Chapter 4 : Deploy Managed and Networked Applications on Kubernetes

Chapter objectives

Lesson 1: Deploy Applications from Images and Templates

Identify the main resources and settings that Kubernetes uses to manage long-lived applications and demonstrate how OpenShift simplifies common application deployment workflows.

Deploying Applications

- Microservices
- DevOps
- Containers-based infrastructures
- The term *applications*

Kubernetes Terminology

Term	Definition
Node	A server that hosts applications in a Kubernetes cluster.
Master Node	A node server that manages the **`control plane`** in a Kubernetes cluster. Master nodes provide basic cluster services such as APIs or controllers.
Worker Node	Also named **`Compute Node`**, worker nodes execute workloads for the cluster. Application pods are scheduled onto worker nodes.
Resource	Resources are any kind of component definition managed by Kubernetes. Resources contain the configuration of the managed component (for example, the role assigned to a node), and the current state of the component (for example, if the node is available).
Controller	A controller is a Kubernetes process that watches resources and makes changes attempting to move the current state towards the desired state.
Label	A key-value pair that can be assigned to any Kubernetes resource. Selectors use labels to filter eligible resources for scheduling and other operations.
Namespace	A scope for Kubernetes resources and processes, so that resources with the same name can be used in different boundaries.

Note

The latest Kubernetes versions implement many controllers as *Operators*. Operators are Kubernetes plug-in components that can react to cluster events and control the state of resources. Operators and CoreOS Operator Framework are outside the scope of this document.

OpenShift Terminology

Term	Definition
Infra Node	A node server containing infrastructure services like monitoring, logging, or external routing.
Console	A web UI provided by the RHOCP cluster that allows developers and administrators to interact with cluster resources.
Project	OpenShift's extension of Kubernetes' namespaces. Allows the definition of user access control (UAC) to resources.

Resources and Resource Definitions

- Configuration pieces of cluster
- Endpoint in Kubernetes API
- Collection of API objects
- Declarative syntax
- Resource type
  a) POD
  b) Deployment
  c) Services
  d) Routes
  e) Config Maps / Secrets
  f) Templates

OpenShift Templates

- Deploying applications requires creating several resources
  a) BuildConfig
  b) Deployment
  c) DeploymentConfig
  d) Service
  e) Quotas
  f) Limits
  g) Routes
  h) Pods
- Template
  a) simplify resources creation
  b) Reusable
  c) Can be processed by dynamic parameters

Listing OpenShift templates

- List installed templates by oc get templates –n openshift project

```
[student@workstation ~]$ oc get templates -n openshift
NAME                      DESCRIPTION
cakephp-mysql-example     An example CakePHP application ...
cakephp-mysql-persistent  An example CakePHP application ...
dancer-mysql-example      An example Dancer application with a MySQL ...
dancer-mysql-persistent   An example Dancer application with a MySQL ...
django-psql-example       An example Django application with a PostgreSQL ...
...output omitted...
rails-pgsql-persistent    An example Rails application with a PostgreSQL ...
rails-postgresql-example  An example Rails application with a PostgreSQL ...
redis-ephemeral           Redis in-memory data structure store, ...
redis-persistent          Redis in-memory data structure store, ...
```

- Extract a particular yaml template definition

Extract a particular yaml template definition

```
[student@workstation ~]$ oc get template mysql-persistent -n openshift -o yaml
apiVersion: template.openshift.io/v1
kind: Template
labels: ...value omitted...
message: ...message omitted ...
metadata:
  annotations:
    description: ...description omitted...
    iconClass: icon-mysql-database
    openshift.io/display-name: MySQL
    openshift.io/documentation-url: ...value o
    openshift.io/long-description: ...value om
    openshift.io/provider-display-name: Red Ha
    openshift.io/support-url: https://access.r
    tags: database,mysql ❶
  labels: ...value omitted...
  name: mysql-persistent ❷
objects: ❸
- apiVersion: v1
  kind: Secret
  metadata:
    annotations: ...annotations omitted...
    name: ${DATABASE_SERVICE_NAME} ❹
  stringData: ...stringData omitted...
- apiVersion: v1
  kind: Service
  metadata:
```

```
  spec: ...spec omitted...
- apiVersion: v1
  kind: DeploymentConfig
  metadata:
    annotations: ...annotations omitted...
    name: ${DATABASE_SERVICE_NAME}
  spec: ...spec omitted...
parameters: ❺
- ...MEMORY_LIMIT parameter omitted...
- ...NAMESPACE parameter omitted...
- description: The name of the OpenShift Service exposed for the database.
  displayName: Database Service Name
  name: DATABASE_SERVICE_NAME ❻
  required: true
  value: mysql
- ...MYSQL_USER parameter omitted...
- description: Password for the MySQL connection user.
  displayName: MySQL Connection Password
  from: '[a-zA-Z0-9]{16}' ❼
  generate: expression
  name: MYSQL_PASSWORD
  required: true
- ...MYSQL_ROOT_PASSWORD parameter omitted...
- ...MYSQL_DATABASE parameter omitted...
- ...VOLUME_CAPACITY parameter omitted...
- ...MYSQL_VERSION parameter omitted...
```

