Lab Exercise 13- Managing Namespaces in Kubernetes

Name – Vishal Pandey
500125280
B2 DevOps

Step 1: Understand Namespaces

Namespaces provide a mechanism for scoping resources in a cluster. Namespaces can be used to:

- Create environments for different applications or teams.
- Apply policies like resource quotas or network policies on a per-namespace basis.
- Separate operational environments (like development and production).

Step 2: List Existing Namespaces

To list all the namespaces in your Kubernetes cluster:

```
kubectl get namespaces
```

```
PS C:\Users\ASUS> kubectl get namespaces
NAME                   STATUS   AGE
default                Active   25m
kube-node-lease        Active   25m
kube-public            Active   25m
kube-system            Active   25m
kubernetes-dashboard   Active   21m
PS C:\Users\ASUS>
```

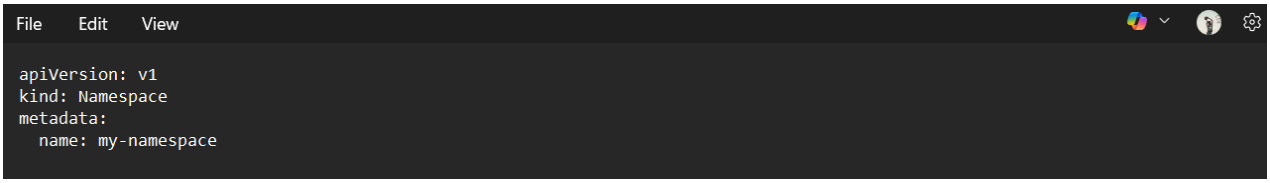
You will typically see default namespaces like default, kube-system, and kube-public.

Step 3: Create a Namespace

You can create a namespace using a YAML file or directly with the kubectl command.

Using YAML File
Create a file named my-namespace.yaml with the following content:

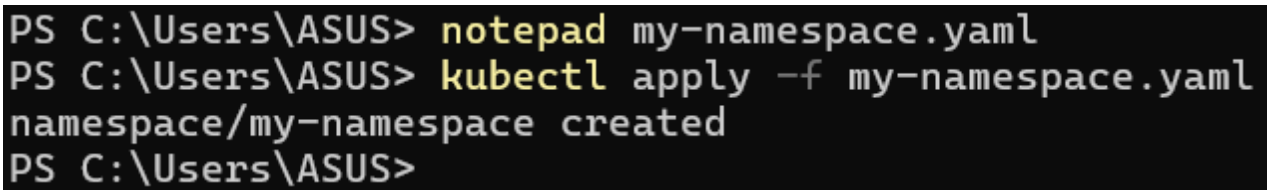
```
apiVersion: v1
kind: Namespace
metadata:
  name: my-namespace
```



Apply this YAML to create the namespace:

```
kubectl apply -f my-namespace.yaml
```

Using kubectl Command



Alternatively, create a namespace using the kubectl command:

```
kubectl create namespace my-namespace
```

Verify that the namespace is created:

```
kubectl get namespaces
```

```
PS C:\Users\ASUS> kubectl get namespaces
NAME                   STATUS   AGE
default                Active   27m
kube-node-lease        Active   27m
kube-public            Active   27m
kube-system            Active   27m
kubernetes-dashboard   Active   22m
my-namespace           Active   19s
PS C:\Users\ASUS>
```

You should see my-namespace listed in the output.

Step 4: Deploy Resources in a Namespace

Create resources such as Pods, Services, or Deployments within the new namespace.

