

Red Hat Training and Certification

OpenShift Administration I - Managing
Containers and Kubernetes

Travis Michette

Table of Contents

1. Introduction to Kubernetes and OpenShift	1
1.1. Red Hat OpenShift Components and Editions	1
1.1.1. Red Hat OpenShift and Kubernetes Overview	1
1.1.1.1. Introducing Kubernetes and Red Hat OpenShift	1
1.1.1.2. Red Hat OpenShift Platform Architecture	1
1.1.1.3. Red Hat OpenShift Editions	1
1.1.1.4. Operators Overview	1
1.1.1.5. Supporting Red Hat OpenShift Container Platform	1
1.2. Navigate The OpenShift Web Console	2
1.2.1. Overview of the Red Hat OpenShift Web Console	2
1.2.1.1. Accessing the OpenShift Web Console	2
1.2.1.2. Web Console Perspectives	2
1.2.1.3. OpenShift Web Console Layout	2
1.2.1.4. Red Hat OpenShift Key Concepts	2
1.3. DEMO: Navigate the OpenShift Web Console	3
1.4. Architecture of Kubernetes and OpenShift	7
1.4.1. Overview of Red Hat OpenShift Container Platform (RHOCP) and Kubernetes Architectures	7
1.4.2. The History of Container Management	7
1.4.3. Kubernetes Features	7
1.4.3.1. Service discovery and load balancing	7
1.4.3.2. Horizontal scaling	7
1.4.3.3. Self-healing	7
1.4.3.4. Automated rollout	7
1.4.3.5. Secrets and configuration management	7
1.4.3.6. Operators	7
1.4.4. Red Hat OpenShift Container Platform Features	7
1.4.4.1. Integrated developer workflow	7
1.4.4.2. Routes	7
1.4.4.3. Metrics and logging	7
1.4.4.4. Unified User Interface	7
1.4.5. Kubernetes Architectural Concepts	7
1.4.5.1. Control Plane Components	7
1.4.5.2. Compute Plane Components	7
1.4.5.3. Additional Concepts	7
1.4.5.4. Red Hat OpenShift Container Platform Architectural Concepts	7

1.4.5.5. Overview of RHOCP Cluster Setups	8
1.5. Monitor an OpenShift Cluster	9
1.5.1. Overview of Nodes, Machines, and Machine Configs	9
1.5.1.1. Accessing Node Logs	9
1.5.1.2. Accessing Pod Logs	9
1.5.1.3. Red Hat OpenShift Container Platform Metrics and Alerts	9
1.5.1.4. Kubernetes Events	9
1.5.1.5. Red Hat OpenShift Container Platform API Explorer	9
1.6. DEMO: Monitoring and OCP Cluster	10
2. Kubernetes and OpenShift Command-Line Interfaces and APIs	14
2.1. The Kubernetes and OpenShift Command-Line Interfaces	14
2.1.1. The Kubernetes and OpenShift Command-Line Interfaces	14
2.1.1.1. Kubernetes Command-Line Tool	14
2.1.1.2. The OpenShift Command-Line Tool	14
2.1.1.3. Managing Resources at the Command Line	14
2.1.1.4. Authentication with OAuth	14
2.2. DEMO: Kubernetes and OpenShift Command-Line Interfaces	15
2.3. Inspect Kubernetes Resources	21
2.3.1. Kubernetes and OpenShift Resources	21
2.3.1.1. Structure of Resources	21
2.3.2. Command Outputs	21
2.3.2.1. YAML Output	21
2.3.2.2. JSON Output	21
2.3.2.3. Custom Output	21
2.4. DEMO: Inspect Kubernetes Resources	22
2.5. Assess the Health of an OpenShift Cluster	25
2.5.1. Query Operator Conditions	25
2.5.2. Examining Cluster Metrics	25
2.5.2.1. Viewing Cluster Metrics	25
2.5.2.2. Viewing Project Metrics	25
2.5.2.3. Viewing Resource Metrics	25
2.5.2.4. Performing Prometheus Queries in the Web Console	25
2.5.3. Query Cluster Events and Alerts	25
2.5.3.1. Kubernetes Alerts	25
2.5.4. Check Node Status	25
2.5.4.1. Check Pod Status	25
2.5.5. Collect Information for Support Requests	25
2.6. DEMO: Assess the Health of an OpenShift Cluster	26

3. Run Applications as Containers and Pods	31
3.1. Introduction to Linux Containers and Kubernetes Pods.....	31
3.1.1. Containers Overview	31
3.1.2. Kubernetes and Containers	31
3.1.3. Creating Containers and Pods	31
3.1.4. User and Group IDs Assignment	31
3.1.4.1. Pod Security	31
3.1.5. Execute Commands in Running Containers	31
3.1.6. Container Logs.....	31
3.1.7. Deleting Resources	31
3.1.8. The CRI-O Container Engine.....	31
3.2. DEMO: Introduction to Linux Containers and Kubernetes Pods.....	32
3.3. Find and Inspect Container Images	37
3.3.1. Container Image Overview.....	37
3.3.2. Container Image Registries.....	37
3.3.2.1. Red Hat Registry	37
3.3.2.2. Quay.io	37
3.3.2.3. Private Registries.....	37
3.3.3. Container Image Identifiers	37
3.3.4. Container Image Components	37
3.3.5. Container Image Instructions and Metadata	37
3.3.6. Base Images	37
3.3.7. Inspecting and Managing Container Images	37
3.3.7.1. Skopeo	37
3.3.7.2. Registry Credentials	37
3.3.7.3. Theoc imageCommand.....	37
3.3.8. Running Containers as Root	37
3.4. DEMO: Find and Inspect Container Images	38
3.5. Troubleshoot Containers and Pods.....	42
3.5.1. Container Troubleshooting Overview	42
3.5.2. CLI Troubleshooting Tools	42
3.5.3. Editing Resources	42
3.5.4. Copy Files to and from Containers	42
3.5.5. Remote Container Access	42
3.5.6. Connect to Running Containers.....	42
3.5.7. Execute Commands in a Container	42
3.5.8. Container Events and Logs	42
3.5.9. Available Linux Commands in Containers	42

3.5.10. Troubleshooting from Inside the Cluster	42
3.6. DEMO: Troubleshoot Containers and Pods.....	43
4. Deploy Managed and Networked Applications on Kubernetes.....	48
4.1. Deploy Applications from Images and Templates	48
4.1.1. Deploying Applications	48
4.1.2. Resources and Resource Definitions	48
4.1.3. Managing Resources	48
4.1.4. Common Resource Types and Their Uses	48
4.1.4.1. Templates	48
4.1.4.2. Pod	48
4.1.4.3. Deployment Configurations	48
4.1.4.4. Deployment	48
4.1.4.5. Projects	48
4.1.4.6. Services	48
4.1.4.7. Persistent Volume Claims	48
4.1.4.8. Secrets	48
4.1.5. Managing Resources from the Command Line	48
4.1.5.1. Imperative Resource Management	48
4.1.5.2. Declarative Resource Management	48
4.1.6. Retrieving Resource Information	48
4.2. DEMO: Deploy Applications from Images and Templates	49
4.3. Manage Long-lived and Short-lived Applications Using the Kubernetes Workload API.....	52
4.3.1. Kubernetes Workload Resources.....	52
4.3.1.1. Jobs	52
4.3.1.2. Cron Jobs	52
4.3.1.3. Deployments	52
4.3.1.4. Stateful Sets	52
4.4. DEMO: Manage Long-lived and Short-lived Applications Using the Kubernetes Workload API ..	53
4.5. Kubernetes Pod and Service Networks	55
4.5.1. The Software-defined Network	55
4.5.2. Kubernetes Networking	55
4.5.2.1. Using Services	55
4.5.3. Kubernetes DNS for Service Discovery	55
4.5.4. Kubernetes Networking Drivers	55
4.5.4.1. The OpenShift Cluster Network Operator	55
4.6. DEMO: Kubernetes Pod and Service Networks	56
4.7. Scale and Expose Applications to External Access	60
4.7.1. IP Addresses for Pods and Services	60

4.7.2. Service Types	60
4.7.3. Using Routes for External Connectivity	60
4.7.4. Using Ingress Objects for External Connectivity	60
4.7.5. Sticky sessions	60
4.7.6. Load Balance and Scale Applications	60
4.7.6.1. Load Balance Pods	60
4.8. DEMO: Scale and Expose Applications to External Access	61
5. Manage Storage for Application Configuration and Data	64
5.1. Externalize the Configuration of Applications	64
5.1.1. Configuring Kubernetes Applications	64
5.1.2. Creating Secrets and Configuration Maps	64
5.1.3. Using Configuration Maps and Secrets to Initialize Environment Variables	64
5.1.4. Using Secrets and Configuration Maps as Volumes	64
5.1.5. Updating Secrets and Configuration Maps	64
5.1.6. Deleting Secrets and Configuration Maps	64
5.2. DEMO: Externalize the Configuration of Applications	65
5.3. Provision Persistent Data Volumes	68
5.3.1. Kubernetes Persistent Storage	68
5.3.2. Persistent Volumes	68
5.3.2.1. Volume Access Mode	68
5.3.2.2. Volume Modes	68
5.3.2.3. Manually Creating a PV	68
5.3.3. Persistent Volume Claims	68
5.3.3.1. Creating a PVC	68
5.3.3.2. Kubernetes Dynamic Provisioning	68
5.3.3.3. PV and PVC lifecycles	68
5.3.3.4. Deleting a Persistent Volume Claim	68
5.4. DEMO: Provision Persistent Data Volumes	69
5.5. Select a Storage Class for an Application	74
5.5.1. Storage Class Selection	74
5.5.1.1. Reclaim Policy	74
5.5.1.2. Kubernetes and Application Responsibilities	74
5.5.2. Use Cases for Storage Classes	74
5.5.3. Create a Storage Class	74
5.5.3.1. Cluster Storage Classes	74
5.5.4. Storage Class Usage	74
5.6. DEMO: Select a Storage Class for an Application	75
5.7. Manage non-Shared Storage with StatefulSets	79

5.7.1. Application Clustering	79
5.7.2. Storage Services	79
5.7.3. Introduction to Stateful Sets	79
5.7.4. Working with Stateful Sets	79
5.8. DEMO: Manage non-Shared Storage with Stateful Sets	80
6. Configure Applications for Reliability	83
6.1. Application High Availability with Kubernetes	83
6.1.1. Concepts of Deploying Highly Available Applications	83
6.1.2. Writing Reliable Applications	83
6.1.3. Kubernetes Application Reliability	83
6.2. DEMO: Externalize the Configuration of Applications	84
6.3. Application Health Probes	86
6.3.1. Kubernetes Probes	86
6.3.2. Authoring Probe Endpoints	86
6.3.3. Probe Types	86
6.3.3.1. Readiness Probes	86
6.3.3.2. Liveness Probes	86
6.3.3.3. Startup Probes	86
6.3.4. Types of Tests	86
6.3.5. Timings and Thresholds	86
6.3.6. Adding Probes via YAML	86
6.3.7. Adding Probes via the CLI	86
6.4. Reserve Compute Capacity for Applications	87
6.4.1. Kubernetes Pod Scheduling	87
6.4.2. Compute Resource Requests	87
6.4.3. Inspecting Cluster Compute Resources	87
6.5. DEMO: Reserve Compute Capacity for Applications	88
6.6. Limit Compute Capacity for Applications	90
6.6.1. Setting Memory Limits	90
6.6.2. Setting CPU Limits	90
6.6.3. Viewing Requests, Limits, and Actual Usage	90
6.7. Application Autoscaling	91
7. Manage Application Updates	92
7.1. Container Image Identity and Tags	92
7.1.1. Kubernetes Image Tags	92
7.1.1.1. Floating Tag Issues	92
7.1.1.2. Using SHA Image ID	92
7.1.2. Selecting a Pull Policy	92

7.1.3. Pruning Images from Cluster Nodes.....	92
7.2. Update Application Image and Settings	93
7.2.1. Application Code, Configuration, and Data.....	93
7.2.2. Deployment Strategies	93
7.2.2.1. Rolling Update Strategy	93
7.2.2.2. Recreate Strategy.....	93
7.2.3. Rolling Out Applications	93
7.2.3.1. Monitoring Replica Sets	93
7.2.3.2. Managing Rollout	93
7.3. Reproducible Deployments with OpenShift Image Streams	94
7.3.1. Image Streams	94
7.3.1.1. Image Stream Tags	94
7.3.1.2. Image Names, Tags, and IDs	94
7.3.2. Creating Image Streams and Tags.....	94
7.3.2.1. Importing Image Stream Tags Periodically.....	94
7.3.2.2. Configuring Image Pull-through	94
7.3.3. Using Image Streams in Deployments	94
7.3.3.1. Enabling the Local Lookup Policy.....	94
7.3.3.2. Configuring Image Streams in Deployments	94
7.4. Automatic Image Updates with OpenShift Image Change Triggers	95
7.4.1. Using Triggers to Manage Images	95
7.4.2. Configuring Image Trigger for Deployments	95
7.4.2.1. Rolling Out Deployments	95
7.4.2.2. Rolling Back Deployments.....	95
7.4.3. Managing Image Stream Tags	95

Chapter 1. Introduction to Kubernetes and OpenShift

1.1. Red Hat OpenShift Components and Editions

Section Info Here

1.1.1. Red Hat OpenShift and Kubernetes Overview

1.1.1.1. Introducing Kubernetes and Red Hat OpenShift

1.1.1.2. Red Hat OpenShift Platform Architecture

1.1.1.3. Red Hat OpenShift Editions

1.1.1.4. Operators Overview

1.1.1.5. Supporting Red Hat OpenShift Container Platform

1.2. Navigate The OpenShift Web Console

Section Info Here

1.2.1. Overview of the Red Hat OpenShift Web Console

1.2.1.1. Accessing the OpenShift Web Console

1.2.1.2. Web Console Perspectives

1.2.1.3. OpenShift Web Console Layout

1.2.1.4. Red Hat OpenShift Key Concepts

1.3. DEMO: Navigate the OpenShift Web Console

Credentials

UN/PW: developer/developer

or

UN/PW: admin/redhatocp

- **Web Console:** <https://console-openshift-console.apps.ocp4.example.com>
- **OpenShift API:** <https://api.ocp4.example.com:6443>

Listing 1. Developer Login



```
oc login -u developer -p developer https://api.ocp4.example.com:6443
```

Listing 2. Administrator Login

```
oc login -u admin -p redhatocp https://api.ocp4.example.com:6443
```

