

KUBERNETES – EXERCISES

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

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

```
kubectl get pods
```

(or)

pod-def.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: web-server
  labels:
    name: web-server
spec:
  containers:
    - name: nginx
      image: nginx:1.19.10
```

The screenshot shows a browser-based lab environment from Kodekloud. The main area is a terminal window titled "Terminal 1" displaying the following command output:

```
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          8s
root@controlplane:~# kubectl get pods
NAME      READY  STATUS    RESTARTS   AGE
web-server  0/1   ContainerCreating   0          18s
root@controlplane:~# kubectl get pods
NAME      READY  STATUS    RESTARTS   AGE
web-server  1/1   Running   0          38s
root@controlplane:~#
```

To the left of the terminal is a sidebar with the following content:

How many Services exist on the system?
in the current(default) namespace

- 1
- 2
- 0
- 4
- 3

A callout bubble in the bottom right corner of the terminal window says "115x30".

The bottom of the screen shows a Windows taskbar with icons for File Explorer, Edge, and other applications. The system tray shows the date and time as 04-05-2021 19:49.

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

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

```
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          8s
root@controlplane:~# kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
web-server  0/1     ContainerCreating   0          18s
root@controlplane:~# kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
web-server  1/1     Running   0          38s
root@controlplane:~# kubectl expose pod web-server --type=ClusterIP --port=80
service/web-server exposed
root@controlplane:~# kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
web-server  1/1     Running   0          2m12s
root@controlplane:~# kubectl describe pods
```

```
root@controlplane:~# kubectl describe pods
Name:           web-server
Namespace:      default
Priority:      0
Node:          controlplane/10.231.40.6
Start Time:    Tue, 04 May 2021 14:19:03 +0000
Labels:         run=web-server
Annotations:   <none>
Status:        Running
IP:            10.244.0.4
IPs:
  IP: 10.244.0.4
Containers:
  web-server:
    Container ID: docker://8ba8496eb5b0363201b53435c6f9642e4d025833fa1519080dfdb3e4acea8877
    Image:        nginx:1.19.10
    Image ID:    docker-pullable://nginx@sha256:75a55d33ecc73c2a242450a9f1cc858499d468f077ea942867e662c247b5e412
    Port:         <none>
    Host Port:   <none>
    State:       Running
      Started:   Tue, 04 May 2021 14:19:32 +0000
    Ready:       True
    Restart Count: 0
    Environment: <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-rkjs7 (ro)
Conditions:
  Type        Status
  Initialized  True
  Ready       True
```

How many Services exist on the system?
in the current(default) namespace

```
Started: Tue, 04 May 2021 14:19:32 +0000
Ready: True
Restart Count: 0
Environment: <none>
Mounts:
    /var/run/secrets/kubernetes.io/serviceaccount from default-token-rkjs7 (ro)
Conditions:
  Type      Status
  Initialized  True
  Ready       True
  ContainersReady  True
  PodScheduled  True
Volumes:
  default-token-rkjs7:
    Type:  Secret (a volume populated by a Secret)
    SecretName: default-token-rkjs7
    Optional:  false
    QoS Class: BestEffort
    Node-Selectors: <none>
    Tolerations: node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
               node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason   Age   From           Message
  ----   -----   --   --   -----
  Normal  Scheduled 2m29s default-scheduler  Successfully assigned default/web-server to controlplane
  Normal  Pulling   2m26s kubelet        Pulling image "nginx:1.19.10"
  Normal  Pulled   2m   kubelet        Successfully pulled image "nginx:1.19.10" in 26.743199304s
  Normal  Created   119s kubelet        Created container web-server
  Normal  Started   119s kubelet        Started container web-server
root@controlplane:~#
```

How many Services exist on the system?
in the current(default) namespace

```
root@controlplane:~# kubectl get svc
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
kubernetes  ClusterIP  10.96.0.1      <none>          443/TCP     8m55s
web-server  ClusterIP  10.105.17.59  <none>          80/TCP      75s
root@controlplane:~# kubectl describe svc
Name:            kubernetes
Namespace:       default
Labels:          component=apiserver
                 provider=kubernetes
Annotations:    <none>
Selector:        <none>
Type:            ClusterIP
IP Families:   <none>
IP:              10.96.0.1
IPs:             10.96.0.1
Port:            https 443/TCP
TargetPort:      6443/TCP
Endpoints:      10.231.40.6:6443
Session Affinity: None
Events:          <none>

Name:            web-server
Namespace:       default
Labels:          run=web-server
Annotations:    <none>
Selector:        run=web-server
Type:            ClusterIP
IP Families:   <none>
IP:              10.105.17.59
```

The screenshot shows a browser-based lab environment from Kodekloud. The URL is <https://kodekloud.com/courses/1120660/lectures/24008540>. The main area displays a terminal window titled "Terminal 1" with the following output:

```
Labels: component=apiserver provider=kubernetes
Annotations: <none>
Selector: <none>
Type: ClusterIP
IP Families: <none>
IP: 10.96.0.1
IPs: 10.96.0.1
Port: https 443/TCP
TargetPort: 6443/TCP
Endpoints: 10.231.40.6:6443
Session Affinity: None
Events: <none>

Name: web-server
Namespace: default
Labels: run=web-server
Annotations: <none>
Selector: run=web-server
Type: ClusterIP
IP Families: <none>
IP: 10.105.17.59
IPs: 10.105.17.59
Port: <unset> 80/TCP
TargetPort: 80/TCP
Endpoints: 10.244.0.4:80
Session Affinity: None
Events: <none>
root@controlplane:~#
```

3) Create a single pod with below images:

- i) nginx:1.19.10
- ii) redis:6.2.2

pod-define.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
    - name: nginx
      image: nginx:1.19.10
      ports:
        containerPort: 3000
    name: redis
    image: nginx:6.2.2
    ports:
      containerPort: 3001
```

The screenshot shows a browser-based interface for a Kubernetes tutorial. On the left, there's a sidebar with 'Task' and 'Hint' buttons, and a timer set to 39:36. Below the timer is a question: "How many Services exist on the system? in the current(default) namespace". Five options are listed: 1, 0, 4, 3, and 2. On the right is a terminal window titled "Terminal 1" showing the following YAML configuration:

```
apiVersion: v1
kind: Pod
metadata:
  name: myfirstpod
  labels:
    app: myfirstpod
spec:
  containers:
    - name: nginx
      image: nginx:1.19.10
      ports:
        - containerPort: 3000
    - name: redis
      image: redis:6.2.5
      ports:
        - containerPort: 3001
```

The terminal also shows the command "pod-define.yaml" and its file statistics: 17L, 372C. The bottom of the screen shows a Windows taskbar with various icons and a system tray indicating the date as 04-05-2021 and time as 15:02.

This screenshot is similar to the one above, showing the same browser-based interface. The terminal window now displays the results of running commands on a control plane host:

```
root@controlplane:~# vim pod-define.yaml
root@controlplane:~# kubectl apply -f pod-define.yaml
pod/myfirstpod unchanged
root@controlplane:~# kubectl get pods
NAME     READY   STATUS    RESTARTS   AGE
myfirstpod 2/2     Running   0          14m
root@controlplane:~#
```

The question and options remain the same as in the first screenshot. The Windows taskbar at the bottom shows the date as 04-05-2021 and time as 15:03.

- 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

pod-define.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: myfirstpod
  labels:
    app: myfirstpod
spec:
  containers:
    - name: nginx
      image: nginx:1.19.10
      ports:
        - containerPort: 3000
    - name: redis
      image: nginx:6.2.2
      ports:
        - containerPort: 3001
```

pod-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: myfirstapp-service
  labels:
    name: myservice
spec:
  type: NodePort
  selector:
    name: myfirstpod
  ports:
    - name: nginx
      protocol: TCP
      port: 80
      targetPort: 3000
      nodePort: 30000
    - name: redis
      protocol: TCP
      port: 6379
      targetPort: 3001
      nodePort: 30001
```

The screenshot shows a browser-based Kubernetes lab interface. In the center is a terminal window titled "Terminal 1". The terminal displays a YAML configuration file named "pod-service.yaml". The file defines a Service object with the following details:

```
apiVersion: v1
kind: Service
metadata:
  name: myfirstapp-service
  labels:
    name: myservice
spec:
  type: NodePort
  selector:
    name: myfirstpod
  ports:
    - name: nginx
      protocol: TCP
      port: 80
      targetPort: 3000
      nodePort: 30000
    - name: redis
      protocol: TCP
      port: 6379
      targetPort: 3001
      nodePort: 30001
```

Below the terminal, a "Task" section asks: "How many Services exist on the system? in the current(default) namespace". The answer choices are 1, 0, 4, 3, and 2. The number 1 is highlighted in blue.

The screenshot shows a browser-based Kubernetes lab interface. In the center is a terminal window titled "Terminal 1". The terminal displays command-line output from a root user on a control plane node:

```
root@controlplane:~# vim pod-service.yaml
root@controlplane:~# kubectl apply -f pod-service.yaml
service/myfirstapp-service created
root@controlplane:~# kubectl get svc
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)
kubernetes     ClusterIP  10.96.0.1   <none>        443/TCP
myfirstapp-service  NodePort  10.97.67.181  <none>        80:30000/TCP,6379:30001/TCP   12s
root@controlplane:~# kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
myfirstpod  2/2     Running   0          19m
root@controlplane:~#
```

Below the terminal, a "Task" section asks: "How many Services exist on the system? in the current(default) namespace". The answer choices are 1, 0, 4, 3, and 2. The number 1 is highlighted in blue.

