

Lab Exercise 9- Managing Namespaces in Kubernetes

Step 1: Understand Namespaces

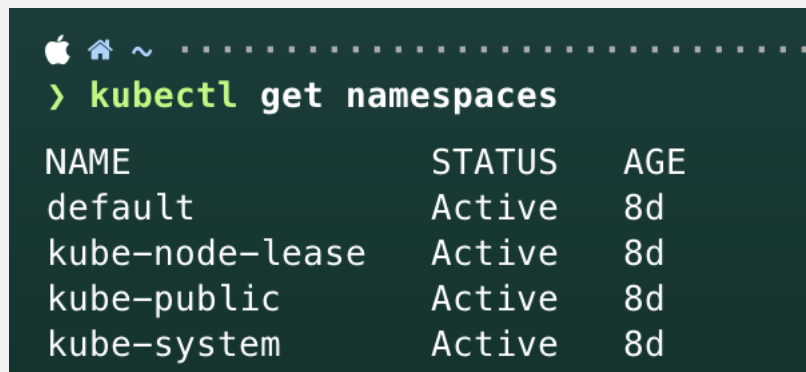
Namespaces provide a mechanism for scoping resources in a cluster. Namespaces can be used to:

- 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
```



A terminal window with a dark background and light text. The prompt is a green '>' followed by the command 'kubectl get namespaces' in green. The output is a table with three columns: NAME, STATUS, and AGE. The rows are: default (Active, 8d), kube-node-lease (Active, 8d), kube-public (Active, 8d), and kube-system (Active, 8d).

NAME	STATUS	AGE
default	Active	8d
kube-node-lease	Active	8d
kube-public	Active	8d
kube-system	Active	8d

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

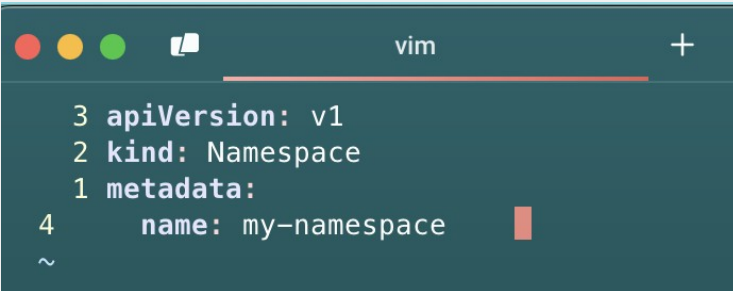
Step 3: Create a Namespace

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
```

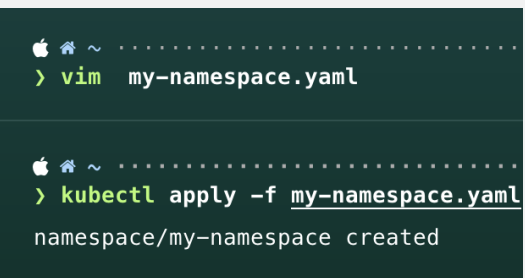
A screenshot of a terminal window with a vim editor. The editor shows the following content:

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

The line numbers 1, 2, 3, and 4 are on the left margin. The vim window has a title bar with "vim" and a plus sign.

Apply this YAML to create the namespace:

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

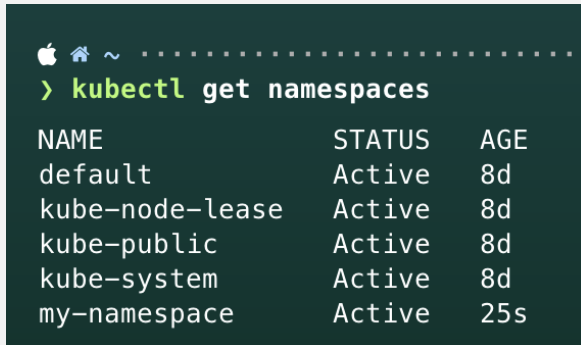
A screenshot of a terminal window showing the execution of the kubectl command. The terminal output is as follows:

```
> vim my-namespace.yaml

> kubectl apply -f my-namespace.yaml
namespace/my-namespace created
```

Verify that the namespace is created:

kubectl get namespaces

A terminal window with a dark green background and white text. The prompt is a green prompt character followed by 'kubectl get namespaces'. The output is a table with three columns: NAME, STATUS, and AGE. The rows are: default (Active, 8d), kube-node-lease (Active, 8d), kube-public (Active, 8d), kube-system (Active, 8d), and my-namespace (Active, 25s).

