

# Lab Exercise 13- Managing Namespaces in Kubernetes

**Name:- Vansh Bhatt**

**SapId:- 500125395**

**R.No:- R 2142231689**

**Batch:- DevOps-B1**

**To:- Hitesh Sharma Sir**

## Step 1: Understand Namespaces

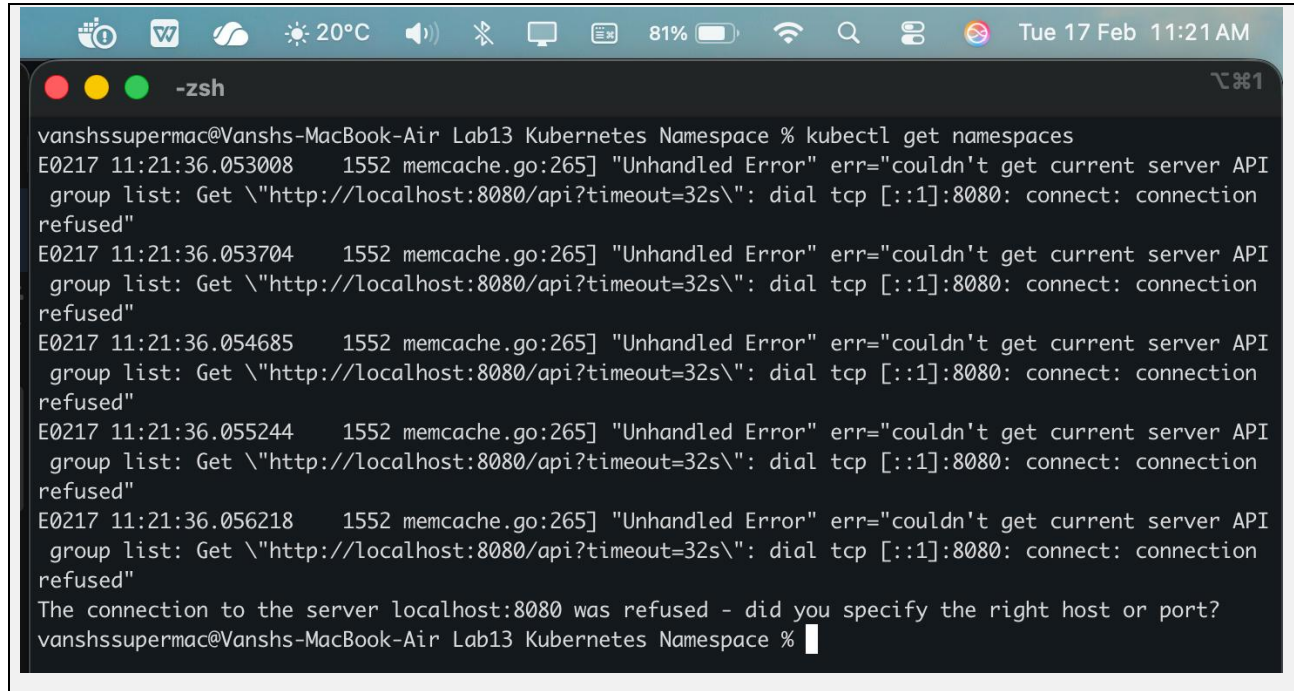
Namespaces provide a mechanism for scoping resources in a cluster. Namespaces can be used to:

- 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 on a MacBook Air. The title bar shows system status icons (Wi-Fi, battery at 81%, 20°C, etc.) and the time 'Tue 17 Feb 11:21 AM'. The terminal window has a dark background and shows a command prompt 'vanshssupermac@Vanshs-MacBook-Air Lab13 Kubernetes Namespace %' followed by 'kubectl get namespaces'. The output consists of five identical error messages: 'E0217 11:21:36.053008 1552 memcache.go:265] "Unhandled Error" err="couldn't get current server API group list: Get \"http://localhost:8080/api?timeout=32s\": dial tcp [::1]:8080: connect: connection refused\"'. The prompt is repeated at the end of the output.

```
vanshssupermac@Vanshs-MacBook-Air Lab13 Kubernetes Namespace % kubectl get namespaces
E0217 11:21:36.053008 1552 memcache.go:265] "Unhandled Error" err="couldn't get current server API
group list: Get \"http://localhost:8080/api?timeout=32s\": dial tcp [::1]:8080: connect: connection
refused"
E0217 11:21:36.053704 1552 memcache.go:265] "Unhandled Error" err="couldn't get current server API
group list: Get \"http://localhost:8080/api?timeout=32s\": dial tcp [::1]:8080: connect: connection
refused"
E0217 11:21:36.054685 1552 memcache.go:265] "Unhandled Error" err="couldn't get current server API
group list: Get \"http://localhost:8080/api?timeout=32s\": dial tcp [::1]:8080: connect: connection
refused"
E0217 11:21:36.055244 1552 memcache.go:265] "Unhandled Error" err="couldn't get current server API
group list: Get \"http://localhost:8080/api?timeout=32s\": dial tcp [::1]:8080: connect: connection
refused"
E0217 11:21:36.056218 1552 memcache.go:265] "Unhandled Error" err="couldn't get current server API
group list: Get \"http://localhost:8080/api?timeout=32s\": dial tcp [::1]:8080: connect: connection
refused"
The connection to the server localhost:8080 was refused - did you specify the right host or port?
vanshssupermac@Vanshs-MacBook-Air Lab13 Kubernetes Namespace %
```

