

## KUBERNETES EXERCISES:

1. Create a pod called web-server with nginx:1.19.10 image

ANS:

```
kubectl run web-server --image=nginx:1.19.10
```

SCREENSHOT:

```
root@controlplane:~# kubectl run web-server --image=nginx:1.19.10
pod/web-server created
root@controlplane:~# kubectl get pods
NAME          READY   STATUS             RESTARTS   AGE
web-server    0/1     ContainerCreating   0           7s
root@controlplane:~# kubectl get pods
NAME          READY   STATUS             RESTARTS   AGE
web-server    0/1     ContainerCreating   0          13s
root@controlplane:~# kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
web-server    1/1     Running   0          48s
root@controlplane:~#
```

2. Expose port 80 of the web-server pod to be reachable within cluster

ANS:

```
kubectl expose pod web-server --type=ClusterIP --port=80 --name=ex2-
service
```

SCREENSHOT:

```
root@controlplane:~# kubectl expose pod web-server --type=ClusterIP --port=80 --name=ex2-service
service/ex2-service exposed
root@controlplane:~# kubectl get services
NAME          TYPE        CLUSTER-IP    EXTERNAL-IP  PORT(S)    AGE
ex2-service   ClusterIP   10.107.216.193 <none>       80/TCP     3s
kubernetes    ClusterIP   10.96.0.1      <none>       443/TCP    17m
```

3. Create a single pod with below images:

i) nginx:1.19.10

ii) redis:6.2.2

ANS:

[pod-definition.yml](#)

apiVersion: v1

kind: Pod

metadata:

name: ex3

spec:

containers:

- name: nginx

image: nginx:1.19.10

- name: redis

image: redis: 6.2.2

[kubectl create -f pod-definition.yml](#)

SCREENSHOTS:

```
root@controlplane:~# vi pod-definition.yml
root@controlplane:~# kubectl create -f pod-definition.yml
pod/ex3 created
root@controlplane:~# kubectl get pods
NAME          READY   STATUS             RESTARTS   AGE
ex3           0/2     ContainerCreating   0           15s
mypod         1/1     Running             0           7m42s
web-server    1/1     Running             0           15m
root@controlplane:~# kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
ex3           2/2     Running   0           24s
mypod         1/1     Running   0           7m51s
web-server    1/1     Running   0           15m
root@controlplane:~# more pod-definition.yml
apiVersion: v1
kind: Pod
metadata:
  name: ex3
spec:
  containers:
  - name: nginx
    image: nginx:1.19.10
  - name: redis
```

4. Expose port 80 and 6379 of the above created pod such that the application can be connected from the outside world using node's IP address

ANS:

[ex4-service.yml](#)

```
apiVersion: v1
```

```
kind: Service
```

```
metadata:
```

```
  name: ex4-service
```

```
spec:
```

```
  selector:
```

```
    app: ex4-pod
```

```
  type: NodePort
```

```
  ports:
```

```
    - name: nginx-service
```

```
      port: 80
```

```
      targetPort: 80
```

```
      nodePort: 30080
```

```
    - name: redis-service
```

```
      port: 6379
```

```
      targetPort: 6379
```

```
      nodePort: 30081
```

[pod-definition.yml](#)

```
apiVersion: v1
```

```
kind: Pod
```

```
metadata:
```

```
  name: ex3
```

```
  labels:
```

```
    app: ex4-pod
```

```
spec:
```

```
  containers:
```

```
    - name: nginx
```

```
      image: nginx:1.19.10
```

```
      ports:
```

- containerPort: 80
- name: redis
- image: redis: 6.2.2
- ports:
- containerPort: 6379

kubect create -f pod-definition.yml

kubect create -f ex4-service.yml

minikube service ex4-service --url

## SCREENSHOTS:

```
service/ex4-service created
$ kubectl get pods -o wide
NAME    READY   STATUS    RESTARTS   AGE   IP           NODE    NOMINATED NODE   READINESS GATES
ex3     2/2     Running   0           79s   172.18.0.4   minikube <none>         <none>
$ kubectl get service -o wide
NAME          TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)                                     AGE   SELECTOR
ex4-service   NodePort    10.110.131.51 <none>        80:30080/TCP,6379:30081/TCP              48s   app=ex4-pod
kubernetes   ClusterIP   10.96.0.1     <none>        443/TCP                                   12m   <none>
$ minikube service ex4-service
|-----|-----|-----|-----|
| NAMESPACE | NAME     | TARGET PORT | URL                                     |
|-----|-----|-----|-----|
| default   | ex4-service | nginx-service | http://172.17.0.45:30080              |
|           |          | redis-service | http://172.17.0.45:30081              |
|-----|-----|-----|-----|
* Opening service default/ex4-service in default browser...
```

```
$ minikube service ex4-service --url
http://172.17.0.45:30080
http://172.17.0.45:30081
$
```

```
$ curl http://172.17.0.61:30080
<!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>
```