Process a template and deploy it

```
$ oc get template mysql-persistent -o yaml \
> -n openshift > mysql-persistent-template.yaml
```

Next, identify appropriate values for the template parameters and process the template:

```
$ oc process -f mysql-persistent-template.yaml \
> -p MYSQL_USER=dev -p MYSQL_PASSWORD=$P4SSD -p MYSQL_DATABASE=bank \
> -p VOLUME_CAPACITY=10Gi | oc create -f -
```

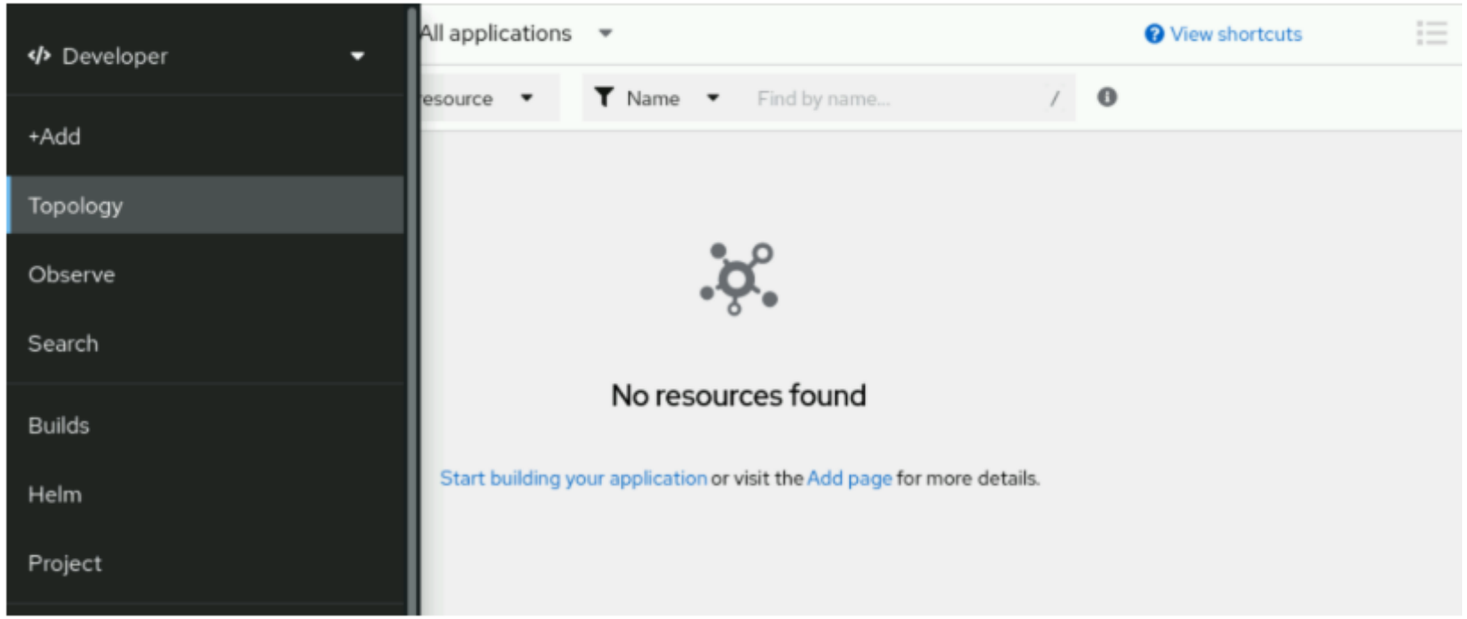
You can also use two slashes (`//`) to provide the namespace as part of the template name:

```
$ oc process openshift//mysql-persistent \
> -p MYSQL_USER=dev -p MYSQL_PASSWORD=$P4SSD -p MYSQL_DATABASE=bank \
> -p VOLUME_CAPACITY=10Gi | oc create -f -
```

