

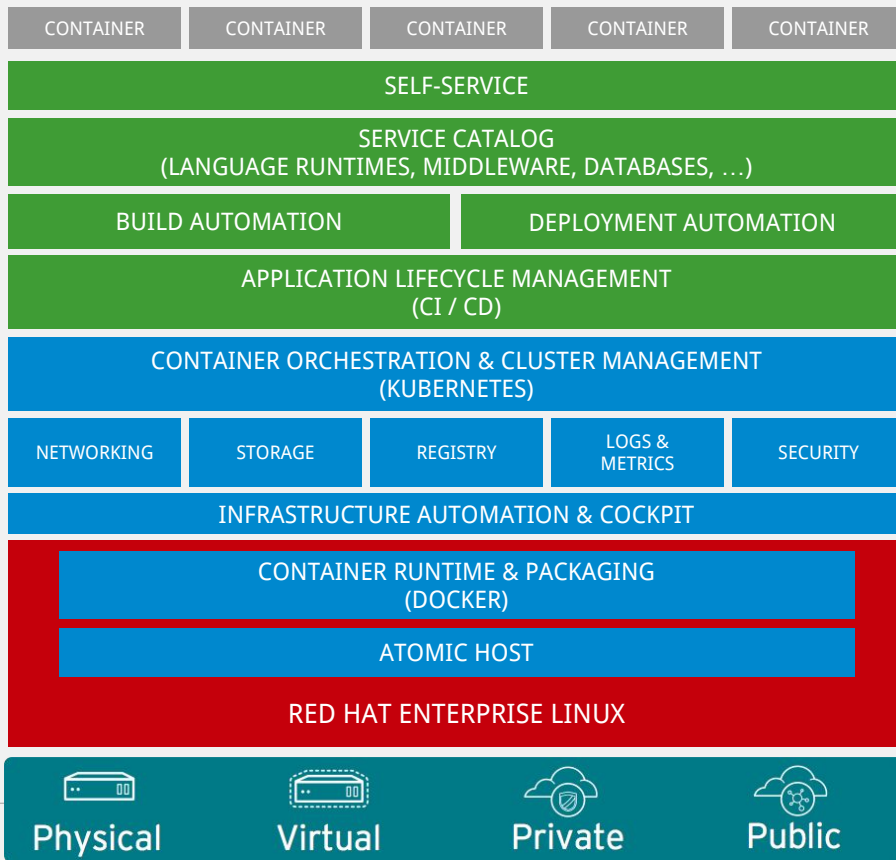
What's New in Red Hat OpenShift Container Platform 3.6

AOS Product Management Team
July 2017



OpenShift = Enterprise Kubernetes+

Build, Deploy and Manage Containerized Apps



OpenShift Roadmap

OpenShift Container Platform 3.4 (January)

- Kubernetes 1.4 & Docker 1.12
- Dynamic Storage provisioning & Storage QoS tiers
- Storage size quota and scopes
- Kubernetes Deployments & Job Scheduler API
- Pod Disruption Budget
- Eviction on File System Usage
- Usability enhancements & first-time user flows
- Build enhancements (performance, integrations)
- CNI integration for openshift-sdn
- Integration to external logging systems (Splunk)
- Registry enhancements

OpenShift Online & Dedicated

- OpenShift Dedicated on Google (Dec 8)
- OpenShift Online Dev Preview User Expansion

OpenShift Container Platform 3.6 (August)

- Kubernetes 1.6 & Docker 1.12
- New Application Services - 3Scale API Mgt OnPrem, SCL 2.4
- Web UX Project Overview enhancements
- Service Catalog/Broker & UX (Tech Preview)
- Ansible Service Broker (Tech Preview)
- Secrets Encryption (3.6.1)
- Signing/Scanning + OpenShift integration
- Storage - Integrated CNS for Reg and Installer, AWS EFS
- OverlayFS with SELinux Support (RHEL 7.4)
- User Namespaces (RHEL 7.4)
- System Containers for docker

OpenShift Online & Dedicated

- OpenShift Online Paid Tier GA (July)

Q2 CY2017

Q4 CY2017

Q1 CY2017

Q3 CY2017

OpenShift Container Platform 3.5 (April)

- Kubernetes 1.5 & Docker 1.12
- Kubernetes StatefulSets (Tech Preview)
- Java S2I for Cloud Native
- CD Pipeline enhancements
- Dynamic Provisioning for Azure block
- Network Policy (Tech Preview) & Multicast
- Opaque Integers (Tech Preview)
- Registry Image Handling Options
- SCC Usability
- SCL 2.3 shipped out of box

OpenShift Online & Dedicated

- OpenShift Online Free Tier GA (May/Summit)
- OpenShift.io Launch (May/Summit)

OpenShift Container Platform 3.7 (November)

- Kubernetes 1.7 & Docker 1.13
- Red Hat OpenShift Application Runtimes (GA)
- Service Catalog/Broker & UX (GA)
- Ansible Service Broker (GA)
- AWS Service Catalog
- Network Policy (GA)
- Cluster Federation (Tech Preview)
- Prometheus Real-Time Metrics (Tech Preview)
- CloudForms CM-Ops (CloudForms 4.6)
- Performance Sensitive Apps & Spark (TBD)

OpenShift Online & Dedicated

- OpenShift Dedicated Enhancements
- OpenShift Online Enhancements

OpenShift Container Platform 3.6 (August)

- Kubernetes 1.6 & Docker 1.12
- New Application Services - 3Scale API Mgt OnPrem, SCL 2.4
- Web UX Project Overview enhancements
- Service Catalog/Broker & UX (Tech Preview)
- Ansible Service Broker (Tech Preview)
- Secrets Encryption (3.6.1)
- Signing/Scanning + OpenShift integration
- Storage - Integrated CNS for Reg and Installer, AWS EFS
- OverlayFS with SELinux Support (RHEL 7.4)
- User Namespaces (RHEL 7.4)
- System Containers for docker

OpenShift Online & Dedicated

- OpenShift Online Paid Tier GA (July)

OpenShift Container Platform 3.7 (November)

- Kubernetes 1.7 & Docker 1.13
- Red Hat OpenShift Application Runtimes (GA)
- Service Catalog/Broker & UX (GA)
- Ansible Service Broker (GA)
- AWS Service Catalog
- Network Policy (GA)
- Cluster Federation (Tech Preview)
- Prometheus Real-Time Metrics (Tech Preview)
- CloudForms CM-Ops (CloudForms 4.6)
- Performance Sensitive Apps & Spark (TBD)

OpenShift Online & Dedicated

- OpenShift Dedicated Enhancements
- OpenShift Online Enhancements

OCP 3.6 - Primary Launch Themes

- Expanded Applications Support
 - ServiceCatalog (Preview), Ansible & Template Service Brokers (Preview)
- Security for Containers
 - Better certificate management, Signing, RHEL 7.4 support
- Kubernetes 1.6
 - Orchestration, controller, advanced scheduling and related enhancements

Service Catalog

install servers

provision services

create credentials

share credentials

manual work/ticket handling



automated service provisioning, connection
and credential configuration

Not for production workloads
Using alpha version of kube service catalog



Microservices Application

ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt.



Mobile Application

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt.



Integration Application

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt.



Business Process Application

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt.

Service Catalog

Languages Middleware



EnMasse Anycast



EnMasse Multicast



EnMasse Queue



EnMasse Topic

RED HAT
JBoss

fis-java-openshift

RED HAT
JBoss

fis-karaf-openshift



Node.js



Perl



PHP



Python



Ruby



WildFly

Not for production workloads
Using alpha version of kube service catalog

+ Create Project

View All

MLB

mlb-parks - created by developer 17 hours ago

enmasse-edbf51b9

created 17 hours ago

Game Apps - Dev environment

dev-project - created by admin a day ago

enmasse

created by system:admin a day ago

service-catalog

created by developer 2 days ago

Not for production workloads
Using alpha version of kube service catalog

THE PIECES

The Service Consumer:

The **human** or **app/service** that uses a service enabled by the broker and catalog

The Catalog

Where services are published for consumption.

Service Broker:

Publishes services and Intermediates service creation and credential configuration with a provider.

Service Provider

The technology delivering the service.

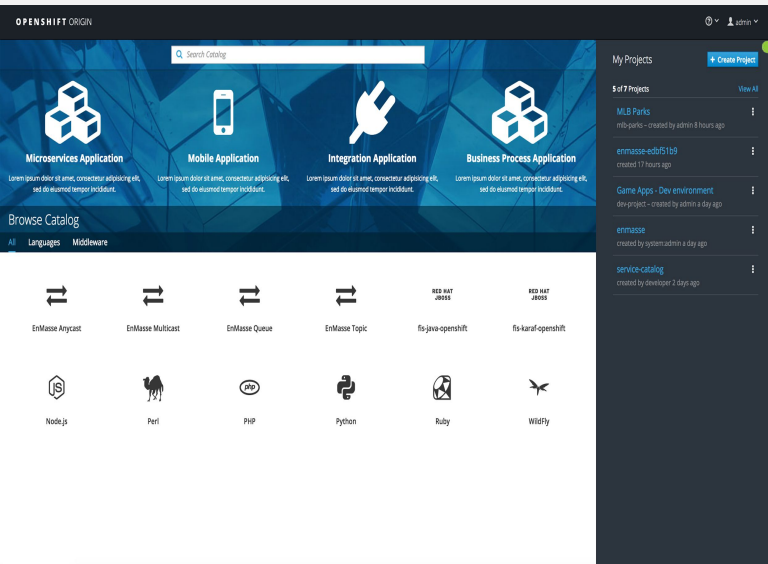


OPEN SERVICE BROKER API™

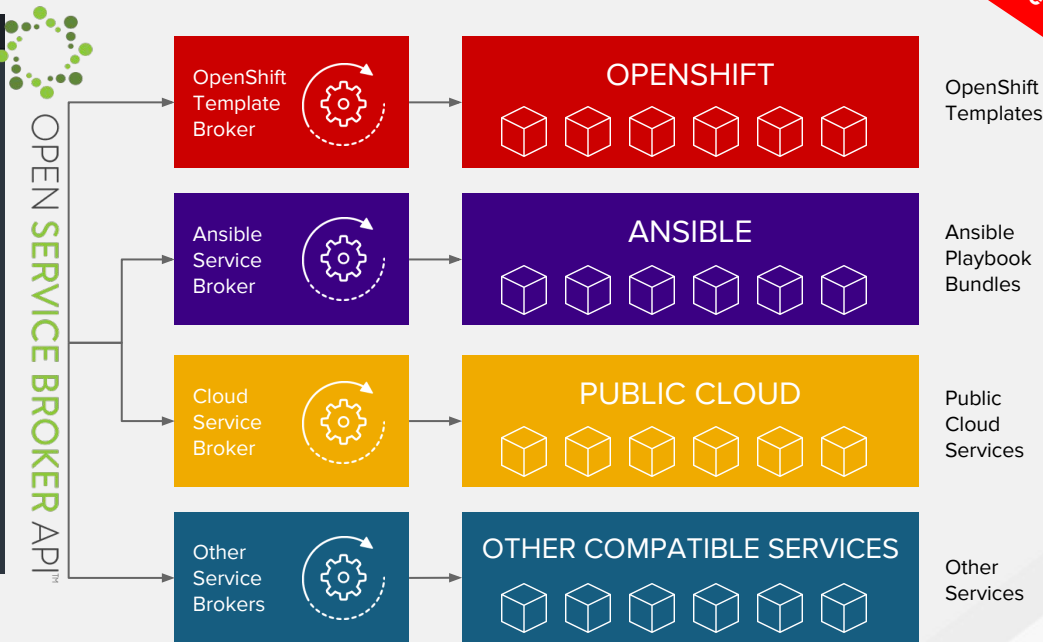
- List Services
- Provision / Deprovision
- Bind / Unbind