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

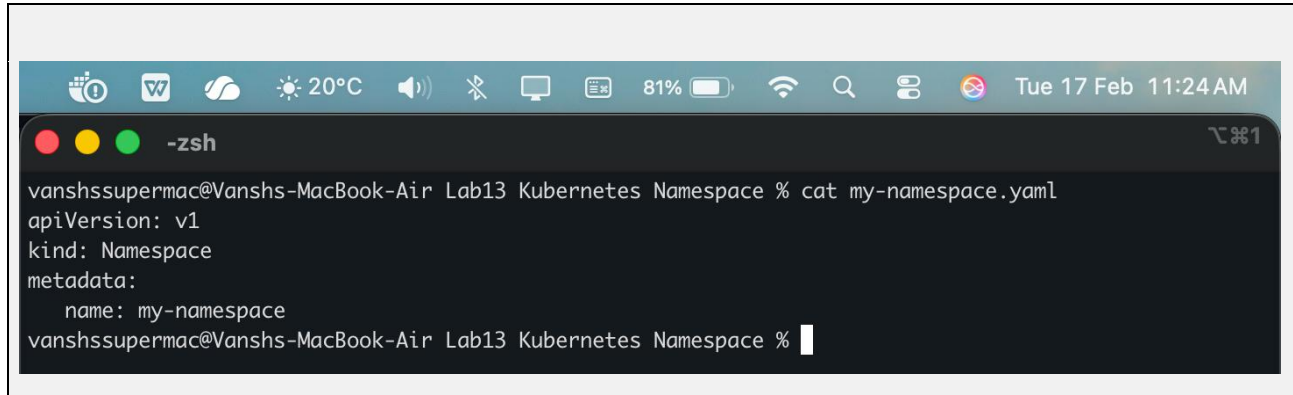
### Step 3: Create a Namespace

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:

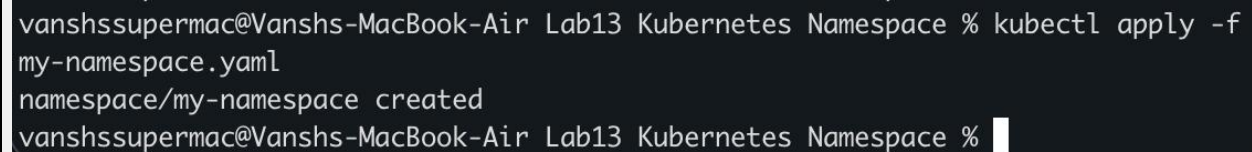
```
apiVersion: v1
kind: Namespace
metadata:
  name: my-namespace
```

A terminal window on a Mac with a dark background. The title bar shows standard macOS window controls and system status icons (Wi-Fi, battery at 81%, temperature at 20°C, etc.) along with the time 'Tue 17 Feb 11:24 AM'. The terminal prompt is 'vanshssupermac@Vanshs-MacBook-Air Lab13 Kubernetes Namespace %'. The user has entered 'cat my-namespace.yaml' and the output is displayed: 'apiVersion: v1', 'kind: Namespace', 'metadata:', ' name: my-namespace'. The cursor is at the end of the prompt line.

```
vanshssupermac@Vanshs-MacBook-Air Lab13 Kubernetes Namespace % cat my-namespace.yaml
apiVersion: v1
kind: Namespace
metadata:
  name: my-namespace
vanshssupermac@Vanshs-MacBook-Air Lab13 Kubernetes Namespace %
```

