Kubernetes Task-2

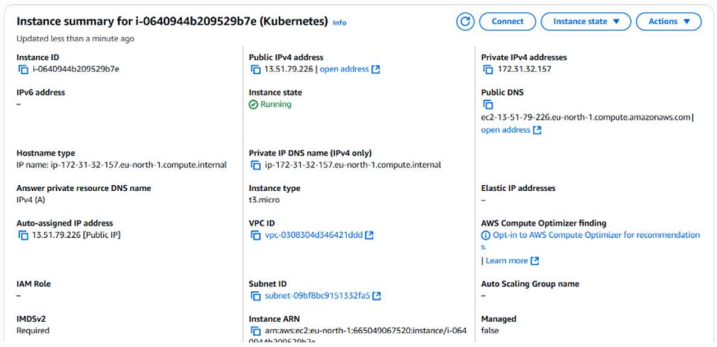
TASKS

Work Flow:

- Create an EC2 instance with the help of AWS Management Console with linux OS of required configuration and ensure that where the instance type should be t2.medium.
- Now, Connect an EC2 instance with an help of Windows Terminal or Gitbash or Vbox.
- To connect an EC2 instance the command is:
    - ssh -i "**key_file**" ec2-user@"**Public_IP_address**"

Key_file --- Key file of the instance with the extension .pem

Public_IP_address --- Public IP address of the instance.



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

<u>Step 1: Install an AWS CLI and kubectl</u>

Install An kubectl :

- ✓ To install an kubernetes in linux machine go to an official website by using below link.
- ✓ Link : https://kubernetes.io/docs/tasks/tools/install-kubectl/
- ✓ Now you can see the instructions given in the official page to install an kubernetes, follow all the steps to install.
- ✓ And also where the command as given below to install an Kubernetes from an official Page.

    - curl -LO "https://dl.k8s.io/release/**$(**curl -L -s https://dl.k8s.io/release/stable.txt**)**/bin/linux/amd64/kubectl"

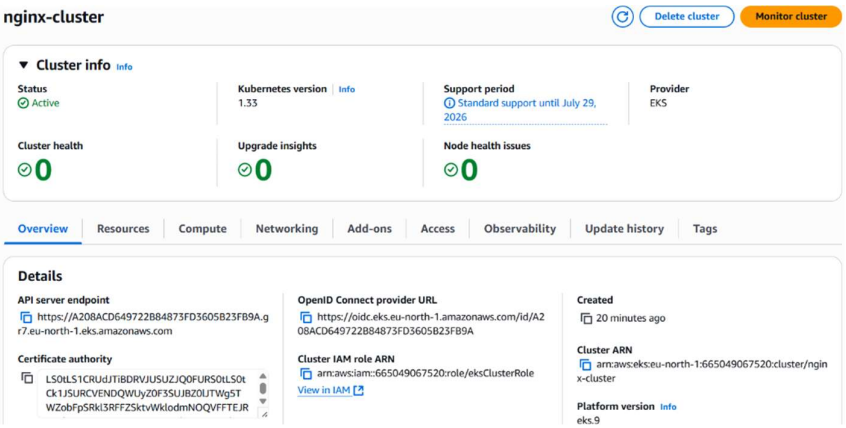
    - sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

```
[ec2-user@ip-172-31-36-72 ~]$ curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   138  100   138    0     0   2755      0 --:--:-- --:--:-- --:--:--  2816
100 57.3M  100 57.3M    0     0   176M      0 --:--:-- --:--:-- --:--:--  257M
```

```
[ec2-user@ip-172-31-36-72 ~]$ sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
[ec2-user@ip-172-31-36-72 ~]$ kubectl version
Client Version: v1.33.3
Kustomize Version: v5.6.0
The connection to the server localhost:8080 was refused - did you specify the right host or port?
```