Service Brokers

Not for production workloads
Using alpha version of kube service catalog



**OPENSHIFT SERVICE CATALOG
(TECH PREVIEW)**

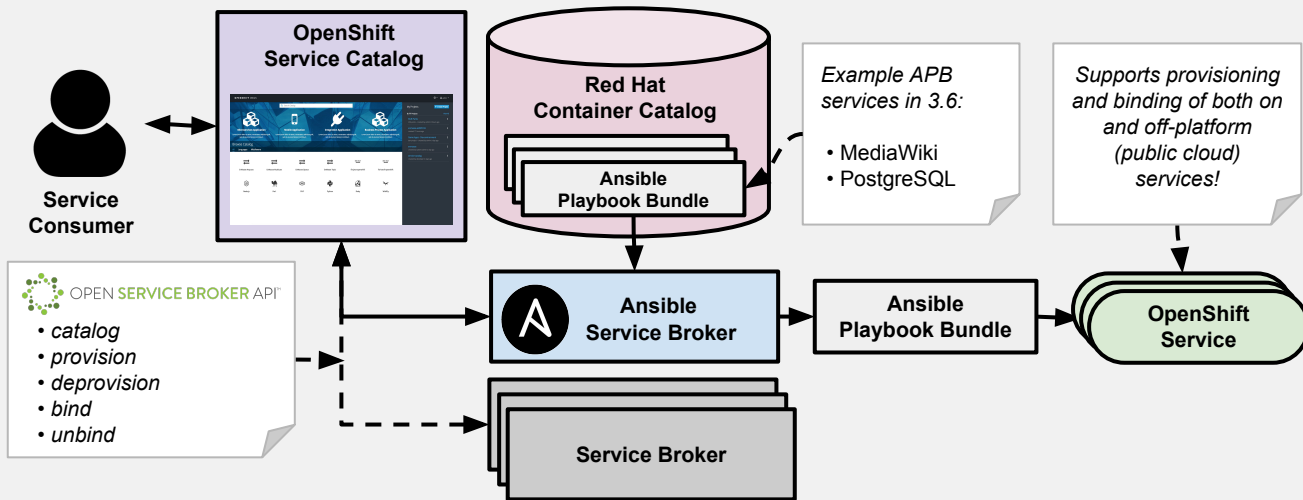


Self-Service / UX tech preview



ANSIBLE

Feature(s): Ansible Service Broker



How it Works:

Service Catalog and ASB must be configured during OpenShift installation. Once enabled, APB services can be deployed right from Service Catalog UI.

Description:

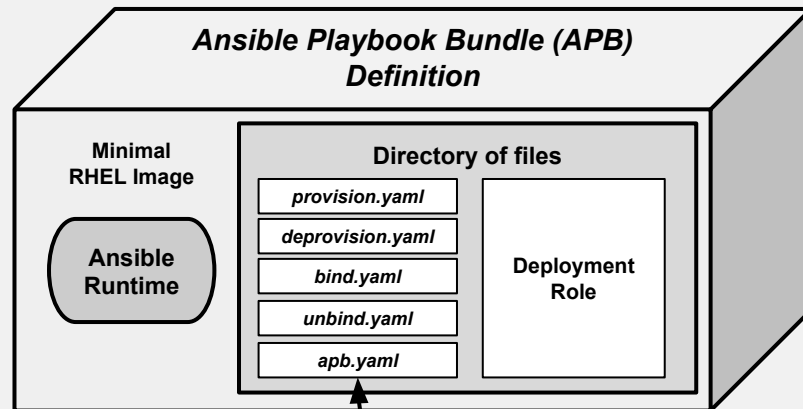
- Implementation of Open Service Broker API that enables users to leverage Ansible for provisioning and managing of services via the Service Catalog on OpenShift
- Standardized approach for delivering “simple” to “complex” multi-container OpenShift services via Ansible
- Works in conjunction with Ansible Playbook Bundles (APB), which is a lightweight meta container comprised of a few named playbooks for each Open Service Broker API operations

Not for production workloads
Using alpha version of kube service catalog

Feature(s): Ansible Playbook Bundles (APB)

Description:

- Short-lived, lightweight container image consisting of:
 - Simple directory structure with named “action” playbooks
 - Metadata consisting of:
 - required/optional parameters
 - dependencies (provision vs bind)
 - Ansible runtime environment
- Leverages existing investment in Ansible Playbooks & Roles
- Developer tooling available for guided approach
- Easily modified or extended
- Example APB services included with 3.6:
 - MediaWiki, PostgreSQL



provision.yaml = Install
deprovision.yaml = Uninstall
bind.yaml = Grant
unbind.yaml = Revoke
appb.yaml = Metadata

How it Works:

- When a user orders an application from the Service Catalog, the Ansible Service Broker will download the associated APB image from the registry and run it. Once the named operation has been performed on the service, the APB image will then terminate.

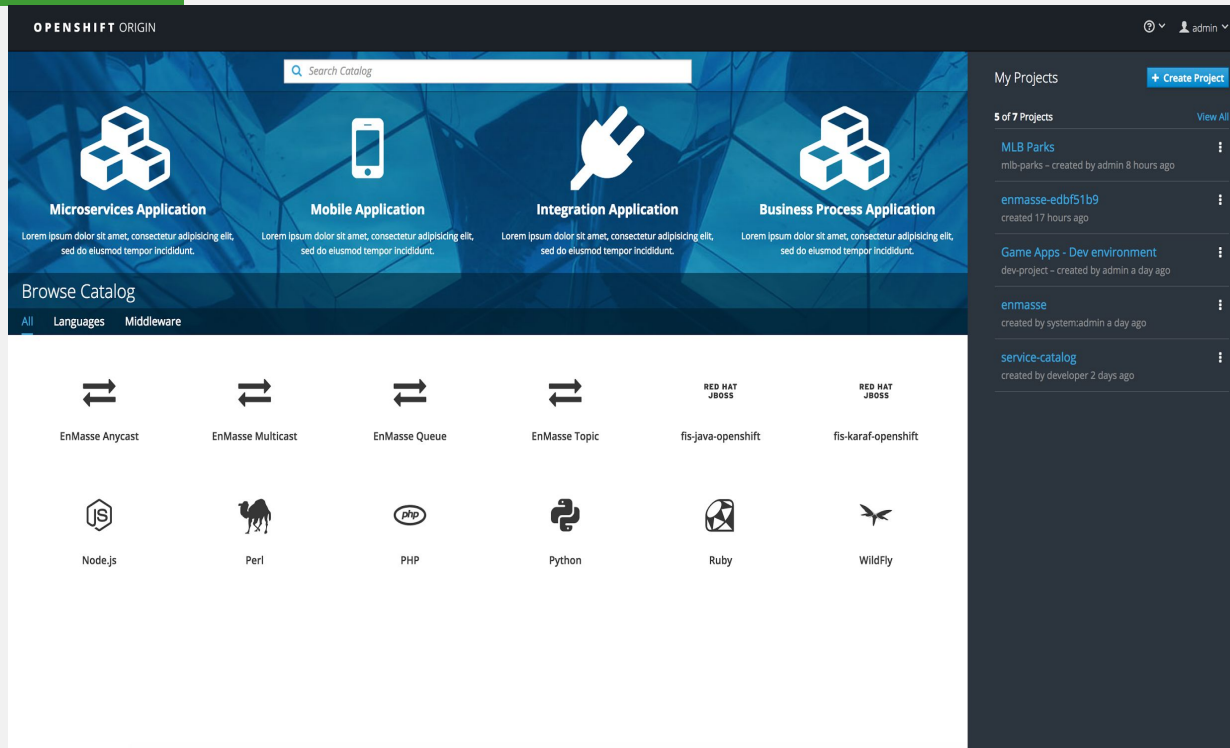
Self-Service / UX tech preview

Feature(s): [Initial experience](#)

Description: Build a better initial experience, motivated by service catalog

How it Works:

- Task focused interface
- Key call outs
- Unified search
- Streamlined navigation



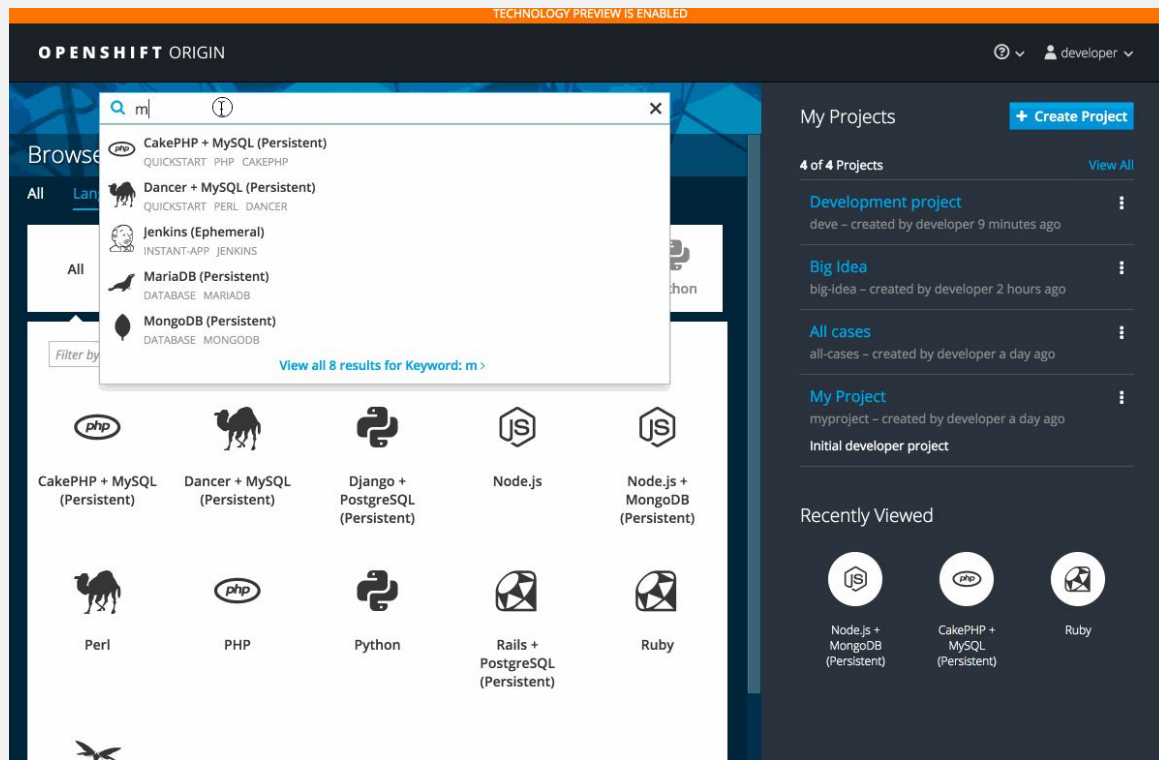
Self-Service / UX tech preview

Feature(s): [Search Catalog](#)

Description: Single simple way to quickly get what you want

How it Works:

- Search for catalog content
- Suggestions
- Based on category



Self-Service / UX tech preview

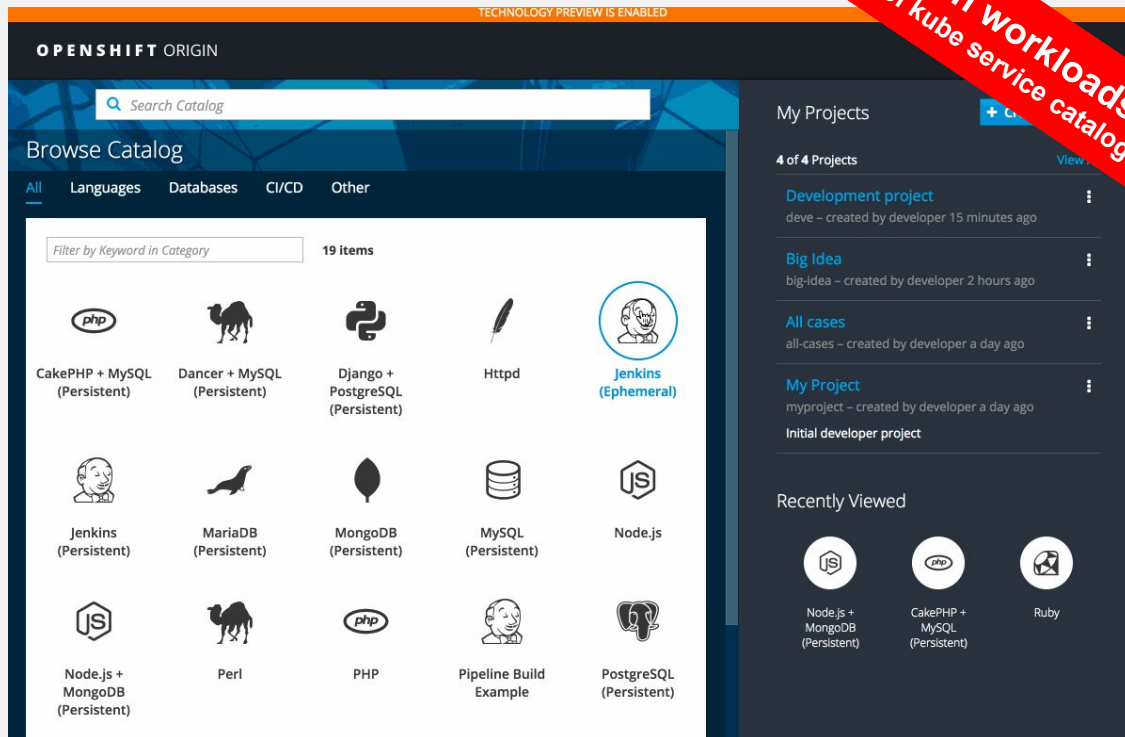
Not for production workloads
Using alpha version of kube service catalog