Example 1. DEMO: Navigating the OpenShift Web Console

1. Login to OCP WebUI

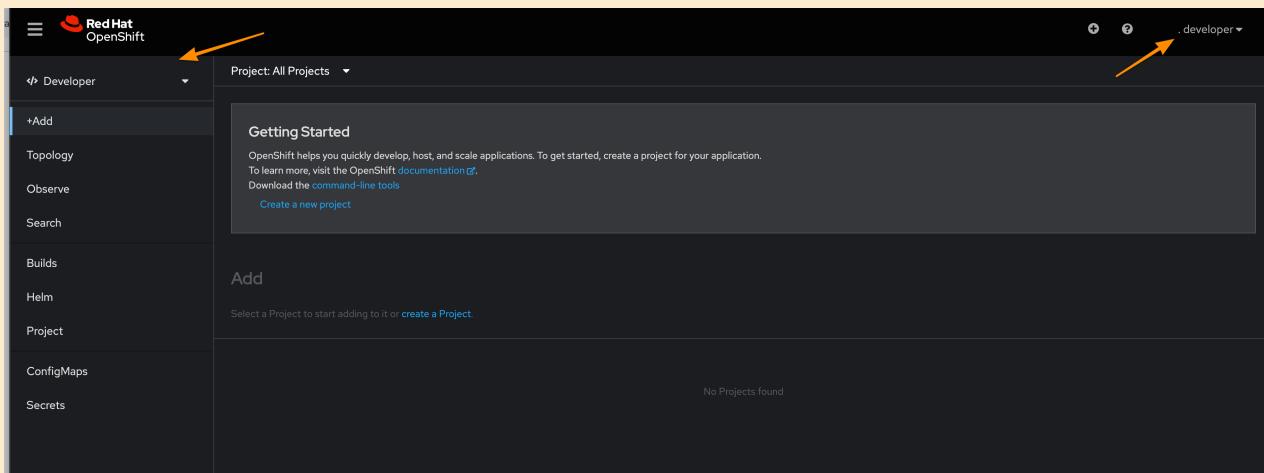


Figure 1. OpenShift Developer Console

2. Create a project (Add ⇒ Create a Project)

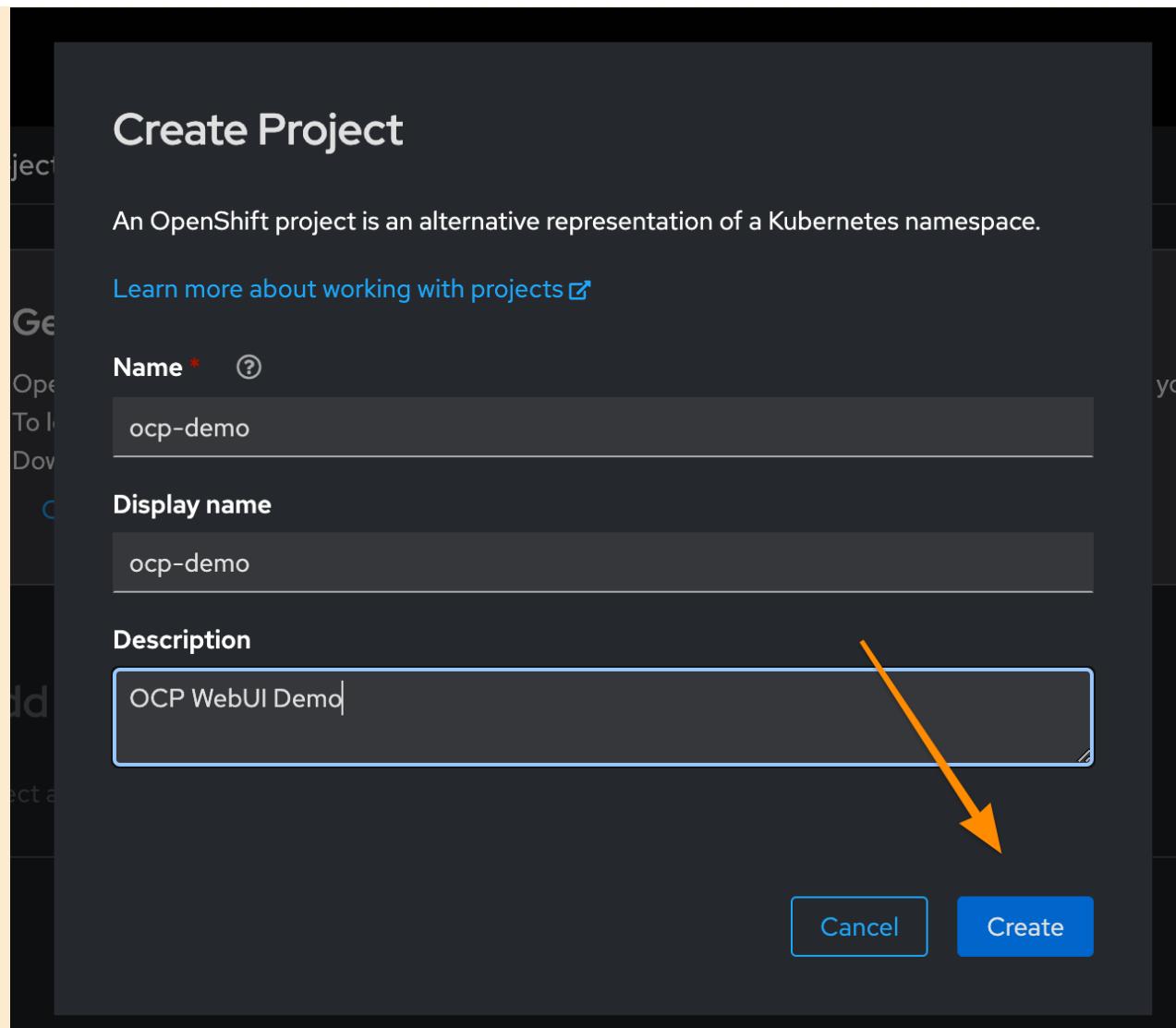


Figure 2. OpenShift New Project

3. Click the "Add" then search for Apache. Select the **HTTPD Samples**" and click *Create

Add

Get started

Choose an application

Basic

View all

Developer catalog

All samples

Browse

and connect to services

Httpd

Create

Apache HTTP Server (httpd)

Apache HTTP Server (httpd)

Httpd

IBM Operator Catalog Enablement

IBM Order Management Software E...

View all developer catalog items (12)

View all samples (1)

Project: ocp-demo

Create Sample application

Name *

A unique name given to the component that will be used to name associated resources.

Builder Image version *

 Apache HTTP Server 2.4 (RHEL 7)
BUILDER HTTPD

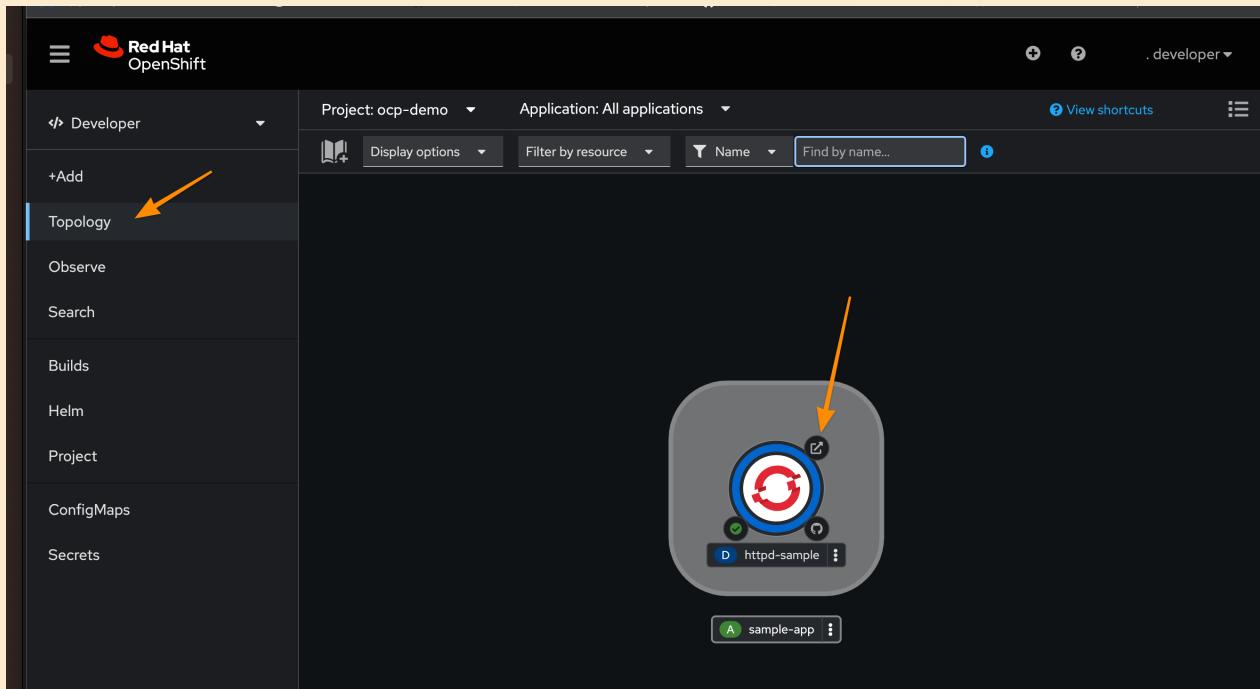
Build and serve static content via Apache HTTP Server (httpd) 2.4 on RHEL 7. For more information about using this builder image, including OpenShift considerations, see <https://github.com/sclorg/httpd-container/blob/master/2.4/README.md>.

Sample repository: <https://github.com/sclorg/httpd-ex.git>

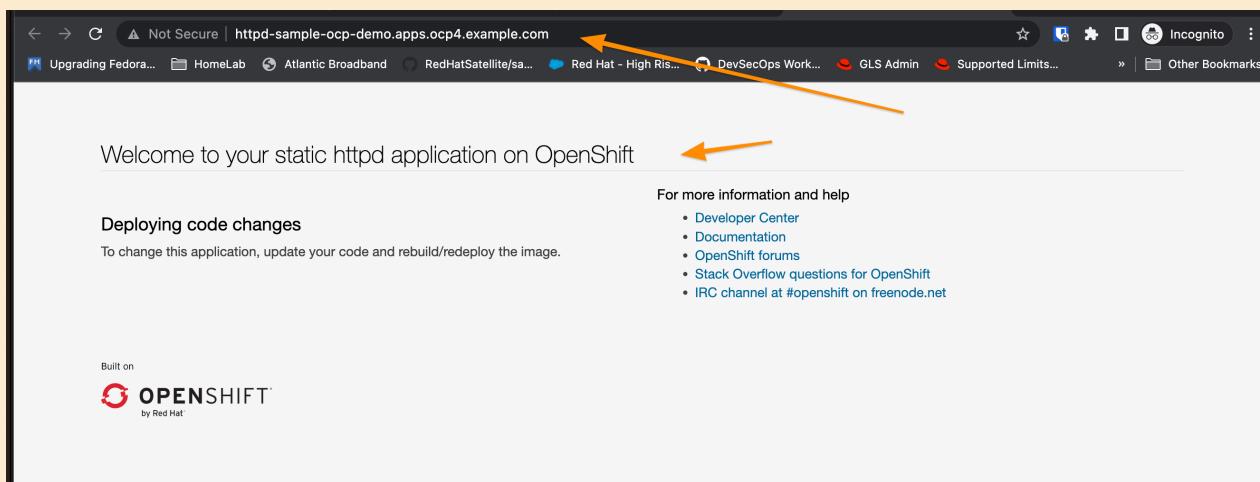
Git repo URL

Create Cancel

4. Examine the new application and look at the website (from the Topology Section)



The screenshot shows the Red Hat OpenShift web interface. On the left, there's a sidebar with various options: Developer (+Add, Topology, Observe, Search), Builds, Helm, Project, ConfigMaps, and Secrets. A yellow arrow points to the 'Topology' option. The main area is titled 'Project: ocp-demo' and 'Application: All applications'. It features a search bar and filter options. In the center, there's a large circular icon representing a pod, with a smaller 'httpd-sample' pod icon below it. Another yellow arrow points to this 'httpd-sample' icon.



The screenshot shows a web browser window. The address bar displays 'Not Secure | httpd-sample-ocp-demo.apps.ocp4.example.com'. Below the address bar, the page content starts with 'Welcome to your static httpd application on OpenShift'. It includes sections for 'Deploying code changes' (instructions to update code and rebuild/redeploy) and 'For more information and help' (links to developer center, documentation, forums, Stack Overflow, and IRC). At the bottom, it says 'Built on OPENSHIFT by Red Hat'.

5. Examine other piece of OCP for the project (from Administrator Perspective)

- Networking ⇒ Services
- Networking ⇒ Routes

1.4. Architecture of Kubernetes and OpenShift

Section Info Here

1.4.1. Overview of Red Hat OpenShift Container Platform (RHOC) and Kubernetes Architectures

1.4.2. The History of Container Management

1.4.3. Kubernetes Features

1.4.3.1. Service discovery and load balancing

1.4.3.2. Horizontal scaling

1.4.3.3. Self-healing

1.4.3.4. Automated rollout

1.4.3.5. Secrets and configuration management

1.4.3.6. Operators

1.4.4. Red Hat OpenShift Container Platform Features

1.4.4.1. Integrated developer workflow

1.4.4.2. Routes

1.4.4.3. Metrics and logging

1.4.4.4. Unified User Interface

1.4.5. Kubernetes Architectural Concepts

1.4.5.1. Control Plane Components

1.4.5.2. Compute Plane Components

1.4.5.3. Additional Concepts

1.4.5.4. Red Hat OpenShift Container Platform Architectural Concepts

1.4.5.5. Overview of RHOCOP Cluster Setups

1.5. Monitor an OpenShift Cluster

Section Info Here

1.5.1. Overview of Nodes, Machines, and Machine Configs

1.5.1.1. Accessing Node Logs

1.5.1.2. Accessing Pod Logs

1.5.1.3. Red Hat OpenShift Container Platform Metrics and Alerts

1.5.1.4. Kubernetes Events

1.5.1.5. Red Hat OpenShift Container Platform API Explorer

1.6. DEMO: Monitoring and OCP Cluster

Credentials

UN/PW: **developer/developer**

or

UN/PW: **admin/redhatocp**

- **Web Console:** <https://console-openshift-console.apps.ocp4.example.com>
- **OpenShift API:** <https://api.ocp4.example.com:6443>

Listing 3. Developer Login



```
oc login -u developer -p developer https://api.ocp4.example.com:6443
```

Listing 4. Administrator Login

```
oc login -u admin -p redhatocp https://api.ocp4.example.com:6443
```

Example 2. DEMO: OCP Cluster Monitoring

1. Login to the OCP Cluster

```
[student@workstation ~]$ oc login -u developer -p developer  
https://api.ocp4.example.com:6443  
Login successful.
```

2. Obtain Console URL

```
[student@workstation ~]$ oc whoami --show-console  
https://console-openshift-console.apps.ocp4.example.com
```

3. Login to the WebUI as an administrator and use the **Administrator** perspective to investigate monitoring capabilities.

- Open <https://console-openshift-console.apps.ocp4.example.com> in Firefox
- UN/PW: **admin/redhatocp**

