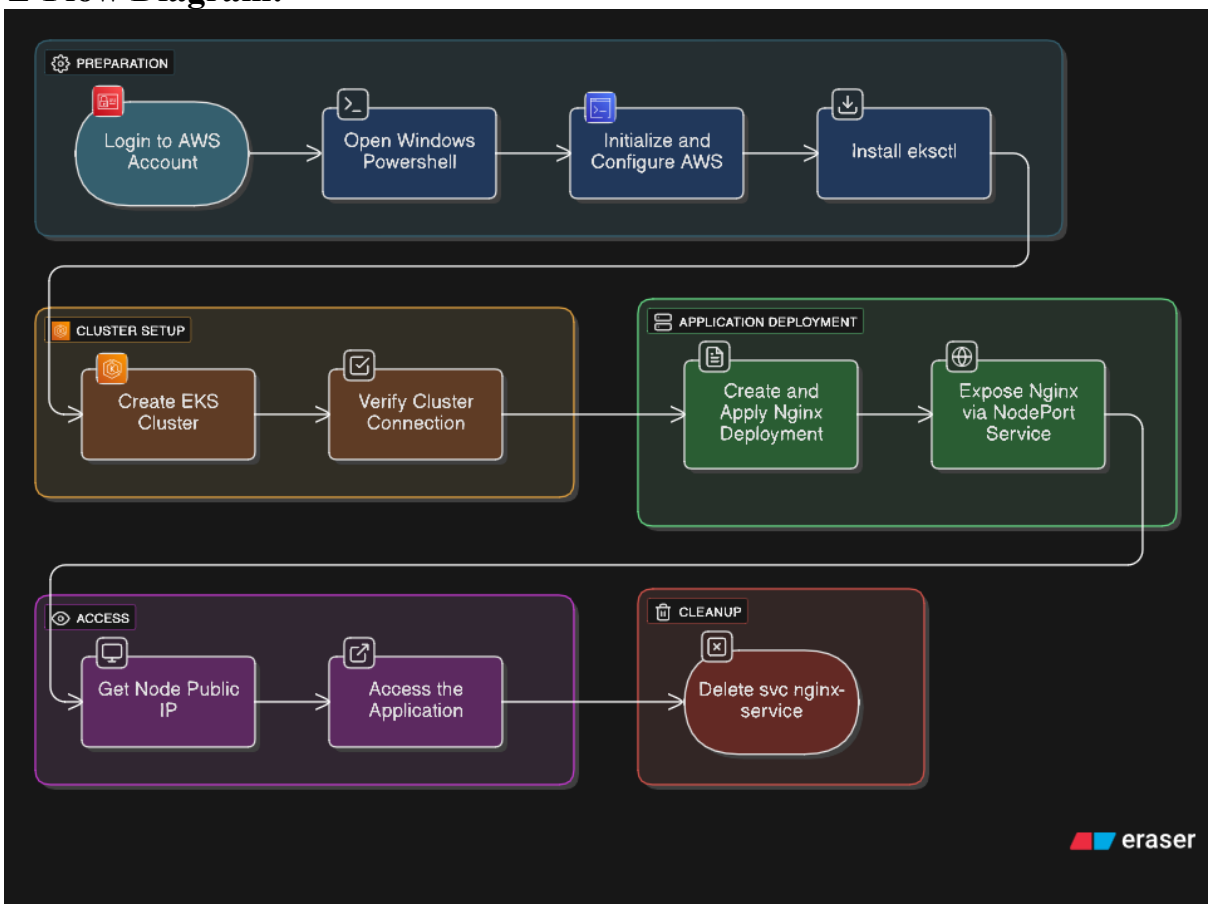
🚀 DEPLOYING A SIMPLE WEB APPLICATION ON KUBERNETES (AWS EKS)

This guide provides a **step-by-step walkthrough** of deploying an **Nginx web application** on **Amazon Elastic Kubernetes Service (EKS)** using the **AWS CLI**, **kubectl**, and **eksctl**.
It includes setup, cluster creation, deployment, service exposure, and cleanup.