The screenshot shows a browser-based Kubernetes lab interface. On the left, there's a sidebar with tabs for 'Task' (selected), 'Hint', and a timer set to 16:25. Below the sidebar, a list of numbers (1, 0, 4, 3, 2) is displayed. The main area is a terminal window titled 'Terminal 1'. The terminal output shows the following commands and their results:

```
root@controlplane:~# kubectl apply -f pod-define.yaml
pod/myfirstpod unchanged
root@controlplane:~# kubectl apply -f pod-service.yaml
service/myfirstapp-service unchanged
root@controlplane:~# kubectl get all
NAME           READY   STATUS    RESTARTS   AGE
pod/myfirstpod 2/2     Running   0          36m

NAME              TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)      AGE
service/kubernetes ClusterIP  <none>       443/TCP      47m
service/myfirstapp-service NodePort  10.96.0.1  <none>       80:30000/TCP,6379:30001/TCP  17m
root@controlplane:~#
```

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

deploy-def.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myfirstdeploy
  labels:
    app: myfirst-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: myfirst-app
  template:
    metadata:
      labels:
        app: myfirst-app
    containers:
      - name: nginx
        image: nginx:1.19.10
        ports:
          - containerPort: 80
```

Task Hint ⏳ 32:09

How many Services exist on the system? in the current(default) namespace

- 3
- 1
- 0
- 4
- 2

Terminal 1

```
root@controlplane:~# kubectl create -f deploy-def.yaml
deployment.apps/myfirstdeploy created
root@controlplane:~# kubectl get deploy -l app=myfirst-app
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
myfirstdeploy  2/2     2            2           38s
root@controlplane:~# kubectl get rs -l app=myfirst-app
NAME           DESIRED   CURRENT   READY   AGE
myfirstdeploy-68d796bb8c  2         2         2         54s
root@controlplane:~# kubectl get pod -l app=myfirst-app
NAME           READY   STATUS    RESTARTS   AGE
myfirstdeploy-68d796bb8c-fj9tt  1/1     Running   0          78s
myfirstdeploy-68d796bb8c-m214w  1/1     Running   0          78s
root@controlplane:~#
```

Complete and Continue →

Windows Taskbar: Type here to search, Start button, File Explorer, Edge, File Manager, Task View, Taskbar icons, Date: 04-05-2021, Time: 17:23

Task Hint ⏳ 31:17

How many Services exist on the system? in the current(default) namespace

- 3
- 1
- 0
- 4
- 2

Terminal 1

```
myfirstdeploy-68d796bb8c  2         2         2         54s
root@controlplane:~# kubectl get pod -l app=myfirst-app
NAME           READY   STATUS    RESTARTS   AGE
myfirstdeploy-68d796bb8c-fj9tt  1/1     Running   0          78s
myfirstdeploy-68d796bb8c-m214w  1/1     Running   0          78s
root@controlplane:~# kubectl describe deploy myfirstdeploy
Name:           myfirstdeploy
Namespace:      default
CreationTimestamp:  Tue, 04 May 2021 11:51:44 +0000
Labels:          app=myfirst-app
Annotations:    deployment.kubernetes.io/revision: 1
Selector:        app=myfirst-app
Replicas:       2 desired | 2 updated | 2 total | 2 available | 0 unavailable
StrategyType:   RollingUpdate
MinReadySeconds: 0
RollingUpdateStrategy: 25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=myfirst-app
  Containers:
    nginx:
      Image:  nginx:1.19.10
      Port:   80/TCP
      Host Port:  0/TCP
      Environment: <none>
      Mounts:   <none>
      Volumes:  <none>
  Conditions:
    Type      Status  Reason
    ----      ----   -----
    Available  True    MinimumReplicasAvailable
```

Complete and Continue →

Windows Taskbar: Type here to search, Start button, File Explorer, Edge, File Manager, Task View, Taskbar icons, Date: 04-05-2021, Time: 17:24