Apply this YAML to create the namespace:

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

A terminal window showing the execution of the 'kubectl apply' command. The prompt is 'vanshssupermac@Vanshs-MacBook-Air Lab13 Kubernetes Namespace %'. The user enters 'kubectl apply -f my-namespace.yaml'. The output is 'namespace/my-namespace created'. The cursor is at the end of the prompt line.

```
vanshssupermac@Vanshs-MacBook-Air Lab13 Kubernetes Namespace % kubectl apply -f
my-namespace.yaml
namespace/my-namespace created
vanshssupermac@Vanshs-MacBook-Air Lab13 Kubernetes Namespace %
```

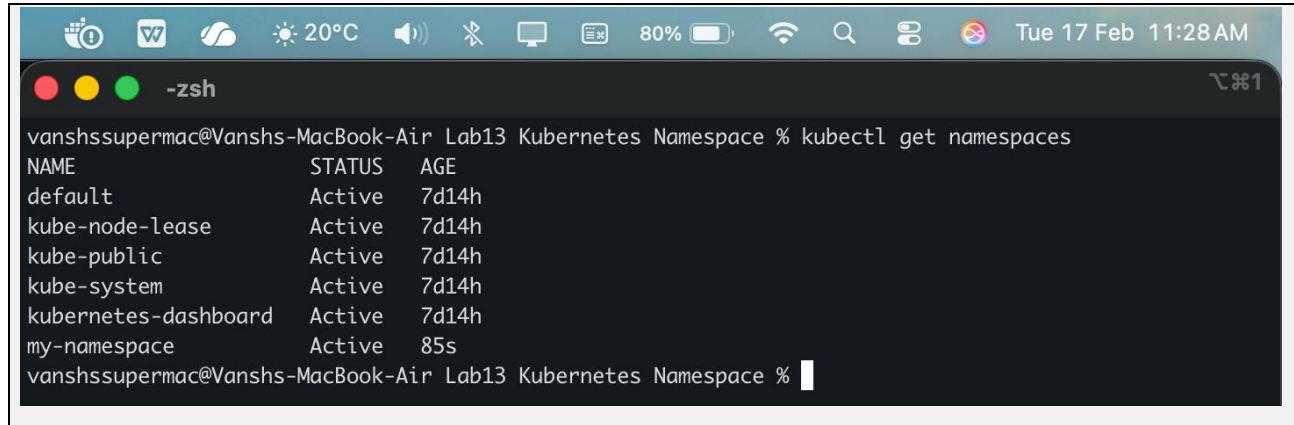
Using kubectl Command

Alternatively, create a namespace using the kubectl command:

```
kubectl create namespace my-namespace
```

Verify that the namespace is created:

```
kubectl get namespaces
```

A screenshot of a macOS terminal window. The title bar shows standard macOS window controls (red, yellow, green buttons) and the text '-zsh'. The terminal content shows the command 'kubectl get namespaces' being executed. The output is a table with three columns: NAME, STATUS, and AGE. The rows are: default (Active, 7d14h), kube-node-lease (Active, 7d14h), kube-public (Active, 7d14h), kube-system (Active, 7d14h), kubernetes-dashboard (Active, 7d14h), and my-namespace (Active, 85s). The prompt 'vanshssupermac@Vanshs-MacBook-Air Lab13 Kubernetes Namespace %' is visible at the bottom.

```
vanshssupermac@Vanshs-MacBook-Air Lab13 Kubernetes Namespace % kubectl get namespaces
NAME                STATUS    AGE
default             Active   7d14h
kube-node-lease     Active   7d14h
kube-public         Active   7d14h
kube-system         Active   7d14h
kubernetes-dashboard Active   7d14h
my-namespace        Active   85s
vanshssupermac@Vanshs-MacBook-Air Lab13 Kubernetes Namespace %
```

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
spec:
  containers:
  - name: nginx
    image: nginx:latest
  ports:
  - containerPort: 80
```

```
apiVersion: v1
kind: pod
metadata:
  name: vb-nginx
  namespace: my-namespace
spec:
  containers:
  - name: nginx
    image: nginx:latest
    ports:
    - containerPort: 80
```

Apply this YAML to create the Pod:

kubectl apply -f nginx-pod.yaml