5. Create a deployment web-deploy with nginx:1.19.10 image of 2 replica

ANS:

dep-def.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-deploy
spec:
  replicas: 2
  selector:
    matchLabels:
      app: ex5
  template:
    metadata:
      name: ex5-pod
      labels:
        app: ex5
    spec:
      containers:
        - name: nginx
          image: nginx:1.19.10
```

kubectl create -f dep-def.yml

kubectl get deployment web-deploy -o wide

SCREENSHOT:

```
$ kubectl create -f dep-def.yml
deployment.apps/web-deploy created
$ kubectl get deployment web-deploy -o wide
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE	CONTAINERS	IMAGES	SELECTOR
web-deploy	2/2	2	2	30s	nginx	nginx:1.19.10	app=ex5

6. Change the image of web-deploy to nginx:1.20.0 and record the change

ANS:

```
kubectl set image deployment.apps/web-deploy nginx=nginx:1.20.0 --record
```

```
kubectl rollout history deployment.apps/web-deploy
```

SCREENSHOT:

```
$ kubectl set image deployment.apps/web-deploy nginx=nginx:1.20.0 --record
deployment.apps/web-deploy image updated
$ kubectl rollout history deployment.apps/web-deploy
deployment.apps/web-deploy
REVISION  CHANGE-CAUSE
1          <none>
2          kubectl set image deployment.apps/web-deploy nginx=nginx:1.20.0 --record=true
$
```

7. Scale web-deploy to 5 replica

ANS:

```
kubectl scale deployment web-deploy --replicas=5
```

SCREENSHOT:

```
$ kubectl scale deployment web-deploy --replicas=5
deployment.apps/web-deploy scaled
$ kubectl get deployment web-deploy
NAME          READY  UP-TO-DATE  AVAILABLE  AGE
web-deploy    5/5    5           5          14m
$
```

8. Create a persistent volume redis-pv with below specs:

- i) hostpath /mnt/redis/data
- ii) storage size 2Gi
- iii) access mode – ReadWriteOnce

ANS:

[pv-def.yml](#)

apiVersion: v1

kind: PersistentVolume

metadata:

name: redis-pv

spec:

storageClassName: manual

capacity:

storage: 2Gi

accessModes:

- ReadWriteOnce

hostPath:

path: "/mnt/redis/data"

[kubectl create -f pv-def.yml](#)

SCREENSHOT:

```
$ vi pv-def.yml
$ kubectl create -f pv-def.yml
persistentvolume/redis-pv created
$ kubectl get pv
NAME          CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS    CLAIM    STORAGECLASS  REASON  AGE
redis-pv      2Gi       RWO           Retain          Available  manual   manual        13s
$
```

9. Create a persistent volume claim redis-pvc that claims redis-pv persistent volume

ANS:

pvc-def.yml

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: redis-pvc

spec:

storageClassName: manual

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 1Gi

kubectl create -f pvc-def.yml

SCREENSHOT:

```
$ vi pvc-def.yml
$ kubectl create -f pvc-def.yml
persistentvolumeclaim/redis-pvc created
$ kubectl get pv
NAME          CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS  CLAIM          STORAGECLASS  REASON  AGE
redis-pv      2Gi       RWO           Retain          Bound   default/redis-pvc  manual                 2m1s
$ kubectl get pvc
NAME          STATUS    VOLUME    CAPACITY  ACCESS MODES  STORAGECLASS  AGE
redis-pvc     Bound    redis-pv   2Gi       RWO           manual        15s
$
```



10. Create a pod redis which binds the redis-pvc to the path /data with image redis:6.2.2

ANS:

[pod-def.yml](#)

```
apiVersion: v1
```

```
kind: Pod
```

```
metadata:
```

```
  name: redis
```

```
spec:
```

```
  volumes:
```

```
  - name: pv-bind
```

```
    persistentVolumeClaim:
```

```
      claimName: redis-pvc
```

```
  containers:
```

```
  - name: redis
```

```
    image: redis:6.2.2
```

```
    volumeMounts:
```

```
    - name: pv-bind
```

```
      mountPath: "/data"
```

