Lab: Working with Deployments

Introduction:

Deployments represent a set of multiple, identical Pods with no unique identities. Deployment runs multiple replicas of your application and automatically replaces any instances that fail or become unresponsive.

In this way, Deployments help ensure that one or more instances of your application are available to serve user requests. Deployments are managed by the Kubernetes Deployment controller.

Deployment ensures that only a certain number of Pods are down while they are being updated. By default, it ensures that at least **75%** of the desired number of Pods are up (25% max unavailable).

Objectives:

- **Create a Deployment with Imperative Method**
- **Generate the Deployment manifest in the yml & json format**
- **Create a Deployment with Declarative Method**
- **Exposing Deployment to a Service i.e., ClusterIP**
- **Scale-Up/Down Deployment**
- **Cleanup**

Ensure that you have logged-in as **root** user on **eoc-controller** node.

1. Create a Deployment with Imperative Method

1.1 Let's **create** deployment named **redis-deployment** and using the Redis image.

```
# kubectl create --help
```

```
# kubectl create deployment --help
```

```
# kubectl create deployment redis-deployment --image=redis
```

Output:

```
[root@eoc-controller ~]#kubectl create deployment redis-deployment --image=redis
deployment.apps/redis-deployment created
```

1.2 Let's **check** the details of the deployment by executing the command.

```
# kubectl describe deployment redis-deployment
```

Output:

```
[root@eoc-controller ~]#kubectl describe deployment redis-deployment
Name:                   redis-deployment
Namespace:              default
CreationTimestamp:      Tue, 05 Sep 2023 09:49:06 -0400
Labels:                 app=redis-deployment
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=redis-deployment
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=redis-deployment
  Containers:
   redis:
    Image:        redis
    Port:         <none>
    Host Port:    <none>
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
```

1.3 Let's **list** the pods by executing the below command.

```
# kubectl get pods
```

Output:

```
[root@eoc-controller ~]#kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
redis-deployment-785c545fd6-znv97   1/1     Running   0          5m36s
```

1.4 Let's **delete** the deployment by executing the below command.

```
# kubectl delete deployment redis-deployment
```

Output:

```
[root@eoc-controller ~]#kubectl delete deployment redis-deployment
deployment.apps "redis-deployment" deleted
```

1.5 Let's **create** another deployment of nginx with 4 **replicas.**

```
# kubectl create deployment nginx-deployment --image=nginx \
--replicas=4 --dry-run=client -o yaml | tee nginx-
deployment.yml
```

Output:

```
[root@eoc-controller ~]# kubectl create deployment nginx-deployment --image=nginx \
> --replicas=4 --dry-run=client -o yaml | tee nginxdeployment.yml
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: nginx-deployment
  name: nginx-deployment
spec:
  replicas: 4
  selector:
    matchLabels:
      app: nginx-deployment
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: nginx-deployment
    spec:
      containers:
      - image: nginx
        name: nginx
        resources: {}
status: {}
```

1.6 Let's use the **nginx-deployment.yml** file created from the above step for creating deployment.

```
# kubectl apply -f nginx-deployment.yml
```

Output:

```
[root@eoc-controller ~]#kubectl apply -f nginx-deployment.yml
deployment.apps/nginx-deployment created
```

1.7 Let's **list** the pods by executing the below command.

```
# kubectl get pods
```

Output:

```
[root@eoc-controller ~]#kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-6d6565499c-dmjcf   1/1     Running   0          35s
nginx-deployment-6d6565499c-h9zl5   1/1     Running   0          35s
nginx-deployment-6d6565499c-l5tvp   1/1     Running   0          35s
nginx-deployment-6d6565499c-r2b44   1/1     Running   0          35s
```

1.8 Let's **delete** the deployment and notice that it deletes the pods as well.

```
# kubectl delete -f nginx-deployment.yml
```

Output:

```
[root@eoc-controller ~]#kubectl delete -f nginx-deployment.yml
deployment.apps "nginx-deployment" deleted
```

2. Create a Deployment with Declarative Method

2.1 Let's **view** the yaml manifest file.

```
# cat -n ~/kubernetes/deployment-webserver.yml
```

Output:

```
[root@eoc-controller ~]#cat -n ~/kubernetes/deployment-webserver.yml
     1  apiVersion: apps/v1
     2  kind: Deployment
     3  metadata:
     4    name: web-server
     5  spec:
     6    replicas: 4
     7    selector:
     8      matchLabels:
     9        tier: web-server
    10    template:
    11      metadata:
    12        labels:
    13          tier: web-server
    14      spec:
    15        containers:
    16        - name: web-server-container
    17          image: nginx:1.19
```

2.2 Let's **create** the deployment using the deployment-webserver.yml file.

```
# kubectl apply -f ~/kubernetes/deployment-webserver.yml
```

