

# Lesson 2: Manage Long-lived and Short-lived Applications by Using the Kubernetes Workload API

Deploy containerized applications as pods that Kubernetes workload resources manage.

# Kubernetes Workload Resources

- Simplify deploying and managing applications
- Example Workloads API:
  - a) Jobs
  - b) Deployments
  - c) Deployment Config
  - d) Pods
  - e) Replication Controllers
  - f) Stateful sets

# Jobs

- One-time task to perform on the cluster.
- Executed via pods.
- If a job's pod fails, then the cluster retries a number of times that the job specifies.
- Differ from using the kubectl run and oc run commands; both only used to create pod(s).
- Case Uses:
  - a) Initializing or migrating a database
  - b) Calculating one-off metrics from information within the cluster
  - c) Creating or restoring from a data backup

# Jobs - create

The following example command creates a job that logs the date and time:

```
[user@host ~]$ oc create job \ ①  
date-job \ ②  
--image registry.access.redhat.com/ubi8/ubi \ ③  
-- /bin/bash -c "date" ④
```

- ① Creates a job resource.
- ② Specifies a job name of date-job.
- ③ Sets registry.access.redhat.com/ubi8/ubi as the container image for the job pods.
- ④ Specifies the command to run within the pods.

Alternatively use web console

Steps: Workloads → Jobs → Create Job

# Cron Jobs

- Regular job resource (recurring tasks)
  - Specify how often the job should run
- Similar to crontab on Linux
- Uses cron format for scheduling
- Created on control plane nodes
- Case Uses:
  - a) Backups
  - b) Report generation

# Jobs Jobs - create

```
[user@host ~]$ oc create cronjob date-cronjob \ ①
--image registry.access.redhat.com/ubi8/ubi \ ②
--schedule "*/1 * * * *" \ ③
-- date ④
```

- ① Creates a cron job resource with a name of date-cronjob.
- ② Sets the registry.access.redhat.com/ubi8/ubi as the container image for the job pods.
- ③ Specifies the schedule for the job in Cron format.
- ④ The command to execute within the pods.

- Alternatively use web console

Steps: Workloads → Cron Jobs → Create CronJob

# Deployments

- Creates and manages replica set to maintain multiple pods
  - add or remove pods specified by deployment specifications
- Replica sets use selectors, such as label
  - to identify pods part of set
- Recreated in even of pods failure or accidentally deletion
- Jobs - on time execution
- Deployment - continuously monitor and maintain applications

