Lab Exercise 6- Create POD in Kubernetes

Name – Sujal Bhandari
SAP_ID – 500106865
Roll_No – R2142220181

Objective:

- Understand the basic structure and syntax of a Kubernetes Pod definition file (YAML).
- Learn to create, inspect, and delete a Pod in a Kubernetes cluster.

Prerequisites

- Kubernetes Cluster: You need a running Kubernetes cluster. You can set up a local cluster using tools like Minikube or kind, or use a cloud-based Kubernetes service.
- kubectl: Install and configure kubectl to interact with your Kubernetes cluster.
- Basic Knowledge of YAML: Familiarity with YAML format will be helpful as Kubernetes resource definitions are written in YAML.
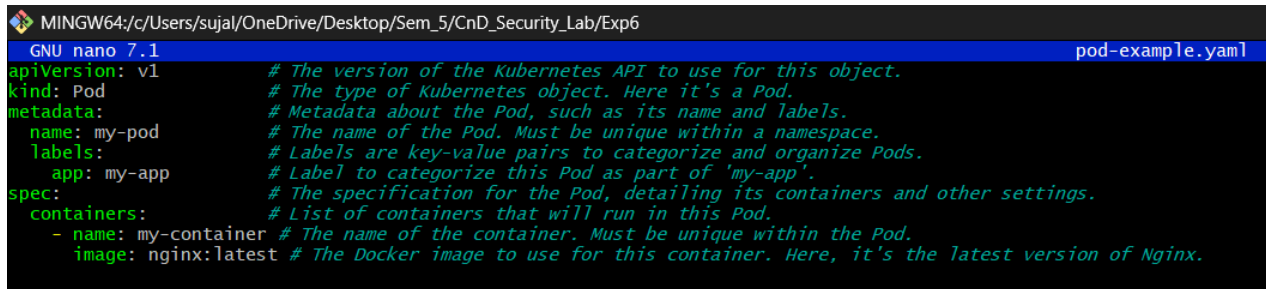
Step-by-Step Guide

Step 1: Create a YAML File for the Pod

We'll create a Pod configuration file named **pod-example.yaml**

```
apiVersion: v1        # The version of the Kubernetes API to use for this object.
kind: Pod             # The type of Kubernetes object. Here it's a Pod.
metadata:             # Metadata about the Pod, such as its name and labels.
  name: my-pod        # The name of the Pod. Must be unique within a namespace.
  labels:             # Labels are key-value pairs to categorize and organize Pods.
    app: my-app       # Label to categorize this Pod as part of 'my-app'.
```

```
spec:            # The specification for the Pod, detailing its containers and other settings.
  containers:        # List of containers that will run in this Pod.
   - name: my-container # The name of the container. Must be unique within the Pod.
     image: nginx:latest # The Docker image to use for this container. Here, it's the latest
version of Nginx.
```

```
 GNU nano 7.1                                                        pod-example.yaml
apiVersion: v1          # The version of the Kubernetes API to use for this object.
kind: Pod               # The type of Kubernetes object. Here it's a Pod.
metadata:               # Metadata about the Pod, such as its name and labels.
  name: my-pod          # The name of the Pod. Must be unique within a namespace.
  labels:               # Labels are key-value pairs to categorize and organize Pods.
    app: my-app         # Label to categorize this Pod as part of 'my-app'.
spec:                   # The specification for the Pod, detailing its containers and other settings.
  containers:           # List of containers that will run in this Pod.
    - name: my-container # The name of the container. Must be unique within the Pod.
      image: nginx:latest # The Docker image to use for this container. Here, it's the latest version of Nginx.
```

Explanation of the YAML File

- apiVersion: Specifies the version of the Kubernetes API to use. For Pods, it's typically v1.

- kind: The type of object being created. Here it's a Pod.

- metadata: Provides metadata about the object, including name and labels. The name must be unique within the namespace, and labels help in identifying and organizing Pods.

- spec: Contains the specifications of the Pod, including:
  - containers: Lists all containers that will run inside the Pod. Each container needs:
    - name: A unique name within the Pod.
    - image: The Docker image to use for the container.
    - ports: The ports that this container exposes.
    - env: Environment variables passed to the container.

Step 2: Apply the YAML File to Create the Pod

Use the kubectl apply command to create the Pod based on the YAML configuration file.

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


This command tells Kubernetes to create a Pod as specified in the pod-example.yaml file.

Step 3: Verify the Pod Creation

To check the status of the Pod and ensure it's running, use:

```
kubectl get pods
```


This command lists all the Pods in the current namespace, showing their status, restart count, and other details.

