

# *OpenShift: Hands-On Examples*

*By Kelvin Lai*

This lab serves to solidify the understanding of container architecture principles covered in the course.

## **Document Convention:**

Ellipses (...) Due to its excessive length, some of the output produced on the command lines have been abbreviated.

Red Arrow (-) Pay special attention to these lines. The trainer will explain them in detail.

**INPUT** The text that is **Bold**, *Italic*, and in **Dark Green** should be completed using your specific information.

## **Table of Contents:**

<b>Lab 1: Managing a Pod</b>	<b>8</b>
<b>Lab 2: Dangers of cluster admin</b>	<b>13</b>
<b>Lab 3: Creating a Deployment</b>	<b>15</b>
<b>Lab 4: Deploying application</b>	<b>21</b>
<b>Lab: Persistent Storage</b>	<b>27</b>
<b>Lab: Authentication and Authorization</b>	<b>27</b>
<b>Lab: SA and SCC</b>	<b>27</b>
<b>Lab: Network Policies</b>	<b>27</b>
<b>Lab: Quota</b>	<b>27</b>

# Lab 1: Managing a Pod

In this Lab, we will cover the following topics:

- Creating and deleting a Project.
- Creating a Pod using command line and YAML.
- Understanding resource definitions and getting help on syntax.
- Editing Pod definitions.
- Executing commands in a container of a running Pod.
- Monitoring events in a Project.
- Deleting resources by name or by label.

## 1. Login to the cluster

```
[user@host ~]$ oc login -u kubeadmin -p pytTS...XpX https://api.crc.testing:6443
```

OR

```
[user@host ~]$ oc login -u kubeadmin https://api.ocp.example.com:6443
Username: kubeadmin
Password: pytTS-GGXrX-h2EdQ-VHXPX
```

## 2. Check/verify current login user

```
[user@host ~]$ oc whoami
kubeadmin
```

## 3. Create three projects named *testing*, *one* and *two*.

```
[user@host ~]$ oc new-project testing
[user@host ~]$ oc new-project one
[user@host ~]$ oc new-project two
```

## 4. Listing projects

- a. List all projects that the user can access.

```
[user@host ~]$ oc projects
```

OR

```
[user@host ~]$ oc get projects
```

- b. Check which is our current/active project.

```
[user@host ~]$ oc project
```

### Note:

- Use the `oc api-resources` command to discover the resource names, short names and determine whether they are namespace-dependent or not.

5. Delete project *one* and *two*. Make sure *testing* is the current project.

```
[user@host ~]$ oc delete project one
[user@host ~]$ oc delete project two
[user@host ~]$ oc project testing
```

6. Create a new pod

a. Name it testing

b. Use the image [registry.access.redhat.com/ubi8/ubi:latest](https://registry.access.redhat.com/ubi8/ubi:latest).

```
[user@host ~]$ oc run testing --image registry.access.redhat.com/ubi8/ubi
pod/testing created

[user@host ~]$ oc get pods
```

NAME	READY	STATUS	RESTARTS	AGE
testing	0/1	Completed	0	2s

**Note:**

- The *latest* tag is optional and doesn't need to be typed.
- The status shows **completed** because there is nothing running in the container.
- Try running `oc get pods -o wide`

7. Get more details from the pod. Look through the information provided by the output of the command.

```
[user@host ~]$ oc describe pod/testing
```

8. Continuously check the pod. Why is it continuously looping between Completed and CrashLoopBackOff?

a. Watch for changes. You can break it with <CTRL>+C after a few lines of output.

```
[user@host ~]$ oc get po -w
```

NAME	READY	STATUS	RESTARTS	AGE
testing	0/1	CrashLoopBackOff	3 (46s ago)	93s
testing	0/1	Completed	4 (53s ago)	100s
testing	0/1	CrashLoopBackOff	4 (15s ago)	115s

b. Check the pod's restart policy by looking at the resource definition of the pod. Every resource in openshift/kubernetes is defined by their resource definition (including the *node* resource type)

```
[user@host ~]$ oc get po testing -o yaml
...
restartPolicy: Always
...
```

**Note:**

- Learning points: basic structure of resource definition apiVersion, metadata, kind, spec.
- If you like JSON instead of YAML, use `oc get po testing -o json`.
- To learn pod resource definition syntax use `oc explain pod.spec`.
- The restartPolicy can only be set to Always(default), Never or OnFailure. It can be set by editing the resource definition or using the `--restart` option of the `oc run` command.

