and **system:image-puller** roles all grant this permission.

## Changes to DNS Records Returned by SRV Requests

OpenShift Container Platform 3.3 has altered the DNS records returned by SRV requests for services to be compatible with Kubernetes 1.3 to support **PetSets** objects ([openshift/origin#9972](#)). The primary change is that SRV records for a name no longer enumerate the list of all available ports; instead, if you want to find a port named **http** over protocol **tcp**, you must specifically ask for that SRV record.

1. The SRV records returned for service names (**<service>.<namespace>.svc.cluster.local**) have changed. Previously, OpenShift Container Platform returned one SRV record per service port, but to be compatible with Kubernetes 1.3, SRV records are now returned representing endpoints (**<endpoint>.<service>.<namespace>.svc.cluster.local**) without port info (a port of **0**).

A clustered service (type **ClusterIP**) will have one record pointing to a generated name (e.g., **340982409.<service>.<namespace>.svc.cluster.local**) and an associated A record pointing to the cluster IP.

A headless service (with **clusterIP=None**) returns one record per address field in the **Endpoints** record (typically one per pod). The endpoint name is either the **hostname** field in the endpoint (read from an annotation on the pod) or a hash of the endpoint address, and has an associated A record pointing to the address matching that name.

2. The SRV records returned for an endpoint name (**<endpoint>.<service>.<namespace>.svc.cluster.local**) have changed: a single SRV record is returned if the endpoint exists (the name matches the generated endpoint name described above) or no record if the endpoint does not exist.
3. The SRV records for a given port (**<portname>.<protocol>.<service>.<namespace>.svc.cluster.local**) behave as they did before, returning port info.

## 2.6. BUG FIXES

This release fixes bugs for the following components:

### Authentication

- Multiple API servers starting simultaneously with an empty etcd datastore would race to populate the default system policy. A partially created policy could result, leaving a new cluster with a policy that would forbid system components from making some API calls. This bug fix updates the policy APIs to perform the same **resourceVersion** checking as other APIs, and fault-tolerant logic was added to the initial policy population step. As a result, new clusters populate default policy as expected. ([BZ#1359900](#))

### Builds

- The transition between serial and parallel builds was not handled correctly. If parallel builds were queued after a running serial build, the first parallel build would also run serially, instead of running all the parallel builds in parallel when the serial build completed. After this bug fix, when the first parallel build is run, any other parallel builds in the queue are also run. As a result, all parallel builds in the queue start simultaneously when the last serial build finishes. ([BZ#1357786](#))

- The S2I builder image value was not getting properly set on an **s2i rebuild** invocation, causing these invocations to fail. This bug fix changes the code so that it inspects the existing image on rebuild and populates the configuration from its labels instead of the builder's labels. The builder image is still inspected on typical **s2i build** invocations. As a result, both **s2i build** and **s2i rebuild** now work as expected. ([BZ#1366475](#))
- Updates to a build configuration via the replace mechanism would previously reset the build sequence count to zero if no value was specified in the update. Builds would fail to start if the reset sequence number caused collisions with existing builds that used those the sequence number previously. After this bug fix, the sequence number is no longer reset during updates to the build configuration. As a result, build configurations can be updated and the existing sequence number is preserved, so new builds do not collide with previously used sequence numbers. ([BZ#1357791](#))

### Command Line Interface

- An improper argument parsing rejected valid values caused parameter values containing equal signs to be incorrectly rejected. This bug fix changes parsing to tolerate values containing equal signs. As a result, parameter values containing equal signs are tolerated. ([BZ#1375275](#))

### Images

- This enhancement updates the Perl S2I builder image to support proxy configurations. Previously, the image could not access remote resources if the customer network required a proxy be used. The Perl image now respects the **HTTP\_PROXY** environment variable for configuring the proxy to use when requesting remote resources during the build process. ([BZ#1348945](#))
- Previously, the timeout for liveness probe for the Jenkins readiness check was too short. This caused Jenkins pods to fail to report as ready then get restarted. This bug fix increases the timeout for the readiness probe, and Jenkins pods now have sufficient time to start before the readiness probe fails. ([BZ#1368967](#))

### Image Registry

- The S3 communication library was not efficient enough to support high loads of data. This caused some pushes to the registry to take relatively long. This bug fix updates both the Docker Distribution code along with the S3 driver. As a result, docker push operations experience improved stability and performance. ([BZ#1314381](#))
- A bug in an older registry version prevented it from working with a Swift storage back-end while having the content-offload feature turned off, causing the registry to be unusable in these conditions. This bug fix updates the registry version, which has reworked storage drivers. As a result, the registry is now usable in these conditions. ([BZ#1348031](#))
- When pruning images, a user was previously presented with too many log details by default. This bug fix hides some debug information behind increased **--loglevel** settings. As a result, logs presented to user should be more readable. ([BZ#1341527](#))

### Installer

- Previously, the installer did not correctly format the registry 2.4 configuration file when using S3 storage. This bug fix corrects this formatting issue and the installer now correctly provisions S3-based registry components when configured to do so. ([BZ#1356823](#))
- Previously, installation would fail with an unrelated error message when **openshift\_hosted\_registry\_storage\_kind=nfs** was specified in the inventory but no

NFS hosts were configured via `openshift_hosted_registry_storage_host` or the `nfs` host group. Playbooks now output an error message indicating that no storage hosts have been configured. ([BZ#1357984](#))

- Previously, containerized nodes mounted `/sys` read-only, which prevented the node from mounting Ceph volumes. This mount for the containerized node has been updated to be read-write, allowing the node to mount Ceph volumes properly. ([BZ#1367937](#))
- The quick installer previously did not verify file system paths when read from a configuration file. This caused the quick installer to attempt to read a file which did not exist, throw a stack trace, and abort the installation. This bug fix ensures that the file system path is now verified to exist when read from a configuration file, and as a result the quick installer no longer crashes. ([BZ#1368296](#))

## Kubernetes

- This enhancement adds volume affinity to OpenShift Container Platform (OCP). Cloud providers typically use multiple zones/regions for their virtual machines and storage offerings. A virtual machine in one zone/region can only mount storage from the same zone/region in which it resides. OCP pods that use cloud storage must be scheduled onto virtual machines in the same zone/region for their associated storage; otherwise, the pods will fail to run. With this enhancement, pods are now scheduled to the same zone/region as their associated storage. Note that if you are not using the default scheduler configuration, you must ensure that the `NoVolumeZoneConflict` scheduler predicate is enabled in your scheduler configuration file in order for volume affinity to function correctly. ([BZ#1356010](#))
- The trigger controller used for handling triggers for deployments was not handling `ImageChangeTriggers` correctly from different namespaces, resulting in hot looping between deployments. This bug fix addresses the issue and it no longer occurs. ([BZ#1366936](#))
- The Horizontal Pod Autoscaler scales based on CPU usages as a percentage of the requested CPU for a pod. It is possible that the desired percentage be over 100 (if the user wants to scale only when the CPU usage of a pod is higher than the amount requested for the pod, but below the limit for the pod). Previously, the CLI would prevent the user from setting such values. Now, it allows setting a target CPU percentage of over 100. ([BZ#1336692](#))
- Jobs were an experimental feature in OpenShift Enterprise 3.1, and templates did not work with jobs. This bug fix stabilizes the job feature. Jobs have been migrated to stable API allowing full support of all the necessary features, including templates. ([BZ#1319929](#))
- Diagnostics previously reported an error when the registry was not backed by a persistent storage volume on the pod, without considering alternative methods of storage. If the registry had been reconfigured to use S3 as storage, for example, diagnostics reported this error. This bug fix updates the diagnostic check to see if registry configuration has been customized and does not report an error if so. As a result, it is assumed the cluster administrator that does the configuration knows what they are doing, and false alerts on S3-configured registries are no longer reported. ([BZ#1359771](#))

## Logging

- This enhancement adds auto-tuning for Elasticsearch memory heap usage based on container limit. Elasticsearch recommends hard limits for proper usage and these limits may significantly exceed what is available to the container. Elasticsearch should limit itself from the onset. With this enhancement, the container runsript evaluates the available memory and sets the minimum and maximum heap size. ([BZ#1370115](#))

- When image streams are created, only a subset of the available tags are imported, and this often excluded the desired tag. If the desired tag is not imported, then the corresponding component never deploys. To work around this issue, import each tag manually:

```
$ oc import-image <name>:<version> --from <prefix><name>:<tag>
```

