

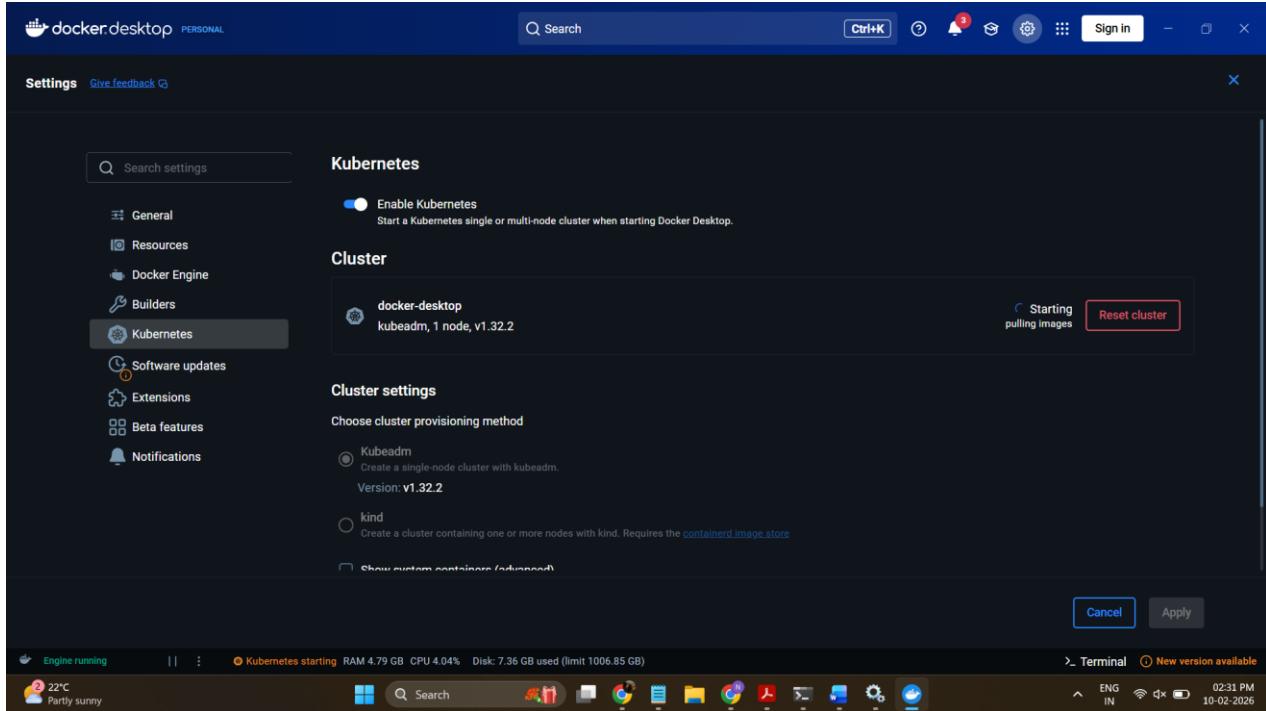
Lab Exercise 8- Create POD in Kubernetes

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.
-

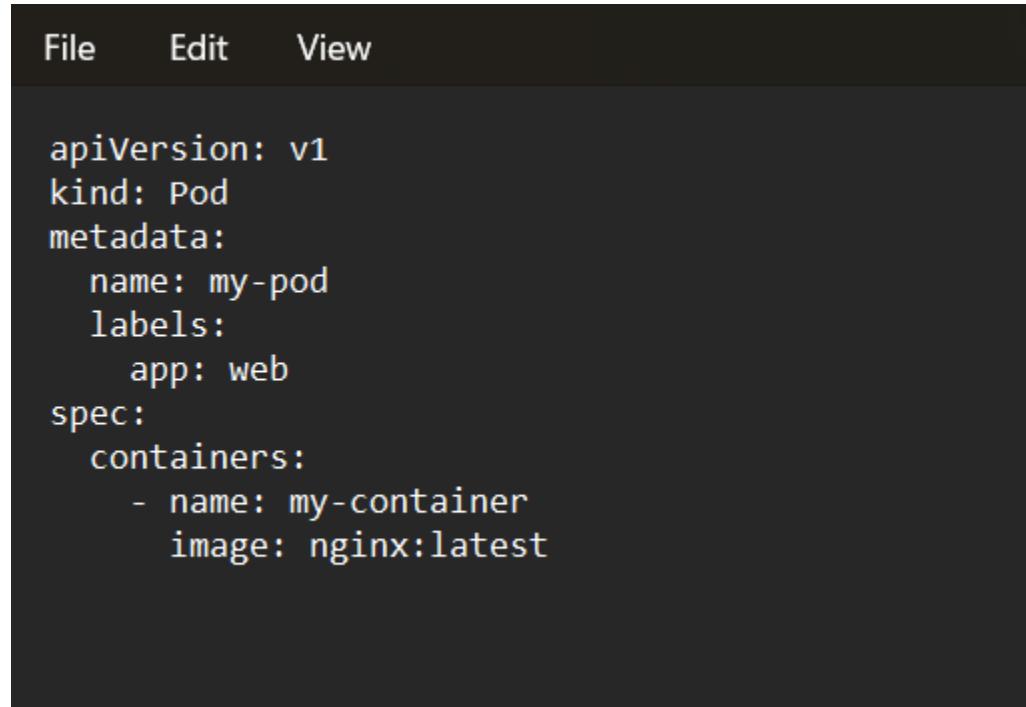


Step-by-Step Guide

Step 1: Create a YAML File for the Pod

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

```
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
  labels:
    app: web
spec:
  containers:
    - name: my-container
      image: nginx:latest
```