**9. Cleanup. Delete the pod.**

```
[user@host ~]$ oc delete po testing
```

**10. Create a new Pod named, *myubi* using the image *quay.io/kelvinlai/myubi:latest*. BEFORE you run the commands, get the instructor to set the repository to private. We will be monitoring the events for this pod creation.**

- a. Monitor the events for your current project.

```
[user@host ~]$ oc get events -w
```

- b. Open another terminal/PowerShell and create the pod.

```
[user@host ~]$ oc run myubi --image quay.io/kelvinlai/myubi:latest
pod/testing created
```

- c. Check the status of the container. Depending on how fast you type the command, you might see these status: *ContainerCreating*, *ErrImagePull*, *ImagePullBackOff*. It means there is a problem pulling the image. Check the events in your first terminal.

```
[user@host ~]$ oc get po
NAME          READY   STATUS              RESTARTS   AGE
pod/myubi     1/1     ImagePullBackOff    0           12m
```

- d. Now, get the instructor to set the repository to public. The pods will successfully pull the image and the status will change to Running.

**Question:** Why did the pod not exit like before in Step 5?

- e. Check what is running inside the pod.

```
[user@host ~]$ oc rsh myubi ps -ef
```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
1000660+	1	0	0	08:50	?	00:00:00	sleep infinity
1000660+	13	0	0	09:34	pts/0	00:00:00	ps -ef

OR

```
[user@host ~]$ oc rsh myubi
```

```
sh-5.1$ ps -ef
```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
1000660+	1	0	0	08:50	?	00:00:00	/usr/bin/coreutils
--coreutils-prog-shebang=sleep /usr/bin/sleep infinity							
1000660+	18	0	1	10:02	pts/0	00:00:00	/bin/sh
1000660+	23	18	0	10:02	pts/0	00:00:00	ps -ef

```
sh-5.1$ exit
```

```
exit
```

```
[user@host ~]$
```

11. Save the resource definition to a file and create *mypod* from that file. After *mypod* is created, add a label *abc=def* to it. Clean up by deleting the pod.

- a. Export the resource definition to a file and edit it to make appropriate changes. Windows users, replace *vi* with *notepad*.

```
[user@host ~]$ oc get po myubi -o yaml > mypod.yaml
```

```
[user@host ~]$ vi mypod.yaml
```

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: myubi
  name: mypod
spec:
  containers:
  - image: quay.io/kelvinlai/myubi      ← image reference
    name: mycontainer
```

- b. Create the pod.

```
[user@host ~]$ oc create -f mypod.yaml
```

- c. Add the `abc=def` label to the pod. Save and exit.

```
[user@host ~]$ oc edit pod mypod
...
labels:
  run: myubi
  abc: def
name: mypod
...
```

OR

```
[user@host ~]$ oc label pod mypod abc=def
...
```

**Note:**

- To remove the label: `oc label pod mypod abc-`

- d. Check the pod has the new label.

```
[user@host ~]$ oc describe po mypod
```

- e. Check the pod has the new label.

```
[user@host ~]$ oc delete po -l abc=def
pod "mypod" deleted
```

12. Lab completed. We will be using `myubi` pod as a testing platform for subsequent labs. Deleted pods in Step 8 and 10e will not be auto-created because there is no `replicationcontroller` or `replicaset` resource defined.

## Lab 2: Dangers of cluster admin

In this Lab, we will cover the following topics:

- Security threat of performing deployment as cluster admin.
- The importance of logging out.
- Deleting everything.

### 1. Login to the cluster

```
[user@host ~]$ oc login -u developer -p developer https://api.ocp.example.com:6443
```

### 2. Create a new project named *dcproj*.

```
[user@host ~]$ oc new-project dcproj
```

### 3. Create a Pod, *myweb-dc* using the image *quay.io/kelvinlai/myweb:openshift*.

```
[user@host ~]$ oc run myweb-dc --image quay.io/kelvinlai/myweb:openshift
pod/myweb-dc created
[user@host ~]$
```

### 4. Verify.