```

CreationTimestamp: Tue, 04 May 2021 11:51:44 +0000
Labels: app=myfirst-app
Annotations: deployment.kubernetes.io/revision: 1
Selector: app=myfirst-app
Replicas: 2 desired | 2 updated | 2 total | 2 available | 0 unavailable
StrategyType: RollingUpdate
MinReadySeconds: 0
RollingUpdateStrategy: 25% max unavailable, 25% max surge
Pod Template:
  Labels: app=myfirst-app
  Containers:
    nginx:
      Image: nginx:1.19.10
      Port: 80/TCP
      Host Port: 0/TCP
      Environment: <none>
      Mounts: <none>
      Volumes: <none>
  Conditions:
    Type Status Reason
    ----
    Available True   MinimumReplicasAvailable
    Progressing True  NewReplicaSetAvailable
    OldReplicaSets: <none>
    NewReplicaSet: myfirstdeploy-68d796bb8c (2/2 replicas created)
  Events:
    Type Reason Age From Message
    ----
    Normal ScalingReplicaSet 2m12s deployment-controller Scaled up replica set myfirstdeploy-68d796bb8c to 2
root@controlplane:~# 

```

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

`kubectl set image deploy myfirstdeploy nginx=nginx:1.20.0 --record`

```

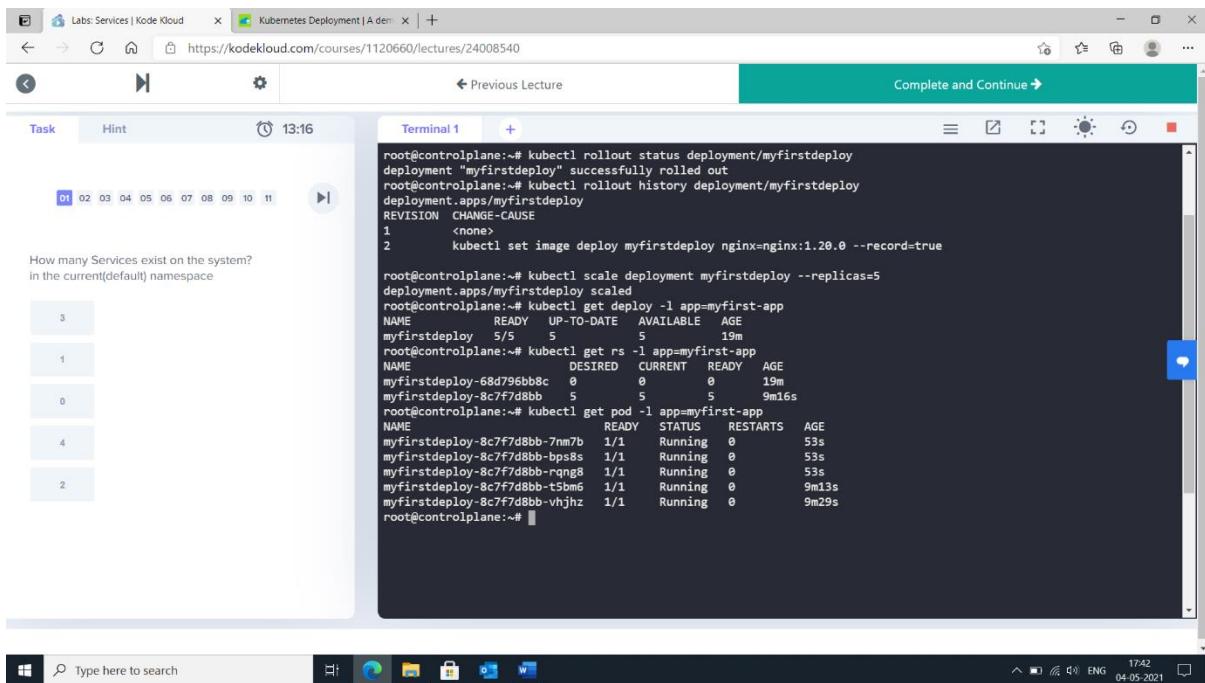
root@controlplane:~# kubectl rollout status deployment/myfirstdeploy
deployment "myfirstdeploy" successfully rolled out
root@controlplane:~# kubectl rollout history deployment/myfirstdeploy
deployment.apps/myfirstdeploy
REVISION CHANGE-CAUSE
1 <none>
2 kubectl set image deploy myfirstdeploy nginx=nginx:1.20.0 --record=true

