ASK AN EXPERT

Red Hat | PRODEVANS

MAKING OF YOUR APPLICATION CONTAINER READY

A Journey through expertise...

Build Your own PaaS, Start App Modernization, and Get your team ready.

2021 VIRTUAL WORKSHOP

May 2021 - June 2021

Red Hat
ADVANCED
BUSINESS PARTNER

PRODEVANS

@TechProdevans

Deepak Mishra
(CEO and Founder of Prodevans Technologies)

Sitikantha Mishra
(Solution Architect, Prodevans Red Hat One Team)

AGENDA

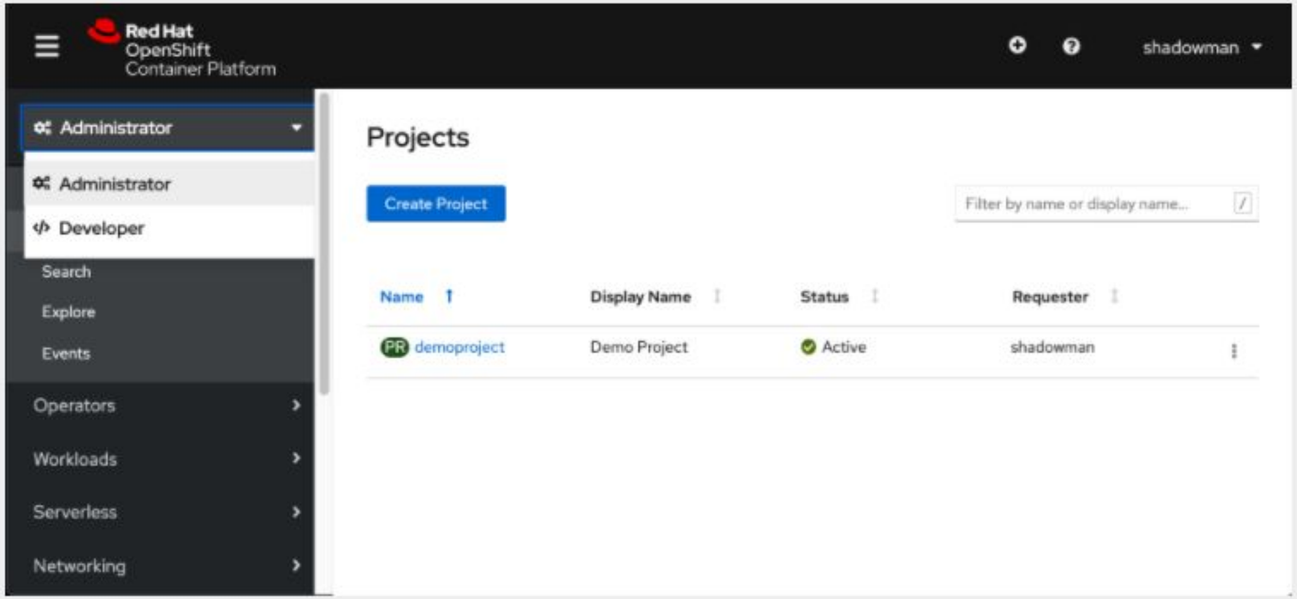
- OpenShift Managed Cluster - User Experience
- OpenShift Administration - Command Line
- Quotas and limits in OpenShift 4
- Monitoring and Metering

Module Topics

- Web Console
- Users and Projects
- Quotas and Limits
- Logging and Monitoring
- Templates, Operators, and Helm 3

Web Console - Overview

- Two *perspectives*:
  - Administrator
  - Developer
- Runs as pods
- Customizable
- Built-in metrics

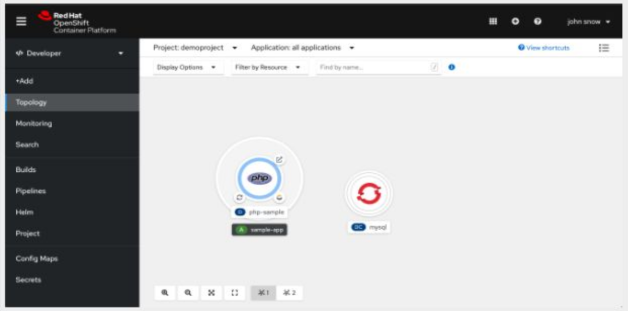


Projects

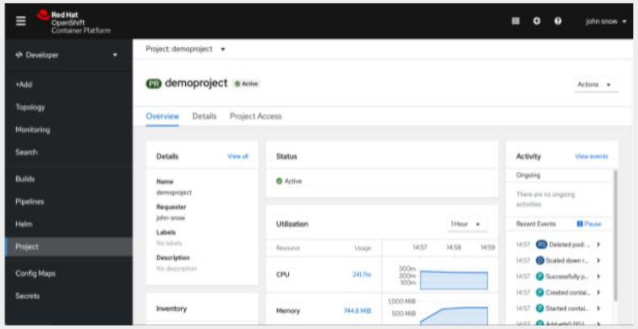
Name	Display Name	Status	Requester
PR demoproject	Demo Project	Active	shadowman

Web Console - Developer Perspective

- Topology view
  - Application-centric
  - Shows components and status, routes, source code
  - Drag arrows to create relationships
  - Add components to applications easily