```
[user@host ~]$ oc get po
NAME      READY   STATUS    RESTARTS   AGE
myweb-dc  1/1     Running   0           27s
```

### 5. Login as kubeadmin

```
[user@host ~]$ oc login -u kubeadmin
```

#### Note:

- You can access the *kubeadmin* account without entering a password due to the token saved in the *~/.kube/config* file. To maintain security, always remember to log out once you have completed your tasks.
6. Repeat Step 3, but name the Pod *myweb-ca*. You may use history recall (up arrow key) and change the name of the pod before executing the command.

```
[user@host ~]$ oc run myweb-ca --image quay.io/kelvinlai/myweb:openshift
pod/myweb-ca created
```

7. Now, log in again as the developer user and list all the pods. Check who is the process owner for both the pods.

```
[user@host ~]$ oc login -u developer
[user@host ~]$ oc get po
NAME          READY   STATUS    RESTARTS   AGE
myweb-ca      1/1     Running   0           18s
myweb-dc      1/1     Running   0           88s
[user@host ~]$ oc rsh myweb-dc whoami
1000740000
[user@host ~]$ oc rsh myweb-ca whoami
root
```

8. Now clean up by deleting everything.

```
[user@host ~]$ oc delete all --all
pod "myweb-ca" deleted
pod "myweb-dc" deleted
[user@host ~]$ oc get all
```

**Note:**

- `oc get all` doesn't really list all resources. Instead it lists all legacy user resources such as dc, deploy, rc, rs, pod, svc, daemonset, statefulset, jobs, cronjobs and hpa.

Reference:

<https://github.com/openshift/origin/blob/release-3.7/vendor/k8s.io/kubernetes/pkg/kubectl/resource/categories.go#L113-L123>

9. (Optional) Repeat from step 3 using `myweb:non_openshift` to observe the results. Summary:

As cluster-admin

- \* able to oc run dangerous images... `myweb:non_openshift`
- \* if using `oc create deploy` or `deploymentconfig` it will fail.

As a regular user

- \* running application using `oc run`, `oc create deploy` or `dc` will all fail.



## Lab 3: Creating a Deployment

In this Lab, we will cover the following topics:

- Create a Deployment and Service.
- Verify auto healing.
- Investigate the resource definition of all the resources.
- Access Pod by using service and pod's ip.
- Creating service, and getting endpoints.
- Give external access by creating a route

1. Create a deployment named *myweb* using [quay.io/kelvinlai/myweb:openshift](https://quay.io/repository/kelvinlai/myweb?tab=tags).

```
[user@host ~]$ oc login -u developer
[user@host ~]$ oc create deploy myweb --image quay.io/kelvinlai/myweb:openshift
deployment.apps/myweb created

[user@host ~]$ oc get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/myweb-c687b9f4d-872gj          1/1     Running   0           75s

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/myweb              1/1     1            1           75s

NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/myweb-c687b9f4d    1         1         1       75s

[user@host ~]$
```

2. Try deleting the pod and see what happens.

```
[user@host ~]$ oc delete po myweb-c687b9f4d-872gj
pod "myweb-c687b9f4d-872gj" deleted

[user@host ~]$ oc get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/myweb-c687b9f4d-9vpr4          1/1     Running   0           4s

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/myweb              1/1     1            1           99s

NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/myweb-c687b9f4d    1         1         1       99s

[user@host ~]$
```

3. Now try deleting the replicaset and see what happens.

```
[user@host ~]$ oc delete rs myweb-c687b9f4d
replicaset.apps "myweb-c687b9f4d" deleted

[user@host ~]$ oc get all

NAME                                READY   STATUS    RESTARTS   AGE
pod/myweb-c687b9f4d-2k9xh          1/1     Running   0           4s
NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/myweb              1/1     1             1           2m4s
NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/myweb-c687b9f4d    1         1         1       4s

[user@host ~]$
```

4. Finally, try deleting the deployment.

```
[user@host ~]$ oc delete deployment myweb
deployment.apps "myweb" deleted

[user@host ~]$ oc get all

No resources found in dcproj namespace.
[user@host ~]$
```

When there is a deployment or deployment configuration (dc/deploy), always prioritize addressing it. Only handle the replication controller (rc) or replica set (rs) if no dc/deploy exists. And finally, only address the Pod directly if there is no rc or rs.

5. Recreate the deployment [myweb](#). We will be using it for the service and route testing.