root@controlplane:~# kubectl scale deployment myfirstdeploy --replicas=5
deployment.apps/myfirstdeploy scaled
root@controlplane:~# kubectl get deploy -l app=myfirst-app
NAME READY UP-TO-DATE AVAILABLE AGE
myfirstdeploy 5/5 5 5 19m
root@controlplane:~# kubectl get rs -l app=myfirst-app
NAME DESIRED CURRENT READY AGE
myfirstdeploy-68d796bb8c 0 0 0 19m
myfirstdeploy-8c7f7d8bb 5 5 5 9m16s
root@controlplane:~# kubectl get pod -l app=myfirst-app
NAME READY STATUS RESTARTS AGE
myfirstdeploy-8c7f7d8bb-7nm7b 1/1 Running 0 53s
myfirstdeploy-8c7f7d8bb-bps8s 1/1 Running 0 53s
myfirstdeploy-8c7f7d8bb-rqng8 1/1 Running 0 53s
myfirstdeploy-8c7f7d8bb-t5bm6 1/1 Running 0 9m13s
myfirstdeploy-8c7f7d8bb-vhjhz 1/1 Running 0 9m29s
root@controlplane:~# 

```

7) Scale web-deploy to 5 replica

```
kubectl scale deployment myfirstdeploy --replicas=5
```



The screenshot shows a browser window with a tab titled "Kubernetes Deployment | A demo". The URL is <https://kodekloud.com/courses/1120660/lectures/24008540>. The main area contains a terminal window with the following command history:

```
root@controlplane:~# kubectl rollout status deployment/myfirstdeploy
deployment "myfirstdeploy" successfully rolled out
root@controlplane:~# kubectl rollout history deployment/myfirstdeploy
deployment.apps/myfirstdeploy
REVISION  CHANGE-CAUSE
1          <none>
2          kubectl set image deploy myfirstdeploy nginx=nginx:1.20.0 --record=true
root@controlplane:~# kubectl scale deployment myfirstdeploy --replicas=5
deployment.apps/myfirstdeploy scaled
root@controlplane:~# kubectl get deploy -l app=myfirst-app
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
myfirstdeploy  5/5     5           5           19m
root@controlplane:~# kubectl get rs -l app=myfirst-app
NAME           DESIRED  CURRENT   READY   AGE
myfirstdeploy-68d796bb8c  0        0        0       19m
myfirstdeploy-8c7f7d8bb  5        5        5       9m16s
root@controlplane:~# kubectl get pod -l app=myfirst-app
NAME           READY   STATUS    RESTARTS   AGE
myfirstdeploy-8c7f7d8bb-7nm7b  1/1   Running   0          53s
myfirstdeploy-8c7f7d8bb-bps8s  1/1   Running   0          53s
myfirstdeploy-8c7f7d8bb-rqng8  1/1   Running   0          53s
myfirstdeploy-8c7f7d8bb-t5bm6  1/1   Running   0          9m13s
myfirstdeploy-8c7f7d8bb-vhjhz  1/1   Running   0          9m29s
root@controlplane:~#
```

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

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

redispv.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: redis-pv
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 2Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/mnt/redis/data"
```

```
kubectl apply -f redispv.yaml
```

```
kubectl get pv redis-pv
```

The screenshot shows a browser-based Kubernetes lab interface. On the left, there's a sidebar with a 'Task' tab, a 'Hint' tab, and a timer set to 48:11. A question asks, "How many Services exist on the system? in the current(namespace)". Below the question are five numbered options: 3, 0, 2, 4, and 1. On the right, a terminal window titled 'Terminal 1' displays the YAML configuration for a PersistentVolume named 'redis-pv'. The configuration specifies a capacity of 2Gi and a hostPath of '/mnt/redis/data'. The terminal also shows the file being saved as 'redispv.yaml'. At the bottom of the terminal, it says '10,14 All'.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: redis-pv
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 2Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/mnt/redis/data"
"redispv.yaml" 14L, 317C
```

The screenshot shows a browser-based Kubernetes lab interface. On the left, there's a sidebar with a 'Task' tab, a 'Hint' tab, and a timer set to 47:43. A question asks, "How many Services exist on the system? in the current(namespace)". Below the question are five numbered options: 3, 0, 2, 4, and 1. On the right, a terminal window titled 'Terminal 1' shows the command 'kubectl apply -f redispv.yaml' being run, followed by the output of 'kubectl get pv'. The output shows a PersistentVolume named 'redis-pv' with a capacity of 2Gi, access mode RWO, reclaim policy Retain, and status Available. It is associated with a StorageClass named 'manual'. The terminal prompt ends with 'root@controlplane:~#'. At the bottom of the terminal, it says '1853 ENG 04-05-2021'.