Feature(s): Add from Catalog

Description: Provision a service from the catalog

How it Works:

- Simply select desired service
- Follow prompts for desired project and configuration details



Self-Service / UX

Feature(s): [Project Overview Redesign](#)

Description: Implement feedback based on customer usage

How it Works:

- Provide 3 focused views: Applications, Pipelines, Resource Types
- More contextual actions
- Rolled up metrics across multiple pods

TECHNOLOGY PREVIEW IS ENABLED

Project All cases Add to Project developer

Overview

Applications

Builds

Resources

Storage

Monitoring

Other Resources

>	DEPLOYMENT cakephp-mysql-persistent, #9	1 pod	⋮
>	DEPLOYMENT jenkins, #1	1 pod	⋮
>	DEPLOYMENT mysql, #1	1 pod	⋮
>	REPLICATION CONTROLLER database-rc-1	0 pods	⋮
>	POD hello-openshift-serviced	1 pod	⋮
>	POD lonely-pod	1 pod	⋮
>	POD service-target-1	1 pod	⋮

Self-Service / UX tech preview

Feature(s): Add to Project

Description: Provision a service, without having to leave project overview

How it Works:

- Go directly to catalog from project, context is preserved
- Can directly provision, then bind

The screenshot displays the OpenShift console interface for a project named 'Development project'. At the top, a dark header bar contains the text 'TECHNOLOGY PREVIEW IS ENABLED'. Below this, the project name 'Development project' is shown with a dropdown arrow and an 'Add to Project' button. A search bar with 'Name' and 'Filter by name' is visible, along with a 'List by' dropdown set to 'Application'. The main content area is divided into two sections: 'Other Resources' and 'Provisioned Services'. Under 'Other Resources', there is a deployment named 'mariadb, #1' with a status of '1 pod'. Under 'Provisioned Services', there is a service named 'MariaDB (Persistent)' with the identifier 'mariadb-persistent-x1sm8' and a 'Create Binding' button. A left-hand navigation sidebar is visible, showing options like Overview, Applications, Builds, Resources, Storage, and Monitoring.

Self-Service / UX tech preview

Feature(s): [Bind in context](#)

Description: Provision a service and bind, without having to leave project overview

How it Works:

- Select deployment and initiate a bind
- Select from bindable services
- Binding is created
- User stays in context

TECHNOLOGY PREVIEW IS ENABLED

Project: Development project ▼ Add to Project ▼ Developer ▼

Overview

Applications >

Builds >

Resources >

Storage

Monitoring

APPLICATION my-app

REPLICATION CONTROLLER database 1 pod

REPLICATION CONTROLLER frontend 2 pods

APPLICATION node <http://node-development.127.0.0.1.nip.io>

DEPLOYMENT node, #1 1 pod

Other Resources

DEPLOYMENT mariadb, #1 1 pod

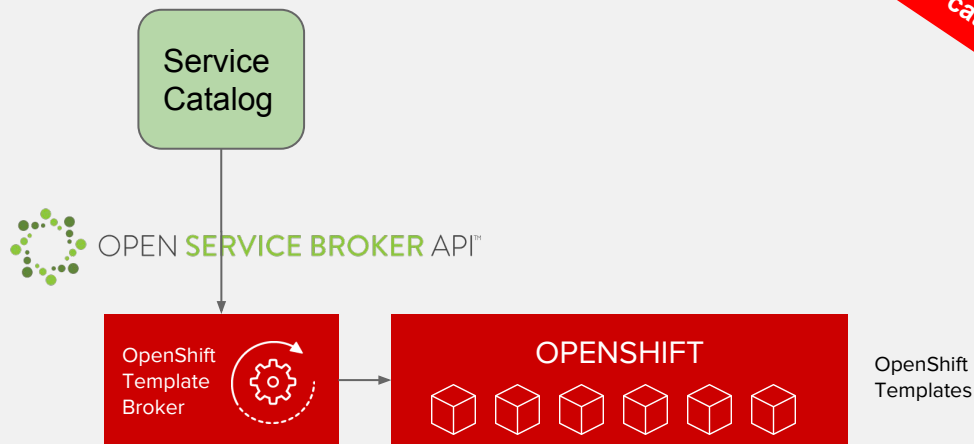
DEPLOYMENT mysql, #1 1 pod

Feature(s): Template Service Broker

Description: Expose OpenShift templates through a OpenServiceBrokerAPI to the Service Catalog

How it Works:

- The Template Broker matches the lifecycles of provision, deprovision, bind, unbind with existing templates
- No changed required to templates, unless you expose bind
- Your application will get injected with config details (bind)



Not for production workloads
Using alpha version of kube service catalog

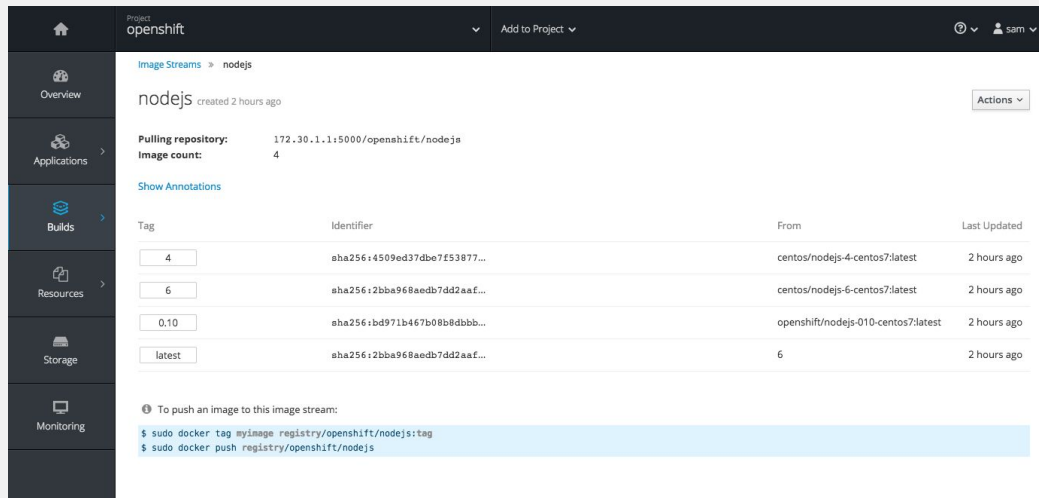
Self-Service / UX

Feature(s): Image stream details

Description: Provide additional details about image streams and their tags

How it Works:

- Leverage Cockpit views for image streams
- Details tags and details about each



The screenshot displays the OpenShift web console interface for an image stream named 'nodejs'. The breadcrumb navigation shows 'Image Streams > nodejs'. The page header includes 'Project openshift' and 'Add to Project'. The main content area shows the image stream 'nodejs' created 2 hours ago, with a 'Pulling repository' of '172.30.1.1:5000/openshift/nodejs' and an 'Image count' of 4. A 'Show Annotations' link is present. Below this is a table with columns for Tag, Identifier, From, and Last Updated. The table lists four tags: '4', '6', '0.10', and 'latest', each with its corresponding SHA256 identifier, source repository, and update time. At the bottom, a terminal snippet shows the commands to push an image to the stream: '\$ sudo docker tag myimage registry/openshift/nodejs:tag' and '\$ sudo docker push registry/openshift/nodejs'.

Tag	Identifier	From	Last Updated
4	sha256:4509ed37dbe7f53877...	centos/nodejs-4-centos7/latest	2 hours ago
6	sha256:2bba968aeb7dd2aa.f...	centos/nodejs-6-centos7/latest	2 hours ago
0.10	sha256:bd971b467b08b8dbbb...	openshift/nodejs-010-centos7/latest	2 hours ago
latest	sha256:2bba968aeb7dd2aa.f...	6	2 hours ago

Self-Service / CLI tech preview

Feature(s): Service Catalog

Description: Bring the Service Catalog experience to the CLI: provision and bind

How it Works:

- Use opportunity to provide extension/plugin to kubectl
- Provision, bind, provision+bind, deprovision, list, etc

```
# Provision an instance of the "mysql" service in the "mine"
namespace with the "2cores" plan
$ kubectl catalog provision mysql --plan=2cores --namespace=mine
```

```
# Bind the existing service instance "mysql" to the (Kube) service
"frontend"
$ kubectl catalog bind mysql --to=svc/frontend --namespace=mine
```

Not for production workloads
Using alpha version of kube service catalog

Self-Service / CLI

Feature(s): Better messages for syntax errors in JSON/YAML

Description: Provide better messages, such as details of the syntax problem and line number

How it Works:

- Validate input on commands such as “oc create -f foo.json” and “oc new-app -f template.yaml”

```
$oc create -f dc.json  
error: json: line 27: invalid character 'y' looking  
for beginning of value
```

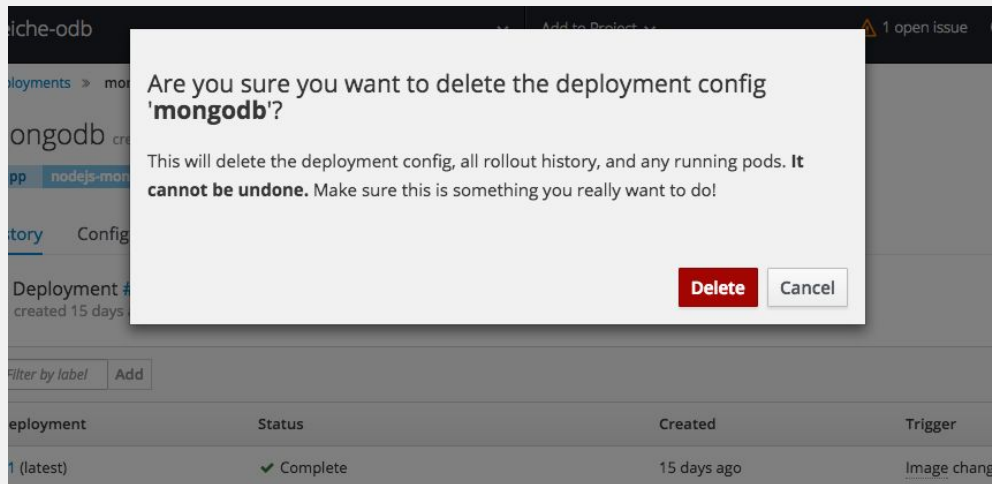
Self-Service / UX

Feature(s): Cascading deletes

Description: When a delete a resource, I want to make sure all generated/dependent resources are also deleted

How it Works:

- For example, selecting a deployment config and deleting will delete the DC, deployment history and any running pods



DevExp / Builds

Feature(s): [Automatic Build Pruning](#)

Description: Previous only 'oc adm prune' could be used, now user can define on a build config

How it Works:

- Define how much build history you want to keep per build config
- You can set successful vs failed separately



DevExp / Builds

Feature(s): [Easier Custom Slave Configuration for Jenkins](#)

Description: Make changes to Jenkins config does not require new image layers

How it Works:

- Slaves are defined as image-streams with appropriate label
- Pod template can be specified as ConfigMap with appropriate label



DevExp / Builds

Feature(s): Detailed Build Timing

Description: Builds now record timing information based on more granular steps.

