Lab Exercise 9- Managing Namespaces in Kubernetes

Step 1: Understand Namespaces

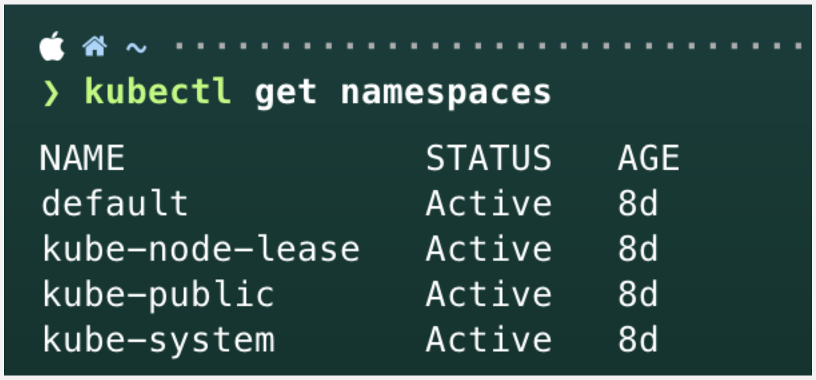
Namespaces provide a mechanism for scoping resources in a cluster. Namespaces can be used to:

- 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
```



```
❯ kubectl get namespaces
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.

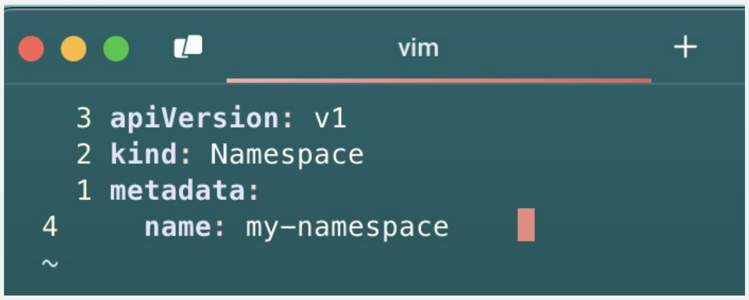
Step 3: Create a Namespace

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:

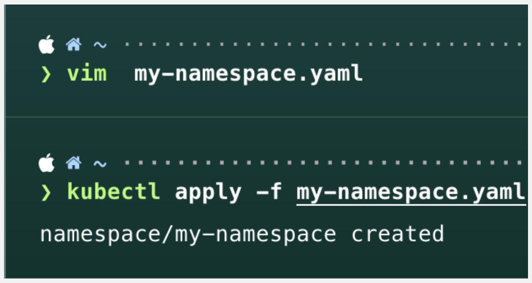
```
apiVersion: v1
kind: Namespace
metadata:
  name: my-namespace
```



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

Apply this YAML to create the namespace:

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

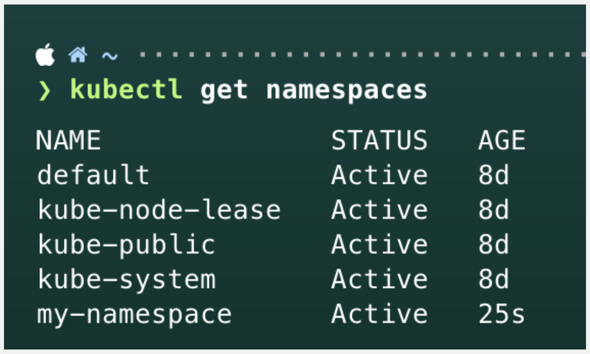


```
 ~ ..............................
❯ vim  my-namespace.yaml

 ~ ..............................
❯ kubectl apply -f my-namespace.yaml
namespace/my-namespace created
```

Verify that the namespace is created:

```
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
my-namespace      Active   25s
```

You should see my-namespace listed in the output.