4. Navigate to **Home** ⇒ **Overview** to see an overview of OCP Resources

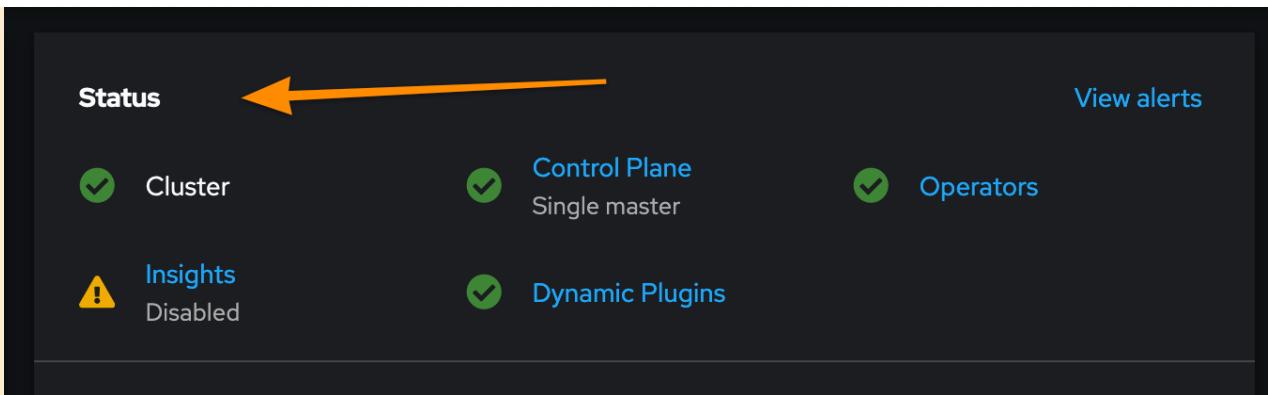


Figure 3. OCP Cluster Status

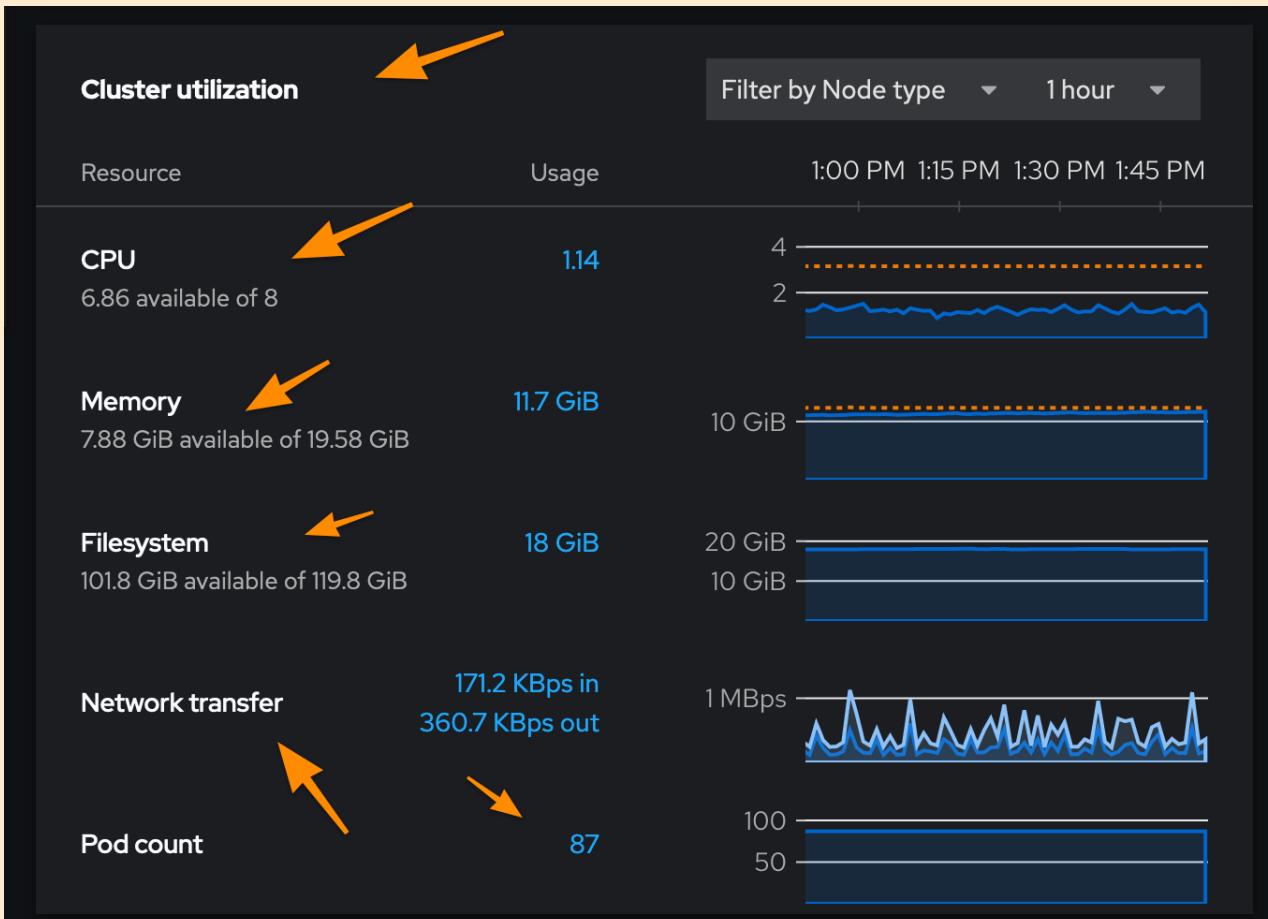


Figure 4. OCP Cluster Utilization

5. Look at and Monitor OCP Node from Terminal

- Compute ⇒ Nodes ⇒ Master01 then open Terminal

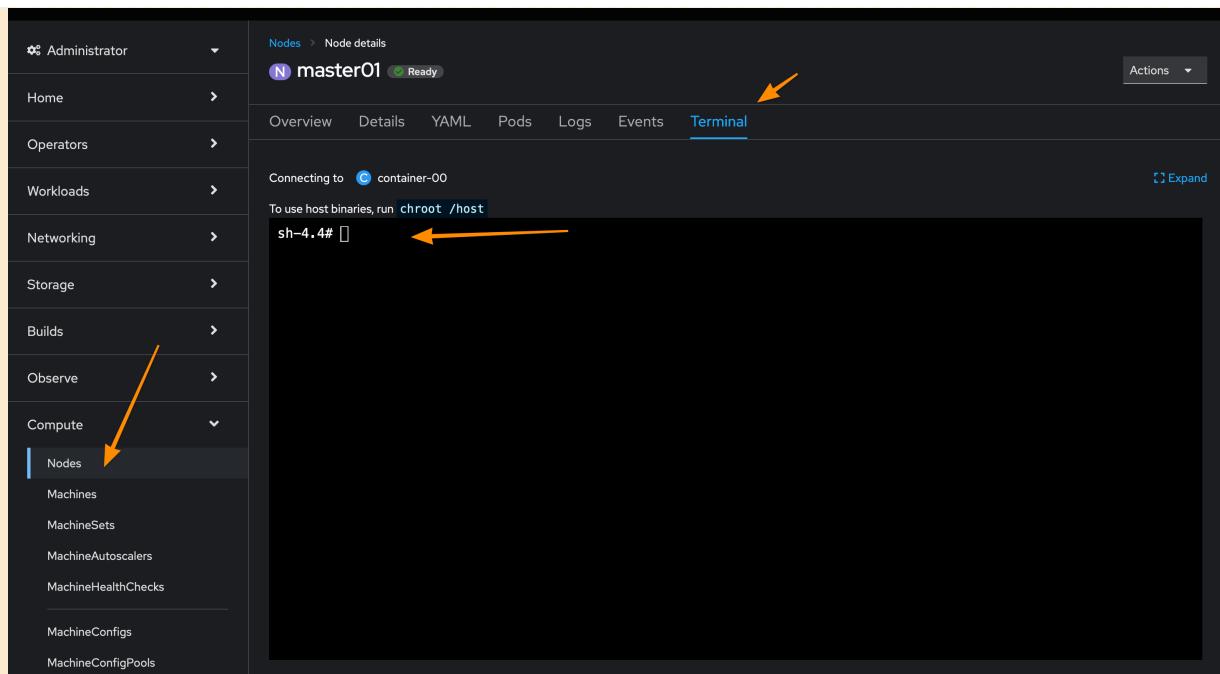


Figure 5. OCP Node Terminal

6. Analyze Node Services

Listing 5. Accessing Host Binaries

```
chroot /host
```

Listing 6. Getting Kubelet Service Status

```
sh-4.4# systemctl status kubelet
● kubelet.service - Kubernetes Kubelet
  Loaded: loaded (/etc/systemd/system/kubelet.service; enabled; vendor preset: disabled)
  Drop-In: /etc/systemd/system/kubelet.service.d
            └─01-kubens.conf, 10-mco-default-madv.conf, 20-logging.conf, 20-nodenet.conf
    Active: active (running) since Wed 2023-04-26 16:05:04 UTC; 1h 54min ago
      Main PID: 3172 (kubelet)
        Tasks: 28 (limit: 127707)
       Memory: 387.5M
          CPU: 20min 41.675s
        CGroup: /system.slice/kubelet.service
```

Listing 7. Obtain CRI-O Container Runtime Service Information

```
sh-4.4# systemctl status crio
● crio.service - Container Runtime Interface for OCI (CRI-O)
  Loaded: loaded (/usr/lib/systemd/system/crio.service; disabled; vendor preset: disabled)
  Active: inactive (dead)
```

```
disabled)
Drop-In: /etc/systemd/system/crio.service.d
└─01-kubens.conf, 10-mco-default-madv.conf, 10-mco-profile-unix-
socket.conf, 20-nodenet.conf
Active: active (running) since Wed 2023-04-26 16:05:03 UTC; 1h 55min ago
Docs: https://github.com/cri-o/cri-o
Main PID: 3134 (crio)
Tasks: 99
Memory: 5.8G
CPU: 13min 33.874s
CGroup: /system.slice/crio.service
└─ 3134 /usr/bin/crio
```

7. Discuss the **Observe** Options

- Alerting
- Metrics
- Dashboards

Chapter 2. Kubernetes and OpenShift Command-Line Interfaces and APIs

2.1. The Kubernetes and OpenShift Command-Line Interfaces

Section Info Here

2.1.1. The Kubernetes and OpenShift Command-Line Interfaces

2.1.1.1. Kubernetes Command-Line Tool

2.1.1.2. The OpenShift Command-Line Tool

2.1.1.3. Managing Resources at the Command Line

2.1.1.4. Authentication with OAuth

2.2. DEMO: Kubernetes and OpenShift Command-Line Interfaces

Credentials

UN/PW: developer/developer

or

UN/PW: admin/redhatocp

- **Web Console:** <https://console-openshift-console.apps.ocp4.example.com>
- **OpenShift API:** <https://api.ocp4.example.com:6443>

Listing 8. Developer Login



```
oc login -u developer -p developer https://api.ocp4.example.com:6443
```

Listing 9. Administrator Login

```
oc login -u admin -p redhatocp https://api.ocp4.example.com:6443
```

kubectl Cheatsheet



https://kubernetes.io/docs/reference/kubectl/_print/#pg-8aba901ac13f124e5782b90ddb166ee2

Example 3. DEMO: Using Kubernetes and OpenShift Command-Line Interfaces

1. Login to OCP Cluster as Administrator

```
[student@workstation ~]$ oc login -u admin -p redhatocp  
https://api.ocp4.example.com:6443  
Login successful.
```

2. Configure Command-Line Completion for Kubectl

```
[student@workstation ~]$ echo "source <(kubectl completion bash)" >> ~/.bashrc ①  
[student@workstation ~]$ source ~/.bashrc ②
```

① Add **kubectl** Completion to BASHRC

② Source the BASHRC file to make it active

3. Test **kubectl** BASH completion and list the help

```
[student@workstation ~]$ kubectl
alpha      cluster-info  diff      label      run
annotate   completion   drain     logs       scale
api-resources config     edit      options    set
api-versions cordon    exec      patch     taint
apply      cp          explain   plugin    top
attach     create      expose    port-forward uncordon
auth       debug      get      proxy      version
autoscale  delete     help      replace    wait
certificate describe   kustomize rollout
```

```
[student@workstation ~]$ kubectl help
kubectl controls the Kubernetes cluster manager.
```

Find more information at: <https://kubernetes.io/docs/reference/kubectl/>

Basic Commands (Beginner):

create	Create a resource from a file or from stdin
expose	Take a replication controller, service, deployment or pod and expose it as a new Kubernetes service
run	Run a particular image on the cluster
set	Set specific features on objects

Basic Commands (Intermediate):

explain	Get documentation for a resource
get	Display one or many resources
edit	Edit a resource on the server
delete	Delete resources by file names, stdin, resources and names, or by resources and label selector

Deploy Commands:

rollout	Manage the rollout of a resource
scale	Set a new size for a deployment, replica set, or replication controller
autoscale	Auto-scale a deployment, replica set, stateful set, or replication controller

Cluster Management Commands:

certificate	Modify certificate resources.
cluster-info	Display cluster information
top	Display resource (CPU/memory) usage
cordon	Mark node as unschedulable

uncordon	Mark node as schedulable
drain	Drain node in preparation for maintenance
taint	Update the taints on one or more nodes

Troubleshooting and Debugging Commands:

describe	Show details of a specific resource or group of resources
logs	Print the logs for a container in a pod
attach	Attach to a running container
exec	Execute a command in a container
port-forward	Forward one or more local ports to a pod
proxy	Run a proxy to the Kubernetes API server
cp	Copy files and directories to and from containers
auth	Inspect authorization
debug	Create debugging sessions for troubleshooting workloads and nodes

Advanced Commands:

diff	Diff the live version against a would-be applied version
apply	Apply a configuration to a resource by file name or stdin
patch	Update fields of a resource
replace	Replace a resource by file name or stdin
wait	Experimental: Wait for a specific condition on one or many resources
kustomize	Build a kustomization target from a directory or URL.

Settings Commands:

label	Update the labels on a resource
annotate	Update the annotations on a resource
completion	Output shell completion code for the specified shell (bash, zsh, fish, or powershell)

Other Commands:

alpha	Commands for features in alpha
api-resources	Print the supported API resources on the server
api-versions	Print the supported API versions on the server, in the form of "group/version"
config	Modify kubeconfig files
plugin	Provides utilities for interacting with plugins
version	Print the client and server version information

Usage:

`kubectl [flags] [options]`

Use "`kubectl <command> --help`" for more information about a given command.

Use "`kubectl options`" for a list of global command-line options (applies to all commands).

4. Obtain Version information with **kubectl** and **oc** commands

```
[student@workstation ~]$ kubectl version --short  
Flag --short has been deprecated, and will be removed in the future. The --short  
output will become the default.  
Client Version: v1.24.1  
Kustomize Version: v4.5.7  
Server Version: v1.25.4+77bec7a
```

```
[student@workstation ~]$ oc version  
Client Version: 4.12.0  
Kustomize Version: v4.5.7  
Server Version: 4.12.0  
Kubernetes Version: v1.25.4+77bec7a
```

Differences between **kubectl** and **oc** Commands



It should be noted that the **kubectl** command cannot display OpenShift specific information and can't act on OpenShift-only resources. The command can only act on the general Kubernetes resources and information and for anything OpenShift-specific, the **oc** command must be used.

5. Get Cluster Info using **oc** and **kubectl**

```
[student@workstation ~]$ kubectl cluster-info  
Kubernetes control plane is running at https://api.ocp4.example.com:6443
```

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.

```
[student@workstation ~]$ oc cluster-info  
Kubernetes control plane is running at https://api.ocp4.example.com:6443
```

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.

6. Obtain information about OCP Nodes

```
[student@workstation ~]$ kubectl get nodes  
NAME      STATUS    ROLES          AGE     VERSION  
master01   Ready     control-plane,master,worker   22d    v1.25.4+77bec7a
```

```
[student@workstation ~]$ oc get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
master01	Ready	control-plane,master,worker	22d	v1.25.4+77bec7a

Available kubectl Commands and oc Commands

There are several more commands supported for OpenShift using the **oc** command vs. using the **kubectl** command.

Listing 10. kubectl Commands

```
[student@workstation ~]$ kubectl
alpha      autoscale   create      exec      logs
rollout    version
annotate   certificate debug      explain    options
run        wait
api-resources cluster-info delete    expose    patch
scale
api-versions completion  describe   get       plugin
set
apply      config      diff       help     port-
forward   taint
attach     cordon      drain      kustomize proxy
top
auth      cp          edit       label    replace
uncordon
```



Listing 11. oc Commands

```
[student@workstation ~]$ oc
adm      cluster-info edit      import-image observe
proxy    secrets
annotate completion exec      kustomize options
registry set
api-resources config   explain   label    patch
replace start-build
api-versions cp       expose    login    plugin
rollback status
apply    create    extract   logout   policy
rollout tag
attach   debug    get      logs    port-
forward rsh      version
auth    delete   help     new-app process
rsync
autoscale describe  idle    new-build project
run
cancel-build diff     image   new-project projects
```

scale

2.3. Inspect Kubernetes Resources

Section Info Here

2.3.1. Kubernetes and OpenShift Resources

2.3.1.1. Structure of Resources

2.3.2. Command Outputs

2.3.2.1. YAML Output

2.3.2.2. JSON Output

2.3.2.3. Custom Output

2.4. DEMO: Inspect Kubernetes Resources

Credentials

UN/PW: developer/developer

or

UN/PW: admin/redhatocp

- **Web Console:** <https://console-openshift-console.apps.ocp4.example.com>
- **OpenShift API:** <https://api.ocp4.example.com:6443>

Listing 12. Developer Login



```
oc login -u developer -p developer https://api.ocp4.example.com:6443
```

Listing 13. Administrator Login

```
oc login -u admin -p redhatocp https://api.ocp4.example.com:6443
```

Example 4. DEMO: Inspect Kubernetes Resources

1. Ensure resources exist

```
[student@workstation ~]$ lab start cli-resources
```

2. Login and switch to correct OCP Project

```
[student@workstation ~]$ oc login -u developer -p developer  
https://api.ocp4.example.com:6443  
Login successful.
```

```
[student@workstation ~]$ oc project cli-resources
```

3. List running pods

```
[student@workstation ~]$ kubectl get pods  
NAME                  READY   STATUS    RESTARTS   AGE  
myapp-77fb5cd997-nbb27 1/1     Running   0          4m40s
```

```
[student@workstation ~]$ oc get pods
NAME           READY   STATUS    RESTARTS   AGE
myapp-77fb5cd997-nbb27   1/1     Running   0          3m56s
```

4. Get details about pod using **describe** or **get** sub-command and format output as YAML

```
[student@workstation ~]$ kubectl describe pod myapp-77fb5cd997-nbb27
Name:           myapp-77fb5cd997-nbb27
Namespace:      cli-resources
Priority:       0
Service Account: default
Node:           master01/192.168.50.10
```

Listing 14. kubectl get Command on Pod

```
[student@workstation ~]$ kubectl get pod myapp-77fb5cd997-nbb27 -o yaml
apiVersion: v1
kind: Pod
metadata:
  annotations:

... OUTPUT OMITTED ...

  hostIP: 192.168.50.10
  phase: Running
  podIP: 10.8.0.82
  podIPs:
    - ip: 10.8.0.82
  qosClass: BestEffort
  startTime: "2023-04-26T19:12:36Z"
```

Listing 15. oc get Command on Pod

```
[student@workstation ~]$ oc get pod myapp-77fb5cd997-nbb27 -o yaml
apiVersion: v1
kind: Pod
metadata:
  annotations:

... OUTPUT OMITTED ...

  hostIP: 192.168.50.10
  phase: Running
  podIP: 10.8.0.82
  podIPs:
```

```
- ip: 10.8.0.82
qosClass: BestEffort
startTime: "2023-04-26T19:12:36Z"
```

5. Obtain Information on Deployments

```
[student@workstation ~]$ kubectl describe deployment myapp
Name:           myapp
Namespace:      cli-resources
CreationTimestamp:  Wed, 26 Apr 2023 15:12:36 -0400
Labels:          app=myapp
Annotations:    deployment.kubernetes.io/revision: 1
Selector:        app=myapp
Replicas:       1 desired | 1 updated | 1 total | 1 available | 0
                unavailable
StrategyType:   RollingUpdate
MinReadySeconds: 0
RollingUpdateStrategy: 25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=myapp
  Containers:
    myapp:
      Image:      registry.ocp4.example.com:8443/ubi8/httpd-24:1-215
      Port:       <none>
      Host Port: <none>
      Environment: <none>
      Mounts:     <none>
      Volumes:    <none>
  Conditions:
    Type  Status  Reason
    ----  -----  -----
    Available  True  MinimumReplicasAvailable
    Progressing  True  NewReplicaSetAvailable
  OldReplicaSets: <none>
  NewReplicaSet:  myapp-77fb5cd997 (1/1 replicas created)
Events:
  Type  Reason          Age   From            Message
  ----  -----          ----  --  -----
  Normal  ScalingReplicaSet  14m  deployment-controller  Scaled up replica set
  myapp-77fb5cd997 to 1
```

6. Cleanup Resources

```
[student@workstation ~]$ lab finish cli-resources
```

2.5. Assess the Health of an OpenShift Cluster

Section Info Here

2.5.1. Query Operator Conditions

Operator Lifecycle Manager (OLM) Operators

sdf

2.5.2. Examining Cluster Metrics

2.5.2.1. Viewing Cluster Metrics

2.5.2.2. Viewing Project Metrics

2.5.2.3. Viewing Resource Metrics

2.5.2.4. Performing Prometheus Queries in the Web Console

2.5.3. Query Cluster Events and Alerts

2.5.3.1. Kubernetes Alerts

2.5.4. Check Node Status

2.5.4.1. Check Pod Status

2.5.5. Collect Information for Support Requests

2.6. DEMO: Assess the Health of an OpenShift Cluster

Credentials

UN/PW: developer/developer

or

UN/PW: admin/redhatocp

- **Web Console:** <https://console-openshift-console.apps.ocp4.example.com>
- **OpenShift API:** <https://api.ocp4.example.com:6443>

Listing 16. Developer Login



```
oc login -u developer -p developer https://api.ocp4.example.com:6443
```

Listing 17. Administrator Login

```
oc login -u admin -p redhatocp https://api.ocp4.example.com:6443
```

Example 5. DEMO: Assessing OCP Cluster Health

1. Prepare environment

```
[student@workstation ~]$ lab start cli-health
```

2. Login as Admin

```
[student@workstation ~]$ oc login -u admin -p redhatocp  
https://api.ocp4.example.com:6443  
Login successful.
```

3. View Operators

```
[student@workstation ~]$ kubectl get operators  
NAME          AGE  
lvms-operator.openshift-storage  22d  
metallb-operator.metallb-system  22d
```

4. View Cluster Operators

Cluster Operators				
Name	Version	Available	Progressing	
authentication	4.12.0	True	False	
False 21m				
baremetal	4.12.0	True	False	
False 22d				
cloud-controller-manager	4.12.0	True	False	
False 22d				
cloud-credential	4.12.0	True	False	
False 22d				
cluster-autoscaler	4.12.0	True	False	
False 22d				
config-operator	4.12.0	True	False	
False 22d				
console	4.12.0	True	False	
False 22d				
control-plane-machine-set	4.12.0	True	False	
False 22d				
csi-snapshot-controller	4.12.0	True	False	
False 22d				
dns	4.12.0	True	False	
False 22d				
etcd	4.12.0	True	False	
False 22d				
image-registry	4.12.0	True	False	
False 22d				
ingress	4.12.0	True	False	
False 22d				
insights	4.12.0	True	False	
False 106s				
kube-apiserver	4.12.0	True	False	
False 22d				
kube-controller-manager	4.12.0	True	False	
False 22d				
kube-scheduler	4.12.0	True	False	
False 22d				
kube-storage-version-migrator	4.12.0	True	False	
False 22d				
machine-api	4.12.0	True	False	
False 22d				
machine approver	4.12.0	True	False	
False 22d				
machine-config	4.12.0	True	False	
False 22d				
marketplace	4.12.0	True	False	
False 22d				

monitoring		4.12.0	True	False
False	3h58m			
network		4.12.0	True	False
False	22d			
node-tuning		4.12.0	True	False
False	22d			
openshift-apiserver		4.12.0	True	False
False	3h59m			
openshift-controller-manager		4.12.0	True	False
False	3h56m			
openshift-samples		4.12.0	True	False
False	22d			
operator-lifecycle-manager		4.12.0	True	False
False	22d			
operator-lifecycle-manager-catalog		4.12.0	True	False
False	22d			
operator-lifecycle-manager-packageserver		4.12.0	True	False
False	3h59m			
service-ca		4.12.0	True	False
False	22d			
storage		4.12.0	True	False
False	22d			

5. View etcd Pods

```
[student@workstation ~]$ kubectl top pods etcd-master01 -n openshift-etcd
--containers
POD           NAME          CPU(cores)   MEMORY(bytes)
etcd-master01  POD          0m          0Mi
etcd-master01  etcd          87m         1146Mi
etcd-master01  etcd-metrics  11m         25Mi
etcd-master01  etcd-readyz  6m          39Mi
etcd-master01  etcdctl       0m          0M
```

6. Check Node Status

```
[student@workstation ~]$ kubectl get nodes
NAME      STATUS    ROLES          AGE     VERSION
master01  Ready     control-plane,master,worker  22d    v1.25.4+77bec7a
```

Listing 18. Detailed Node Information regarding CPU

```
[student@workstation ~]$ kubectl get node master01 -o jsonpath=\'Allocatable: {.status.allocatable.cpu}{\"\n\"}\'Capacity: {.status.capacity.cpu}{\"\n\"}\'\'
Allocatable: 7500m
```

Capacity: 8

7. Get Node Stats with **top** Command

```
[student@workstation ~]$ kubectl top node
NAME      CPU(cores)   CPU%    MEMORY(bytes)   MEMORY%
master01   1714m       22%    12770Mi        67%
```

8. Open a Debug shell to nodes

```
[student@workstation ~]$ oc debug node/master01
Temporary namespace openshift-debug-9h2p9 is created for debugging node...
Starting pod/master01-debug ...
To use host binaries, run `chroot /host`
Pod IP: 192.168.50.10
If you don't see a command prompt, try pressing enter.
sh-4.4#
```

```
sh-4.4# chroot /host
sh-4.4#
```

Node Console Session



Debug pods can open a console session just like we did earlier when in the WebUI where we accessed the Node console through the GUI. It is important to remember that access to the OCP Nodes can only be done through a Debug console and the WebUI console. You can't **normally** login to an OCP node through SSH unless you have the **CoreOS** SSH keys and known credentials for the **core** user.

9. Verify Services

Listing 19. Verifying CRI-O Service

```
sh-4.4# systemctl is-active crio
active
```

Listing 20. Verifying the Kubelet Service

```
sh-4.4# systemctl is-active kubelet
active
```

10. Additional Verifications

Listing 21. Using crictl to Get Info

```
sh-4.4# crictl info
{
  "status": {
    "conditions": [
      {
        "type": "RuntimeReady",
        "status": true,
        "reason": "",
        "message": ""
      },
      {
        "type": "NetworkReady",
        "status": true,
        "reason": "",
        "message": ""
      }
    ]
  }
}
```

11. List Pods running on a Node

```
sh-4.4# crictl pods
POD ID          CREATED           STATE          ATTEMPT      NAME          RUNTIME
NAMESPACE
7a46b73b46311   6 minutes ago   Ready          0            master01-debug
openshift-debug-9h2p9
dc46f0c6e6381    14 minutes ago  NotReady       collect-profiles-
28042335-ktxhc
0                (default)
e9468576e3417    29 minutes ago  NotReady       collect-profiles-
28042320-m45fr
...
... OUTPUT OMITTED ...
```

12. Cleanup Lab environment

```
lab finish cli-health
```

Chapter 3. Run Applications as Containers and Pods

3.1. Introduction to Linux Containers and Kubernetes Pods

Section Info Here

3.1.1. Containers Overview

3.1.2. Kubernetes and Containers

3.1.3. Creating Containers and Pods

3.1.4. User and Group IDs Assignment

3.1.4.1. Pod Security

3.1.5. Execute Commands in Running Containers

3.1.6. Container Logs

3.1.7. Deleting Resources

3.1.8. The CRI-O Container Engine

3.2. DEMO: Introduction to Linux Containers and Kubernetes Pods

Credentials

UN/PW: developer/developer

or

UN/PW: admin/redhatocp

- **Web Console:** <https://console-openshift-console.apps.ocp4.example.com>
- **OpenShift API:** <https://api.ocp4.example.com:6443>

Listing 22. Developer Login



```
oc login -u developer -p developer https://api.ocp4.example.com:6443
```

Listing 23. Administrator Login

```
oc login -u admin -p redhatocp https://api.ocp4.example.com:6443
```

Example 6. DEMO: Linux Containers and Kubernetes Pods (Introduction)

1. Show Container Image Registry
 - **URL:** <https://registry.ocp4.example.com:8443/repository/>
 - **UN/PW Combination:** developer/developer

The screenshot shows the Quay web interface with the following details:

- Header:** QUAY, EXPLORE, REPOSITORIES (highlighted), TUTORIAL, search bar, a bell icon, and a 'D developer...' button.
- Section Title:** Repositories
- Table Headers:** REPOSITORY NAME, LAST MODIFIED, STATE, ACTIVITY (with a downward arrow), and STAR.
- Table Data:** A list of 12 repositories, each with a color-coded icon (O, R, U) and a brief description:
 - O openshift / release
 - R rhel8 / mysql-80
 - U ubi8 / nodejs-16
 - R rhel8 / postgresql-12
 - R rhel8 / postgresql-13
 - U ubi8 / python-39
 - R redhat-openjdk-18 / openjdk18-openshift
 - R rhsc1 / postgresql-10-rhel7
 - U ubi8 / php-74
 - U ubi8 / php-73
 - R rhsc1 / httpd-24-rhel7
- Sidebar:** 'Users and Organizations' section listing various developer accounts with their corresponding icons.

Figure 6. Internal Classroom Registry - Quay

2. Ensure Lab Environment is Ready

```
[student@workstation ~]$ lab start pods-containers
```

3. Login to OCP as a Developer

```
[student@workstation ~]$ oc login -u developer -p developer
https://api.ocp4.example.com:6443
Login successful
```

4. Create Demo Project

```
[student@workstation ~]$ oc new-project container-demo
```

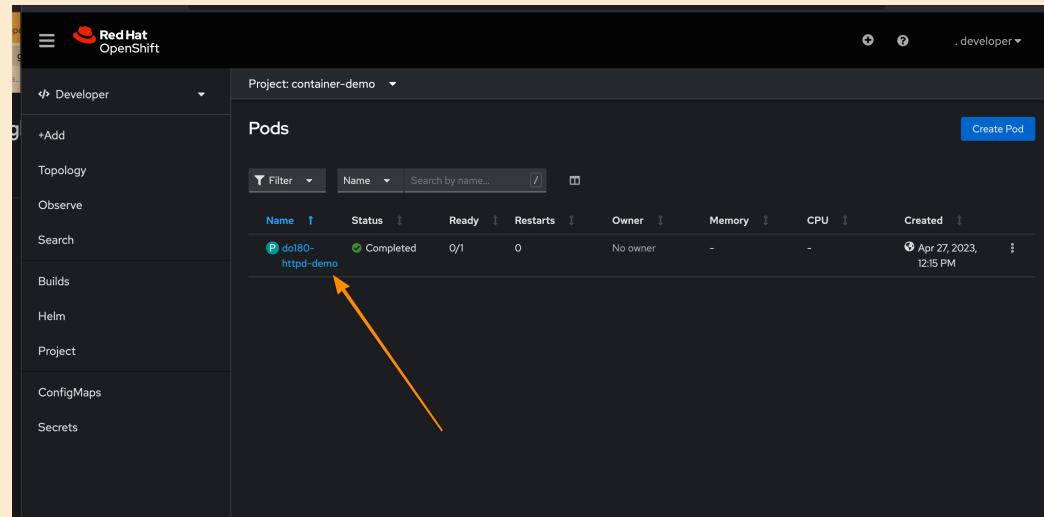
5. Run a command from a container image

```
[student@workstation ~]$ oc run -it do180-httpd-demo --restart 'Never' --image
registry.ocp4.example.com:8443/redhattraining/do180-httpd-app:v1 -- /bin/bash -c
"whoami && id && cat /etc/redhat-release"
1000710000
uid=100071000(100071000) gid=0(root) groups=0(root),100071000
Red Hat Enterprise Linux release 8.7 (Ootpa)
```



[View Container in OCP WebUI](#)

The pod and container were running long enough for the BASH commands to execute and then the container completed, so the pod no longer has a running container.



Name	Status	Ready	Restarts	Owner	Memory	CPU	Created
do180-httpd-demo	Completed	0/1	0	No owner	-	-	Apr 27, 2023, 12:15 PM

6. Remove Pod

```
[student@workstation ~]$ oc delete pod do180-httpd-demo
```

7. Gain Interactive CLI Prompt in Container

```
[student@workstation ~]$ oc run -it do180-httpd-demo --restart 'Never' --image registry.ocp4.example.com:8443/redhattraining/do180-httpd-app:v1 -- /bin/bash
If you don't see a command prompt, try pressing enter
```

8. Test content of web directory

```
bash-4.4$ cat /var/www/html/index.php
<?php
$hostname = gethostname();
print "Welcome to Red Hat Training, from $hostname\r\n";
?>
```

9. Change content of web container and then exit the container shell.

```
bash-4.4$ echo "I'm welcoming you from Index.html" > /var/www/html/index.html
bash-4.4$ ls /var/www/html/
index.html index.php
```

```
bash-4.4$ cat /var/www/html/index.html  
I'm welcoming you from Index.html  
  
bash-4.4$ exit  
exit  
[student@workstation ~]$
```

10. Remove existing container and restart container to show content is missing as the container storage is ephemeral.