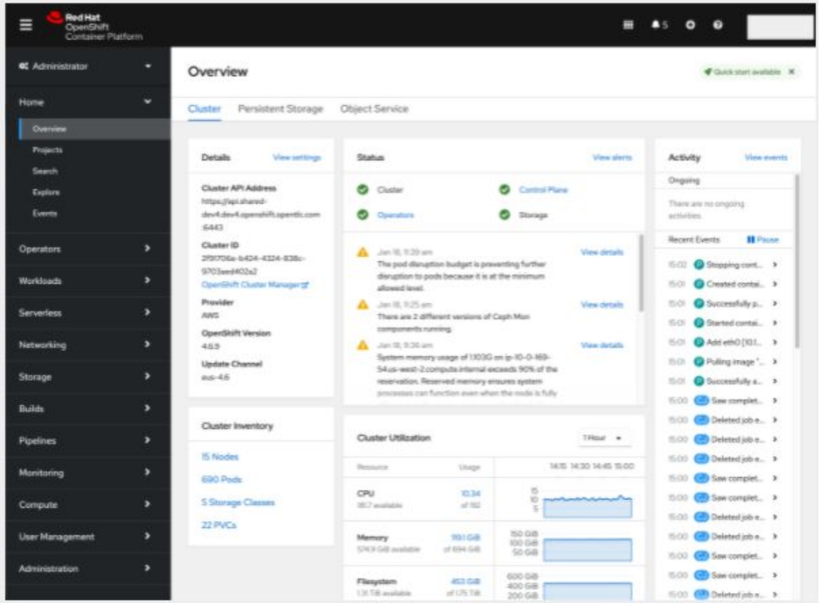


- Project
  - Status, Utilization, Events, Quotas
- Project Access
  - Control users and groups
- Metrics



Web Console - Administrator Perspective

- For Project and Cluster Administrators
- Overview (cluster admin, only)
- Status, Activity/Events, Inventory, Capacity, Utilization
- Every common resource type manageable
- Logging and metrics
- Advanced Settings to view: Updates, Operators, CRDs, role bindings, resource quotas



Cluster API Address
https://api.shared-dev4.dev4.openshift.opentlc.com:6443

Cluster ID
2f91706a-b424-4324-838c-9703aed402a2
OpenShift Cluster Manager

Provider
AWS

OpenShift Version
4.6.9

Update Channel
eus-4.6

Jan 18, 11:20 am
The pod disruption budget is preventing further disruption to pods because it is at the minimum allowed level.

Jan 18, 11:25 am
There are 2 different versions of Ceph Mon components running.

Jan 18, 11:26 am
System memory usage of 1.103G on ip-10-0-169-54.us-west-2.compute.internal exceeds 90% of the reservation. Reserved memory ensures system processes can function even when the node is fully

15 Nodes
690 Pods
5 Storage Classes
22 PVCs

Resource	Usage	14:15 14:30 14:45 15:00
CPU 181.7 available	10.34 of 192	
Memory 574.9 GiB available	119.1 GiB of 694 GiB	
Filesystem 1.31 TiB available	463 GiB of 1.75 TiB	

Ongoing
There are no ongoing activities.

15:02 Stopping cont...
15:01 Created contai...
15:01 Successfully p...
15:01 Started contai...
15:01 Add eth0 [10.1...
15:01 Pulling image "...
15:01 Successfully a...
15:00 Saw complet...
15:00 Deleted job e...
15:00 Deleted job e...
15:00 Deleted job e...
15:00 Saw complet...
15:00 Saw complet...
15:00 Deleted job e...
15:00 Deleted job e...
15:00 Saw complet...
15:00 Saw complet...
15:00 Deleted job e...

Users and Project

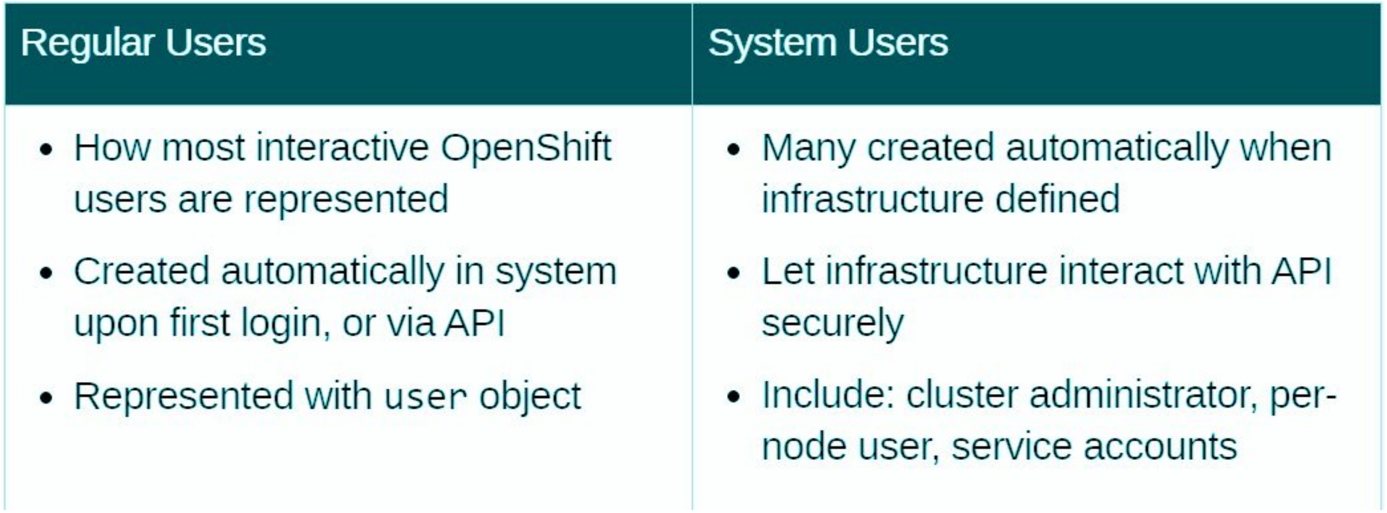
Project

- Allows groups of users or developers to work together
- Unit of isolation and collaboration
- Defines scope of resources
- Allows project administrators and collaborators to manage resources
- Restricts and tracks use of resources with quotas and limits
- Kubernetes namespace with additional annotations
  - Central vehicle for managing resource access for regular users
  - Lets community of users organize and manage content in isolation from other communities
- Users:
  - Receive access to projects from administrators
  - Have access to own projects if allowed to create them
- Each project has own:
  - **Objects**: Pods, services, replication controllers, etc.
  - **Policies**: Rules that specify which users can or cannot perform actions on objects
  - **Constraints**: Quotas for objects that can be limited
  - **Service accounts**: Users that act automatically with access to project objects

Users and Projects

Users and User Types

- Interactions with OpenShift® always associated with user
    - System permissions granted by adding roles to users or groups
- User types:



Regular Users	System Users
• How most interactive OpenShift users are represented • Created automatically in system upon first login, or via API • Represented with `user` object	• Many created automatically when infrastructure defined • Let infrastructure interact with API securely • Include: cluster administrator, per-node user, service accounts

Users and Projects

Login and Authentication

- Every user must authenticate to access OpenShift
- API requests lacking valid authentication are authenticated as anonymous user
- Policy determines what user is authorized to do

Web Console Authentication

- Access web console at URL provided by your administrator
- Provide login credentials to obtain token to make API calls
- Use web console to navigate projects

CLI Login

- Use oc tool to log in to same address as web console
- Provide login credentials to obtain token to make API calls
  - oc login -u <my-user-name> --server="<master-api-public-addr>:<master-public-port>"
- Administrators can have key generated by cluster for password-less authentication

Quotas and Limits

Resource Quotas

- OpenShift can limit:
  - Number of objects created in project
  - Amount of compute/memory/storage resources requested across objects in project
  - Based on specified label
    - Examples: To limit to department of developers or environment such as test
- Multiple teams can share single OpenShift cluster
  - Each team in own project or projects
  - Resource quotas prevent teams from depriving each other of cluster resources
- ResourceQuota object enumerates hard resource usage limits *per project*
- ClusterResourceQuota object enumerates hard resource usage limits for users across the cluster

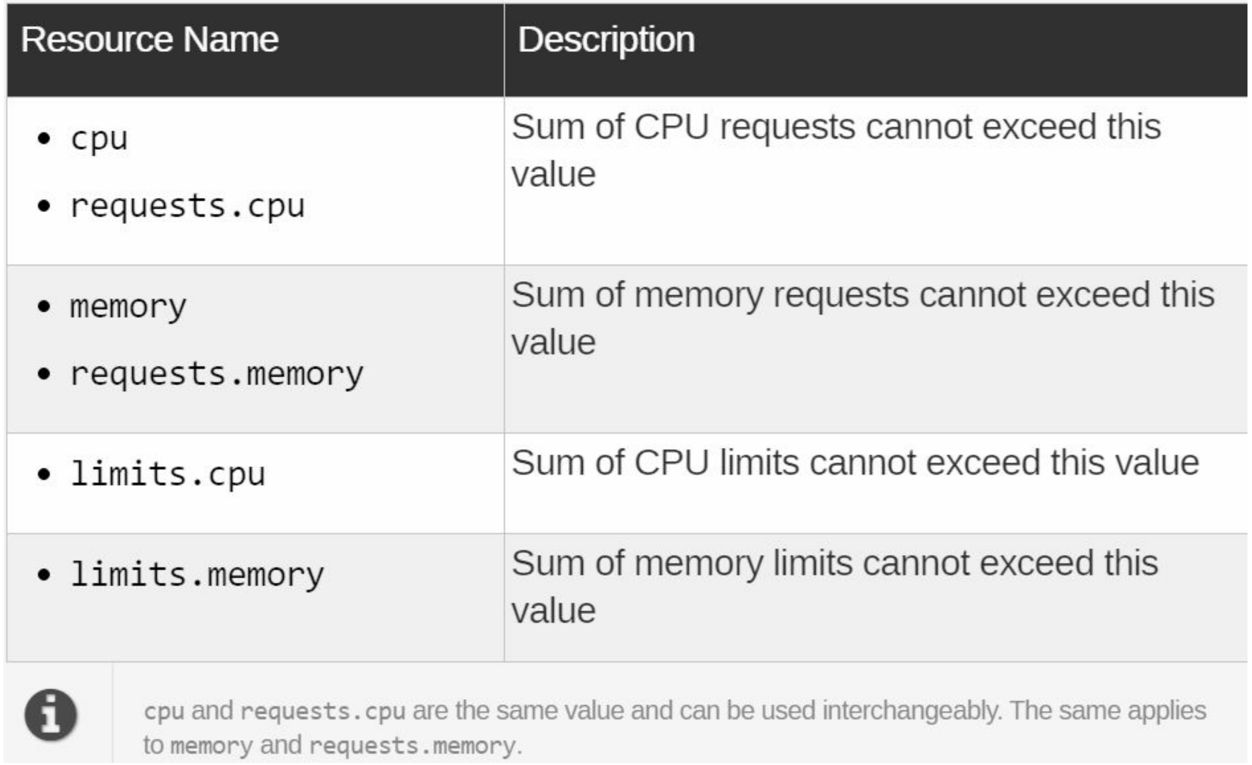
Ouotas and Limits

LimitRanges

- LimitRanges express CPU and memory requirements of pods' containers
  - Set request and limit of CPU and memory particular pods' container may consume
  - Aid OpenShift scheduler in assigning pods to nodes
- LimitRanges express quality of service tiers:
  - Best Effort
  - Burstable
  - Guaranteed
- Default LimitRange for all pods/containers can be set for each project