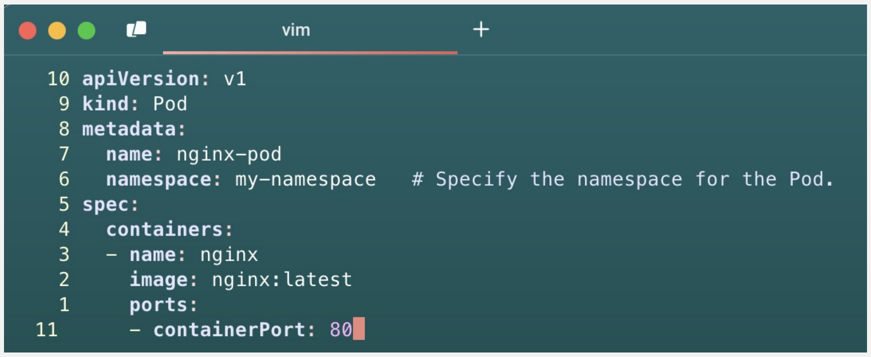
Step 4: Deploy Resources in a Namespace

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:

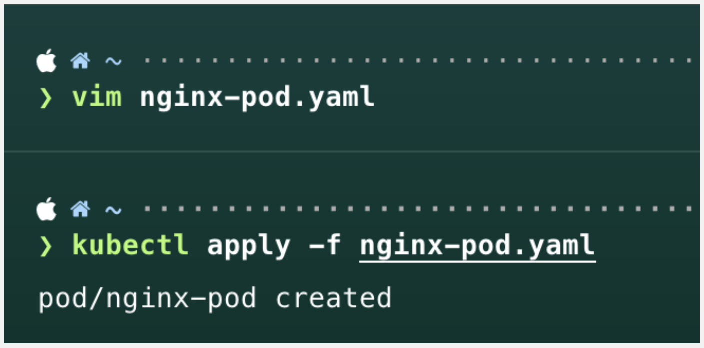
```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
  namespace: my-namespace   # Specify the namespace for the Pod.
spec:
  containers:
  - name: nginx
    image: nginx:latest
  ports:
- containerPort: 80
```



```
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
```

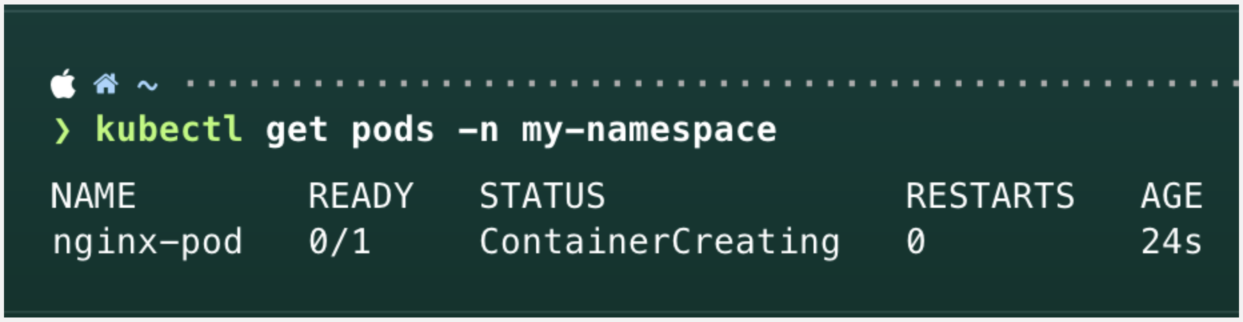


```
❯ vim nginx-pod.yaml

❯ kubectl apply -f nginx-pod.yaml
pod/nginx-pod created
```

Check the status of the Pod within the namespace:

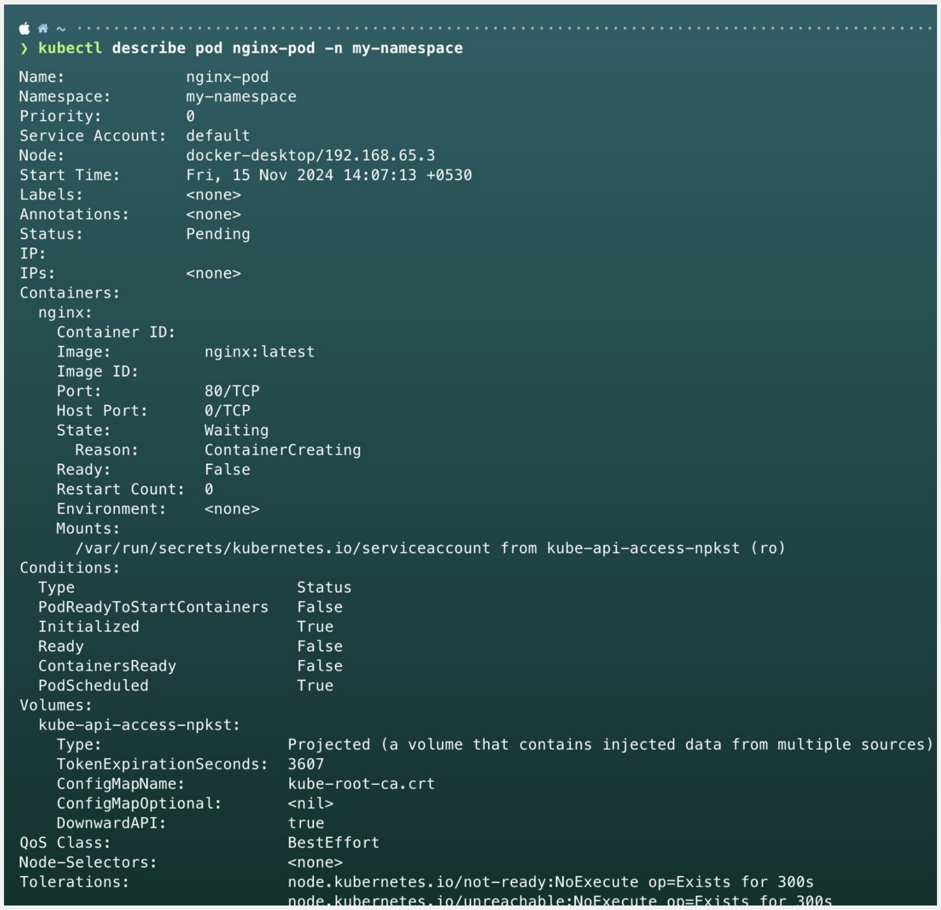
```
kubectl get pods -n my-namespace
```



```
❯ 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

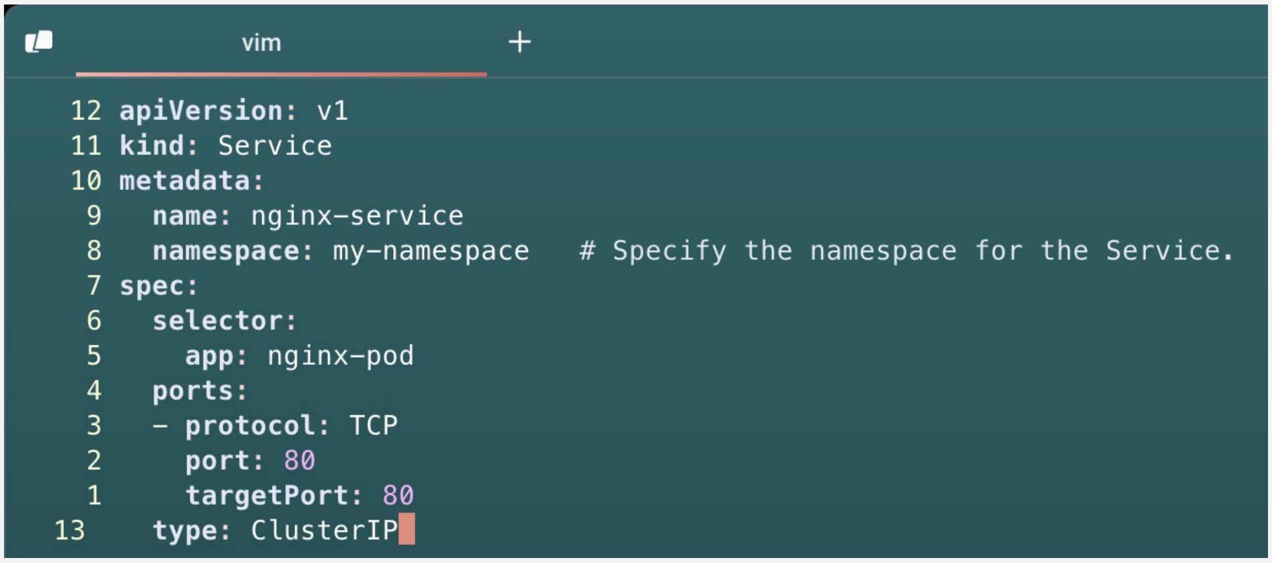


