Kubernetes Task-2

Task Description:

Create the K8s EKS,further you have to do the deployment of the Nginx application and access the application outside the cluster.

➔ Preconditions:
- IAM user with Administrator access.
- AWS credentials
- EC2 attached with IAM role with EKS permission

➔ Step 1: Under EC2 install:
- AWS CLI v2
- Kubectl
- Eksctl

➔ Step 2: Configure your AWS credentials

➔ Step 3: Decide your cluster name and region

```
ubuntu@ip-172-31-42-154:~$ CLUSTER_NAME="my-eks-cluster"
REGION="ap-south-1"
NODEGROUP_NAME="ng-workers"
NODE_TYPE="t3.medium"
NODES=2
ubuntu@ip-172-31-42-154:~$ eksctl create cluster \
  --name ${CLUSTER_NAME} \
  --region ${REGION} \
  --nodegroup-name ${NODEGROUP_NAME} \
  --node-type ${NODE_TYPE} \
  --nodes ${NODES} \
  --nodes-min 1 \
  --nodes-max 3 \
  --managed
2025-09-07 06:08:01 [ℹ]  eksctl version 0.214.0
2025-09-07 06:08:01 [ℹ]  using region ap-south-1
```

➔ Step 4: Create EKS cluster with eksctl

```
ubuntu@ip-172-31-42-154:~$ eksctl create cluster \
  --name ${CLUSTER_NAME} \
  --region ${REGION} \
  --nodegroup-name ${NODEGROUP_NAME} \
  --node-type ${NODE_TYPE} \
  --nodes ${NODES} \
  --nodes-min 1 \
  --nodes-max 3 \
  --managed
```

- Running this command creates a VPC, subnets, EKS control plane and a managed nodegroup.

```
2025-09-07 06:20:48 [✓]  created 1 managed nodegroup(s) in cluster "my-eks-cluster"
2025-09-07 06:20:49 [ℹ]  kubectl command should work with "/home/ubuntu/.kube/config", try 'kubectl get nodes'
2025-09-07 06:20:49 [✓]  EKS cluster "my-eks-cluster" in "ap-south-1" region is ready
```

➔ Step 5: After successfully running validate your clusters and nodes

```
ubuntu@ip-172-31-42-154:~$ kubectl config current-context
terraform-user@my-eks-cluster.ap-south-1.eksctl.io
ubuntu@ip-172-31-42-154:~$ kubectl get nodes -o wide
NAME                                           STATUS   ROLES    AGE     VERSION              INTERNAL-IP     EXTERNAL-IP     OS-IMAGE                        KERNEL-VERS
ION                  CONTAINER-RUNTIME
ip-192-168-49-93.ap-south-1.compute.internal   Ready    <none>   2m51s   v1.32.8-eks-99d6cc0   192.168.49.93   13.201.90.59    Amazon Linux 2023.8.20250818   6.1.147-172
.266.amzn2023.x86_64   containerd://1.7.27
ip-192-168-91-87.ap-south-1.compute.internal   Ready    <none>   2m49s   v1.32.8-eks-99d6cc0   192.168.91.87   13.203.213.81   Amazon Linux 2023.8.20250818   6.1.147-172
.266.amzn2023.x86_64   containerd://1.7.27
ubuntu@ip-172-31-42-154:~$ eksctl get cluster --region ${REGION}
NAME           REGION         EKSCTL CREATED
my-eks-cluster  ap-south-1     True
```

➔ Step 6: Create Nginx deployment and service files(.yaml) and apply them.

```
ubuntu@ip-172-31-42-154:~$ nano nginx-deployment.yaml
ubuntu@ip-172-31-42-154:~$ nano nginx-service.yaml
ubuntu@ip-172-31-42-154:~$ kubectl apply -f nginx-deployment.yaml
deployment.apps/nginx-deployment created
ubuntu@ip-172-31-42-154:~$ kubectl apply -f nginx-service.yaml
service/nginx-service created
ubuntu@ip-172-31-42-154:~$ kubectl get svc nginx-service -o wide
NAME            TYPE           CLUSTER-IP      EXTERNAL-IP                                                                PORT(S)        AGE   SELECTOR
nginx-service   LoadBalancer   10.100.96.198   a10633a851b7f4b04a47277ae77d51fd-944977878.ap-south-1.elb.amazonaws.com   80:30902/TCP   21s   app=nginx
ubuntu@ip-172-31-42-154:~$ kubectl get svc nginx-service -o jsonpath='{.status.loadBalancer.ingress[0].hostname}{"\n"}'
a10633a851b7f4b04a47277ae77d51fd-944977878.ap-south-1.elb.amazonaws.com
ubuntu@ip-172-31-42-154:~$ kubectl get svc nginx-service
NAME            TYPE           CLUSTER-IP      EXTERNAL-IP                                                                PORT(S)        AGE
nginx-service   LoadBalancer   10.100.96.198   a10633a851b7f4b04a47277ae77d51fd-944977878.ap-south-1.elb.amazonaws.com   80:30902/TCP   69s
```

➔ Step 7: Get the external IP address to access the nginx.