Lab Exercise 9- 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
```

```
sai@Sais-Mac K8S % kubectl get ns
NAME              STATUS   AGE
default           Active   16m
kube-node-lease   Active   16m
kube-public       Active   16m
kube-system       Active   16m
```

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

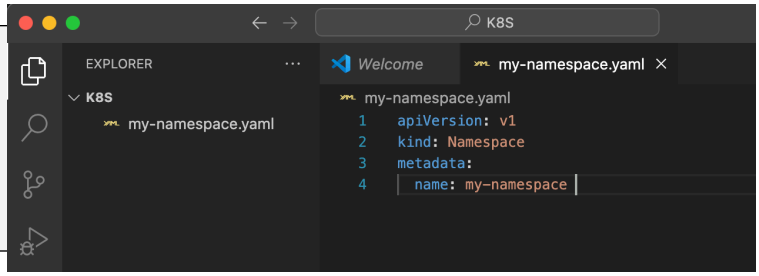
Step 3: Create a Namespace

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
```

```
sai@Sais-Mac K8S % kubectl apply -f my-namespace.yaml
namespace/my-namespace created
```

Verify that the namespace is created:

```
kubectl get namespaces
```

```
sai@Sais-Mac K8S % kubectl get ns
NAME              STATUS   AGE
default           Active   16m
kube-node-lease   Active   16m
kube-public       Active   16m
kube-system       Active   16m
my-namespace      Active   3m23s
```

You should see my-namespace listed in the output.

Step 4: Deploy Resources in a Namespace

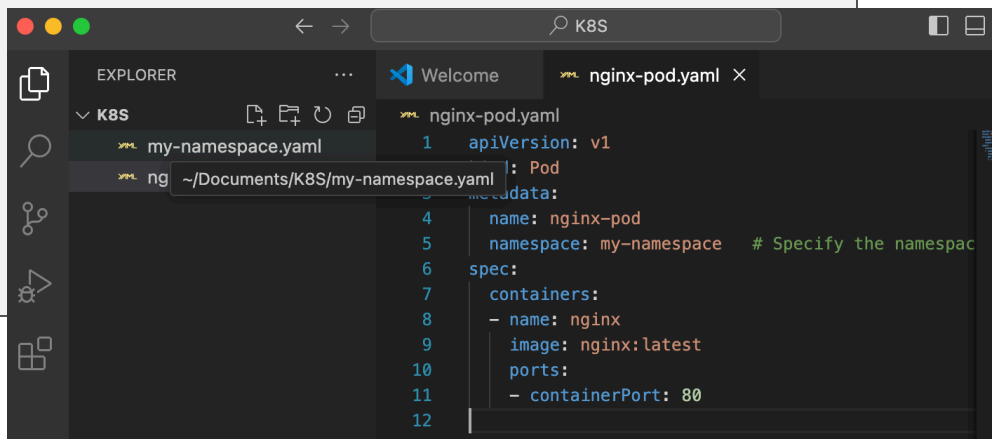
Create resources such as Pods, Services, or Deployments within the new namespace.

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   # Specify the namespace for the Pod.
spec:
  containers:
  - name: nginx
    image: nginx:latest
    ports:
    - containerPort: 80
```



Apply this YAML to create the Pod:

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