```
root@controlplane:~# kubectl apply -f redispv.yaml
persistentvolume/redis-pv created
root@controlplane:~# vim redispv.yaml
root@controlplane:~# kubectl get pv
NAME      CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM   STORAGECLASS   REASON   AGE
redis-pv   2Gi        RWO          Retain           Available   manual
root@controlplane:~#
```

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

redis-pvc.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: redis-pvc
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

kubectl apply -f redis-pvc.yaml

kubectl get pv redis-pv

kubectl get pvc redis-pvc

The screenshot shows a terminal window titled "Terminal 1" displaying a YAML configuration for a PersistentVolumeClaim named "redis-pvc". The configuration specifies a storage class of "manual" and access modes of "ReadWriteOnce". It also includes requests for 1Gi of storage. The terminal window is part of a larger interface with a taskbar, a hint section, and a navigation bar.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: redis-pvc
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

The screenshot shows a terminal window titled "Terminal 1" displaying the output of two Kubernetes commands: "kubectl get pv" and "kubectl get pvc". The "kubectl get pv" command shows a PersistentVolume named "redis-pv" with a capacity of 2Gi and an RWO access mode. The "kubectl get pvc" command shows a PersistentVolumeClaim named "redis-pvc" with a status of Bound, a volume of "redis-pv", and an RWO access mode. Both commands indicate a storage class of "manual" and an age of 5m56s. The terminal window is part of a larger interface with a taskbar, a hint section, and a navigation bar.

```
root@controlplane:~# kubectl get pv redis-pv
NAME      CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM           STORAGECLASS   REASON   AGE
redis-pv   2Gi        RWO          Retain          Bound    default/redis-pvc   manual        5m56s
root@controlplane:~# kubectl get pvc redis-pvc
NAME      STATUS   VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   AGE
redis-pvc   Bound   redis-pv   2Gi        RWO          manual        80s
root@controlplane:~#
```

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

pod-def.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: redis
spec:
  volumes:
    - name: redis-storage
      persistentVolumeClaim:
        claimName: redis-pvc
  containers:
    - name: redispv-container
      image: redis:6.2.2
      ports:
        - containerPort: 80
          name: redis
  volumeMounts:
    - mountPath: "/data"
      name: redis-storage
```

The screenshot shows a web browser window for a Kubernetes lab. The URL is <https://kodekloud.com/courses/1120660/lectures/24008540>. The title bar includes tabs for "Configure a Pod to Use a PersistentVolume" and "Kubernetes for the Absolute Beginner". The main area has a "Task" tab, a "Hint" tab, and a terminal window. The terminal shows the following command output:

```
root@controlplane:~# kubectl apply -f pod-def.yaml
pod/redis created
root@controlplane:~# kubectl get pod redis
NAME     READY   STATUS    RESTARTS   AGE
redis   1/1     Running   0          69s
root@controlplane:~#
```

To the left of the terminal, there is a question: "How many Services exist on the system? in the current(default) namespace". Below the question is a list of five options: 3, 0, 2, 4, and 1. The option "3" is highlighted with a blue background.

The screenshot shows a web browser window for a Kubernetes lab, identical to the one above but with a different terminal command. The URL is the same: <https://kodekloud.com/courses/1120660/lectures/24008540>. The terminal shows:

```
root@controlplane:/mnt/redis/data# touch index.html
root@controlplane:/mnt/redis/data# vim index.html
root@controlplane:/mnt/redis/data# ls
index.html
root@controlplane:/mnt/redis/data# cat index.html
Hello Friends, This is Sakthi Sri
root@controlplane:/mnt/redis/data#
```

To the left of the terminal, there is a question: "How many Services exist on the system? in the current(default) namespace". Below the question is a list of five options: 3, 0, 2, 4, and 1. The option "3" is highlighted with a blue background.

The screenshot shows a browser window with two tabs: "Configure a Pod to Use a Persistent Volume" and "Kubernetes for the Absolute Beginner". The main content area has a "Terminal 1" tab open. The terminal window title is "Labs: Services | Kode Kloud". The terminal content shows the following command output:

```
root@controlplane:~# pwd
/root
root@controlplane:~# ls
pod-def.yaml redispv.yaml redispvc.yaml service-definition-1.yaml
root@controlplane:~# kubectl apply -f redispv.yaml
persistentvolume/redis-pv unchanged
root@controlplane:~# kubectl apply -f redispvc.yaml
persistentvolumeclaim/redis-pvc unchanged
root@controlplane:~# kubectl apply -f pod-def.yaml
pod/redis configured
root@controlplane:~# kubectl get pod redis
NAME READY STATUS RESTARTS AGE
redis 1/1 Running 0 11m
root@controlplane:~# kubectl get pv redis-pv
NAME CAPACITY ACCESS MODES RECLAIM POLICY STATUS CLAIM STORAGECLASS REASON AGE
redis-pv 2Gi RWO Retain Bound default/redis-pvc manual 23m
root@controlplane:~# kubectl get pvc redis-pvc
NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS AGE
redis-pvc Bound redis-pv 2Gi RWO manual 18m
root@controlplane:~#
```

To the left of the terminal window, there is a sidebar with a "Task" tab, a "Hint" tab, and a timer showing "24:58". Below the timer is a numbered list from 01 to 11. At the bottom of the sidebar, there is a question: "How many Services exist on the system? in the current(default) namespace".