GitHub Link: git@github.com:johnmicky1/Nginx-Web-App-Deployment-on-Kubernetes-AWS-EKS-.git

---

□ Flow Diagram:



Source: *https://www.eraser.io/diagramgpt*

The process includes:

1. Setting up AWS and IAM credentials
2. Installing required CLI tools
3. Creating and configuring Kubernetes manifest files
4. Launching an EKS cluster
5. Deploying and exposing an Nginx application
6. Accessing the application via browser
7. Cleaning up AWS resources

□ Prerequisites

A. AWS CLI

The AWS CLI (Command Line Interface) is a tool that allows users to manage AWS services from the command line. It's used to interact with AWS services, including EKS.

Use the link below to download and install a CLI under the section having **AWS CLI install and update instructions.**

For installation instructions, expand the section for your operating system.

https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html

---

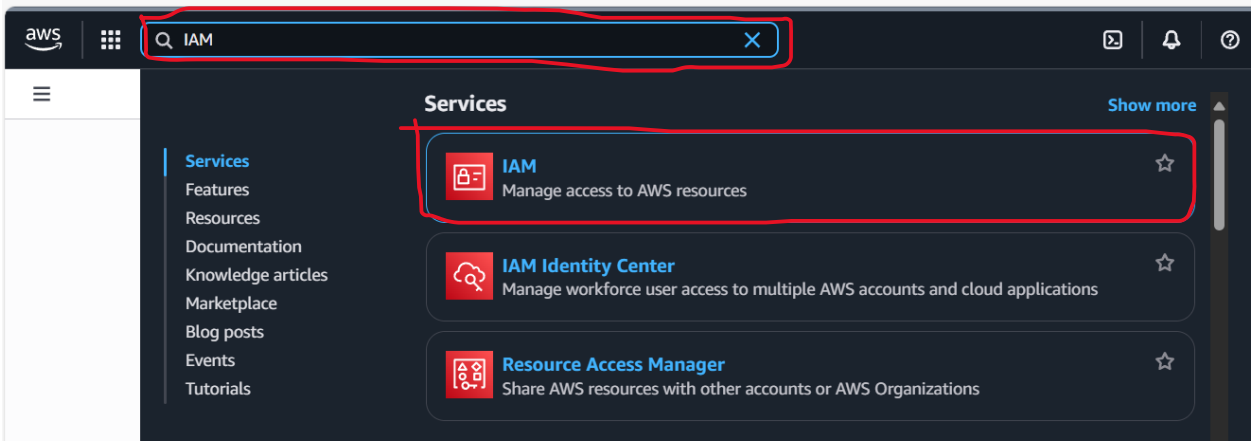
🥷 Step 1: Configure AWS IAM User

1.1. IAM User and setting up Kubernetes policies

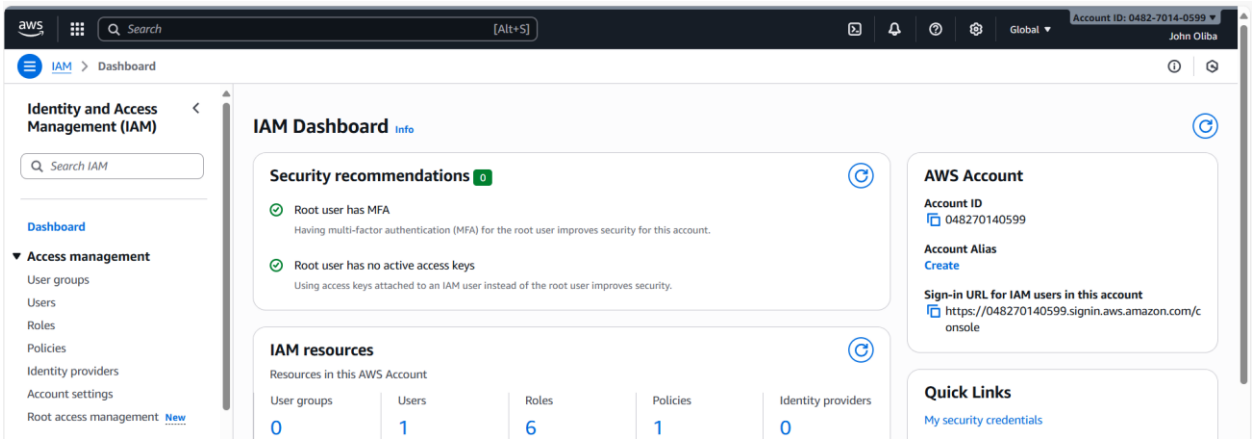
An IAM (Identity and Access Management) User is a user account that can be created within an AWS account. It's used to manage access to AWS resources. Add **AdministratorAccess, AmazonEKSClusterPolicy, EKSServicePolicy and AmazonEC2ContainerRegistryReadOnly** then create an **Access Key** and **Secret Access Key** using the steps below:

Create Access Key and Secret Access Key

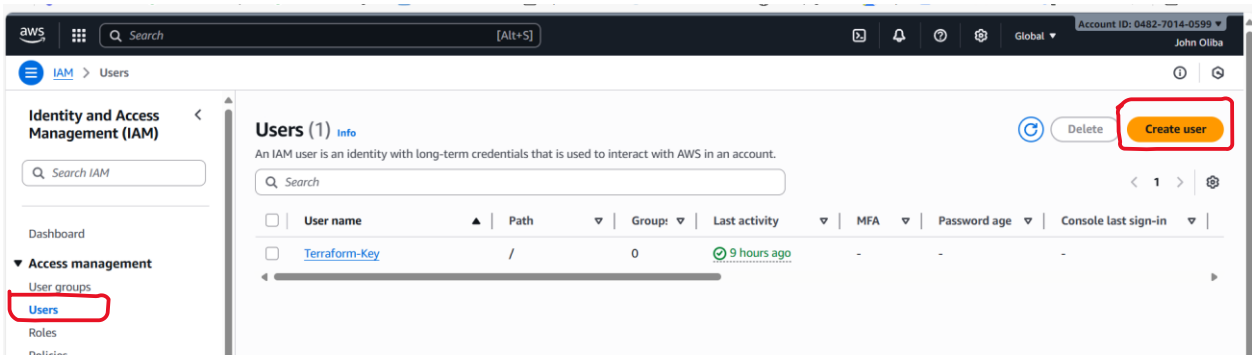
a) Log in to the AWS Management Console and type IAM in the search box



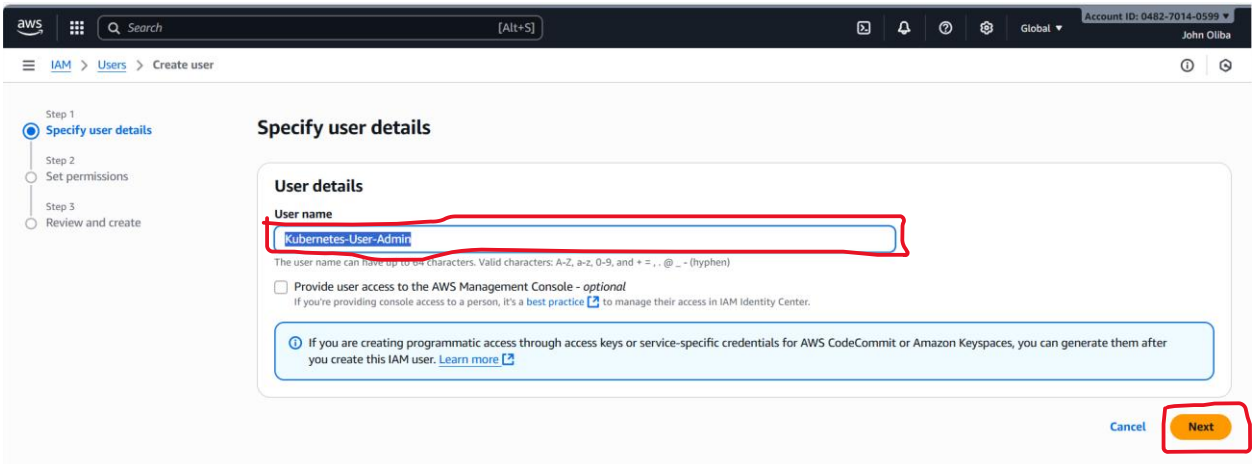
b) Navigate to the IAM (Identity and Access Management) dashboard.