Step 2: Creating an EKS Cluster Using AWS EKS Console

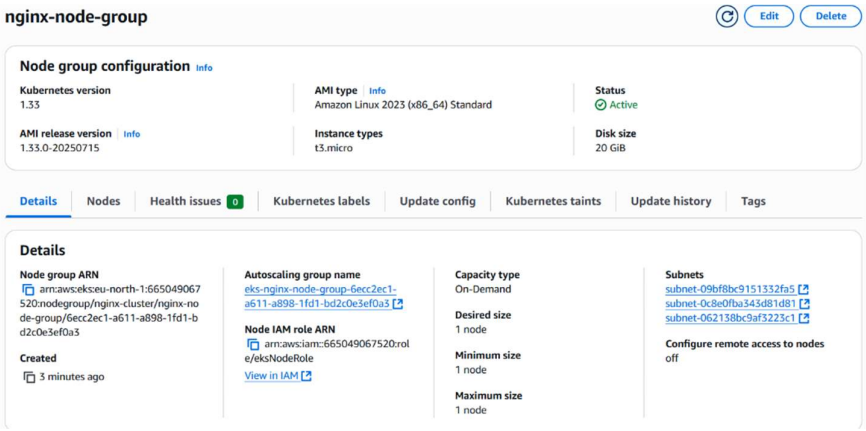
- ✓ **Log in to the AWS Management Console** and select the appropriate **region** (e.g., us-east-1).
- ✓ Navigate to **Elastic Kubernetes Service (EKS)** from the AWS services list.
- ✓ Click **"Add cluster" → "Create"**.
- ✓ Fill in the **Cluster Configuration**:
  - **Cluster name**: nginx-cluster
  - **Kubernetes version**: Choose the latest stable version
  - **Cluster Service Role**:
    - If not available, click to **create a new IAM role** in a separate tab.
    - Choose **EKS - Cluster** permissions.
    - Name the role **eksClusterRole**, create it, and return to EKS to select it.
- ✓ Click **Next** to proceed to the **Networking** step.
- ✓ Under **Networking**:
  - Create a new VPC or select an existing one.
  - Ensure that public and private subnets are properly selected.
  - Use default or custom security groups.
- ✓ Click **Next** for Logging (optional), enable if needed.
- ✓ Review the configuration and click **Create**.
- ✓ Wait until the cluster status changes to **"Active"** (~10–15 minutes).



Step 3: Creating a Node Group (Worker Nodes)

- ✓ Once the cluster is active, go to the **"Compute"** tab and click **"Add Node Group"**.
- ✓ **Node Group Configuration**:
  - Name: nginx-node-group
  - Role: Create or select an IAM role with EKS - Nodegroup permissions (e.g., eksNodeRole)

- ✓ **Node Group Details**:
  - Instance Type: t3.micro (Free Tier Eligible)
  - Number of Nodes: 1
  - Subnets: Choose subnets from the same VPC
- ✓ Leave the rest as default and click **Next**.
- ✓ Review and click **Create**.
- ✓ Wait for the node group status to change to **Active**.



Step 4: Connect EKS Cluster to kubectl

After the EKS cluster and node group have been successfully created, the next step is to configure the local kubectl tool to communicate with the newly created EKS cluster.

- ✓ Open the terminal on your EC2 instance (or any system where awscli and kubectl are installed).

```
[ec2-user@ip-172-31-32-157 ~]$ aws configure
AWS Access Key ID [None]: AKIAZVWADOAAHSJN66BV
AWS Secret Access Key [None]: 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
Default region name [None]: eu-north-1
Default output format [None]: json
```

- ✓ Run the following command to update the Kubernetes configuration file with the EKS cluster context:
  - **aws eks --region eu-north-1 update-kubeconfig --name nginx-cluster**

```
[ec2-user@ip-172-31-32-157 ~]$ aws eks --region eu-north-1 update-kubeconfig --name nginx-cluster
Added new context arn:aws:eks:eu-north-1:665049067520:cluster/nginx-cluster to /home/ec2-user/.kube/config
```

- ✓ Verify the connection by listing the nodes in the cluster:
  - **kubectl get nodes**

```
[ec2-user@ip-172-31-32-157 ~]$ aws eks --region eu-north-1 update-kubeconfig --name nginx-cluster
Updated context arn:aws:eks:eu-north-1:665049067520:cluster/nginx-cluster in /home/ec2-user/.kube/config
[ec2-user@ip-172-31-32-157 ~]$ kubectl get nodes
NAME                                          STATUS   ROLES    AGE   VERSION
ip-172-31-36-187.eu-north-1.compute.internal   Ready    <none>   17m   v1.33.0-eks-802817d
```

- ✓ This command should return a list of worker nodes in Ready status, confirming that the connection to the EKS cluster is successful and kubectl is properly configured.