```
[student@workstation ~]$ oc delete pod do180-httpd-demo  
pod "do180-httpd-demo" deleted
```

```
[student@workstation ~]$ oc run -it do180-httpd-demo --restart 'Never' --image  
registry.ocp4.example.com:8443/redhattraining/do180-httpd-app:v1 -- /bin/bash
```

```
[student@workstation ~]$ oc run -it do180-httpd-demo --restart 'Never' --image  
registry.ocp4.example.com:8443/redhattraining/do180-httpd-app:v1 -- /bin/bash  
If you don't see a command prompt, try pressing enter.
```

```
bash-4.4$ ls /var/www/html/  
index.php
```

Ephemeral Storage



When launching a new pod (container) with the same image, it is easy to see that the index.html file created on the previous pod does not exist in this new instance. That is because running containers have ephemeral storage (unless persistent storage is added).

11. Run image and demonstrate **oc exec** Command

```
[student@workstation ~]$ oc run do180-httpd-demo --restart 'Never' --image  
registry.ocp4.example.com:8443/redhattraining/do180-httpd-app:v1  
pod/do180-httpd-demo created
```

Listing 24. Verify Running Pods

```
[student@workstation ~]$ oc get pods  
NAME          READY   STATUS    RESTARTS   AGE  
do180-httpd-demo  1/1     Running   0          13s
```

Listing 25. Create Shell Into Pod with oc exec Command

```
[student@workstation ~]$ oc exec -it pods/do180-httpd-demo -- /bin/bash  
bash-4.4$
```

**Difference Between oc attach and oc exec**

The **oc attach** command will drop the user into the existing processes and attempt to attach to a shell. Alternatively, the **oc exec** command will create a new shell process inside the container. This is often needed as some containers are running system services and processes and attach would place you in that same shell.

Listing 26. Demo of Running HTTPD Server on Container

```
bash-4.4$ curl localhost:8080  
Welcome to Red Hat Training, from do180-httpd-demo
```

12. Clean up Project

```
[student@workstation ~]$ oc delete project container-demo  
project.project.openshift.io "container-demo" deleted
```

3.3. Find and Inspect Container Images

Section Info Here

3.3.1. Container Image Overview

3.3.2. Container Image Registries

3.3.2.1. Red Hat Registry

3.3.2.2. Quay.io

3.3.2.3. Private Registries

3.3.3. Container Image Identifiers

3.3.4. Container Image Components

3.3.5. Container Image Instructions and Metadata

3.3.6. Base Images

3.3.7. Inspecting and Managing Container Images

3.3.7.1. Skopeo

3.3.7.2. Registry Credentials

3.3.7.3. Theoc imageCommand

3.3.8. Running Containers as Root

3.4. DEMO: Find and Inspect Container Images

Credentials

UN/PW: developer/developer

or

UN/PW: admin/redhatocp

- **Web Console:** <https://console-openshift-console.apps.ocp4.example.com>
- **OpenShift API:** <https://api.ocp4.example.com:6443>

Listing 27. Developer Login



```
oc login -u developer -p developer https://api.ocp4.example.com:6443
```

Listing 28. Administrator Login

```
oc login -u admin -p redhatocp https://api.ocp4.example.com:6443
```

Example 7. DEMO: Find and Inspect Container Images

1. Ensure Lab Environment is Ready

```
[student@workstation ~]$ lab start pods-images
```

2. Create a Project

```
[student@workstation ~]$ oc new-project pod-image-demo
```

3. Login to Container Image Registry

```
[student@workstation ~]$ skopeo login registry.ocp4.example.com:8443
Username: developer
Password:
Login Succeeded!
```

4. Use **skopeo** to list available image tags

```
[student@workstation ~]$ skopeo list-tags
```

```
docker://registry.ocp4.example.com:8443/redhattraining/do180-httpd-app
{
  "Repository": "registry.ocp4.example.com:8443/redhattraining/do180-httpd-app",
  "Tags": [
    "v1"
  ]
}
```

Listing 29. Source Description

```
[student@workstation ~]$ skopeo inspect --config
docker://registry.ocp4.example.com:8443/redhattraining/do180-httpd-app:v1

... OUTPUT OMITTED ...

"config": {
  "User": "1001", ①
  "ExposedPorts": {
    "8080/tcp": {} ②
  },
  "Env": [
    "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin",
    "container=oci"
  ],
  "Cmd": [
    "/bin/sh",
    "-c",
    "php-fpm -D ; httpd -D FOREGROUND" ③
  ],
}

... OUTPUT OMITTED ...
```

① Shows user that container image is using

② Shows the exposed ports in the image

③ Specifies the command to be run when a container is launched with this container image.

1. Gather image information using the **oc image info** command

```
[student@workstation ~]$ oc image info
registry.ocp4.example.com:8443/redhattraining/do180-httpd-app:v1
Name:      registry.ocp4.example.com:8443/redhattraining/do180-httpd-app:v1
Digest:
sha256:5d8b36a4a557215ccfc5e30af8665528c7bf96a6589009bc66041d7fae0e76f4
Media Type: application/vnd.oci.image.manifest.v1+json

... OUTPUT OMITTED ...
```

```
Command:      /bin/sh -c php-fpm -D ; httpd -D FOREGROUND
User:        1001 ①
Expose Ports: 8080/tcp ②

... OUTPUT OMITTED ...
```

④ Container image user

⑤ Container image exposed ports

- Run the container image as a container

```
[student@workstation ~]$ oc run do180-httpd-demo --restart 'Never' --image
registry.ocp4.example.com:8443/redhattraining/do180-httpd-app:v1
pod/do180-httpd-demo created
```

- Obtain POD IP address

```
[student@workstation ~]$ oc get pods do180-httpd-demo -o json | jq
.status.podIP
"10.8.0.80"
```

- Login as **administrator** user and open a Debug Pod on the Master01 node

```
[student@workstation ~]$ oc login -u admin -p redhatocp
https://api.ocp4.example.com:6443
```

```
[student@workstation ~]$ oc debug node/master01
sh-4.4# chroot /host
```

- Verify the container network and website

```
sh-4.4# curl 10.8.0.80:8080
Welcome to Red Hat Training, from do180-httpd-demo
```

Compute Node and SDN Container Networking



Since we've switched to the compute node where the container is running, and we've launched a debug shell we have access to the SDN container network. It is possible to use the networking to check network services

from pods/containers.

5. Exit out of Debug Node Shell and Return to the Developer User

```
sh-4.4# exit  
exit  
sh-4.4# exit  
exit  
  
Removing debug pod ...  
Temporary namespace openshift-debug-m49ll was removed.
```

```
[student@workstation ~]$ oc login -u developer -p developer  
https://api.ocp4.example.com:6443  
Login successful
```

6. Remove demo pods and project

```
[student@workstation ~]$ oc delete project pod-image-demo  
project.project.openshift.io "pod-image-demo" deleted
```

3.5. Troubleshoot Containers and Pods

Section Info Here

3.5.1. Container Troubleshooting Overview

3.5.2. CLI Troubleshooting Tools

3.5.3. Editing Resources

3.5.4. Copy Files to and from Containers

3.5.5. Remote Container Access

3.5.6. Connect to Running Containers

3.5.7. Execute Commands in a Container

3.5.8. Container Events and Logs

3.5.9. Available Linux Commands in Containers

3.5.10. Troubleshooting from Inside the Cluster

3.6. DEMO: Troubleshoot Containers and Pods

Credentials

UN/PW: developer/developer

or

UN/PW: admin/redhatocp

- **Web Console:** <https://console-openshift-console.apps.ocp4.example.com>
- **OpenShift API:** <https://api.ocp4.example.com:6443>

Listing 30. Developer Login



```
oc login -u developer -p developer https://api.ocp4.example.com:6443
```

Listing 31. Administrator Login



```
oc login -u admin -p redhatocp https://api.ocp4.example.com:6443
```

Getting VIM as the editor and setting up guides for the oc edit command.

Listing 32. .vimrc File contents

```
setlocal ai ts=2 sw=2 et colorcolumn=3,5,7,9,11
```

Listing 33. Setting Environment Variables

```
[student@workstation ~]$ export OC_EDITOR=vim
```

```
[student@workstation ~]$ export EDITOR=vim
```

Example 8. DEMO: Troubleshoot Containers and Pods

1. Ensure environment is ready and login as a developer

```
[student@workstation ~]$ lab start pods-troubleshooting
```

```
[student@workstation ~]$ oc login -u developer -p developer https://api.ocp4.example.com:6443
```

2. Create Troubleshooting Demo Project

```
[student@workstation ~]$ oc new-project troubleshooting-demo
```

3. Launch Container

```
[student@workstation ~]$ oc run do180-httpd-demo --restart 'Never' --image
registry.ocp4.example.com:8443/redhattraining/do180-httpd-app
pod/do180-httpd-demo created
```

4. View Pod Status

```
[student@workstation ~]$ oc get pods
NAME           READY   STATUS      RESTARTS   AGE
do180-httpd-demo   0/1     ErrImagePull   0          33s
```

5. Get Events

```
[student@workstation ~]$ oc get events
LAST SEEN    TYPE      REASON          OBJECT          MESSAGE
70s          Normal    Scheduled        pod/do180-httpd-demo  Successfully
assigned troubleshooting-demo/do180-httpd-demo to master01
70s          Normal    AddedInterface  pod/do180-httpd-demo  Add eth0
[10.8.0.83/23] from ovn-kubernetes
29s          Normal    Pulling         pod/do180-httpd-demo  Pulling image
"registry.ocp4.example.com:8443/redhattraining/do180-httpd-app"
29s          Warning   Failed          pod/do180-httpd-demo  Failed to pull image
"registry.ocp4.example.com:8443/redhattraining/do180-httpd-app": rpc error: code =
Unknown desc = reading manifest latest in
registry.ocp4.example.com:8443/redhattraining/do180-httpd-app: manifest unknown:
manifest unknown ①
```

① Showing that manifest is unknown and image isn't available

6. Look at Logs

```
[student@workstation ~]$ oc logs do180-httpd-demo
Error from server (BadRequest): container "do180-httpd-demo" in pod "do180-httpd-
demo" is waiting to start: trying and failing to pull image
```

7. Use the **skopeo** command to inspect images and the registry

```
[student@workstation ~]$ skopeo login registry.ocp4.example.com:8443
```

```
[student@workstation ~]$ skopeo list-tags
docker://registry.ocp4.example.com:8443/redhattraining/do180-httpd-app
{
    "Repository": "registry.ocp4.example.com:8443/redhattraining/do180-httpd-app",
    "Tags": [
        "v1" ①
    ]
}
```

- ① In this instance, the repository and container image are here, but there is no **LATEST** tag and only the **v1** tag

8. Inspect the image with the OC command and correct tag

```
[student@workstation ~]$ oc image info
registry.ocp4.example.com:8443/redhattraining/do180-httpd-app:v1
Name:          registry.ocp4.example.com:8443/redhattraining/do180-httpd-app:v1
Digest:        sha256:5d8b36a4a557215ccfc5e30af8665528c7bf96a6589009bc66041d7fae0e76f4
Media Type:   application/vnd.oci.image.manifest.v1+json
```

9. Edit the POD with the **oc edit** command and provide the correct tag to the image.

```
[student@workstation ~]$ oc edit pod/do180-httpd-demo
...
spec:
  containers:
    - image: registry.ocp4.example.com:8443/redhattraining/do180-httpd-app:v1 ①
      imagePullPolicy: Always
      name: do180-httpd-demo
      resources: {}

...
pod/do180-httpd-demo edited ②
```

- ① Locate the **image** and add the image tag using **:v1** to the end of the file.
② After saving the file, the **POD** definition will show as updated.

10. Get Pod status

```
[student@workstation ~]$ oc get pods
NAME           READY   STATUS    RESTARTS   AGE
do180-httpd-demo   1/1     Running   0          14m
```

11. Open an **rsh** connection into the container

```
[student@workstation ~]$ oc rsh do180-httpd-demo
sh-4.4$
```

12. Verify web services

```
sh-4.4$ curl localhost:8080
Welcome to Red Hat Training, from do180-httpd-demo
```

13. Exit **rsh** session

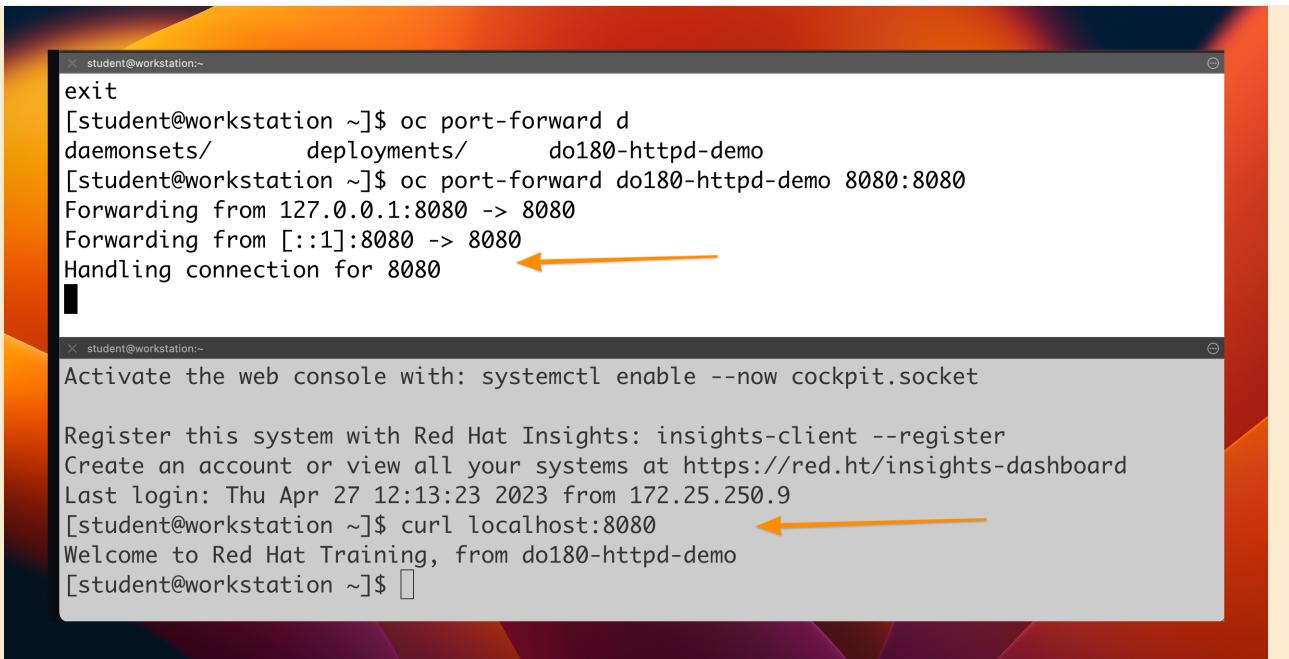
```
sh-4.4$ exit
exit
[student@workstation ~]$
```

14. Test Connectivity Into Container using **oc port-forward** to forward (localhost) port to container

```
[student@workstation ~]$ oc port-forward do180-httpd-demo 8080:8080
Forwarding from 127.0.0.1:8080 -> 8080
Forwarding from [::1]:8080 -> 8080
```

```
[student@workstation ~]$ curl localhost:8080 ①
Welcome to Red Hat Training, from do180-httpd-demo
```

① Perform curl from a second terminal



The screenshot shows two terminal windows. The top window displays the command `oc port-forward` being used to forward port 8080 from a local host to a container named `do180-httdp-demo`. The bottom window shows the result of a `curl` command to the forwarded port, displaying a welcome message from Red Hat Training.

```
student@workstation:~$ oc port-forward d
daemons/      deployments/      do180-httdp-demo
[student@workstation ~]$ oc port-forward do180-httdp-demo 8080:8080
Forwarding from 127.0.0.1:8080 -> 8080
Forwarding from [::1]:8080 -> 8080
Handling connection for 8080
```

```
Activate the web console with: systemctl enable --now cockpit.socket

Register this system with Red Hat Insights: insights-client --register
Create an account or view all your systems at https://red.ht/insights-dashboard
Last login: Thu Apr 27 12:13:23 2023 from 172.25.250.9
[student@workstation ~]$ curl localhost:8080
Welcome to Red Hat Training, from do180-httdp-demo
[student@workstation ~]$
```

oc port-forward and Networking



The **oc port-forward** command is a way to check containers and networking. Actually using OCP networking and configuring networking for applications will be discussed in a later chapter.

15. Kill the **oc port-forward** and delete the project

```
[student@workstation ~]$ oc delete project troubleshooting-demo
project.project.openshift.io "troubleshooting-demo" deleted
```

Chapter 4. Deploy Managed and Networked Applications on Kubernetes

4.1. Deploy Applications from Images and Templates

Section Info Here

4.1.1. Deploying Applications

4.1.2. Resources and Resource Definitions

4.1.3. Managing Resources

4.1.4. Common Resource Types and Their Uses

4.1.4.1. Templates

4.1.4.2. Pod

4.1.4.3. Deployment Configurations

4.1.4.4. Deployment

4.1.4.5. Projects

4.1.4.6. Services

4.1.4.7. Persistent Volume Claims

4.1.4.8. Secrets

4.1.5. Managing Resources from the Command Line

4.1.5.1. Imperative Resource Management

4.1.5.2. Declarative Resource Management

4.1.6. Retrieving Resource Information

4.2. DEMO: Deploy Applications from Images and Templates

Credentials

UN/PW: **developer/developer**

or

UN/PW: **admin/redhatocp**

- **Web Console:** <https://console-openshift-console.apps.ocp4.example.com>
- **OpenShift API:** <https://api.ocp4.example.com:6443>

Listing 34. Developer Login



```
oc login -u developer -p developer https://api.ocp4.example.com:6443
```

Listing 35. Administrator Login

```
oc login -u admin -p redhatocp https://api.ocp4.example.com:6443
```

Example 9. DEMO: Using OpenShift Templates

1. Ensure lab environment is ready and login as the **developer** user.

