

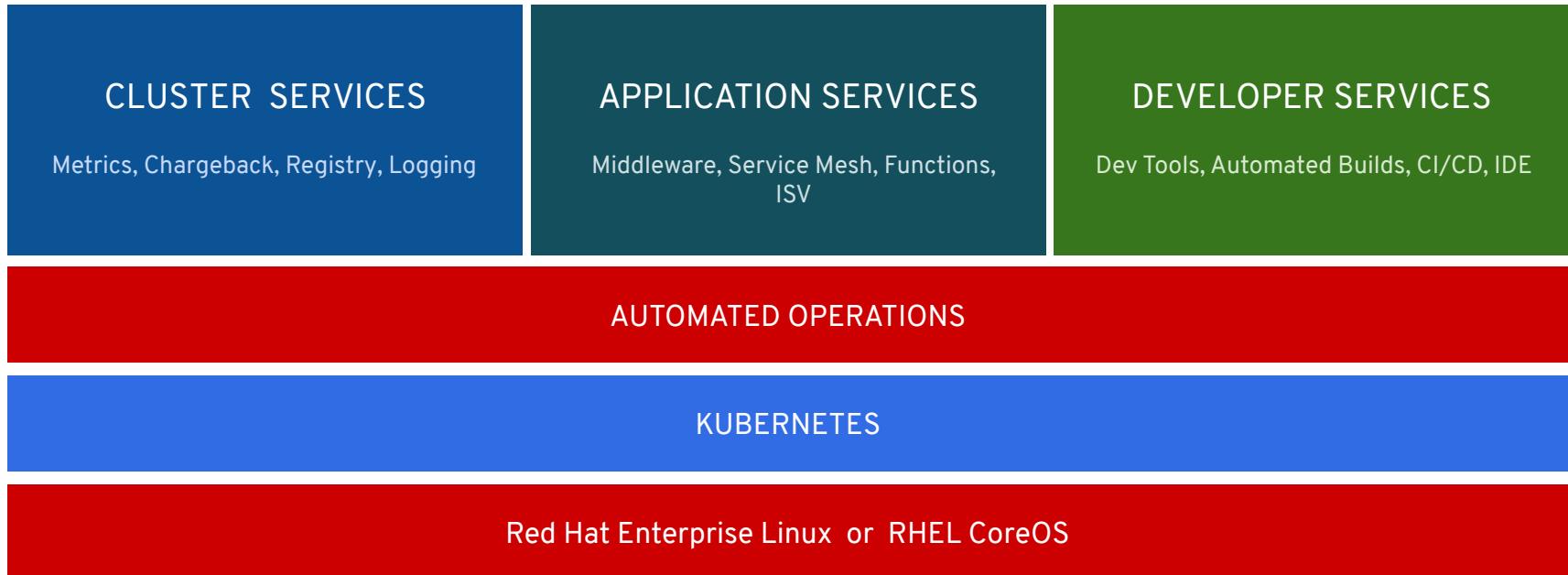


# What's New in OpenShift 4.1

---

OpenShift Product Management

# OpenShift 4 Platform



Best IT Ops Experience

CaaS  $\longleftrightarrow$  PaaS  $\longleftrightarrow$  FaaS

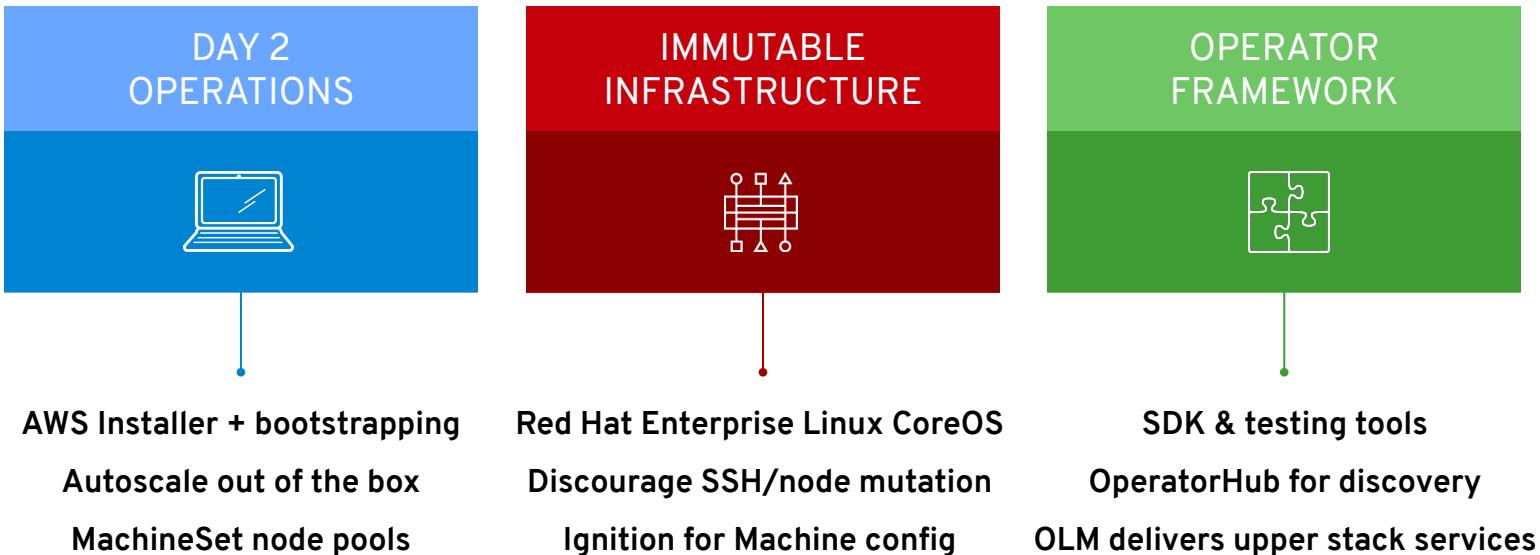
Best Developer Experience



# 2019 Roadmap

Q2 CY2019 OpenShift 4.1		Q3 CY2019 OpenShift 4.2		Q4 CY19/Q1 CY20 OpenShift 4.3	
HOSTED	PLATFORM	HOSTED	PLATFORM	HOSTED	PLATFORM
HOSTED	PLATFORM	HOSTED	PLATFORM	HOSTED	PLATFORM
HOSTED	APP	HOSTED	APP	HOSTED	APP
HOSTED	DEV	HOSTED	DEV	HOSTED	DEV
<ul style="list-style-type: none"><li>• OpenShift Serverless (Knative) - DP</li><li>• OpenShift Pipelines (Tekton) Dev Preview</li><li>• CodeReady Workspaces</li><li>• CodeReady Containers Alpha</li><li>• Developer CLI (odo) Beta</li></ul>	<ul style="list-style-type: none"><li>• OperatorHub</li><li>• Operator Lifecycle Manager</li><li>• Service Mesh (~2 month after)</li></ul>	<ul style="list-style-type: none"><li>• Kubernetes 1.13 with CRI-O runtime</li><li>• RHEL CoreOS, RHEL7</li><li>• Automated Installer for AWS</li><li>• Pre-existing Infra Installer for Bare Metal, VMware, AWS</li><li>• Automated, one-click updates</li><li>• Multus (Kubernetes multi-network)</li><li>• Quay v3</li></ul>	<ul style="list-style-type: none"><li>• GPU metering</li><li>• OperatorHub Enhancements</li><li>• Operator Deployment Field Forms</li><li>• Application Binding with Operators</li><li>• Application Migration Console</li></ul>	<ul style="list-style-type: none"><li>• Kubernetes 1.14 w/ CRI-O runtime</li><li>• Disconnected Install and Update</li><li>• Automated Installer for Azure, OSP, GCP</li><li>• OVN Tech Preview</li><li>• FIPS</li><li>• Federation Workload API</li><li>• Automated App cert rotation</li><li>• OpenShift Container Storage 4.2</li></ul>	<ul style="list-style-type: none"><li>• Kubernetes 1.15 w/ CRI-O runtime</li><li>• Automated Installer for IBM Cloud, Alibaba, RHV, Bare Metal Hardware Appliance</li><li>• Pre-existing Infra Installer for Azure, OSP, GCP</li><li>• OVN GA w/ Windows Networking Integration</li></ul>
<ul style="list-style-type: none"><li>• cloud.redhat.com - Multi-Cluster Mgmt</li><li>• OCP Cluster Subscription Management</li><li>• Azure Red Hat OpenShift</li><li>• OpenShift Dedicated consumption pricing</li></ul>		<ul style="list-style-type: none"><li>• cloud.redhat.com - Multi-Cluster Deployment</li><li>• Proactive Support Operator</li></ul>		<ul style="list-style-type: none"><li>• cloud.redhat.com - Subscription Mgmt Consumption Improvements</li></ul>	

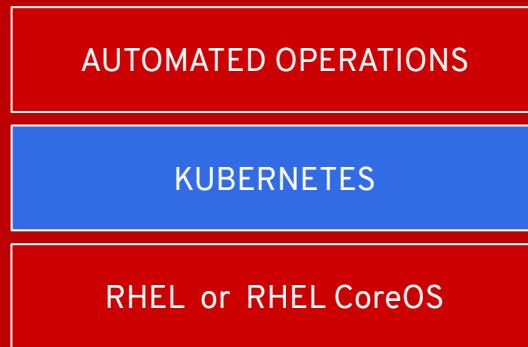
# OpenShift 4.1 Workstreams Lifecycle



# The New Platform Boundary

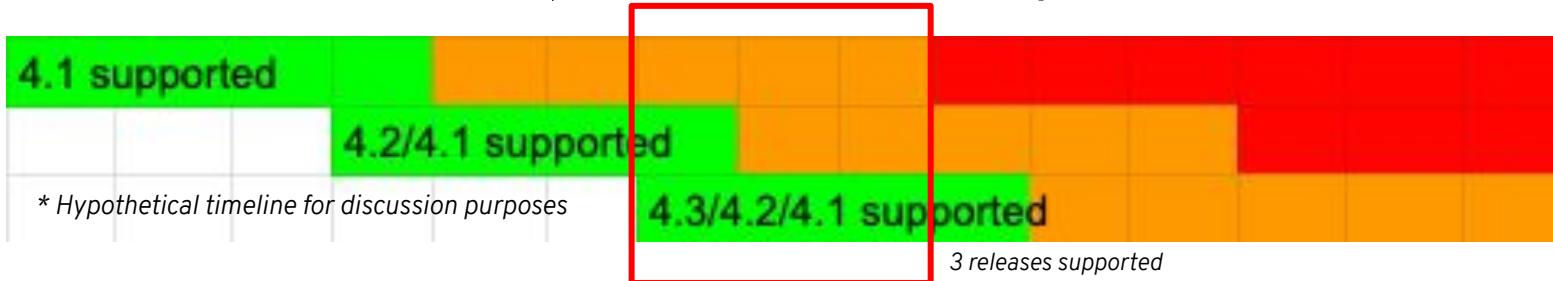
OpenShift 4 is aware of the entire infrastructure and  
brings the Operating System under management

**OpenShift & Kubernetes**  
certificates & security settings  
container runtime config  
allowed maintenance windows  
software defined networking



kernel modules  
device drivers  
network interfaces  
security groups  
**Nodes & Operating System**

# OpenShift 4 Lifecycle



- Release driven. Moved away from declaring dates and will now be based on releases (ie N-2)
- The 4.x Major release will be supported for 3 years.
  - Min of 2 years full support and 1 year maintenance wherein the 2 year full support is has a non declared end (will run longer) and the maintenance cycle starts when full support ends
- Current Release (N) offers full support (All bug fixes, all security fixes). Once a new minor version is released, that full support for the old release will continue for 1 month on the overlap
- After that month, we will offer 5 months of critical bug and critical security fixes for the older release
- At any given time there will be 3 releases supported
- OTA Upgrade will work up to 2 releases in a serial manner. If you fall more than 2 releases behind, you must use the application migration tooling to move to a new cluster
- Red Hat will release EUS updates that are supported for 14 months of critical bug and critical security fixes instead of the normal 5 months. If you stay on the EUS for its entire life, you must use the application migration tooling to move to a new cluster

# Installation Experiences

## OPENSIFT CONTAINER PLATFORM

### Full Stack Automated

Simplified opinionated “Best Practices” for cluster provisioning

Fully automated installation and updates including host container OS.



**Red Hat**  
Enterprise Linux  
CoreOS

### Pre-existing Infrastructure

Customer managed resources & infrastructure provisioning

Plug into existing DNS and security boundaries



**Red Hat**  
Enterprise Linux  
CoreOS



**Red Hat**  
Enterprise Linux

## HOSTED OPENSIFT

### Azure Red Hat OpenShift

Deploy directly from the Azure console. Jointly managed by Red Hat and Microsoft Azure engineers.

### OpenShift Dedicated

Get a powerful cluster, fully Managed by Red Hat engineers and support.

## 4.1 Supported Providers\*

Full Stack Automated



Pre-existing Infrastructure



Bare Metal



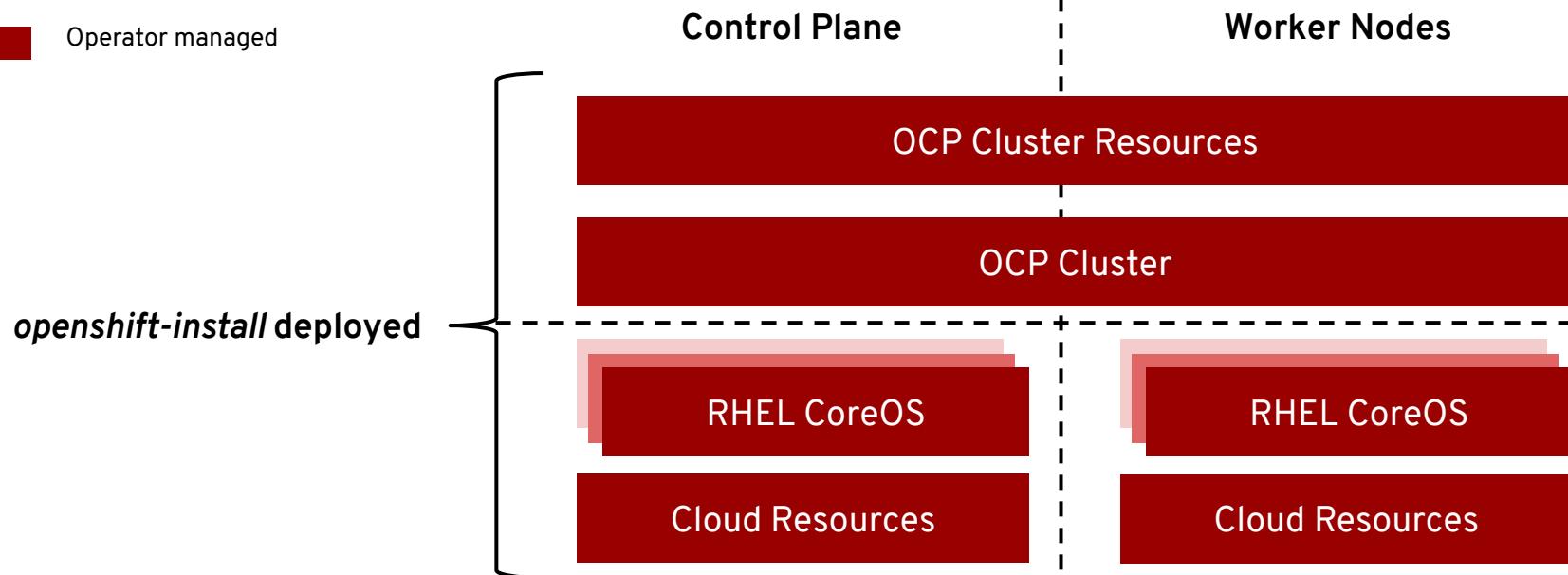
\* Requires Internet connectivity; support for cluster-wide proxy  
& disconnected installation/updating tentatively planned for 4.2

# Full Stack Automated Deployments

Day 1: openshift-install - Day 2: Operators

User managed

Operator managed



# Full Stack Automated Deployments

## Simplified Cluster Creation

Designed to easily provision a “best practices” OpenShift cluster

- New CLI-based installer with interactive guided workflow that allows for customization at each step
- Installer takes care of provisioning the underlying Infrastructure significantly reducing deployment complexity
- Leverages RHEL CoreOS for all node types enabling full stack automation of installation and updates of both platform and host OS content