Ouotas and Limits

Compute Resources Managed by Quota Across Pods in Non-Terminal State

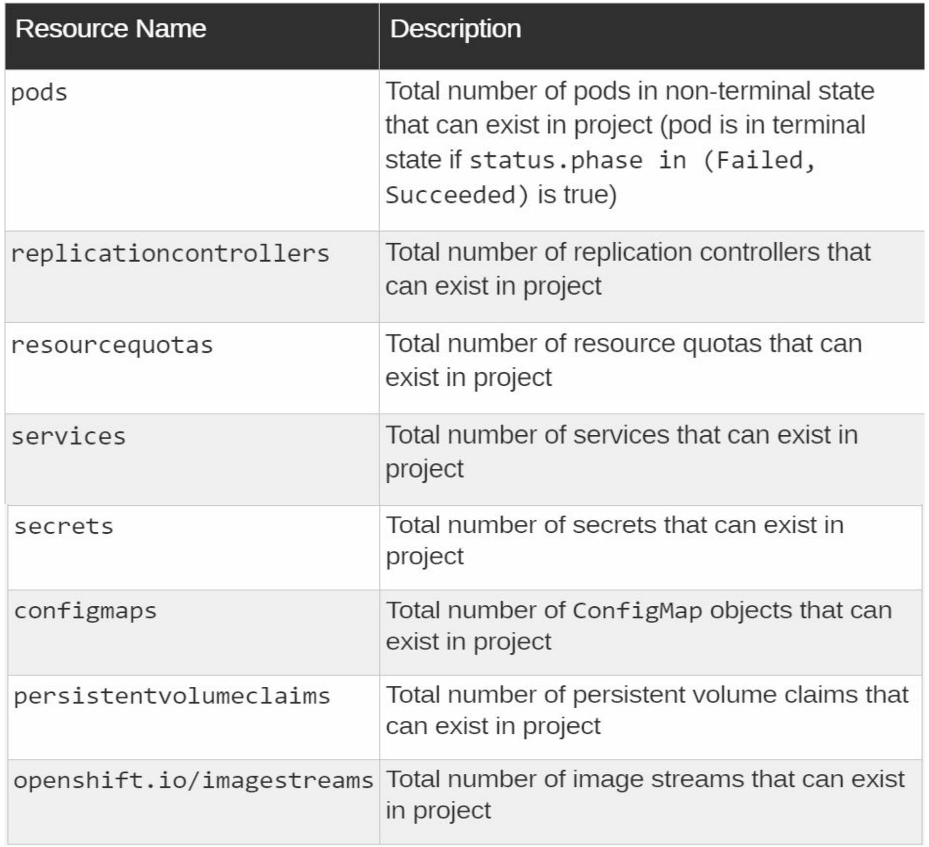


Resource Name	Description
• `cpu` • `requests.cpu`	Sum of CPU requests cannot exceed this value
• `memory` • `requests.memory`	Sum of memory requests cannot exceed this value
• `limits.cpu`	Sum of CPU limits cannot exceed this value
• `limits.memory`	Sum of memory limits cannot exceed this value

ℹ `cpu` and `requests.cpu` are the same value and can be used interchangeably. The same applies to `memory` and `requests.memory`.

Ouotas and Limits

Object Counts Managed by Quota



Resource Name	Description
`pods`	Total number of pods in non-terminal state that can exist in project (pod is in terminal state if `status.phase in (Failed, Succeeded)` is true)
`replicationcontrollers`	Total number of replication controllers that can exist in project
`resourcequotas`	Total number of resource quotas that can exist in project
`services`	Total number of services that can exist in project
`secrets`	Total number of secrets that can exist in project
`configmaps`	Total number of `ConfigMap` objects that can exist in project
`persistentvolumeclaims`	Total number of persistent volume claims that can exist in project
`openshift.io/imagestreams`	Total number of image streams that can exist in project

Ouotas and Limits

Quota Enforcement

- After quota created in project:
  - Project restricts ability to create resources that may violate quota constraint
  - Usage statistics calculated every few seconds (configurable)
- If project modification exceeds quota:
  - Server denies action
  - Returns error message
- Error message includes:
  - Quota constraint violated
  - Current system usage statistics

Ouotas and Limits

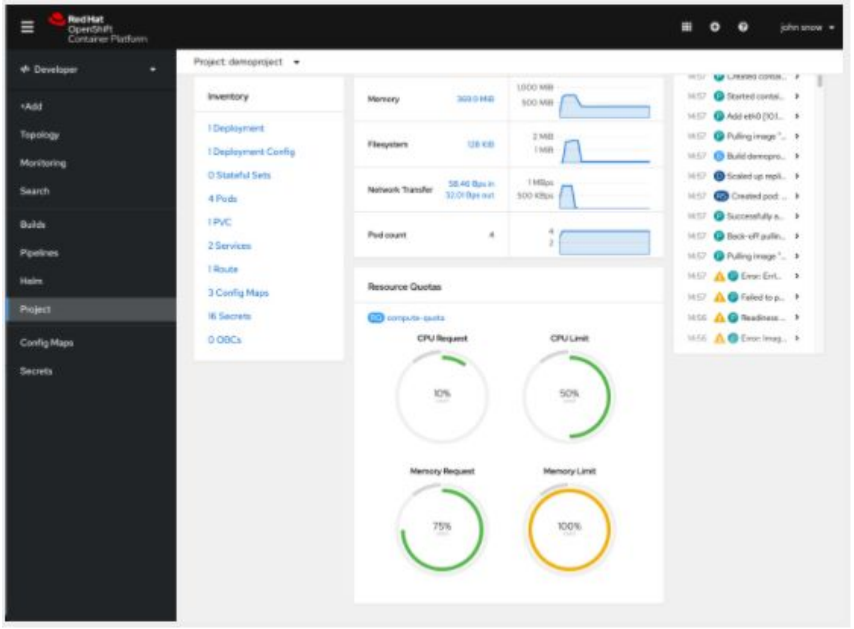
Viewing Quota:

From web console, select:

- demoproject project
- **Developer → Project**
- Scroll to see resource usage, availability
- Based on requests and limits for CPU, memory

Click RQ compute-resources:

- Shows specific resource type quotas, usage reports
- Shows combined container and pod requests, limits