```
[student@workstation ~]$ lab start deploy-newapp  
[student@workstation ~]$ oc login -u developer -p developer  
https://api.ocp4.example.com:6443
```

2. Create Demo project

```
[student@workstation ~]$ oc new-project demo-app-template
```

3. Look for Apache template (httpd) and use **oc describe** to get information about template.

```
[student@workstation ~]$ oc describe template httpd -n openshift  
Name:      httpd-example  
Namespace:  openshift  
Created:   3 weeks ago
```

... OUTPUT OMITTED ...

```
Name:      SOURCE_REPOSITORY_URL  
Display Name: Git Repository URL  
Description: The URL of the repository with your application source code.  
Required:   true  
Value:     https://github.com/sclorg/httpd-ex.git
```

... OUTPUT OMITTED ...

```
Name:      CONTEXT_DIR  
Display Name: Context Directory  
Description: Set this to the relative path to your project if it is not in  
the root of your repository.  
Required:   false  
Value:     <none>
```

... OUTPUT OMITTED ...

4. Create a new application with the template

```
[student@workstation ~]$ oc new-app --template httpd-example -p  
SOURCE_REPOSITORY_URL=https://github.com/tmichett/OCP_Demos.git -p  
CONTEXT_DIR=/Web_HTML -p HTTPD_VERSION=2.4-e17  
--> Deploying template "demo-app-template/httpd-example" to project demo-app-  
template
```

5. Check Status and Builds

```
[student@workstation ~]$ oc status  
In project demo-app-template on server https://api.ocp4.example.com:6443  
  
http://httpd-example-demo-app-template.apps.ocp4.example.com (svc/httpd-example)  
dc/httpd-example deploys istag/httpd-example:latest <-  
bc/httpd-example source builds https://github.com/tmichett/OCP_Demos.git on  
openshift/httpd:2.4-e17  
deployment #1 deployed about a minute ago - 1 pod  
  
[student@workstation ~]$ oc get build
```

6. Check Deployed Pods

```
[student@workstation ~]$ oc get pods  
NAME          READY   STATUS    RESTARTS   AGE
```

httpd-example-1-build	0/1	Completed	0	2m10s
httpd-example-1-deploy	0/1	Completed	0	103s
httpd-example-1-lz58b	1/1	Running	0	99s

7. Check application services and routes

```
[student@workstation ~]$ oc get svc
NAME            TYPE      CLUSTER-IP        EXTERNAL-IP    PORT(S)      AGE
httpd-example   ClusterIP  172.30.55.134    <none>        8080/TCP    2m44s
```

```
[student@workstation ~]$ oc get route
NAME          HOST/PORT           PATH
SERVICES      PORT    TERMINATION    WILDCARD
httpd-example httpd-example-demo-app-template.apps.ocp4.example.com
httpd-example <all>              None
```

8. Check running container exposed services

```
[student@workstation ~]$ curl  httpd-example-demo-app-
template.apps.ocp4.example.com
This is a test and demo for the Containerfile build.
```




Figure 7. Website shown in Firefox

9. Clean-up Project

```
[student@workstation ~]$ oc delete project demo-app-template
project.project.openshift.io "demo-app-template" deleted
```

4.3. Manage Long-lived and Short-lived Applications Using the Kubernetes Workload API

Section Info Here

4.3.1. Kubernetes Workload Resources

4.3.1.1. Jobs

4.3.1.2. Cron Jobs

4.3.1.3. Deployments

4.3.1.4. Stateful Sets

4.4. DEMO: Manage Long-lived and Short-lived Applications Using the Kubernetes Workload API

Credentials

UN/PW: developer/developer

or

UN/PW: admin/redhatocp

- **Web Console:** <https://console-openshift-console.apps.ocp4.example.com>
- **OpenShift API:** <https://api.ocp4.example.com:6443>

Listing 36. Developer Login



```
oc login -u developer -p developer https://api.ocp4.example.com:6443
```

Listing 37. Administrator Login

```
oc login -u admin -p redhatocp https://api.ocp4.example.com:6443
```

Example 10. DEMO: Managing Applications using Kubernetes Workload API

1. Verify lab environment is ready

```
[student@workstation ~]$ lab start deploy-workloads
```

2. Login to OCP as a Developer and Create a Project

```
[student@workstation ~]$ oc login -u developer -p developer  
https://api.ocp4.example.com:6443
```

```
[student@workstation ~]$ oc new-project demo-workloads  
Now using project "demo-workloads" on server "https://api.ocp4.example.com:6443".
```

3. Create a deployment for a webserver (Apache/HTTPPD)

Listing 38. Source Description

```
[student@workstation ~]$ oc create deployment demo-web --image
```

```
registry.ocp4.example.com:8443/redhattraining/do180-httdp-app:v1
```

4. Check Deployment and Pod status

```
[student@workstation ~]$ oc get deployment  
NAME      READY  UP-TO-DATE  AVAILABLE  AGE  
demo-web  1/1    1           1           40s
```

```
[student@workstation ~]$ oc get pods -o wide  
NAME                           READY  STATUS   RESTARTS  AGE   IP          NODE  
NOMINATED NODE     READINESS GATES  
demo-web-776b5c97c5-h8v2f  1/1    Running   0        80s  10.8.0.41  
master01 <none>             <none>
```

5. Delete the Pod created by the deployment

```
[student@workstation ~]$ oc delete pod -l app=demo-web  
pod "demo-web-776b5c97c5-h8v2f" deleted
```

6. Verify that since the pod was deleted, the deployment caused a new pod to start.

```
[student@workstation ~]$ oc get pods -o wide  
NAME                           READY  STATUS   RESTARTS  AGE   IP          NODE  
NOMINATED NODE     READINESS GATES  
demo-web-776b5c97c5-hbzft  1/1    Running   0        47s  10.8.0.43  
master01 <none>             <none>
```

7. Clean-Up the Environment

```
[student@workstation ~]$ oc delete project demo-workloads  
project.project.openshift.io "demo-workloads" deleted
```

4.5. Kubernetes Pod and Service Networks

Section Info Here

4.5.1. The Software-defined Network

4.5.2. Kubernetes Networking

4.5.2.1. Using Services

4.5.3. Kubernetes DNS for Service Discovery

4.5.4. Kubernetes Networking Drivers

4.5.4.1. The OpenShift Cluster Network Operator

4.6. DEMO: Kubernetes Pod and Service Networks

Credentials

UN/PW: developer/developer

or

UN/PW: admin/redhatocp

- **Web Console:** <https://console-openshift-console.apps.ocp4.example.com>
- **OpenShift API:** <https://api.ocp4.example.com:6443>

Listing 39. Developer Login



```
oc login -u developer -p developer https://api.ocp4.example.com:6443
```

Listing 40. Administrator Login

```
oc login -u admin -p redhatocp https://api.ocp4.example.com:6443
```

Example 11. DEMO: Using Kubernetes Pod and Service Networks

1. Verify lab environment is ready

```
[student@workstation ~]$ lab start deploy-workloads
```

2. Login to OCP as a Developer and Create a Project

```
[student@workstation ~]$ oc login -u developer -p developer  
https://api.ocp4.example.com:6443
```

```
[student@workstation ~]$ oc new-project demo-services  
Now using project "demo-services" on server "https://api.ocp4.example.com:6443".
```

3. Create a deployment and expose a service port

```
[student@workstation ~]$ oc create deployment demo-service-web --port 8080 --image  
registry.ocp4.example.com:8443/redhattraining/do180-httpd-app:v1
```

4. Retrieve Pod and Deployment Information

```
[student@workstation ~]$ oc get pods -o wide
NAME                               READY   STATUS    RESTARTS   AGE   IP
NODE      NOMINATED-NODE  READINESS GATES
demo-service-web-79b45b9778-h2499  1/1     Running   0          26s   10.8.0.67
master01   <none>           <none>
```

```
[student@workstation ~]$ oc get deployment
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
demo-service-web  1/1       1           1          64s
```

5. Verify no services exist

```
[student@workstation ~]$ oc get service demo-service-web
Error from server (NotFound): services "demo-service-web" not found
```

6. Create a and expose a service

```
[student@workstation ~]$ oc expose deployment/demo-service-web
service/demo-service-web exposed
```

7. View exposed service(s)

```
[student@workstation ~]$ oc get service demo-service-web -o wide
NAME        TYPE        CLUSTER-IP      EXTERNAL-IP    PORT(S)        AGE
SELECTOR
demo-service-web  ClusterIP  172.30.19.218  <none>        8080/TCP    32s
app=demo-service-web
```

8. View Endpoints

```
[student@workstation ~]$ oc get endpoints
NAME        ENDPOINTS        AGE
demo-service-web  10.8.0.68:8080  50s
```

9. Launch a shell using a UBI Container image

```
[student@workstation ~]$ oc run -it ubi9-shell --restart 'Never' --image
registry.ocp4.example.com:8443/ubi9/ubi -- /bin/bash
If you don't see a command prompt, try pressing enter.
```

```
bash-5.1$
```

10. Bring back the website from the other container

Listing 41. Using Container IP Address

```
bash-5.1$ curl 10.8.0.68:8080
Welcome to Red Hat Training, from demo-service-web-6694f6d566-mdmx5
```

Listing 42. Using Service IP Address

```
bash-5.1$ curl 172.30.19.218:8080
Welcome to Red Hat Training, from demo-service-web-6694f6d566-mdmx5
```

Connectivity (Service vs. Container Network)



Both the Service and Container OCP SDNs are available to other running containers in the environment. However, the container network IP address changes each time a new pod is created/deployed.

11. Using a second terminal window, kill the pod and allow the deployment to create a new instance of the web server

```
[student@workstation ~]$ oc delete pod demo-service-web-6694f6d566-mdmx5
pod "demo-service-web-6694f6d566-mdmx5" deleted
```

12. Verify a new Pod has been created and the IP addresses associated with the new pod.

[student@workstation ~]\$ oc get pods -o wide						
NAME	NOMINATED NODE	READY	STATUS	RESTARTS	AGE	IP
NODE	READYNESS GATES					
demo-service-web-6694f6d566-qdfhb	<none>	1/1	Running	0	47s	10.8.0.72
master01	<none>	①				
ubi9-shell		1/1	Running	0	6m9s	10.8.0.71
master01	<none>					

① The IP address has changed now to **10.8.0.72**

13. Perform the same curl commands from the **ubi9-shell** Pod

```
bash-5.1$ curl 10.8.0.68:8080 ①
^C
bash-5.1$ curl 172.30.19.218:8080 ②
```

Welcome to Red Hat Training, from demo-service-web-6694f6d566-qdfhb

- ① It can no longer connect to the old webserver instance as the container IP address has changed.
- ② It can still connect to the demo-web-service as the **ServiceIP** address remains constant as part of the deployment and it just points to the new container IP address automatically.

14. Cleanup the Environment

```
[student@workstation ~]$ oc delete project demo-services  
project.project.openshift.io "demo-services" deleted
```

4.7. Scale and Expose Applications to External Access

Section Info Here

4.7.1. IP Addresses for Pods and Services

4.7.2. Service Types

4.7.3. Using Routes for External Connectivity

4.7.4. Using Ingress Objects for External Connectivity

4.7.5. Sticky sessions

4.7.6. Load Balance and Scale Applications

4.7.6.1. Load Balance Pods

4.8. DEMO: Scale and Expose Applications to External Access

Credentials

UN/PW: **developer/developer**

or

UN/PW: **admin/redhatocp**

- **Web Console:** <https://console-openshift-console.apps.ocp4.example.com>
- **OpenShift API:** <https://api.ocp4.example.com:6443>

Listing 43. Developer Login



```
oc login -u developer -p developer https://api.ocp4.example.com:6443
```

Listing 44. Administrator Login

```
oc login -u admin -p redhatocp https://api.ocp4.example.com:6443
```

Example 12. DEMO: Scaling and Exposing Applications to External Access using OCP Routes

1. Ensure lab environment is ready and login as the **developer** user.

```
[student@workstation ~]$ lab start deploy-newapp  
[student@workstation ~]$ oc login -u developer -p developer  
https://api.ocp4.example.com:6443
```

2. Create Demo project

```
[student@workstation ~]$ oc new-project demo-routes
```

3. Create HTTPD (Apache) Deployment with a Service exposed on port 8080

Listing 45. Creating Deployment with Service to Expose

```
[student@workstation ~]$ oc create deployment demo-service-web --port 8080 --image  
registry.ocp4.example.com:8443/redhattraining/do180-httpd-app:v1
```

Listing 46. Exposing the Service

```
[student@workstation ~]$ oc expose deployment demo-service-web  
service/demo-service-web exposed
```

4. Get Services and Endpoints

```
[student@workstation ~]$ oc get services  
NAME          TYPE      CLUSTER-IP      EXTERNAL-IP    PORT(S)      AGE  
demo-service-web   ClusterIP  172.30.162.76  <none>        8080/TCP    11s
```

```
[student@workstation ~]$ oc get endpoints  
NAME           ENDPOINTS      AGE  
demo-service-web  10.8.0.79:8080  30s
```

5. Expose Route to the Service and Get the Route

```
[student@workstation ~]$ oc expose service demo-service-web  
route.route.openshift.io/demo-service-web exposed
```

```
[student@workstation ~]$ oc get route  
NAME          HOST/PORT      PATH  
SERVICES      PORT  TERMINATION  WILDCARD  
demo-service-web  demo-service-web-demo-routes.apps.ocp4.example.com  
demo-service-web  8080           None
```

6. View Website

```
[student@workstation ~]$ curl demo-service-web-demo-routes.apps.ocp4.example.com  
Welcome to Red Hat Training, from demo-service-web-6694f6d566-26dj8
```

7. Scale out Application

```
[student@workstation ~]$ oc scale deployment demo-service-web --replicas 2  
deployment.apps/demo-service-web scaled
```

8. Verify additional pods

```
[student@workstation ~]$ oc get pods -o wide
```

NAME	NODE	NOMINATED NODE	READY	STATUS	RESTARTS	AGE	IP
			READINESS	GATES			
demo-service-web-6694f6d566-26dj8	master01	<none>	1/1	Running	0	6m1s	10.8.0.79
demo-service-web-6694f6d566-6hkx6	master01	<none>	1/1	Running	0	22s	10.8.0.80

9. Use the **Curl** command on the routes to show it going to both systems

```
[student@workstation ~]$ curl demo-service-web-demo-routes.apps.ocp4.example.com
Welcome to Red Hat Training, from demo-service-web-6694f6d566-6hkx6 ①
[student@workstation ~]$ curl demo-service-web-demo-routes.apps.ocp4.example.com
Welcome to Red Hat Training, from demo-service-web-6694f6d566-26dj8 ②
```

① Shows it going to pod **6hkx6**

② Shows it going to pod **26dj8**

OCP Load Balancer



The OCP Load Balancer will send the traffic to all pods connected to the same route with the same service. It might be necessary to repeat the curl command a few times to show the load-balancer at work. It is possible to use things like cookies and other work-arounds to preserve or create session "stickyness"

10. Clean Up Environment

```
[student@workstation ~]$ oc delete project demo-routes
project.project.openshift.io "demo-routes" deleted
```

Chapter 5. Manage Storage for Application Configuration and Data

5.1. Externalize the Configuration of Applications

Section Info Here

5.1.1. Configuring Kubernetes Applications

5.1.2. Creating Secrets and Configuration Maps

5.1.3. Using Configuration Maps and Secrets to Initialize Environment Variables

5.1.4. Using Secrets and Configuration Maps as Volumes

5.1.5. Updating Secrets and Configuration Maps

5.1.6. Deleting Secrets and Configuration Maps

5.2. DEMO: Externalize the Configuration of Applications

Credentials

UN/PW: **developer/developer**

or

UN/PW: **admin/redhatocp**

- **Web Console:** <https://console-openshift-console.apps.ocp4.example.com>
- **OpenShift API:** <https://api.ocp4.example.com:6443>

Listing 47. Developer Login



```
oc login -u developer -p developer https://api.ocp4.example.com:6443
```

Listing 48. Administrator Login

```
oc login -u admin -p redhatocp https://api.ocp4.example.com:6443
```

Example 13. DEMO: Externally Configuring Applications

1. Ensure lab environment is ready and login as the **developer** user.