Deploy a Pod in the Namespace

Create a YAML file named nginx-pod.yaml with the following content:

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
  namespace: my-namespace
spec:
  containers:
  - name: nginx
```

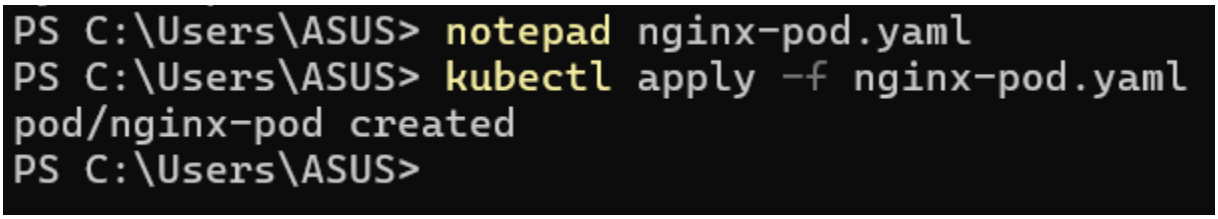
```
  image: nginx:latest
  ports:
  - containerPort: 80
```

File Edit View

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
  namespace: my-namespace
spec:
  containers:
  - name: nginx
    image: nginx:latest
    ports:
    - containerPort: 80
```

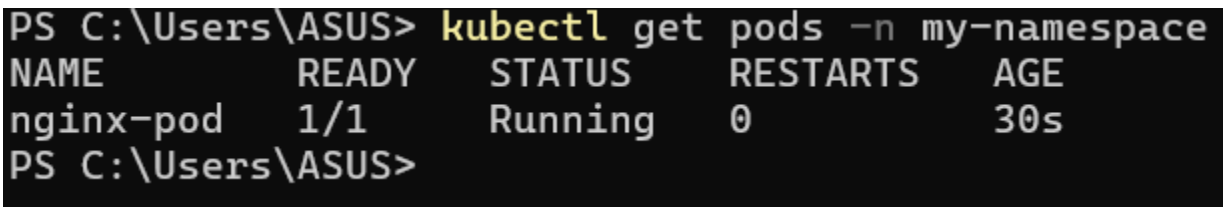
Apply this YAML to create the Pod:

```
kubectl apply -f nginx-pod.yaml
```



Check the status of the Pod within the namespace:

```
kubectl get pods -n my-namespace
```



To describe the Pod and see detailed information:

kubectl describe pod nginx-pod -n my-namespace

```
PS C:\Users\ASUS> kubectl describe pod nginx-pod -n my-namespace
Name:             nginx-pod
Namespace:        my-namespace
Priority:         0
Service Account:  default
Node:             docker-desktop/192.168.65.3
Start Time:       Sun, 22 Feb 2026 14:49:59 +0530
Labels:           <none>
Annotations:      <none>
Status:           Running
IP:               10.1.0.8
IPs:
  IP:  10.1.0.8
Containers:
  nginx:
    Container ID:   docker://169c0acf5f00c77f71fbe0bc7b9516c42347e56cf342c5d
c9256f7c94c61b95c
    Image:          nginx:latest
    Image ID:       docker-pullable://nginx@sha256:341bf0f3ce6c5277d6002cf6e
1fb0319fa4252add24ab6a0e262e0056d313208
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Sun, 22 Feb 2026 14:50:16 +0530
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-2xv
gd (ro)
Conditions:
  Type                        Status
  PodReadyToStartContainers   True
  Initialized                 True
  Ready                       True
  ContainersReady             True
  PodScheduled                True
Volumes:
  kube-api-access-2xvgd:
    Type:                    Projected (a volume that contains injected data
 from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    Optional:                false
    DownwardAPI:             true
QoS Class:                   BestEffort
```

```
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exist
s for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exi
sts for 300s
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  45s   default-scheduler  Successfully assigned my-names
pace/nginx-pod to docker-desktop
  Normal  Pulling    45s   kubelet            Pulling image "nginx:latest"
  Normal  Pulled     29s   kubelet            Successfully pulled image "ngi
nx:latest" in 15.931s (15.931s including waiting). Image size: 62939286 byte
s.
  Normal  Created    28s   kubelet            Created container: nginx
  Normal  Started    28s   kubelet            Started container nginx
PS C:\Users\ASUS>
```

Create a Service in the Namespace

Create a YAML file named **nginx-service.yaml** with the following content:

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
  namespace: my-namespace
spec:
  selector:
    app: nginx-pod
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
  type: ClusterIP
```

File Edit View

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
  namespace: my-namespace
spec:
  selector:
    app: nginx-pod
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
  type: ClusterIP
```

Apply this YAML to create the Service:

```
kubectl apply -f nginx-service.yaml
```

```
PS C:\Users\ASUS> notepad nginx-service.yaml
PS C:\Users\ASUS> kubectl apply -f nginx-service.yaml
service/nginx-service created
PS C:\Users\ASUS>
```

Check the status of the Service within the namespace:

```
kubectl get services -n my-namespace
```

```
PS C:\Users\ASUS> kubectl get services -n my-namespace
NAME            TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
nginx-service   ClusterIP   10.108.146.94   <none>        80/TCP    32s
PS C:\Users\ASUS>
```

To describe the Service and see detailed information:

```
kubectl describe service nginx-service -n my-namespace
```

```
PS C:\Users\ASUS> kubectl describe service nginx-service -n my-namespace
Name:                     nginx-service
Namespace:                my-namespace
Labels:                   <none>
Annotations:              <none>
Selector:                 app=nginx-pod
Type:                     ClusterIP
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.108.146.94
IPs:                      10.108.146.94
Port:                     <unset>  80/TCP
TargetPort:               80/TCP
Endpoints:
Session Affinity:         None
Internal Traffic Policy:  Cluster
Events:                   <none>
PS C:\Users\ASUS>
```

Step 5: Switching Context Between Namespaces

When working with multiple namespaces, you can specify the namespace in kubectl commands or switch the default context.

Specify Namespace in Commands

You can specify the namespace directly in kubectl commands using the -n or --namespace flag:

```
kubectl get pods -n my-namespace
```

```
PS C:\Users\ASUS> kubectl get pods -n my-namespace
NAME        READY   STATUS    RESTARTS   AGE
nginx-pod   1/1     Running   0          3m41s
PS C:\Users\ASUS>
```

Set Default Namespace for kubectl Commands

To avoid specifying the namespace every time, you can set the default namespace for the current context:

```
kubectl config set-context --current --namespace=my-namespace
```

```
PS C:\Users\ASUS> kubectl config set-context --current --namespace=my-namesp
ace
Context "docker-desktop" modified.
PS C:\Users\ASUS>
```

Verify the current context's namespace:

```
kubectl config view --minify | grep namespace
```

```
PS C:\Users\ASUS> kubectl config view --minify
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: DATA+OMITTED
    server: https://kubernetes.docker.internal:6443
  name: docker-desktop
contexts:
- context:
    cluster: docker-desktop
    namespace: my-namespace
    user: docker-desktop
  name: docker-desktop
current-context: docker-desktop
kind: Config
users:
- name: docker-desktop
  user:
    client-certificate-data: DATA+OMITTED
    client-key-data: DATA+OMITTED
PS C:\Users\ASUS>
```

Step 6: Clean Up Resources

To delete the resources and the namespace you created:

```
kubectl delete -f nginx-pod.yaml
kubectl delete -f nginx-service.yaml
kubectl delete namespace my-namespace
```

```
PS C:\Users\ASUS> kubectl delete -f nginx-pod.yaml
pod "nginx-pod" deleted from my-namespace namespace
PS C:\Users\ASUS> kubectl delete -f nginx-service.yaml
service "nginx-service" deleted from my-namespace namespace
PS C:\Users\ASUS> kubectl delete namespace my-namespace
namespace "my-namespace" deleted
```

Ensure that the namespace and all its resources are deleted:

```
kubectl get namespaces
```

```
PS C:\Users\ASUS> kubectl get namespaces
NAME                   STATUS   AGE
default                Active   33m
kube-node-lease        Active   33m
kube-public            Active   33m
kube-system            Active   33m
kubernetes-dashboard   Active   28m
PS C:\Users\ASUS>
```