The screenshot shows a browser window with two tabs: "Configure a Pod to Use a Persistent Volume" and "Kubernetes for the Absolute Beginner". The main content area has a "Terminal 1" tab open. The terminal window title is "Labs: Services | Kode Kloud". The terminal content shows the following command output:

```
root@controlplane:~# kubectl exec -it redis -- /bin/bash
root@redis:/# cat index.html
Hello Friends, This is Sakthi Sri
root@redis:/# ls
index.html
root@redis:/data# ls
index.html
root@redis:/data# exit
exit
root@controlplane:~# ls
pod-def.yaml redispv.yaml redispvc.yaml service-definition-1.yaml
root@controlplane:~#
```

To the left of the terminal window, there is a sidebar with a "Task" tab, a "Hint" tab, and a timer showing "22:31". Below the timer is a numbered list from 01 to 11. At the bottom of the sidebar, there is a question: "How many Services exist on the system? in the current(default) namespace".

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

`kubectl edit pv redis-pv`

```

apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","kind":"PersistentVolume","metadata":{"annotations":{},"labels":{"type":"local"},"name":"redis-pv"},"spec":{"accessModes":["ReadWriteOnce"],"capacity":{"storage":"2Gi"},"hostPath":{"path":"/mnt/redis/data"}}, "storageClassName":"manual"}
    pv.kubernetes.io/bound-by-controller: "yes"
  creationTimestamp: "2021-05-04T13:22:55Z"
  finalizers:
  - kubernetes.io/pv-protection
  labels:
    type: local
    name: redis-pv
  resourceVersion: "3526"
  uid: 650bd81c-775d-4573-b899-63ea2801fdac
spec:
  accessModes:
  - RReadWriteOnce
  capacity:
    storage: 2Gi
  claimRef:
  apiVersion: v1
  kind: PersistentVolumeClaim
  "/tmp/kubectl-edit-uvf9x.yaml" 39L, 1254C

```

```

apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","kind":"PersistentVolume","metadata":{"annotations":{},"labels":{"type":"local"},"name":"redis-pv"},"spec":{"accessModes":["ReadWriteOnce"],"capacity":{"storage":"2Gi"},"hostPath":{"path":"/mnt/redis/data"}}, "storageClassName":"manual"}
    pv.kubernetes.io/bound-by-controller: "yes"
  creationTimestamp: "2021-05-04T13:22:55Z"
  finalizers:
  - kubernetes.io/pv-protection
  labels:
    type: local
    name: redis-pv
  resourceVersion: "3526"
  uid: 650bd81c-775d-4573-b899-63ea2801fdac
spec:
  accessModes:
  - RReadWriteOnce
  capacity:
    storage: 3Gi
  claimRef:
  apiVersion: v1
  kind: PersistentVolumeClaim
  name: redis-pvc
  namespace: default
  resourceVersion: "1731"
  uid: 278d28e2-5b23-4403-bce2-c92df79075ef
  hostPath:
    path: /mnt/redis/data
    type: ""
:wq

```

The screenshot shows a web-based lab environment from Kodekloud. At the top, there are tabs for "Kubernetes for the Absolute Beginner", "Labs: Services | Kode Kloud", and "Resizing Persistent Volumes using". The main area has a "Terminal 1" tab open, displaying the following command output:

```
root@controlplane:~# kubectl edit pv redis-pv --record
persistentvolume/redis-pv edited
root@controlplane:~# kubectl get pv redis-pv
NAME      CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM      STORAGECLASS   REASON   AGE
redis-pv   3Gi        RWO          Retain          Bound    default/redis-pvc   manual   30m
root@controlplane:~# kubectl get pvc redis-pvc
NAME      STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   AGE
redis-pvc Bound    redis-pv  2Gi        RWO          manual       26m
root@controlplane:~# kubectl get pod redis
NAME    READY   STATUS    RESTARTS   AGE
redis  1/1    Running   0          19m
root@controlplane:~#
```

To the left of the terminal, there is a sidebar titled "Task" with a "Hint" section containing the question: "How many Services exist on the system? in the current(default) namespace". Below this are five numbered options: 3, 0, 2, 4, and 1. The option "3" is highlighted with a blue border.