```
[student@workstation ~]$ lab start storage-configs  
[student@workstation ~]$ oc login -u developer -p developer  
https://api.ocp4.example.com:6443
```

2. Create Demo project

```
[student@workstation ~]$ oc new-project demo-storage-configs
```

3. Create a deployment with a service and a route exposed.

```
[student@workstation ~]$ oc create deployment demo-service-web --port 8080 --image  
registry.ocp4.example.com:8443/redhattraining/do180-httpd-app:v1
```

Listing 49. Expose Service

```
[student@workstation ~]$ oc expose deployment demo-service-web
service/demo-service-web exposed
```

Listing 50. Expose the Route

```
[student@workstation ~]$ oc expose service demo-service-web
route.route.openshift.io/demo-service-web exposed
```

4. Get Route and **curl** the website

[student@workstation ~]\$ oc get route						
NAME	HOST/PORT	PATH	SERVICES	PORT	TERMINATION	WILDCARD
demo-service-web	demo-service-web-demo-storage-configs.apps.ocp4.example.com	demo-service-web	demo-service-web	8080	None	

```
[student@workstation ~]$ curl demo-service-web-demo-storage-
configs.apps.ocp4.example.com
Welcome to Red Hat Training, from demo-service-web-6694f6d566-kk2m8
[student@workstation ~]$
```

5. Clone down repository for website soure.

```
[student@workstation ~]$ git clone https://github.com/tmichett/OCP_Demos.git
```

6. Create ConfigMaps for the website file (index.html) and the required image (penguin3.jpg).

Listing 51. Creating Directory Config Map

```
[student@workstation ~]$ oc create configmap website --from
-file=/home/student/OCP_Demos/D0188_Web
configmap/website created
```

7. Apply ConfigMaps to the deployment

Listing 52. Creating Volume for ConfigMap for Website Directory

```
[student@workstation ~]$ oc set volume deployment/demo-service-web --add --type
configmap --configmap-name website --mount-path /var/www/html/
info: Generated volume name: volume-ksv8p
```

8. Ensure new pod(s) are ready

```
[student@workstation ~]$ oc get pods
NAME                      READY   STATUS      RESTARTS   AGE
demo-service-web-5558d7cd7d-tvjqk   1/1     Terminating   0          91s
demo-service-web-6c9c56d74-jrs2p   1/1     Running     0          17s
```

9. Perform the **curl** command again and view in Firefox

```
[student@workstation ~]$ curl demo-service-web-demo-storage-
configs.apps.ocp4.example.com
... OUTPUT OMITTED ...
```

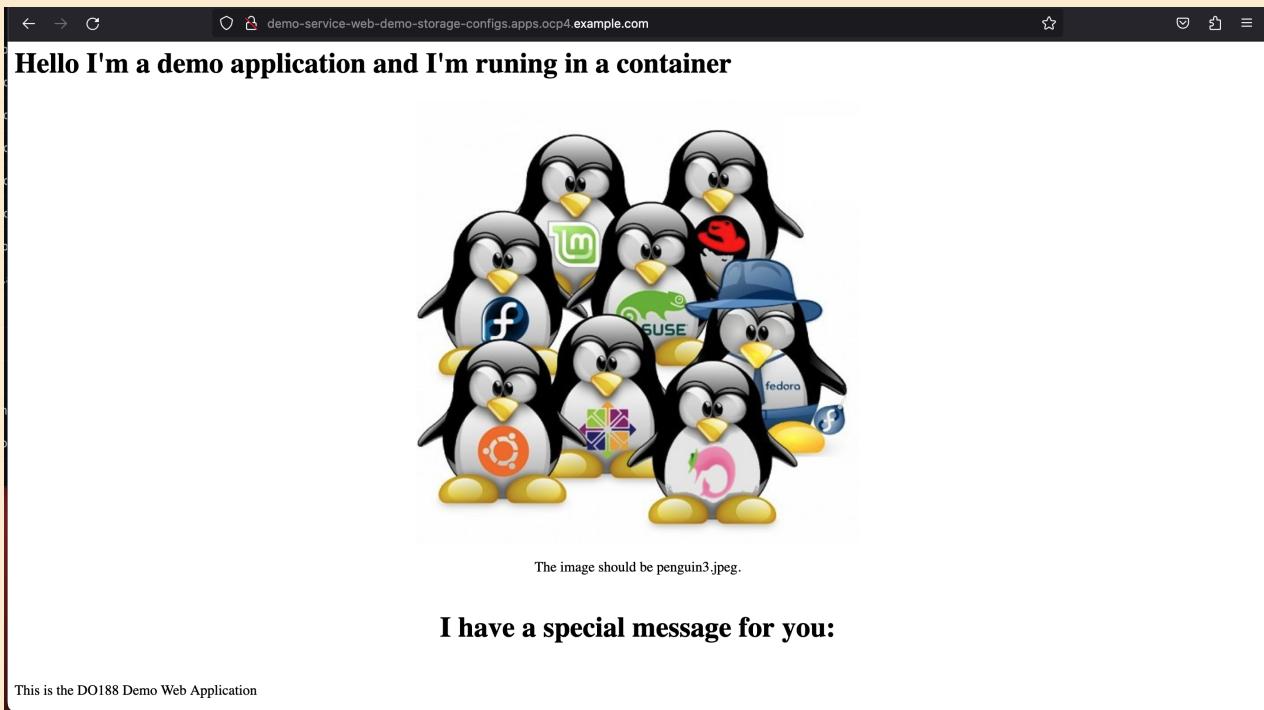


Figure 8. Website from Container ConfigMaps

10. Clean up environment

```
[student@workstation ~]$ oc delete project demo-storage-configs
project.project.openshift.io "demo-storage-configs" deleted
```

5.3. Provision Persistent Data Volumes

Section Info Here

5.3.1. Kubernetes Persistent Storage

5.3.2. Persistent Volumes

5.3.2.1. Volume Access Mode

5.3.2.2. Volume Modes

5.3.2.3. Manually Creating a PV

5.3.3. Persistent Volume Claims

5.3.3.1. Creating a PVC

5.3.3.2. Kubernetes Dynamic Provisioning

5.3.3.3. PV and PVC lifecycles

5.3.3.4. Deleting a Persistent Volume Claim

5.4. DEMO: Provision Persistent Data Volumes

Credentials

UN/PW: developer/developer

or

UN/PW: admin/redhatocp

- **Web Console:** <https://console-openshift-console.apps.ocp4.example.com>
- **OpenShift API:** <https://api.ocp4.example.com:6443>

Listing 53. Developer Login



```
oc login -u developer -p developer https://api.ocp4.example.com:6443
```

Listing 54. Administrator Login

```
oc login -u admin -p redhatocp https://api.ocp4.example.com:6443
```

Example 14. DEMO: Provisioning a Persistent Data Volume

1. Ensure lab environment is ready and login as the **developer** user.

```
[student@workstation ~]$ lab start storage-volumes  
[student@workstation ~]$ oc login -u developer -p developer  
https://api.ocp4.example.com:6443
```

2. Create Demo project

```
[student@workstation ~]$ oc new-project demo-storage-pvs
```

3. Determine available storage classes

```
[student@workstation ~]$ oc get storageclass  
NAME          PROVISIONER  
RECLAIMPOLICY VOLUMEBINDINGMODE      ALLOWVOLUMEEXPANSION AGE  
lvms-vg1       topolv�.io           Delete  
WaitForFirstConsumer true            24d  
nfs-storage (default)   k8s-sigs.io/nfs-subdir-external-provisioner Delete
```

Immediate	false	24d
-----------	-------	-----

4. Create a deployment, service, and route

```
[student@workstation ~]$ oc create deployment demo-service-web --port 8080 --image registry.ocp4.example.com:redhattraining/do180-httpd-app:v1
```

Listing 55. Expose Service

```
[student@workstation ~]$ oc expose deployment demo-service-web  
service/demo-service-web exposed
```

Listing 56. Expose the Route

```
[student@workstation ~]$ oc expose service demo-service-web  
route.route.openshift.io/demo-service-web exposed
```

5. Get route and verify application

```
[student@workstation ~]$ oc get route  
NAME          HOST/PORT  
PATH  SERVICES      PORT  TERMINATION   WILDCARD  
demo-service-web  demo-service-web-demo-storage-configs.apps.ocp4.example.com  
demo-service-web    8080           None
```

```
[student@workstation ~]$ curl demo-service-web-demo-storage-  
configs.apps.ocp4.example.com  
Welcome to Red Hat Training, from demo-service-web-6694f6d566-kk2m8  
[student@workstation ~]$
```

6. Create a persistent volume claim and apply it to the deployment

```
[student@workstation ~]$ oc set volumes deployment/demo-service-web --add --name  
nfs-volume-storage --type pvc --claim-mode rwo --claim-size 1Gi --mount-path  
/var/www/html --claim-name demo-service-web-pvc
```

7. Verify that PVC was created and bound

```
[student@workstation ~]$ oc get pvc  
NAME          STATUS  VOLUME
```

```
CAPACITY ACCESS MODES STORAGECLASS AGE
demo-service-web-pvc Bound pvc-1a09dd19-d243-434f-824b-97c6aebf0c5b 1Gi
RWO nfs-storage 48s
```

8. Verify the deployment

```
[student@workstation ~]$ oc get deployment/demo-service-web -o yaml
apiVersion: apps/v1
kind: Deployment
metadata:
...
volumeMounts:
- mountPath: /var/www/html
  name: nfs-volume-storage ①
...
volumes:
- name: nfs-volume-storage
  persistentVolumeClaim:
    claimName: demo-service-web-pvc ②
...

```

① Persistent Volume Name

② Name of Persistent Volume Claim (PVC)

9. Copy files to persistent volume

```
[student@workstation ~]$ oc cp /home/student/OCP_Demos/D0188_Web/index.html demo-
service-web-555dbb644f-xjsnv:/var/www/html/
[student@workstation ~]$ oc cp /home/student/OCP_Demos/D0188_Web/penguin3.jpeg
demo-service-web-555dbb644f-xjsnv:/var/www/html/
```

10. Check web service

```
[student@workstation ~]$ curl demo-service-web-demo-storage-
pvs.apps.ocp4.example.com
```

11. Remove the volume from the deployment

```
[student@workstation ~]$ oc set volumes deployment/demo-service-web --remove  
--name nfs-volume-storage
```

12. Perform curl command on deployment

```
[student@workstation ~]$ curl demo-service-web-demo-storage-  
pvs.apps.ocp4.example.com  
Welcome to Red Hat Training, from demo-service-web-6694f6d566-dxv6w ①
```

① Persistent volume was removed and container returned to original state

13. Verify that PVC exists

```
[student@workstation ~]$ oc get pvc  
NAME           STATUS  VOLUME  
CAPACITY      ACCESS MODES  STORAGECLASS  AGE  
demo-service-web-pvc  Bound    pvc-1a09dd19-d243-434f-824b-97c6aebf0c5b  1Gi  
RWO            nfs-storage  9m32s
```

14. Create a new deployment, service, and route

```
[student@workstation ~]$ oc create deployment demo-service-web2 --port 8080  
--image registry.ocp4.example.com:8443/redhattraining/do180-nginx-app:v1
```

Listing 57. Expose Service

```
[student@workstation ~]$ oc expose deployment demo-service-web2  
service/demo-service-web exposed
```

Listing 58. Expose the Route

```
[student@workstation ~]$ oc expose service demo-service-web2  
route.route.openshift.io/demo-service-web exposed
```

15. Retrieve route and verify content

```
[student@workstation ~]$ oc get route  
NAME          HOST/PORT  
PATH  SERVICES      PORT  TERMINATION  WILDCARD  
demo-service-web  demo-service-web-demo-storage-pvs.apps.ocp4.example.com  
demo-service-web  8080        None
```

demo-service-web2	demo-service-web2-demo-storage-pvs.apps.ocp4.example.com
demo-service-web2	8080
	None

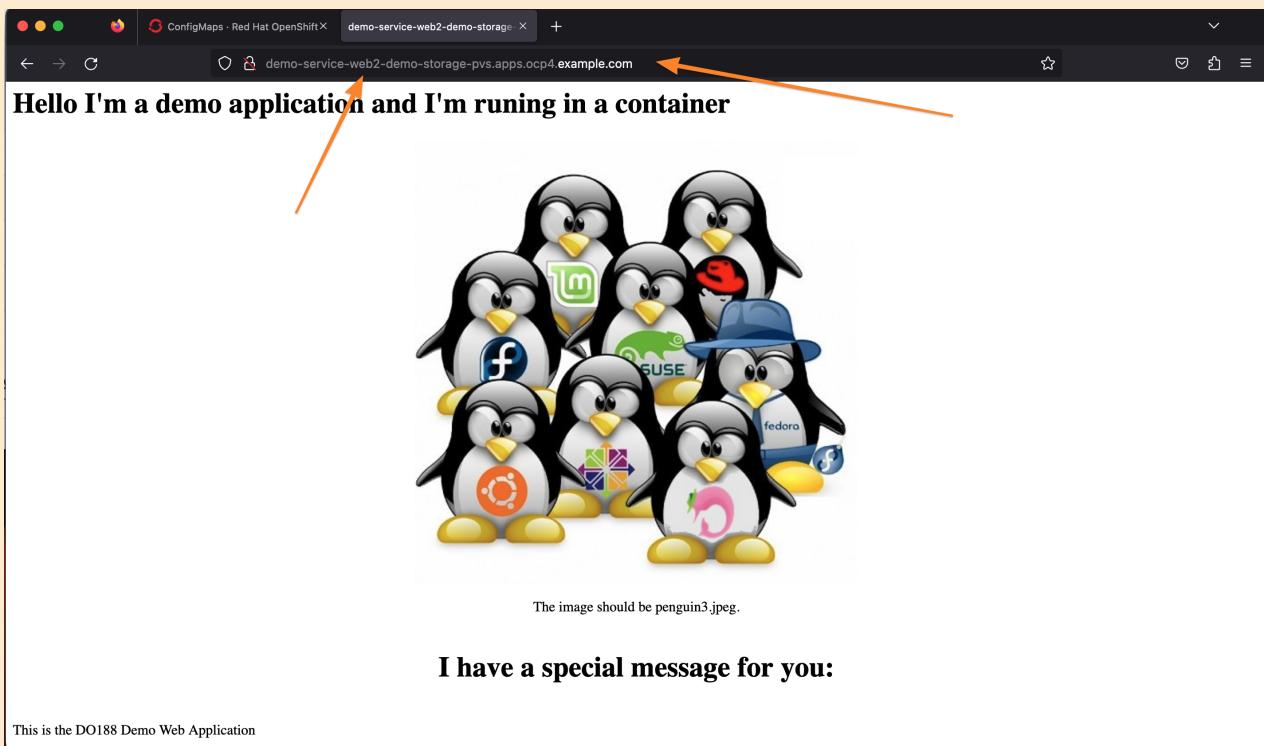
```
[student@workstation ~]$ curl demo-service-web2-demo-storage-pvs.apps.ocp4.example.com  
Welcome to Red Hat Training, from demo-service-web2-69ff6b796-hrgmb
```

16. Attach persistent storage volume to new deployment

```
[student@workstation ~]$ oc set volumes deployment/demo-service-web2 --add --type pvc --mount-path /var/www/html --claim-name demo-service-web-pvc
```

17. Verify the volume has been mounted by looking at website content

```
[student@workstation ~]$ curl demo-service-web2-demo-storage-pvs.apps.ocp4.example.com
```



18. Clean up environment

```
[student@workstation ~]$ oc delete project demo-storage-pvs  
project.project.openshift.io "demo-storage-pvs" deleted
```

5.5. Select a Storage Class for an Application

Section Info Here

5.5.1. Storage Class Selection

5.5.1.1. Reclaim Policy

5.5.1.2. Kubernetes and Application Responsibilities

5.5.2. Use Cases for Storage Classes

5.5.3. Create a Storage Class

5.5.3.1. Cluster Storage Classes

5.5.4. Storage Class Usage

5.6. DEMO: Select a Storage Class for an Application

Credentials

UN/PW: developer/developer

or

UN/PW: admin/redhatocp

- **Web Console:** <https://console-openshift-console.apps.ocp4.example.com>
- **OpenShift API:** <https://api.ocp4.example.com:6443>

Listing 59. Developer Login



```
oc login -u developer -p developer https://api.ocp4.example.com:6443
```

Listing 60. Administrator Login

```
oc login -u admin -p redhatocp https://api.ocp4.example.com:6443
```

Example 15. DEMO: Selecting a Storage Class for an Application



This demo is meant to show different storage classes and selecting something other than the NFS default storage class. It also shows how to create an OCP Custom Resource Definition (CRD) for a persistent volume claim (PVC) and attach the resource to a deployment.

1. Ensure lab environment is ready and login as the **developer** user.