How it Works:

- Information such as how long it took to pull the base image, clone the source, build the source, and push the image

```
$ oc describe build nodejs-ex-1
Name:          nodejs-ex-1
Namespace:     myproject
Created:       2 minutes ago

Status:        Complete
Started:       Fri, 07 Jul 2017 17:49:37 EDT
Duration:      2m23s
  FetchInputs:    2s
  CommitContainer: 6s
  Assemble:       36s
  PostCommit:     0s
  PushImage:      1m0s
```

- [Webhooks for GitLab and Bitbucket](#)
- httpd 2.4 s2i support (static content)
- Separate build events for: start, cancelled, success, fail
- [Support for ARGs in Dockerfiles](#)
- [Env vars in Pipeline builds](#)
- Creds for Jenkins Sync plugin for ease of working external Jenkins instance
- [ValueFrom Support in Build Environment Variables](#)
- Deprecated Jenkins v1 image

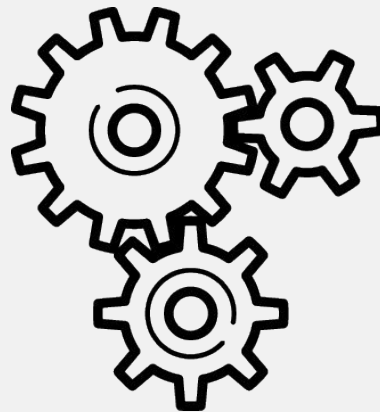
Dev Tools - Local Dev

CDK 3.1 / minishift 1.3.0:

- Caching of OpenShift images
- Update command
- Profile parameter
- Interactively select OpenShift version

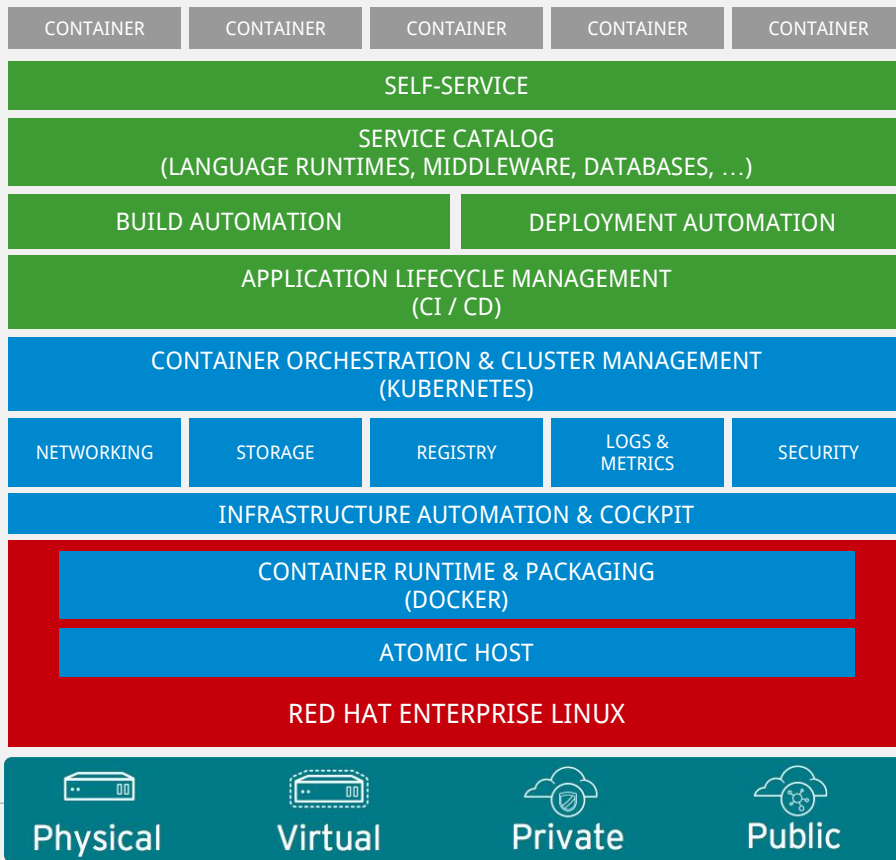
oc cluster up:

- Support launching service catalog
- Switch to nip.io from xip.io, improved stability



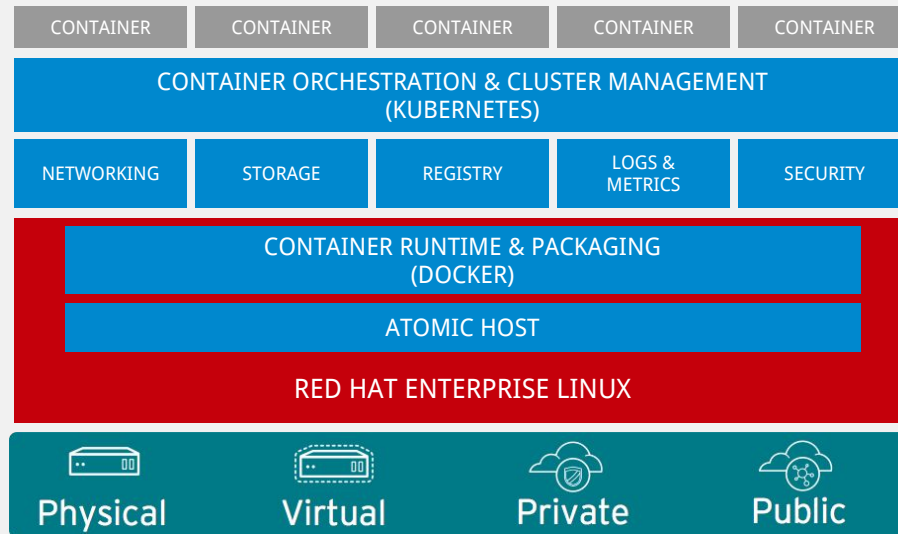
OpenShift = Enterprise Kubernetes+

Build, Deploy and Manage Containerized Apps



Clustered Container Infrastructure

Applications Run Across Multiple Containers & Hosts



Container Orchestration

Feature(s): Kubernetes Upstream

Description: Many core features Google announced in March for Kubernetes 1.6 were the result of OpenShift engineering. Red Hat continues to influence the product in the areas of Storage, Networking, Resource Management, Authentication&Authorization, Multi-tenancy, Security, Service Deployments and templating, and Controller functionality.

Contributing Projects:

- RBAC
- Pod QoS
- Env from Secrets/ConfigMap
- Service Broker
- Federation
- External Provisioners
- StorageClasses
- Init Containers
- FlexVolume
- StatefulSets
- ReplicaSet Termination Handling

Container Orchestration

Feature(s): [OpenShift now uses the CRI interface for kublet to docker interaction](#)

Description: As the container space matures and choices become more available, OpenShift needs an agnostic interface in Kubernetes for container runtime interactions. OCP 3.6 switches the default config to use the Kubernetes docker CRI interface.

How it Works: There is a enable-cri setting in the node-config.yaml configuration file. A value of true will enable the use of the interface. Change it by editing the file and stopping/starting the atomic-openshift-node.service

```
$ cat /etc/origin/node/node-config.yaml  
enable-cri:
```

```
- 'true'
```

Although the docker CRI is stable and the default, the overall CRI interface in Kubernetes is still under development. Red Hat does not support cri-o, rkt, or frakti in this OCP 3.6 release.

Container Orchestration

Feature(s): [cluster-capacity utility for checking true allocatable space.](#)

Description: Just like a disk drive, a cluster can become fragmented over time. When you ask the cluster how much space is left, the addition of all the free space doesn't indicate how many actual workloads can run. For example, it might say there is 10gb left, but it could be that no single node can take more than 512mb.

How it Works: [We created a new container that you can launch as a cmdline or a Job.](#) The container allows you to supply a popular workload (image) with a commonly requested CPU and MEM limit and request. The logs from the container will tell you how many of that workload can be deployed.

oc log output:

small-pod pod requirements:

- CPU: 150m
- Memory: 100Mi

The cluster can schedule 52 instance(s) of the pod small-pod.

Termination reason: Unschedulable: No nodes are available that match all of the following predicates:: Insufficient cpu (2).

Pod distribution among nodes:

small-pod

- 192.168.124.214: 26 instance(s)
- 192.168.124.120: 26 instance(s)

Container Orchestration

Feature(s): [Quota on how much \(size and type\) remote storage a project can use.](#)

Description: [You can now control](#) what classes of storage projects are allowed to access, how much (total size) of that class, as well as how many claims.

How it Works: This feature leverages the ResourceQuota object and allows you to call out storage classes by name for size and claim settings.

```
$ oc create quota my-quota-1
--hard=slow.storageclass.storage.k8s.io/requests.storage=20Gi,slow.stor
ageclass.storage.k8s.io/persistentvolumeclaims=15
```

```
$ oc describe quota my-quota-1
```

Name:	my-quota-1	
Namespace:	default	
Resource	Used	Hard
-----	----	---
slow .storageclass.storage.k8s.io/persistentvolumeclaims	0	15
slow .storageclass.storage.k8s.io/requests.storage	0	20Gi

Container Orchestration

Feature(s): [Project configMaps, secrets, and downward API into the same directory](#)

Description: When you mount a memory backed volume into a container, it leverages a directory. Now you can place all sources of configuration for your application (configMaps, secrets, and downward API) into the same directory path.

How it Works: New projected line in the volume definition allows you to tell multiple volumes to leverage the same mount point while guarding for path collisions.

volumes:

- name: all-in-one

projected:

sources:

- secret:

name: test-secret

items:

- key: data-1

path: mysecret/my-username

- key: data-2

path: mysecret/my-passwd

- downwardAPI:

items:

- path: mydapi/labels

fieldRef:

fieldPath: metadata.labels

- path: mydapi/name

fieldRef:

fieldPath: metadata.name

- path: mydapi/cpu_limit

resourceFieldRef:

containerName: allinone-normal

resource: limits.cpu

divisor: "1m"

- configMap:

name: special-config

items:

- key: special.how

path: myconfigmap/shared-config

- key: special.type

path: myconfigmap/private-config

Container Orchestration

Feature(s): [Init Containers](#)

Description: Init containers provide a chained, pre-deployment mechanism for running arbitrary containers/programs as part of getting ready for your application to run. They aid in application deployment (pre-check, pre-start coordination, data population, etc) and, to some degree, dependency management.

How it Works: Init containers run to completion and each container must finish before the next one starts, and will run for each pod instance. The init containers will honor the restart policy. Leverage initContainers in the podspec.

```
$ cat init-containers.yaml
```

```
apiVersion: v1
```

```
kind: Pod
```

```
metadata:
```

```
  name: init-loop
```

```
spec:
```

```
  containers:
```

```
    - name: nginx
```

```
      image: nginx
```

```
      ports:
```

```
        - containerPort: 80
```

```
      volumeMounts:
```

```
        - name: workdir
```

```
          mountPath: /usr/share/nginx/html
```

```
  initContainers:
```

```
    - name: init
```

```
      image: centos:centos7
```

```
      command:
```

```
        - /bin/bash
```

```
        - "-c"
```

```
        - "while ;; do sleep 2; echo hello init container; done"
```

```
  volumes:
```

```
    - name: workdir
```

```
      emptyDir: {}
```

```
$ oc get -f init-containers.yaml
```

```
NAME    READY   STATUS    RESTARTS   AGE
nginx   0/1     Init:0/1  0           6m
```

Container Orchestration

Feature(s): [Multiple schedulers at the same time.](#)

Description: Kubernetes now supports extending the default scheduler implementation with custom schedulers.

How it Works: After [configuring and deploying](#) your new scheduler, you can call it by name from the podspec via schedulerName. These new schedulers are packaged into container images and run as pods inside the cluster.

```
cat pod-custom-scheduler.yaml
apiVersion: v1
kind: Pod
metadata:
  name: custom-scheduler
spec:
  schedulerName:
    custom-scheduler
containers:
- name: hello
  image:
    docker.io/ocpqe/hello-pod
```

Container Orchestration

Feature(s): Turn configMap content into environment variables within the container.

Description: Instead of individually declaring environment variables in a Pod definition, a configmap can be “imported” and all of its content can be dynamically turned into environment variables.

How it Works: In the pod spec leverage the **envFrom** object and reference the desired configMap.

```
env:  
- name: duplicate_key  
  value: FROM_ENV  
- name: expansion  
  value: $(REPLACE_ME)  
envFrom:  
- configMapRef:  
  name: env-config
```

Container Orchestration

Feature(s): [Node Affinity and Anti-Affinity](#)

Description: Control which nodes your workload will land on in a more generic and powerful way as compared to nodeSelector.