Output:

```
[root@eoc-controller ~]#kubectl apply -f ~/kubernetes/deployment-webserver.yml
deployment.apps/web-server created
```

2.3 Let's **list** the deployment by executing the below command.

```
# kubectl get deployment
```

Output:

```
[root@eoc-controller ~]#kubectl get deployment
NAME         READY   UP-TO-DATE   AVAILABLE   AGE
web-server   4/4     4            4           25s
```

2.4 Let's list the **labels** automatically generated for each Pod by executing the below command.

```
# kubectl get pods --show-labels
```

Output:

```
[root@eoc-controller ~]#kubectl get pods --show-labels
NAME                          READY   STATUS    RESTARTS   AGE   LABELS
web-server-5dcf74945f-2xnkq   1/1     Running   0          70s   pod-template-hash=5dcf74945f,tier=web-server
web-server-5dcf74945f-7xkrf   1/1     Running   0          70s   pod-template-hash=5dcf74945f,tier=web-server
web-server-5dcf74945f-f5c6g   1/1     Running   0          70s   pod-template-hash=5dcf74945f,tier=web-server
web-server-5dcf74945f-vq2gp   1/1     Running   0          70s   pod-template-hash=5dcf74945f,tier=web-server
```

2.5 Let's **list** the ReplicaSet by executing the below command.

```
# kubectl get rs
```

Output:

```
[root@eoc-controller ~]#kubectl get rs
NAME                    DESIRED   CURRENT   READY   AGE
web-server-5dcf74945f   4         4         4       119s
```

3. Let's **expose** the deployment to the service to access the application.

```
# kubectl expose deployment web-server --name \
demo-service --type ClusterIP --labels app=nginx --port 80
```

Output:

```
[root@eoc-controller ~]#kubectl expose deployment web-server --name \
> demo-service --type ClusterIP --labels app=nginx --port 80
service/demo-service exposed
```

3.1 Let's **list** the service by executing the below command.

```
# kubectl get svc demo-service
```

Output:

```
[root@eoc-controller ~]#kubectl get svc demo-service
NAME           TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
demo-service   ClusterIP   10.96.124.168   <none>        80/TCP    28s
```

3.2 Let's **access** the application using the cluster-ip by executing the below command.

```
# CIP=`kubectl get svc demo-service -o
jsonpath={.spec.clusterIP}`
```

```
# kubectl get svc demo-service -o jsonpath={.spec.clusterIP}
```

```
# curl $CIP
```

```
# echo $CIP
```

Output:

```
[root@eoc-controller ~]#echo $CIP
10.96.124.168
```

Output:

```
[root@eoc-controller ~]#curl $CIP
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

4. Let's **scale up** the deployment by executing the below command.

```
# kubectl scale deployment web-server --replicas=6
```

Output:

```
[root@eoc-controller ~]#kubectl scale deployment web-server --replicas=6
deployment.apps/web-server scaled
```

4.1 Let's **verify** the self-healing property assured by the deployment.

```
# kubectl delete pod --all
```

Output:

```
[root@eoc-controller ~]#kubectl delete pod --all
pod "web-server-5dcf74945f-2xnkq" deleted
pod "web-server-5dcf74945f-54zwt" deleted
pod "web-server-5dcf74945f-7xkrf" deleted
pod "web-server-5dcf74945f-f5c6g" deleted
pod "web-server-5dcf74945f-t79fx" deleted
pod "web-server-5dcf74945f-vq2gp" deleted
```

4.2 Let's **list** the ReplicaSet by executing the below command.

```
# kubectl get rs
```

Output:

```
[root@eoc-controller ~]#kubectl get rs
NAME                    DESIRED   CURRENT   READY   AGE
web-server-5dcf74945f   6         6         6       6m10s
```

4.3 Lets **scale** down to 2 replicas by executing below command.

```
# kubectl scale deployment web-server --replicas=2
```

Output:

```
[root@eoc-kubemaster ~]#kubectl scale deployment web-server --replicas=2
deployment.apps/web-server scaled
```

4.4 Let's **list** the ReplicaSet by executing the below command.

```
# kubectl get rs
```

Output:

```
[root@eoc-controller ~]#kubectl scale deployment web-server --replicas=2
deployment.apps/web-server scaled
```

5 Cleanup

5.1 Let's **delete** the deployment by executing below command.

```
# kubectl delete deployment web-server
```

Output:

```
[root@eoc-controller ~]#kubectl delete deployment web-server
deployment.apps "web-server" deleted
```

5.2 Let's **delete** the service by executing below command.

```
# kubectl delete svc demo-service
```

Output:

```
[root@eoc-controller ~]#kubectl delete svc demo-service
service "demo-service" deleted
```