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 \ ❶
date-job \ ❷
--image registry.access.redhat.com/ubi8/ubi \ ❸
-- /bin/bash -c "date" ❹
```

❶ Creates a job resource.

❷ Specifies a job name of `date-job`.

❸ Sets `registry.access.redhat.com/ubi8/ubi` as the container image for the job pods.

❹ 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

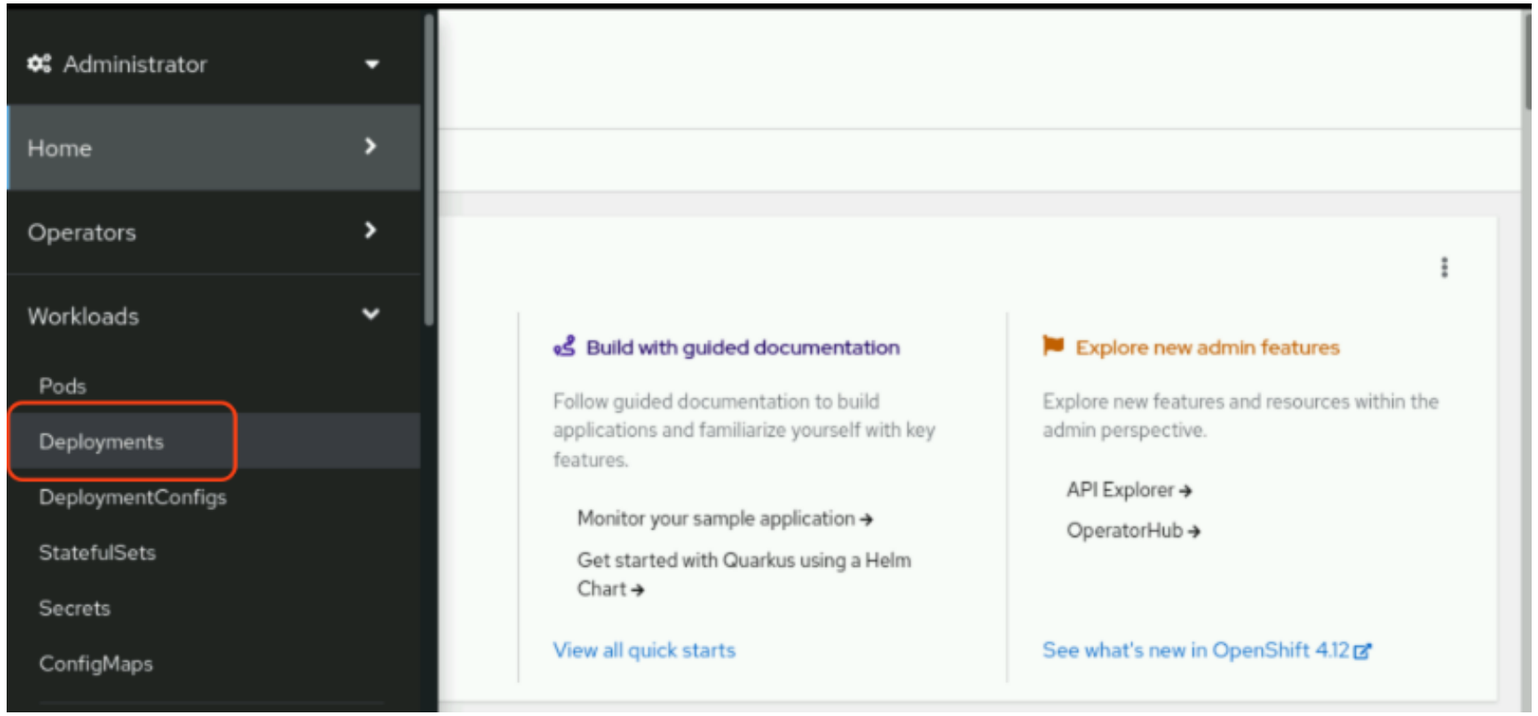
- Creates and manages replica set to maintain multiple pods
  - add or remove pods specified by deployment specifications