## Faster Install

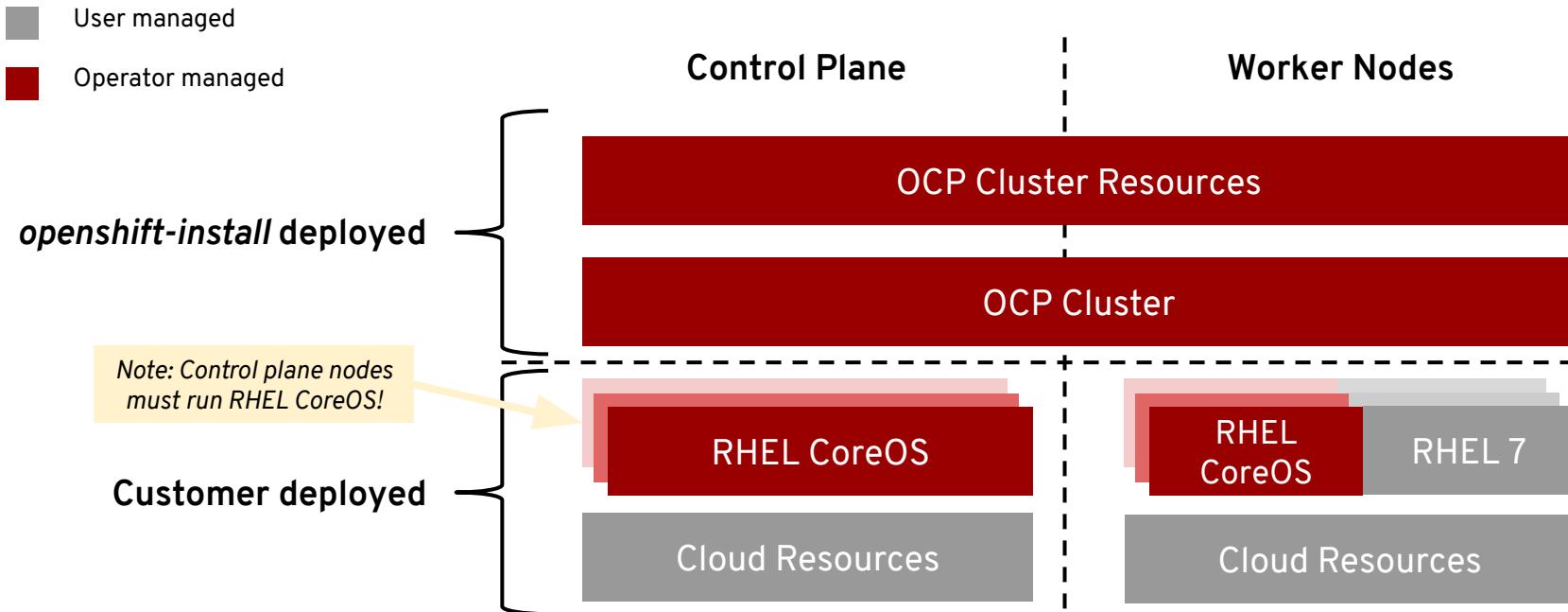
The installer typically finishes within 30 minutes

- Only minimal user input needed with all non-essential install config options now handled by component operator CRD's
- 4.1 provides support for AWS deployments with additional provider support planned in future releases
- [See the OpenShift documentation for more details](#)

```
$ ./openshift-install --dir ./demo create cluster
? SSH Public Key /Users/demo/.ssh/id_rsa.pub
? Platform aws
? Region us-west-2
? Base Domain example.com
? Cluster Name demo
? Pull Secret [? for help]
*****
INFO Creating cluster...
INFO Waiting up to 30m0s for the Kubernetes API...
INFO API v1.11.0+fc69f926354 up
INFO Waiting up to 30m0s for the bootstrap-complete event...
INFO Destroying the bootstrap resources...
INFO Waiting up to 10m0s for the openshift-console route to be created...
INFO Install complete!
INFO Run 'export KUBECONFIG=<your working directory>/auth/kubeconfig' to
manage the cluster with 'oc', the OpenShift CLI.
INFO The cluster is ready when 'oc login -u kubeadmin -p <provided>' succeeds (wait a few minutes).
INFO Access the OpenShift web-console here:
https://console-openshift-console.apps.demo.example.com
INFO Login to the console with user: kubeadmin, password: <provided>
```

# Deploying to Pre-existing Infrastructure

Day 1: openshift-install - Day 2: Operators + Customer Managed Infra & Workers



# Deploying to Pre-existing Infrastructure

## Customized OpenShift Deployments

Enables OpenShift to be deployed to user managed resources and pre-existing infrastructure.

- Customers are responsible for provisioning all infrastructure objects including networks, load balancers, DNS, hardware/VMs and performing host OS installation
- Deployments can be performed both on-premise and to the public cloud
- OpenShift installer handles generating cluster assets (such as node ignition configs and kubeconfig) and aids with cluster bring-up by monitoring for bootstrap-complete and cluster-ready events
- While RHEL CoreOS is mandatory for the control plane, either RHEL CoreOS or RHEL 7 can be used for the worker/infra nodes
- Node auto-scaling can be setup for providers with OpenShift Machine API support
- [See the OpenShift documentation for more details](#)

```
$ cat ./demo/install-config.yaml
apiVersion: v1
baseDomain: example.com
compute:
- name: worker
  replicas: 0
controlPlane:
  name: master
...
$ ./openshift-install --dir ./demo create ignition-config
INFO Consuming "Install Config" from target directory
$ ./openshift-install --dir ./demo wait-for bootstrap-complete
INFO Waiting up to 30m0s for the Kubernetes API at
https://api.demo.example.com:6443...
INFO API v1.11.0+c69f926354 up
INFO Waiting up to 30m0s for the bootstrap-complete event...
$ ./openshift-install --dir ./demo wait-for cluster-ready
INFO Waiting up to 30m0s for the cluster at
https://api.demo.example.com:6443 to initialize...
INFO Install complete!
```

# AWS: Deploying to Pre-existing Infrastructure

## Deploy OpenShift on customized AWS infrastructure

Install OpenShift clusters on user-managed AWS infrastructure.

- Customers are responsible for provisioning:
  - VPC/Subnets, Route53 Zone & DNS entries, Load Balancers & listeners, Security groups, IAM roles, S3 bucket, and EC2 instances for Bootstrap, Master, and Worker nodes
  - Public hosted zone in Route53 in the same account as your cluster; ensure the zone is "authoritative" for the domain
- To make AWS infrastructure provisioning easier, example AWS CloudFormation templates have been included as a guide for how to prepare your environment for OpenShift deployments
  - While templates are only intended to provide basic functionality, they can be further enhanced based on your operation needs or even automated using orchestration technologies like Ansible
- Additionally, you will likely need to review and increase the service limits of your AWS account
- [See the OpenShift documentation for more details](#)

Stack Name	Created Time	Status	Drift Status	Description
kalexand-master	2019-04-29 11:56:59 UTC-0400	CREATE_COMPLETE	NOT_CHECKED	Template for Openshift Cluster ...
kalexand-bootstrap	2019-04-29 11:39:03 UTC-0400	CREATE_COMPLETE	NOT_CHECKED	Template for Openshift Cluster ...
kalexand-security	2019-04-29 11:27:42 UTC-0400	CREATE_COMPLETE	NOT_CHECKED	Template for Openshift Cluster ...
kalexand-network	2019-04-29 11:17:00 UTC-0400	CREATE_COMPLETE	NOT_CHECKED	Template for Openshift Cluster ...
kalexand-vpc	2019-04-29 10:56:10 UTC-0400	CREATE_COMPLETE	NOT_CHECKED	Template for Best Practice VPC...

# OpenShift Bootstrap Process: Self-Managed Kubernetes

## How to boot a self-managed cluster:

- OpenShift 4 is unique in that management extends all the way down to the operating system
- Every machine boots with a configuration that references resources hosted in the cluster it joins enabling cluster to manage itself
- Downside is that every machine looking to join the cluster is waiting on the cluster to be created
- Dependency loop is broken using a bootstrap machine, which acts as a temporary control plane whose sole purpose is bringing up the permanent control plane nodes
- Permanent control plane nodes get booted and join the cluster leveraging the control plane on the bootstrap machine
- Once the pivot to the permanent control plane takes place, the remaining worker nodes can be booted and join the cluster

## Bootstrapping process step by step:

1. Bootstrap machine boots and starts hosting the remote resources required for master machines to boot.
2. Master machines fetch the remote resources from the bootstrap machine and finish booting.
3. Master machines use the bootstrap node to form an etcd cluster.
4. Bootstrap node starts a temporary Kubernetes control plane using the newly-created etcd cluster.
5. Temporary control plane schedules the production control plane to the master machines.
6. Temporary control plane shuts down, yielding to the production control plane.
7. Bootstrap node injects OpenShift-specific components into the newly formed control plane.
8. Installer then tears down the bootstrap node or if user-provisioned, this needs to be performed by the administrator.

# Comparison between deployments methods

	Full Stack Automation	Pre-existing Infrastructure
Build Network	Installer	User
Setup Load Balancers	Installer	User
Configure DNS	Installer	User
Hardware/VM Provisioning	Installer	User
OS Installation	Installer	User
Generate Ignition Configs	Installer	Installer
OS Support	Installer: RHEL CoreOS	User: RHEL CoreOS + RHEL 7
Node Provisioning / Autoscaling	Yes	Only for providers with OpenShift Machine API support
Customization & Provider Support	Best Practices: AWS	Yes: AWS, Bare Metal, & VMware

# RHEL 7 Worker/Infra Nodes

## Add RHEL 7.6 machines with Ansible

Use openshift-ansible to prepare, configure and join your RHEL 7 nodes to the cluster. After you have a functional control plane, nodes can be added to the cluster with the `scaleup` playbook.

- Pending certificates signing request (CSRs) for each RHEL machine must be approved before joining cluster
- [See the OpenShift documentation for more details](#)

## Mixed clusters of RHEL 7 and RHEL CoreOS are ok

RHEL CoreOS is mandatory for the control plane, but mixed clusters of RHEL 7.6 and RHEL CoreOS are supported for any other node pools.

## RHEL 8 is not yet supported for worker/infra nodes

Support will come in a later 4.x release

## Upgrading OpenShift node components

With the upgrade playbook, RHEL 7 OpenShift components can be upgraded. Optionally, pre/post hooks are also available for performing custom tasks.

## RHEL admins are responsible for:

### Keeping host inventory up to date

Refresh your list of hosts before an upgrade, to make sure no machines are missed

### Defining Ansible hooks

Run playbooks that will cordon/uncordon machines, along with any pre/post upgrade actions

### Updating RHEL RPM content

Security, performance and regular updates from Red Hat

### Partitioning disks

Configure, maintain and health check your disks

### Configuring network interfaces

Configure, secure and maintain settings within your data center's specifications

# Adding & Updating RHEL 7 Worker Nodes

## System Requirements

- OpenShift 4 cluster deployed to pre-existing infrastructure (not supported on installer provisioned clusters)
- Latest release of RHEL 7 Server installed on every machine joining the cluster
  - Ansible >= 2.7.8, OpenShift Client (oc)
  - Necessary entitlements and YUM repo access pre-configured for every machine

## Create an Ansible Inventory

- Inventory file with the appropriate groups and variables defined
  - An example inventory can be found in [inventory/hosts.example](#)
- Required variables include:
  - openshift\_kubeconfig\_path: Path to the kubeconfig for the cluster
  - openshift\_pull\_secret\_path: Path to the pull secret to the image registry

## Run RHEL 7 node ‘scaleup’ playbook

- ```
$ cd openshift-ansible; ansible-playbook -i inventory/hosts playbooks/scaleup.yml
```
- Pending certificates signing request (CSRs) for each RHEL machine must be approved before joining cluster

```
$ oc adm certificate approve <csr_name>
```

## Upgrading RHEL 7 OpenShift node components

- Leverages upgrade section of Ansible Inventory to specify nodes
- Custom tasks can be performed during upgrades at different stages of the upgrade; refer to ‘hooks’ in the documentation for more information.

```
$ cd openshift-ansible; ansible-playbook -i inventory/hosts playbooks/upgrade.yml
```

# Red Hat Enterprise Linux

| RED HAT® ENTERPRISE LINUX®       |                                                                                                                                                                                                                                                               |
|----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BENEFITS                         | General Purpose OS                                                                                                                                                                                                                                            |
|                                  | <ul style="list-style-type: none"><li>• 10+ year enterprise life cycle</li><li>• Industry standard security</li><li>• High performance on any infrastructure</li><li>• Customizable and compatible with wide ecosystem of partner solutions</li></ul>         |
| WHEN TO USE                      | When customization and integration with additional solutions is required                                                                                                                                                                                      |
| RED HAT® ENTERPRISE LINUX CoreOS |                                                                                                                                                                                                                                                               |
|                                  | Immutable container host                                                                                                                                                                                                                                      |
|                                  | <ul style="list-style-type: none"><li>• Self-managing, over-the-air updates</li><li>• Immutable and tightly integrated with OpenShift</li><li>• Host isolation is enforced via Containers</li><li>• Optimized performance on popular infrastructure</li></ul> |
| WHEN TO USE                      | When cloud-native, hands-free operations are a top priority                                                                                                                                                                                                   |

# Red Hat Enterprise Linux CoreOS

## 4.1 Image Availability:

- Amazon: AMIs
- vSphere: OVA
- Bare Metal: UEFI & BIOS

## Installation Requirements:

- RHCOS image + ignition config (installer generated)

## RHCOS Details

- RHEL 8 bits (4.18 kernel)
- Includes all packages required for OpenShift
- Over-The-Air updates encompass OCP & RHCOS
- Transactional host updates
- Read-only OS binaries
- Preconfigured for most environments

## Bare Metal Installer (ISO or PXE):

```
coreos.inst=yes  
coreos.inst.install_dev=sda  
coreos.inst.image_url=http://10.10.10.1/rhcos-metal-uefi.raw.gz  
coreos.inst.ignition_url=http://10.10.10.1/master.ign
```

# Immutable Operating System

**Red Hat Enterprise Linux CoreOS is versioned with OpenShift**

CoreOS is tested and shipped in conjunction with the platform. Red Hat runs thousands of tests against these configurations.

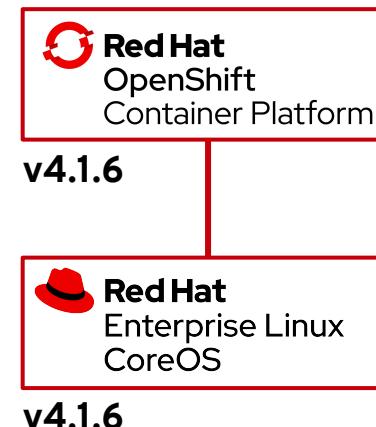
**Red Hat Enterprise Linux CoreOS is managed by the cluster**

The Operating system is operated as part of the cluster, with the config for components managed by Machine Config Operator:

- CRI-O config
- Kubelet config
- Authorized registries
- SSH config

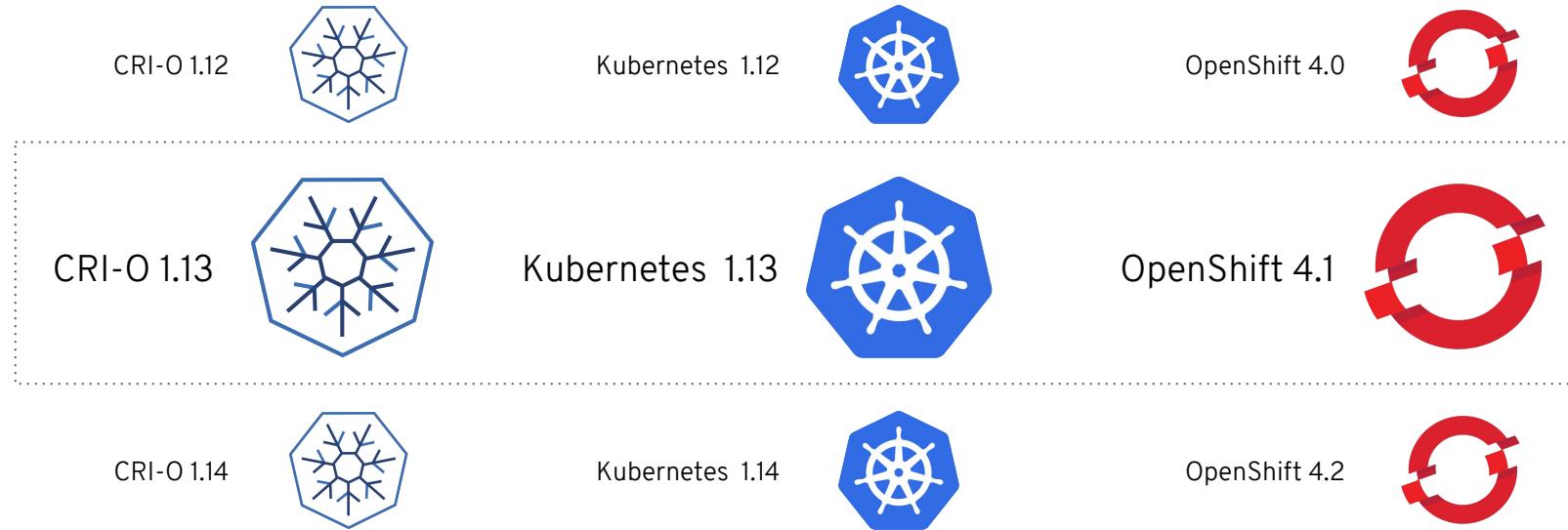
**RHEL CoreOS admins are responsible for:**

Nothing. 



# CRI-O Support in OpenShift

CRI-O tracks and versions identical to Kubernetes, simplifying support permutations



# Machine Config Operator (MCO)

## Example with CRI-O:

- Simplified management across entire cluster
- Container Runtime Config (CRC) is specialized Machine Config (MC) for CRI-O
- Container Runtime Config for CRI-O exposes configuration knobs

## Leverages

- Custom Resource Definitions
- Machine Config Pools
- Machine Config Pool Selector
- Machine Configs
- Container Runtime Config (specialized MC)

```
apiVersion: machineconfiguration.openshift.io/v1
kind: ContainerRuntimeConfig
metadata:
  name: set-log-and-pid
spec:
  machineConfigPoolSelector:
    matchLabels:
      debug-crio: config-log-and-pid
  containerRuntimeConfig:
    pidsLimit: 2048
    logLevel: debug
```

```
$ oc get ContainerRuntimeConfig
```

| NAME            | AGE |
|-----------------|-----|
| set-log-and-pid | 22h |

# Discovering Configuration Options

## Cluster config objects are the human/machine APIs

These can be discovered with the commands to the right.

