**KUBERNETES HANDS ON AWS EC2**

**PROJECT 1: SET UP A KUBERNETES CLUSTER ON AWS EC2**

**Objective: Install Kubernetes on EC2 instances (Ubuntu 22.04) and configure a cluster.**

**Step 1: Launch AWS EC2 Instances**

1. Go to **AWS Console → EC2 → Launch Instance**.
2. Select **Ubuntu 22.04 LTS**.
3. Choose **Instance Type**: t2.medium (at least 2 vCPUs and 4GB RAM).
4. Configure security group:
   * Allow **SSH** (Port 22)
   * Allow **Kubernetes API** (Port 6443)
   * Allow **NodePort Services** (Range 30000-32767)
5. Launch **two instances**:
   * **Master Node**
   * **Worker Node**
6. Connect to instances using SSH:

ssh -i your-key.pem ubuntu@your-ec2-public-ip

**Step 2: Install Docker on All Nodes**

Run these commands on both **Master** and **Worker** nodes:

sudo apt update -y

sudo apt install -y docker.io

sudo systemctl enable docker

sudo systemctl start docker

**Step 3: Install Kubernetes (kubectl, kubeadm, kubelet)**

Run the following commands **on both nodes**:

sudo apt update && sudo apt install -y apt-transport-https ca-certificates curl

sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg https://packages.cloud.google.com/apt/doc/apt-key.gpg

echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] 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 systemctl enable kubelet

**Step 4: Initialize Kubernetes on Master Node**

On the **Master Node**, run:

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

After the installation completes, you’ll see a **join command** like:

kubeadm join <master-ip>:6443 --token <token> --discovery-token-ca-cert-hash sha256:<hash>

Copy this command. You’ll use it for the **Worker Node**.

**Step 5: Configure kubectl on Master Node**

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

Verify the cluster status:

kubectl get nodes

**Step 6: Join Worker Node to Cluster**

On the **Worker Node**, run the join command you copied earlier:

sudo kubeadm join <master-ip>:6443 --token <token> --discovery-token-ca-cert-hash sha256:<hash>

Back on the **Master Node**, check if the worker node joined:

kubectl get nodes

**EXERCISE 2: DEPLOY AN NGINX POD AND SERVICE**

**Objective: Deploy an Nginx pod and expose it with a NodePort service.**

**Step 1: Deploy an Nginx Pod**

On the **Master Node**, create a pod:

kubectl run nginx-pod --image=nginx --restart=Never

**Step 2: Verify the Pod**

kubectl get pods

kubectl describe pod nginx-pod

**Step 3: Expose Pod as a NodePort Service**

kubectl expose pod nginx-pod --type=NodePort --port=80

**Step 4: Get Service Details**

kubectl get svc nginx-pod

Find the **NodePort** (e.g., 31234).

**Step 5: Access the Nginx Service**

On your browser, open:

php-template

CopyEdit

http://<EC2-Public-IP>:<NodePort>

You should see the **Nginx welcome page**.

**Step 6: Cleanup**

kubectl delete pod nginx-pod

kubectl delete svc nginx-pod

**EXERCISE 3: DEPLOY AN NGINX DEPLOYMENT AND SCALE IT**

**Objective: Deploy an Nginx deployment and scale it up.**

**Step 1: Create a Deployment**

kubectl create deployment nginx-deployment --image=nginx

**Step 2: Verify the Deployment**

kubectl get deployments

kubectl get pods

**Step 3: Scale the Deployment to 3 Replicas**

kubectl scale deployment nginx-deployment --replicas=3

**Step 4: Verify the Scaling**

kubectl get pods -o wide

**Step 5: Cleanup**

kubectl delete deployment nginx-deployment

**EXERCISE 4: DEPLOY A STATEFUL SET FOR A DATABASE**

**Objective: Deploy MySQL using Stateful Set and Persistent Volume.**

**Step 1: Create a Persistent Volume (PV) and Claim (PVC)**

Create mysql-pv.yaml:

apiVersion: v1

kind: PersistentVolume

metadata:

name: mysql-pv

spec:

capacity:

storage: 1Gi

accessModes:

- ReadWriteOnce

hostPath:

path: "/mnt/data"

---

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: mysql-pvc

spec:

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 1Gi

Apply it:

kubectl apply -f mysql-pv.yaml

**Step 2: Deploy MySQL Stateful Set**

Create mysql-statefulset.yaml:

apiVersion: apps/v1

kind: StatefulSet

metadata:

name: mysql

spec:

serviceName: "mysql"

replicas: 1

selector:

matchLabels:

app: mysql

template:

metadata:

labels:

app: mysql

spec:

containers:

- name: mysql

image: mysql:5.7

env:

- name: MYSQL\_ROOT\_PASSWORD

value: "mypassword"

volumeMounts:

- name: mysql-storage

mountPath: /var/lib/mysql

volumeClaimTemplates:

- metadata:

name: mysql-storage

spec:

accessModes: [ "ReadWriteOnce" ]

resources:

requests:

storage: 1Gi

Apply it:

kubectl apply -f mysql-statefulset.yaml

**Step 3: Verify the StatefulSet**

kubectl get statefulsets

kubectl get pods -o wide

**Step 4: Access MySQL Pod**

kubectl exec -it mysql-0 -- mysql -uroot -pmypassword

Inside MySQL shell, check databases:

SHOW DATABASES;

Exit MySQL shell:

exit;

**Step 5: Cleanup**

kubectl delete -f mysql-statefulset.yaml

kubectl delete -f mysql-pv.yaml