```
vanshssupermac@Vanshs-MacBook-Air Lab13 Kubernetes Namespace % kubectl apply -f nginx-pod.yaml
pod/vb-nginx created
vanshssupermac@Vanshs-MacBook-Air Lab13 Kubernetes Namespace %
```

Check the status of the Pod within the namespace:

kubectl get pods -n my-namespace

```
vanshssupermac@Vanshs-MacBook-Air Lab13 Kubernetes Namespace % kubectl get pods -n my-namespace
NAME      READY   STATUS             RESTARTS   AGE
vb-nginx  0/1     ContainerCreating   0          35s
vanshssupermac@Vanshs-MacBook-Air Lab13 Kubernetes Namespace %
```

To describe the Pod and see detailed information:

kubectl describe pod nginx-pod -n my-namespace

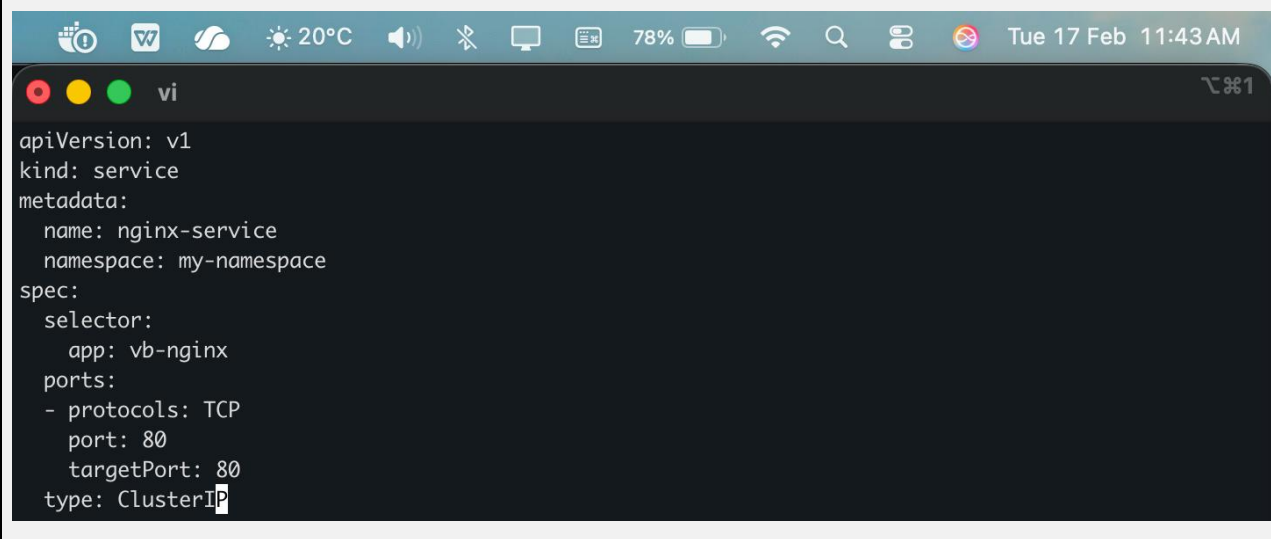
```
vanshssupermac@Vanshs-MacBook-Air Lab13 Kubernetes Namespace % kubectl describe pod vb-nginx -n my-na
namespace
Name:          vb-nginx
Namespace:     my-namespace
Priority:       0
Service Account: default
Node:          minikube/192.168.49.2
Start Time:    Tue, 17 Feb 2026 11:38:09 +0530
Labels:        <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-6wrbs (ro)
Conditions:
  Type                                Status
  PodReadyToStartContainers          False
  Initialized                        True
  Ready                              False
  ContainersReady                    False
  PodScheduled                       True
Volumes:
  kube-api-access-6wrbs:
    Type:          Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds: 3607
    ConfigMapName:    kube-root-ca.crt
    Optional:         false
```

Create a Service in the Namespace

Create a YAML file named **nginx-service.yaml** with the following content:

```
apiVersion: v1
```

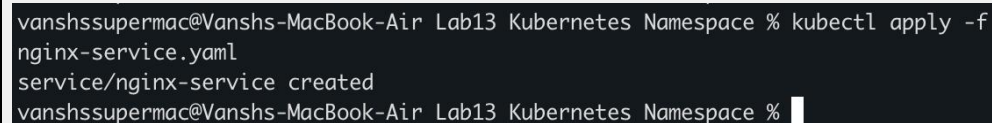
```
kind: Service
metadata:
  name: nginx-service
  namespace: my-namespace
spec:
  selector:
    app: nginx-pod
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
  type: ClusterIP
```