Configuration set via this mechanism will be carried forward through cluster updates.

## Configuration Resources

|                |                |
|----------------|----------------|
| Apiserver      | Authentication |
| Build          | Clusterversion |
| Console        | Dns            |
| Featuregate    | Image          |
| Infrastructure | Ingress        |
| Network        | Oauth          |
| Project        | Scheduler      |

```
$ oc api-resources --api-group config.openshift.io -o name  
| cut -d . -f 1  
apiservers  
authentications  
...  
scheduler
```

```
$ oc explain --api-version=config.openshift.io/v1  
scheduler.spec  
KIND: Scheduler  
VERSION: config.openshift.io/v1  
  
FIELDS:  
  defaultNodeSelector <string>  
    defaultNodeSelector helps set the cluster-wide default  
    node selector to restrict pod placement to specific...  
  
  policy <Object>
```

```
$ oc explain --api-version=config.openshift.io/v1  
scheduler.spec.policy
```

# Graphical Re-configuration

## Global Configuration

You complete most of the cluster configuration and customization after you deploy your OpenShift Container Platform cluster.

### Change via Cluster Settings screen

Once you have discovered your desired settings (prev. slide), changes can be made via Console or CLI.

### Operators apply these updates

One or more Operators are responsible for propagating these settings through the infrastructure

The screenshot shows the Red Hat OpenShift Container Platform interface. The top navigation bar includes the Red Hat logo and the text "OpenShift Container Platform". A user "kube:admin" is logged in. The left sidebar has a dark theme with white text. It lists several categories: Home, Catalog, Workloads, Networking, Storage, Builds, Monitoring, Compute (with sub-options Nodes, Machines, Machine Sets, Machine Configs, Machine Config Pools), Administration (with sub-options Cluster Status, Cluster Settings, Namespaces, Service Accounts, Roles, Role Bindings, Resource Quotas, Limit Ranges, Custom Resource Definitions), and Project. The "Cluster Settings" option under Administration is currently selected. The main content area is titled "Cluster Settings" and displays a list of "CONFIGURATION RESOURCE" items. Each item has a link and an "Edit YAML" button. The items listed are: APIServer, Authentication, Build, ClusterVersion, Console, DNS, FeatureGate, Image, Infrastructure, Ingress, Network, OAuth, Project, and Scheduler. The "Edit YAML" buttons are located at the end of each row.

# Network Configuration

## Example #1: Operator-Assisted Ingress Ctrlr “Sharding”

In 4.1, the way you create a router to work with a shard is different (API call versus ‘oc adm’ command). A simple config (example to right) acted upon by the ingress operator automatically integrates sharding with the external (front-end) DNS/LB configured at install-time.,

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  namespace: openshift-ingress-operator
  name: internal-apps
spec:
  domain: internal-apps.dmace.devcluster.openshift.com
  routeSelector:
    matchLabels:
      environment: internal
```

## Example #2: Create a Second Router

Ingress controller configuration is now a first-class object, meaning additional Ingress controllers can be created by making multiple Ingress objects. This is the preferred method for giving teams their own subdomains, replacing the ‘oc adm’ method (see right).

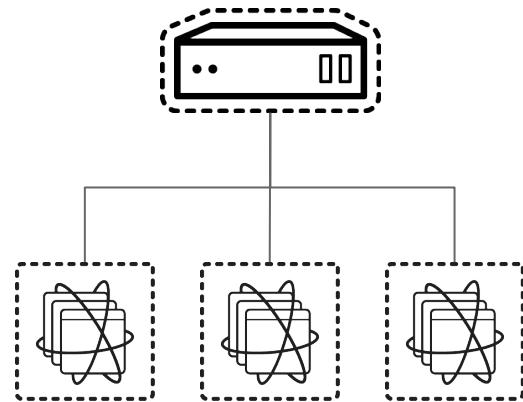
```
$ cat <<EOF | oc create -f -
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  namespace: openshift-ingress-operator
  name: finance-apps
spec:
  domain: finance-apps.openshift.example.com
EOF
```

# HAProxy Optimization

## HAProxy Large Performance Increase

OCP 4.1 deploys 2 router replicas by default (improving the observed performance ~2x), and sets 4 router threads by default (also improving observed performance another ~2x).

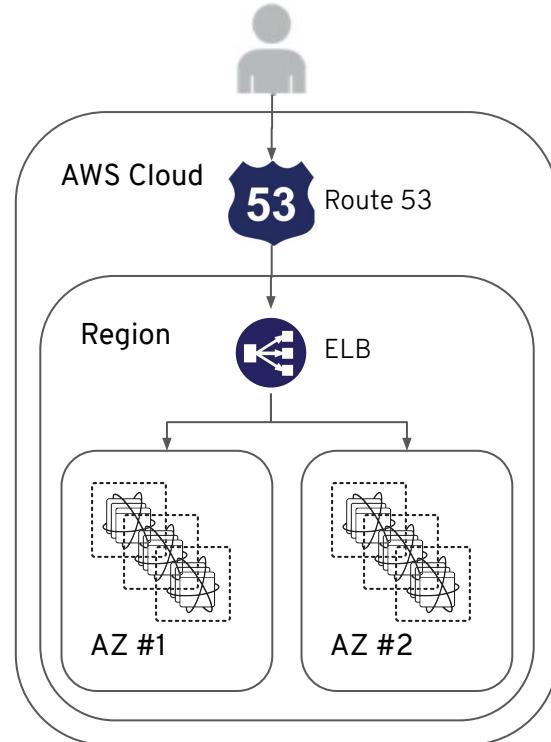
We increased these defaults for HAProxy so that the cluster can serve more routes.



# Platform-Integrated Networking

## Installer Provisioned Networking Infrastructure

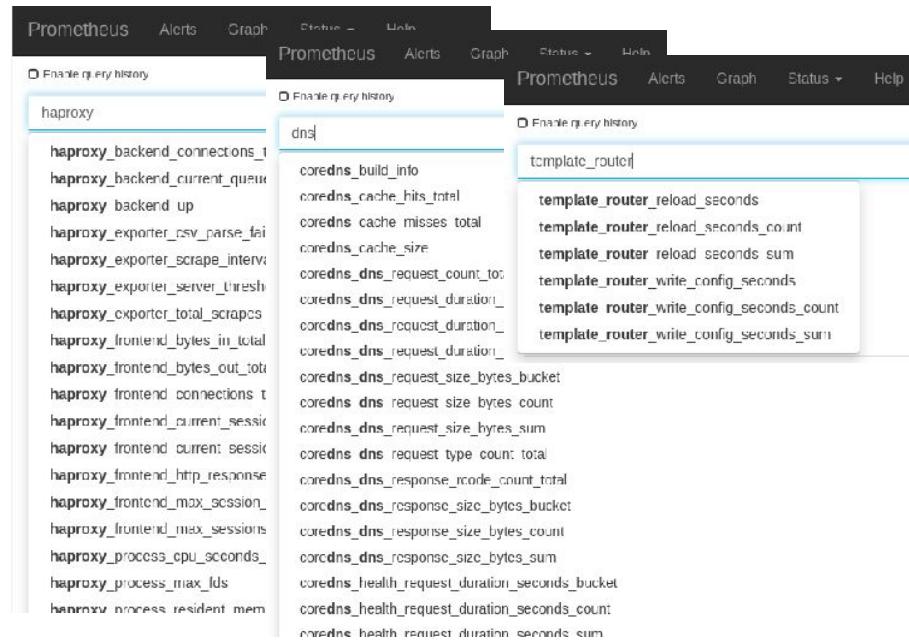
The OCP 4.1 installer takes full advantage of the provider networking infrastructure, including automatic integration with the provider's external DNS and Load Balancing.



# Networking Metrics

## Networking Metrics Visibility in Prometheus

Networking metrics (HAProxy/Router, CoreDNS, etc.)  
for consumption by Prometheus are GA at 4.1.



# Networking Re-configuration

## Networking Advanced Settings

These are the OpenShift SDN settings that can be tweaked at install-time:

- Mode: NetworkPolicy, Multitenant, Subnet
- VXLAN Port Number
- MTU (autodetected, once)
- External OpenVSwitch

## [How to Modify Advanced Network Configuration Parameters](#)

```
spec:  
  defaultNetwork:  
    type: OpenShiftSDN  
    openshiftSDNConfig:  
      mode: NetworkPolicy  
      vxlanPort: 4789  
      mtu: 1450  
      useExternalOpenvswitch: false
```

**NOTE:** Most network settings cannot be changed safely and affect the entire cluster. The operator will prevent unsafe changes. If you need to force a change to a *non-production* cluster, see the operator README for the command, but a cluster re-install is likely to be the better choice.

# kube-proxy Re-configuration

## kube-proxy Advanced Settings

These are the kube-proxy settings that can be tweaked at install-time:

- iptablesSyncPeriod
- bindAddress
- proxyArguments (a list of kube-proxy command-line flags)

## [How to Modify Advanced Network Configuration Parameters](#)

```
spec:  
  kubeProxyConfig:  
    iptablesSyncPeriod: 30s  
    bindAddress: 0.0.0.0  
    proxyArguments:  
      iptables-min-sync-period: ["30s"]
```

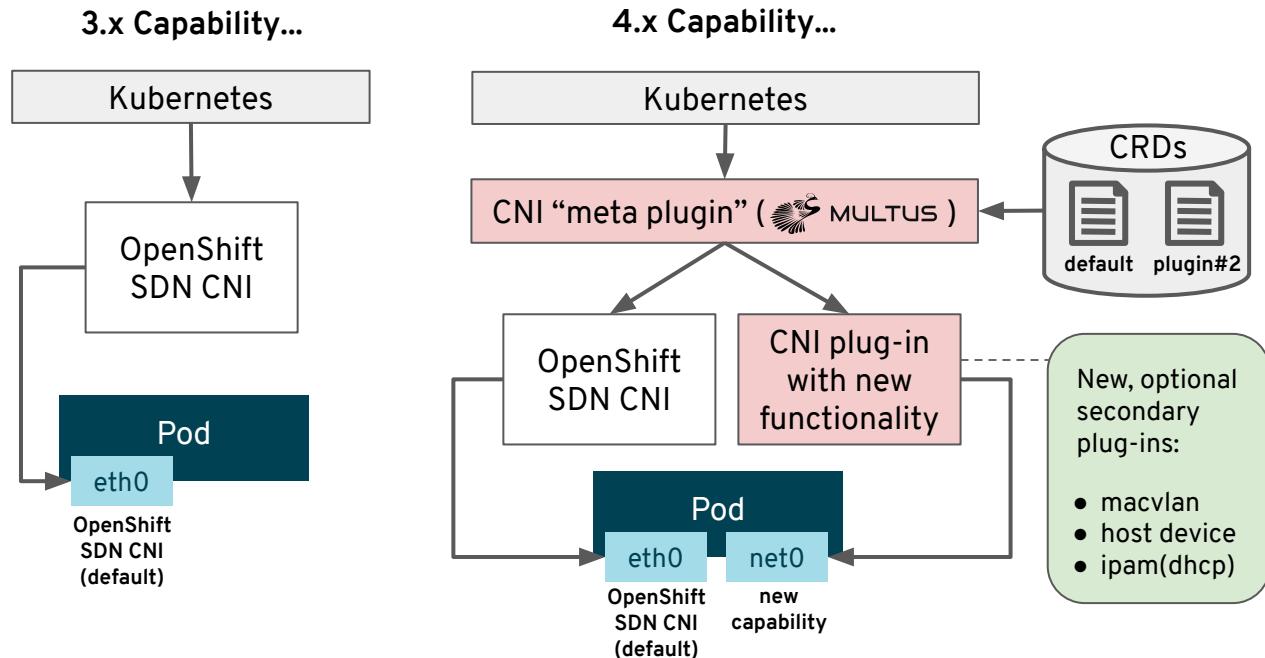
**NOTE:** Most network settings cannot be changed safely and affect the entire cluster. The operator will prevent unsafe changes. If you need to force a change to a *non-production* cluster, see the operator README for the command, but a cluster re-install is likely to be the better choice.

# Networking Plug-ins

## Multus Enables Multiple Networks & New Functionality to Existing Networking

The Multus CNI “meta plugin” for Kubernetes enables one to create multiple network interfaces per pod, and assign a CNI plugin to each interface created.

1. Create pod annotation(s) to call out a list of intended network attachments...
2. ...each pointing to CNI network configurations packed inside CRD objects

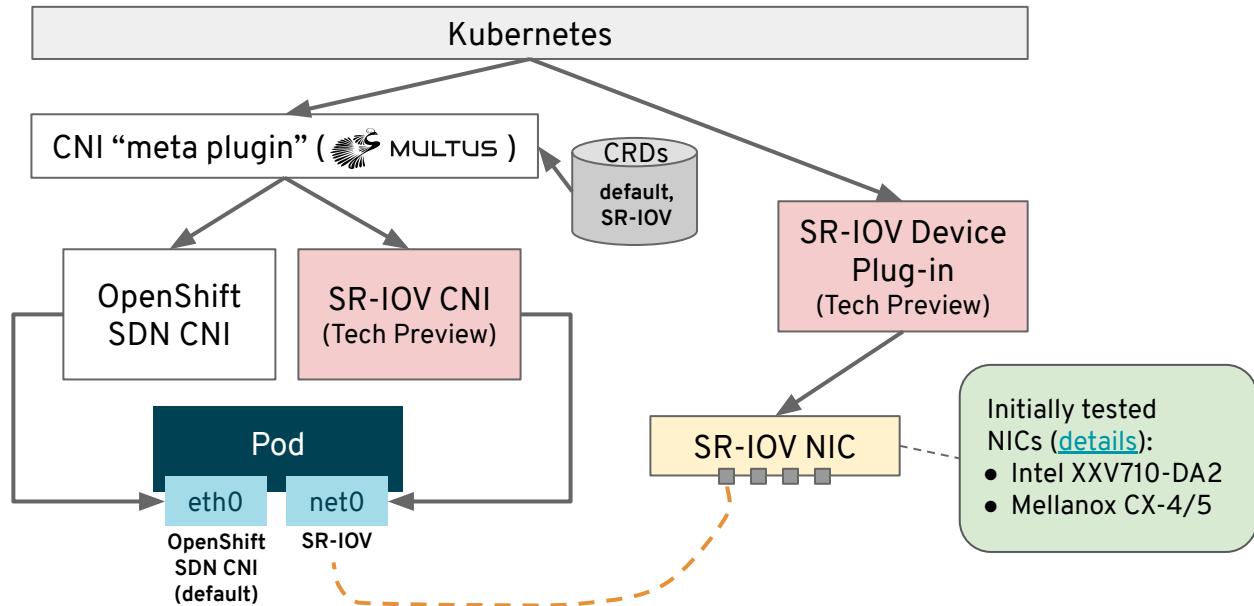


# High-Performance Networking (Tech Preview)

## Approach Line-Rate Performance to the Pod

OCP 4.1 includes the capability to use specific SR-IOV hardware on cluster nodes to brings near-line rate network performance to cluster pods.

## Configuring SR-IOV



# Storage

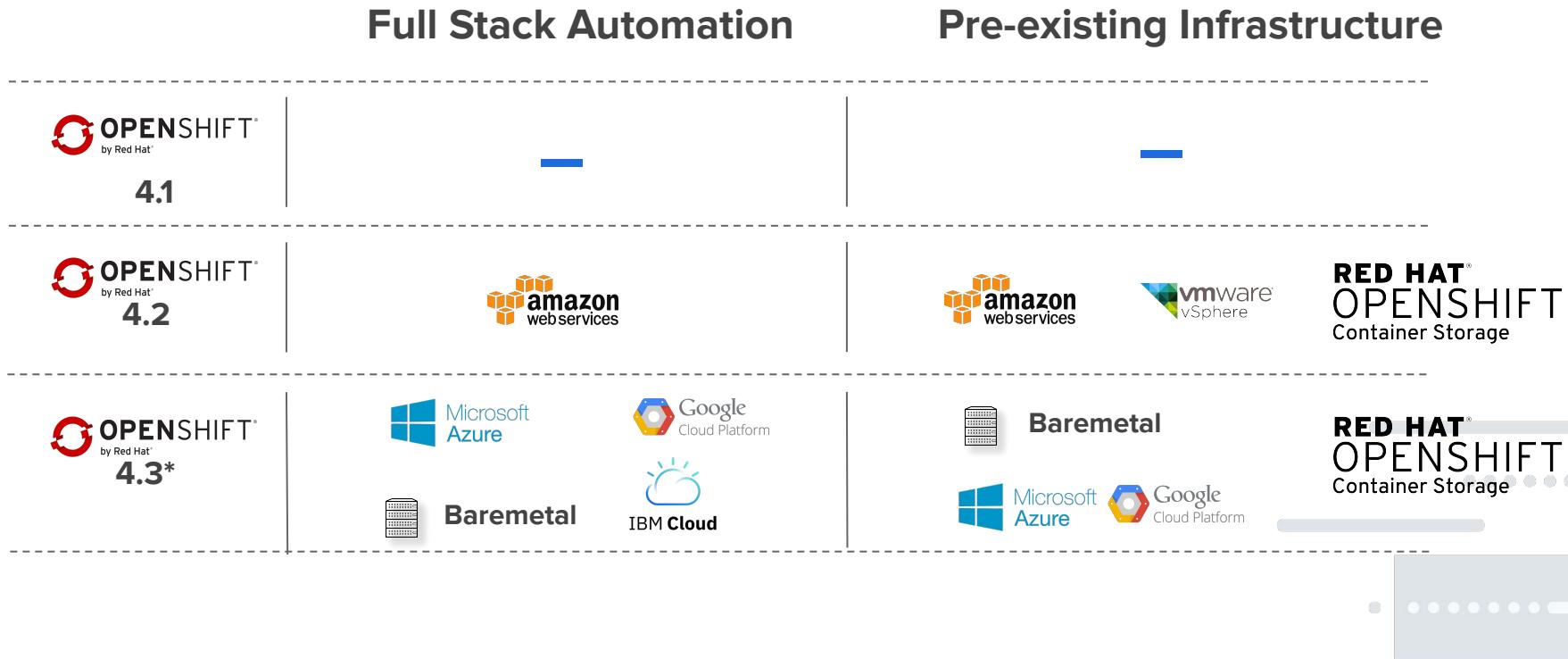
## Storage Focus

- Cluster Storage Operator
  - Sets up the default storage class
  - Looks through cloud provider and sets up the correct storage class
- Drivers themselves remain in-tree for now
- Focus has been on RHEL7 and RHCOS8 validating:
  - AWS EBS
  - vSphere Default Storage Class

| Supported      | Dev Preview   |
|----------------|---------------|
| NFS            | Snapshot*     |
| iSCSI          | EFS*          |
| AWS EBS        | Local Volume* |
| VMware vSphere | Raw Block     |

\* via external provisioner

# PROVIDER ROADMAP FOR OPENSOURCE CONTAINER STORAGE 4



# Configuring an Identity Provider

## The Cluster Authentication Operator

- Use the ***cluster-authentication-operator*** to configure an Identity Provider. The configuration is stored in the ***oauth/cluster*** custom resource object inside the cluster.
- Once that's done, you may choose to remove kubeadmin (warning: there's no way to add it back).
- All the identity providers supported in 3.11 are supported in 4.1: LDAP, GitHub, GitHub Enterprise, GitLab, Google; OpenID Connect, HTTP request headers (for SSO), Keystone, Basic authentication.
- For more information:  
[Understanding identity provider configuration](#)  
[cluster-authentication-operator](#)

Sample identity provider CR