You can get detailed information about the Pod using:

```
kubectl describe pod my-pod
```

```
sujal@HP-Victus MINGW64 ~/OneDrive/Desktop/Sem_5/CnD_Security_Lab/Exp6
$ kubectl describe pod my-pod
Name:             my-pod
Namespace:        default
Priority:         0
Service Account:  default
Node:             docker-desktop/192.168.65.3
Start Time:       Fri, 25 Oct 2024 11:45:09 +0530
Labels:           app=my-app
Annotations:      <none>
Status:           Running
IP:               10.1.0.6
IPs:
  IP:  10.1.0.6
Containers:
  my-container:
    Container ID:   docker://1f04485d761cb642e350bc30dbf8a4b6b375c2f5a912a5189c8dc15831f06dce
    Image:          nginx:latest
    Image ID:       docker-pullable://nginx@sha256:28402db69fec7c17e179ea87882667f1e054391138f77ffaf0c3eb388efc3ffb
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Fri, 25 Oct 2024 11:45:18 +0530
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-tn7mk (ro)
Conditions:
  Type                        Status
  PodReadyToStartContainers   True
  Initialized                 True
  Ready                       True
  ContainersReady             True
  PodScheduled                True
Volumes:
  kube-api-access-tn7mk:
    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

Events:
  Type    Reason     Age    From               Message
  ----    ------     ----   ----               -------
  Normal  Scheduled  40s    default-scheduler  Successfully assigned default/my-pod to docker-desktop
  Normal  Pulling    40s    kubelet            Pulling image "nginx:latest"
  Normal  Pulled     31s    kubelet            Successfully pulled image "nginx:latest" in 8.5s (8.5s including waiting)
  Normal  Created    31s    kubelet            Created container my-container
  Normal  Started    31s    kubelet            Started container my-container
```

This command provides detailed information about the Pod, including its events, container specifications, and resource usage.

Step 4: Interact with the Pod

You can interact with the running Pod in various ways, such as accessing the logs or executing commands inside the container.

View Logs: To view the logs of the container in the Pod:
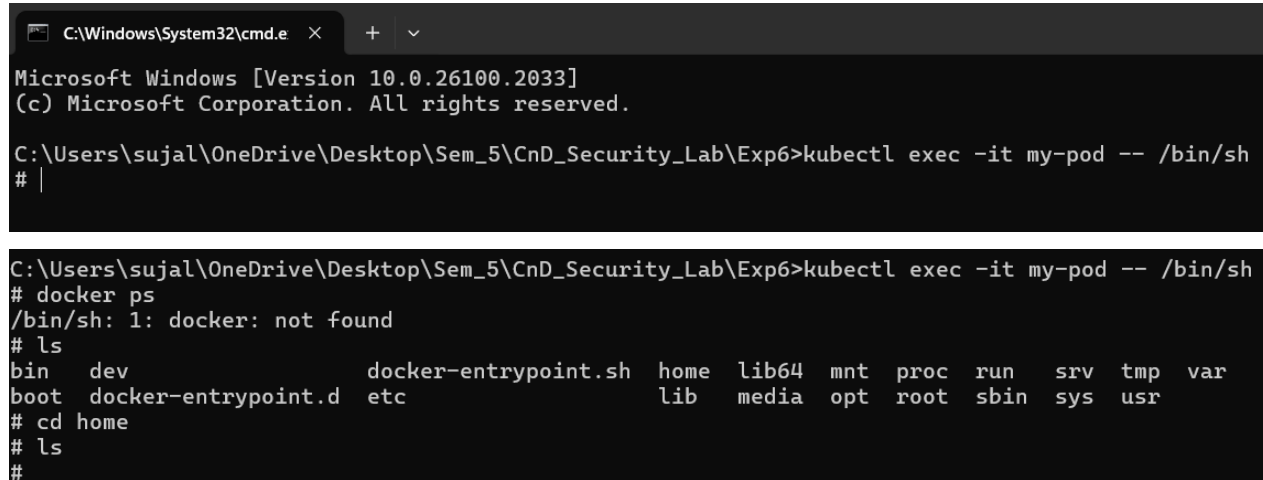
```
kubectl logs my-pod
```

Execute a Command: To run a command inside the container:

```
kubectl exec -it my-pod -- /bin/bash
```


The -it flag opens an interactive terminal session inside the container, allowing you to run commands.

Step 5: Delete the Pod

To clean up and remove the Pod when you're done, use the following command:

```
kubectl delete pod my-pod
```

```
C:\Users\sujal\OneDrive\Desktop\Sem_5\CnD_Security_Lab\Exp6>kubectl delete pod my-pod
pod "my-pod" deleted
```

This command deletes the specified Pod from the cluster.