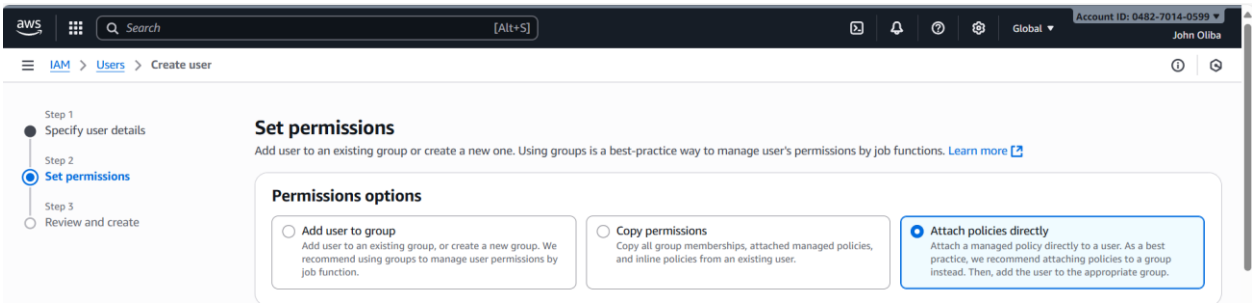
c) In the left sidebar, select Users and click on create user



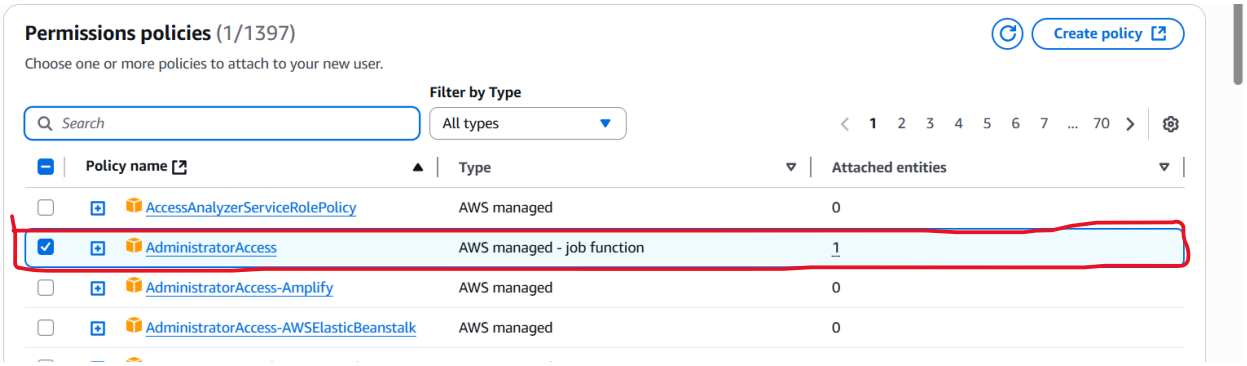
d) Specify User details by giving a user name and click next (e.g Kubernetes-User-Admin)