```
apiVersion: config.openshift.io/v1
kind: OAuth
metadata:
  name: cluster
spec:
  identityProviders:
    - name: my_identity_provider ①
      mappingMethod: claim ②
      type: HTPasswd
      htpasswd:
        fileData:
          name: htpass-secret ③
```

① This provider name is prefixed to provider user names to form an identity name.

② Controls how mappings are established between this provider's identities and user objects.

③ An existing secret containing a file generated using `htpasswd`.

# Service Certificates and the Service CA

## The Service CA Operator

- The Service Certificate Authority (CA) can sign server certificates for services running in the OpenShift cluster.
- The **service-ca operator** manages 3 controllers:  
**service-ca controller**  
**configmap-cabundle-injector controller**  
**apiservice-cabundle-injector controller**
- The Service CA is valid for one year after installation. Manual steps to refresh the Service CA are documented. Automation is targeted for 4.2.
- For more information:  
[Service Serving Certificate Secrets](#)  
[openshift-service-ca-operator](#)

```
$ oc create configmap foobar --from-literal=key1=foo
configmap/foobar created
$ oc get configmap/foobar -o yaml
apiVersion: v1
data:
  key1: foo
kind: ConfigMap
metadata:
  creationTimestamp: 2018-09-11T23:44:56Z
  name: foobar
  namespace: myproject
  resourceVersion: "56490"
  selfLink: /api/v1/namespaces/myproject/configmaps/foobar
  uid: afee501b-b61c-11e8-833b-c85b762603b0
$ oc annotate configmap foobar service.beta.openshift.io/inject-cabundle="true"
configmap/foobar annotated
$ oc get configmap/foobar -o yaml
apiVersion: v1
data:
  key1: foo
  service-ca.crt: |
    -----BEGIN CERTIFICATE-----
    MIIDCjCCAfKgAwIBAgIBATANBgkqhkiG9w0BAQsFADA2MTQwMgYDVQQDDCtvGVu
    c2hpZnQtc2VydmljZS1zZXJ2aW5nLXnpZ25lckAxNTM2Njk1NTIxMBA4XDTE4MDkx
    MTE5NTIwMVoxDTIzMdkxMDE5NTIwMlowNjE0MDIGA1UEAwrb3BlbnNoawZ0LXN1
    cnZpY2Utc2VydmluZy1zaWduZXJAMTUzNjY5NTUyMTCCASiwDQYJKoZIhvNAQEB
    BQADggEPADCCAQoCggEBANP9Asc657SkWP0ohmMlrXqirl7taaarmM5l3/pNeO
```

# Custom Certificates for External Endpoints

## Custom certificates for Ingress Controllers

- In OpenShift 4 adding custom certificates for external endpoints is a post-installation task.
- Cluster ingress is managed by the Ingress Operator which configures Ingress Controllers to route traffic as specified by OpenShift [Route](#) and Kubernetes [Ingress](#) resources.
- Updating the certificate for the ingress controller(s) covers the Web console, the API endpoint, the internal Registry and custom applications. For the Registry, you can choose to [define a separate route with certificate](#).
- For example:  
[Requesting and Installing Let's Encrypt Certificates for OpenShift 4](#)

## Requesting and Installing Let's Encrypt Certificates for OpenShift 4

The Router expects the certificates in a Secret. This secret needs to be created in the project openshift-ingress.

1. Use the following command to create the secret – and if you have existing certificates, make sure to provide the path to your certificates instead.

```
oc create secret tls router-certs --cert=${CERTDIR}/fullchain.pem --key=${CERTDIR}/key.pem
```

2. Now update the Custom Resource for your router. The default custom resource is of type IngressController, is named default and is located in the openshift-ingress-operator project. Note that this project is different from where you created the secret earlier.

```
oc patch ingresscontroller default -n openshift-ingress-operator --type=merge --patch='{"
```

3. This is all you need to do. After you update the IngressController object the OpenShift ingress operator notices that the custom resource has changed and therefore redeploys the router.

You now have proper certificates on the router – and this includes custom applications, the Web Console for your OpenShift Cluster and the API Endpoint.

# Registry Configuration

## Image Registry Operator Configuration Parameters

Several parameters support changing the registry configuration. Those are defined in the `configs.imageregistry.operator.openshift.io` resource

Additional configuration by ConfigMap and Secret resources located within the `openshift-image-registry` namespace

### Documentation:

[https://docs.openshift.com/container-platform/4.1/registry/configuring-registry-operator.html#registry-operator-configuration-resource-overview\\_configuring-registry-operator](https://docs.openshift.com/container-platform/4.1/registry/configuring-registry-operator.html#registry-operator-configuration-resource-overview_configuring-registry-operator)

### Image Registry Operator configuration parameters

The `configs.imageregistry.operator.openshift.io` resource offers the following configuration parameters.

| Parameter                    | Description                                                                                                                                                                               |
|------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>ManagementState</code> | <code>Managed</code> : The Operator updates the registry as configuration resources are updated.<br><code>Unmanaged</code> : The Operator ignores changes to the configuration resources. |
| <code>Removed</code>         | The Operator removes the registry instance and tear down any storage that the Operator provisioned.                                                                                       |
| <code>Logging</code>         | Sets <code>loglevel</code> of the registry instance.                                                                                                                                      |
| <code>HTTPSecret</code>      | Value needed by the registry to secure uploads, generated by default.                                                                                                                     |
| <code>Proxy</code>           | Defines the Proxy to be used when calling master API and upstream registries.                                                                                                             |
| <code>Storage</code>         | <code>StorageType</code> : Details for configuring registry storage, for example S3 bucket coordinates. Normally configured by default.                                                   |
| <code>Requests</code>        | API Request Limit details. Controls how many parallel requests a given registry instance will handle before queuing additional requests.                                                  |
| <code>DefaultRoute</code>    | Determines whether or not an external route is defined using the default hostname. If enabled, the route uses re-encrypt encryption. Defaults to false.                                   |
| <code>Routes</code>          | Array of additional routes to create. You provide the hostname and certificate for the route.                                                                                             |
| <code>Replicas</code>        | Replica count for the registry.                                                                                                                                                           |

# Red Hat Quay v3

## Quay v3 Key Features

- Full support for Docker Registry API v2\_s2
- Multi-arch (manifest) and Windows images
- Rebranded and rebased (RHEL)
- New config UI to save & upload Quay config
- Parallel upgrade (2x 5min downtime only)

## Red Hat Quay and OpenShift

Red Hat Quay and OpenShift are working well together already today. This includes both running Quay **on** and using Quay **with** OpenShift. Documentation and UX improvements for both coming soon.



- Customer facing blog post: <https://red.ht/2GOViSc>
- RH Internal blog post: <https://red.ht/2XXtFMF>
- Quay Mojo Group: <https://mojo.redhat.com/groups/quay>

# Moving components to dedicated Node pools

## Create a new Node pool

First, make a new MachineSet with a template that contains a custom label, like `role=logging`. Optionally customize the resources or security groups on this set of Nodes.

[Documentation: Roles in OpenShift](#)

## Update Node selectors

Change the node selectors to your new Node pool labels. This process uses different CRDs based on the component:

- [Cluster Monitoring](#) (docs)
- [Router](#) (lab)
- [Registry](#) (lab)
- [Logging](#) (docs)

```
# new Node pools added
$ oc get nodes
NAME
ip-10-0-132-151.us-east-2.compute.internal
Ip-10-0-137-86.us-east-2.compute.internal
ip-10-0-138-38.us-east-2.compute.internal
ip-10-0-139-204.us-east-2.compute.internal
ip-10-0-139-249.us-east-2.compute.internal
ip-10-0-144-70.us-east-2.compute.internal
ip-10-0-145-199.us-east-2.compute.internal

```

| NAME                                       | ROLE       |
|--------------------------------------------|------------|
| ip-10-0-132-151.us-east-2.compute.internal | worker     |
| Ip-10-0-137-86.us-east-2.compute.internal  | worker     |
| ip-10-0-138-38.us-east-2.compute.internal  | worker     |
| ip-10-0-139-204.us-east-2.compute.internal | worker     |
| ip-10-0-139-249.us-east-2.compute.internal | master     |
| ip-10-0-144-70.us-east-2.compute.internal  | master     |
| ip-10-0-145-199.us-east-2.compute.internal | master     |
|                                            | logging    |
|                                            | logging    |
|                                            | monitoring |
|                                            | monitoring |
|                                            | router     |
|                                            | router     |

# Additional Build Configurations

## Default build configurations for buildconfigs

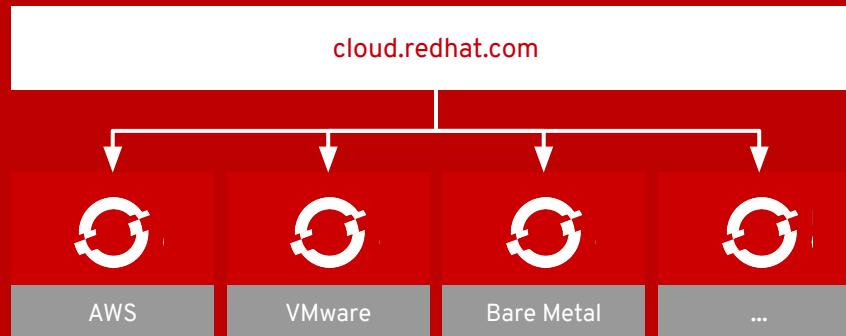
- Additional CAs to be trusted for image pull/push
- Proxy setting for image pull/push and source download
- Proxy settings for git commands
- Environment variables to set on all builds
- Docker labels to apply to resulting images
- Build resource requirements
- Default values to override on builds even if user has provided values on the buildconfig

```
apiVersion: config.openshift.io/v1
kind: Build
metadata:
  name: cluster
spec:
  additionalTrustedCA:
    name: trustedCAsConfigMap
  buildDefaults:
    defaultProxy: # http, https and no proxy
    gitProxy: # http, https and no proxy
    env: # key-values
    imageLabels:
    resources:
      limits: # cpu, memory
      requests: # cpu, memory
  buildOverrides:
    imageLabels:
    nodeSelector:
    tolerations:
```