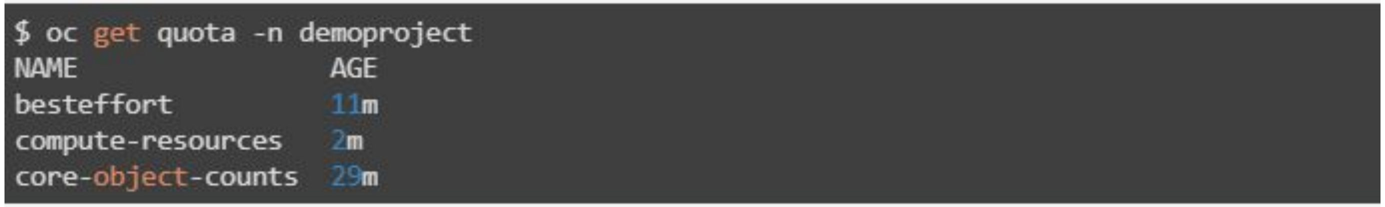


Ouotas and Limits

Viewing Quota

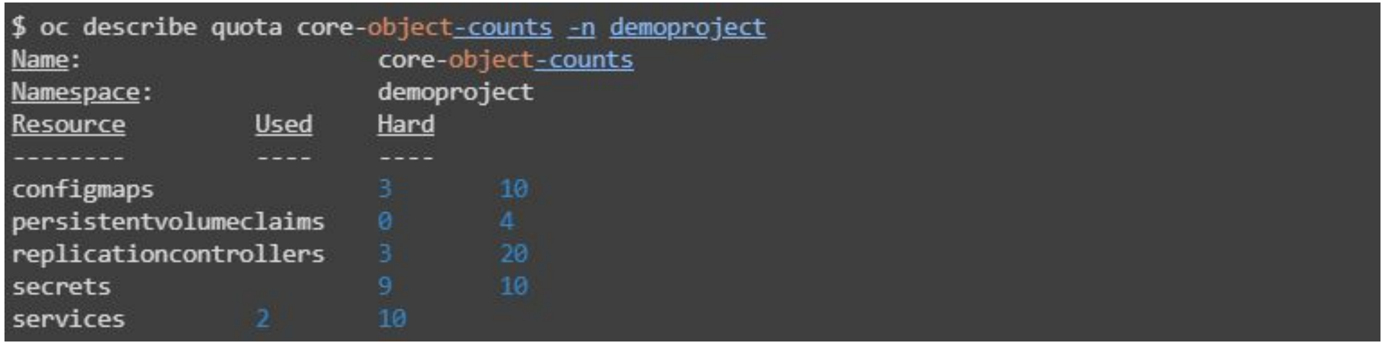
- Alternatively, use CLI to view quota details:

  Example: Get list of quotas defined in demoproject project:



```
$ oc get quota -n demoproject
NAME                AGE
besteffort          11m
compute-resources   2m
core-object-counts  29m
```

  Example: Describe core-object counts quota in demoproject project:

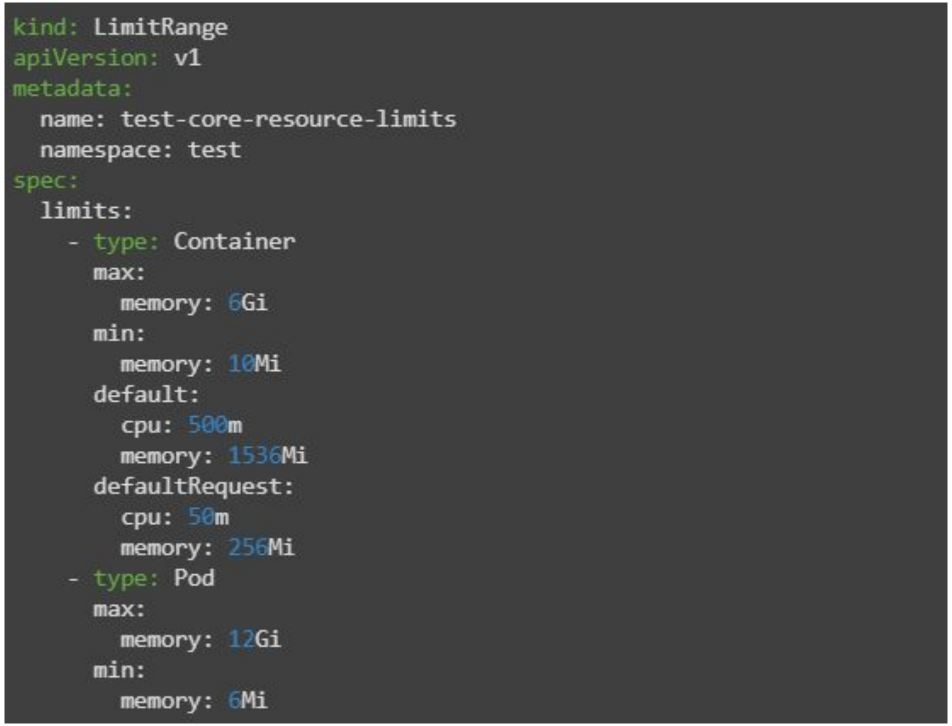


```
$ oc describe quota core-object-counts -n demoproject
Name:                    core-object-counts
Namespace:               demoproject
Resource        Used     Hard
--------        ----     ----
configmaps               3       10
persistentvolumeclaims   0       4
replicationcontrollers   3       20
secrets                  9       10
services        2        10
```

Ouotas and Limits

Viewing LimitRanges

- CPU and memory requirements of pods and containers
- Defaults for containers:



```yaml
kind: LimitRange
apiVersion: v1
metadata:
  name: test-core-resource-limits
  namespace: test
spec:
  limits:
    - type: Container
      max:
        memory: 6Gi
      min:
        memory: 10Mi
      default:
        cpu: 500m
        memory: 1536Mi
      defaultRequest:
        cpu: 50m
        memory: 256Mi
    - type: Pod
      max:
        memory: 12Gi
      min:
        memory: 6Mi
```

Logging and Monitoring

Container Log Aggregation

- Using EFK stack, cluster administrators can aggregate logs for range of OpenShift services
  - Able to provide access for application developers to view them
- Modified version of EFK stack (ELK) can be found at:
  - https://www.elastic.co/videos/introduction-to-the-elk-stack
- EFK stack useful for viewing logs aggregated from hosts and applications
  - May come from multiple containers or even deleted pods

