**Name: Harshita Singh**
**Class: D10C**
**Roll no. : 66**

# CVT Experiment 9

Setting and Installing Kubernetes, Pod Creation, Basic Commands, and Troubleshooting

## AIM:

To set up Kubernetes on a system, test basic commands, create a pod, identify its IP address, and perform basic troubleshooting using logs.

## Theory:

Kubernetes is a powerful open-source platform designed to automate the deployment, scaling, and management of containerized applications. It uses a **master-worker** architecture, where the **control plane (master node)** manages the cluster, and **worker nodes** run the actual applications inside **pods**.

Key components include:

- **Pod**: The smallest deployable unit in Kubernetes, which can contain one or more containers.
- **Node**: A physical or virtual machine that runs pods.
- **Kubectl**: A command-line tool used to interact with the Kubernetes cluster.

Kubernetes provides features like **load balancing**, **self-healing**, **automated rollouts and rollbacks**, and **service discovery**. In this practical, we set up a Kubernetes environment (e.g., using Minikube), test basic `kubectl` commands, create pods/nodes, fetch pod IPs, and troubleshoot issues using `kubectl logs`.

## Procedure:

### Step 1: Install Kubernetes Components

```
sudo apt update && sudo apt install -y apt-transport-https curl
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo
apt-key add -

echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" |
sudo tee /etc/apt/sources.list.d/kubernetes.list

sudo apt update
sudo apt install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
```

### Step 2: Initialize Kubernetes (Master Node)

```
sudo kubeadm init --pod-network-cidr=192.168.0.0/16
```

After successful init, note the `kubeadm join` command (used for adding worker nodes).

### Step 3: Set Up `kubectl` Access

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

### Step 4: Deploy a Network Add-On (e.g., Calico or Flannel)

```
kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml
```

**Running Basic Commands:**

1.Lists all the nodes (physical or virtual machines) in your Kubernetes cluster.

```
controlplane:~$ kubectl get nodes
NAME           STATUS   ROLES           AGE   VERSION
controlplane   Ready    control-plane   22d   v1.32.1
node01         Ready    <none>          22d   v1.32.1
controlplane:~$
controlplane:~$
```

2. Check client/server versions

```
controlplane:~$ kubectl version
Client Version: v1.32.1
Kustomize Version: v5.5.0
Server Version: v1.32.1
controlplane:~$
```

3.Get Pod IP Address

```
controlplane:~$ kubectl get nodes -o wide
NAME           STATUS   ROLES           AGE   VERSION   INTERNAL-IP    EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION     CONTAINER-RUNTIME
controlplane   Ready    control-plane   22d   v1.32.1   172.30.1.2     <none>        Ubuntu 24.04.1 LTS   6.8.0-51-generic   containerd://1.7.24
node01         Ready    <none>          22d   v1.32.1   172.30.2.2     <none>        Ubuntu 24.04.1 LTS   6.8.0-51-generic   containerd://1.7.24
controlplane:~$
```

4.Lists all the **pods** running in your current namespace.

```
controlplane:~$ kubectl get pods
No resources found in default namespace.
controlplane:~$
controlplane:~$
```

5.Lists all the **pods** running in your current specific namespace

```
controlplane:~$ kubectl get pods -n kube-system
NAME                                       READY   STATUS    RESTARTS      AGE
calico-kube-controllers-fdf5f5495-dgc76    1/1     Running   2 (12m ago)   22d
canal-9hc7x                                2/2     Running   2 (12m ago)   22d
canal-b5cnm                                2/2     Running   2 (12m ago)   22d
coredns-7695687499-2vdd4                   1/1     Running   1 (12m ago)   22d
coredns-7695687499-ltw2v                   1/1     Running   1 (12m ago)   22d
etcd-controlplane                          1/1     Running   3 (12m ago)   22d
kube-apiserver-controlplane                1/1     Running   2 (12m ago)   22d
kube-controller-manager-controlplane       1/1     Running   2 (12m ago)   22d
kube-proxy-f7jnk                           1/1     Running   2 (12m ago)   22d
kube-proxy-fbkjh                           1/1     Running   1 (12m ago)   22d
kube-scheduler-controlplane                1/1     Running   2 (12m ago)   22d
```

6.Displays information about the Kubernetes cluster (e.g., API server URL).

```
controlplane:~$ kubectl cluster-info
Kubernetes control plane is running at https://172.30.1.2:6443
CoreDNS is running at https://172.30.1.2:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
controlplane:~$
```

7.2.Deleting a pod

```
controlplane $ kubectl delete pod app
pod "app" deleted
controlplane $
controlplane $
controlplane $ kubectl get pods
No resources found in default namespace.
```

## **Deployings Pods(Imperitavely) :**

- Create a Pod

```
controlplane:~$ kubectl run web --image=nginx
pod/web created
controlplane:~$
```

```
controlplane:~$ kubectl get pods
NAME   READY   STATUS    RESTARTS   AGE
web    1/1     Running   0          39s
controlplane:~$
```

- Get Pod IP Address for using curl command

```
controlplane:~$ kubectl get pods -o wide
NAME   READY   STATUS    RESTARTS   AGE   IP           NODE     NOMINATED NODE   READINESS GATES
web    1/1     Running   0          78s   192.168.1.4  node01   <none>           <none>
controlplane:~$
```

- Curl to the received ip address of the running pod

```
controlplane:~$ curl -v 192.168.1.4
*   Trying 192.168.1.4:80...
* Connected to 192.168.1.4 (192.168.1.4) port 80
> GET / HTTP/1.1
> Host: 192.168.1.4
> User-Agent: curl/8.5.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: nginx/1.27.4
< Date: Mon, 14 Apr 2025 15:47:51 GMT
< Content-Type: text/html
< Content-Length: 615
< Last-Modified: Wed, 05 Feb 2025 11:06:32 GMT
< Connection: keep-alive
< ETag: "67a34638-267"
< Accept-Ranges: bytes
<
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
```

**Deploying a pod Declaratively(Using a Manifest file):**

```
controlplane:~$ kubectl run db --image=redis --dry-run=client -o yaml > db.yaml
controlplane:~$
controlplane:~$ cat db.yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: db
  name: db
spec:
  containers:
  - image: redis
    name: db
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
controlplane:~$
```

```
controlplane:~$ kubectl apply -f db.yaml
pod/db created
controlplane:~$
```

```
controlplane:~$ kubectl get pods
NAME   READY   STATUS    RESTARTS   AGE
db     1/1     Running   0          28s
web    1/1     Running   0          20m
controlplane:~$
```

### Getting Logs for Troubleshooting

```
controlplane $ kubectl logs web --follow
/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
2024/04/24 14:53:39 [notice] 1#1: using the "epoll" event method
2024/04/24 14:53:39 [notice] 1#1: nginx/1.25.5
2024/04/24 14:53:39 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2024/04/24 14:53:39 [notice] 1#1: OS: Linux 5.4.0-131-generic
2024/04/24 14:53:39 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2024/04/24 14:53:39 [notice] 1#1: start worker processes
2024/04/24 14:53:39 [notice] 1#1: start worker process 29
192.168.0.0 - - [24/Apr/2024:14:55:49 +0000] "GET / HTTP/1.1" 200 615 "-" "curl/7.68.0" "-"
```

## Conclusion:

In this experiment, Kubernetes was successfully set up on a host system using `kubeadm`. We tested basic commands, deployed a pod using the `kubectl` tool, and identified its IP. We also explored log-based troubleshooting to monitor and debug the pod's state