NAME	STATUS	AGE
default	Active	8d
kube-node-lease	Active	8d
kube-public	Active	8d
kube-system	Active	8d
my-namespace	Active	25s

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

```
vim
10 apiVersion: v1
9 kind: Pod
8 metadata:
7   name: nginx-pod
6   namespace: my-namespace # Specify the namespace for the Pod.
5 spec:
4   containers:
3     - name: nginx
2       image: nginx:latest
1   ports:
11     - containerPort: 80
```

Apply this YAML to create the Pod:

kubectl apply -f nginx-pod.yaml

```
Apple Home ~ .....
> vim nginx-pod.yaml

Apple Home ~ .....
> 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

```
Apple Home ~ .....
> kubectl get pods -n my-namespace
```

NAME	READY	STATUS	RESTARTS	AGE
nginx-pod	0/1	ContainerCreating	0	24s

To describe the Pod and see detailed information:

kubectl describe pod nginx-pod -n my-namespace

```

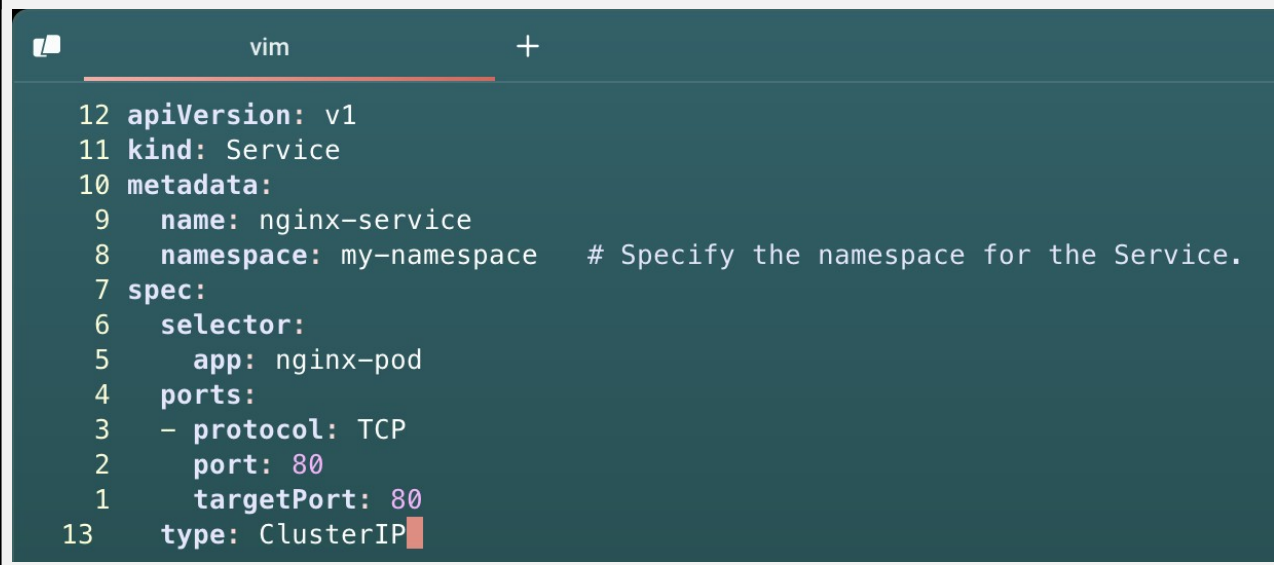
Name:          nginx-pod
Namespace:     my-namespace
Priority:       0
Service Account: default
Node:          docker-desktop/192.168.65.3
Start Time:    Fri, 15 Nov 2024 14:07:13 +0530
Labels:        <none>
Annotations:   <none>
Status:        Pending
IP:            <none>
IPs:           <none>
Containers:
  nginx:
    Container ID:
    Image:        nginx:latest
    Image ID:
    Port:         80/TCP
    Host Port:    0/TCP
    State:        Waiting
      Reason:     ContainerCreating
    Ready:        False
    Restart Count: 0
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-npkst (ro)
Conditions:
  Type                               Status
  PodReadyToStartContainers          False
  Initialized                        True
  Ready                              False
  ContainersReady                    False
  PodScheduled                       True
Volumes:
  kube-api-access-npkst:
    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

```

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
```

A screenshot of a vim editor window. The window has a dark teal background. At the top, there's a header bar with a file icon, the text 'vim', and a '+' sign. The main area shows the content of the 'nginx-service.yaml' file, with line numbers 1 through 13 on the left. The text is as follows:
12 apiVersion: v1
11 kind: Service
10 metadata:
9 name: nginx-service
8 namespace: my-namespace # Specify the namespace for the Service.
7 spec:
6 selector:
5 app: nginx-pod
4 ports:
3 - protocol: TCP
2 port: 80
1 targetPort: 80
13 type: ClusterIP
A red cursor is visible at the end of line 13.

