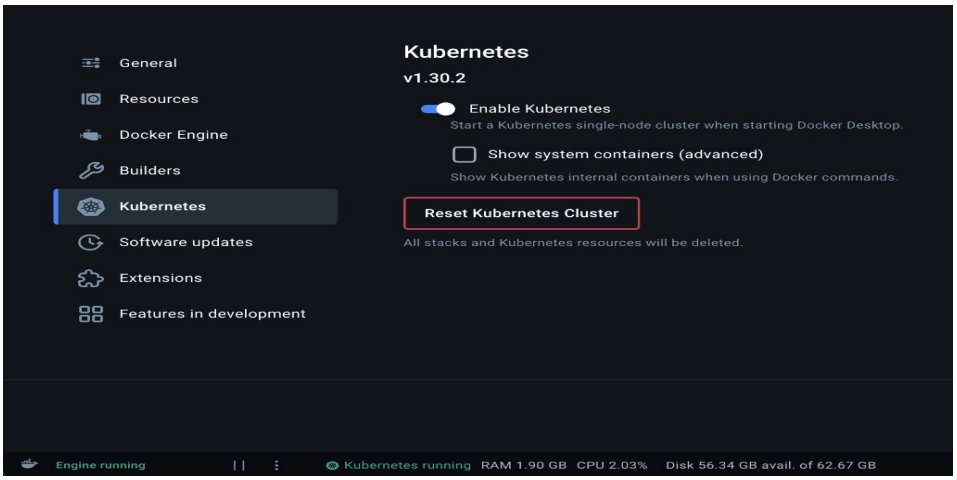
# Lab Exercise 6- Create POD in Kubernetes

## 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.




```
minikube version: v1.34.0
commit: 210b148df93a80eb872ecbeb7e35281b3c582c61

~ ..................................................................... 09:51:56 PM
❯ minikube start --driver=docker
😄 minikube v1.34.0 on Darwin 15.1 (arm64)
✨ Using the docker driver based on user configuration
📌 Using Docker Desktop driver with root privileges
👍 Starting "minikube" primary control-plane node in "minikube" cluster
🚜 Pulling base image v0.0.45 ...
💾 Downloading Kubernetes v1.31.0 preload ...
    > preloaded-images-k8s-v18-v1...:  307.61 MiB / 307.61 MiB  100.00% 1.62 Mi
    > gcr.io/k8s-minikube/kicbase...:  441.45 MiB / 441.45 MiB  100.00% 2.02 Mi
🔥 Creating docker container (CPUs=2, Memory=2200MB) ...
🐳 Preparing Kubernetes v1.31.0 on Docker 27.2.0 ...
    ▪ Generating certificates and keys ...
    ▪ Booting up control plane ...
    ▪ Configuring RBAC rules ...
🔗 Configuring bridge CNI (Container Networking Interface) ...
🔎 Verifying Kubernetes components...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟 Enabled addons: storage-provisioner, default-storageclass
🏄 Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

## 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.
```

**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

```
❯ vim pod-example.yaml

❯ kubectl apply -f pod-example.yaml
pod/my-pod created
```

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

```
❯ kubectl get pods
NAME      READY   STATUS              RESTARTS   AGE
my-pod    0/1     ContainerCreating   0          49s
```

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

```
❯ kubectl describe pod my-pod
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Wed, 06 Nov 2024 22:04:52 +0530
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-tzr6j (ro)
Conditions:
  Type                        Status
  PodReadyToStartContainers   True
  Initialized                 True
  Ready                       True
  ContainersReady             True
  PodScheduled                True
Volumes:
  kube-api-access-tzr6j:
    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  82s   default-scheduler  Successfully assigned default/my-pod to minikube
  Normal  Pulling    81s   kubelet            Pulling image "nginx:latest"
  Normal  Pulled     33s   kubelet            Successfully pulled image "nginx:latest" in 47.974s (47.974s including waiting). Image size: 196880357 bytes.
  Normal  Created    33s   kubelet            Created container my-container
  Normal  Started    33s   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

```
❯ 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
2024/11/06 16:34:52 [notice] 1#1: using the "epoll" event method
2024/11/06 16:34:52 [notice] 1#1: nginx/1.27.2
2024/11/06 16:34:52 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2024/11/06 16:34:52 [notice] 1#1: OS: Linux 6.10.11-linuxkit
2024/11/06 16:34:52 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2024/11/06 16:34:52 [notice] 1#1: start worker processes
2024/11/06 16:34:52 [notice] 1#1: start worker process 29
2024/11/06 16:34:52 [notice] 1#1: start worker process 30
2024/11/06 16:34:52 [notice] 1#1: start worker process 31
2024/11/06 16:34:52 [notice] 1#1: start worker process 32
2024/11/06 16:34:52 [notice] 1#1: start worker process 33
2024/11/06 16:34:52 [notice] 1#1: start worker process 34
2024/11/06 16:34:52 [notice] 1#1: start worker process 35
2024/11/06 16:34:52 [notice] 1#1: start worker process 36
```

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

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

```
❯ kubectl exec -it my-pod -- /bin/bash
root@my-pod:/# pwd
/
root@my-pod:/# ls -l
total 64
lrwxrwxrwx   1 root root    7 Oct 16 00:00 bin -> usr/bin
drwxr-xr-x   2 root root 4096 Aug 14 16:10 boot
drwxr-xr-x   5 root root  360 Nov  6 16:34 dev
drwxr-xr-x   1 root root 4096 Oct 17 13:33 docker-entrypoint.d
-rwxr-xr-x   1 root root 1620 Oct 17 13:33 docker-entrypoint.sh
drwxr-xr-x   1 root root 4096 Nov  6 16:34 etc
drwxr-xr-x   2 root root 4096 Aug 14 16:10 home
lrwxrwxrwx   1 root root    7 Oct 16 00:00 lib -> usr/lib
drwxr-xr-x   2 root root 4096 Oct 16 00:00 media
drwxr-xr-x   2 root root 4096 Oct 16 00:00 mnt
drwxr-xr-x   2 root root 4096 Oct 16 00:00 opt
dr-xr-xr-x 299 root root    0 Nov  6 16:34 proc
drwx------   2 root root 4096 Oct 16 00:00 root
drwxr-xr-x   1 root root 4096 Nov  6 16:34 run
lrwxrwxrwx   1 root root    8 Oct 16 00:00 sbin -> usr/sbin
drwxr-xr-x   2 root root 4096 Oct 16 00:00 srv
dr-xr-xr-x  11 root root    0 Nov  6 16:34 sys
drwxrwxrwt   2 root root 4096 Oct 16 00:00 tmp
drwxr-xr-x   1 root root 4096 Oct 16 00:00 usr
drwxr-xr-x   1 root root 4096 Oct 16 00:00 var
root@my-pod:/#
```

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
```

```
❯ kubectl delete pod my-pod
pod "my-pod" deleted
```

This command deletes the specified Pod from the cluster.