```
[user@host ~]$ oc create deploy myweb --image quay.io/kelvinlai/myweb:openshift
deployment.apps/myweb created

[user@host ~]$ oc get all

NAME                                READY   STATUS    RESTARTS   AGE
pod/myweb-c687b9f4d-pdhwf          1/1     Running   0           9s
NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/myweb              1/1     1             1           9s
NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/myweb-c687b9f4d    1         1         1       9s

[user@host ~]$
```

6. Investigate the deployment, replicaset and pod resource definition.

```
[user@host ~]$ oc get deploy myweb -o yaml
...
[user@host ~]$ oc get rs myweb-c687b9f4d -o yaml
...
[user@host ~]$ oc get po myweb-c687b9f4d-pdhwf -o yaml
...
```

**Note:**

- Notice that the deployment doesn't have any 'ports' being declared. This is expected. The `oc create deploy` command does not read the exposed port metadata information from the image. This is true, regardless of whether you have an EXPOSE or io.openshift.expose-service LABEL declared in the Containerfile that was used to create that image.
- Notice the relationship of deploy, rs and pod's resource definition.

7. Get the Pod's IP

```
[user@host ~]$ oc get po -o wide
NAME             READY   STATUS    RESTARTS   AGE   IP            NODE             NOMINATED NODE   READINESS GATES
myweb-c687b9f4d-pdhwf 1/1     Running   0           39s   10.217.0.103  crc               <none>           <none>
[user@host ~]$
```

8. As the kubeadmin user, try accessing the myweb website from the myubi pod. You should be able to access the website using the Pod's IP address.

```
[user@host ~]$ oc login -u kubeadmin
Logged into "https://api.crc.testing:6443" as "kubeadmin" using existing credentials.
You have access to 68 projects, the list has been suppressed. You can list all
projects with 'oc projects'
Using project "dcproj".
[user@host ~]$ oc get po -n testing
NAME    READY   STATUS    RESTARTS   AGE
myubi   1/1     Running   1           12h

[user@host ~]$ oc rsh -n testing myubi curl 10.217.0.103:8080
Hello World
```

**Note:**

- When accessing another pod from a different namespace, you have to specify the namespace first before referring to the pod name or else the command will fail to find the pod.

9. Try accessing it from the local machine. This will **fail**, because the local machine does not belong to the cluster.

```
[user@host ~]$ curl 10.217.0.103:8080
```

10. Log back in as developer.

```
[user@host ~]$ oc login -u developer
Logged into "https://api.crc.testing:6443" as "developer" using existing credentials.
You have one project on this server: "dcproj"
Using project "dcproj".
```

11. Next try scaling the replicaset. You may use the **oc scale** command or **oc edit** to change the replicas of the replicaset.

```
[user@host ~]$ oc scale rs myweb-c687b9f4d --replicas=2
replicaset.apps/myweb-c687b9f4d scaled
[user@host ~]$ oc get po
NAME                                READY   STATUS    RESTARTS   AGE
myweb-c687b9f4d-pdhwf              1/1     Running   0           15m
[user@host ~]$
```

**Note:**

- It may appear that the command doesn't work because only one pod is running. However, if you monitor the events in real-time using **oc get events -w** in another terminal, you'll see that the underlying issue is related to the deployment. Since a replica set (rs) is created by the deployment and the deployment has its replicas set to 1, the replica set will be reset to 1 by the deployment.

12. Now, do it the right way by scaling the deployment.

```
[user@host ~]$ oc scale deploy myweb --replicas 2
deployment.apps/myweb scaled
[user@host ~]$ oc get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/myweb-c687b9f4d-pdhwf          1/1     Running   0           16m
pod/myweb-c687b9f4d-qdb24          0/1     Pending   0           4s
NAME                                TYPE          CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
service/myweb                       ClusterIP     10.217.5.34  <none>        8080/TCP   4m29s
NAME                                READY   UP-TO-DATE   AVAILABLE     AGE
deployment.apps/myweb               1/2     2             1             16m
NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/myweb-c687b9f4d     2         2         1       16m
```

13. Create a service. You need to specify the port number as the deployment didn't have the containers exposed port information.

