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

```
PS C:\Github Repositores\CDS-LAB-SUBMISSION-2022-26\R2142221383> kubectl get namespaces
NAME              STATUS   AGE
default           Active   21d
kube-node-lease   Active   21d
kube-public       Active   21d
kube-system       Active   21d
```

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
```

my-namespace.yaml

```
1  apiVersion: v1
2  kind: Namespace
3  metadata:
4    name: my-namespace
5
```

Apply this YAML to create the namespace:

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

```
PS C:\Github Repositores\CDS-LAB-SUBMISSION-2022-26\R2142221383> kubectl apply -f my-namespace.yaml
namespace/my-namespace created
```

Verify that the namespace is created:

```
kubectl get namespaces
```

```
PS C:\Github Repositores\CDS-LAB-SUBMISSION-2022-26\R2142221383> kubectl get namespaces
NAME              STATUS   AGE
default           Active   21d
kube-node-lease   Active   21d
kube-public       Active   21d
kube-system       Active   21d
my-namespace      Active   21s
```

You should see my-namespace listed in the output.

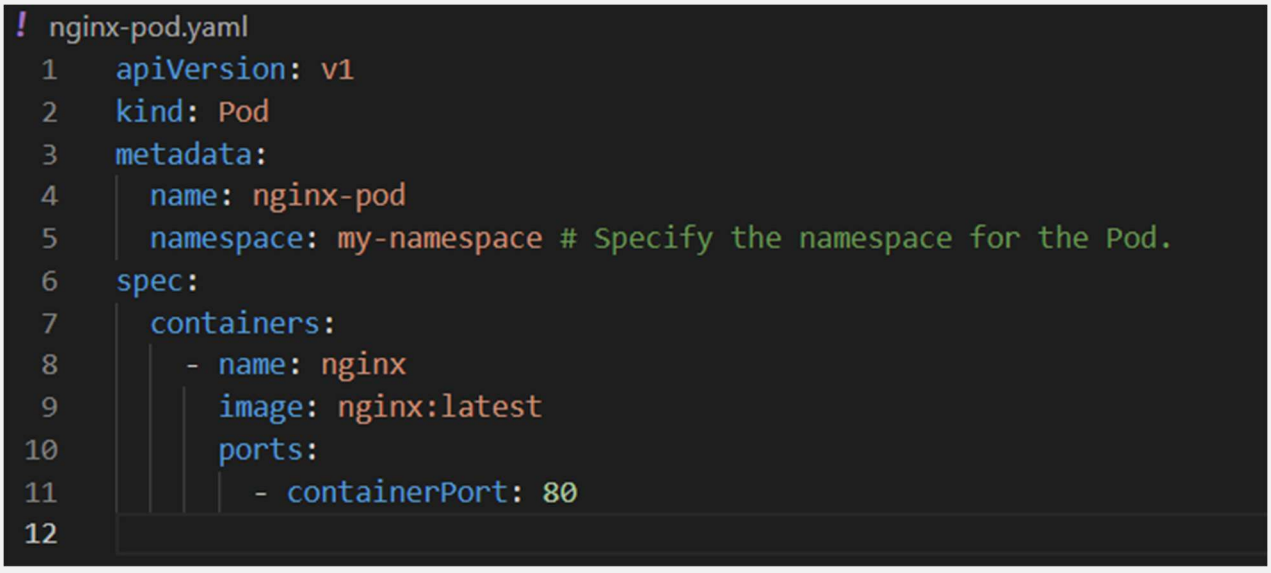
Step 4: Deploy Resources in a Namespace

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

```
PS C:\Github Repositores\CDS-LAB-SUBMISSION-2022-26\R2142221383> 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

```
PS C:\Github Repositores\CDS-LAB-SUBMISSION-2022-26\R2142221383> kubectl get pods -n my-namespace
NAME        READY   STATUS    RESTARTS   AGE
nginx-pod   1/1     Running   0          53s
```

To describe the Pod and see detailed information:

kubectl describe pod nginx-pod -n my-namespace

```
PS C:\Github Repositores\CDS-LAB-SUBMISSION-2022-26\R2142221383> 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 11:41:07 +0530
Labels:           <none>
Annotations:      <none>
Status:           Running
IP:               10.1.0.16
IPs:
  IP:  10.1.0.16
```

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
```

nginx-service.yaml

```yaml
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

```
PS C:\Github Repositores\CDS-LAB-SUBMISSION-2022-26\R2142221383> 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

```
PS C:\Github Repositores\CDS-LAB-SUBMISSION-2022-26\R2142221383> kubectl get services -n my-namespace
NAME            TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)   AGE
nginx-service   ClusterIP   10.99.2.45    <none>        80/TCP    37s
```

To describe the Service and see detailed information:

kubectl describe service nginx-service -n my-namespace

```
PS C:\Github Repositores\CDS-LAB-SUBMISSION-2022-26\R2142221383> 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:                10.99.2.45
IPs:               10.99.2.45
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

```
PS C:\Github Repositores\CDS-LAB-SUBMISSION-2022-26\R2142221383> kubectl get pods -n my-namespace
NAME        READY   STATUS    RESTARTS   AGE
nginx-pod   1/1     Running   0          4m13s
```

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:\Github Repositores\CDS-LAB-SUBMISSION-2022-26\R2142221383> kubectl config set-context --current --namespace=my-namespace
Context "docker-desktop" modified.
```

Verify the current context's namespace:

kubectl config view --minify | grep namespace:

```
an626@HP MINGW64 /c/Github Repositores/CDS-LAB-SUBMISSION-2022-26/R2142221383 (m
ain)
$ 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

```
PS C:\Github Repositores\CDS-LAB-SUBMISSION-2022-26\R2142221383> 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

```
PS C:\Github Repositores\CDS-LAB-SUBMISSION-2022-26\R2142221383> kubectl get namespaces
NAME              STATUS   AGE
default           Active   21d
kube-node-lease   Active   21d
kube-public       Active   21d
kube-system       Active   21d
```