Alternatively, it is possible to create an application using the **oc new-app** command passing the template name as the `--template` option argument:

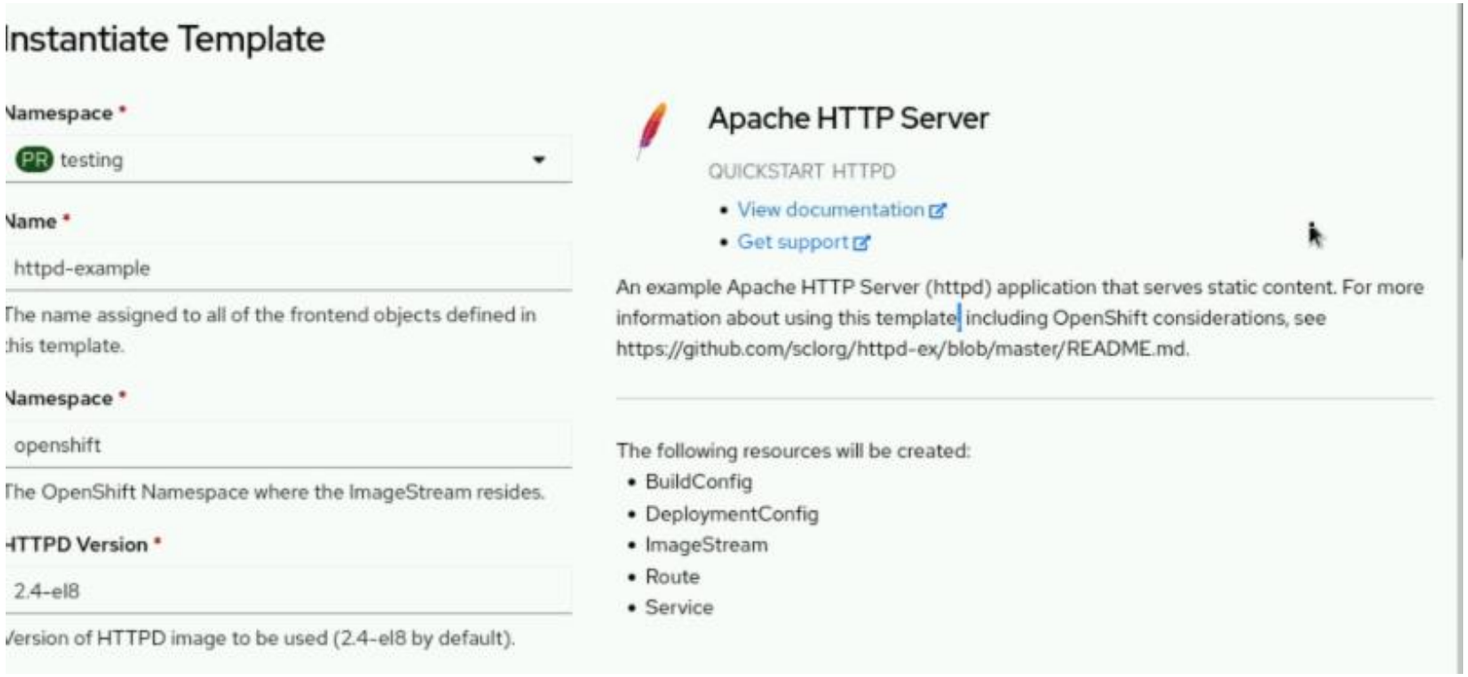
```
$ oc new-app --template=mysql-persistent \
> -p MYSQL_USER=dev -p MYSQL_PASSWORD=$P4SSD -p MYSQL_DATABASE=bank \
> -p VOLUME_CAPACITY=10Gi \
> --as-deployment-config
```

Instantiate Templates from web Console



- Change to **Developer** perspective → **Topology Menu** > Click on **Start Building your application** link

Instantiate Templates from web Console



- Change default values
  a) Git repository
  b) Application version
  c) Memory limits and much more

Describing K8s Resource Types

Namespace

a single cluster used by an organization can be divided and categorized into multiple sub-clusters and managed individually.

