Lab Exercise 13- Managing Namespaces in Kubernetes

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
```

```
devanksilswal@devanks-MacBook-Air ex_s % kubectl get namespaces
NAME                   STATUS   AGE
default                Active   39m
kube-node-lease        Active   39m
kube-public            Active   39m
kube-system            Active   39m
kubernetes-dashboard   Active   33m
```

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

Step 3: Create a Namespace

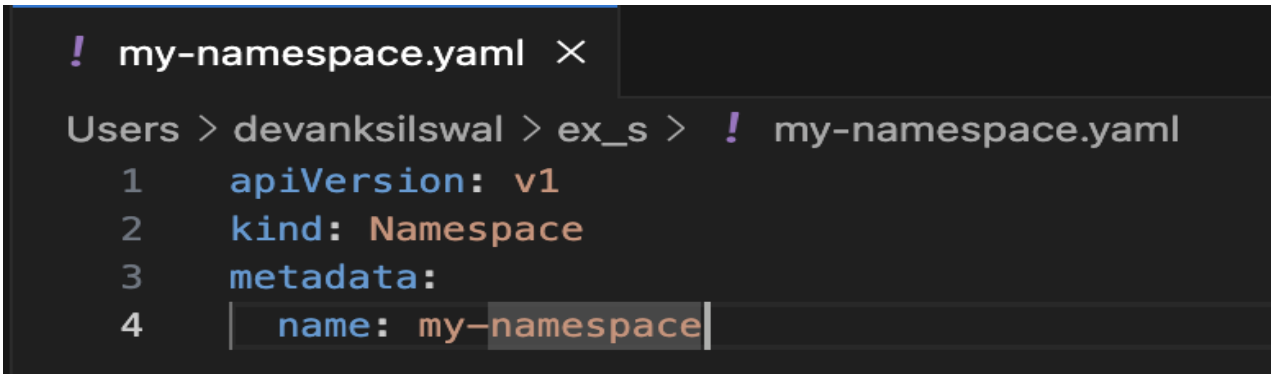
You can create a namespace using a YAML file or directly with the kubectl command.

Using YAML File

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

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

```
devanksilswal@devanks-MacBook-Air ex_s % touch my-namespace.yaml
```



Apply this YAML to create the namespace:

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

Using kubectl Command

```
devanksilswal@devanks-MacBook-Air ex_s % kubectl apply -f my-namespace.yaml
namespace/my-namespace created
```

Alternatively, create a namespace using the kubectl command:

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

```
devanksilswal@devanks-MacBook-Air ex_s % kubectl create namespace my-namespace
Error from server (AlreadyExists): namespaces "my-namespace" already exists
```

Verify that the namespace is created:

```
kubectl get namespaces
```

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
```

```
devanksilswal@devanks-MacBook-Air ex_s % touch nginx-pod.yaml
```

Users > devanksilswal > ex_s > ! nginx-pod.yaml

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

Apply this YAML to create the Pod:

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

```
devanksilswal@devanks-MacBook-Air ex_s % 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
```

```
devanksilswal@devanks-MacBook-Air ex_s % kubectl get pods -n my-namespace
NAME        READY   STATUS    RESTARTS   AGE
nginx-pod   1/1     Running   0          115s
```

To describe the Pod and see detailed information:

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

```
devanksilswal@devanks-MacBook-Air ex_s % 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:       Tue, 24 Feb 2026 20:14:24 +0530
Labels:           app=nginx
Annotations:      <none>
Status:           Running
IP:               10.1.0.20
IPs:
  IP:  10.1.0.20
Containers:
  nginx:
    Container ID:   docker://247dfd47a429394bdc6f5adf9b7ad49509f19930a785009da2b17fa5450f256b
    Image:          nginx:latest
    Image ID:       docker-pullable://nginx@sha256:341bf0f3ce6c5277d6002cf6e1fb0319fa4252add24ab6a0e262e0056d313208
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Tue, 24 Feb 2026 20:14:43 +0530
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-45f2v (ro)
Conditions:
  Type                        Status
  PodReadyToStartContainers   True
  Initialized                 True
  Ready                       True
  ContainersReady             True
  PodScheduled                True
Volumes:
  kube-api-access-45f2v:
    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=Exists for 300s
                              node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason     Age    From               Message
  ----    ------     ----   ----               -------
  Normal  Scheduled  2m26s  default-scheduler  Successfully assigned my-namespace/nginx-pod to docker-desktop
  Normal  Pulling    2m26s  kubelet            spec.containers{nginx}: Pulling image "nginx:latest"
  Normal  Pulled     2m7s   kubelet            spec.containers{nginx}: Successfully pulled image "nginx:latest" in 18.1
78s (18.178s including waiting). Image size: 61268012 bytes.
  Normal  Created    2m7s   kubelet            spec.containers{nginx}: Created container: nginx
  Normal  Started    2m7s   kubelet            spec.containers{nginx}: Started container nginx
```