A screenshot of a terminal window with a dark background. At the top, there is a menu bar with three items: "File", "Edit", and "View". Below the menu, the terminal displays the YAML configuration file. The file defines a Pod named "my-pod" with a single container named "my-container" using the "nginx:latest" image.

```
File Edit View

apiVersion: v1
kind: Pod
metadata:
  name: my-pod
  labels:
    app: web
spec:
  containers:
    - name: my-container
      image: nginx:latest
```

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.

```
C:\Users\namit>kubectl apply -f pod-example.yaml
pod/my-pod created
```

```
C:\Users\namit>
```

Step 3: Verify the Pod Creation

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

```
kubectl get pods
```

```
C:\Users\namit>kubectl get pods
NAME      READY   STATUS            RESTARTS   AGE
my-pod    0/1     ContainerCreating   0          15s

C:\Users\namit>
```

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
```

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

```
C:\Users\namit>kubectl describe pod my-pod
Name:           my-pod
Namespace:      default
Priority:       0
Service Account: default
Node:          minikube/192.168.49.2
Start Time:    Tue, 10 Feb 2026 14:33:34 +0530
Labels:         app=web
Annotations:   <none>
Status:        Running
IP:            10.244.0.3
IPs:
  IP:  10.244.0.3
Containers:
  my-container:
    Container ID: docker://48c7351bdff1debf94b2134424f6bceff662a08615f4b565fba62f03911bd00c
    Image:        nginx:latest
    Image ID:    docker-pullable://nginx@sha256:341bf0f3ce6c5277d6002cf6e1fb0319fa4252add24ab6a0e262e0056d313208
    Port:         <none>
    Host Port:   <none>
    State:       Running
      Started:   Tue, 10 Feb 2026 14:34:05 +0530
    Ready:        True
    Restart Count: 0
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-szljf (ro)
Conditions:
  Type        Status
  PodReadyToStartContainers  True
  Initialized  True
  Ready        True
  ContainersReady  True
  PodScheduled  True
Volumes:
  kube-api-access-szljf:
    Type:        Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds: 3607
    ConfigMapName:  kube-root-ca.crt
    ConfigMapOptional: <nil>
Events:
  Type     Reason     Age   From           Message
  ----     ----     --   --            --
  Normal   Scheduled  37s  default-scheduler  Successfully assigned default/my-pod to minikube
  Normal   Pulling    36s  kubelet        Pulling image "nginx:latest"
  Normal   Pulled     6s   kubelet        Successfully pulled image "nginx:latest" in 29.206s (29.206s including waiting). Image size: 160850673 bytes.
  Normal   Created    6s   kubelet        Container created
  Normal   Started    6s   kubelet        Container started
C:\Users\namit>
```

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
```

```
C:\Users\namit>kubectl logs my-pod
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2026/02/10 09:04:05 [notice] 1#1: using the "epoll" event method
2026/02/10 09:04:05 [notice] 1#1: nginx/1.29.5
2026/02/10 09:04:05 [notice] 1#1: built by gcc 14.2.0 (Debian 14.2.0-19)
2026/02/10 09:04:05 [notice] 1#1: OS: Linux 6.6.87.2-microsoft-standard-WSL2
2026/02/10 09:04:05 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2026/02/10 09:04:05 [notice] 1#1: start worker processes
2026/02/10 09:04:05 [notice] 1#1: start worker process 29
2026/02/10 09:04:05 [notice] 1#1: start worker process 30
2026/02/10 09:04:05 [notice] 1#1: start worker process 31
2026/02/10 09:04:05 [notice] 1#1: start worker process 32
2026/02/10 09:04:05 [notice] 1#1: start worker process 33
2026/02/10 09:04:05 [notice] 1#1: start worker process 34
2026/02/10 09:04:05 [notice] 1#1: start worker process 35
2026/02/10 09:04:05 [notice] 1#1: start worker process 36
2026/02/10 09:04:05 [notice] 1#1: start worker process 37
2026/02/10 09:04:05 [notice] 1#1: start worker process 38
2026/02/10 09:04:05 [notice] 1#1: start worker process 39
2026/02/10 09:04:05 [notice] 1#1: start worker process 40

C:\Users\namit>
```

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.

```
C:\Users\namit>kubectl exec -it my-pod -- /bin/bash
root@my-pod:/#
```

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
```

This command deletes the specified Pod from the cluster.

```
C:\Users\namit>kubectl delete pod my-pod  
pod "my-pod" deleted
```

```
C:\Users\namit>
```