[kubectl create -f pod-def.yml](#)

## SCREENSHOT:

```
$ vi pod-def.yml
$ kubectl create -f pod-def.yml
pod/redis created
$ kubectl describe pod redis
Name:         redis
Namespace:    default
Priority:      0
Node:         minikube/172.17.0.17
```

```
Mounts:
  /data from pv-bind (rw)
  /var/run/secrets/kubernetes.io/serviceaccount from default-token-htqhb (ro)
Conditions:
  Type             Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  pv-bind:
    Type:          PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same namespace)
    ClaimName:     redis-pvc
    ReadOnly:      false
  default-token-htqhb:
    Type:          Secret (a volume populated by a Secret)
```

11. Update the storage size of the redis persistent volume to 3Gi and record the change

ANS:

`vi pv-def.yml`

`apiVersion: v1`

`kind: PersistentVolume`

`metadata:`

`name: redis-pv`

`spec:`

`storageClassName: manual`

`capacity:`

`storage: 3Gi`

`accessModes:`

`- ReadWriteOnce`

`hostPath:`

`path: "/mnt/redis/data"`

`kubectl apply -f pv-def.yml --record`

SCREENSHOTS:

```
$ vi pv-def.yml
$ kubectl apply -f pv-def.yml --record
persistentvolume/redis-pv configured
```

```
$ kubectl get pv redis-pv -o yaml > cause-ex11.yml
$ cat cause-ex11.yml
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","kind":"PersistentVolume","metadata":{"annotations":{"kubernetes.io/change-cause":"kubectl apply --filename=pv-def.yml --record=true"},"name":"redis-pv"},"spec":{"accessModes":["ReadWriteOnce"],"capacity":{"storage":"3Gi"},"hostPath":{"path":"/mnt/redis/data"},"storageClassName":"manual"}}
    kubernetes.io/change-cause: kubectl apply --filename=pv-def.yml --record=true
pv.kubernetes.io/bound-by-controller: "yes"
```

## 12. Create an Ingress for web-deploy deployment with wildcard hostname

ANS:

```
kubectl expose deployment web-deploy --port=80 --type=NodePort
```

[ingress-def.yml](#)

```
apiVersion: networking.k8s.io/v1
```

```
kind: Ingress
```

```
metadata:
```

```
  name: ex5-ingress
```

```
spec:
```

```
  rules:
```

```
  - host: "*.ex5.com"
```

```
    http:
```

```
      paths:
```

```
      - path: /ex5
```

```
        pathType: Prefix
```

```
        backend:
```

```
          service:
```

```
            name: web-deploy
```

```
            port:
```

```
              number: 80
```

```
kubectl create -f ingress-def.yml
```

### SCREENSHOTS:

```
root@controlplane:~# kubectl get deploy web-deploy
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
web-deploy    5/5     5            5           98s
root@controlplane:~# kubectl expose deployment web-deploy --port=80 --type=NodePort
service/web-deploy exposed
root@controlplane:~# kubectl get svc
NAME          TYPE          CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
kubernetes    ClusterIP     10.96.0.1    <none>        443/TCP          9m47s
web-deploy    NodePort      10.104.214.12 <none>        80:31635/TCP     16s
```

```
root@controlplane:~# vi ingress-def.yml
root@controlplane:~# kubectl create -f ingress-def.yml
ingress.networking.k8s.io/ex5-ingres created
```

```
root@controlplane:/etc# kubectl describe service web-deploy
Name: web-deploy
Namespace: default
Labels: <none>
Annotations: <none>
Selector: app=ex5
Type: NodePort
IP Families: <none>
IP: 10.104.214.12
IPs: 10.104.214.12
Port: <unset> 80/TCP
TargetPort: 80/TCP
NodePort: <unset> 31635/TCP
Endpoints: 10.244.0.4:80,10.244.0.5:80,10.244.0.6:80 + 2 more...
Session Affinity: None
External Traffic Policy: Cluster
Events: <none>
root@controlplane:/etc# vi hosts
```

```
root@controlplane:/etc# curl abc.ex5.com
<!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>
root@controlplane:/etc#
```

## Result:

The exercises in kubernetes are successfully executed