How it Works: [You can select the label value](#) you would like the operator to compare against (eg: In, NotIn, Exists, DoesNotExist, Gt, and Lt). You can choose to make satisfying the operator required or preferred. Preferred means search for the match, but, if you can't find one, ignore it.

```
affinity:
  nodeAffinity:
    preferredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
        - matchExpressions:
            - key: "failure-domain.beta.kubernetes.io/zone"
              operator: In
              values: ["us-central1-a"]
```

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
        - matchExpressions:
            - key: "failure-domain.beta.kubernetes.io/zone"
              operator: NotIn
              values: ["us-central1-a"]
```

Container Orchestration

Feature(s): [Pod Affinity and Anti-Affinity](#)

Description: Let us say you want to allow kube the freedom to select which zone an application lands in, but whichever it chooses you'd like to make sure another component of that app lands in the same zone. Or let's say you have two app components that, due to security reasons, cannot be on the same physical box. But you don't want to lock them into labels on nodes. You want them to land anywhere, but still honor your anti-affinity desire.

How it Works: Many of the same high level concepts mentioned in the node affinity and anti-affinity hold true here. [For pods, you declare a topologyKey](#) which will be used as the boundary object for the placement logic.

```
affinity:
  podAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      - labelSelector:
          matchExpressions:
            - key: service
              operator: In
              values: ["S1"]
        topologyKey: failure-domain.beta.kubernetes.io/zone
```

```
affinity:
  podAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      - labelSelector:
          matchExpressions:
            - key: service
              operator: In
              values: ["S1"]
        topologyKey: kubernetes.io/hostname
```


Container Orchestration

Feature(s): [Taints and Tolerations](#)

Description: Allows you to taint a node so that pods without a toleration for the same taint can't use it. Taints are placed on the nodes or resources and tolerations are placed on the pods or workloads looking for resources.

How it Works: Taints can be keys, values, and effects. These are matched and then executed via an logic operator declared in the toleration. Effects can be NoSchedule, PreferNoSchedule, NoExecute. NoExecute means to evict the pods that are already running on the node before the taint was applied that no longer match.

Set the taint from the command line:

```
$ oc taint nodes node1 key=value:NoSchedule
```

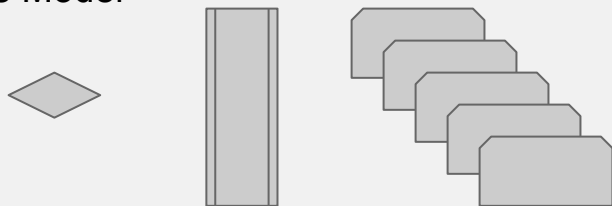
Set toleration in the podspec:

```
tolerations:  
- key: "key"  
  operator: "Equal"  
  value: "value"  
  effect: "NoSchedule"
```

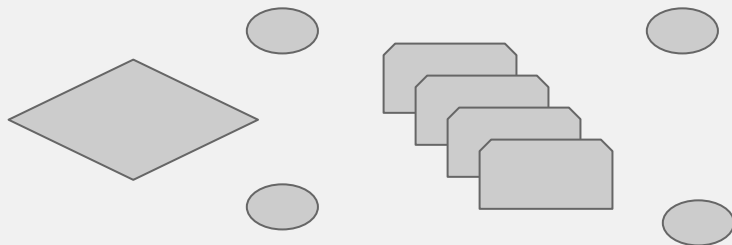
Container Orchestration

Feature(s): Federation Decision Deliberation

Current Control Plane Model



Potential Client Model



Registry

Feature(s): [Validating Image Signatures show appropriate metadata](#)

Description: When I'm working with image signatures as the image-admin role, I need to know the status of the images in terms of their signatures

How it Works: We now offer an 'oc adm verify-image-signature' command that will allow us to save or remove signatures. The resulting 'oc describe istag' will display additional metadata about the signature's status.

```
$ oc describe istag origin-pod:latest
```

```
Image Signatures:
```

```
  Name:
```

```
sha256:c13060b74c0348577cbe07dedcdb698f7d893ea6f74847154e5ef3c8c9369b2c@f66d720cfaced1b33e8141a844e793be
```

```
  Type: atomic
```

```
  Status: Unverified
```

```
# Verify the image and save the result back to image stream
```

```
$ oadm verify-image-signature
```

```
sha256:c13060b74c0348577cbe07dedcdb698f7d893ea6f74847154e5ef3c8c9369b2c \
```

```
--expected-identity=172.30.204.70:5000/test/origin-pod:latest --save
```

```
--as=system:admin
```

```
sha256:c13060b74c0348577cbe07dedcdb698f7d893ea6f74847154e5ef3c8c9369b2c signature 0 is verified (signed by key: "172B61E538AAC0EE")
```

```
# Check the image status
```

```
$ oc describe istag origin-pod:latest
```

```
Image Signatures:
```

```
  Name:
```

```
sha256:c13060b74c0348577cbe07dedcdb698f7d893ea6f74847154e5ef3c8c9369b2c@f66d720cfaced1b33e8141a844e793be
```

```
  Type: atomic
```

```
  Status: Verified
```

```
  Issued By: 172B61E538AAC0EE
```

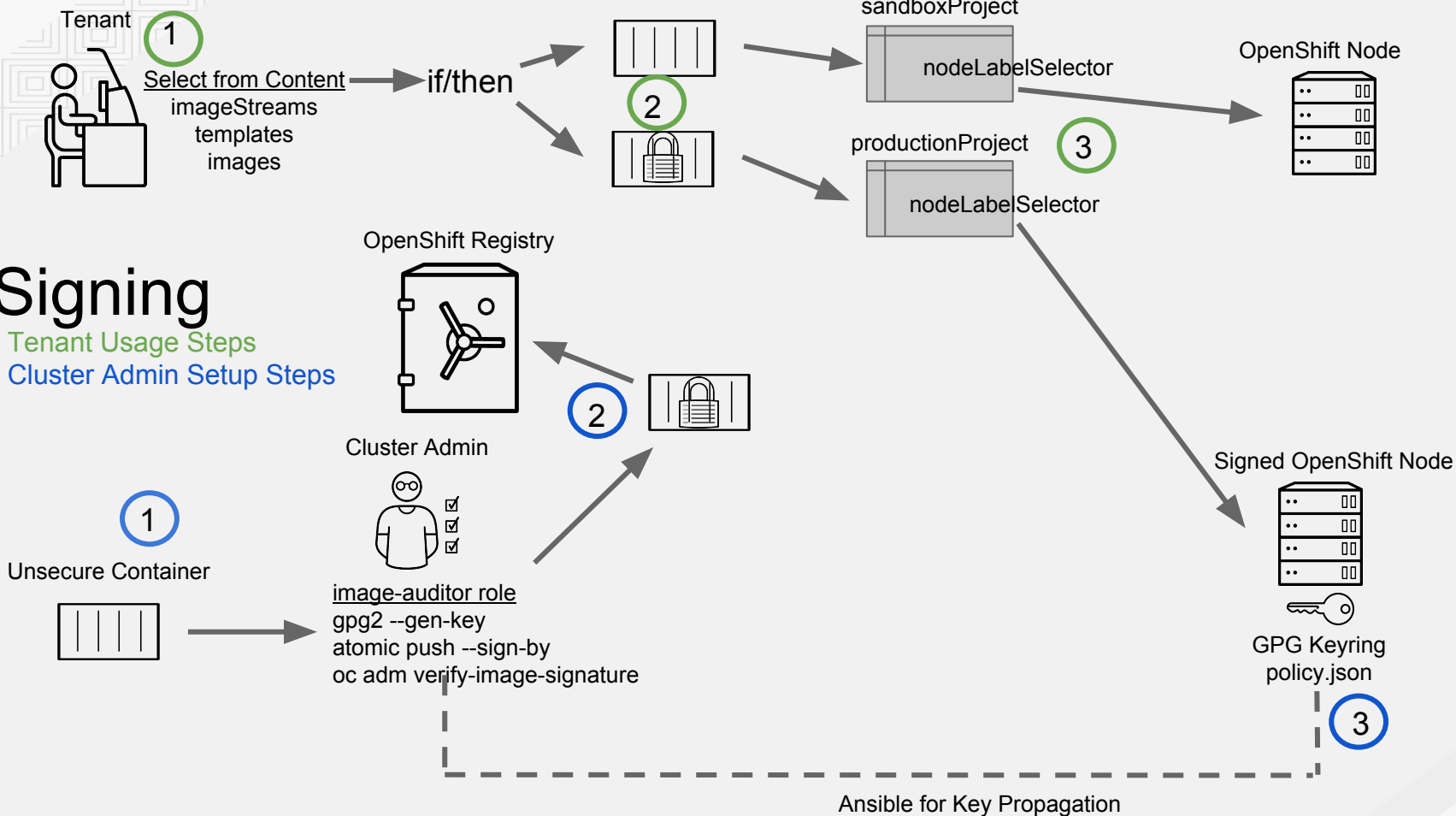
```
  Signature is Trusted (verified by user "system:admin" on 2017-04-28 12:32:25 +0200 CEST)
```

```
  Signature is ForImage ( on 2017-04-28 12:32:25 +0200 CEST)
```

Signing

Tenant Usage Steps

Cluster Admin Setup Steps



Registry

Feature(s): [Registry REST endpoint for reading and writing image signatures](#)

Description: Need a programmable way to read and write signatures using only the docker registry API.

How it Works: To read the user must be authenticated to the registry and to write they must have the image-signer role.

READ:

```
PUT /extensions/v2/{namespace}/{name}/signatures/{digest}
$ curl
http://<user>:<token>@<registry-endpoint>:5000/extensions/v2/<namespace>/<name>/signatures/sha256:<digest>
```

JSON:

```
{
  "version": 2,
  "type": "atomic",
  "name":
"sha256:4028782c08eae4a8c9a28bf661c0a8d1c2fc8e19dbaee2b018b21011
197e1484@cddeb7006d914716e2728000746a0b23",
  "content": "<base64 encoded signature>",
}
```

WRITE:

```
GET /extensions/v2/{namespace}/{name}/signatures/{digest}
$ curl
http://<user>:<token>@<registry-endpoint>:5000/extensions/v2/<namespace>/<name>/signatures/sha256:<digest>
```

```
{
  "signatures": [
    {
      "version": 2,
      "type": "atomic",
      "name":
"sha256:4028782c08eae4a8c9a28bf661c0a8d1c2fc8e19dbaee2b018b21011
197e1484@cddeb7006d914716e2728000746a0b23",
      "content": "<base64 encoded signature>",
    }
  ]
}
```

Installation

tech
preview

Feature(s): [Automated installation of CloudForms 4.5 inside OpenShift](#)

Description: The installation of containerized CloudForms inside OpenShift is now part of the main installer. It is being treated like other common components (metrics, logging, etc).

How it Works: After the OpenShift cluster is provisioned, there is an additional playbook that can be run to deploy CloudForms into the environment (using the `openshift_cfme_install_app` flag in the hosts file)

```
$ ansible-playbook -v -i <INVENTORY_FILE>  
playbooks/byo/openshift-cfme/config.yml
```

Requirements:

Type	Size	CPUs	Memory
Masters	1+	8	12GB
Nodes	2+	4	8GB
PV Storage	25GB	N/A	N/A

NFS is the only storage option for the Postgres DB at this time.

Installation

tech
preview

Feature(s): Installation of [etcd](#), [docker daemon](#), and [ansible installer](#) as System Containers.

Description: RHEL System Containers offer more control over the life cycle of the services that do not run inside OpenShift/Kubernetes. Additional system containers will be offered over time.

How it Works: System Containers leverage the OSTree on RHEL or Atomic Host. They are controlled by the kernel init system and therefore can be leveraged earlier in the boot sequence. This feature is enabled in the installer configuration.