Pods (po)

Collection of containers that share resources. Basic unit of work for k8s

Services (svc)

Provide stable interface to all pods in deployment. By default, services connect clients to pods in round-robin fashion

Replication Controllers (rc)

A k8s resource that defines how pods are replicated and scheduled onto nodes. Basic k8s service to provide HA for pods and containers

Describing K8s Resource Types

Persistent Volume (pv)

Provide permanent storage to pods

Persistent Volume Claims (pvc)

PVCs links PV to pod so that containers can mount storage to container's file system

ConfigMaps (cm) and Secrets

Contains set of keys values pair that can be used by resources.

Provides centralized configuration values used by resources.

Secrets values are always encoded and restricted to authorized users only

Describing OpenShift Resource Types

Project

logical and organizational isolation to separate your application component resources. Resources in one project can access resources in other projects, but not by default.

Deployment config(dc)

Like deployment, represents set of containers included in pod. Specifies deployment strategies to be used. A dc also provides basic but extensible continuous delivery workflow

Build Config (bc)

Defines process to be executed in project.

Used by S2I feature to build container image from application source code stored in Git repo.

A bc works with dc to provide basic but extensible continuous integration and continuous delivery workflow

Routes

Provides fully-qualified domain name recognized by OpenShift router as an ingress point for application and microservices

K8s Namespace vs OpenShift Project

A project is essentially the same as a namespace, but OpenShift provides additional **administrative controls** for projects.

```
[student@workstation ~]$ oc describe project cli-review
Name:                     cli-review
Created:                  22 hours ago
Labels:                   kubernetes.io/metadata.name=cli-review
                          pod-security.kubernetes.io/audit=restricted
                          pod-security.kubernetes.io/audit-version=v1.24
                          pod-security.kubernetes.io/warn=restricted
                          pod-security.kubernetes.io/warn-version=v1.24
Annotations:              openshift.io/description=
                          openshift.io/display-name=
                          openshift.io/requester=admin
                          openshift.io/sa.scc.mcs=s0:c26,c20
                          openshift.io/sa.scc.supplemental-groups=1000690000/10000
                          openshift.io/sa.scc.uid-range=1000690000/10000
Display Name:             <none>
Description:              <none>
Status:                   Active
Node Selector:            <none>
Quota:                    <none>
Resource limits:          <none>
```

```
[student@workstation ~]$ oc describe namespace cli-review
Name:         cli-review
Labels:       kubernetes.io/metadata.name=cli-review
              pod-security.kubernetes.io/audit=restricted
              pod-security.kubernetes.io/audit-version=v1.24
              pod-security.kubernetes.io/warn=restricted
              pod-security.kubernetes.io/warn-version=v1.24
Annotations:  openshift.io/description:
              openshift.io/display-name:
              openshift.io/requester: admin
              openshift.io/sa.scc.mcs: s0:c26,c20
              openshift.io/sa.scc.supplemental-groups: 1000690000/10000
              openshift.io/sa.scc.uid-range: 1000690000/10000
Status:       Active

No resource quota.

No LimitRange resource.
```

OpenShift DeploymentConfig vs K8s Deployment

```yaml
apiVersion: v1
kind: DeploymentConfig
metadata:
  name: frontend
spec:
  replicas: 5
  selector:
    name: frontend
  template: { ... }
  triggers:
  - type: ConfigChange  (1)
  - imageChangeParams:
      automatic: true
      containerNames:
      - helloworld
      from:
        kind: ImageStreamTag
        name: hello-openshift:latest
    type: ImageChange  (2)
  strategy:
    type: Rolling  (3)
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-openshift
spec:
  replicas: 1
  selector:
    matchLabels:
      app: hello-openshift
  template:
    metadata:
      labels:
        app: hello-openshift
    spec:
      containers:
      - name: hello-openshift
        image: openshift/hello-openshift:latest
        ports:
        - containerPort: 80
```

Resource type - Networking

Container IP

- Ephemeral IP address
- Assigned from internal network that is accessible only from node running the container

Software-Defined network (SDN)

- Provides communication between container in pods between nodes
- Access to SDN only works from inside same Kubernetes cluster

Resource type - Networking

Services

Containers do not connect each other dynamic IP address directly

Uses services by linking more stable IP addresses from SDN to pods

Pods restarted, replicated, rescheduled to different nodes – services get updated, providing scalability and high availability

External Access

Is more complicated.