```
oc edit build.config.openshift.io/cluster
```

# Cloud-like Simplicity, Everywhere

Full-stack automated operations across any on-premises,  
cloud, or hybrid infrastructure



# OpenShift Cluster Manager on cloud.redhat.com

## Automatic registration of OpenShift clusters

View cluster versions and capacity in one place, no matter what infrastructure you are running on. Integrated with RHSM.

## OpenShift Dedicated cluster management

Self-service cluster deployment, scaling, and management for OpenShift Dedicated coming soon.

## Azure Red Hat OpenShift

Information about these clusters will be coming at a later date.

## Hosted in the United States

Other geographies may come later. You can [opt-out](#) too.

The screenshot shows the Red Hat OpenShift Cluster Manager interface. At the top, there's a header with the Red Hat logo and 'RED HAT OPENSHIFT'. Below it, a 'Clusters' section has a 'Create Cluster' button. A table lists two clusters: 'prd-west-2100' and 'prd-west-2104', both marked as 'Self-managed'. To the right, a detailed view for 'prd-0501-2100' is shown, with tabs for 'Overview' and 'Resource Usage'. Under 'Resource Usage', there are two donut charts: one for CPU ('1.58 Cores used') and one for MEMORY ('12.38 GiB used'). The interface also includes 'Launch Console', 'Admin Credentials', and 'Actions' buttons.

# OpenShift Subscription Management

## Moves from node management to cluster management

Entitle clusters and not nodes. Nodes too dynamic. We do not block on usage. Requires telemeter Opt-In.

## Dynamically adds and removes nodes

UHC will dynamically add and remove nodes from your subscription allocations to the cluster in 24 hour intervals. This will move to instantaneous across the next several releases.



This cluster is overcommitting resources.

Please check the [Red Hat Customer Portal](#) to make sure all clusters are covered by subscriptions and [contact sales](#) if required.

Last checked: 5/19/2019, 2:20:00 AM

Generally Available

## Connected to the same backend as Subscription Portal and Satellite

Allocation numbers you see at [cloud.redhat.com](#) for OCP can be also seen on the subscription portal at [access.redhat.com](#)

## Removes OCP Infrastructure from the count

UHC will figure out which pods are your OCP infra pods and subtract out their usage from your core count so you are not charged.

Systems

Below is a list of systems for this account.

Filter by Name, UUID, or Cloud Provider

| <input type="checkbox"/>            | Name                                 |
|-------------------------------------|--------------------------------------|
| <input checked="" type="checkbox"/> | eb121bf1-aa59-422a-a417-2e5fcfa7ffd4 |

Show 100 entries



# Automated Container Operations

Fully automated day-1 and day-2 operations

INSTALL

DEPLOY

HARDEN

OPERATE

## AUTOMATED OPERATIONS

Infra provisioning

Full-stack deployment

Secure defaults

Multi-cluster aware

Embedded OS

On-premises and cloud

Network isolation

Monitoring and alerts

Unified experience

Audit and logs

Full-stack patch & upgrade

Signing and policies

Zero downtime upgrades

Vulnerability scanning

# Smarter Software Updates

## No downtime for well behaving apps

Applications with multiple replicas, using liveness probes, health checks and taints/tolerations  
Node Pools with more than one worker and slack resources

## Maintenance window for entire cluster

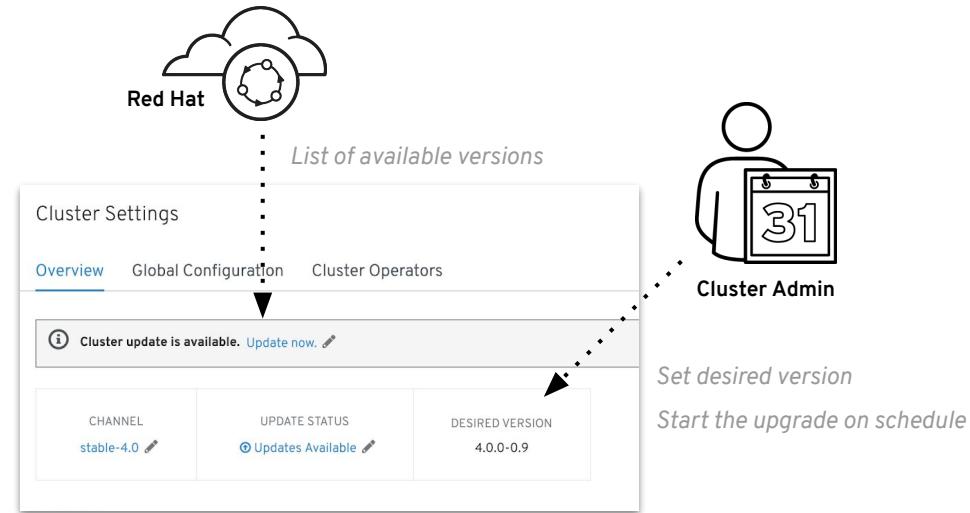
No need for separate windows for each component

## Upgrade runs completely on the cluster

No more long running processes on a workstation

## Constant health checking from each Operator

Operators are constantly looking for incompatibilities and issues that might arise



# Rolling Machine Updates

## Single-click updates

- RHEL CoreOS version & config
- Kubernetes core components
- OpenShift cluster components

## Configure how many machines can be unavailable

Set the “maxUnavailable” setting in the MachineConfigPool to maintain high availability while rolling out updates.

The default is 1.

## Machine Config Operator (MCO) controls updates

This is a DaemonSet that runs on all Nodes in the cluster. When you upgrade with `oc adm upgrade`, the MCO executes these changes.

The screenshot shows the Red Hat OpenShift web console with the URL `https://mcp-console.apps.ocp3.svc.cluster.local/`. The main title is "MCP master". Below it, there are tabs for "Overview" (which is selected), "YAML", and "Machine Configs".

**Machine Config Pool Overview:**

| TOTAL MACHINE COUNT | READY MACHINES | UPDATED COUNT | UNAVAILABLE COUNT |
|---------------------|----------------|---------------|-------------------|
| 3 machines          | 3 machines     | 3 machines    | 0 machines        |

**Machine Config Details:**

|                         |                                                                              |                                                                                                                                                                                                                            |
|-------------------------|------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NAME                    | master                                                                       | MAX UNAVAILABLE MACHINES                                                                                                                                                                                                   |
| LABELS                  | <code>operator.machineconfiguration.openshift.io/required-for-upgrade</code> | CURRENT CONFIGURATION                                                                                                                                                                                                      |
| ANNOTATIONS             | 0 Annotations                                                                | CURRENT CONFIGURATION SOURCE                                                                                                                                                                                               |
| MACHINE CONFIG SELECTOR | <code>Q machineconfiguration.openshift.io/role=master</code>                 | <code>MC 00-master</code><br><code>MC 01-master-container-runtime</code><br><code>MC 01-master-kubelet</code><br><code>MC 99-master-c3a87158-6aa3-11e9-9e74-06fb310be02-registries</code><br><code>MC 99-master-ssh</code> |

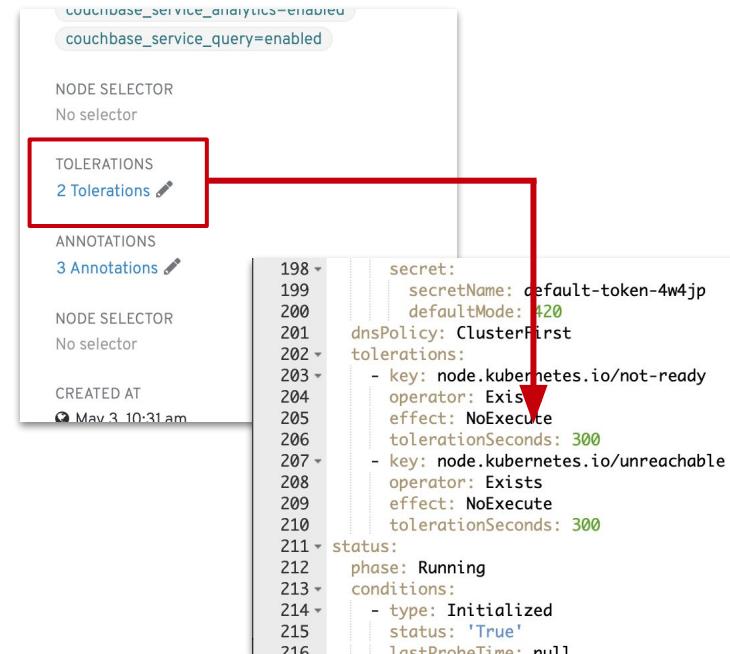
# Pod Eviction Behavior

## Eviction has been moved from Node Conditions to Taints

A new `DefaultTolerationSeconds` mutating admission plugin will add a 5 min eviction timeout unless specifically set by the Pod. This gives users more control vs the previous cluster-wide setting.

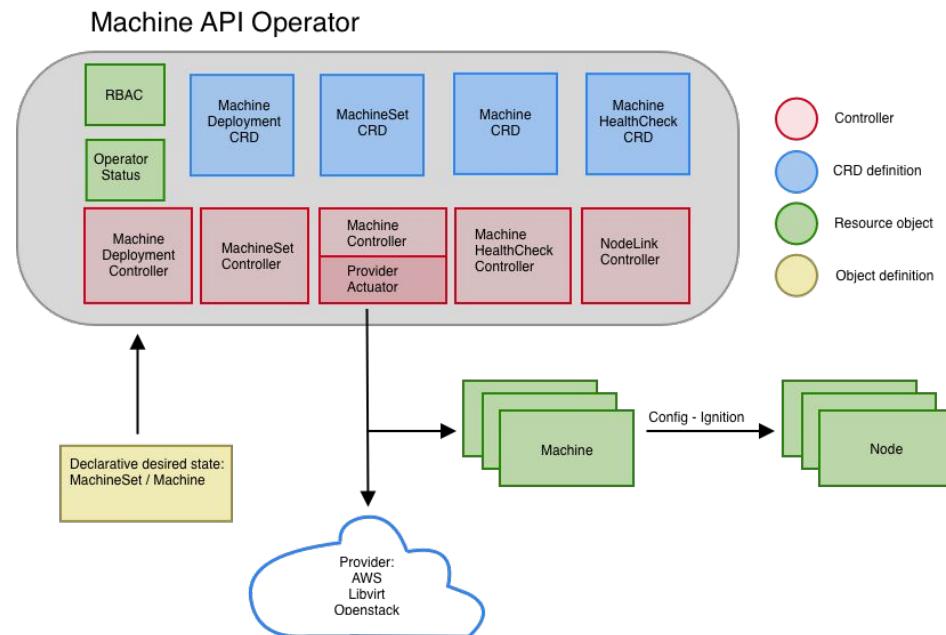
## Don't tolerate the `node.kubernetes.io/unreachable` taint

This taint is used by the cluster to evict and reschedule Nodes. If your Pod tolerates this, it will not be rescheduled on Node failure.



# Cloud API

- Provide a single view and control across multiple cluster types
- *Machine API*:
  - Set up definitions via CRDs
  - Machine: a node
  - MachineSet: think ReplicaSet
  - Actuators roll definitions across clusters
  - Nodes are drained before deletion
- *Cluster Autoscaler*: provide/remove additional nodes on demand
- AWS (4.1), Azure/GCP (target 4.2), VMWare (Future)



# Cluster Monitoring

## Cluster monitoring is installed by default

- Exposes resource metrics for Horizontal Pod Autoscaling (HPA) by default
  - HPA based on custom metric is tech preview
- No manual etcd monitoring configuration anymore
- New screens for managing Alerts & Silences
- More metrics available for troubleshooting purposes (e.g. HAProxy)
- Configuration via ConfigMaps and Secrets

The screenshot shows the Red Hat OpenShift Container Platform web interface. On the left, there's a sidebar with the Red Hat logo and the text "RED HAT OPENSHIFT Container Platform". Below this are several navigation items: "OperatorHub", "Operator Management", "Workloads", "Networking", "Storage", "Builds", "Monitoring" (which is currently selected), "Alerts", "Silences", "Metrics", "Dashboards", and "Compute". Under "Monitoring", there are sub-options: "Alerts", "Silences", "Metrics", and "Dashboards". On the right, the main content area is titled "Alerts" and includes a sub-link "Alertmanager UI". It says "Alerts help notify you when certain conditions in your environment are met. Learn more about how alerts are triggered". Below this, there are four alert entries:

| NAME                              | STATE  | SINCE                  |
|-----------------------------------|--------|------------------------|
| AL CPUThrottlingHigh              | Firing | Since Apr 29, 11:52 am |
| AL CPUThrottlingHigh              | Firing | Since May 2, 6:47 am   |
| AL CPUThrottlingHigh              | Firing | Since May 2, 11:52 am  |
| AL KubeDeploymentReplicasMismatch | Firing | Since May 2, 1:34 pm   |
| AL KubePodCrashLooping            | Firing | Since Apr 29, 2:52 pm  |

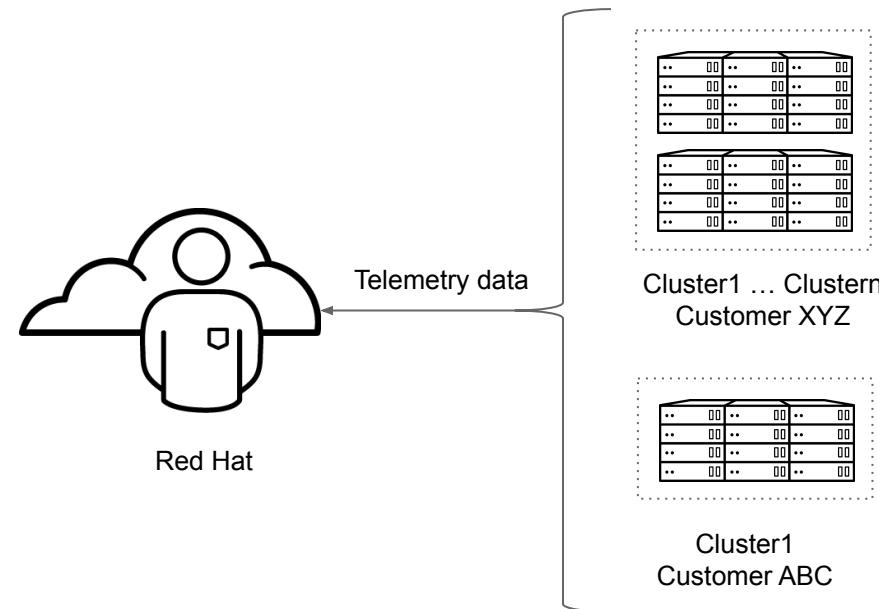
# Telemetry

**Collects anonymized data from any OpenShift 4 cluster deployment**

- Red Hat gains quality assurance with anonymous data reporting faults encountered during upgrade
- Show utilization of all your clusters at [cloud.redhat.com](http://cloud.redhat.com)
- Perform subscription management at [cloud.redhat.com](http://cloud.redhat.com)

*Opt-out is only available for self-managed OpenShift clusters but we strongly discourage that as you will lose all of the features described above.*

[Complete list of collected metrics](#)



# Cluster Logging

## Cluster Logging is lifecycle managed via Operator Lifecycle Management

- Install the Elasticsearch and Cluster Logging Operators from OperatorHub
- Create an instance of Cluster Logging. fluentd, Elasticsearch and Kibana (with Operators) are created
- Changing the out-of-box configuration:
  - CPU, memory requests and limits, PVC sizes etc can be changed by editing the Cluster Logging Operator YAML
- Direct Elasticsearch and Kibana Deployments to dedicated Nodes (recommended for production usage)

### Create Operator Subscription

Keep your service up to date by selecting a channel and approval strategy. The strategy determines either manual or automatic updates.

**Installation Mode \***

All namespaces   **Operator will be available in This mode**  A specific namespace on the cluster   **Operator will be available in a single namespace only.**

**Update Channel \***

preview

**Approval Strategy \***

Automatic    Manual

**Subscribe** **Cancel**

```
# configure via CRD
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: "openshift-logging"
spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 3
      resources:
        limits:
          cpu: 800m
          memory: 1Gi
        requests:
          cpu: 800m
          memory: 1Gi
    storage:
      storageClassName: gp2
      size: 100G
      redundancyPolicy: "SingleRedundancy"
  visualization:
    type: "kibana"
    kibana:
      replicas: 1
```

# Infrastructure MachineSets

## MachineSets that are purpose built for Infrastructure Services

- Elasticsearch, Prometheus, Router, Registry
- Out-of-box installer does not create a MachineSets dedicated for Infra services
- Create a MachineSet via console or cli and label them with desired roles
- Redeploy Infra Services with nodeSelector set to the designated role

The screenshot shows the Red Hat OpenShift Container Platform web interface. The left sidebar has 'Compute' selected under 'Machine Sets'. The main area shows a list of existing Machine Sets: 'demo-r6pf4-worker-us-east-1a', 'demo-r6pf4-worker-us-east-1b', 'demo-r6pf4-worker-us-east-1c', and 'demo-r6pf4-worker-us-east-1d'. A 'Create Machine Set' button is visible. On the right, a large code editor displays the YAML configuration for a new Machine Set:

```

apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <clusterID>
    name: <clusterID>-<role>-us-east-1a
    namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <clusterID>
      machine.openshift.io/cluster-api-machine-role: <role>
      machine.openshift.io/cluster-api-machine-type: <role>
      machine.openshift.io/cluster-api-machineset: <clusterID>-<role>-us-east-1a
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <clusterID>
        machine.openshift.io/cluster-api-machine-role: <role>
        machine.openshift.io/cluster-api-machine-type: <role>
        machine.openshift.io/cluster-api-machineset: <clusterID>-<role>-us-east-1a
    spec:
      selector:
        labels:
          node-role.kubernetes.io/<role>: ""
      providerSpec:
        ami:
          id: <amiID>
          apiVersion: awsproviderconfig.openshift.io/v1beta1
          blockDevices:
            - ebs:
                iops: 0
                volumeSize: 120
                volumeType: gp2
            credentialSecret:
              name: aws-cloud-credentials
              deviceIndex: 0
              iamInstanceProfile:
                id: <clusterID>-worker-profile
                instanceType: m4.large
            kind: AWSMachineProviderConfig
            placement:
              availabilityZone: us-east-1a
              region: us-east-1
              securityGroups:
                - filters:
                    - name: tag:Name