```
sai@Sais-Mac K8S % 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
```

```
sai@Sais-Mac K8S % kubectl get pods -n my-namespace
NAME        READY   STATUS    RESTARTS   AGE
nginx-pod   1/1     Running   0          2m37s
```

To describe the Pod and see detailed information:

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

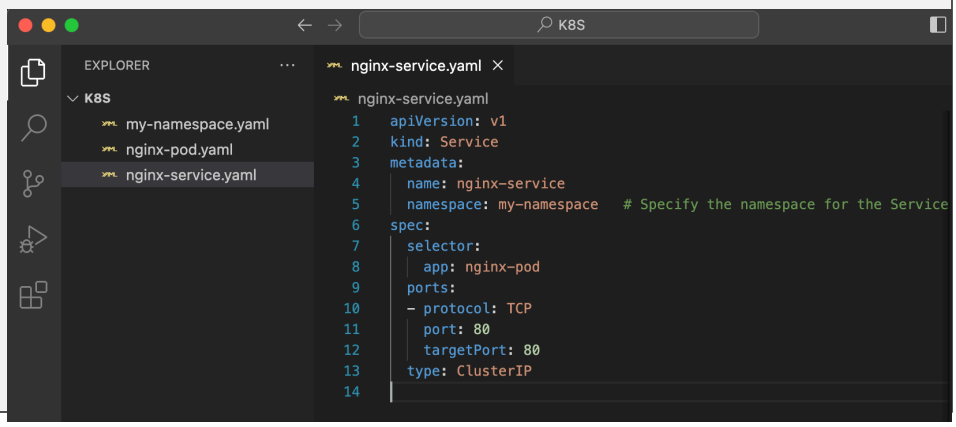
```
sai@Sais-Mac K8S % kubectl describe pod nginx-pod -n my-namespace
Name:             nginx-pod
Namespace:        my-namespace
Priority:         0
Service Account:  default
Node:             docker-desktop/192.168.65.3
Start Time:       Mon, 11 Nov 2024 11:38:34 +0530
Labels:           <none>
Annotations:      <none>
Status:           Running
IP:               10.1.0.6
IPs:
  IP:  10.1.0.6
Containers:
  nginx:
    Container ID:   docker://0c40f772b4b2e1f88450d5844606c2c886276e714239b4adcb8ffc36b44c5257
    Image:          nginx:latest
    Image ID:       docker-pullable://nginx@sha256:28402db69fec7c17e179ea87882667f1e054391138f77ffaf0c3eb388efc3ffb
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Mon, 11 Nov 2024 11:38:48 +0530
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-26z57 (ro)
Conditions:
  Type                        Status
  PodReadyToStartContainers   True
  Initialized                 True
  Ready                       True
  ContainersReady             True
  PodScheduled                True
Volumes:
  kube-api-access-26z57:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    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  4m21s  default-scheduler  Successfully assigned my-namespace/nginx-pod to docker-desktop
  Normal  Pulling    4m21s  kubelet            Pulling image "nginx:latest"
  Normal  Pulled     4m7s   kubelet            Successfully pulled image "nginx:latest" in 13.718s (13.719s including waiting). Image size: 196880357 bytes.
  Normal  Created    4m7s   kubelet            Created container nginx
  Normal  Started    4m7s   kubelet            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   # Specify the namespace for the Service.
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
```

```
sai@Sais-Mac K8S % 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
```

```
sai@Sais-Mac K8S % kubectl get services -n my-namespace
NAME            TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
nginx-service   ClusterIP   10.96.117.130   <none>        80/TCP    39s
```

To describe the Service and see detailed information:

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

```
sai@Sais-Mac K8S % 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:                10.96.117.130
IPs:               10.96.117.130
Port:              <unset>  80/TCP
TargetPort:        80/TCP
Endpoints:         <none>
Session Affinity:  None
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
```

```
sai@Sais-Mac K8S % kubectl get pods -n my-namespace
NAME        READY   STATUS    RESTARTS   AGE
nginx-pod   1/1     Running   0          10m
```

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
```

```
sai@Sais-Mac K8S % kubectl config set-context --current --namespace=my-namespace

Context "docker-desktop" modified.
```

Verify the current context's namespace:

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

```
sai@Sais-Mac K8S % kubectl config view --minify | grep namespace:
    namespace: my-namespace
```

Don't forget to set it to default namespace:

```
sai@Sais-Mac K8S % kubectl config set-context --current --namespace=default

Context "docker-desktop" modified.
```

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
```

```
sai@Sais-Mac K8S % kubectl delete -f nginx-pod.yaml
kubectl delete -f nginx-service.yaml
kubectl delete namespace my-namespace
pod "nginx-pod" deleted
service "nginx-service" deleted
namespace "my-namespace" deleted
```

Ensure that the namespace and all its resources are deleted:

```
kubectl get namespaces
```

```
sai@Sais-Mac K8S % kubectl get namespaces

NAME              STATUS   AGE
default           Active   39m
kube-node-lease   Active   39m
kube-public       Active   39m
kube-system       Active   39m
```