```
apiVersion: v1
kind: service
metadata:
  name: nginx-service
  namespace: my-namespace
spec:
  selector:
    app: vb-nginx
  ports:
  - protocols: TCP
    port: 80
    targetPort: 80
  type: ClusterIP
```

Apply this YAML to create the Service:

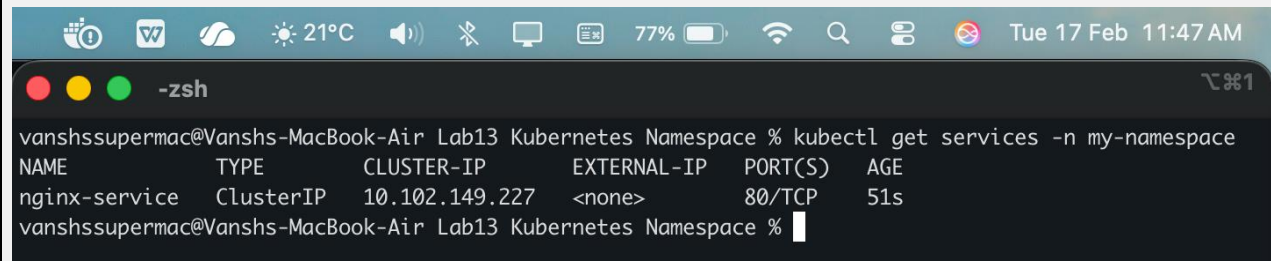
```
kubectl apply -f nginx-service.yaml
```



```
vanshssupermac@Vanshs-MacBook-Air Lab13 Kubernetes Namespace % kubectl apply -f
nginx-service.yaml
service/nginx-service created
vanshssupermac@Vanshs-MacBook-Air Lab13 Kubernetes Namespace %
```

Check the status of the Service within the namespace:

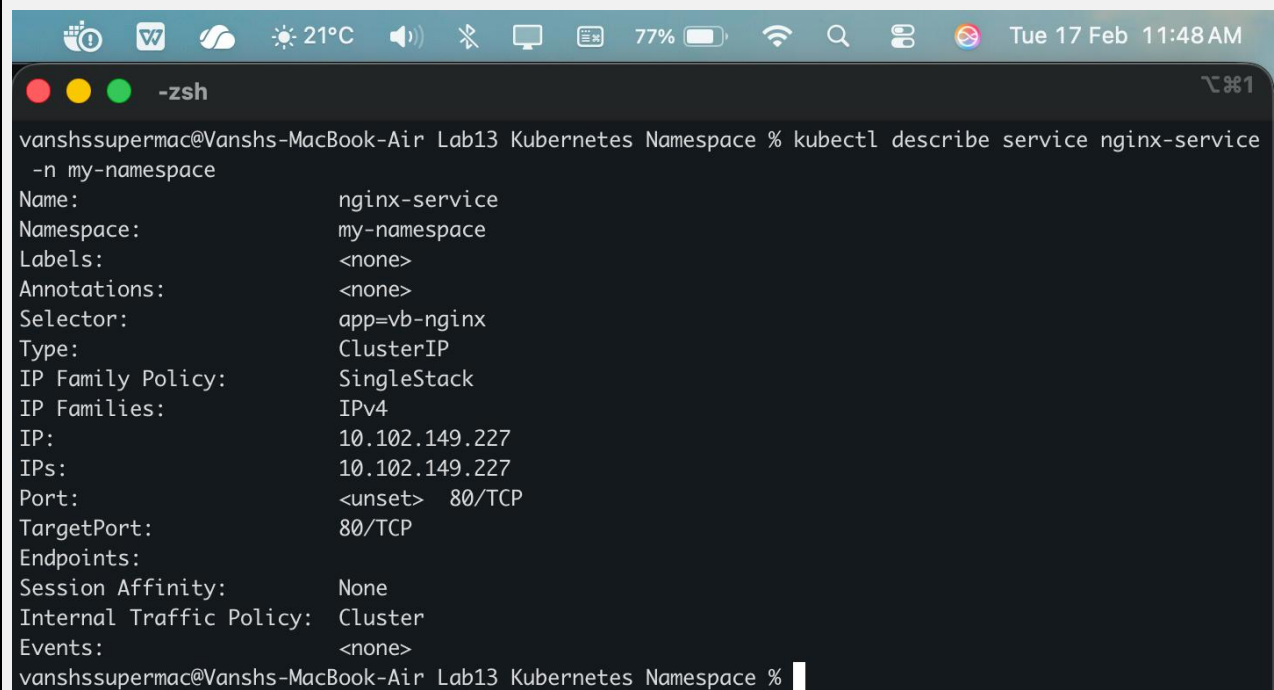
kubectl get services -n my-namespace