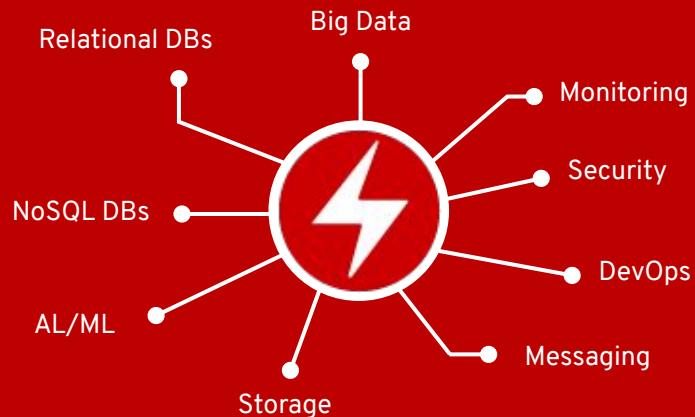
```

A red oval highlights the 'node-role.kubernetes.io/<role>: ""' line in the 'spec.selector.labels' section of the YAML code.

[Documentation: Creating Infrastructure MachineSets](#)

# A broad ecosystem of workloads

Operator-backed services allow for a  
SaaS experience on your own infrastructure



# Red Hat Certified Operators

## DEVOPS



## APM



## DATA SERVICES



## DATABASE



## SECURITY



## STORAGE



# OperatorHub data sources

## Requires an online cluster

- For 4.1, the cluster must have connectivity to the internet
- Later 4.x releases will add offline capabilities

## Operator Metadata

- Stored in quay.io
- Fetches channels and available versions for each Operator

## Container Images

- Red Hat products and certified partners come from RHCC
- Community content comes from a variety of registries

The screenshot shows the OperatorHub interface within a web browser. The top navigation bar includes a dropdown for 'Project: default', a search bar, and a 'Developer Catalog' link. Below this is a section titled 'OperatorHub' with a sub-section 'Discover Operators from the Kubernetes community and Red Hat partners, curated by Red Hat. Operators can be installed on your clusters to provide optional add-ons and shared services to your developers. Once installed appear in the Developer Catalog, providing a self-service experience.' A sidebar on the left lists categories such as All Items, AI/Machine Learning, Application Monitoring, Big Data, Database, Developer Tools, Integration & Delivery, Logging & Tracing, Monitoring, Networking, OpenShift Optional, Security, Security Policy Management, Storage, Streaming & Messaging, and Other. Under 'All Items', there are two tabs: 'All Items' (selected) and '43 items'. The main content area displays a grid of operator cards, each with a thumbnail, name, provider, and a brief description. Some operators are marked as 'Community' while others are 'Provided by Red Hat, Inc.'. The cards include: AMQ Streams (provided by AppDynamics LLC), AppDynamics ClusterAgent (provided by AppDynamics LLC), Aqua Security Operator (provided by Aqua Security, Inc.), Automation Broker Operator (provided by Red Hat, Inc.), Camel-K Operator (provided by The Apache Software Foundation), CockroachDB (provided by Helm Community), Community Jaeger Operator (provided by CNCF), Couchbase Operator (provided by Couchbase), Crunchy PostgreSQL Enterprise (provided by Crunchy Data), Descheduler (provided by Red Hat), Federation (provided by Red Hat), Elasticsearch Operator (provided by Red Hat, Inc.), FederationAI (provided by ProphetStar Data Services, Inc.), FederatorAI (provided by ProphetStar Data Services, Inc.), Hazelcast Operator (provided by Hazelcast, Inc.), and Install Hazelcast Enterprise cluster.

# Services ready for your developers

## New Developer Catalog aggregates apps

- Blended view of Operators, Templates and Broker backed services
- Operators can expose multiple CRDs. Example:
  - MongoDBReplicaSet
  - MongoDBSharded Cluster
  - MongoDBStandalone
- Developers can't see any of the admin screens

## Self-service is key for productivity

- Developers with access can change settings and test out new services at any time

The screenshot shows the Red Hat Developer Catalog interface. At the top, it says "Project: production-api-backend". Below that is the title "Developer Catalog" and a sub-instruction: "Add shared apps, services, or source-to-image builders to your project from the Developer Catalog. Cluster admins can install additional apps which will show up here automatically." On the left, there's a sidebar with categories: "All items" (selected), "Languages", "Databases", "Middleware", and "Other". A "Filter by keyword..." input field is also present. The main area is a grid of service cards. The first row contains four cards: ".NET Core" (Apache HTTP Server icon), "Apache HTTP Server (httpd)" (Kafka icon), "Kafka" (Kafka Connect icon), and "Kafka Connect" (Kafka icon). The second row contains four cards: "Kafka Connect S2I" (Kafka User icon), "Kafka MirrorMaker" (Kafka Topic icon), "Kafka Topic" (Kafka User icon), and "Kafka User" (Kafka User icon). The third row contains four cards: "MongoDB Replica Set" (MongoDB Sharded Cluster icon), "MongoDB Sharded Cluster" (MongoDB Standalone icon), "MongoDB Standalone" (MongoDB Deployment icon), and "NGINX" (NGINX icon). Each card has a small description below its icon.

# Operators as a First-Class Citizen

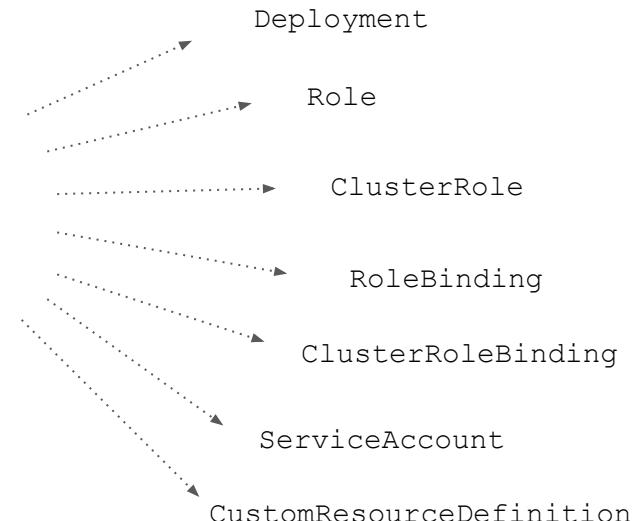


YourOperator v1.1.2  
Bundle



OPERATOR  
**LIFECYCLE MANAGER**

Operator Deployment  
Custom Resource  
Definitions  
RBAC  
API Dependencies  
Update Path  
Metadata



# Operator Lifecycle Management

Operator Catalog

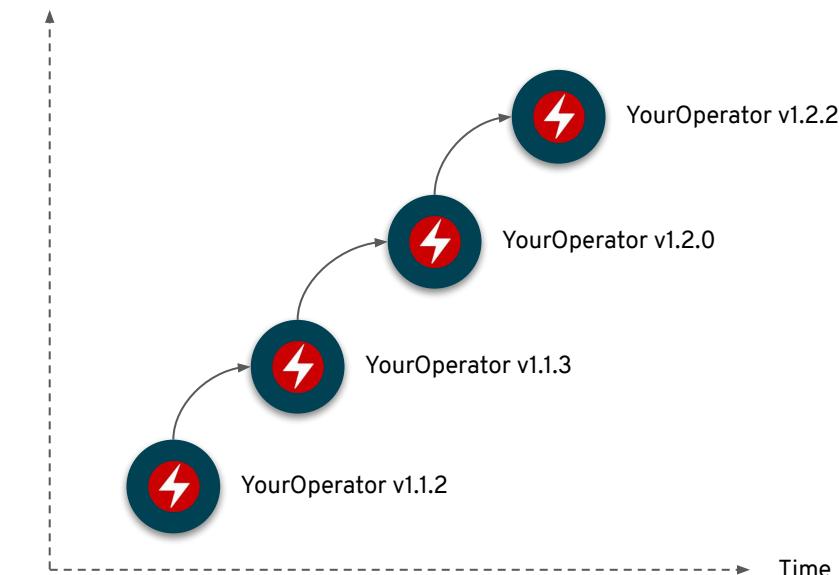


 **OPERATOR LIFECYCLE MANAGER**



Subscription for  
YourOperator

Version



# Operator Lifecycle Management

Operator Catalog

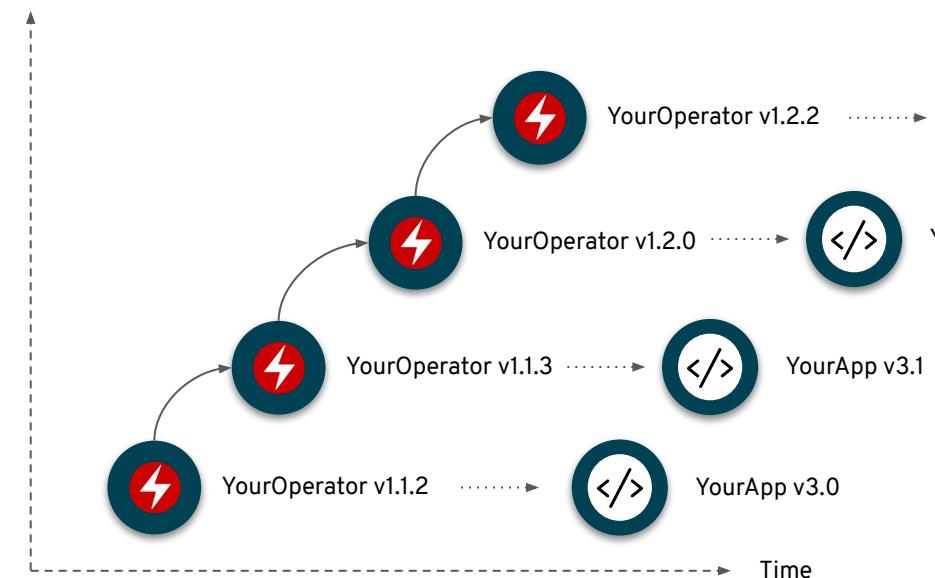


**OPERATOR  
LIFECYCLE MANAGER**



Subscription for  
YourOperator

Version



# Operator Upgrade in Detail

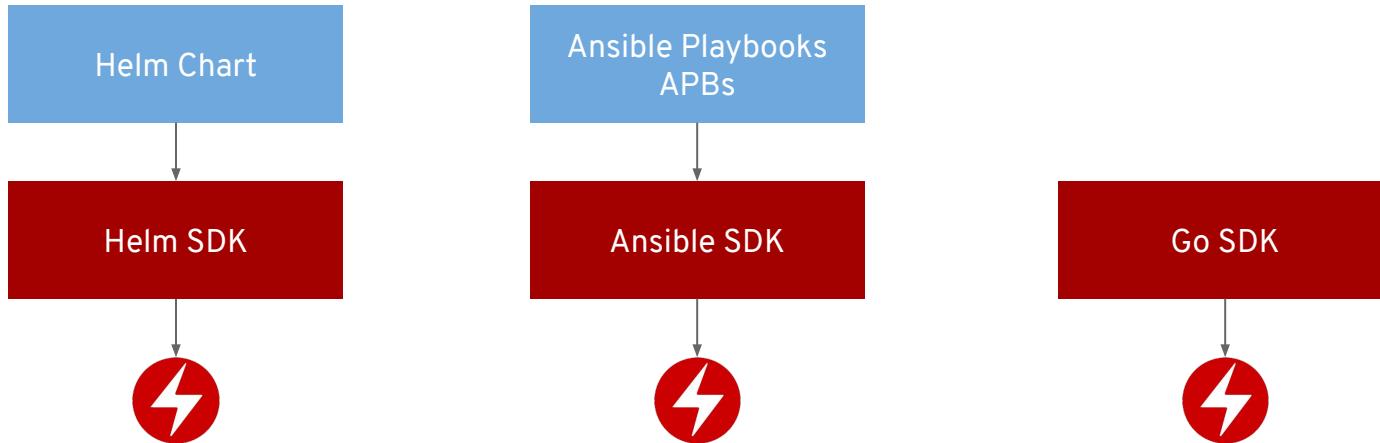
## OperatorHub facilitates upgrades of installed Operators

- Manual or automatic modes can be chosen per Operator
- The Operator itself is upgraded by OLM via Deployment and a regular rolling upgrade
- The objects managed by the Operator use built in mechanisms to maintain HA
  - Deployments/StatefulSets
  - affinity/anti-affinity
  - taints/tolerations
  - PodDisruptionBudgets
- Behavior is dependent on the maturity of the Operator
- Optional cluster components like Cluster Logging are well behaved during upgrades

The screenshot shows the Red Hat OpenShift Container Platform interface. The top navigation bar includes the Red Hat logo, 'RED HAT OPENSHIFT Container Platform', and a dropdown for 'kube:admin'. The main menu on the left has sections for 'Home', 'Catalog' (with 'Developer Catalog', 'Installed Operators', and 'OperatorHub' sub-options), and 'Operator Management' (selected). Under 'Workloads', there are links for 'Pods', 'Deployments', 'Deployment Configs', 'Stateful Sets', 'Secrets', 'Config Maps', 'Cron Jobs', 'Jobs', 'Daemon Sets', 'Replica Sets', 'Replication Controller', 'Horizontal Pod Affinity', and 'Networking'. The central area is titled 'Operator Management' and contains tabs for 'Operator Subscriptions', 'Operator Catalogs', and 'Install Plans'. A sub-section titled 'Operator Subscriptions keep your services up to date by tracking a channel in a package. The approval strategy determines either manual or automatic updates.' is shown. A 'Create Subscription' button is at the top of the list. A table lists four operator subscriptions:

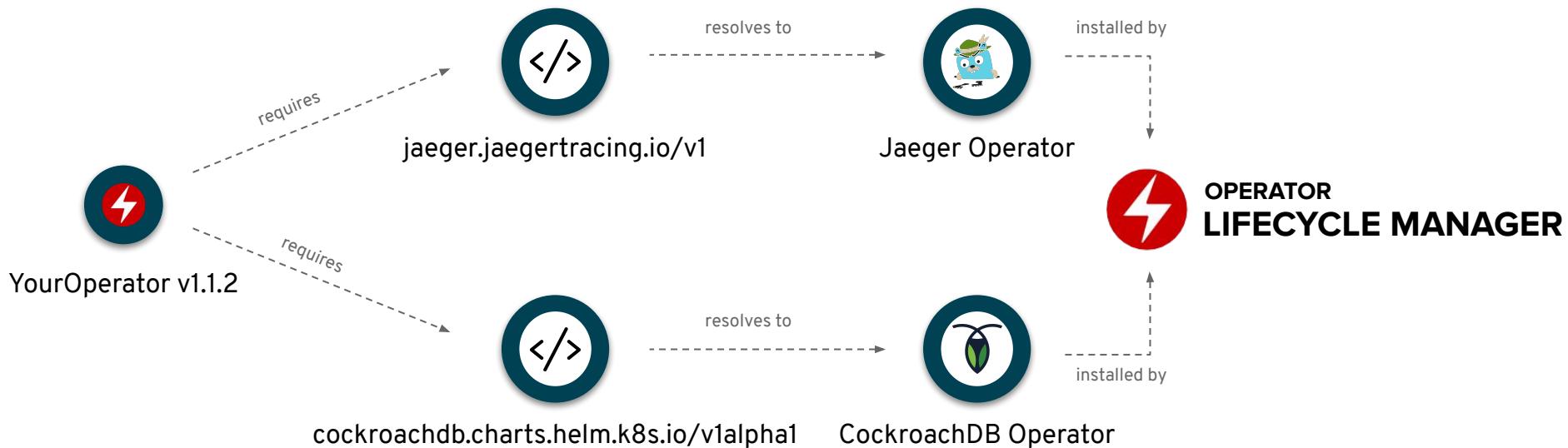
| NAME                           | NAMESPACE              | STATUS     | CHANNEL | APPROVAL STRATEGY |
|--------------------------------|------------------------|------------|---------|-------------------|
| amq-streams                    | openshift-operators    | Up to date | stable  | Automatic         |
| cockroachdb                    | openshift-operators    | Up to date | stable  | Automatic         |
| codeready-workspaces           | codeready              | Up to date | final   | Automatic         |
| couchbase-enterprise-certified | robszumski-api-backend | Up to date | preview | Automatic         |

# Build Operators for your apps



# Depend on other Operators

Operator Framework Dependency Graphs



# Red Hat Middleware

## Same experience as 3.x for developers

- Admins install Service Brokers via OperatorHub
- Devs consume via Developer Catalog

## Transitioning to Operators

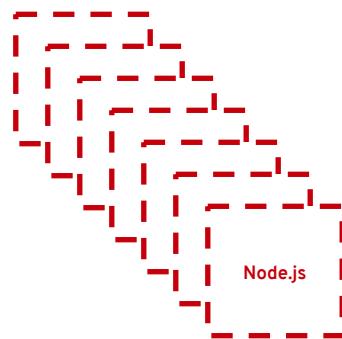
- First Operators are out
  - AMQ Streams (Kafka)
  - Fuse Online
  - CodeReady Workspaces
  - Business Automation (Tech Preview)
  - Data Grid
- More to follow in 2019
  - Red Hat Integration - July:
    - AMQ Interconnect, AMQ Broker
    - 3scale API Management
    - Apicurio API Designer
  - Business Automation - July (GA)
  - Red Hat Application Runtimes
    - MW Component Operator - July

# Red Hat Universal Base Image

Enable an ecosystem of freely distributable operators for Kubernetes/OpenShift



Base  
Images

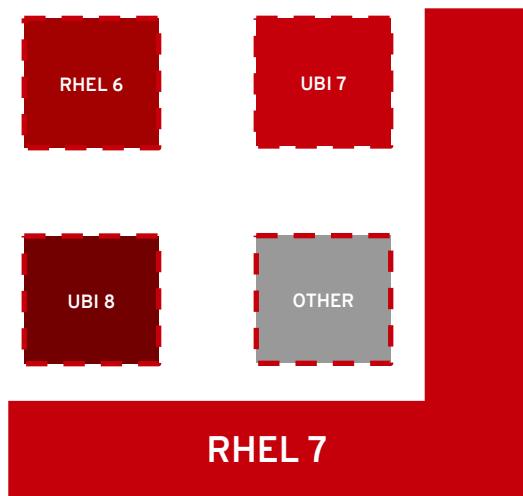


Pre-Built  
Language  
Images

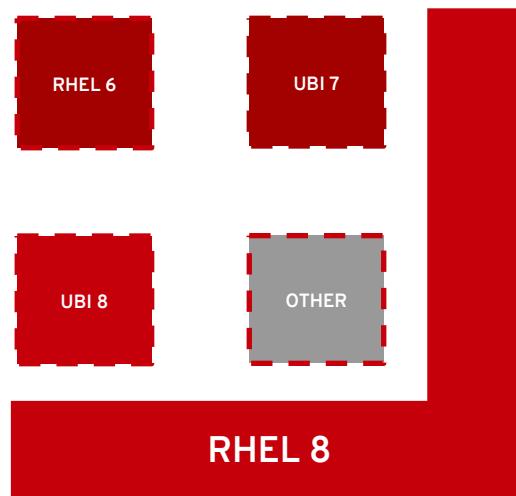


Package  
Subset

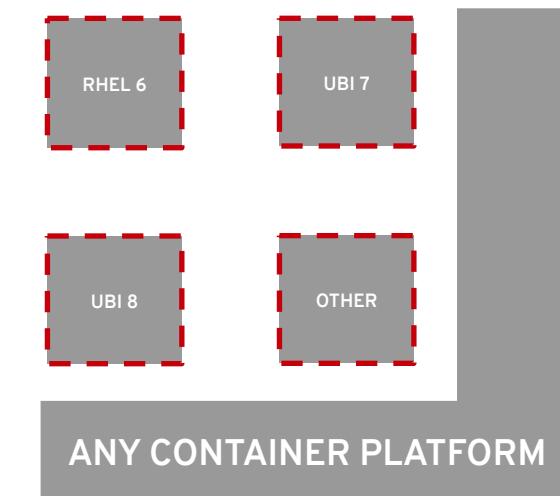
# UBI and Host interactions



Red Hat Enterprise Linux 7



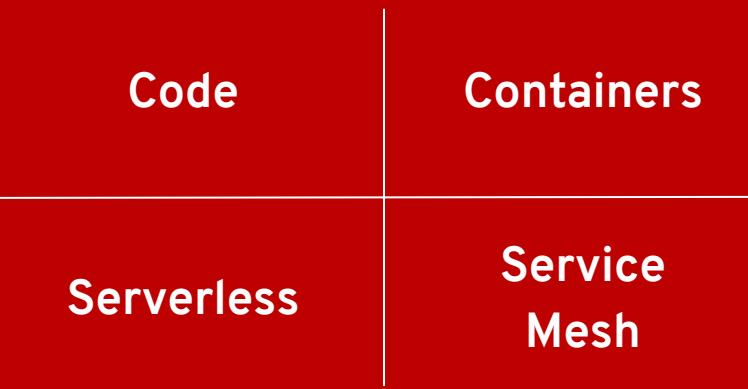
Generally Available



Like any community distro

# Next wave of developer tools

OpenShift has all of the latest tools to make  
your devs more productive



# Cloud-native CI/CD with OpenShift Pipelines

- Based on Tekton Pipelines
- Runs serverless (no babysitting!)
- Containers as building blocks
- Deploy to multiple platforms
- Standard CRDs
- Pipelines portable to any Kubernetes
- Available in OperatorHub

Project: Default

Pipelines

| NAME         | LAST PIPELINE RUN  | LAST RUN STATUS | TASK COMPLETED | LAST RUN STARTED |
|--------------|--------------------|-----------------|----------------|------------------|
| P Pipeline-a | P Pipeline-run-a-1 | Running         | 2 of 4         | 3 seconds ago    |
| P Pipeline-B | P Pipeline-run2    | Running         | 3 of 5         | 2 minutes ago    |
| P Pipeline-C | P Pipeline-run23   | Succeeded       | 3 of 3         | 4 minutes ago    |
| P Pipeline-D | P Pipeline-run4    | Failed          | 2 of 4         | 6 minutes ago    |
| P Pipeline-E | P Pipeline-run34   | Succeeded       | 2 of 2         | 8 minutes ago    |

Storage

Builds

Monitoring

Developer Tools

Integration & Delivery

Logging & Tracing

Monitoring

Networking

OpenShift Optional

Knative Serving Operator

provided by Red Hat

Knative Serving builds on Kubernetes to support deploying and serving of serverless applications and functions.

Community

OpenShift Pipelines Operator

provided by Red Hat, Inc.

OpenShift Pipelines is a cloud-native CI/CD solution for building pipelines using Tekton.

Dev Preview on OCP 4.1 (June)

# Cloud-native CI/CD with OpenShift Pipelines