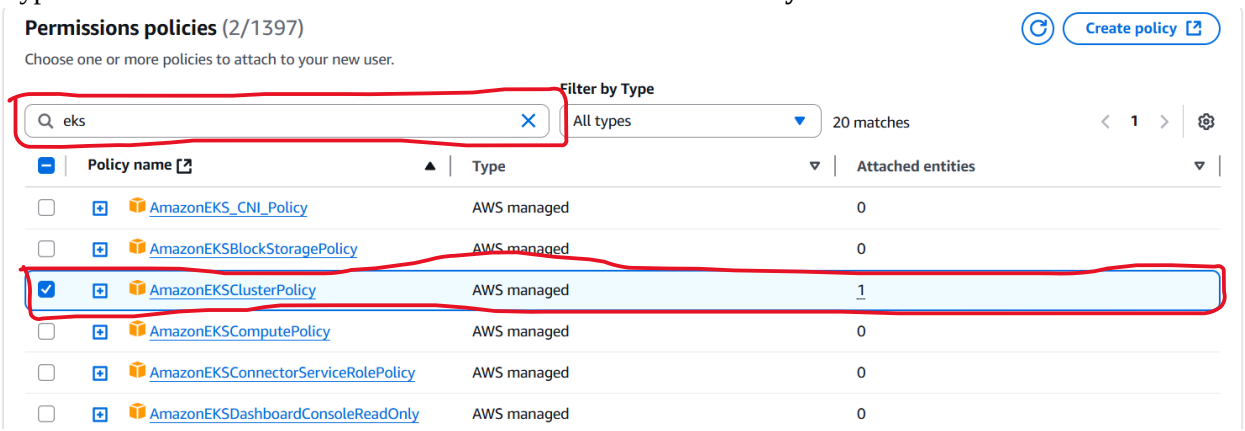
e) To set permissions; click on attach policies directly and scroll down



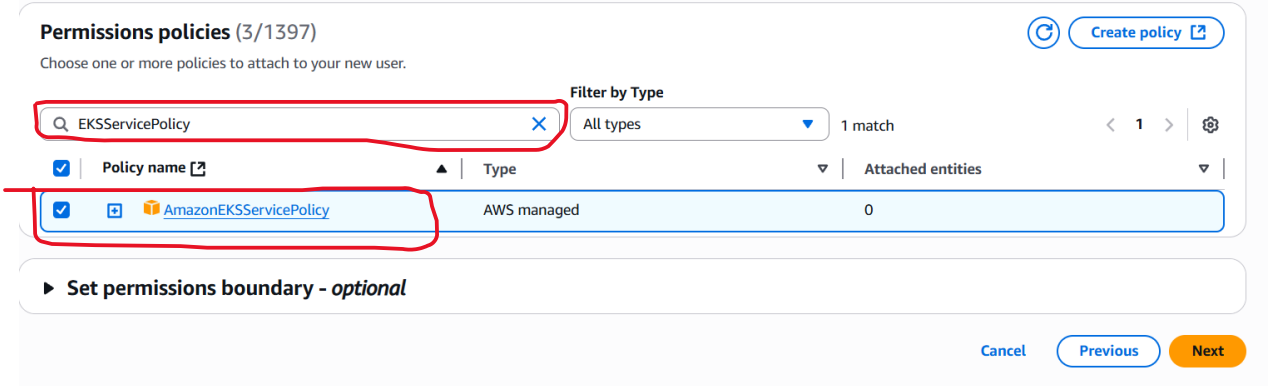
f) Select **AdministratorAccess**



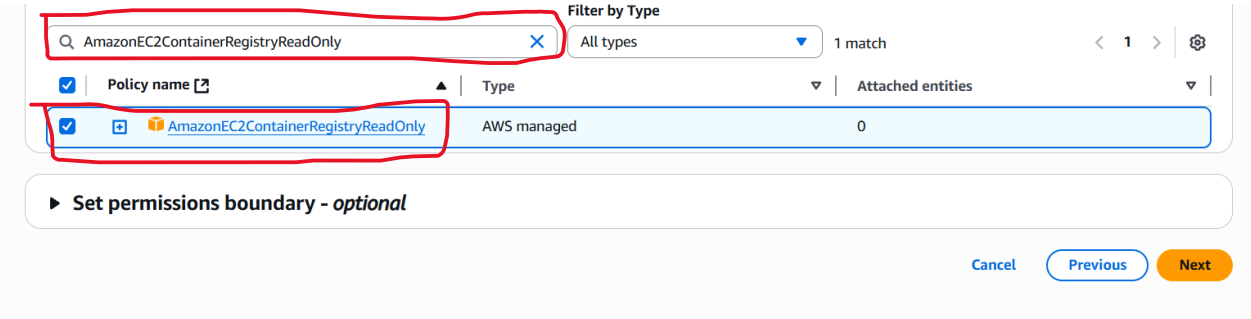
g) Type **EKS** in the search box and selec**t AmazonEKSClusterPolicy**