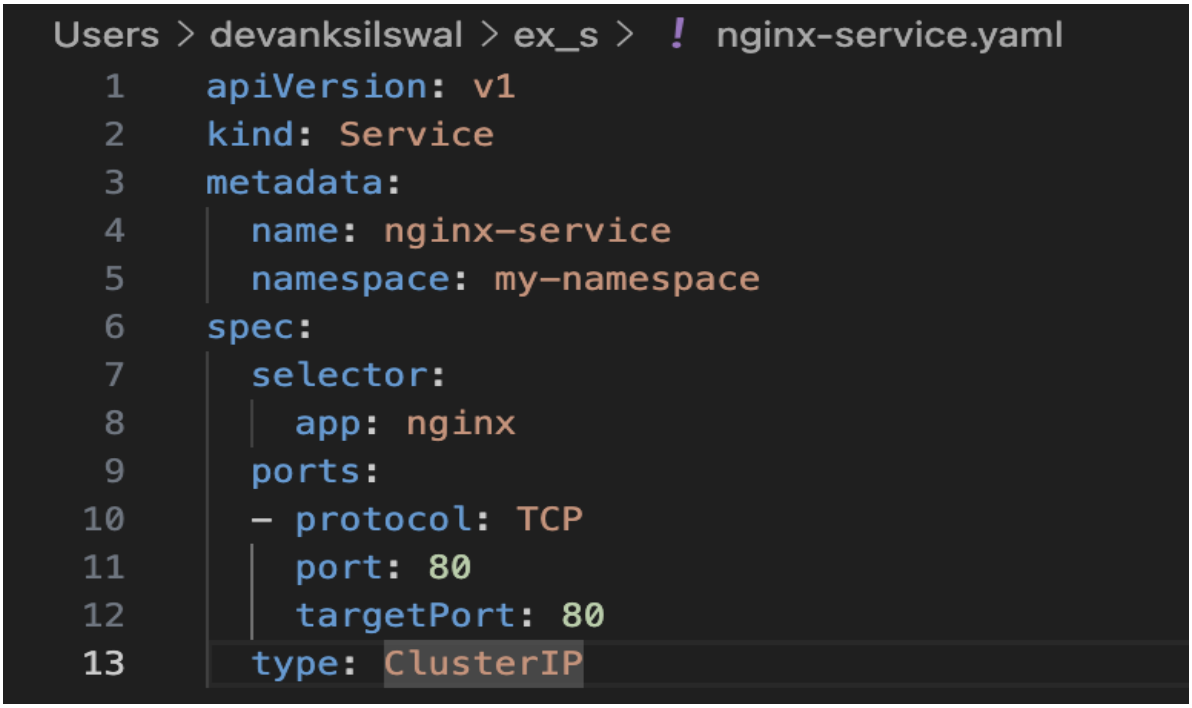
Create a Service in the Namespace

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
```

```
devanksilswal@devanks-MacBook-Air ex_s % touch nginx-service.yaml
```



Apply this YAML to create the Service:

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

```
devanksilswal@devanks-MacBook-Air ex_s % 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
```

```
devanksilswal@devanks-MacBook-Air ex_s % kubectl get services -n my-namespace
NAME            TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE
nginx-service   ClusterIP   10.96.228.27   <none>        80/TCP    11m
```

To describe the Service and see detailed information:

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

```
devanksilswal@devanks-MacBook-Air ex_s % kubectl describe service nginx-service -n my-namespace
Name:                     nginx-service
Namespace:                my-namespace
Labels:                   <none>
Annotations:              <none>
Selector:                 app=nginx
Type:                     ClusterIP
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.96.228.27
IPs:                      10.96.228.27
Port:                     <unset>  80/TCP
TargetPort:               80/TCP
Endpoints:                10.1.0.20:80
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
```

```
devanksilswal@devanks-MacBook-Air ex_s % kubectl get pods -n my-namespace
NAME        READY   STATUS    RESTARTS   AGE
nginx-pod   1/1     Running   0          15m
```

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
```

```
devanksilswal@devanks-MacBook-Air ex_s % kubectl config set-context --current --namespace=my-namespace
Context "docker-desktop" modified.
```

Verify the current context's namespace:

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

```
devanksilswal@devanks-MacBook-Air ex_s % kubectl config view --minify | grep 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
```

```
devanksilswal@devanks-MacBook-Air ex_s % kubectl delete -f nginx-pod.yaml
pod "nginx-pod" deleted from my-namespace namespace
devanksilswal@devanks-MacBook-Air ex_s % kubectl delete -f nginx-service.yaml
service "nginx-service" deleted from my-namespace namespace
devanksilswal@devanks-MacBook-Air ex_s % kubectl delete namespace my-namespace
namespace "my-namespace" deleted
```

Ensure that the namespace and all its resources are deleted:

```
kubectl get namespaces
```

```
devanksilswal@devanks-MacBook-Air ex_s % kubectl get namespaces
NAME                   STATUS   AGE
default                Active   61m
kube-node-lease        Active   61m
kube-public            Active   61m
kube-system            Active   61m
kubernetes-dashboard   Active   55m
```