```
apiVersion: tekton.dev/v1alpha1
kind: Pipeline
metadata:
  name: funky-deploy-pipeline
spec:
  resources:
    ... # git, images, etc
  tasks:
    - name: build-app
      taskRef:
        name: mvn-build
    ...
    - name: build-image
      taskRef:
        name: s2i-java
    ...
    - name: deploy
      taskRef:
        name: openshift-cli
    ...
```

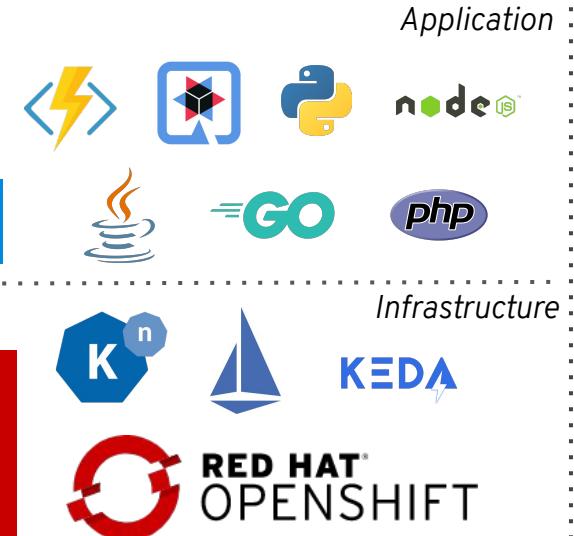
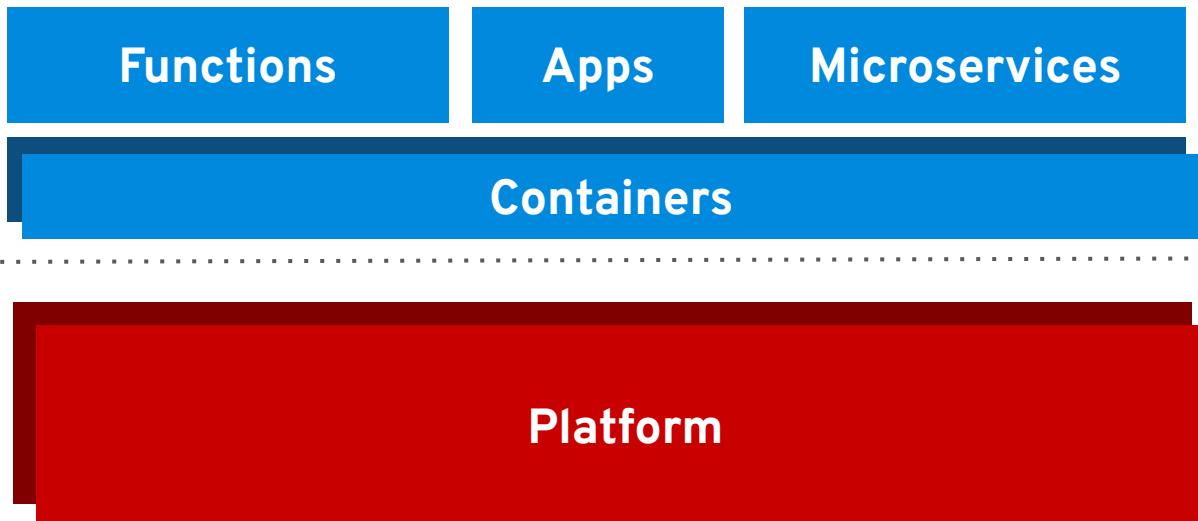
Inputs (e.g. git repo) to and outputs (e.g. images) from the pipeline

Provided task library:  
s2i, buildah, oc, jib, kaniko, etc

User can create custom ones



# OpenShift Serverless





# OpenShift Serverless

## Key Features

- Familiar to Kubernetes users. Native.
- Scale to 0 and autoscale to N based on demand
- Applications and functions. Any container workload.
- Powerful eventing model with multiple event sources.
- Operator available via OperatorHub
- Knative v0.6 (v1beta1 APIs)
- No vendor lock in

## Learn more

<https://openshift.com/learn/topics/knative>

<http://bit.ly/knative-tutorial>

The screenshot shows the Red Hat OpenShift Container Platform interface. On the left is a sidebar with 'kiali' branding and links for Graph, Namespaces, Workloads, Services, Istio Config, and Distributed Tracing. The main content area has a header 'Red Hat OpenShift Container Platform' and a message 'You are logged in as a temporary administrative user. Update your password now.' Below the header, it says 'Project: openshift-operators'. The main content area contains several cards:

- Cloud Provider**: AMQ Streams provided by Red Hat, Inc.
- Developer Tools**: Red Hat AMQ Streams is a massively scalable, distributed, and high performance data stream.
- Integration & Delivery**: Business Automation Operator can deploy RHPAM/RHDM environments in the form of KieApp objects.
- Logging & Tracing**: Business Automation Operator can support monitoring and logging.
- Monitoring**: Networking
- Networking**: OpenShift Optional
- Operator Management**: Security
- Storage**: Security Policy Management
- Streaming & Messaging**: Storage

A specific card for the **Knative Serving Operator** is highlighted with a red box. It includes a 'Community' section, a description 'Knative Serving Operator provided by Red Hat', and a note 'Knative Serving builds on Kubernetes to support deploying and serving of'. The 'Installed' checkbox is checked. Below the card is a diagram illustrating the Knative architecture, showing components like knative-ingressgateway, activator, knative-serving, helloworld-go-00001, and autoscaler, along with traffic flow and metrics graphs.



```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
  labels:
    app: guestbook
spec:
  selector:
    matchLabels:
      app: guestbook
      tier: frontend
  replicas: 1
  template:
    metadata:
      labels:
        app: guestbook
        tier: frontend
    spec:
      containers:
        - image: markusthoemmes/guestbook
          name: guestbook
          resources:
            requests:
              cpu: 100m
              memory: 100Mi
          env:
            - name: GET_HOSTS_FROM
              value: dns
          ports:
            - containerPort: 80

```

## Kubernetes

```

---
apiVersion: v1
kind: Service
metadata:
  name: frontend-service
  labels:
    app: guestbook
    tier: frontend
spec:
  ports:
    - port: 80
  selector:
    app: guestbook
    tier: frontend
---
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: frontend-route
spec:
  to:
    kind: Service
    name: frontend-service

```

**53 lines**

## Knative

```

apiVersion: serving.knative.dev/v1alpha1
kind: Service
metadata:
  name: frontend
spec:
  template:
    metadata:
      labels:
        app: guestbook
        tier: frontend
    spec:
      containers:
        - image: markusthoemmes/guestbook
          resources:
            requests:
              cpu: 100m
              memory: 100Mi
          env:
            - name: GET_HOSTS_FROM
              value: dns
          ports:
            - containerPort: 80

```

**22 lines**



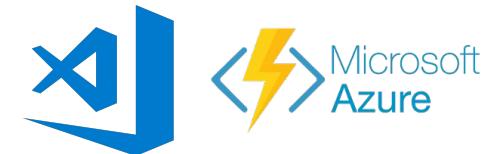
# OpenShift Serverless + Azure Functions

## Key Features

- Enable FaaS in OpenShift
- Familiar developer experience using VS Code and Azure CLI
- Polling based auto-scaling for Azure Queues, Kafka...
- Reuse Knative event sources, HTTP auto-scaling
- On premise or Any cloud.

## Learn more

<https://github.com/kedacore/keda>



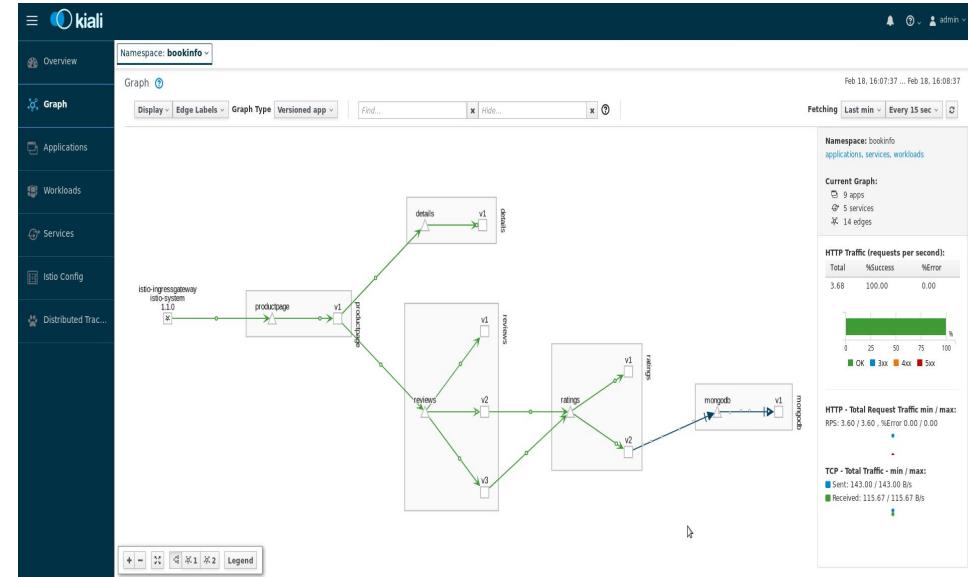
In partnership with



# Red Hat Service Mesh

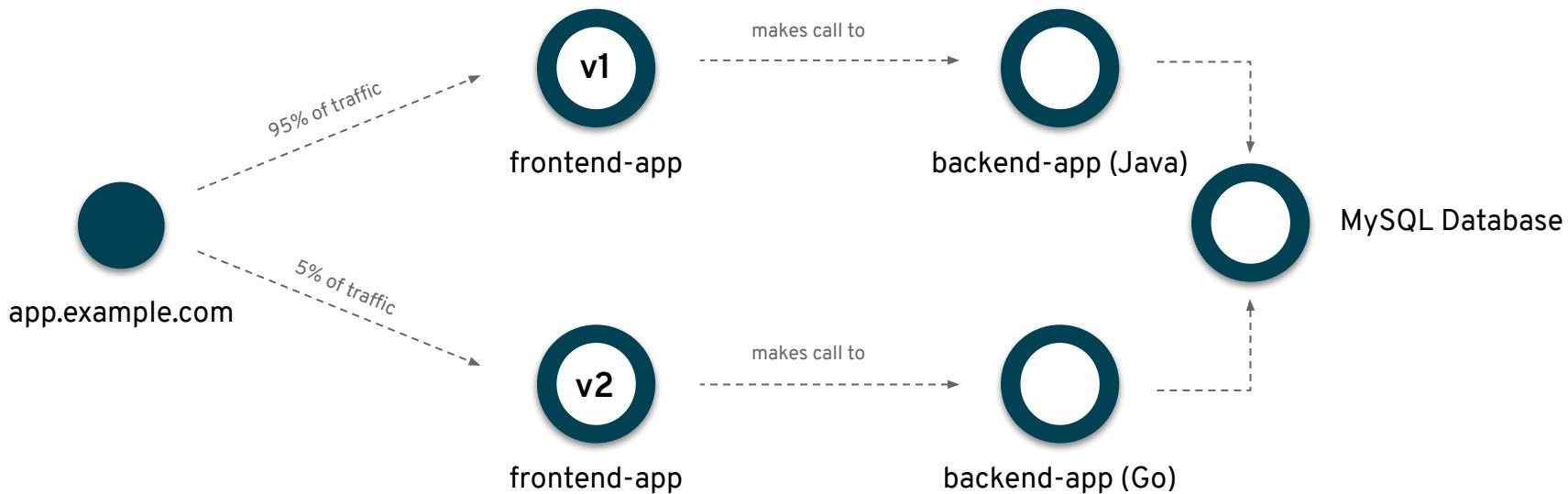
## Key Features

- A dedicated network for service to service communications
- Observability and distributed tracing
- Policy-driven security
- Routing rules & chaos engineering
- Powerful visualization & monitoring
- Will be available via OperatorHub

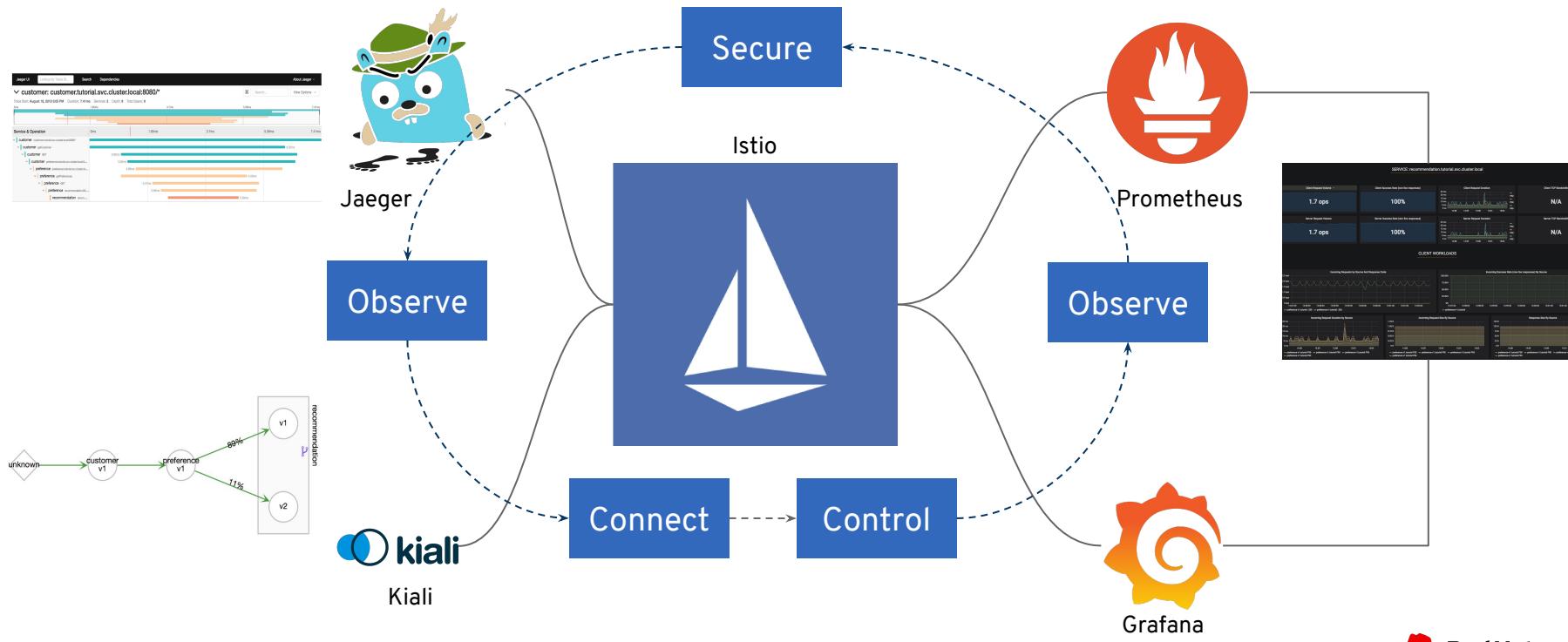


# Control Traffic Flow

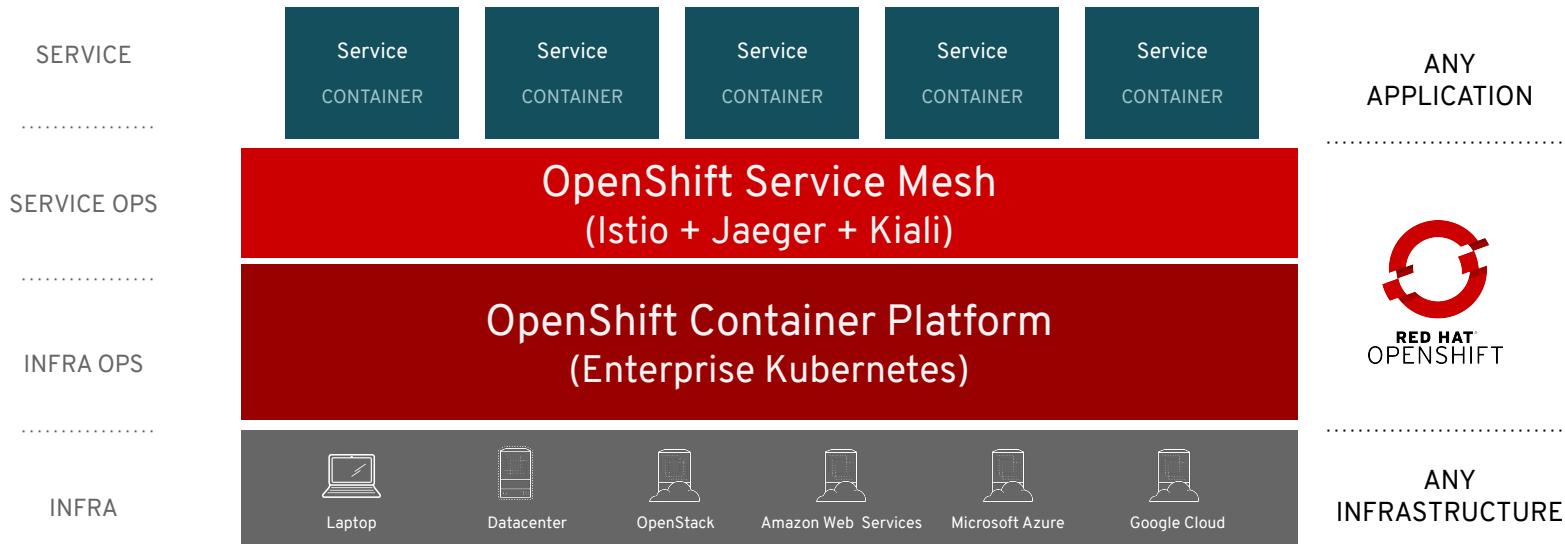
Control flow of traffic between application components



# Red Hat Service Mesh



# Red Hat Service Mesh



# CodeReady Workspaces

**The collaborative OpenShift-Native IDE.** Free for any customer of OpenShift Dedicated or OpenShift Container Platform.

## Container Workspaces



Workspace replicas to end “works on my machine” and enable team collaboration.

## DevOps Integrations



Reference developer workspaces from any issue, failed build, or git notification.

## Protect Source Code

Full access to source code without any of it landing on hard-to-secure laptops.

Based on the open Eclipse Che project

Red Hat Linux and Application Infrastructure

Plugin model for extensibility

Serverless support (coming later)

**Use It To:** Replace VDI for devs, and enable true container-based DevOps.

# CodeReady Workspaces

Collaborative web IDE

Supported Eclipse Che

Available in OperatorHub

Included in OCP/OSD SKU

The screenshot shows the Red Hat OpenShift Container Platform interface. On the left, the navigation sidebar includes Home, Projects, Status, Search, Events, Catalog (with OperatorHub selected), Operator Management, Workloads, Networking, Storage, Builds, Monitoring, Compute, and Administration. The main content area is titled "OperatorHub" and displays the "Red Hat CodeReady Workspaces" operator. The operator details are as follows:

- Project:** pipelines-tutorial
- Version:** 1.1.0
- Provider:** Red Hat, Inc.
- Repository:** <https://github.com/eclipse-che/operator>
- Container Image:** registry.access.redhat.com/codeready-workspaces/server-operator:1.1
- Created At:** 2019-03-06 11:59:59
- Support:** Red Hat, Inc.

The "Install" button is highlighted in blue. To the right of the operator details, there is a "Pre-Reqs" section with instructions for granting cluster roles to the operator service account. Below this, there are two code snippets for creating cluster roles and role bindings. A note at the bottom states that \${NAMESPACE} is an OpenShift project where the operator was installed.



# Visual Studio Code Extensions

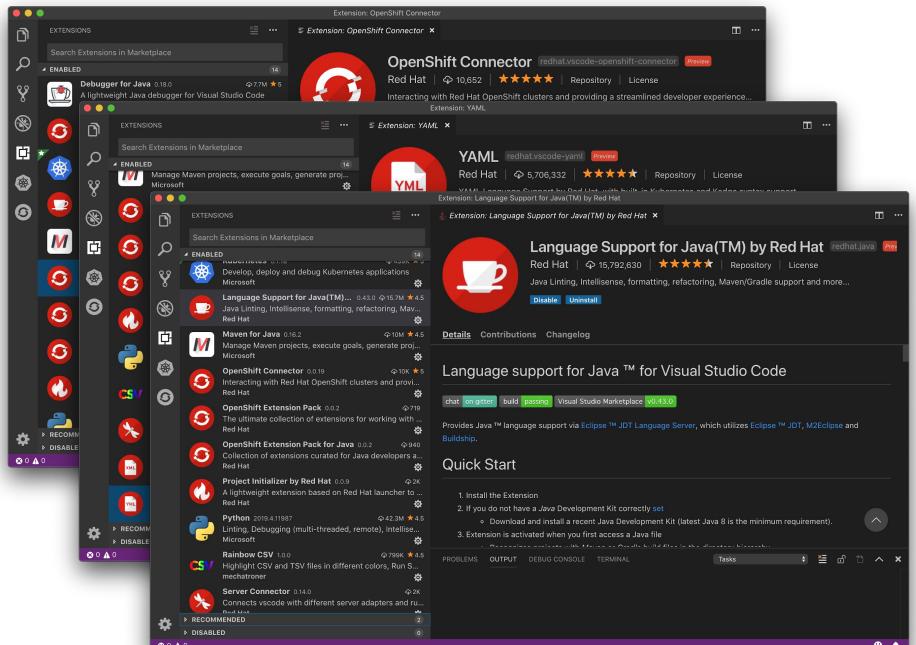
## Extremely popular plugins

Our first extension, Language Support for Java™ by Red Hat, was published in Sept., 2016 as an experiment.

Over 50 releases later, it's been downloaded nearly 16 million times by over 2.5 Million developers!

## More coming soon

We've been adding more extensions to help developers using VS Code have a fantastic experience when coding in Java, XML, Yaml, etc., or when working with OpenShift or other technologies where Red Hat is the expert.



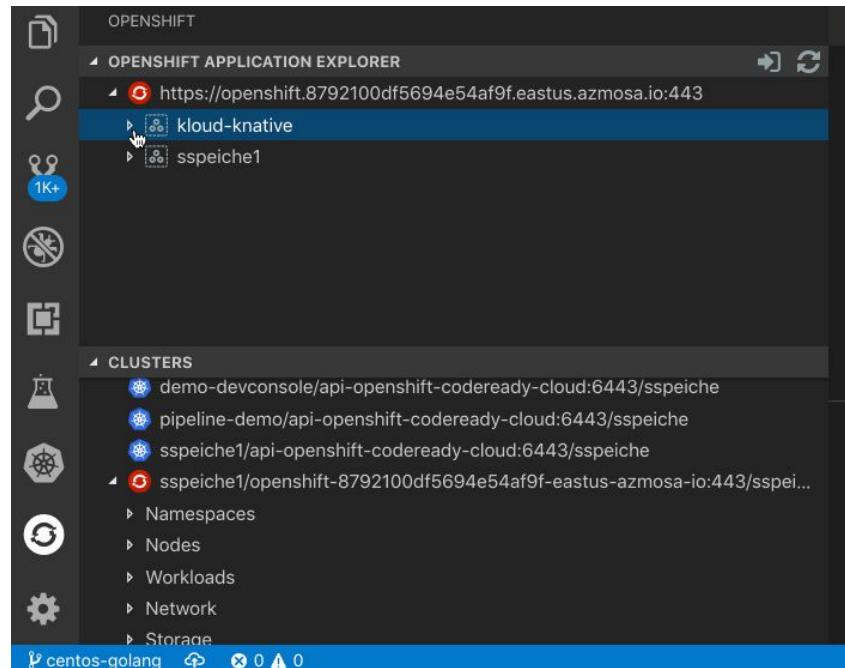
# VS Code Kubernetes Extension

## Kubernetes Extension Improvements

- Collaboration spearheaded by Red Hat and OpenShift needs
- Many improvements around:
  - Non-cluster-admin use cases
  - Auto-hide Helm features when no Tiller installed
  - Add nodes to navigator

## OpenShift Improvements

- OpenShift logo on OpenShift clusters
- Add: Routes, DeploymentConfig, Projects, ImageStreams
- Ability to set Project context



# A developer-focused command-line tool for rapid development iterations on OpenShift

`$odo create`

Create app from supported runtimes

`$odo push`

Build and deploy app from current directory

`$odo watch`

Sync local changes to running pods on OpenShift

The screenshot shows the Red Hat OpenShift Container Platform web interface. At the top, there's a navigation bar with the Red Hat logo and "OpenShift Container Platform". On the right side of the top bar, there are icons for documentation, help, and user "siamak". Below the top bar, there's a main header with "Project: pipelines-tutorial" and a dropdown for "Add". A secondary navigation bar on the left has "Home", "Catalog", and "Developer Catalog" options. The "Catalog" option is currently selected. On the right, there are links for "Documentation", "Command Line Tools" (which is highlighted with a red box), and "About".

This screenshot shows the "Command Line Tools" page within the Red Hat OpenShift Container Platform. The page title is "Command Line Tools". It features two main sections: "oc - OpenShift Command Line Interface (CLI)" and "odo - Developer-focused CLI for OpenShift".  
**oc - OpenShift Command Line Interface (CLI)**  
With the OpenShift command line interface, you can create applications and manage OpenShift projects from a terminal.  
The oc binary offers the same capabilities as the kubectl binary, but it is further extended to natively support OpenShift Container Platform features.  
[Download oc](#) | [Copy Login Command](#)  
  
**odo - Developer-focused CLI for OpenShift**  
Tech Preview  
OpenShift Do (odo) is a fast, iterative, and straightforward CLI tool for developers who write, build, and deploy applications on OpenShift.  
odo abstracts away complex Kubernetes and OpenShift concepts, thus allowing developers to focus on what is most important to them: code.  
[Download odo](#)  
  
At the bottom right of the page, there's a "Close" button.

# odo: OpenShift's Dev-focused CLI

A developer-focused command-line tool for rapid development iterations on OpenShift.

Available for download from Web Console

