

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
```

```
C:\Users\pk294>kubectl get namespaces
NAME                STATUS    AGE
default             Active    31d
kube-node-lease     Active    31d
kube-public         Active    31d
kube-system         Active    31d
```

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
```

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

Apply this YAML to create the namespace:

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

```
C:\Users\pk294>kubectl apply -f my-namespace.yaml
namespace/my-namespace created
```

Verify that the namespace is created:

```
kubectl get namespaces
```

```
C:\Users\pk294>kubectl get namespaces
NAME                STATUS    AGE
default             Active    31d
kube-node-lease     Active    31d
kube-public         Active    31d
kube-system         Active    31d
my-namespace        Active    56s
```

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
```

```
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
```

```
C:\Users\pk294>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
```

```
C:\Users\pk294>kubectl get pods -n my-namespace
NAME          READY   STATUS    RESTARTS   AGE
nginx-pod     1/1     Running   0           48s
```

To describe the Pod and see detailed information:

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

```
C:\Users\pk294>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:    Thu, 21 Nov 2024 20:19:43 +0530
Labels:        app=nginx
Annotations:    <none>
Status:        Running
IP:            10.1.0.20
IPs:
  IP: 10.1.0.20
Containers:
  nginx:
    Container ID:  docker://f8580b7929a36f075b56bd9073d8ea19734e63b37c895703e6e0f04907236b
    Image:         nginx:latest
    Image ID:      docker-pullable://nginx@sha256:bc5eac5eafc581aeda3008b4b1f07ebba230de2f27d47767129a6a905c84f470
    Port:          80/TCP
    Host Port:     0/TCP
    State:         Running
      Started:     Thu, 21 Nov 2024 20:19:46 +0530
    Ready:         True
    Restart Count: 0
    Environment:   <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-q2zgn (ro)
Conditions:
  Type                               Status
  PodReadyToStartContainers         True
  Initialized                       True
  Ready                             True
  ContainersReady                   True
  PodScheduled                      True
Volumes:
  kube-api-access-q2zgn:
    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 # Specify the namespace for the Service.
spec:
  selector:
    app: nginx-pod
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

```
type: ClusterIP
```

```
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
```

```
C:\Users\pk294>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
```

```
C:\Users\pk294>kubectl get pods -n my-namespace
NAME          READY   STATUS    RESTARTS   AGE
nginx-pod     1/1     Running   0           5m
```

To describe the Service and see detailed information:

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

```
C:\Users\pk294>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:                 10.99.146.189
IPs:                10.99.146.189
Port:               <unset> 80/TCP
TargetPort:         80/TCP
Endpoints:          10.1.0.20:80
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
```

```
C:\Users\iamyo>kubectl get pods -n my-namespace
NAME          READY   STATUS    RESTARTS   AGE
nginx-pod     1/1     Running   0           4m4s
```

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
```

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

Verify the current context's namespace:

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

```
C:\Users\iamyo>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
```

```
C:\Users\iamyo>kubectl delete -f nginx-pod.yaml  
pod "nginx-pod" deleted
```

```
C:\Users\iamyo>kubectl delete -f nginx-service.yaml  
service "nginx-service" deleted
```

```
C:\Users\iamyo>kubectl delete namespace my-namespace  
namespace "my-namespace" deleted
```

Ensure that the namespace and all its resources are deleted:

```
kubectl get namespaces
```

```
C:\Users\iamyo>kubectl get namespaces
```

NAME	STATUS	AGE
default	Active	21d
kube-node-lease	Active	21d
kube-public	Active	21d
kube-system	Active	21d