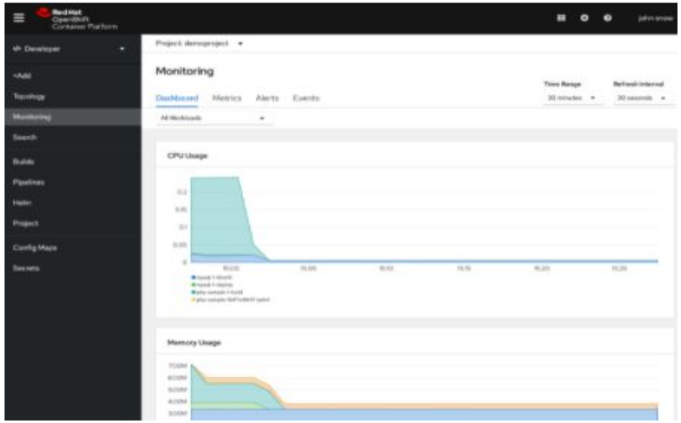
Container Log Aggregation

- Three components make up EFK logging stack:
  - **Elasticsearch**: Object store where all logs stored
  - **Fluentd**: Gathers logs from nodes, feeds them to Elasticsearch
  - **Kibana**: Web UI for Elasticsearch
- After EFK deployed, stack aggregates logs from all nodes and projects into Elasticsearch
  - Provides Kibana UI to view them
- Cluster administrators can view all logs
- Developers can view only logs for projects for which they have permission

Logging and Monitoring

Fluentd

- Pulls logs from container file system and OpenShift services on host
- Sends them to respective Elasticsearch clusters that store aggregated log data
- Users and platform admins access respective Kibana UIs to see application's or platform's aggregated logs

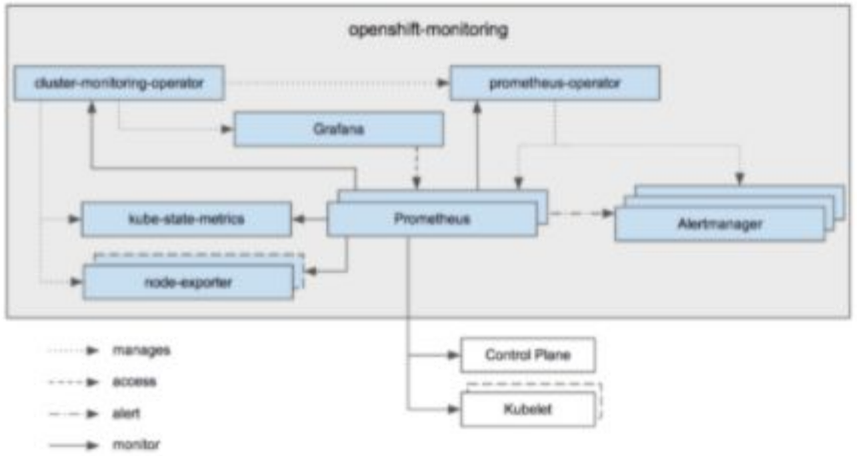


Metrics Collection and Alerting with Prometheus - Features

- Pre-configured and self-updating monitoring stack based on Prometheus open source project
- Provides monitoring of cluster components
- Ships with set of alerts and dashboards
  - May also install and use Grafana dashboards
- One UI for OpenShift administrator to view cluster's metrics from all containers, components
- Metrics used by horizontal pod autoscalers (HPAs) to determine when and how to scale
  - CPU and memory-based metrics viewable from OpenShift Container Platform web console

Logging and Monitoring

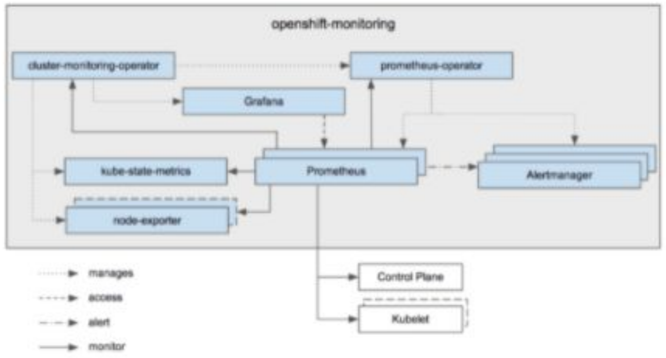
- Cluster Monitoring Operator (CMO)
    - Watches deployed monitoring components, resources; keeps up to date
- Prometheus Operator (PO)
    - Manages Prometheus and Alertmanager
    - Automatically generates monitoring targets based on Kubernetes label queries
    - Alertmanager processes client alerts and routes to email, PagerDuty, etc.
- node-exporter: Agent to collect metrics
- kube-state-metrics: Converts objects to metrics



Logging and Monitoring

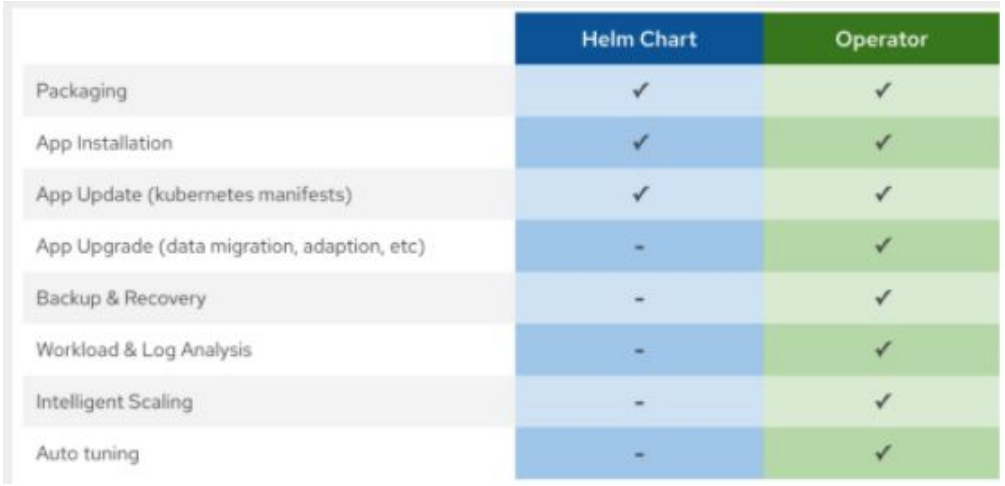
Metrics Collection and Alerting with Prometheus - HPA

- Configure horizontal pod autoscaling (HPA) based on any metric from Prometheus
  - Autoscale based on any cluster-level metrics from OpenShift
  - Autoscale based on any application metrics
- Autoscales pods and machines
  - Prometheus Adapter connects to single Prometheus instance (or via Kubernetes service)
  - Manual deployment and configuration of adapter
  - Cluster administrator needs to whitelist cluster metrics for HPA



Templates, Operators, and Helm 3

- **Templates**:
  - Can only create resources
  - Cannot manage or delete resources
  - Not associated with pod
- **Operators**:
  - Create, manage, and delete resources
  - Implemented by operator pods
- **Helm 3**:
  - Package of templates
  - How application packaged
  - How package installed



	Helm Chart	Operator
Packaging	✓	✓
App Installation	✓	✓
App Update (kubernetes manifests)	✓	✓
App Upgrade (data migration, adaption, etc)	-	✓
Backup & Recovery	-	✓
Workload & Log Analysis	-	✓
Intelligent Scaling	-	✓
Auto tuning	-	✓

Templates, Operators, and Helm 3

What Is a Template?