```
[user@host ~]$ oc expose deployment myweb --port 8080
service/myweb exposed
[user@host ~]$ oc get svc
NAME                TYPE                CLUSTER-IP      EXTERNAL-IP      PORT(S) AGE
service/myweb       ClusterIP           10.217.5.34      <none>            8080/TCP   8s
[user@host ~]$ oc describe svc myweb
...
[user@host ~]$ oc get svc myweb -o yaml
```

14. Check the service and endpoints.

```
[user@host ~]$ oc get po -o wide
NAME                READY    STATUS RESTARTS   AGE    IP              NODE             NOMINATED NODE
READINESS GATES
myweb-c687b9f4d-pdhwf 1/1      Running    0                17m    10.217.0.103    crc              <none>
myweb-c687b9f4d-qdb24 1/1      Running    0                57s    10.217.0.120    crc              <none>

[user@host ~]$ oc get svc
NAME                TYPE                CLUSTER-IP      EXTERNAL-IP      PORT(S) AGE
myweb               ClusterIP           10.217.5.34      <none>            8080/TCP   6m10s
[user@host ~]$ oc get endpoints
NAME                ENDPOINTS                                     AGE
myweb               10.217.0.103:8080,10.217.0.120:8080        6m13s
```

15. Test the service using ip and internal DNS name of the service. Check the internal service variables injected by kubernetes into the Pod.

```
[user@host ~]$ oc login -u kubeadmin
Logged into "https://api.crc.testing:6443" as "kubeadmin" using existing credentials.
You have access to 68 projects, the list has been suppressed. You can list all
projects with 'oc projects'
Using project "dcproj".
[user@host ~]$ oc -n testing rsh myubi
# curl 10.217.0.103:8080
Hello World
# curl 10.217.0.120:8080
Hello World
# curl myweb.dcproj.svc:8080
Hello World
# env | grep SERVICE_HOST
MYWEB_SERVICE_HOST = 10.217.5.34
# env | grep SERVICE_PORT
MYWEB_SERVICE_PORT = 8080
# exit
```

**Note:**

- The internal dns domain is cluster.local. All services can be referenced by `SERVICE_NAME.PROJECT_NAME.svc[.cluster.local]`.

## 16. Login again as developer. Create and test the route resource.

```
[user@host ~]$ oc login -u developer
Logged into "https://api.crc.testing:6443" as "developer" using existing credentials.
You have one project on this server: "dcproj"
Using project "dcproj".

[user@host ~]$ oc get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/myweb-c687b9f4d-pdhwf           1/1     Running   0           21m
pod/myweb-c687b9f4d-qdb24           1/1     Running   0           4m37s
NAME                                 TYPE          CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
service/myweb                       ClusterIP     10.217.5.34   <none>         8080/TCP    9m2s
NAME                                READY   UP-TO-DATE   AVAILABLE     AGE
deployment.apps/myweb               2/2     2             2             21m
NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/myweb-c687b9f4d     2         2         2       21m

[user@host ~]$ oc expose svc myweb
route/myweb exposed
[user@host ~]$ oc get route
NAME                                HOST/PORT                                          PATH   SERVICES
PORT   TERMINATION   WILDCARD
route.route.openshift.io/myweb     myweb-dcproj.apps-crc.testing                  myweb  8080
None

[user@host ~]$ oc get route myweb -o yaml
...
[user@host ~]$ curl myweb-dcproj.apps-crc.testing
Hello World
```

## Lab 4: Deploying application

In this Lab, we will cover the following topics:

- Managed life cycle application
- Source-To-Image(S2I) through BuildConfig
- Parameter passing to applications
- Inter-project communication
- Port forwarding
- Image Stream
- Setting environment variables during and after application creation
- Tips on simple yaml creation.

Create two projects, one to host the database using an existing ImageStream and the other, a web application that makes use of the database.

1. Login as the developer and create a project using your id, **USERNAME-dbproj**

```
[user@host ~]$ oc login -u developer
[user@host ~]$ oc new-project kel-dbproj
Now using project "kel-dbproj" on server "https://api.cluster.kel-ocp.com:6443".
...
```

2. List the imagestreams that are available and search for mysql. Windows user replace **grep** with **FindStr**

```
[user@host ~]$ oc get is -n openshift | grep mysql
mysql
image-registry.openshift-image-registry.svc:5000/openshift/mysql
      8.0,8.0-el7,8.0-el8,8.0-el9,latest                25 hours ago
```

3. Learn how to use the new-app command. Look at the syntax at the bottom. Windows user, replace **tail -4** with **Select-Object -Last 4**.