K8s uses NodePort attribute to provide external access. But it is insecured and doesn’t scale well

Resource type - Networking

OpenShift Routes - External Access

- Simpler by defining route resources
- Route defines external-facing DNS names and ports for service
- A router (ingress controller) forwards HTTP(s) requests to service addresses inside K8s SDN.
- OpenShift map IP addresses of RHOCP router nodes

Managing Resources from Command Line

- K8s and RHOCP share common commands
- Exclusivity on some commands
- Two categories :

a) **Declarative** : Defines state that cluster attempts to match
   - Use manifest to create resources
   - Automatically necessary resources
   - Less control / customization over resources

b) **Imperative** : Instructs cluster what to perform exactly
   - Faster way of creating pods (do not require pod object definition
   - Cannot handle versioning or incremental change to pod definition
   - Better customization needs

Examples of Imperative

The `create` command is an imperative way to create resources, and is included in both of the `oc` and `kubectl` commands. For example, the following command creates a deployment called `my-app` that creates pods that are based on the specified image.

```
[user@host ~]$ oc create deployment my-app --image example.com/my-image:dev
deployment.apps/my-app created
```

Use the `set` command to define attributes on a resource, such as environment variables. For example, the following command adds the `TEAM=red` environment variable to the preceding deployment.

```
[user@host ~]$ oc set env deployment/my-app TEAM=red
deployment.apps/my-app updated
```

Another imperative approach to creating a resource is the `run` command. In the following example, the `run` command creates the `example-pod` pod.

```
[user@host ~]$ oc run example-pod \ (1)
--image=registry.access.redhat.com/ubi8/httpd-24 \ (2)
--env GREETING='Hello from the awesome container' \ (3)
--port 8080 (4)
pod/example-pod created
```

Examples of Declarative

```
[user@host ~]$ oc create -f my-app-deployment.yaml
deployment.apps/my-app created
```

```
[user@host ~]$ oc new-app --file=./example/my-app.yaml
...output omitted...
--> Creating resources ...
    imagestream.image.openshift.io "my-app" created
    deployment.apps "my-app" created
    services "my-app" created
...output omitted...
```

```
[user@host ~]$ oc new-app --template mysql-persistent \
--param MYSQL_USER=operator --param MYSQL_PASSWORD=myP@55 \
--param MYSQL_DATABASE=mydata \
--param DATABASE_SERVICE_NAME=db
--> Deploying template "db-app/mysql-persistent" to project db-app
...output omitted...
     The following service(s) have been created in your project: db.

            Username: operator
            Password: myP@55
       Database Name: mydata
      Connection URL: mysql://db:3306/
...output omitted...
     * With parameters:
        * Memory Limit=512Mi
        * Namespace=openshift
        * Database Service Name=db
        * MySQL Connection Username=operator
        * MySQL Connection Password=myP@55
        * MySQL root user Password=tlH8BThuVgnIrCon # generated
        * MySQL Database Name=mydata
        * Volume Capacity=1Gi
        * Version of MySQL Image=8.0-el8

--> Creating resources ...
    secret "db" created
```

USe oc new-app imperatively

```
[user@host ~]$ oc new-app --image example.com/my-app:dev
...output omitted...
--> Creating resources ...
    imagestream.image.openshift.io "my-app" created
    deployment.apps "my-app" created
...output omitted...
```

```
[user@host ~]$ oc new-app https://github.com/apache/httpd.git#2.4.56
...output omitted...
--> Creating resources ...
    imagestream.image.openshift.io "httpd24" created
    deployment.apps "httpd24" created
...output omitted...
```

Retrieving Resource Information

- Use oc describe subcommand to retrieve details information about resources
- The syntax:

```
$ oc describe RESOURCE_TYPE
```

- Resource_type:
  a) pod
  b) deployment
  c) deployment config or dc
  d) service or svc
  e) route
  f) build config or bc
  g) imagestream or is