```
vanshssupermac@Vanshs-MacBook-Air Lab13 Kubernetes Namespace % kubectl get services -n my-namespace
NAME          TYPE        CLUSTER-IP    EXTERNAL-IP  PORT(S)    AGE
nginx-service  ClusterIP   10.102.149.227 <none>       80/TCP     51s
vanshssupermac@Vanshs-MacBook-Air Lab13 Kubernetes Namespace %
```

To describe the Service and see detailed information:

kubectl describe service nginx-service -n my-namespace



```
vanshssupermac@Vanshs-MacBook-Air Lab13 Kubernetes Namespace % 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:            10.102.149.227
IPs:           10.102.149.227
Port:          <unset> 80/TCP
TargetPort:    80/TCP
Endpoints:
Session Affinity: None
Internal Traffic Policy: Cluster
Events:        <none>
vanshssupermac@Vanshs-MacBook-Air Lab13 Kubernetes Namespace %
```

## Step 5: Switching Context Between Namespaces

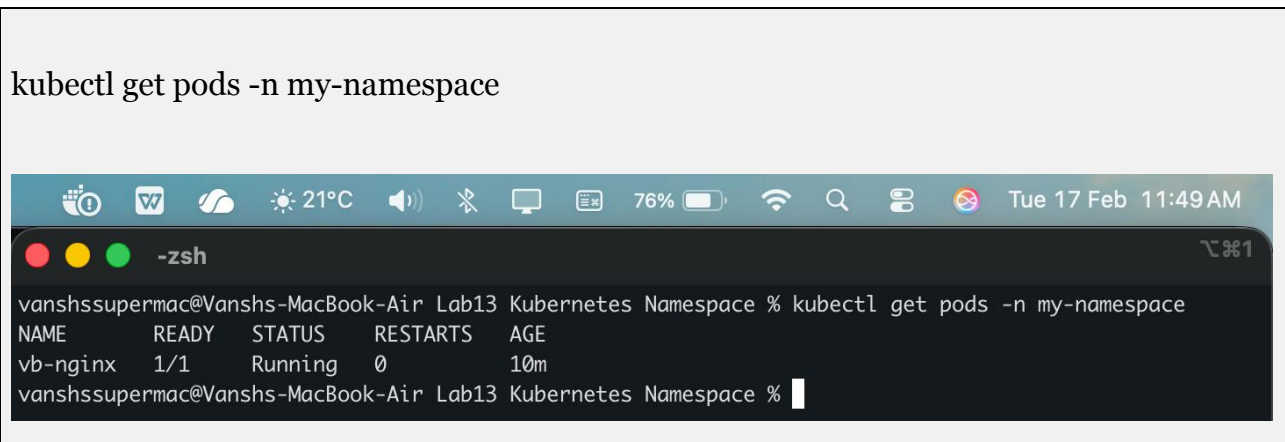


When working with multiple namespaces, you can specify the namespace in kubectl commands or switch the default context.

### 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
```



The screenshot shows a macOS terminal window titled "-zsh" with a status bar at the top displaying system icons, temperature (21°C), battery (76%), and time (Tue 17 Feb 11:49 AM). The terminal prompt is `vanshssupermac@Vanshs-MacBook-Air Lab13 Kubernetes Namespace %`. The command `kubectl get pods -n my-namespace` has been executed, resulting in the following output:

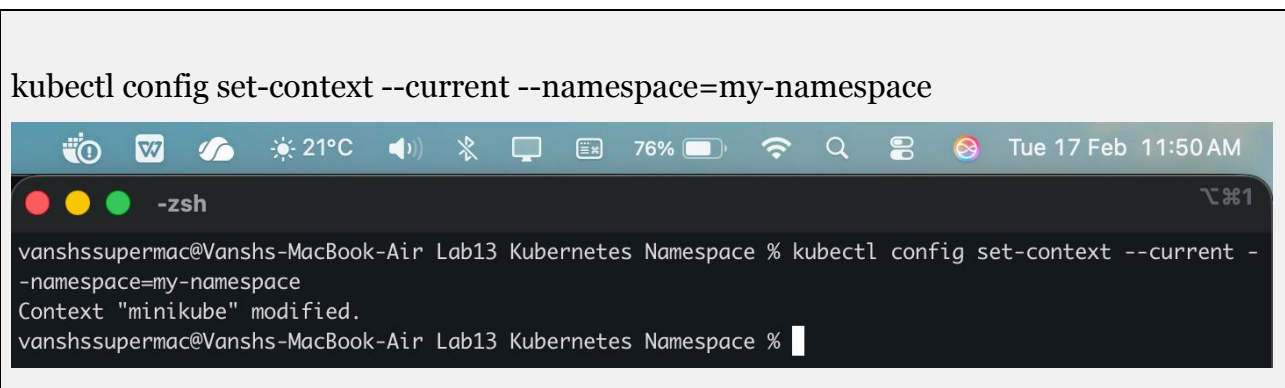
NAME	READY	STATUS	RESTARTS	AGE
vb-nginx	1/1	Running	0	10m

The terminal prompt is now `vanshssupermac@Vanshs-MacBook-Air Lab13 Kubernetes Namespace %`.

### 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
```



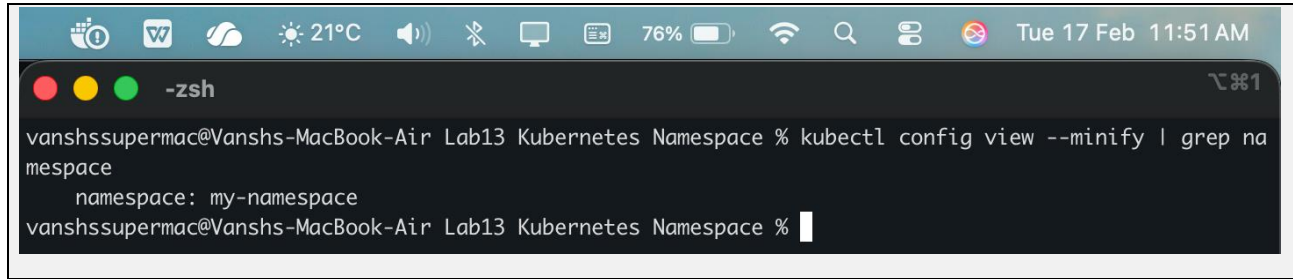
The screenshot shows the same macOS terminal window. The status bar now shows the time as Tue 17 Feb 11:50 AM. The terminal prompt is `vanshssupermac@Vanshs-MacBook-Air Lab13 Kubernetes Namespace %`. The command `kubectl config set-context --current --namespace=my-namespace` has been executed, resulting in the following output:

```
kubectl config set-context --current --namespace=my-namespace
Context "minikube" modified.
vanshssupermac@Vanshs-MacBook-Air Lab13 Kubernetes Namespace %
```

The terminal prompt is now `vanshssupermac@Vanshs-MacBook-Air Lab13 Kubernetes Namespace %`.

Verify the current context's namespace:

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

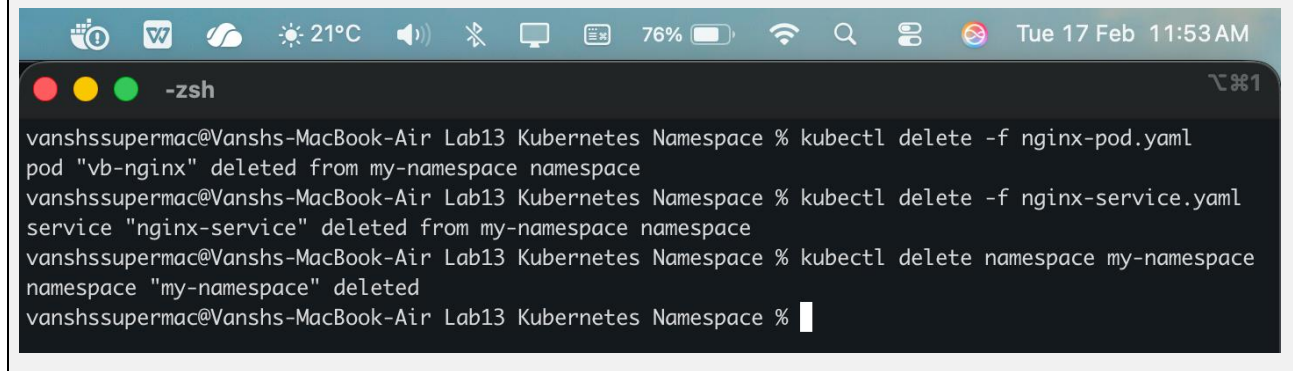