```
[user@host ~]$ oc new-app --help | tail -4
Usage:
  oc new-app (IMAGE | IMAGESTREAM | TEMPLATE | PATH | URL ...) [flags] [options]

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

4. Create the application named mydb using the mysql:latest imagestreamtag.

```
[user@host ~]$ oc new-app mysql --name mydb

--> Creating resources ...
    deployment.apps "mydb" created
    service "mydb" created
--> Success
    Application is not exposed. You can expose services to the outside world by
    executing one or more of the commands below:
    'oc expose service/mydb'
    Run 'oc status' to view your app.
```

5. Check the resources. There will be an error from the pod.

```
[user@host ~]$ oc get all
Warning: apps.openshift.io/v1 DeploymentConfig is deprecated in v4.14+, unavailable in
v4.10000+
NAME                                READY    STATUS    RESTARTS   AGE
pod/mydb-5dd7cfd5f5-mc8cw          0/1      Error    3 (37s ago)  70s

NAME          TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S) AGE
service/mydb  ClusterIP   172.30.166.181 <none>         3306/TCP  71s

NAME          READY    UP-TO-DATE    AVAILABLE    AGE
deployment.apps/mydb  0/1      1              0            71s

NAME          DESIRED    CURRENT    READY    AGE
replicaset.apps/mydb-5bddd857b  1          0          0        71s
replicaset.apps/mydb-5dd7cfd5f5  1          1          0        71s
```

**Note:**

- Notice that when we deploy an application using the new-app command, OpenShift will automatically help us create the service if images metadata contain the exposed port information.

6. Investigate what caused the failure. You can either check the logs of the pod or the deployment.

```
[user@host ~]$ oc logs mydb-5dd7cfd5f5-mc8cw
=> sourcing 20-validate-variables.sh ...
You must either specify the following environment variables:
    MYSQL_USER (regex: '^[a-zA-Z0-9_]+$')
    MYSQL_PASSWORD (regex: '^[a-zA-Z0-9_~!@#%&*()-=<>,.?;:|]+$')
    MYSQL_DATABASE (regex: '^[a-zA-Z0-9_]+$')
...
```



7. The mysql pod needs to have environment variables to help us initialize the database. In order to protect our information, we will be using a secret. Create the secret and check the resource definition.

```
[user@host ~]$ oc create secret generic dbsecret --from-literal MYSQL_USER=albert
--from-literal MYSQL_PASSWORD=einstein --from-literal MYSQL_DATABASE=okd
secret/dbsecret created
[user@host ~]$ oc get secret
NAME                                TYPE                                DATA  AGE
...
dbsecret                            Opaque                              3      11s
...
[user@host ~]$ oc get secret dbsecret -o yaml
apiVersion: v1
data:
  MYSQL_DATABASE: bXlkYg==
  MYSQL_PASSWORD: ZWluc3RlaW4=
  MYSQL_USER: YWxiZXJ0
kind: Secret
metadata:
  creationTimestamp: "2024-06-26T15:35:39Z"
  name: dbsecret
  namespace: kel-dbproj
  resourceVersion: "314496"
  uid: 8796ee49-d87e-460b-8ef0-b4e5e52a3a8b
type: Opaque
```

**Note:**

- You may use --help to find the syntax of the commands:  
`oc create secret --help | tail -5`  
`oc create secret generic --help | tail -5`
- Secrets are just simple base64 encoding. You can decode it using -d option. Eg. `echo bXlkYg== | base64 -d`

8. Pass in the secret as environment variables to the deployment.

```
[user@host ~]$ oc set env deploy/mydb --from secret/dbsecret
deployment.apps/mydb updated
```