```
[student@workstation ~]$ lab start storage-classes  
  
[student@workstation ~]$ oc login -u developer -p developer  
https://api.ocp4.example.com:6443
```

2. Create Demo project

```
[student@workstation ~]$ oc new-project demo-storage-classes
```

3. Retrieve available storage classes

```
[student@workstation ~]$ oc get sc
```

NAME	PROVISIONER	RECLAIMPOLICY	VOLUMEBINDINGMODE	ALLOWVOLUMEEXPANSION	AGE	
lvms-vg1	topolvm.io	WaitForFirstConsumer	true	24d		Delete
nfs-storage (default)	k8s-sigs.io/nfs-subdir-external-provisioner	Immediate	false	24d		Delete

4. Retrieve information about the LVMS Storage class

```
[student@workstation ~]$ oc describe sc lvms-vg1
Name:           lvms-vg1
IsDefaultClass: No
Annotations:    description=Provides RWO and RWOP Filesystem & Block
                  volumes
Provisioner:    topolvm.io
Parameters:     csi.storage.k8s.io/fstype=xfs,topolvm.io/device-class=vg1
AllowVolumeExpansion: True
MountOptions:   <none>
ReclaimPolicy: Delete
VolumeBindingMode: WaitForFirstConsumer
Events:         <none>
```

5. Create a PVC Resource File (vim lvms-pvc.yml)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: lvms-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: lvms-vg1
```

6. Create the PVC resource from the CRD file

```
[student@workstation ~]$ oc create -f lvms-pvc.yml
persistentvolumeclaim/lvms-pvc created
```

7. Analyze the PVC that was created

```
[student@workstation ~]$ oc describe pvc lvms-pvc
Name:          lvms-pvc
Namespace:     demo-storage-classes
StorageClass:  lvms-vg1
Status:        Pending
Volume:
Labels:        <none>
Annotations:   <none>
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:
Access Modes:
VolumeMode:    Filesystem
Used By:       <none>
Events:
  Type      Reason           Age           From
Message
  ----      -----           ----         -----
  Normal   WaitForFirstConsumer  3s (x3 over 32s)  persistentvolume-controller
  waiting for first consumer to be created before binding
```

8. Create deployment, attach the PVC, and open for web

```
[student@workstation ~]$ oc create deployment demo-service-web --port 8080 --image
registry.ocp4.example.com:8443/redhattraining/do180-httpd-app:v1
```

Listing 61. Expose Service

```
[student@workstation ~]$ oc expose deployment demo-service-web
service/demo-service-web exposed
```

Listing 62. Expose the Route

```
[student@workstation ~]$ oc expose service demo-service-web
route.route.openshift.io/demo-service-web exposed
```

Listing 63. Add the PV using PVC Claim Created Earlier

```
[student@workstation ~]$ oc set volumes deployment/demo-service-web --add --name
lvms-volume --claim-name lvms-pvc --mount-path /var/www/html
```

9. Add data to the PVC

```
[student@workstation ~]$ oc cp /home/student/OCP_Demos/D0188_Web/index.html demo-service-web-555dbb644f-xjsnv:/var/www/html/  
  
[student@workstation ~]$ oc cp /home/student/OCP_Demos/D0188_Web/penguin3.jpeg demo-service-web-555dbb644f-xjsnv:/var/www/html/
```

10. Check data from the PVC exists in the application

```
[student@workstation ~]$ curl demo-service-web-demo-storage-classes.apps.ocp4.example.com
```

11. Clean up environment

```
[student@workstation ~]$ oc delete project demo-storage-classes  
project.project.openshift.io "demo-storage-classes" deleted
```

5.7. Manage non-Shared Storage with StatefulSets

Section Info Here

5.7.1. Application Clustering

5.7.2. Storage Services

5.7.3. Introduction to Stateful Sets

5.7.4. Working with Stateful Sets

5.8. DEMO: Manage non-Shared Storage with Stateful Sets

Credentials

UN/PW: developer/developer

or

UN/PW: admin/redhatocp

- **Web Console:** <https://console-openshift-console.apps.ocp4.example.com>
- **OpenShift API:** <https://api.ocp4.example.com:6443>

Listing 64. Developer Login



```
oc login -u developer -p developer https://api.ocp4.example.com:6443
```

Listing 65. Administrator Login

```
oc login -u admin -p redhatocp https://api.ocp4.example.com:6443
```

Example 16. DEMO: Manage non-Shared Storage with Stateful Sets

1. Ensure lab environment is ready and login as the **developer** user.

```
[student@workstation ~]$ lab start storage-statefulsets  
[student@workstation ~]$ oc login -u developer -p developer  
https://api.ocp4.example.com:6443
```

2. Create Demo project

```
[student@workstation ~]$ oc new-project demo-statefulsets
```

3. Change into the Github **OCP_Demo** Project

```
[student@workstation ~]$ cd /home/student/OCP_Demos/Stateful_Sets  
[student@workstation Stateful_Sets]$
```

4. Create the OCP Resources from the CRD File

Listing 66. Examine Resource File

```
[student@workstation Stateful_Sets]$ cat statefulset-demo.yml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: webserver
spec:
  selector:
    matchLabels:
      app: webserver
  replicas: 2
  template:
    metadata:
      labels:
        app: webserver
  spec:
    terminationGracePeriodSeconds: 10
    containers:
      - name: webserver
        image: registry.ocp4.example.com:8443/redhattraining/do180-httpd-app:v1
        ports:
          - name: webport
            containerPort: 8080
        volumeMounts:
          - name: webdata
            mountPath: /var/www/html
    volumeClaimTemplates:
      - metadata:
          name: webdata
        spec:
          accessModes: ["ReadWriteOnce"]
          storageClassName: "lvms-vg1"
        resources:
          requests:
            storage: 1Gi
```

Listing 67. Creating Systems

```
[student@workstation Stateful_Sets]$ oc create -f statefulset-demo.yml
statefulset.apps/webserver created
```

5. Verify Pods and PVCs

```
[student@workstation Stateful_Sets]$ oc get pods
NAME        READY     STATUS    RESTARTS   AGE

```

webserver-0	1/1	Running	0	6m8s
webserver-1	1/1	Running	0	6m6s

```
[student@workstation Stateful_Sets]$ oc get pvc
NAME          STATUS  VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS AGE
webdata-webserver-0  Bound   pvc-237b855e-82da-4067-8311-c5dcc0fa2889  1Gi
RWO          lvms-vg1    6m36s
webdata-webserver-1  Bound   pvc-20cc7b66-7d68-4f9e-beec-e912846b9b9a  1Gi
RWO          lvms-vg1    6m34s
```

6. Run script to copy data, expose services, and expose routes

```
[student@workstation Stateful_Sets]$ ./create_web.sh
```

7. List the created routes

```
[student@workstation Stateful_Sets]$ oc get routes
NAME      HOST/PORT                                     PATH
SERVICES PORT  TERMINATION  WILDCARD
webserver-0 webserver-0-demo-statefulsets.apps.ocp4.example.com
webserver-0 8080           None
webserver-1 webserver-1-demo-statefulsets.apps.ocp4.example.com
webserver-1 8080           None
```

8. Verify websites are different

```
[student@workstation Stateful_Sets]$ curl webserver-0-demo-
statefulsets.apps.ocp4.example.com
Hello from the stateful set - This is Webserver-0 and my hostname is webserver-0

[student@workstation Stateful_Sets]$ curl webserver-1-demo-
statefulsets.apps.ocp4.example.com
Hello from the stateful set - This is Webserver-1 and my hostname is webserver-1
```

9. Clean up environment

```
[student@workstation Stateful_Sets]$ oc delete project demo-statefulsets
project.project.openshift.io "demo-statefulsets" deleted
```

Chapter 6. Configure Applications for Reliability

6.1. Application High Availability with Kubernetes

Section Info Here

6.1.1. Concepts of Deploying Highly Available Applications

6.1.2. Writing Reliable Applications

6.1.3. Kubernetes Application Reliability

6.2. DEMO: Externalize the Configuration of Applications

Credentials

UN/PW: developer/developer

or

UN/PW: admin/redhatocp

- **Web Console:** <https://console-openshift-console.apps.ocp4.example.com>
- **OpenShift API:** <https://api.ocp4.example.com:6443>

Listing 68. Developer Login



```
oc login -u developer -p developer https://api.ocp4.example.com:6443
```

Listing 69. Administrator Login

```
oc login -u admin -p redhatocp https://api.ocp4.example.com:6443
```

Example 17. DEMO: Externally Configuring Applications

Listing 70. Ensure Lab Environment ready to begin

```
[student@workstation ~]$ lab start reliability-ha  
[student@workstation ~]$ oc login -u developer -p developer  
https://api.ocp4.example.com:6443
```

Listing 71. Switch to Project

```
[student@workstation ~]$ oc project reliability-ha
```

Listing 72. Open Directory from OCP_Demos Repository

```
[student@workstation ~]$ cd /home/student/OCP_Demos/D0180v412/CH6
```

Listing 73. Apply resource from file to build load pods

```
[student@workstation CH6]$ oc apply -f load_demo.yml
```

Listing 74. Observe Pods

```
[student@workstation CH6]$ oc get pods -o wide
```

Listing 75. Verify the health endpoint

```
[student@workstation CH6]$ oc exec load-demo -- curl -s localhost:3000/health
```

Listing 76. Force a Crash within Container and Pod

```
[student@workstation CH6]$ oc exec load-demo -- curl -s localhost:3000/destruct  
command terminated with exit code 52
```

Listing 77. Verify that App Recovered with New Pod

```
[student@workstation CH6]$ oc get pods  
NAME      READY   STATUS    RESTARTS   AGE  
load-demo  1/1     Running   1 (51s ago)  2m53s ①
```

① Pod restarted

Listing 78. Cause Failure

```
[student@workstation CH6]$ oc exec load-demo-fail -- curl -s localhost:3000/destruct  
command terminated with exit code 52
```

Listing 79. Verify Pod status of Error

```
[student@workstation CH6]$ oc get pods  
NAME      READY   STATUS    RESTARTS   AGE  
load-demo-fail  0/1     Error    0          91s
```

Listing 80. Cleanup Resources and Project

```
[student@workstation CH6]$ oc delete project reliability-ha
```

6.3. Application Health Probes

Section Info Here

6.3.1. Kubernetes Probes

6.3.2. Authoring Probe Endpoints

6.3.3. Probe Types

6.3.3.1. Readiness Probes

6.3.3.2. Liveness Probes

6.3.3.3. Startup Probes

6.3.4. Types of Tests

6.3.5. Timings and Thresholds

6.3.6. Adding Probes via YAML

6.3.7. Adding Probes via the CLI

6.4. Reserve Compute Capacity for Applications

Section Info Here

6.4.1. Kubernetes Pod Scheduling

6.4.2. Compute Resource Requests

6.4.3. Inspecting Cluster Compute Resources

6.5. DEMO: Reserve Compute Capacity for Applications

Credentials

UN/PW: developer/developer

or

UN/PW: admin/redhatocp

- **Web Console:** <https://console-openshift-console.apps.ocp4.example.com>
- **OpenShift API:** <https://api.ocp4.example.com:6443>

Listing 81. Developer Login



```
oc login -u developer -p developer https://api.ocp4.example.com:6443
```

Listing 82. Administrator Login

```
oc login -u admin -p redhatocp https://api.ocp4.example.com:6443
```

Example 18. DEMO: Externally Configuring Applications

Listing 83. Check Resources

```
[student@workstation CH6]$ lab start reliability-requests
```

Listing 84. Login as the Admin User

```
[student@workstation CH6]$ oc login https://api.ocp4.example.com:6443 -u admin
```

Listing 85. Apply OpenShift CRD for the demo-load-deploy

```
[student@workstation CH6]$ oc apply -f demo-load-deploy.yml
deployment.apps/demo-load-limits created
service/demo-load-limits created
route.route.openshift.io/demo-load-limits created
```

Listing 86. Verify deployed pods

```
[student@workstation CH6]$ oc get pods
NAME                  READY   STATUS    RESTARTS   AGE
demo-load-limits-6df9b97cd4-jjcnz   1/1     Running   0          12s
demo-load-limits-6df9b97cd4-k646h   1/1     Running   0          16s
```

```
demo-load-limits-6df9b97cd4-m8gcw    1/1      Running   0          20s
```

Listing 87. Scale to 10 Pods

```
[student@workstation CH6]$ oc scale deploy/demo-load-limits --replicas 10  
deployment.apps/demo-load-limits scaled
```

Listing 88. Verify not all pods starting

```
[student@workstation CH6]$ oc get pods  
NAME                      READY   STATUS    RESTARTS   AGE  
demo-load-limits-645b4fd54c-2b78v   1/1     Running   0          9s  
demo-load-limits-645b4fd54c-5l6rx   0/1     Pending   0          3s
```

Listing 89. View Failed Events

```
[student@workstation CH6]$ oc get events --field-selector reason="FailedScheduling"  
... OUTPUT OMITTED ...  
  
78s      Warning  FailedScheduling  pod/demo-load-limits-645b4fd54c-vt5kt  0/1  
nodes are available: 1 Insufficient memory.  preemption: 0/1 nodes are available: 1 No  
preemption victims found for incoming pod.  
78s      Warning  FailedScheduling  pod/demo-load-limits-645b4fd54c-zhl97  0/1  
nodes are available: 1 Insufficient memory.  preemption: 0/1 nodes are available: 1 No  
preemption victims found for incoming pod. ①
```

① Not enough memory on available compute nodes for scheduling

Listing 90. Update Resource Requests Manually to Decrease Memory

```
[student@workstation CH6]$ oc set resources deploy/demo-load-limits --requests  
memory=250Mi  
deployment.apps/demo-load-limits resource requirements updated
```

Listing 91. Remove project and all resources

```
[student@workstation CH6]$ lab finish reliability-requests
```

6.6. Limit Compute Capacity for Applications

Section Info Here

6.6.1. Setting Memory Limits

6.6.2. Setting CPU Limits

6.6.3. Viewing Requests, Limits, and Actual Usage

6.7. Application Autoscaling

Section Info Here

Chapter 7. Manage Application Updates

7.1. Container Image Identity and Tags

Section Info Here

7.1.1. Kubernetes Image Tags

7.1.1.1. Floating Tag Issues

7.1.1.2. Using SHA Image ID

7.1.2. Selecting a Pull Policy

7.1.3. Pruning Images from Cluster Nodes

7.2. Update Application Image and Settings

Section Info Here

7.2.1. Application Code, Configuration, and Data

7.2.2. Deployment Strategies

7.2.2.1. Rolling Update Strategy

7.2.2.2. Recreate Strategy

7.2.3. Rolling Out Applications

7.2.3.1. Monitoring Replica Sets

7.2.3.2. Managing Rollout

7.3. Reproducible Deployments with OpenShift Image Streams

Section Info Here

7.3.1. Image Streams

7.3.1.1. Image Stream Tags

7.3.1.2. Image Names, Tags, and IDs

7.3.2. Creating Image Streams and Tags

7.3.2.1. Importing Image Stream Tags Periodically

7.3.2.2. Configuring Image Pull-through

7.3.3. Using Image Streams in Deployments

7.3.3.1. Enabling the Local Lookup Policy

7.3.3.2. Configuring Image Streams in Deployments

7.4. Automatic Image Updates with OpenShift Image Change Triggers

Section Info Here

7.4.1. Using Triggers to Manage Images

7.4.2. Configuring Image Trigger for Deployments

7.4.2.1. Rolling Out Deployments

7.4.2.2. Rolling Back Deployments

7.4.3. Managing Image Stream Tags