```
vanshssupermac@Vanshs-MacBook-Air Lab13 Kubernetes Namespace % kubectl config view --minify | grep namespace
namespace: my-namespace
vanshssupermac@Vanshs-MacBook-Air Lab13 Kubernetes Namespace %
```

## Step 6: Clean Up Resources

To delete the resources and the namespace you created:

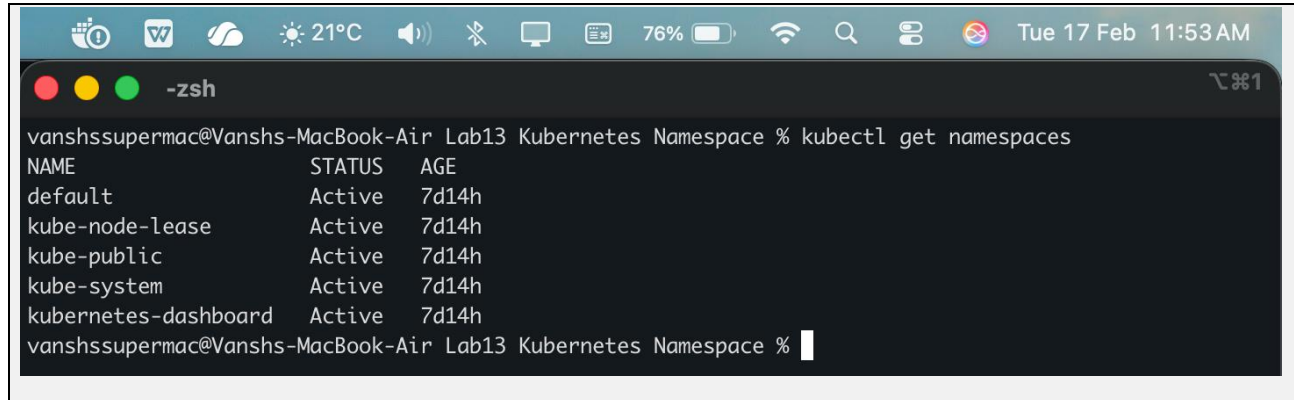
```
kubectl delete -f nginx-pod.yaml
kubectl delete -f nginx-service.yaml
kubectl delete namespace my-namespace
```



```
vanshssupermac@Vanshs-MacBook-Air Lab13 Kubernetes Namespace % kubectl delete -f nginx-pod.yaml
pod "vb-nginx" deleted from my-namespace namespace
vanshssupermac@Vanshs-MacBook-Air Lab13 Kubernetes Namespace % kubectl delete -f nginx-service.yaml
service "nginx-service" deleted from my-namespace namespace
vanshssupermac@Vanshs-MacBook-Air Lab13 Kubernetes Namespace % kubectl delete namespace my-namespace
namespace "my-namespace" deleted
vanshssupermac@Vanshs-MacBook-Air Lab13 Kubernetes Namespace %
```

Ensure that the namespace and all its resources are deleted:

```
kubectl get namespaces
```

A screenshot of a macOS terminal window. The title bar shows standard macOS window controls (red, yellow, green buttons) and the text '-zsh'. The status bar at the top displays system information: weather (21°C), volume, Bluetooth, display, keyboard, 76% battery, Wi-Fi, search, and the date/time 'Tue 17 Feb 11:53 AM'. The terminal content shows a user 'vanshssupermac' at 'Vanshs-MacBook-Air' in the 'Lab13' directory, running the command 'kubectl get namespaces'. The output is a table with three columns: NAME, STATUS, and AGE. It lists five namespaces: 'default', 'kube-node-lease', 'kube-public', 'kube-system', and 'kubernetes-dashboard', all with a status of 'Active' and an age of '7d14h'. The prompt 'vanshssupermac@Vanshs-MacBook-Air Lab13 Kubernetes Namespace %' is visible at the bottom with a cursor.

```
vanshssupermac@Vanshs-MacBook-Air Lab13 Kubernetes Namespace % kubectl get namespaces
NAME                STATUS    AGE
default             Active   7d14h
kube-node-lease     Active   7d14h
kube-public         Active   7d14h
kube-system         Active   7d14h
kubernetes-dashboard Active   7d14h
vanshssupermac@Vanshs-MacBook-Air Lab13 Kubernetes Namespace %
```

**Thank You**