Lab Exercise 13- Managing Namespaces in Kubernetes

Name-Misha

SAP ID- 500119679

Batch-2

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

```
D:\kubs>kubectl get namespaces
NAME                   STATUS   AGE
default                Active   20m
kube-node-lease        Active   20m
kube-public            Active   20m
kube-system            Active   20m
kubernetes-dashboard   Active   12m
local-path-storage     Active   20m
```

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

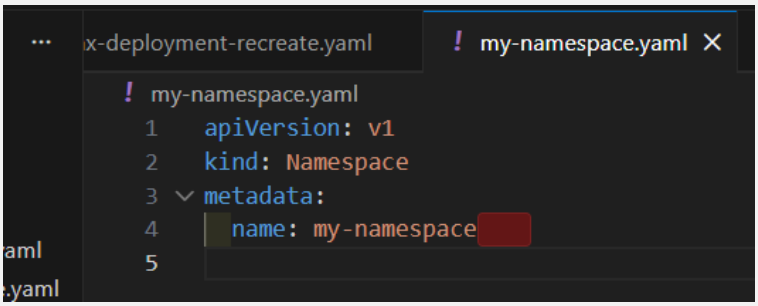
Step 3: Create a Namespace

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
```

```
D:\kubs>kubectl apply -f my-namespace.yaml
namespace/my-namespace created
```

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
```

```
D:\kubs>kubectl get namespaces
NAME                   STATUS   AGE
default                Active   22m
kube-node-lease        Active   22m
kube-public            Active   22m
kube-system            Active   22m
kubernetes-dashboard   Active   15m
local-path-storage     Active   22m
my-namespace           Active   36s
```

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
```

Apply this YAML to create the Pod:

kubectl apply -f nginx-pod.yaml

```
D:\kubs>kubectl apply -f nginx-pod.yaml
pod/nginx-pod created
```

Check the status of the Pod within the namespace:

kubectl get pods -n my-namespace

```
D:\kubs>kubectl get pods -n my-namespace
NAME        READY   STATUS    RESTARTS   AGE
nginx-pod   1/1     Running   0          16s
```

To describe the Pod and see detailed information:

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

```
D:\kubs>kubectl describe pod nginx-pod -n my-namespace
Name:             nginx-pod
Namespace:        my-namespace
Priority:         0
Service Account:  default
Node:             desktop-control-plane/172.19.0.2
Start Time:       Mon, 16 Feb 2026 23:10:14 +0530
Labels:           <none>
Annotations:      <none>
Status:           Running
IP:               10.244.0.7
IPs:
  IP:  10.244.0.7
Containers:
  nginx:
    Container ID:   containerd://b4c0aa0e6bd56fd73f46a6184784d1f948f38831bc8018338660487419bb82e8
    Image:          nginx:latest
    Image ID:       docker.io/library/nginx@sha256:c881927c4077710ac4b1da63b83aa163937fb47457950c267d92f7e4dedf4aec
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Mon, 16 Feb 2026 23:10:19 +0530
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-lxgbx (ro)
Conditions:
  Type                        Status
  PodReadyToStartContainers   True
  Initialized                 True
  Ready                       True
  ContainersReady             True
  PodScheduled                True
Volumes:
  kube-api-access-lxgbx:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
```


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
```

Apply this YAML to create the Service:

kubectl apply -f nginx-service.yaml

```
D:\kubs>kubectl apply -f nginx-service.yaml
service/nginx-service created
```

Check the status of the Service within the namespace:

kubectl get services -n my-namespace

```
D:\kubs>kubectl get services -n my-namespace
NAME            TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)   AGE
nginx-service   ClusterIP   10.96.83.89   <none>        80/TCP    20s
```

To describe the Service and see detailed information:

kubectl describe service nginx-service -n my-namespace

```
D:\kubs>kubectl get services -n my-namespace
NAME            TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)   AGE
nginx-service   ClusterIP   10.96.83.89   <none>        80/TCP    20s

D:\kubs>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:                       10.96.83.89
IPs:                      10.96.83.89
Port:                     <unset>  80/TCP
TargetPort:               80/TCP
Endpoints:
Session Affinity:         None
Internal Traffic Policy:  Cluster
Events:                   <none>
```

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

```
D:\kubs>kubectl get pods -n my-namespace
NAME        READY   STATUS    RESTARTS   AGE
nginx-pod   1/1     Running   0          3m
```

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

```
D:\kubs>kubectl config set-context --current --namespace=my-namespace
Context "docker-desktop" modified.
```

Verify the current context's namespace:

kubectl config view --minify | grep namespace

v]

```
D:\kubs>kubectl config view --minify | findstr namespace
    namespace: my-namespace
```

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

```
D:\kubs>kubectl delete -f nginx-pod.yaml
pod "nginx-pod" deleted from my-namespace namespace

D:\kubs>kubectl delete -f nginx-service.yaml
service "nginx-service" deleted from my-namespace namespace

D:\kubs>kubectl delete namespace my-namespace
namespace "my-namespace" deleted
```

Ensure that the namespace and all its resources are deleted:

kubectl get namespaces

```
D:\kubs>kubectl get namespaces
NAME                   STATUS   AGE
default                Active   30m
kube-node-lease        Active   30m
kube-public            Active   30m
kube-system            Active   30m
kubernetes-dashboard   Active   22m
local-path-storage     Active   30m

D:\kubs>
```