This bug is fixed in OpenShift Container Platform 3.3 by not relying on image streams and deployment configuration triggers for deployment. As a result, deployment occurs as expected. ([BZ#1338965](#))

- When a project was deleted, the plug-in for Fluentd was not properly handling the fetching of metadata and would exit, restarting the Fluentd pod. This bug fix updates the **kubeclient** and **rest-client** gems for Fluentd. As a result, Fluentd is able to properly handle cases where the project was deleted for logs it is processing. ([BZ#1365422](#))
- When reading in rolled over log messages into Fluentd, if the rolled over file name was not in a specific format, Fluentd would fail while processing the date for that record. This was to adjust for a gap where logs from the previous year would be interpreted as logs that take place in the future since there was not a year field on the log records. This could cause a loss of log records. With this bug fix, in addition to container logs, Fluentd now only reads in records from `/var/log/messages` instead of `/var/log/messages*`. As a result, Fluentd no longer reads in log records from rolled over files. ([BZ#1347871](#))
- The **OpenShift-Elasticsearch-Plugin** did not remove the `.a11` Kibana mapping for users that were **cluster-admin** but then had the role reverted. If a user was no longer **cluster-admin**, they could still be able to view the `.a11` Kibana mapping. They would not be able to see the logs for projects they did not have access to, but they would still incorrectly see the mapping. This bug fix updates the **OpenShift-Elasticsearch-Plugin** to remove the `.a11` Kibana mapping to users that are not **cluster-admin**. As a result, non-**cluster-admin** users are not able to see the `.a11` mapping if they are no longer **cluster-admin**. ([BZ#1372277](#))

## Web Console

- The builder images in the web console were not ordered by semantic version. In some cases, a newer technology version could be hidden under a **See All** link because it had a lower sort order. With this bug fix, the builders are now properly ordered by their semantic version. As a result, more recent version are sorted to the top and are no longer hidden. ([BZ#1325069](#))
- When configuring a build to use a GitHub git source and setting a context directory or reference, the source repository appeared as the full link to the context directory or reference in GitHub, which is a long unreadable URL. This bug fix updates the web console to not show the full link. As a result, the visual representation of the source repository is only the source repository, and the target of the link includes the context directory and reference. ([BZ#1364950](#))
- The web console prevented users from deleting replication controllers with active pods to avoid orphaning them. The **Delete** menu item was disabled for replication controllers when they have active replicas, but it was not obvious why. The web console now provides help text explaining as well as example commands for deleting from the CLI (which will scale the replication controller down automatically). ([BZ#1365582](#))
- This enhancement adds a cancel deployment link to the **Overview** page. The cancel deployment action could be difficult to discover on the deployment details page, so deployments can now be canceled directly from the **Overview**. ([BZ#1365666](#))
- The web console did not set a **TERM** environment variable when the terminal execs into a pod using the `/bin/sh` command. This caused certain commands like **clear**, **less**, and **top** to not



behave as expected. This bug fix sets the environment variable **TERM=xterm** when **/bin/sh** is used to connect to the pod. As a result, commands like **clear**, **less**, and **top** now behave properly. ([BZ#1367337](#))

- In some cases, a warning could be resolved while the tooltip describing the warning was open. When this happened, the tooltip could not be dismissed. This bug fix updates the web console to now properly close the tooltip when the warning disappears, and as a result the open tooltip will disappear with the warning icon. ([BZ#1347520](#))
- On the pod metrics tab in the web console, the available CPU and memory is shown for pods that have resource limits. If a pod was using more CPU or memory than its limit, the available amount would show as a negative value. This bug fix updates the web console to show the amount over the limit in these cases. As a result, negative values no longer display for available pod CPU and memory. ([BZ#1369160](#))

## Metrics

- The web console previously used the client's clock to calculate the start time for displaying metrics. If the client's clock was more than one hour faster than the server clock, an occur would occur when opening the metrics tab in the web console. The web console now uses the server time for calculating start and end times for metrics. As a result, metrics display properly even if the client clock is out of sync with the server. ([BZ#1361061](#))

## Networking

- The new unidling feature had a bug where it removed the service proxier when unidling was disabled, causing the service to not work. This bug fix addresses this issue, and the service now works properly. ([BZ#1370435](#))
- When ipfailover was configured for the router, **keepalived** pods were previously being labeled with the selector of the router service. The router service then selected both router pods and **keepalived** pods. Because both types of pods use host networking by default, their IP addresses would be the same if deployed to the same hosts, and the service would appear to be selecting duplicate endpoints. This bug fix ensures that **keepalived** pods are now given a label that is distinct from that applied to the router pods. As a result, the router service no longer displays duplicate IP addresses when ipfailover is configured. ([BZ#1365176](#))

## Quick Starts

- This enhancement adds default resource limits to templates. Systems which require limits be set would prevent deployment of templates when the template did not specify resource limits. Templates can now be deployed on systems that require resource limits be specified. ([BZ#1314899](#))

## REST API

- Access to new endpoints was not automatically added to existing discovery roles during an upgrade. Checking the server version from the command line using **oc version** would display a forbidden error. This bug fix correctly adds permission to the new endpoint during an upgrade. As a result, **oc version** displays the server version as expected. ([BZ#1372579](#))

## Routing

- Erroneous [Patch the Router Deployment Configuration to Create a Privileged Container](#) documentation caused pods to not have enough privilege to edit **iptables**. This bug fix updates the documentation with the correct procedure. ([BZ#1269488](#))

- Multiple routers may be needed to support different features (sharding). This enhancement adds the ability to set the internal SNI port with an environment variable, allowing all ports to be changed so that multiple routers can be run on a single node. ([BZ#1343083](#))
- Editing a route then deleting it and re-creating it caused the router to panic and crash. This was due to the deletion code leading to a different, unexpected state, with an empty array after an edit was made. This bug fix hardens the code to not result in that state and to tolerate the state should it accidentally occur. As a result, the router is more robust. ([BZ#1371826](#))
- When an edge-terminated route had **`insecureEdgeTerminationPolicy`** set to **`Allow`** (meaning that the route could be accessed by both HTTP and HTTPS), the inserted session cookie was always flagged as Secure. When a client connected over HTTP, the secure cookie would be dropped, breaking session persistence. This bug fix ensures that cookies for edge-terminated routes that allow insecure connections are now set to be non-secure. As a result, session persistence for such routes is maintained. ([BZ#1368525](#))
- The F5 iControl REST API usually returns JSON payloads in its responses, but it sometimes returns error responses with HTML payloads. In particular, it can return HTML payloads with HTTP 401 and 404 responses. Previously, the router would always try to decode the payload as JSON. If the F5 iControl REST API returned an HTML response, the router logs would show the following: "error: Decoder.Decode failed: invalid character '<' looking for beginning of value". This bug fix updates the F5 router plug-in to now gracefully handle HTML responses by ignoring the response payload for HTTP 4xx and 5xx responses if decoding as JSON fails. As a result, if the F5 iControl REST API returns an HTML response, the router logs will now show a message similar to the following: "error: HTTP code: 401." ([BZ#1316463](#))
- A comment in the **`haproxy-config.template`** file about creating back ends was incomplete, causing confusion. The comment has now been completed. ([BZ#1368031](#))

## Storage

- A race condition in OpenShift Container Platform (OCP) code could cause persistent volume (PV) objects to not be deleted when their retention policy was set to Delete and the appropriate persistent volume claim (PVC) was deleted. PV handling was rewritten in OCP 3.3, and as a result PVs are now deleted at the end of their lifetime. ([BZ#1339154](#))
- A race condition in OpenShift Container Platform (OCP) code could cause an AWS EBS volume not to be detached from a node when a pod that used the volume was terminated. The volume would be attached to the node forever, consuming AWS resources. This volume had to be detached manually. The code that attaches and detaches volume to and from nodes has been rewritten in OCP 3.3, and as a result AWS EBS volumes are now detached from nodes when the last pod that uses the volume is terminated. ([BZ#1327384](#))