- Describes set of objects that can be parameterized and processed to produce list of objects for OpenShift to create
- Process templates to create anything you have permission to create within project
  - Examples: Service, build configuration, deployment configuration
- Can also define set of labels to apply to every object defined in template

What Are Templates For?

- Create instantly deployable applications for developers or customers
- Provide option to use preset variables or randomize values (like passwords)

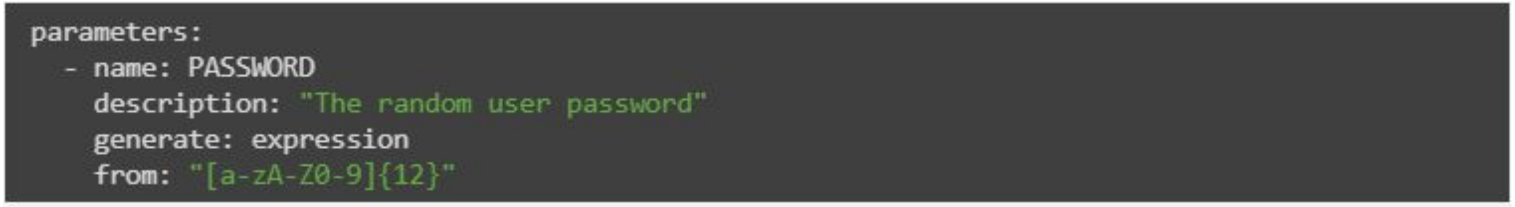
Labels in Templates

- Used to manage generated resources
- Apply to *every* resource generated from template
- Organize, group, or select objects and resources
- Resources and pods are "tagged" with labels
- Labels allow services and replication controllers to:
  - Determine pods they relate to
  - Reference groups of pods
  - Treat pods with different containers as similar entities

Templates, Operators, and Helm 3

Parameters in Templates

- Share configuration values between different objects in template
- Values can be static or generated by template
- Templates let you define parameters that take on values
  - Value substituted wherever parameter is referenced
  - Can define references in any text field in object definition
- Example:
  - Set generate to expression to specify generated values
  - from specifies pattern for generating value using pseudo-regular expression syntax



```
parameters:
  - name: PASSWORD
    description: "The random user password"
    generate: expression
    from: "[a-zA-Z0-9]{12}"
```

Templates, Operators, and Helm 3

Creating Templates from Existing Objects

- Can export existing objects from project in template form
- Modify exported template by adding parameters and customizations
- To export objects from project in template form:

```
$ oc export all --as-template=<template_name>
```

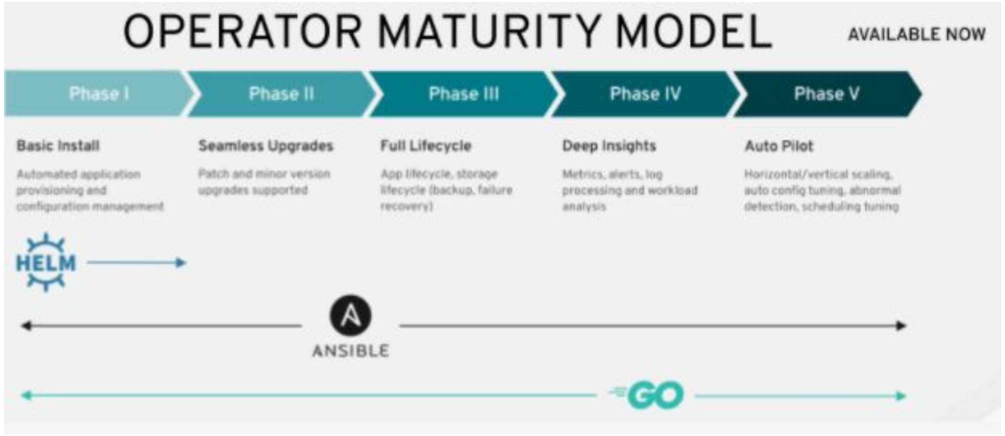
Operator Hub

- Deploy and manage application in cluster with Operator
- Discover Operators from Kubernetes community and Red Hat® partners
- Install Operators on clusters to provide optional add-ons and shared services to developers
- Capabilities provided by Operator appear in Developer Catalog, providing self-service experience
- Add shared applications, services, or source-to-image builders to your project
- Cluster administrators can install additional applications that show up automatically

Templates, Operators, and Helm 3

Helm Charts and Operators

- Helm charts similar to templates
    - Collection of files that describe related set of Kubernetes resources
- Choose from many available Helm charts
- Use charts with Helm operator to deploy application easily