```
$ atomic install --system --set  
INVENTORY_FILE=$(pwd)/inventory  
registry:port/openshift3/ose-ansible:v3.6
```

```
$ systemctl start ose-ansible-v3.6
```

Must be used in a non-production sandbox.

Not for production workloads

Installation

Feature(s): [etcd3 data model for fresh installs](#)

Description: Starting with fresh installs of OCP 3.6, we will be defaulting to the etcd3 v3 data model. We will also release a migration script to allow people who upgraded to OCP 3.6 to migrate to the etcd3 v3 data model. Then, in OCP 3.7, we will stop the upgrade if the installer detects data is not migrated to the etcd3 data model. This will be a mandatory and manual step.

How it Works: This is controlled by the `openshift_master_disable_etcdv3=true` setting

Problems Solved by Moving to the v3 data model:

- Larger in memory space to enable larger cluster sizes
- Increased stability in adding and removing nodes in general life cycle actions
- Significant performance boost

Installation

Feature(s): [Cluster wide control of CA expiry.](#)

Description: You now have the ability to change the certificate expiration date en mass across the cluster for the various framework components that use TLS.

How it Works: We offer new cluster variables per framework area so that you can use different timeframes for different framework components. Once set, you simply issue the new redeploy-openshift-ca playbook.

New Cluster Variables

```
# CA, node and master certificate expiry
openshift_ca_cert_expire_days=1825
openshift_node_cert_expire_days=730
openshift_master_cert_expire_days=730
```

```
# Registry certificate expiry
openshift_hosted_registry_cert_expire_days=730
```

```
# Etcd CA, peer, server and client certificate expiry
etcd_ca_default_days=1825
```

```
$ ansible-playbook -i inventory
openshift-ansible/playbook/byo/openshift-cluster/red
eploy-openshift-ca.yml
```

Installation

Feature(s): General Stability

Description: Final stage of Online Operations and customer shipping code consolidation project. Idempotency refactoring.

How it Works: Stability release for consolidated code base.

Final Steps Achieved:

- [Upgrade 3.5 → 3.6](#)
- [Idempotency refactoring of configuration role](#)
- [Swap handling during installation](#)
- [All byo playbooks pull from a normalized group source](#)
- [Final port of operation's ansible modules](#)
- [Refactoring of excluder roles](#)

Storage

Feature(s): [AWS EFS Provisioner](#)

Description: Allows you to dynamically use the AWS EFS end point to get NFS remote persistent volumes on AWS.

How it Works: [Leverages the external dynamic provisioner interface](#). It is provided as a [docker image](#) that you configure with a configMap and deploy on OpenShift. You then use a storageClass with the appropriate configuration.

Storage Class Example:

```
apiVersion: storage.k8s.io/v1beta1
kind: StorageClass
metadata:
  name: slow
provisioner: foobar.io/aws-efs
parameters:
  gidMin: "40000"
  gidMax: "50000"
```

gidMin + gidMax : The minimum and maximum value of GID range for the storage class. A unique value (GID) in this range (gidMin-gidMax) will be used for dynamically provisioned volumes

Storage

Feature(s): [VMWare vSphere Storage](#)

Description: Allows you to dynamically use the VMWare vSphere storage options ranging from VSANDatastore, ext3, vmdk, and VSAN while honoring vSphere Storage Policy (SPBM) mappings.

How it Works: Leverages the [cloud provider interface](#) in Kubernetes to trigger this in tree dynamic storage provisioner. Once the cloud provider has the correct credential information, tenants can [leverage storageClass to select the desired storage](#).

Storage Class Example:

```
kind: StorageClass
apiVersion: storage.k8s.io/v1beta1
metadata:
  name: fast
provisioner: kubernetes.io/vsphere-volume
parameters:
  diskformat: zeroedthick
  storagePolicyName: gold
  fstype: ext3
```

Storage

Feature(s): [iSCSI CHAP](#) and [mount options](#)

Description: You can now use CHAP authentication for your iSCSI remote persistent volumes. Also, you can annotate your PVs to leverage any mount options that are supported by that underlying storage technology.

How it Works: The tenant supplies the correct username and password for the CHAP authentication as a secret in their podspec. For mount options, [you supply the annotation in the PV.](#)

volumes:

- name: iscsivol

iscsi:

targetPortal: 127.0.0.1

iqn: iqn.2015-02.example.com:test

lun: 0

fsType: ext4

readOnly: true

chapAuthDiscovery: true

chapAuthSession: true

secretRef:

name: chap-secret

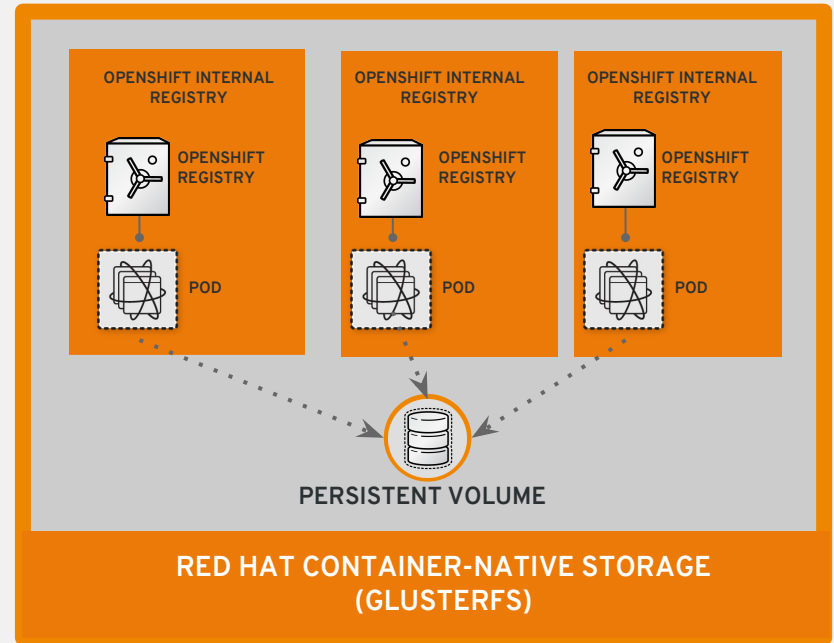
```
set 'volume.beta.kubernetes.io/mount-options' to  
'volume.beta.kubernetes.io/mount-options: rw,nfsvers=4,noexec'
```

Storage

Feature(s): [Improved and fully automated support for CNS backed OCP hosted registry](#)

Description: The installer is now capable of provisioning the OpenShift registry backed with Container Native Storage (Gluster). This enables user to take advantage of the globally available storage capacity, strong read/write consistency, three way replica, and RHGS data management features.

How it Works: The feature is provided through integrations in the [OCP advanced installation](#). A simple change to the inventory file is all that's required.



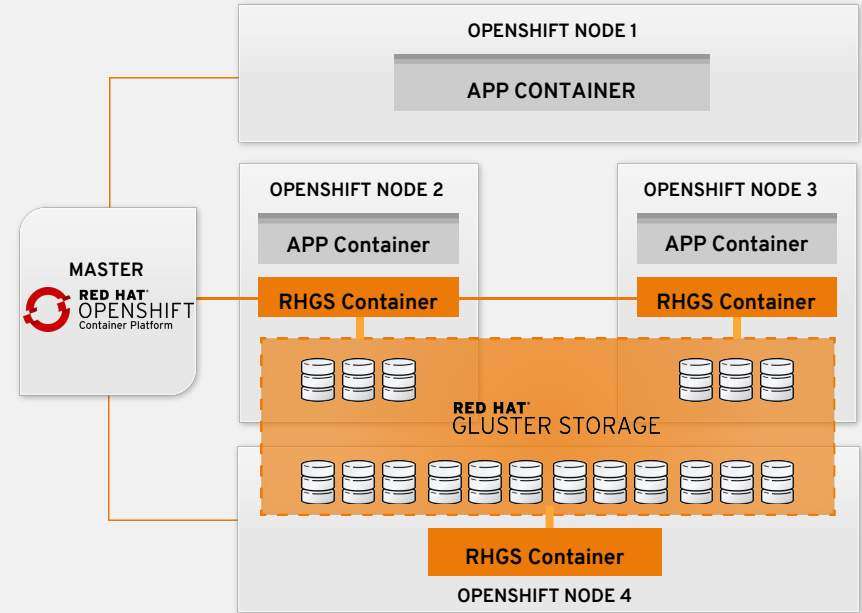
- **Highly Available and Scalable**
- **Infrastructure/cloud Provider Agnostic**
- **Three Replica**
- **RHGS Data management & Data Protection**

Storage

Feature(s): [Automated Container Native Storage \(CNS\) deployment with OCP Advanced Installation](#)

Description: The advanced installer now includes automated & integrated support for deployment of CNS, correctly configured and highly available out-of-the-box.

How it Works: CNS storage device details are added to the installer's inventory file (examples provided in OCP advanced installation documentation). The installer manages configuration and deployment of CNS, its dynamic provisioner, and other pertinent details.



- **Co-Locate Storage and Apps**
- **Storage managed by OpenShift**
- **RHGS Data management & Data protection**
- **Improved out-of-the-box user experience**

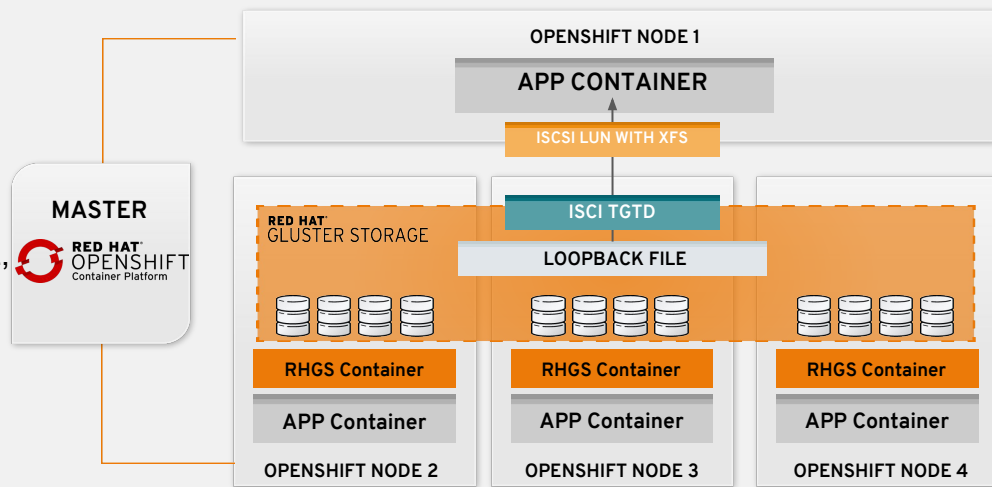
Storage

Feature(s): [OCP 3.6 + CNS 3.6 \(Sep 2017\)](#)

Description: CNS 3.6 (Sep 2017) with OCP 3.6 will support block, higher volume density & S3 object store

How it Works:

- RWO volumes backed by Gluster-iSCSI block storage provide better performance for MySQL, PostgreSQL, etc. and targeting support for Elastic Search (OCP Logging)
- 3x volume density per cluster (1000+) with lower memory footprint of RHGS (Brick-mux)
- S3 object service for OCP (based on swift-on-file, *Tech Preview*)



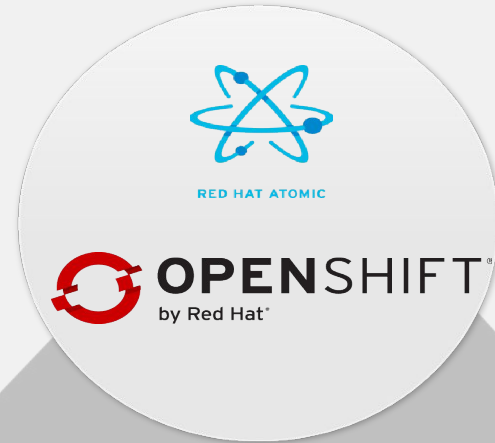
Storage

Feature(s): OCP Commercial Evaluation subscription includes CNS/CRS

Description: OpenShift Commercial Eval subscription will include Container Native Storage/Container Ready Storage solutions & eval subscription

How it Works: OpenShift evaluation subscription SKU will bundle the CNS/CRS bits as well as additional entitlements to evaluate OpenShift with CNS/CRS. T&C of CNS/CRS subscription aligns with OCP Eval subscription

Note- A new add on CNS/CRS SKU for OpenShift will be available soon.



CNS/CRS based on RHGS

- RUNS ON TOP OF OPENSIFT
- AVAILABLE OUT OF THE BOX 3.6
- RUNS EVERYWHERE
- SCALABLE , RWX SUPPORT
- GEO-REPLICATION, SNAPSHOTS

Networking

Feature(s): [Multiple destinations in egress-router](#)

Description:

- Ability to connect to multiple destinations from a project without needing to reserve a separate source IP for each of them
- Optional “fallback” IP (all other ports to this), otherwise reject outliers
- Old syntax continues to behave the same
- No change to EGRESS_SOURCE and EGRESS_GATEWAY definitions

How it Works (snippets of YAML):


OLD WAY