## Upgrades

- Previous versions allowed the user to specify **`AWS_ACCESS_KEY_ID`** and **`AWS_SECRET_ACCESS_KEY`** in their **`/etc/sysconfig/`** files for OpenShift Container Platform services. During upgrade, these files were updated according to a template, and if the user had not yet switched to using the new cloud provider framework their pre-existing AWS variables would be overwritten. The upgrade process has been modified to preserve these variables if they are present during upgrade, and a cloud provider is not configured. ([BZ#1353354](#))
- Previously, a bug in a script which cleans out all pre-existing images and containers during a **`docker`** 1.10 upgrade would cause the script to miss some images with name and tag **`none`**, potentially resulting in a slower or failed **`docker`** upgrade. This script has been updated to use a more robust method of clean-up which also catches orphaned images. ([BZ#1351406](#))

- Previously, nodes had their schedulability state reset to the state defined in the inventory used during an upgrade. If the scheduling state had been modified since the inventory file was created, this would be a surprise to administrators. The upgrade process has been modified to preserve the current schedulability state during upgrade so that nodes do not change state after an upgrade. ([BZ#1372594](#))

## 2.7. 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:

- [OpenShift Pipelines](#)
- [Extended Builds](#)
- [Service Serving Certificate Secrets](#)
- [Init Containers](#)
- Introduced in OpenShift Enterprise 3.1.1, [dynamic provisioning](#) of persistent storage volumes from Amazon EBS, Google Compute Disk, OpenStack Cinder storage providers remains in Technology Preview for OpenShift Container Platform 3.3.

## 2.8. KNOWN ISSUES

- Setting the `forks` parameter in the `/etc/ansible/ansible.cfg` file to 11 or higher is known to cause OpenShift Container Platform installations to hang with Ansible 2.2. The current default is 5. See [http://docs.ansible.com/ansible/intro\\_configuration.html#forks](http://docs.ansible.com/ansible/intro_configuration.html#forks) for more on this parameter. ([BZ#1367948](#))

## 2.9. ASYNCHRONOUS ERRATA UPDATES

Security, bug fix, and enhancement updates for OpenShift Container Platform 3.3 are released as asynchronous errata through the Red Hat Network. All OpenShift Container Platform 3.3 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 3.3. Versioned asynchronous

releases, for example with the form OpenShift Container Platform 3.3.z, will be detailed in subsections. In addition, releases in which the errata text cannot fit in the space provided by the advisory will be detailed in subsections that follow.



## IMPORTANT

For any OpenShift Container Platform release, always review the instructions on [upgrading your cluster](#) properly.

### 2.9.1. RHBA-2016:1988 - OpenShift Container Platform 3.3.0.34 Bug Fix Update

Issued: 2016-10-04

OpenShift Container Platform release 3.3.0.34 is now available. The list of packages and bug fixes included in the update are documented in the [RHBA-2016:1988](#) advisory. The list of container images included in the update are documented in the [RHBA-2016:1987](#) advisory.

#### 2.9.1.1. Upgrading

To upgrade an existing OpenShift Container Platform 3.2 or 3.3 cluster to this latest release, use the automated upgrade playbook. See [Performing Automated In-place Cluster Upgrades](#) for instructions.

### 2.9.2. RHBA-2016:2084 - OpenShift Container Platform 3.3.1.3 Bug Fix and Enhancement Update

Issued: 2016-10-27

OpenShift Container Platform release 3.3.1.3 is now available. The list of packages included in the update are documented in the [RHBA-2016:2084](#) advisory. The list of container images included in the update are documented in the [RHBA-2016:2085](#) advisory.

The following advisories are also related to the 3.3.1.3 release:

- OpenShift Container Platform logging-auth-proxy bug fix update ([RHSA-2016:2101](#) and [RHBA-2016:2100](#))
- OpenShift Container Platform Jenkins enhancement update ([RHEA-2016:2102](#) and [RHEA-2016:2103](#))
  - An updated container image for Jenkins 2 LTS has been pushed to the Red Hat Container Registry in preparation for the upcoming OpenShift Container Platform 3.4 release. Official image streams and templates will be shipped with the 3.4 release.

Space precluded documenting all of the bug fixes for this release in their advisories. See the following sections for notes on upgrading and details on the [Technology Preview](#) features and bug fixes included in this release.

#### 2.9.2.1. Upgrading

To upgrade an existing OpenShift Container Platform 3.2 or 3.3 cluster to this latest release, use the automated upgrade playbook. See [Performing Automated In-place Cluster Upgrades](#) for instructions.

#### 2.9.2.2. Technology Preview Features

##### Scheduled Jobs

Scheduled jobs build upon the [job](#) object by allowing you to specifically schedule how the job should be run. See [Scheduled Jobs](#) for more details.

### Sysctl Support

Namespaced sysctl settings can now be exposed via Kubernetes, allowing users to modify kernel parameters at runtime for namespaces within a container. Only sysctls that are namespaced can be set independently on pods; if a sysctl is not namespaced (called *node-level*), it cannot be set within OpenShift Container Platform.

OpenShift Container Platform whitelists a subset of namespaced sysctls for use in pods:

- `kernel.shm_rmid_forced`
- `net.ipv4.ip_local_port_range`

These whitelisted sysctls are considered *safe* and supported because they cannot be misused to influence other containers, for example by blocking resources like memory outside of the pods' defined memory limits. If a namespaced sysctl is not whitelisted, it is considered *unsafe*.



### IMPORTANT

The `net.ipv4.tcp_syncookies` sysctl has been whitelisted upstream because it has been namespaced in kernels  $\geq 4.6$ . However, it is not yet supported in OpenShift Container Platform 3.3 as it is not yet namespaced in RHEL 7 kernels. See [BZ#1373119](#) for details.

See [Sysctls](#) for more details and usage information.

### 2.9.2.3. Bug Fixes

#### BZ#1380544

Binaries compiled with Golang versions prior to 1.7 will segfault most of the time in macOS Sierra (10.12) given incompatibilities between the Go syscall wrappers and Darwin. Users of the OpenShift Container Platform (OCP) command-line tools (`oc`, `oadm`, and others) in macOS Sierra (10.12) get a stack trace in the attempt of running commands. The Go 1.7 fix was backported by the go-tools team to Go 1.6, which was then used to compile OCP's command-line tools in this release. As a result, users of the OCP command-line tools can use it normally in macOS Sierra (10.12).

#### BZ#1382020

With a malformed master certificate (e.g., expired, mismatched host name), the latest version of `oc login` will not ignore this problem even when `--insecure-skip-tls-verify` is set. This makes users unable to log in with `oc` when the server master certificate is invalid. This bug fix handles TLS failures more precisely and allows `--insecure-skip-tls-verify` to bypass the following error causes:

- Mismatched certificate host name
- Expired certificate
- Unauthorized CA
- Too many intermediates
- Incompatible usage with the certificate purpose

As a result, users can bypass the certificate error and log in with `--insecure-skip-tls-verify`.

#### BZ#1375480

In the web console, if you deployed an image from an image stream tag and changed the default name, an incorrect image change trigger would be set in the deployment configuration. A deployment would then fail to run because the image stream tag trigger was wrong. This bug fix updates the web console to use the correct image stream for the deployment configuration trigger. As a result, you can now change the default name on the **Add to Project** → **Deploy Image** page.

#### BZ#1377492

The download CLI link from the web console would not work if the CLI download was hosted as a web console extension. This update fixes the download link so that it will always download from the server. As a result, you can host the CLI as a static file using web console extensions.

#### BZ#1380392

In the JVM web console for A-MQ applications, a missing hawtio UI data table configuration option caused data table components to not show up correctly. This bug fix adds the primary key configuration option, and as a result the data table component now appears as expected.

#### BZ#1380421

In the JVM web console for Camel applications, a JavaScript code referencing an invalid Array function caused Camel routes to not show up correctly in the tree view. This bug fix changes the reference to a valid JavaScript Array function, and as a result Camel routes now appear in the tree view and their details are displayed as expected.

#### BZ#1381151

Previously, the download link for the CLI pointed to the OpenShift Origin repository on GitHub instead of the official product download page for OpenShift Container Platform on the Customer Portal. This bug fix updates the link to correctly link to <https://access.redhat.com/downloads/content/290>.

#### BZ#1382512

In some edge cases, a service would not appear on the **Overview** page of the web console. This could happen when a service grouped with another was also a primary service of a route with alternate back ends, causing the alternate service to not appear. This bug fix ensures that all alternate services are now shown for a route on the **Overview** page.