```
❯ kubectl describe pod nginx-pod -n my-namespace
Name:             nginx-pod
Namespace:        my-namespace
Priority:         0
Service Account:  default
Node:             docker-desktop/192.168.65.3
Start Time:       Fri, 15 Nov 2024 14:07:13 +0530
Labels:           <none>
Annotations:      <none>
Status:           Pending
IP:
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:

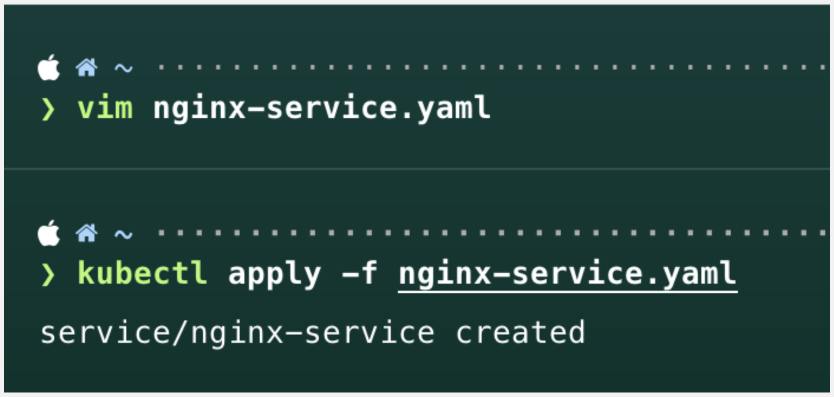
```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
  namespace: my-namespace   # Specify the namespace for the Service.
spec:
  selector:
    app: nginx-pod
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
  type: ClusterIP
```



```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
```

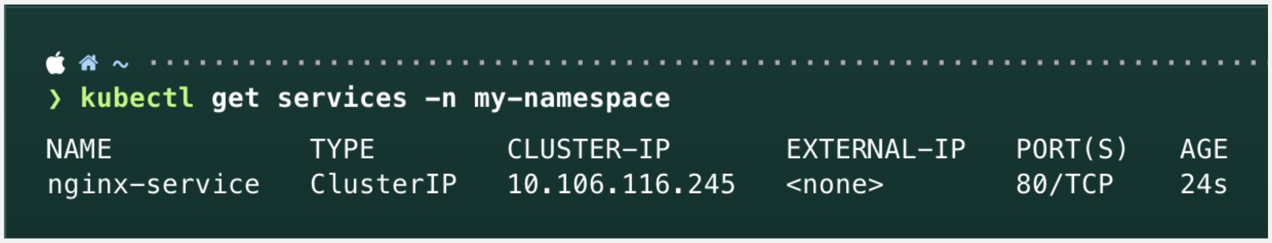


```
~ ································
❯ vim nginx-service.yaml

~ ································
❯ kubectl apply -f nginx-service.yaml
service/nginx-service created
```

Check the status of the Service within the namespace:

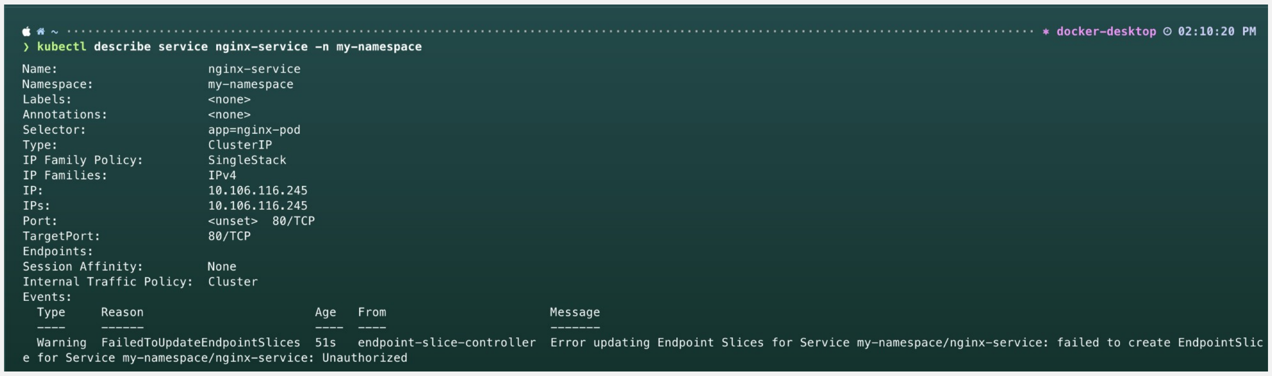
```
kubectl get services -n my-namespace
```



```
~ ····························································
❯ 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
```



```
~ ······························································ * docker-desktop ⊙ 02:10:20 PM
❯ 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:                       10.106.116.245
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 EndpointSlic
e for Service my-namespace/nginx-service: Unauthorized
```

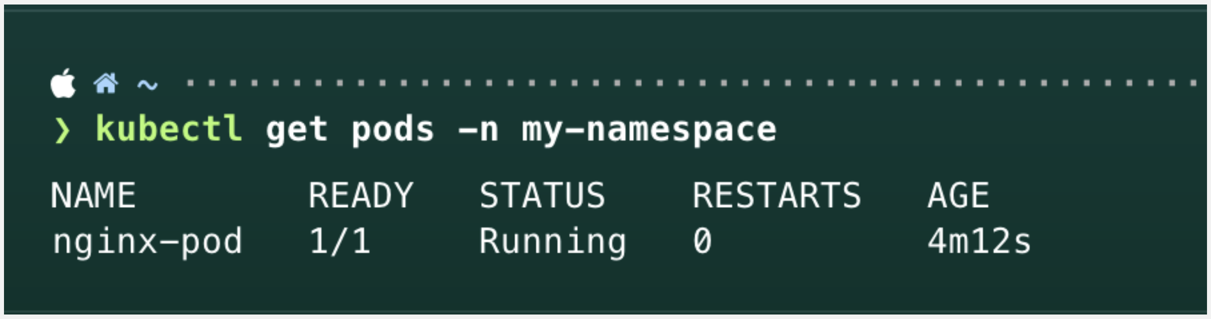
Step 5: Switching Context Between Namespaces

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
```

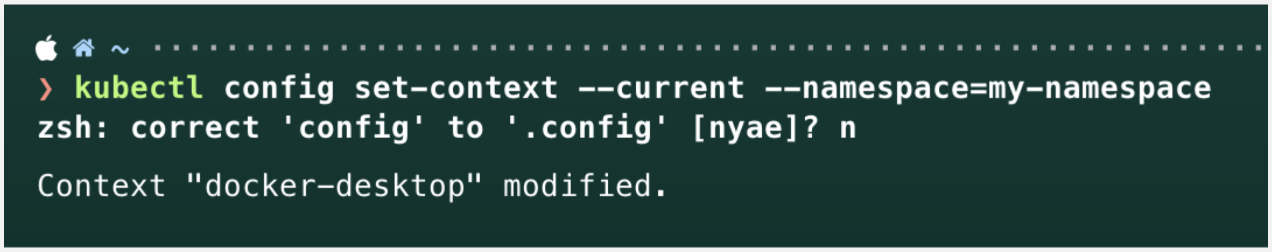