# Deployments

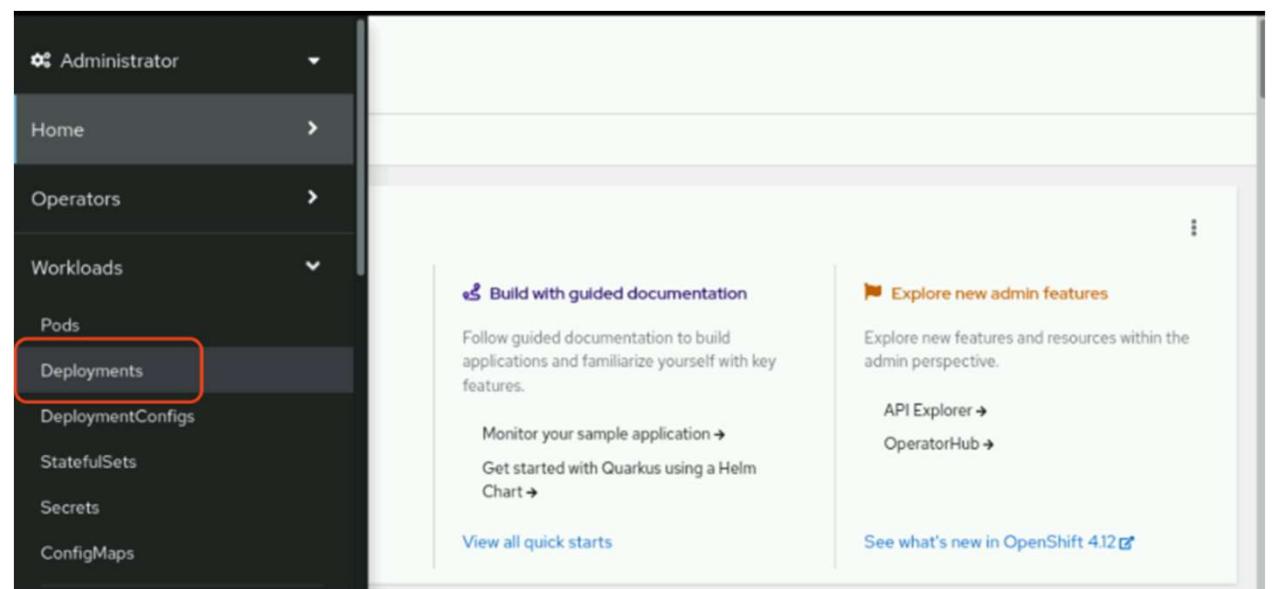
## - create

---

```
[user@host ~]$ oc create deployment \ ①  
my-deployment \ ②  
--image registry.access.redhat.com/ubi8/ubi \ ③  
--replicas 3 ④
```

- ① Creates a deployment resource.
- ② Specifies my-deployment as the deployment name.
- ③ Sets registry.access.redhat.com/ubi8/ubi as the container image for the pods.
- ④ Sets the deployment to maintain three instances of the pod.

# Deployments - create



# Deployments

## - create

The screenshot shows a 'Create Deployment' form. At the top, there are dropdown menus for 'Project: metallb-system' and 'Application: All applications'. Below that, the title 'Create Deployment' is centered. Underneath the title, there's a note: 'Configure via:  Form view  YAML view'. A blue info box contains the text: 'Note: Some fields may not be represented in this form view. Please select "YAML view" for full control.' To the right of the info box is a small 'X' icon. The next section is titled 'Deployment strategy' with a 'Strategy type' dropdown menu. At the bottom of the form are two buttons: 'Create' (gray) and 'Cancel' (blue).

# Resource Labels

```
apiVersion: apps/v1
kind: ReplicaSet
...output omitted...
spec:
  replicas: 1
  selector:
    matchLabels:
      app: httpd
      pod-template-hash: 7c84fbdb57
...output omitted...
```

- Metadata represented as string key-value pairs
- Common traits for resources
- Example of using label  
\$ oc get all -l app=httpd
- \$ oc delete all -l app=httpd

# Stateful Sets

- Pods require state within cluster
- Stateful Sets manages stateful pods
- Deployments are stateless pods
- Each pod is Stateful sets created uniquely
- Case Uses:
  - a) Pod needs unique network identifier
  - b) or needs persistent storage
- Discuss further in later chapter

# Guided Exercise: Manage Long-lived and Short-lived Applications by Using the Kubernetes Workload API

In this exercise, you deploy a database server and a batch application that are both managed by workload resources:

- Create deployments.
- Update environment variables on a pod template.
- Create and run job resources.
- Retrieve the logs and termination status of a job.
- View the pod template of a job resource.

## In this chapter, you learned:



The OpenShift web console provides a GUI for visualizing and managing OpenShift resources.



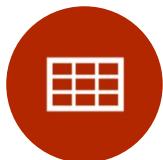
Some resources feature a specialized page that makes creating and editing resources more convenient than writing YAML by hand, such as the Edit Key/Value Secret editor, which automatically handles Base64 encoding and decoding.



You can install partner and community operators from the embedded OperatorHub page.



Cluster-wide metrics such as CPU, memory, and storage usage are displayed on the Dashboards page.



Project Details pages display metrics specific to the project, such as the top ten memory consumers by pod and the current resource quota usage.