- Replica sets use selectors, such as label
  - to identify pods part of set
- Recreated in even of pods failure or accidentally deletion
- Jobs - on time execution
- Deployment -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

Deployments - create

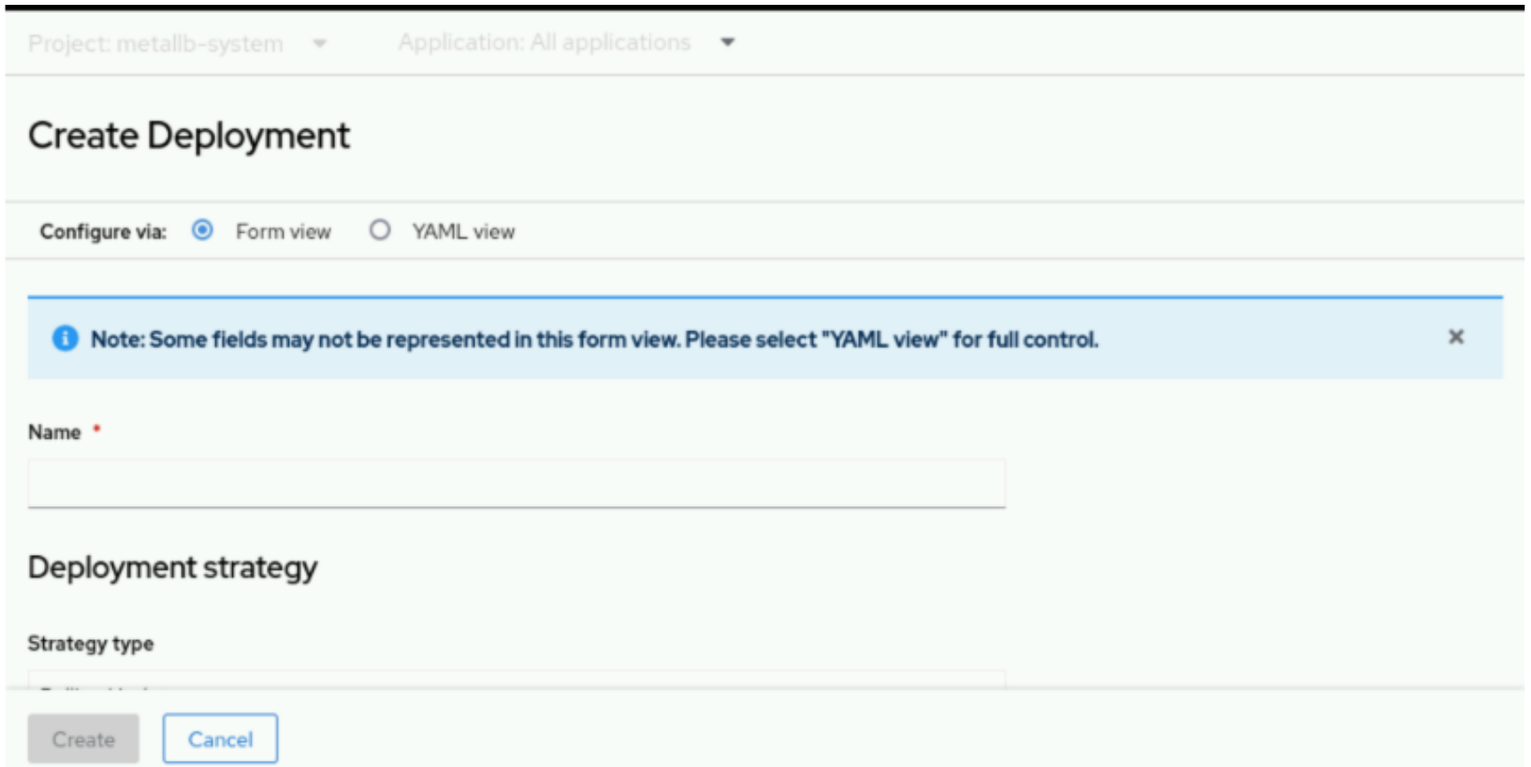
```
[user@host ~]$ oc create deployment \ (1)
my-deployment \ (2)
--image registry.access.redhat.com/ubi8/ubi \ (3)
--replicas 3 (4)
```

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



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.