Lab 5: Kubernetes on AWS using EKS (Elastic Kubernetes Service)

Lab Title:

Deploying Applications to Kubernetes on AWS (EKS)

Objective:

Introduction to Amazon Elastic Kubernetes Service (EKS), create a cluster, deploy a containerized application, and manage it using `kubectl`.

Duration:

2.5 – 3 hours

Pre-requisites:

- AWS Free Tier account with **EKS, IAM, EC2,** and **VPC** access
- Completed Lab 3 (Docker image pushed to ECR)
- AWS CLI v2, kubectl, and eksctl installed (CloudShell can be used with adjustments)

Part A: Setup Kubernetes Tools (20 mins)

1. Launch AWS CloudShell

```
aws --version
kubectl version --client
eksctl version
```

If `kubectl` or `eksctl` is missing:

```
curl -LO https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz
mkdir -p ~/bin && tar -xzf eksctl_*.tar.gz -C ~/bin && export PATH=$PATH:~/bin
```

Part B: Create EKS Cluster (45 mins)

1. Create EKS Cluster using eksctl

```
eksctl create cluster \
  --name demo-cluster \
  --version 1.29 \
  --region us-east-1 \
  --nodegroup-name demo-nodes \
  --node-type t3.small \
  --nodes 2 \
  --managed
```

⚠ This step takes ~10–15 minutes. It creates:

- A VPC
- EKS control plane
- Worker nodes (EC2)

Part C: Configure Access and Test Cluster (15 mins)

1. Update kubeconfig to access the cluster

```
aws eks --region ap-south-1 update-kubeconfig --name demo-cluster
```

2. Verify cluster access

```
kubectl get nodes
kubectl get svc
```

Part D: Deploy Your Container App (40 mins)

1. Create Deployment YAML file (use nano / vi)

```
#cat <<EOF > deployment.yaml

apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: web
  template:
    metadata:
      labels:
        app: web
    spec:
      containers:
      - name: nginx
        image: <your-account-id>.dkr.ecr.ap-south-1.amazonaws.com/sample-web-
app:latest
        ports:
        - containerPort: 80

#EOF
```

2. Apply Deployment

```
kubectl apply -f deployment.yaml
kubectl get deployments
```

3. Expose the Deployment

```
kubectl expose deployment web-app --type=LoadBalancer --port=80
```

4. Get External IP (may take a few minutes)

```
kubectl get svc web-app
```

Use the **EXTERNAL-IP** to open the deployed app in your browser.

Part E: Scale and Manage Pods (20 mins)

1. Scale Deployment

```
kubectl scale deployment web-app --replicas=4
kubectl get pods
```

2. Rollout an Update (Optional)

```
kubectl set image deployment/web-app nginx=<new-image>
kubectl rollout status deployment/web-app
```

Part F: Clean-up (Optional)

1. Delete Kubernetes resources

```
kubectl delete service web-app
kubectl delete deployment web-app
```

2. Delete EKS Cluster

```
eksctl delete cluster --name demo-cluster --region us-east-1
```

Troubleshooting commands

1. To check if IAM role is assigned to the node-group

```
aws eks describe-nodegroup --cluster-name demo-cluster --nodegroup-name demo-nodes --region us-east-1 --query "nodegroup.nodeRole" --output text
```

2. To restart deployment

```
kubectl apply -f deployment.yaml

kubectl rollout restart deployment web-app
```

3. Monitor pods

```
kubectl get pods -w

kubectl get svc web-app
```