Apply this YAML to create the Service:

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

```
Apple Home ~ .....
> vim nginx-service.yaml

Apple Home ~ .....
> 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
```

```
Apple Home ~ .....
> kubectl get services -n my-namespace
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
nginx-service	ClusterIP	10.106.116.245	<none>	80/TCP	24s

To describe the Service and see detailed information:

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

```
Apple Home ~ ..... * docker-desktop © 02:10:20 PM
> kubectl describe service nginx-service -n my-namespace
Name: nginx-service
Namespace: my-namespace
Labels:
Annotations:
Selector: app=nginx-pod
Type: ClusterIP
IP Family Policy: SingleStack
IP Families: IPv4
IP: 10.106.116.245
IPs: 10.106.116.245
Port: <unset> 80/TCP
TargetPort: 80/TCP
Endpoints:
Session Affinity: None
Internal Traffic Policy: Cluster
Events:
  Type    Reason              Age    From          Message
  ----    -
Warning  FailedToUpdateEndpointSlices  51s    endpoint-slice-controller  Error updating Endpoint Slices for Service my-namespace/nginx-service: failed to create EndpointSlice for Service my-namespace/nginx-service: Unauthorized
```

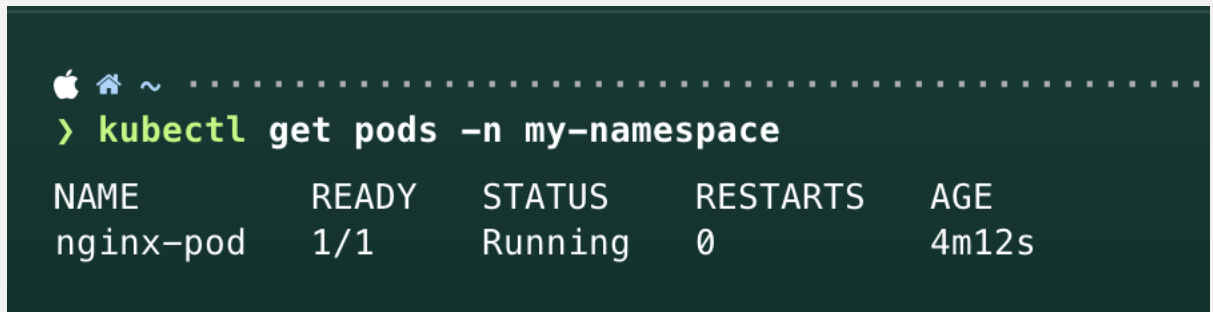
Step 5: Switching Context Between Namespaces

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
```



```
Apple Home ~ .....  
> kubectl get pods -n my-namespace  

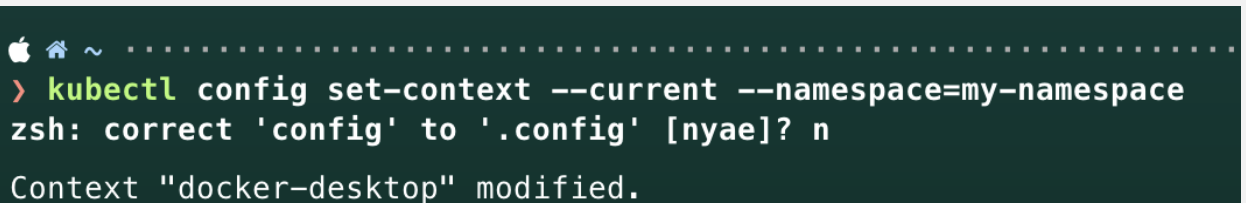

| NAME      | READY | STATUS  | RESTARTS | AGE   |
|-----------|-------|---------|----------|-------|
| nginx-pod | 1/1   | Running | 0        | 4m12s |


```

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
```



```
Apple Home ~ .....  
> kubectl config set-context --current --namespace=my-namespace  
zsh: correct 'config' to '.config' [nyae]? n  
Context "docker-desktop" modified.
```


Verify the current context's namespace:

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

```
Apple Home ~ .....  
> kubectl config view --minify | grep namespace:  
zsh: correct 'config' to '.config' [nyae]? n  
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
```

```
Apple Home ~ .....  
> 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

```
🍏 🏠 ~ .....  
> kubectl get namespaces
```

NAME	STATUS	AGE
default	Active	8d
kube-node-lease	Active	8d
kube-public	Active	8d
kube-system	Active	8d