```
$ odo create php frontend  
Component 'frontend' was created.  
To push source code to the component run 'odo push'  
  
$ odo push  
Pushing changes to component: frontend  
  
$ odo url create  
frontend - http://frontend-myapp.192.168.99.100.nip.io  
  
$ odo watch  
Waiting for something to change in /dev/frontend
```

odo - Developer-focused CLI for OpenShift

Tech Preview

OpenShift Do (odo) is a fast, iterative, and straightforward CLI tool for developers who write, build, and deploy applications on OpenShift.

odo abstracts away complex Kubernetes and OpenShift concepts, thus allowing developers to focus on what is most important to them: code.

[Download odo](#) ↗

**Use It To:** Enable the 'git push' flow developers love, but with OpenShift Kubernetes.

# CodeReady Containers

Provides a pre-built development environment based on **Red Hat Enterprise Linux** and **OpenShift** for quick container-based application development. Use with OpenShift on-premises or cloud.

## Internal Alpha Details

- Linux (KVM) provides a single machine (node) instance
- Commands: setup, start, stop, delete
- Internal-only until pull secret is externalized / configured
- PoCs exist for Windows and MacOS (VirtualBox)

[Internal Alpha email](#)

**Use It To:** Simplify direct-to-OpenShift 4 development on laptops.

# Hosted OpenShift

Get the best of OpenShift without being on call



# Hosted OpenShift Benefits

## OPENSHIFT CONTAINER PLATFORM

Full Stack  
Automated

Pre-existing  
Infrastructure

Simplifies deployment & management of clusters  
Customer managed services & infrastructure  
Cluster provisioning

“Skip the on-call rotation”

Red Hat engineers keep you up to date

Expand capacity without hassle

## HOSTED OPENSHIFT

Azure Red Hat  
OpenShift

Deploy directly from the  
Azure console.

Jointly managed by Red  
Hat and Azure engineers.

Free your team from the  
distraction of ops

OpenShift  
Dedicated

Powerful cluster, no  
maintenance needed

Managed by Red Hat  
engineers and support

Free your team from the  
distraction of ops

# Azure Red Hat OpenShift

Jointly engineered, operated, and supported by both Microsoft and Red Hat with an integrated support experience

**Experience OpenShift as a native Microsoft Azure service.**

- Create fully managed OpenShift clusters in minutes using `az openshift create`
- Add or remove compute nodes to match resource demand using `az openshift scale`
- 99.9% SLA
- Will soon inherit Azure regulatory compliance
- Pricing available at

<https://azure.microsoft.com/en-us/pricing/details/openshift/>



# OpenShift Dedicated

## Dedicated with OpenShift 3

Available today, hosted on Amazon Web Services

Consumption based billing now available

Bring Your Own Cloud Account

## Dedicated with OpenShift 4

Initial availability June 2019

Broader availability in fiscal Q2

### **OperatorHub**

Red Hat products and certified Operators will be added in a curated catalog later in the year.

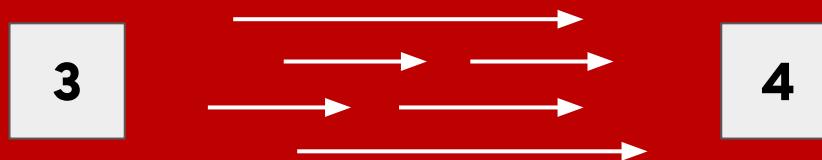
The Service Catalog and Brokers will not migrate to Dedicated due to their deprecation.

### **Connected to [cloud.redhat.com](https://cloud.redhat.com)**

Clusters will appear beside other self-managed installs

# Migrating to OpenShift 4

Tooling and advice for moving from OpenShift 3.x to 4.x



# App migration experience

## Using open source tooling based on Velero

Velero is an upstream project previously known as Ark. Check out [this video](#) if you are curious and want to get a sneak peek at our capabilities.

## What's moved during a migration

- Namespaces
- Persistent Volumes (move or copy)
- All important resource objects (Deployments, StatefulSets, etc)

## Available in OpenShift 4.2

Customers are anxious to get their hands on this, but we want to get it right. We would love to receive sample application workloads to test.

Product Manager: Maria Bracho

Not Available Yet

| Name      | Migrations | Source                     | Target         | Repository   | Persistent Volumes | Last Status      |
|-----------|------------|----------------------------|----------------|--------------|--------------------|------------------|
| demo plan | ⌚ 2        | Summit Demo Source Cluster | Target cluster | mydemobucket | 2                  | Migrated Success |
| demo2     | ⌚ 2        | Summit Demo Source Cluster | Target cluster | mydemobucket | 2                  | Migrated Success |

# Why did we choose this migration strategy?

## Reducing risk

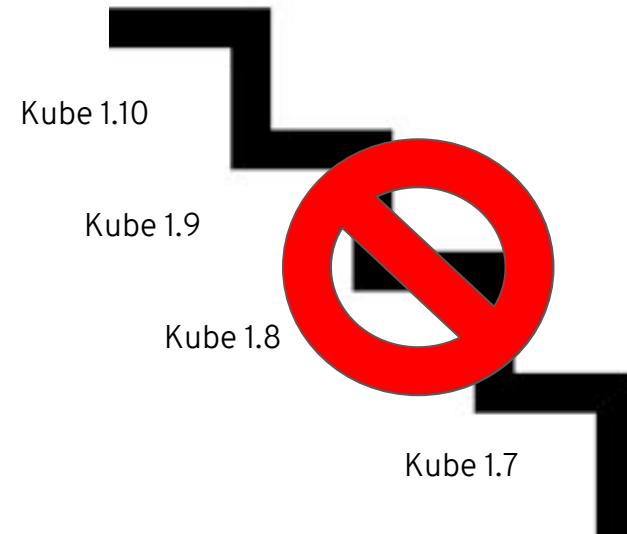
A ton of innovation went into OpenShift 4, and an in-place upgrade would have risk of failure in which there is no forwards or backwards remediation. It allows you to skip from 3.7/3.9/3.10/3.11 to 4.x. Skipping the need to install each one.

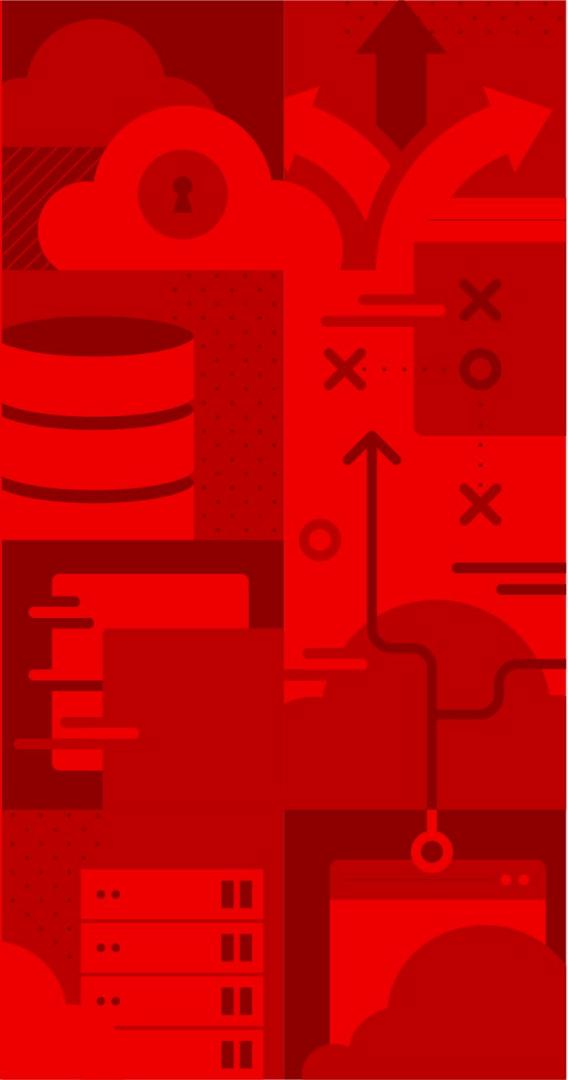
## Useful for 4-to-4 migrations

A general migration tool is frequently requested and a better long term investment. Helps you build a foundation towards making your cluster investments less fragile.

## Allows for staging

Stage a mock migration before doing it live, on a Project by Project basis. Extremely useful for success.





# Questions?

 [linkedin.com/company/red-hat](https://www.linkedin.com/company/red-hat)

 [facebook.com/redhatinc](https://www.facebook.com/redhatinc)

 [youtube.com/user/RedHatVideos](https://www.youtube.com/user/RedHatVideos)

 [twitter.com/RedHat](https://twitter.com/RedHat)