```
 ~ ..................................................
❯ kubectl get pods -n my-namespace
NAME         READY   STATUS    RESTARTS   AGE
nginx-pod    1/1     Running   0          4m12s
```

Set Default Namespace for kubectl Commands

To avoid specifying the namespace every time, you can set the default namespace for the current context:

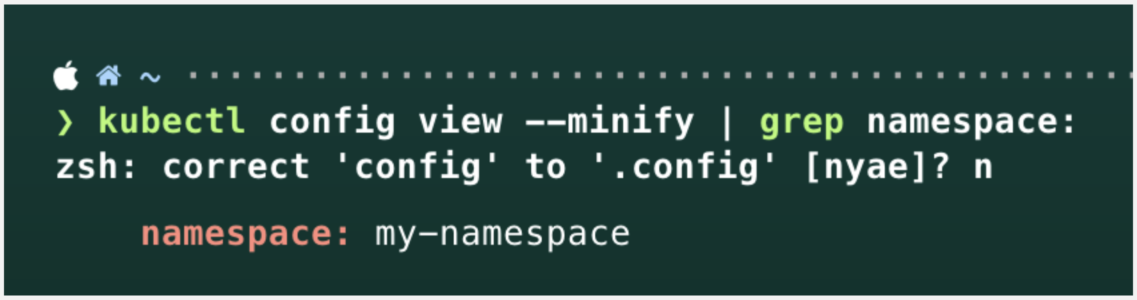
```
kubectl config set-context --current –namespace=my-namespace
```



```
 ~ ..................................................
❯ 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:
```



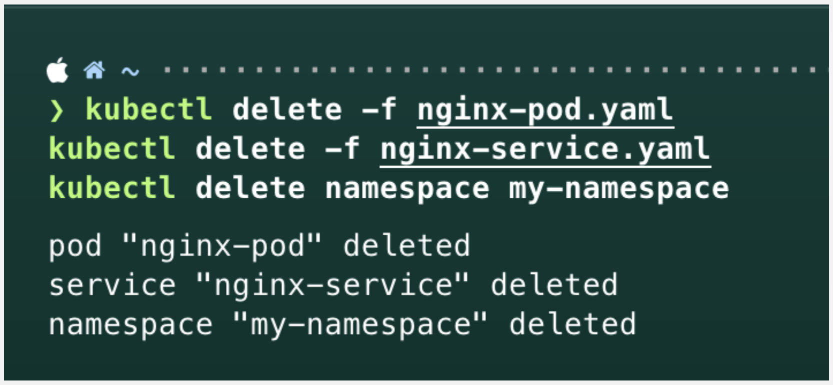
```
❯ kubectl config view --minify | grep namespace:
zsh: correct 'config' to '.config' [nyae]? n

    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
```

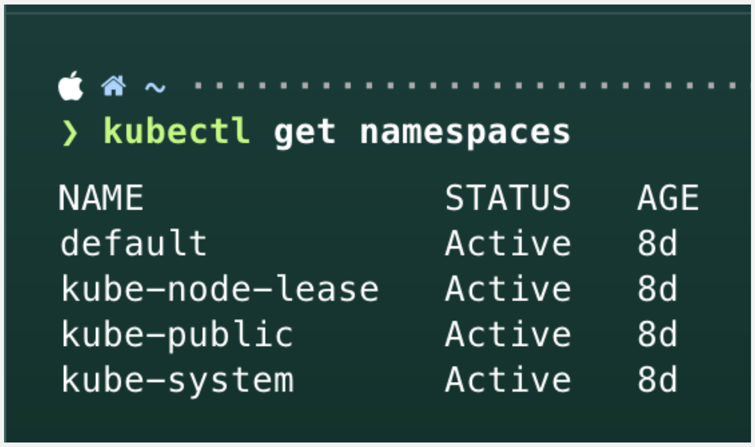


```
❯ kubectl delete -f nginx-pod.yaml
kubectl delete -f nginx-service.yaml
kubectl delete namespace my-namespace

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
```