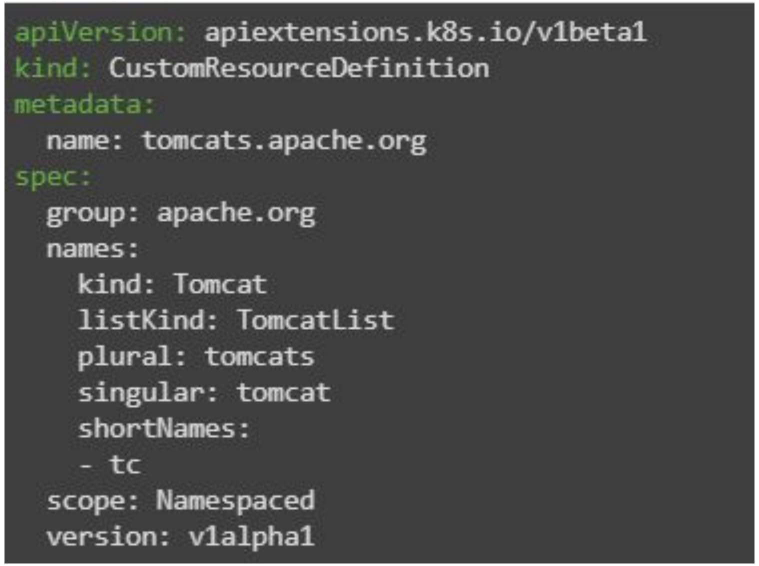

OPERATOR MATURITY MODEL
AVAILABLE NOW
Phase I
Phase II
Phase III
Phase IV
Phase V
Basic Install
Automated application provisioning and configuration management
Seamless Upgrades
Patch and minor version upgrades supported
Full Lifecycle
App lifecycle, storage lifecycle (backup, failure recovery)
Deep Insights
Metrics, alerts, log processing and workload analysis
Auto Pilot
Horizontal/vertical scaling, auto config tuning, abnormal detection, scheduling tuning
HELM
ANSIBLE
GO

Templates, Operators, and Helm 3

Operator Interaction with OpenShift

- Operators are pods that take advantage of custom resource definitions (CRDs)
- CRDs allow extension of Kubernetes/OpenShift API
    - API then knows new resources
- CRDs allow creation of custom resources (CRs)
- Operator watches for creation of CR, reacts by creating application
- CRs managed in same way as stock OpenShift objects
    - create, get, describe, delete, etc.
    - Example: oc get tomcats

Custom Resource Definition :



```
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: tomcats.apache.org
spec:
  group: apache.org
  names:
    kind: Tomcat
    listKind: TomcatList
    plural: tomcats
    singular: tomcat
    shortNames:
    - tc
  scope: Namespaced
  version: v1alpha1
```

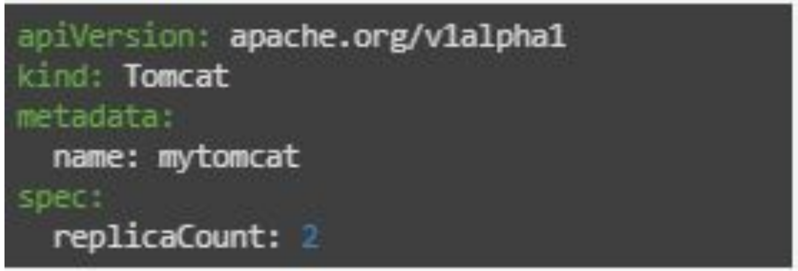
Templates, Operators, and Helm 3

Operators - Custom Resource (Application) Creation

- Custom Resources are simple definitions of resource you want Operator to create
- Running Operator watches for CR to be created in either:
    - Entire OpenShift cluster
    - Project that Operator is running in
- When CR created, operator receives event
- Operator then creates all OpenShift resources that make up application

Example: Custom Resource Creation

- To create CR instance:
    - Create YAML file with definition of CR:



```
apiVersion: apache.org/v1alpha1
kind: Tomcat
metadata:
  name: mytomcat
spec:
  replicaCount: 2
```

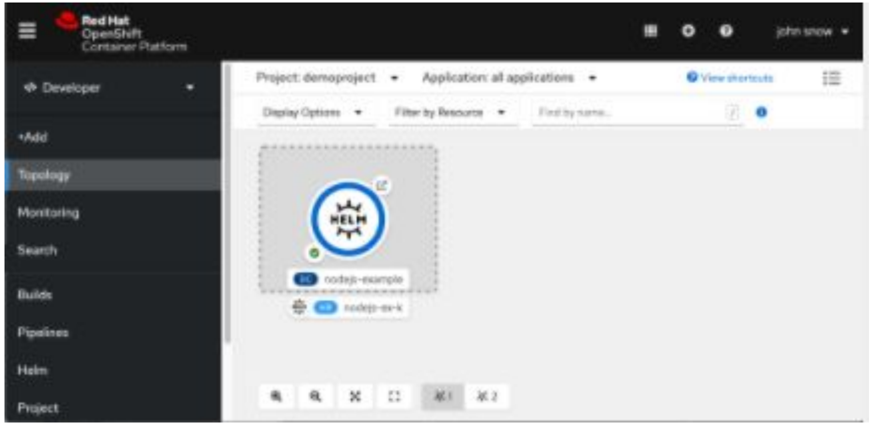
- Create resource in OpenShift:

```
oc create -f mytomcat.yaml
```

Templates, Operators, and Helm 3

Helm Charts

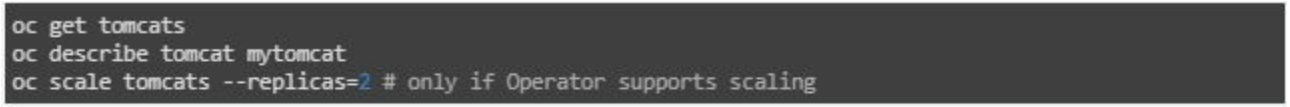
- When Helm chart installed:
  - Values in values file replaced in chart templates
  - Release produced as instance of chart running in Kubernetes cluster
- Install same chart multiple times to create many releases
- Helm v3
  - No server-side component (Tiller)
  - Helm CLI interacts with Kubernetes APIs



Templates, Operators, and Helm 3

Operators - Custom Resource Management

- To manipulate and examine CR, use oc commands:



```
oc get tomcats
oc describe tomcat mytomcat
oc scale tomcats --replicas=2 # only if Operator supports scaling
```

Do not scale ReplicaSets, Deployment, StatefulSets directly

- Use Operator to scale
- Operator continues to watch created resources and sets them back to initial states

To delete all created OpenShift API objects, delete CR:

```
oc delete tomcat mytomcat
```

Any Queries?