#### BZ#1384617

Previously, the URL for a webhook in the build configuration editor was assembled incorrectly, where variable names were used instead of the replaced values for the build configuration and project names. This bug fix addresses this issue and the correct replaced values are now used.

#### BZ#1378000

Kernels in the RHEL 7.3 beta and upcoming GA releases changed how traffic shaping is configured on network interfaces, exposing a bug in OpenShift SDN's traffic shaping feature. When traffic shaping was enabled for a pod, no traffic could be sent or received from the pod. This update fixes the openshift-sdn bug, and traffic shaping functionality with OpenShift SDN now works correctly.

#### BZ#1385824

When generating persistent volume claims (PVCs) with the logging deployer, specifying **false** for the **es-pvc-dynamic** and **es-ops-pvc-dynamic** parameters would still generate a PVC with the dynamic annotation (**volume.alpha.kubernetes.io/storage-class: dynamic**). This meant that no matter what, the generated PVC would have the dynamic annotation on them, which may be undesired. This bug fix updates the way the values of these parameters are checked to correctly evaluate if they are **true** or **false**. As a result, when generating PVCs with the logging deployer, PVCs with the dynamic annotation are only generated if these parameters are set to **true**.

#### BZ#1371220

The EFK deployer now configures the **terminationGracePeriodSeconds** for Elasticsearch and

Fluentd pods. Sometimes Elasticsearch in particular would end up in a state where it did not remove its **node.lock** file at shutdown. Elasticsearch shuts down properly and **node.lock** should be deleted, but if it takes too long to shut down, OpenShift Container Platform will hard-kill it after 30 seconds by default. If the **node.lock** is not removed from persistent storage, then when the instance is started again, Elasticsearch treats the data directory as locked and starts with a fresh data directory, effectively losing all its data. The explicit **terminationGracePeriodSeconds** gives both Fluentd and Elasticsearch more time to flush data and terminate properly so that this situation should occur less often. It cannot be completely eliminated; for example if Elasticsearch runs into an out-of-memory situation, it may be hung indefinitely and still end up being killed, leaving the **node.lock** file. This extended termination time, however, should make normal shutdown scenarios safer.

### 2.9.3. RHBA-2016:2122 - atomic-openshift-utils Bug Fix Update

Issued: 2016-10-27

OpenShift Container Platform bug fix advisory [RHBA-2016:2122](#), providing updated **atomic-openshift-utils** and **openshift-ansible** packages that fix several bugs and add enhancements, is now available.

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

#### 2.9.3.1. Upgrading

To apply this update, run the following on all hosts where you intend to initiate Ansible-based installation or upgrade procedures:

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

#### 2.9.3.2. Bug Fixes

##### BZ#1367948

In order to overcome performance regressions seen in Ansible 2.1, the installer previously updated to an early Ansible 2.2 development build. The installer is now updated to Ansible 2.2 RC1, bringing considerable reliability improvements especially when dealing with large numbers of hosts.

##### BZ#1383004

A callback plug-in method in the installer was not setting a variable to update the current play. When certain callback methods were called, the required play object is not found, causing the following error:

```
'NoneType' object has no attribute 'strategy'
```

This bug fix assigns the play object in the play start callback method, and now Ansible can call all callback methods that require the playbook object, avoiding this error.

##### BZ#1337089

The example advanced configuration hosts file documented the **openshift\_builddefaults\_json** parameter without specifying all the possible options. This bug fix updates the example value to express all the possible options currently available.

##### BZ#1366522

The **debug\_level** inventory variable was only being applied to node configuration. Debug level is now correctly set within master and node configuration, but can also be set individually via the **openshift\_master\_debug\_level** or **openshift\_node\_debug\_level** parameters.

**BZ#1369410**

Previously, nothing in a containerized installation would remove the `/etc/systemd/system/docker.service.d/docker-sdn-ovs.conf` file. At uninstall time, the **docker** service would fail to restart because of stale references in this configuration file. This bug fix updates the uninstall playbook to now remove this file for containerized installs.

**BZ#1373106**

The OpenShift Container Platform registry created by the installer is now secured by default. Management of the registry can be disabled by setting **openshift\_hosted\_manage\_registry=false** in the inventory.

**BZ#1381335**

The node `scaleup.yml` playbook did not regenerate master facts before adding new nodes, which meant that any master configuration changes made to the advanced installation hosts file were not used when configuring the additional nodes. With this bug fix, master facts are regenerated ensuring configuration changes are applied when adding additional nodes.

**BZ#1342028, BZ#1381710**

Environment variable lookups and other variable expansion within the Ansible inventory were not correctly interpreted. With this bug fix, these variables are now interpreted correctly, for example:

```
openshift_cloudprovider_aws_access_key="{{
lookup('env', 'AWS_ACCESS_KEY_ID') }}"
```

causes the **AWS\_ACCESS\_KEY\_ID** environment variable to be set as the AWS cloud provider access key.

**BZ#1371852**

A deployer script bug caused it to ignore some persistent volume claim (PVC) parameters when supplied via ConfigMap objects. This caused the deployer to not create PVCs even though the user specified **es{, -ops}-pvc-size** in the deployer ConfigMap, and Elasticsearch would start up without storage. With this bug fix, the script now references the correct script variables, not environment variables set from the deployer template parameters. As a result, PVC creation now works as expected.

**BZ#1382172**

The **MONGODB\_VERSION** parameter has been added to the MongoDB templates, allowing users to choose which version of MongoDB to deploy.

**BZ#1382636**

The **oadm** symlink incorrectly pointing to **oc** rather than the **openshift** binary on containerized master hosts. This caused upgrades to fail, complaining about missing **oadm** functionality. This bug fix transitions to using **oc adm** throughout the playbooks. As a result, the upgrade will now pass in these environments.

**BZ#1380317**

The upgrade procedure assumed the **docker** RPM package would be available in repositories on separate etcd nodes. This meant upgrades could fail if the etcd node in question did not have the **rhel-7-server-extras-rpms** repository enabled. With this bug fix, the upgrade no longer checks what version of **docker** is available if **docker** is not installed at all. As a result, the upgrade now proceeds on etcd nodes which do not have **docker** installed or available in their repositories.

**BZ#1372609**

Handlers in upgrade playbooks which restart the node service could trigger only after the node was marked schedulable again. This meant nodes could be upgraded, marked schedulable again, then immediately restarted. This bug fix ensures that handlers are now explicitly run before marking the



node schedulable again. As a result, nodes will restart before being set schedulable again.

#### BZ#1382380

The node service was incorrectly being restarted after upgrading master RPM packages. In some environments, a version mismatch could trigger between the node service and the master service that had not been restarted yet, causing the upgrade to fail. This bug fix removes the incorrect node restart and shuffles logic to ensure masters are upgraded and restarted before proceeding to node upgrade and restart. As a result, the upgrade now completes successfully.

#### BZ#1382694

Previously, the upgrade procedure restarted the node service before restarting the master services on hosts that are both masters and nodes. This caused the upgrade to fail because the master services must be updated before the node services in order to ensure new API endpoints and security policies are applied. Now the node service is only restarted when updating the node services which happens after the masters have been upgraded avoiding ensuring upgrades work as expected.

#### BZ#1361677

Due to incorrect logic interpreting inventory variables that control the version of docker to configure, it was not possible to upgrade OpenShift Container Platform and stay on a **docker** version less than 1.10. This bug fix ensures that upgrades now respect the **docker\_version** and **docker\_upgrade** inventory variables. As a result, users can now upgrade and control the version of docker to be installed more explicitly.

#### BZ#1381411

Previously, the quick installer incorrectly chose the upgrade playbook used when upgrading from OpenShift Container Platform 3.2 to 3.3, preventing upgrades from completing properly. This bug fix updates the quick installer to now use the correct playbook, ensuring upgrades from 3.2 to 3.3 work correctly.

#### BZ#1371459

The registry console is now deployed by default during OpenShift Container Platform installations.

#### BZ#1375946

Previously, the quick installer could have labeled unschedulable nodes as **infra** nodes. This would prevent the registry and router from deploying as the nodes were unschedulable. This bug fix updates the quick installer to only assign the **infra** label to schedulable nodes, ensuring that the registry and router are deployed properly.

#### BZ#1368414

When using the quick installer, additional configuration variables previously could only be defined by setting the **other\_variables** parameter in the quick installer configuration file. This bug fix updates the quick installer to include any variables defined in the **hosts** section.

