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:

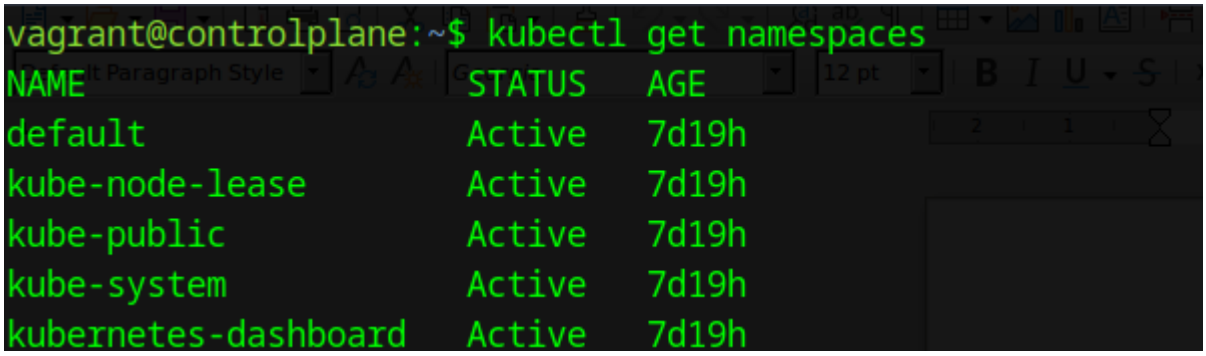
- Create environments for different applications or teams.
- Apply policies like resource quotas or network policies on a per-namespace basis.
- Separate operational environments (like development and production).

Step 2: List Existing Namespaces

To list all the namespaces in your Kubernetes cluster:

```
kubectl get namespaces
```

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
```

Apply this YAML to create the namespace:

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

Verify that the namespace is created:

```
kubectl get namespaces
```

You should see my-namespace listed in the output.

```
vagrant@controlplane:~$ kubectl create ns my-ns
namespace/my-ns created
vagrant@controlplane:~$ kubectl get namespaces
NAME                   STATUS   AGE
default                Active   7d19h
kube-node-lease        Active   7d19h
kube-public            Active   7d19h
kube-system            Active   7d19h
kubernetes-dashboard   Active   7d19h
my-ns                  Active   2s
```

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:
  labels:
    app: nginx-app
  name: easy-drive-pod
  namespace: my-ns
spec:
  containers:
    - image: booraraman/easy-drive-rentals:1
```

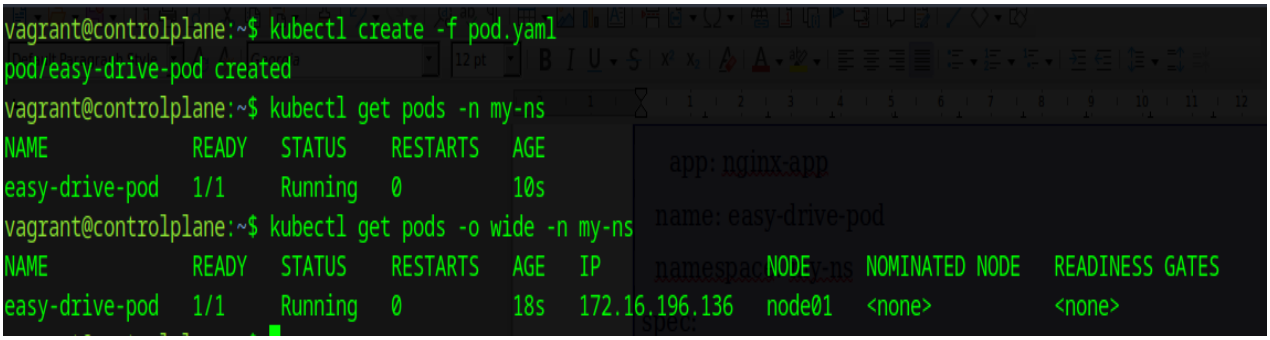
```
    name: nginx
    ports:
      - containerPort: 5600
```

Apply this YAML to create the Pod:

```
kubectl apply -f pod.yaml
```

Check the status of the Pod within the namespace:

```
kubectl get pods -n my-ns
```



Create a Service in the Namespace

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

```
apiVersion: v1
kind: Service
metadata:
  name: nodeport-service
  namespace: my-ns
```

```
spec:
  selector:
    app: nginx-app  # Matches the label of the Deployment Pods
  ports:
     - port: 5600                # Port the Service exposes internally within the cluster
      targetPort: 5600          # Port the NGINX container is listening on inside the Pods
      nodePort: 30007         # Exposes the service on port 30007 on each node in the cluster
  type: NodePort  # Exposes the service via a NodePort
```

Apply this YAML to create the Service:

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

Check the status of the Service within the namespace:

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

```
vagrant@controlplane:~$ kubectl get svc
NAME                TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
kubernetes          ClusterIP   172.17.0.1     <none>        443/TCP        7d19h
nodeport-service    NodePort    172.17.52.22   <none>        5600:30007/TCP 7d19h
```

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
```

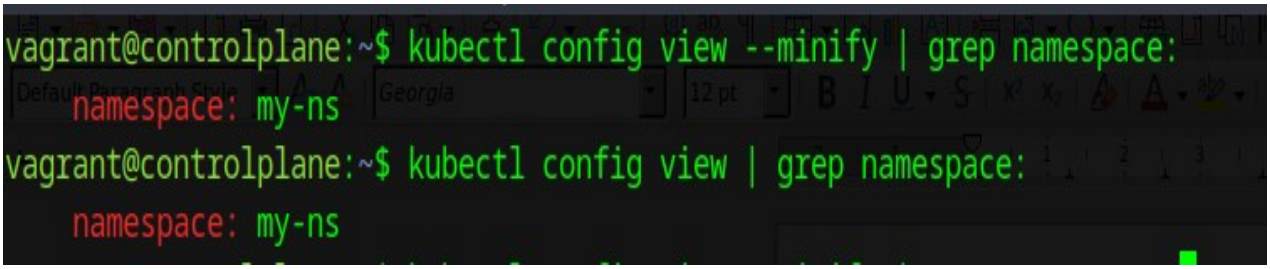
Set Default Namespace for kubectl Commands

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
```

Verify the current context's namespace:

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



Step 6: Clean Up Resources

To delete the resources and the namespace you created:

```
kubectl delete -f pod.yaml
kubectl delete -f svcc.yaml
kubectl delete namespace my-ns
```

Ensure that the namespace and all its resources are deleted:

```
kubectl get namespaces
```

```
vagrant@controlplane:~$ kubectl get ns
NAME                   STATUS   AGE
default                Active   7d19h
kube-node-lease        Active   7d19h
kube-public            Active   7d19h
kube-system            Active   7d19h
kubernetes-dashboard   Active   7d19h
vagrant@controlplane:~$ kubectl get pods
No resources found in my-ns namespace.
```