```
- name: EGRESS_DESTINATION
  value: 203.0.113.25
```

NEW WAY

```
- name: EGRESS_DESTINATION
  value: |
    80 tcp 1.2.3.4
    8080 tcp 5.6.7.8 80
    8443 tcp 9.10.11.12 443
    13.14.15.16
```

```
localport udp|tcp dest-ip [dest-port]
```



Networking

Feature(s): [Add an HTTP Proxy Mode for Egress Router](#)

Description:

- TLS connections (certificate validations) don't easily work because the client needs to connect to the egress router's IP (or name) rather than to the destination server's IP/name
- Egress router now can be run as a (squid) proxy rather than just redirecting packets
- `EGRESS_HTTP_PROXY_DESTINATION` rules:
 - Acted upon, in order
 - "!" prefix = deny proxying to this destination
 - If "*" on the last line then anything that hasn't been denied will be allowed to proxy, otherwise denied
- `https://www.redhat.com` → HTTP/1.1 403 Forbidden
- `http://www.google.com` → HTTP/1.1 200 OK
- `https://www.google.com` →
HTTP/1.1 200 Connection established
HTTP/1.1 200 OK

```
...
spec:
  initContainers:
  - name: egress-router-setup
    imagePullPolicy: IfNotPresent
    image: openshift3/ose-egress-router
    securityContext:
      privileged: true
    env:
    - name: EGRESS_SOURCE
      value: 172.17.0.9
    - name: EGRESS_GATEWAY
      value: 172.17.0.1
    - name: EGRESS_ROUTER_MODE
      value: http-proxy
  containers:
  - name: egress-router-proxy
    imagePullPolicy: IfNotPresent
    image: openshift3/ose-egress-http-proxy
    env:
    - name: EGRESS_HTTP_PROXY_DESTINATION
      value: |
        !*.redhat.com
        *
```

Networking




Feature(s): [Use DNS Names with Egress Firewall](#)

Description:

Benefits of using DNS names vs IP addresses:

- “Tracks” DNS mapping changes
- Humanized / Easily remembered naming
- Potentially backed by multiple IP addresses

How it Works:

- Create project and pod
- Deploy egress network policy w/ DNS names
- Validate firewall
 - nslookup stopdisablingsexlinux.com
 - 104.24.114.57, 104.24.115.57
 - ping 104.24.114.57 
 - ping 104.24.115.57 
 - ping yahoo.com 

Egress Policy Example:

```
{
  "kind": "EgressNetworkPolicy",
  "apiVersion": "v1",
  "metadata": {
    "name": "policy-test"
  },
  "spec": {
    "egress": [
      {
        "type": "Allow",
        "to": {
          "dnsName": "stopdisablingsexlinux.com"
        }
      },
      {
        "type": "Deny",
        "to": {
          "cidrSelector": "0.0.0.0/0"
        }
      }
    ]
  }
}
```

Networking

Tech
Preview

Feature(s): [Network Policy](#)

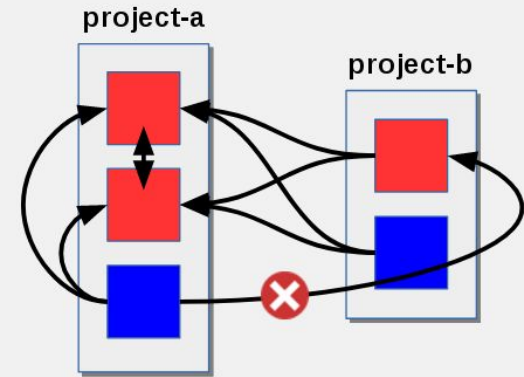
Description: Optional plugin specification of how selections of pods are allowed to communicate with each other and other network endpoints.

How it Works: Fine-grained network namespace isolation using labels and port specifications

- what ingress traffic is allowed to any pod, from any other pod
- on specific ports
- including traffic from pods located in other projects

Additional Enhancements for 3.6:

- [Make services work with NetworkPolicy](#)
- [Implement NetworkPolicy support with PodSelectors \(v1\)](#)
- [Implement NetworkPolicy support for specific ports](#)
- [Implement NetworkPolicy watching/parsing, handle simple policies](#)



Policy applied to namespace: project-a

```
kind: NetworkPolicy
apiVersion: extensions/v1beta1
metadata:
  name: allow-to-red
spec:
  podSelector:
    matchLabels:
      type: red
  ingress:
  - {}
```

Networking

Feature(s): [Router Template Format \(and How to Use Annotations\)](#)

Description: Vastly improved revision of the router customization documentation. Many RFEs could be solved with better documentation around the HAProxy features/functions which we have added, and their customizable fields via annotations/environment variable. For example, router annotations to do per-route things.

Example: Change the behavior of HAProxy (round-robin load balancing) through annotating a route:

```
oc annotate route/ab haproxy.router.openshift.io/balance=roundrobin
```



Networking

Feature(s): [Use a different F5 partition other than /Common](#)

Description: Added the ability to use custom F5 partitions for properly securing/isolating OpenShift route synchronization / configuration.

How it Works:

- The default is still /Common or global partition if not specified and behavior is unchanged from today if the partition path is not specified
- Makes sure all the referenced objects are in the same partition, including virtual servers (http/https)



Networking

Feature(s): [Support IPv6 Terminated at the Router with Internal IPv4](#)

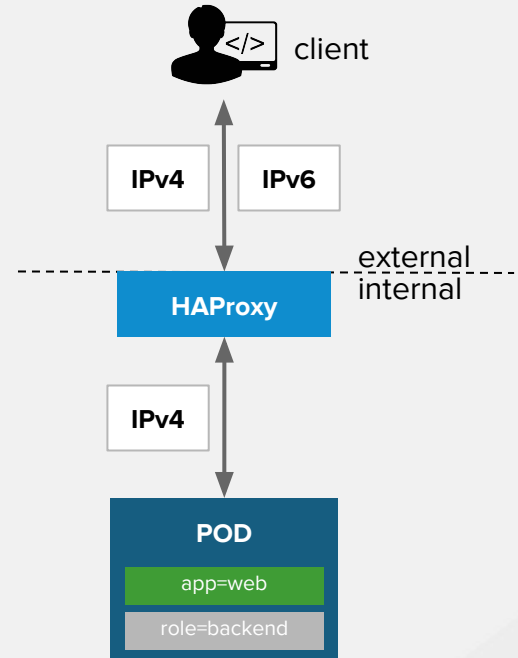
Description: The router container is able to terminate IPv6 traffic and pass HTTP[S] through to the backend pod.

How it Works:

- The IPv6 interfaces on the router should be enabled, have IPv6 addresses listening (:::80, :::443).
- The client should be able to reach the router node using IPv6.
- IPv4 should be unaffected and continue to work, even if IPv6 is disabled.

Caveat: HAProxy can only terminate IPv6 traffic when the router uses the network stack of the host (default), because when using the container network stack:

```
(oadm router --service-account=router  
--host-network=false),  
there is no global IPv6 address for the pod.
```



Metrics & Logging

Feature(s):

- [Removing metrics deployer](#)
- [Removing logging deployer](#)

Description: The metrics and logging deployers were replaced with `playbook2image` for 'oc cluster up' so that `openshift-ansible` is used to install logging and metrics.

How it Works:

- `oc cluster up --logging --metrics`
- Check metrics and logging pod status:
 - `oc get pod -n openshift-infra`
 - `oc get pod -n logging`

Metrics & Logging

Feature(s): [Expose Elasticsearch as a Route](#)

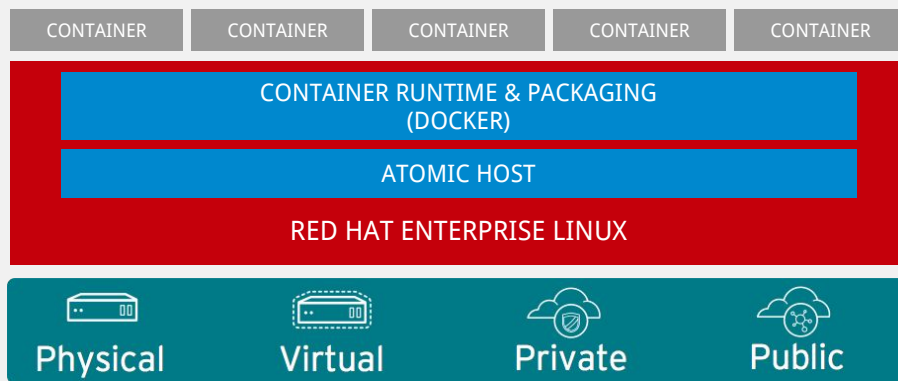
Description: Elasticsearch deployed with OpenShift aggregated logging is not accessible from outside the logging cluster, by default. This enables a route for external access to Elasticsearch for those tools that want to access its data.

How it Works:

- Direct access to ES using only my OpenShift token
 - add the ability to provide the external ES and ES Ops hostnames when creating the server cert (similar to Kibana)
- Ansible tasks to simplify route deployment

Trusted Container OS

Containers Depend on Linux



RHEL / Atomic Host

Feature(s): RHEL 7.3 - 7.4 support.

Description: OpenShift Container Platform 3.6 is supported on RHEL 7.3 and newer with the latest packages from Extras, including docker 1.12.

- Customers are always encouraged to use the latest version of RHEL.
- RHEL 7.2 support was removed due to the fact RHEL 7.4 ships before OCP 3.6.

Quick References:

[Extras Release Notes](#)

[Container Support Policy](#)

[How is container orchestration supported in RHEL?](#)

RHEL 7.4 Highlights

Feature(s): Container Host improvements in RHEL & Atomic Host 7.4.

Description:

Security

- rsyslog 8 rebase & audit improvements
- Network Bound Disk Encryption
- ASLR & KASLR ^(TP) - prevents known memory location vulnerabilities.

Kernel

- Full support for user namespace
- Many performance enhancements

Storage

- SELinux support with OverlayFS
- Full support for overlay2 graph driver
- devicemapper remains the default, and overlay2 is planned to become the default in 7.5

Atomic Host

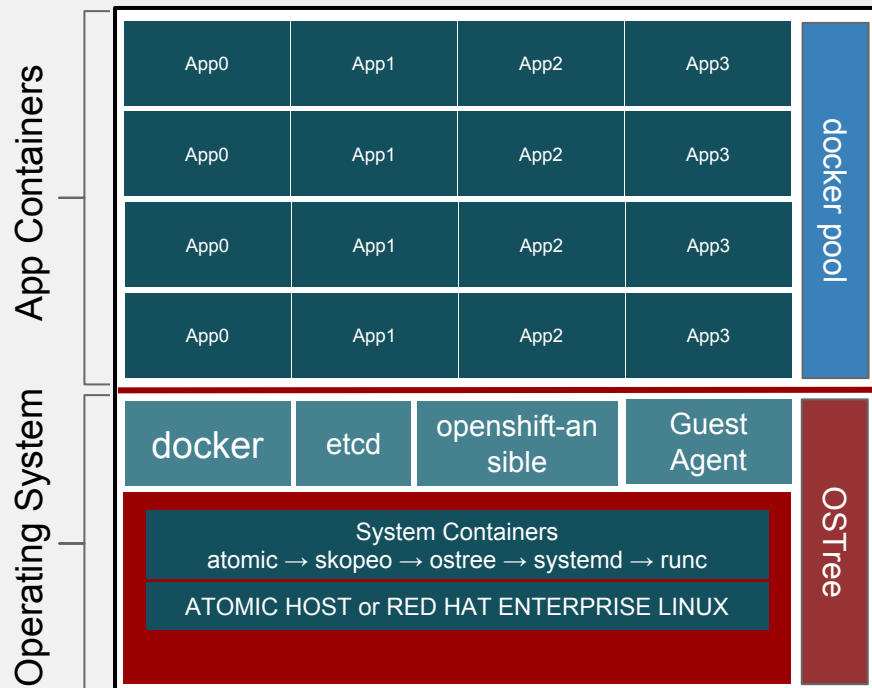
- Flexible partitioning for /var
- Full support for package layering
- LiveFS (tech preview) - Increases system uptime by enabling userspace patching & package layering without rebooting.

RHEL / Atomic Host

Feature(s): System containers for further isolating OCP from OS Updates

Description: OCP 3.6 includes an installation option for system containers as tech preview.

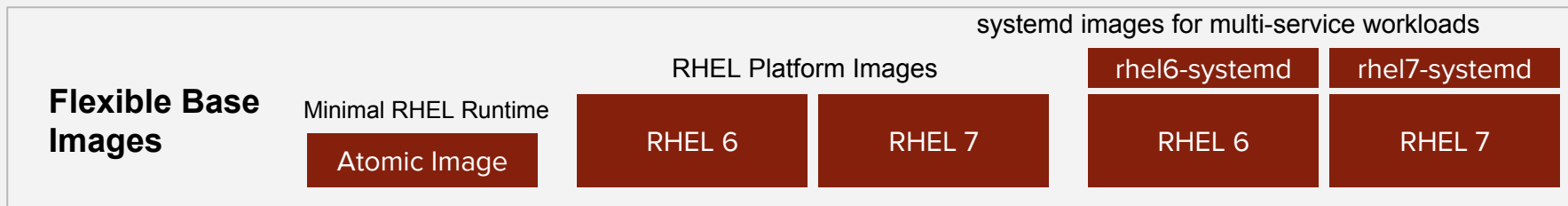
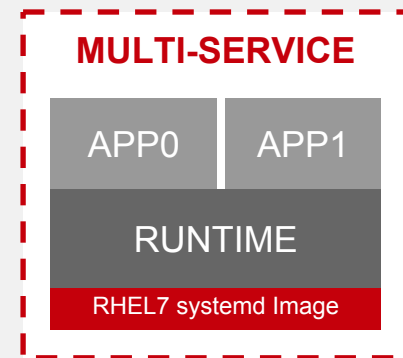
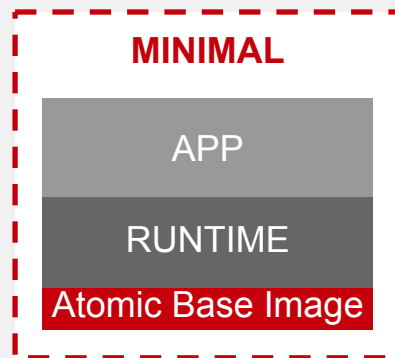
- OCP components can move more independently from the host operating system.
- Images use OSTree for dedupe storage and persist when the docker pool is reset.
- Simple upgrade / rollback
- Managed by openshift-ansible and/or the Atomic CLI.
- OCP Node, Master, SDN planned for 3.7.



Containers

Feature(s): RHEL Atomic (minimal), Platform, and systemd base image availability

Description: RHEL base images are available in several types to ease workload portability and application migrations.



RHEL & ATOMIC HOST
Server Operating Environment

PHYSICAL

VIRTUAL

PRIVATE

PUBLIC

Public Cloud



RED HAT® OPENS SHIFT Update Online

Apps are idled,
slept and
eventually
archived

Available in
multiple regions



Starter

For individual learning and experimenting.

1 Project

1GiB Memory Included

1GiB Terminating Memory Included ⓘ

1GiB Storage Included

Resource Hibernation ⓘ



Summit 2017



Pro

For professional projects and hosting.

10 Projects

2GiB Memory Included
Up to 48GiB Memory Available

2GiB Terminating Memory Included ⓘ
Up to 20GiB Terminating Memory Available

Up to 100GiB Storage

Always On, Unlimited Usage

Invite Collaborators to Projects

Supports Custom Domains

Scheduled Jobs

July 2017

Includes 2 GiB RAM

Add RAM and
Storage in 1 GiB
increments



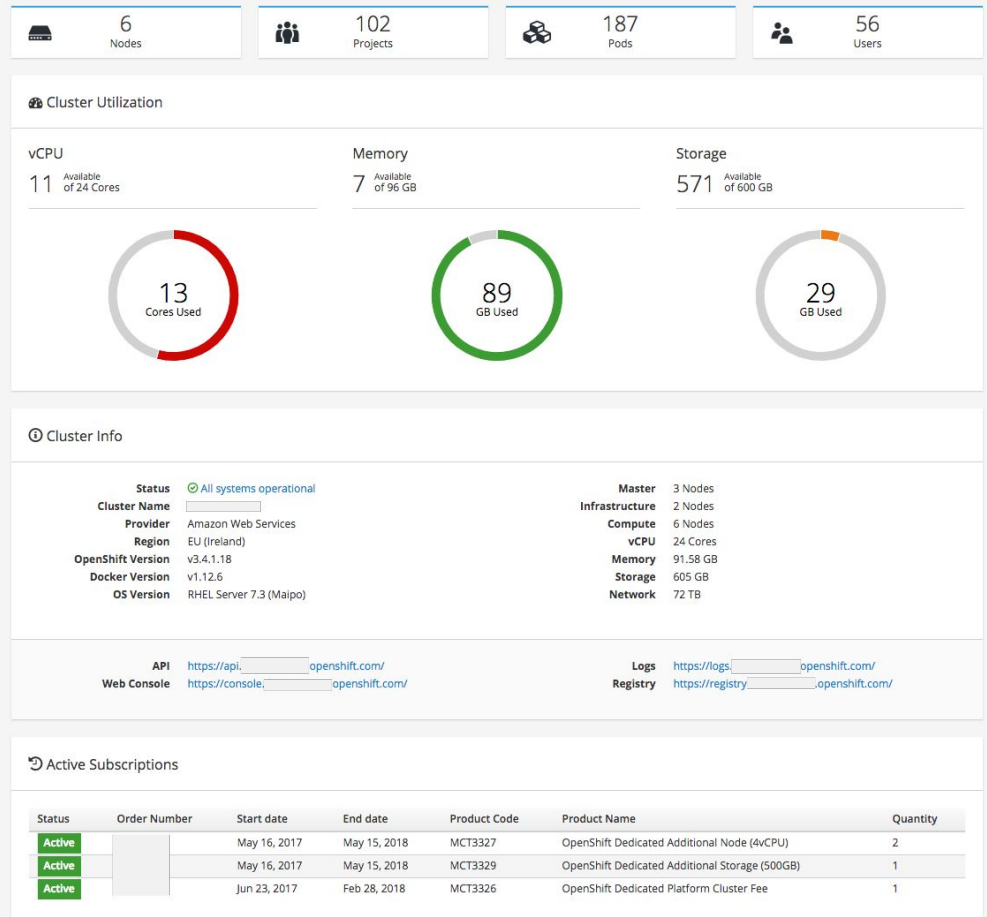
RED HAT® OPENS SHIFT DEDICATED

Feature(s): Customer Portal (early preview)

Description: Customers and Red Hat need a way see capacity and other key data about clusters in a self-service way.

How it Works:

- New application created that aggregates data
- Pulls from cluster stats and Oracle on order details
- Additional phases will provide additional management scenarios



OpenShift Roadmap

OpenShift Container Platform 3.4 (January)

- Kubernetes 1.4 & Docker 1.12
- Dynamic Storage provisioning & Storage QoS tiers
- Storage size quota and scopes
- Kubernetes Deployments & Job Scheduler API
- Pod Disruption Budget
- Eviction on File System Usage
- Usability enhancements & first-time user flows
- Build enhancements (performance, integrations)
- CNI integration for openshift-sdn
- Integration to external logging systems (Splunk)
- Registry enhancements

OpenShift Online & Dedicated

- OpenShift Dedicated on Google (Dec 8)
- OpenShift Online Dev Preview User Expansion

OpenShift Container Platform 3.6 (August)

- Kubernetes 1.6 & Docker 1.12
- New Application Services - 3Scale API Mgt OnPrem, SCL 2.4
- Web UX Project Overview enhancements
- Service Catalog/Broker & UX (Tech Preview)
- Ansible Service Broker (Tech Preview)
- Secrets Encryption (3.6.1)
- Signing/Scanning + OpenShift integration
- Storage - Integrated CNS for Reg and Installer, AWS EFS
- OverlayFS with SELinux Support (RHEL 7.4)
- User Namespaces (RHEL 7.4)
- System Containers for docker

OpenShift Online & Dedicated

- OpenShift Online Paid Tier GA (July)

Q2 CY2017

Q4 CY2017

Q1 CY2017

Q3 CY2017

OpenShift Container Platform 3.5 (April)

- Kubernetes 1.5 & Docker 1.12
- Kubernetes StatefulSets (Tech Preview)
- Java S2I for Cloud Native
- CD Pipeline enhancements
- Dynamic Provisioning for Azure block
- Network Policy (Tech Preview) & Multicast
- Opaque Integers (Tech Preview)
- Registry Image Handling Options
- SCC Usability
- SCL 2.3 shipped out of box

OpenShift Online & Dedicated

- OpenShift Online Free Tier GA (May/Summit)
- OpenShift.io Launch (May/Summit)

OpenShift Container Platform 3.7 (November)

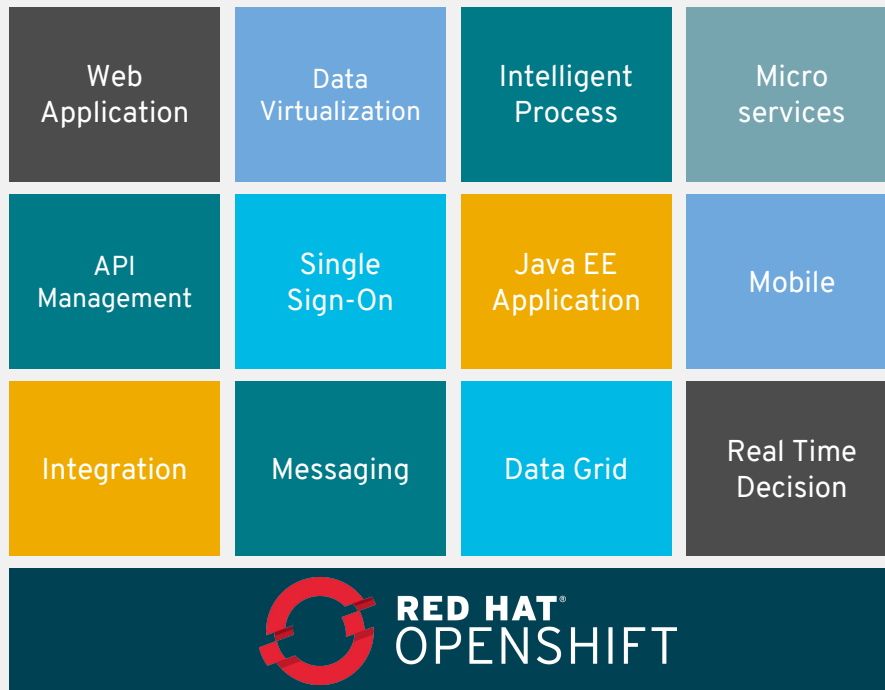
- Kubernetes 1.7 & Docker 1.13
- Red Hat OpenShift Application Runtimes (GA)
- Service Catalog/Broker & UX (GA)
- Ansible Service Broker (GA)
- AWS Service Catalog
- Network Policy (GA)
- Cluster Federation (Tech Preview)
- Prometheus Real-Time Metrics (Tech Preview)
- CloudForms CM-Ops (CloudForms 4.6)
- Performance Sensitive Apps & Spark (TBD)

OpenShift Online & Dedicated

- OpenShift Dedicated Enhancements
- OpenShift Online Enhancements

Questions

A PLATFORM THAT GROWS WITH YOUR BUSINESS



A PLATFORM THAT GROWS WITH YOUR BUSINESS

Intelligent
Process

Upgraded to 6.4 - July 12th

Real Time
Decision

Business Central for OpenShift will be a 7.0 feature.



A PLATFORM THAT GROWS WITH YOUR BUSINESS

Single
Sign-On

Update to SSO 7.1 - concurrent with product
release.

Data Grid

JDG 7.1 for OpenShift - Aug 15th



A PLATFORM THAT GROWS WITH YOUR BUSINESS

Messaging

A-MQ 6.3 - June 29th



A PLATFORM THAT GROWS WITH YOUR BUSINESS

Java EE
Application

EAP 7.1 Beta - concurrent with product - July 25th
EAP 7.1 GA - concurrent with product
Support for Image Source resources