#### BZ#1298336

Installations would fail if the root user's **kubeconfig** context had been changed to a different project prior to running the installer. The installer now uses a temporary **kubeconfig** file and ensures that the correct namespace is used for each OpenShift Container Platform client operation.

#### BZ#1366125

A new advanced installer configuration variable named **openshift\_master\_ingress\_ip\_network\_cidr** has been added to configure the **ingressIPNetworkCIDR**. For more information on this feature, see [Assigning Unique External IPs for Ingress Traffic](#).

### 2.9.3.3. Known Issues

- Previously when upgrading to OpenShift Container Platform 3.3, if the internal registry was not secured, the **--insecure-registry** flag was improperly removed from the

`/etc/sysconfig/docker` file on hosts, resulting in failed pushes and pulls from the integrated registry. This bug has been fixed for RPM-based hosts, and the upgrade playbooks now correctly preserve that flag during an upgrade, ensuring that insecure registries continue to work properly after upgrade.

However, this issue persists for containerized hosts. To workaround this issue, after the upgrade completes, on each containerized host set the `--insecure-registry` flag back in place for the integrated registry in the `/etc/sysconfig/docker` file, then restart the `docker` service. See the [Installing Docker](#) section of the Installation and Configuration guide for more details.

([BZ#1388016](#))

## 2.9.4. RHSA-2016:2696 - OpenShift Container Platform 3.3.1.4 Security and Bug Fix Update

Issued: 2016-11-15

OpenShift Container Platform release 3.3.1.4 is now available. The list of packages and bug fixes included in the update are documented in the [RHSA-2016:2696](#) advisory. The list of container images included in the update are documented in the [RHBA-2016:2697](#) advisory.

### 2.9.4.1. Upgrading

To upgrade an existing OpenShift Container Platform 3.2 or 3.3 cluster to this latest release, use the automated upgrade playbook. See [Performing Automated In-place Cluster Upgrades](#) for instructions.

## 2.9.5. RHBA-2016:2801 - OpenShift Container Platform 3.3.1.5 Bug Fix Update

Issued: 2016-11-17

OpenShift Container Platform release 3.3.1.5 is now available. The list of packages and bug fixes included in the update are documented in the [RHBA-2016:2801](#) advisory. The list of container images included in the update are documented in the [RHBA-2016:2800](#) advisory.

### 2.9.5.1. Upgrading

To upgrade an existing OpenShift Container Platform 3.2 or 3.3 cluster to this latest release, use the automated upgrade playbook. See [Performing Automated In-place Cluster Upgrades](#) for instructions.

## 2.9.6. RHSA-2016:2915 - OpenShift Container Platform 3.3.1.7 Security and Bug Fix Update

Issued: 2016-12-07

OpenShift Container Platform release 3.3.1.7 is now available. The list of packages and bug fixes included in the update are documented in the [RHSA-2016:2915](#) advisory. The list of container images included in the update are documented in the [RHBA-2016:2916](#) advisory.

### 2.9.6.1. Upgrading

To upgrade an existing OpenShift Container Platform 3.2 or 3.3 cluster to this latest release, use the automated upgrade playbook. See [Performing Automated In-place Cluster Upgrades](#) for instructions.

## 2.9.7. RHBA-2017:0199 - OpenShift Container Platform 3.3.1.11 Bug Fix Update

Issued: 2017-01-26

OpenShift Container Platform release 3.3.1.11 is now available. The list of packages and bug fixes included in the update are documented in the [RHBA-2017:0199](#) advisory. The list of container images included in the update are documented in the [RHBA-2017:0204](#) advisory.

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

### 2.9.7.1. Upgrading

To upgrade an existing OpenShift Container Platform 3.2 or 3.3 cluster to this latest release, use the automated upgrade playbook. See [Performing Automated In-place Cluster Upgrades](#) for instructions.

### 2.9.7.2. Bug Fixes

#### BZ#1399781

LDAP group synchronization previously failed if member DNs contained extra spaces. This bug fix corrects this issue and group synchronization now works correctly with DNs that contain spaces.

#### BZ#1414522

Proxy value validation prevented the use of default cluster proxy settings with SSH Git URLs. This caused build configurations that used SSH Git URLs in a cluster with default proxy settings to get a validation error unless the proxy value was explicitly set to empty string in the build configuration. This bug fix ensures that validation no longer rejects build configurations that use SSH Git URLs and have a proxy value set. However, the proxy value will not be used when an SSH Git URL is supplied.

#### BZ#1380555

When using the **oc cluster up** command, OpenShift Container Platform (OCP) previously attempted to load an image version of OpenShift Origin that did not exist, based on the OCP version. This caused **oc cluster up** to result in an error. This bug fix ensures that running **oc cluster up** when using the OpenShift Container Platform binary now uses the correct image name and repository, and as a result the command now works correctly.

#### BZ#1395380

Previously, running **oc cluster up** using OpenShift Container Platform (OCP) binaries would pull image streams from the OpenShift Origin community rather than using OCP image streams. This bug fix ensures that OCP image streams are now used in this case.

#### BZ#1406889

OpenShift Container Platform can now be deployed in the AWS us-east-2c region.

#### BZ#1401131

A race condition in the libcontainer SELinux library could have resulted in secret mount points being created without the rootcontext properly set, resulting in failures to read the secret. This bug fix addresses the race condition, ensuring appropriate access to secrets.

#### BZ#1400609

In OpenShift SDN, the IP addresses for a node were not sorted. When the first IP was chosen, it may be different from the last one used, so the IP address appeared to have changed. OpenShift Container Platform would then update the node-to-IP mapping, causing problems with everything moving from one interface to another. This bug fix updates OpenShift SDN to sort the addresses, and as a result the traffic flows correctly and the addresses do not change.

#### BZ#1411712

Previously, the **EgressNetworkPolicy** functionality could stop working on a node after restarting the node service. This bug fix addresses this issue.

#### BZ#1415282

In OpenShift SDN, the use of a nil pointer caused nodes to panic and generate a segment error. This bug fix checks the pointer before dereferencing, and this issue no longer occurs.

#### **BZ#1410157**

This release adds an option to allow HAProxy to expect incoming connections on port 80 or port 443 to use the **PROXY** protocol. The source IP address can pass through a load balancer if the load balancer supports the protocol, for example Amazon ELB. As a result, if the **ROUTER\_USE\_PROXY\_PROTOCOL** environment variable is set to **true** or **TRUE**, HAProxy now expects incoming connections to use the PROXY protocol.

#### **BZ#1415280**

The extended certificate validation code (now enabled by default) would not allow some certificates that should be considered valid. This caused self-signed, expired, or not yet current certificates that were otherwise well-formed to be rejected. This bug fix changes the extended validation to allow those cases. As a result, those types of certificates are now allowed.

#### **BZ#1404032**

OpenShift Container Platform previously made requests to delete volumes without any sort of exponential back-off. Making too many requests to the cloud provider could exhaust the API quota. This bug fix implements exponential back-off when trying to delete a volume. As a result, the API quota is no longer exhausted as easily.

#### **BZ#1403696**

The **atomic-openshift-excluder** and **atomic-openshift-docker-excluder** packages did not properly configure yum to exclude the relevant packages. The excluder scripts have been updated to ensure the proper yum configuration is modified ensuring that the appropriate packages are excluded from yum operations.

### **2.9.8. RHBA-2017:0289 - OpenShift Container Platform 3.3.1.14 Bug Fix Update**

Issued: 2017-02-22

OpenShift Container Platform release 3.3.1.14 is now available. The list of packages and bug fixes included in the update are documented in the [RHBA-2017:0289](#) advisory. The list of container images included in the update are documented in the [RHBA-2017:0290](#) advisory.

The container images in this release have been updated using the **rhel:7.3-66** and **jboss-base-7/jdk8:1.3-6** base images, where applicable.

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

#### **2.9.8.1. Upgrading**

To upgrade an existing OpenShift Container Platform 3.2 or 3.3 cluster to this latest release, use the automated upgrade playbook. See [Performing Automated In-place Cluster Upgrades](#) for instructions.

#### **2.9.8.2. Bug Fixes**

##### **BZ#1416506**

A race condition when updating a batch of nodes to schedulable or unschedulable with **oadm manage-node --schedulable=<true|false>** caused several nodes to not get updated, failing with the "object has been modified" error. This bug fix patches the **unschedulable** field of the node object instead of a full update. As a result, all nodes can be properly updated schedulable or unschedulable.