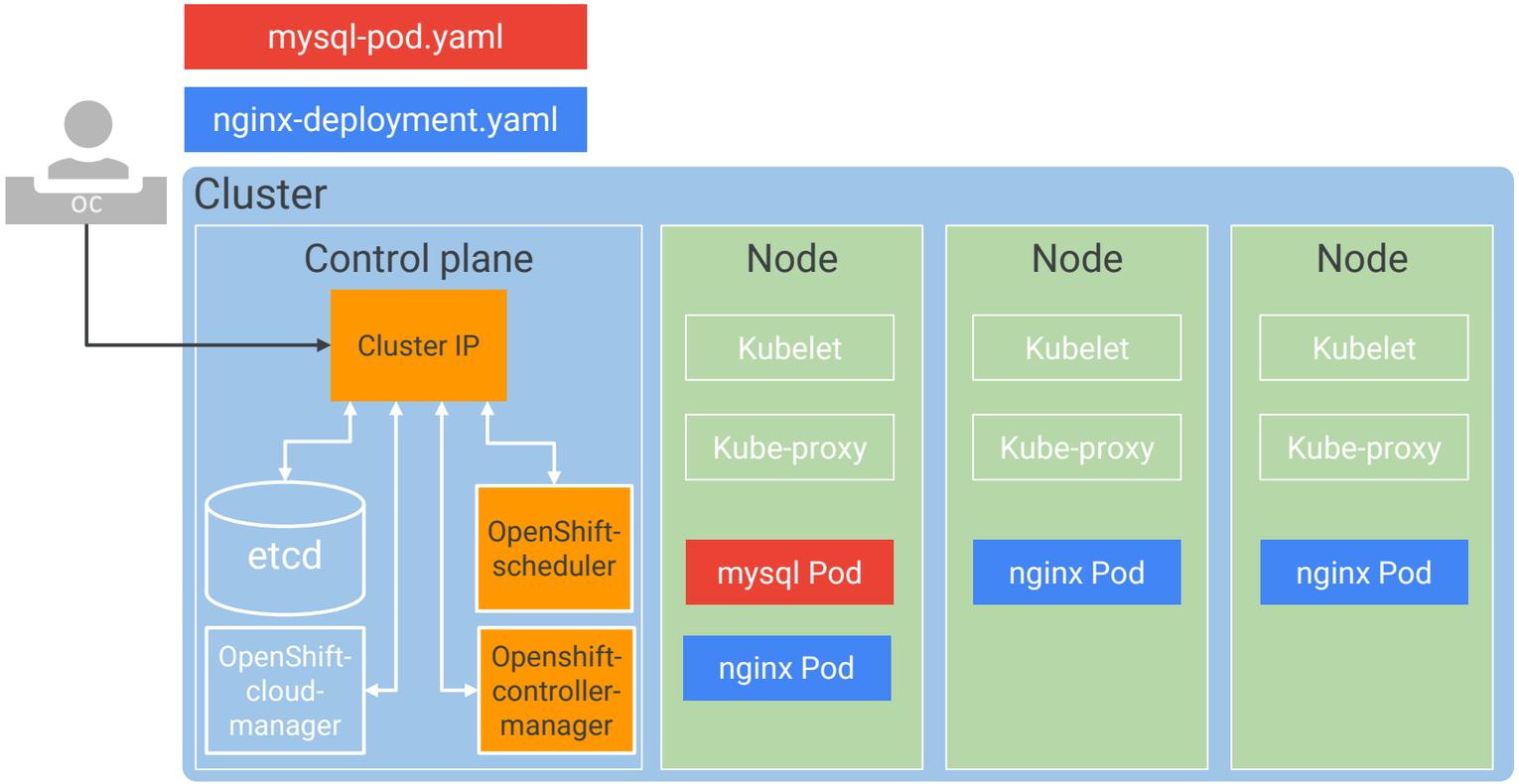
The oc describe command example

- Retrieve detailed information on particular pod

```
$ oc describe pod mysql-openshift-1-glqrp
```

```
Name:       mysql-openshift-1-glqrp
Namespace:          mysql-openshift
Priority:           0
PriorityClassName:  none
Node:               cluster-worker-1/172.25.250.52
Start Time:         Fri, 15 Feb 2019 02:14:34 +0000
Labels:             app=mysql-openshift
                    deployment=mysql-openshift-1
                    deploymentconfig=mysql-openshift
Annotations:        openshift.io/deployment-config.latest-version: 1
                    openshift.io/deployment-config.name: mysql-openshift
                    openshift.io/deployment.name: mysql-openshift-1
                    openshift.io/generated-by: OpenShiftNewApp
                    openshift.io/scc: restricted
Status:             Running
IP:                 10.129.0.85
```

A Pod or A Deployment?



Guided Exercise: Deploy Applications from Images and Templates

In this exercise, you deploy two MySQL database server pods to compare deployment methods and the resources that each creates.

- Deploy a database from a template.
- Deploy a database from a container image.