**Note:**

- Learn how to set environment: `oc set env --help`

9. Verify the deployment. A new rs will be created and the pod which is deployed will now be running.

```
[user@host ~]$ oc get all
Warning: apps.openshift.io/v1 DeploymentConfig is deprecated in v4.14+, unavailable in v4.10000+
NAME                                READY   STATUS    RESTARTS   AGE
pod/mydb-84fd78f9d8-jtlg8          1/1     Running   0           6s

NAME                TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
service/mydb        ClusterIP   172.30.166.181 <none>         3306/TCP    12m

NAME                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mydb 1/1     1            1           12m

NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/mydb-5bddd857b      0         0         0       12m
replicaset.apps/mydb-5dd7cfd5f5     0         0         0       12m
replicaset.apps/mydb-84fd78f9d8     1         1         1       6s
```

10. You may check the pod to see that the secret was passed as environment variables.

```
[user@host ~]$ oc rsh mydb-84fd78f9d8-jtlg8 env | grep MYSQL
MYSQL_USER=albert
MYSQL_DATABASE=mydb
MYSQL_PASSWORD=einstein
MYSQL_VERSION=8.0
MYSQL_PREFIX=/usr
```

11. Create the web application project, **USERNAME-webproj**

```
[user@host ~]$ oc new-project kel-webproj
```

12. The application we will deploy is using php. Check what php builder images are available. Windows users replace grep with FindStr.

```
[user@host ~]$ oc get is -n openshift | grep php
php
image-registry.openshift-image-registry.svc:5000/openshift/php
7.3,7.3-ubi7,7.4-ubi8,8.0-ubi8,8.0-ubi9 + 1 more...    25 hours ago
```