**BZ#1417723**

Previously **atomic-openshift-docker-excluder** did not include **docker-1.12** in its exclusions list. OpenShift Container Platform 3.3 is not tested with **docker-1.12**, so that version has been added to the exclusion list to prevent its installation.

**BZ#1415276**

Previously, the router would not reload HAProxy after the initial sync if the last item of the initial list of any of the watched resources did not reach the router to trigger the commit. This could be caused by a route being rejected for any reason, for example specifying a host claimed by another namespace. The router could be left in its initial state (without any routes configured) until another commit-triggering event occurred, such as a watch event. This bug fix updates the router to always reload after initial sync. As a result, routes are available after the initial sync.

**BZ#1408129**

The HAProxy router configuration previously specified the directory from where certificates would be used for serving. Based on the SNI header, HAProxy would pick up the first certificate that matched the secure request and use that as the serving certificate. This was an indeterministic method if multiple routes specified the same certificate or expired versions of the same certificate. This bug fix now uses a new map and serves the certificate based on the SNI header and host in a deterministic fashion.

**BZ#1412829**

Previously, the extended validation in the router was turned off by default. In addition, if extended validation was enabled, a route with an expired certificate would be marked as an extended validation failure. With this bug fix, the router now admits routes with expired certificates. In addition, this backport enhances the extended validation with changes made in 3.4 and also turns on extended validation by default.

**BZ#1412367, BZ#1414438**

The registry S3 storage driver now supports the **ca-central-1** AWS region.

## 2.9.9. RHSA-2017:0448 - ansible and openshift-ansible Security and Bug Fix Update

Issued: 2017-03-06

OpenShift Container Platform security and bug fix advisory [RHSA-2017:0448](#), providing updated **atomic-openshift-utils**, **ansible**, and **openshift-ansible** packages that fix several bugs and a security issue, is now available.

The security issue is documented in the advisory. However, space precluded documenting all of the non-security bug fixes for this release in the advisory. See the following sections for notes on upgrading and details on the bug fixes included in this release.

### 2.9.9.1. Upgrading

To apply this update, run the following on all hosts where you intend to initiate Ansible-based installation or upgrade procedures:

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

### 2.9.9.2. Bug Fixes

This release fixes bugs for the following components:

## Installer

- Previously, if **ansible\_user** was a Windows domain user in the format of **dom\user**, the installation playbooks would fail. This user name is now escaped properly, ensuring playbooks run successfully. ([BZ#1414276](#))
- OpenShift Container Platform 3.4 and 3.3 introduced a requirement on the **contrack** executable, but this dependency was not enforced at install time, so service proxy management may have failed post-installation. The installer now ensures that **contrack** is installed. ([BZ#1420395](#))
- An Ansible 2.2.1.0 compatibility issue has been fixed in the quick installer. ([BZ#1421061](#))
- When executing the installer on a remote host that is also included in the inventory, the firewall configuration could potentially cause the installer to hang. A 10 second delay has been added after resetting the firewall to avoid this problem from occurring. ([BZ#1416926](#))
- A [certificate expiry checker](#) has been added to the installer tools. ([BZ#1417682](#))

## Upgrades

- Previously when upgrading to OpenShift Container Platform 3.3, if the integrated registry was not secured, the **--insecure-registry** flag was improperly removed from the **/etc/sysconfig/docker** file, resulting in failed pushes and pulls from the integrated registry. With this bug fix, the upgrade playbooks have been updated to preserve that flag during an upgrade, ensuring that insecure registries continue to work properly after upgrade. ([BZ#1388016](#))
- Previously, API verification during upgrades was performed from the Ansible control host, which may not have network access to each API server in some network topologies. With this bug fix, API server verification happens from the master hosts, avoiding problems with network access. ([BZ#1393000](#))

### 2.9.10. RHBA-2017:0512 - OpenShift Container Platform 3.3.1.17 Bug Fix Update

Issued: 2017-03-15

OpenShift Container Platform release 3.3.1.17 is now available. The list of packages and bug fixes included in the update are documented in the [RHBA-2017:0512](#) advisory. The list of container images included in the update are documented in the [RHBA-2017:0513](#) advisory.

The container images in this release have been updated using the **rhel:7.3-74** and **jboss-base-7/jdk8:1.3-10** base images, where applicable.

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

#### 2.9.10.1. Upgrading

To upgrade an existing OpenShift Container Platform 3.2 or 3.3 cluster to this latest release, use the automated upgrade playbook. See [Performing Automated In-place Cluster Upgrades](#) for instructions.

#### 2.9.10.2. Bug Fixes

This release fixes bugs for the following components:

#### Metrics

- The Heapster password was being set via a property value, so the password could be leaked by processes such as **ps**. This bug fix ensures the password is now being set via a system property. As a result, the password is no longer leaked by such processes. ([BZ#1427542](#))
- The passwords for Hawkular Metrics were being set via a property value, so the passwords could be leaked by processes such as **ps**. This bug fix ensures the passwords are now being set via a property file. As a result, the passwords are no longer leaked by such processes. ([BZ#1424137](#))
- If cluster metrics are enabled, size tiered compaction in Cassandra can be inefficient with time series data. SSTables are merged solely based on file size, which means expired data can get mixed with live data. This can result in tombstones (i.e., deletion markers) being scanned on read requests. In and of itself, this is not a problem but it will slow down reads and can increase the chances of an **OutOfMemoryError** since the deletion markers are loaded into the Java heap. This bug fix switches Cassandra to use time window compaction, which is tailored specifically for time series data. It merges SSTables in a such a way that expired data is excluded. As a result, reads and Cassandra overall are more stable. ([BZ#1416850](#))

## Storage

- If the same iSCSI device was used by multiple pods on same node, when one pod shut down, the iSCSI device for the other pod would be unavailable. This bug fix addresses the issue and it no longer occurs. ([BZ#1426775](#))

## 2.9.11. RHBA-2017:0865 - OpenShift Container Platform 3.3.1.17-4 Bug Fix Update

Issued: 2017-04-04

OpenShift Container Platform release 3.3.1.17-4 is now available. The list of packages and bug fixes included in the update are documented in the [RHBA-2017:0865](#) advisory. The list of container images included in the update are documented in the [RHBA-2017:0866](#) advisory.

The container images in this release have been updated using the **rhel:7.3-74** base image, where applicable.

### 2.9.11.1. Upgrading

To upgrade an existing OpenShift Container Platform 3.2 or 3.3 cluster to this latest release, use the automated upgrade playbook. See [Performing Automated In-place Cluster Upgrades](#) for instructions.

## 2.9.12. RHBA-2017:0989 - OpenShift Container Platform 3.3.1.19 Bug Fix Update

Issued: 2017-04-19

OpenShift Container Platform release 3.3.1.19 is now available. The list of packages and bug fixes included in the update are documented in the [RHBA-2017:0989](#) advisory. The list of container images included in the update are documented in the [RHBA-2017:0990](#) advisory.

### 2.9.12.1. Upgrading

To upgrade an existing OpenShift Container Platform 3.2 or 3.3 cluster to this latest release, use the automated upgrade playbook. See [Performing Automated In-place Cluster Upgrades](#) for instructions.

## 2.9.13. RHBA-2017:1129 - OpenShift Container Platform 3.3.1.20 Bug Fix Update

Issued: 2017-04-26

OpenShift Container Platform release 3.3.1.20 is now available. The list of packages and bug fixes included in the update are documented in the [RHBA-2017:1129](#) advisory. The list of container images included in the update are documented in the [RHBA-2017:1130](#) advisory.

### 2.9.13.1. Upgrading

To upgrade an existing OpenShift Container Platform 3.2 or 3.3 cluster to this latest release, use the automated upgrade playbook. See [Performing Automated In-place Cluster Upgrades](#) for instructions.

## 2.9.14. RHBA-2017:1235 - OpenShift Container Platform 3.3.1.25 Bug Fix Update

Issued: 2017-05-18

OpenShift Container Platform release 3.3.1.25 is now available. The list of packages and bug fixes included in the update are documented in the [RHBA-2017:1235](#) advisory. The list of container images included in the update are documented in the [RHBA-2017:1236](#) advisory.