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

deploy-def.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-deploy
  labels:
    name: web-deploy
spec:
  selector:
    matchLabels:
      name: web-server
  replicas: 2
  template:
    metadata:
      name: web-server
      labels:
        name: web-server
    spec:
      containers:
        - name: redis
          image: redis:6.2.2
```

service-def.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: web-service
  labels:
    name: web-service
spec:
  ports:
  - port: 80
    targetPort: 8080
  selector:
    name: web-server
```

ingress-def.yaml

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: web-ingress
  labels:
    name: web-ingress
spec:
  rules:
  - http:
      paths:
      - pathType: Prefix
        path: "/"
        backend:
          service:
            name: web-service
            port:
              number: 8080
```

How many Services exist on the system?
in the current(default) namespace

Task Hint ⏱ 54:12

Terminal 1 +

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-deploy
  labels:
    name: web-deploy
spec:
  selector:
    matchLabels:
      name: web-server
  replicas: 2
  template:
    metadata:
      name: web-server
      labels:
        name: web-server
    spec:
      containers:
        - name: redis
          image: redis:6.2.2
```

"deploy-def.yaml" 22L, 589C 20,24 All

How many Services exist on the system?
in the current(default) namespace

Task Hint ⏱ 51:10

Terminal 1 +

```
root@controlplane:~# vim deploy-def.yaml
root@controlplane:~# kubectl apply -f deploy-def.yaml
deployment.apps/web-deploy created
root@controlplane:~# vim deploy-def.yaml
root@controlplane:~# vim service-def.yaml
root@controlplane:~# kubectl apply -f service-def.yaml
service/web-service created
root@controlplane:~#
```



How many Services exist on the system?
in the current(default) namespace

Task Hint

01 02 03 04 05 06 07 08 09 10 11

```
Terminal 1
apiVersion: v1
kind: Service
metadata:
  name: web-service
  labels:
    name: web-service
spec:
  ports:
    - port: 80
      targetPort: 8080
  selector:
    name: web-service
```

12,32 All

Type here to search

17:17 05-05-2021

How many Services exist on the system?
in the current(default) namespace

Task Hint

01 02 03 04 05 06 07 08 09 10 11

```
Terminal 1
root@controlplane:~# vim ingress-def.yaml
root@controlplane:~# kubectl apply -f ingress-def.yaml
ingress.networking.k8s.io/web-ingress created
root@controlplane:~# kubectl get ingress web-ingress
NAME      CLASS      HOSTS      ADDRESS      PORTS      AGE
web-ingress  <none>    *          80          118s
root@controlplane:~# kubectl describe ingress web-ingress
Name:           web-ingress
Namespace:      default
Address:
Default backend: default-http-backend:80 (<error: endpoints "default-http-backend" not found>)
Rules:
Host      Path      Backends
----      ---      -----
*          /          web-service:8080 (10.244.0.4:8080,10.244.0.5:8080)
Annotations: <none>
Events:    <none>
root@controlplane:~# kubectl get svc
NAME        TYPE        CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
kubernetes  ClusterIP  10.96.0.1      <none>          443/TCP      18m
web-service  ClusterIP  10.108.54.45   <none>          80/TCP       7m45s
root@controlplane:~# kubectl get deploy
NAME        READY     UP-TO-DATE   AVAILABLE   AGE
web-deploy  2/2      2           2           11m
root@controlplane:~# kubectl get pods
NAME        READY     STATUS    RESTARTS   AGE
web-deploy-7dc7cbcfb8-ns9gf  1/1      Running   0          11m
web-deploy-7dc7cbcfb8-vdtwz  1/1      Running   0          11m
root@controlplane:~#
```

Type here to search

17:25 05-05-2021

The screenshot shows a Windows desktop environment. In the center is a terminal window titled "Terminal 1" displaying the output of a "kubectl get all" command. The output lists pods, services, and replicaset information. To the left of the terminal is a sidebar with a "Task" tab and a "Hint" tab. The "Hint" tab contains the question "How many Services exist on the system? in the current(default) namespace" and five numbered options (0, 1, 2, 3, 4). The number 3 is highlighted. At the bottom of the screen is a taskbar with various icons and a search bar.

```
root@controlplane:~# kubectl get all
NAME                           READY   STATUS    RESTARTS   AGE
pod/web-deploy-7dc7cbcf8-ns9gf  1/1    Running   0          11m
pod/web-deploy-7dc7cbcf8-vdtwz  1/1    Running   0          11m

NAME              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   19m
service/web-service ClusterIP   10.108.54.45  <none>        88/TCP    8m14s

NAME           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/web-deploy  2/2     2           2           11m

NAME           DESIRED  CURRENT  READY   AGE
replicaset.apps/web-deploy-7dc7cbcf8  2        2        2       11m
root@controlplane:~#
```