13. Create the application. Best method of forcing the new-app to disable auto detection of which builder image to use, is by using the tilde(~) symbol to separate the builder image tag to use and the source code. Let us try passing in the DB connection info as variables from the command line now.

```
[user@host ~]$ oc new-app --name myapp php:7.4-ubi8~https://github.com/kelvinlnx/myphp
-e DB_HOST=mydb.kel-dbproj.svc -e DB_USER=albert -e DB_PASS=einstein -e DB_NAME=okd
--> Found image d25c7b9 (31 hours old) in image stream "openshift/php" under tag
"7.4-ubi8" for "php:7.4-ubi8"

Tags: builder, php, php74, php-74

* A source build using source code from https://github.com/kelvinlnx/myphp will
be created
* The resulting image will be pushed to image stream tag "myapp:latest"
* Use 'oc start-build' to trigger a new build

--> Creating resources ...
imagestream.image.openshift.io "myapp" created
buildconfig.build.openshift.io "myapp" created
deployment.apps "myapp" created
service "myapp" created
--> Success
Build scheduled, use 'oc logs -f buildconfig/myapp' to track its progress.
Application is not exposed. You can expose services to the outside world by
executing one or more of the commands below:
'oc expose service/myapp'
Run 'oc status' to view your app.
[user@host ~]$ oc get all
NAME                                READY    STATUS RESTARTS   AGE
pod/myapp-1-build                  1/1      Running    0                 37s

NAME                                TYPE           CLUSTER-IP      EXTERNAL-IP      PORT(S)              AGE
service/myapp                      ClusterIP      172.30.194.74   <none>           8080/TCP,8443/TCP    37s

NAME                                READY    UP-TO-DATE    AVAILABLE    AGE
deployment.apps/myapp              0/1      0              0            37s

NAME                                DESIRED    CURRENT    READY    AGE
replicaset.apps/myapp-79b9cd7b6c    1          0          0        37s

NAME                                TYPE      FROM    LATEST
buildconfig.build.openshift.io/myapp Source    Git     1

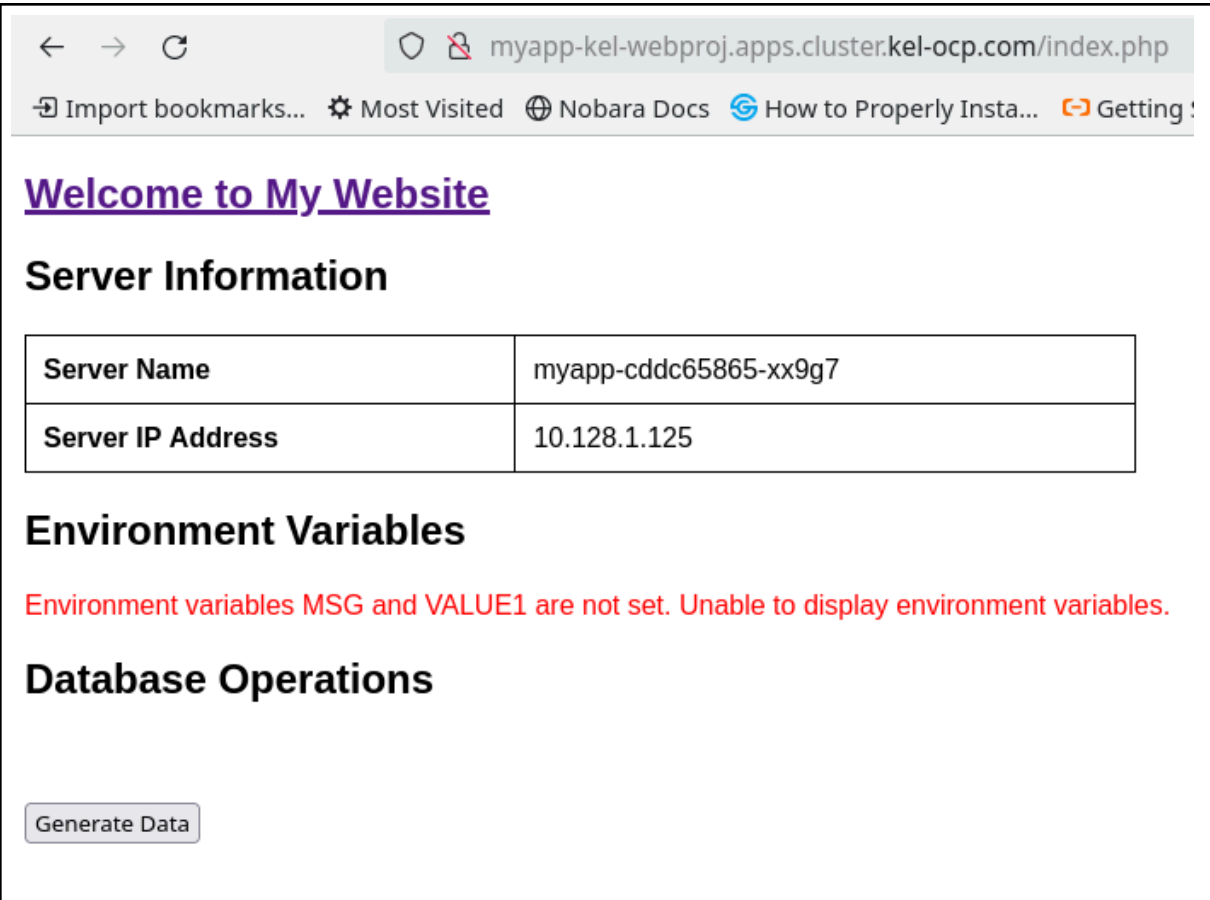
NAME                                TYPE      FROM              STATUS    STARTED
DURATION
build.build.openshift.io/myapp-1    Source    Git@fb2e8ba      Running   38 seconds ago

NAME                                IMAGE REPOSITORY
TAGS    UPDATED
imagestream.image.openshift.io/myapp
image-registry.openshift-image-registry.svc:5000/kel-webproj/myapp
```

14. Create a route and use your browser to browse to the website.

```
[user@host ~]$ oc expose svc myapp
route/myapp exposed
[user@host ~]$ oc get route
NAME          PATH          SERVICES    PORT
TERMINATION    WILDCARD
myapp         myapp-kel-webproj.apps.cluster.kel-ocp.com    myapp  8080-tcp
None
```

[user@host ~]\$



The screenshot shows a web browser window with the address bar displaying `myapp-kel-webproj.apps.cluster.kel-ocp.com/index.php`. The page content includes a purple heading **Welcome to My Website**, followed by a section titled **Server Information** containing a table with the following data:

Server Name	myapp-cddc65865-xx9g7
Server IP Address	10.128.1.125

Below the table is a section titled **Environment Variables** with a red error message: **Environment variables MSG and VALUE1 are not set. Unable to display environment variables.**

Next is a section titled **Database Operations** with a button labeled **Generate Data**.

15. (Optional) Create a configmap that holds the 2 variables, MSG and VALUE1. Pass them into the deployment as environment variables.

Lab: Persistent Storage

Lab: Authentication and Authorization

Lab: SA and SCC

Lab: Network Policies

Lab: Quota