### 2.9.14.1. Upgrading

To upgrade an existing OpenShift Container Platform 3.2 or 3.3 cluster to this latest release, use the automated upgrade playbook. See [Performing Automated In-place Cluster Upgrades](#) for instructions.

## 2.9.15. RHBA-2017:1235 - OpenShift Container Platform 3.3.1.35 Bug Fix Update

Issued: 2017-06-15

OpenShift Container Platform release 3.3.1.35 is now available. The packages and bug fixes included in the update are documented in the [RHBA-2017:1425](#) advisory. The container images included in the update are provided by the [RHBA-2017:1426](#) advisory and listed in [Images](#).

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

### 2.9.15.1. Upgrading

To upgrade an existing OpenShift Container Platform 3.2 or 3.3 cluster to this latest release, use the automated upgrade playbook. See [Performing Automated In-place Cluster Upgrades](#) for instructions.

### 2.9.15.2. Bug Fixes

- The OpenShift Container Platform cluster used to store manifests of all images in the etcd database. They occupied a lot of space in the database, making it slow. OpenShift Container Registry (OCR) now stores manifests in its associated storage rather than in etcd. Manifests of remote images are not stored, but fetched from external registries when needed. A provided migration script moves manifests from all existing images in the cluster into OCR. Newly pushed images do not cause the etcd database to grow so fast. By using the script, the administrator reduces the etcd size. ([BZ1418358](#))
- Multiple node IP addresses were reported in random order by node status. Consequently, the SDN controller picked up a random one each time. This bug fix maintains the stickiness of the IP once it is chosen until valid, and IP addresses are no longer switched unexpectedly. ([BZ#1451818](#))



- The ARP cache size tuning parameters were not set when performing an installation on bare metal hosts. The bare metal profiles are now updated to ensure that the ARP cache is set correctly on bare metal hosts. ([BZ#1452402](#))

### 2.9.15.3. Images

This release updates the Red Hat Container Registry ([registry.access.redhat.com](#)) with the following images:

```
openshift3/ose-pod:v3.3.1.35-2
rhel7/pod-infrastructure:v3.3.1.35-2
openshift3/ose:v3.3.1.35-2
openshift3/ose-docker-registry:v3.3.1.35-2
openshift3/ose-egress-router:v3.3.1.35-2
openshift3/ose-keepalived-ipfailover:v3.3.1.35-2
openshift3/ose-f5-router:v3.3.1.35-2
openshift3/ose-deployer:v3.3.1.35-2
openshift3/ose-haproxy-router:v3.3.1.35-2
openshift3/node:v3.3.1.35-2
openshift3/ose-recycler:v3.3.1.35-2
openshift3/ose-sti-builder:v3.3.1.35-2
openshift3/ose-docker-builder:v3.3.1.35-2
openshift3/logging-deployer:v3.3.1.35-2
openshift3/metrics-deployer:v3.3.1.35-2
openshift3/openvswitch:v3.3.1.35-2
openshift3/logging-auth-proxy:3.3.1-16
openshift3/logging-curator:3.3.1-16
openshift3/logging-elasticsearch:3.3.1-16
openshift3/logging-fluentd:3.3.1-17
openshift3/logging-kibana:3.3.1-16
openshift3/metrics-cassandra:3.3.1-17
openshift3/metrics-hawkular-metrics:3.3.1-17
openshift3/metrics-heapster:3.3.1-18
openshift3/registry-console:3.3-20
```

### 2.9.16. RHBA-2017:1492 - OpenShift Container Platform 3.3.1.38 Bug Fix Update

Issued: 2017-06-20

OpenShift Container Platform release 3.3.1.38 is now available. The list of packages and bug fixes included in the update are documented in the [RHBA-2017:1492](#) advisory. The list of container images included in the update are documented in the [RHBA-2017:1493](#) advisory.

#### 2.9.16.1. Upgrading

To upgrade an existing OpenShift Container Platform 3.2 or 3.3 cluster to this latest release, use the automated upgrade playbook. See [Performing Automated In-place Cluster Upgrades](#) for instructions.

### 2.9.17. RHBA-2017:1666 - atomic-openshift-utils Bug Fix and Enhancement Update

Issued: 2017-06-29

OpenShift Container Platform bug fix and enhancement advisory [RHBA-2017:1666](#), providing updated **atomic-openshift-utils** and **openshift-ansible** packages that fix several bugs and add enhancements, is now available.

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

### 2.9.17.1. Upgrading

To apply this update, run the following on all hosts where you intend to initiate Ansible-based installation or upgrade procedures:

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

### 2.9.17.2. Bug Fixes

- If etcd 3.x or later was running on the host, a v3 snapshot database must be backed up as part of the backup process. If this directory is not included in the backup, then etcd failed to restore the backup even though v3 data was not used. This bug fix amends the etcd backup steps to ensure that the v3 snapshot database is included in backups. ([BZ#1440299](#))
- The OpenShift CA redeployment playbook (*playbooks/byo/openshift-cluster/redeploy-openshift-ca.yml*) would fail to restart services if certificates were previously expired. This bug fix ensures that service restarts are now skipped within the OpenShift CA redeployment playbook when expired certificates are detected. Expired cluster certificates may be replaced with the certificate redeployment playbook (*playbooks/byo/openshift-cluster/redeploy-certificates.yml*) after the OpenShift CA certificate has been replaced via the OpenShift CA redeployment playbook. ([BZ#1460971](#))
- Previously, installation would fail in multi-master environments in which the load balanced API was listening on a different port than that of the OpenShift Container Platform API and web console. This bug fix accounts for this difference and ensures the master loopback client configuration is configured to interact with the local master. ([BZ#1462282](#))
- During certificate expiration checking or redeployment, certificates with large serial numbers could not be parsed using the existing manual parser workaround on hosts that were missing the OpenSSL python library. This bug fix updates the manual parser to account for the format of certificates with large serial numbers. As a result, these certificates can now be parsed. ([BZ#1462282](#))

### 2.9.17.3. Enhancements

- Previously, it was only possible to redeploy the etcd CA certificate by also redeploying the OpenShift CA certificate, which was unnecessary maintenance. With this enhancement, the etcd CA certificate may now be replaced independent of the OpenShift CA certificate using the etcd CA certificate redeployment playbook (*playbooks/byo/openshift-cluster/redeploy-etcd-ca.yml*). Note that the OpenShift CA redeployment playbook (*playbooks/byo/openshift-cluster/redeploy-openshift-ca.yml*) now only replaces the OpenShift CA certificate. Similarly, the etcd CA redeployment playbook only redeployes the etcd CA certificate. ([BZ#1463774](#))
- Each OpenShift Container Platform version works properly with specific range of versions of packages. Thus, the package versions must be limited and the ranges enforced. This enhancement extends the installation and upgrade playbooks to install the **\*-excluder** packages

that protects RPMs against upgrading to undesired versions. As a result, the range of versions of packages for each OpenShift Container Platform version (since 3.3) is now protected.

([BZ#1436348](#))

## 2.9.18. RHBA-2017:1828 - OpenShift Container Platform 3.3.1.46 Bug Fix Update

Issued: 2017-08-31

OpenShift Container Platform release 3.3.1.46 is now available. The packages and bug fixes included in the update are documented in the [RHBA-2017:1828](#) advisory. The container images included in the update are provided by the [RHBA-2017:1829](#) advisory and listed in [Images](#).

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

### 2.9.18.1. Images

This release updates the Red Hat Container Registry ([registry.access.redhat.com](#)) with the following images:

```
openshift3/ose-pod:v3.3.1.46.11-3
rhel7/pod-infrastructure:v3.3.1.46.11-3
openshift3/ose-ansible:v3.3.1.46.11-3
openshift3/ose:v3.3.1.46.11-3
openshift3/ose-docker-registry:v3.3.1.46.11-3
openshift3/ose-egress-router:v3.3.1.46.11-3
openshift3/ose-keepalived-ipfailover:v3.3.1.46.11-3
openshift3/ose-f5-router:v3.3.1.46.11-3
openshift3/ose-deployer:v3.3.1.46.11-3
openshift3/ose-haproxy-router:v3.3.1.46.11-3
openshift3/node:v3.3.1.46.11-3
openshift3/ose-recycler:v3.3.1.46.11-3
openshift3/ose-sti-builder:v3.3.1.46.11-3
openshift3/ose-docker-builder:v3.3.1.46.11-3
openshift3/logging-deployer:v3.3.1.46.11-3
openshift3/logging-curator:v3.3.1.46.13-3
openshift3/metrics-deployer:v3.3.1.46.11-3
openshift3/openvswitch:v3.3.1.46.11-3
openshift3/logging-auth-proxy:3.3.1-22
openshift3/logging-elasticsearch:3.3.1-22
openshift3/logging-fluentd:3.3.1-23
openshift3/logging-kibana:3.3.1-22
openshift3/metrics-cassandra:3.3.1-25
openshift3/metrics-hawkular-metrics:3.3.1-25
openshift3/metrics-heapster:3.3.1-24
openshift3/registry-console:3.3-28
```

### 2.9.18.2. Upgrading

To upgrade an existing OpenShift Container Platform 3.2 or 3.3 cluster to this latest release, use the automated upgrade playbook. See [Performing Automated In-place Cluster Upgrades](#) for instructions.