Step 5: Deploy the Nginx Application

- ✓ To deploy the Nginx web server on the EKS cluster, create a Kubernetes Deployment configuration file named nginx-deployment.yaml.
- ✓ This file defines the desired number of replicas, container image, and exposed port for the application.
- ✓ Apply this configuration using the kubectl command, which will instruct the cluster to deploy the Nginx pods as defined.
- ✓ Create a file named nginx-deployment.yaml with the following content:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
```

- ✓ To Apply the deployment, the command is :
  - **kubectl apply -f nginx-deployment.yaml**

```
[ec2-user@ip-172-31-32-157 ~]$ touch nginx-service.yaml
[ec2-user@ip-172-31-32-157 ~]$ vi nginx-service.yaml
[ec2-user@ip-172-31-32-157 ~]$ kubectl apply -f nginx-service.yaml
service/nginx-service created
```

- ✓ To verify the deployment, The command is:
  - **kubectl get deployments**
  - **kubectl get pods**

```
[ec2-user@ip-172-31-32-157 ~]$ kubectl get svc
NAME            TYPE           CLUSTER-IP      EXTERNAL-IP                                                              PORT(S)        AGE
kubernetes      ClusterIP      10.100.0.1      <none>                                                                   443/TCP        36m
nginx-service   LoadBalancer   10.100.87.145   a653ae0303d254dbc9f5d58caef21cd7-615126490.eu-north-1.elb.amazonaws.com   80:32160/TCP   14s
```

Step 6: Expose the Nginx Application Externally

- ✓ To make the deployed Nginx application accessible outside the Kubernetes cluster, we create a Kubernetes Service of type LoadBalancer.

- ✓ This service provisions an external IP address through the cloud provider (AWS), allowing users to access the application over the internet.
- ✓ Create a file named nginx-service.yaml:

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  type: LoadBalancer
  selector:
    app: nginx
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
```

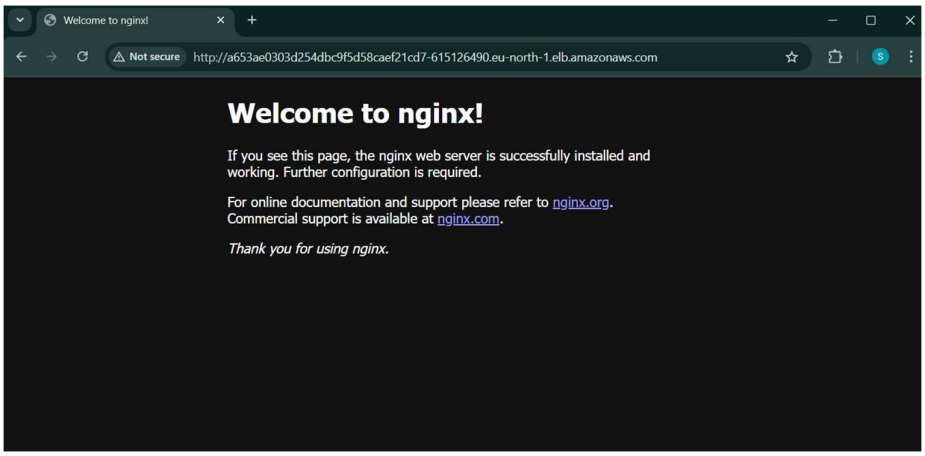
- ✓ To apply the created services, the command is:
  - **kubectl apply -f nginx-service.yaml**

```
[ec2-user@ip-172-31-32-157 ~]$ touch nginx-service.yaml
[ec2-user@ip-172-31-32-157 ~]$ vi nginx-service.yaml
[ec2-user@ip-172-31-32-157 ~]$ kubectl apply -f nginx-service.yaml
service/nginx-service created
```

- ✓ Now to get an external IP, The command is:
  - **kubectl get svc**

```
[ec2-user@ip-172-31-32-157 ~]$ kubectl get svc
NAME            TYPE           CLUSTER-IP      EXTERNAL-IP                                                               PORT(S)        AGE
kubernetes      ClusterIP      10.100.0.1      <none>                                                                    443/TCP        36m
nginx-service   LoadBalancer   10.100.87.145   a653ae0303d254dbc9f5d58caef21cd7-615126490.eu-north-1.elb.amazonaws.com   80:32160/TCP   14s
```

- ✓ Look for the EXTERNAL-IP column for nginx-service. Once it shows an IP address, open it in a browser , You should see the Nginx welcome page.



Step 7: Cleanup Resources (Optional)

- ✓ To avoid unnecessary charges, delete the EKS cluster and associated resources, The command is:
  - **eksctl delete cluster --name nginx-cluster --region us-east-1**

- ✓ Or manually delete via AWS Console:
  - Delete the Node Group
  - Delete the Cluster
  - Delete associated VPC and roles (if created manually)

Successfully created an EKS cluster on AWS, deployed an Nginx application, and exposed it externally using a LoadBalancer service.

********** TASK COMPLETED **********