## 2.9.19. RHSA-2018:3754 - OpenShift Container Platform 3.3.1.46.45 Security and Bug Fix Update

Issued: 2018-12-04

OpenShift Container Platform release 3.3.1.46.45 is now available. The list of packages and bug fixes included in the update are documented in the [RHSA-2018:3754](#) advisory. The list of container images included in the update are documented in the [RHBA-2018:3753](#) advisory.

### **2.9.19.1. Upgrading**

To upgrade an existing OpenShift Container Platform 3.2 or 3.3 cluster to this latest release, use the automated upgrade playbook. See [Performing Automated In-place Cluster Upgrades](#) for instructions.

## CHAPTER 3. XPAAS RELEASE NOTES

The release notes for xPaaS docs have migrated to their own book on the [Red Hat customer portal](#).

## CHAPTER 4. COMPARING WITH OPENSIFT 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 vs 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 **OPENSIFT\_MYSQL\_DB\_HOST** and **OPENSIFT\_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 PaaS 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 VS 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 VS 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 VS PROJECT

A [project](#) is essentially a v2 domain.



## CHAPTER 5. REVISION HISTORY: RELEASE NOTES

### 5.1. THU AUG 31 2017

Affected Topic	Description of Change
<a href="#">OpenShift Container Platform 3.3 Release Notes</a>	Added release notes for <a href="#">RHBA-2017:1828 - OpenShift Container Platform 3.3.1.46 Bug Fix Update</a> .

### 5.2. WED JUL 12 2017

Affected Topic	Description of Change
<a href="#">Overview</a>	Clarified that "client" referred to the <b>oc</b> client.

### 5.3. THU JUN 29 2017

Affected Topic	Description of Change
<a href="#">OpenShift Container Platform 3.3 Release Notes</a>	Added release notes for <a href="#">RHBA-2017:1666 - atomic-openshift-utils Bug Fix and Enhancement Update</a> .

### 5.4. THU JUN 22 2017

Affected Topic	Description of Change
<a href="#">OpenShift Container Platform 3.3 Release Notes</a>	Added release notes for <a href="#">RHBA-2017:1492 - OpenShift Container Platform 3.3.1.38 Bug Fix Update</a> .
	Added issued dates for all <a href="#">Asynchronous Errata Updates</a> . ( <a href="#">BZ#1463721</a> )

### 5.5. WED JUNE 14 2017

Affected Topic	Description of Change
<a href="#">OpenShift Container Platform 3.3 Release Notes</a>	Added release notes for <a href="#">RHBA-2017:1235 - OpenShift Container Platform 3.3.1.35 Bug Fix Update</a> .

### 5.6. THU MAY 18 2017

Affected Topic	Description of Change
<a href="#">OpenShift Container Platform 3.3 Release Notes</a>	Added release notes for <a href="#">RHBA-2017:1235 - OpenShift Container Platform 3.3.1.25 Bug Fix Update</a> .

## 5.7. TUE APR 25 2017

Affected Topic	Description of Change
<a href="#">OpenShift Container Platform 3.3 Release Notes</a>	Added release notes for <a href="#">RHBA-2017:0989 - OpenShift Container Platform 3.3.1.19 Bug Fix Update</a> and <a href="#">RHBA-2017:1129 - OpenShift Container Platform 3.3.1.20 Bug Fix Update</a> .

## 5.8. THU APR 06 2017

Affected Topic	Description of Change
<a href="#">OpenShift Container Platform 3.3 Release Notes</a>	Added release notes for <a href="#">RHBA-2017:0855 - OpenShift Container Platform 3.3.1.17-4 Bug Fix Update</a> .

## 5.9. WED MAR 15 2017

Affected Topic	Description of Change
<a href="#">OpenShift Container Platform 3.3 Release Notes</a>	Added release notes for <a href="#">RHBA-2017:0512 - OpenShift Container Platform 3.3.1.17 Bug Fix Update</a> .

## 5.10. TUE MAR 07 2017

Affected Topic	Description of Change
<a href="#">OpenShift Container Platform 3.3 Release Notes</a>	Added release notes for <a href="#">RHSA-2017:0448 - ansible and openshift-ansible Security and Bug Fix Update</a> .

## 5.11. WED FEB 22 2017

Affected Topic	Description of Change
<a href="#">OpenShift Container Platform 3.3 Release Notes</a>	Added release notes for <a href="#">RHBA-2017:0289 - OpenShift Container Platform 3.3.1.14 Bug Fix Update</a> .

## 5.12. THU JAN 26 2017

Affected Topic	Description of Change
<a href="#">OpenShift Container Platform 3.3 Release Notes</a>	Added release notes for <a href="#">RHBA-2017:0199 - OpenShift Container Platform 3.3.1.11 Bug Fix Update</a> .

## 5.13. WED DEC 07 2016

Affected Topic	Description of Change
<a href="#">OpenShift Container Platform 3.3 Release Notes</a>	Added release notes for <a href="#">RHSA-2016:2915 - OpenShift Container Platform 3.3.1.7 Security and Bug Fix Update</a> .

## 5.14. MON NOV 21 2016

Affected Topic	Description of Change
<a href="#">OpenShift Container Platform 3.3 Release Notes</a>	Updated entry on <a href="#">sysctl support</a> to link to the new dedicated <a href="#">Sysctls</a> topic.

## 5.15. THU NOV 17 2016

Affected Topic	Description of Change
<a href="#">OpenShift Container Platform 3.3 Release Notes</a>	Added release notes for <a href="#">RHBA-2016:2801 - OpenShift Container Platform 3.3.1.5 Bug Fix Update</a> .

## 5.16. TUE NOV 15 2016

Affected Topic	Description of Change
<a href="#">OpenShift Container Platform 3.3 Release Notes</a>	Added release notes for <a href="#">RHSA-2016:2696 - OpenShift Container Platform 3.3.1.4 Security and Bug Fix Update</a> .

## 5.17. WED NOV 02 2016

Affected Topic	Description of Change
<a href="#">OpenShift Container Platform 3.3 Release Notes</a>	Updated release notes for <a href="#">RHBA-2016:2084 - OpenShift Container Platform 3.3.1.3 Bug Fix Update</a> to document <a href="#">Technology Preview Features</a> included in the release: scheduled jobs and systemctl support.

## 5.18. THU OCT 27 2016

OpenShift Container Platform 3.3.1 release.

Affected Topic	Description of Change
<a href="#">OpenShift Container Platform 3.3 Release Notes</a>	Added release notes for <a href="#">RHBA-2016:2084 - OpenShift Container Platform 3.3.1.3 Bug Fix Update</a> and <a href="#">RHBA-2016:2122 - atomic-openshift-utils Bug Fix Update</a> .

## 5.19. WED OCT 05 2016

Affected Topic	Description of Change
<a href="#">OpenShift Container Platform 3.3 Release Notes</a>	Added Service Serving Certificate Secrets to <a href="#">Technology Preview Features</a> .

## 5.20. TUE OCT 04 2016

Affected Topic	Description of Change
<a href="#">OpenShift Container Platform 3.3 Release Notes</a>	Added release notes for <a href="#">RHBA-2016:1988 - OpenShift Container Platform 3.3.0.34 Bug Fix Update</a> .

## 5.21. TUE SEP 27 2016

OpenShift Container Platform 3.3 initial release.

Affected Topic	Description of Change
<a href="#">OpenShift Container Platform 3.3 Release Notes</a>	Added release notes for initial release.