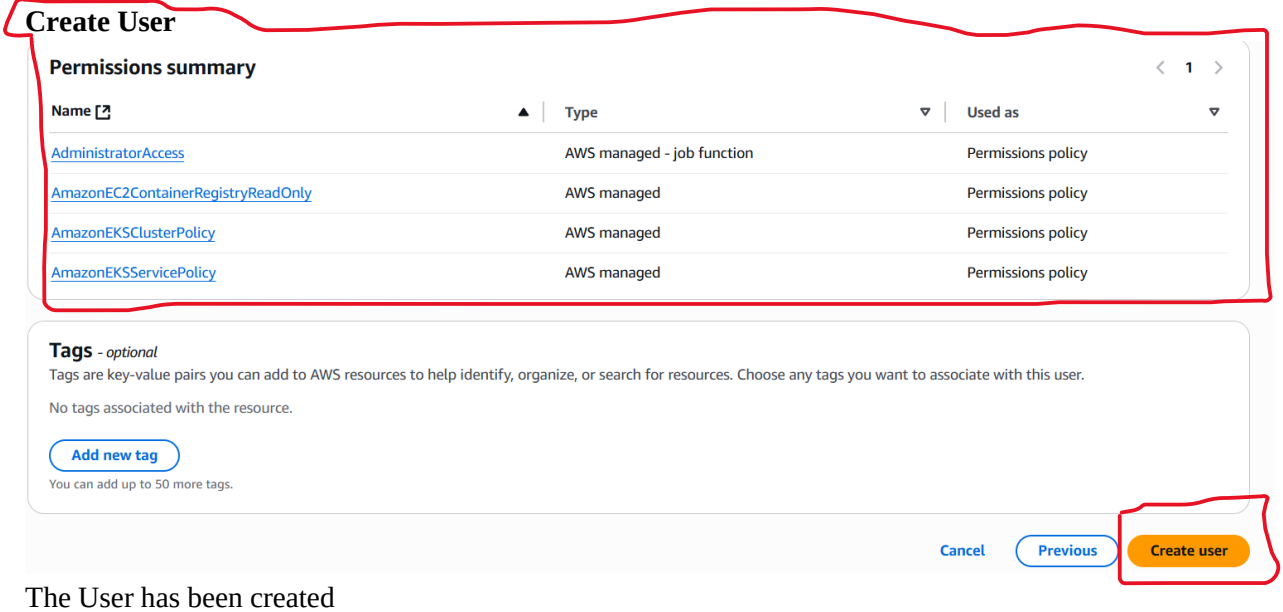
Type **EKSServicePolicy** in the search box and select **AmazonEKSServicePolicy**



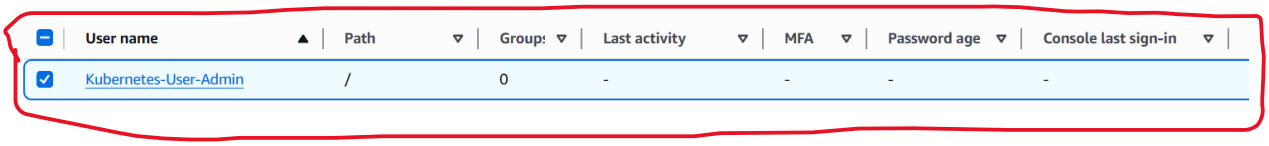
h) Type **AmazonEC2ContainerRegistryReadOnly** in the search box and select **AmazonEC2ContainerRegistryReadOnly** then click **Next**



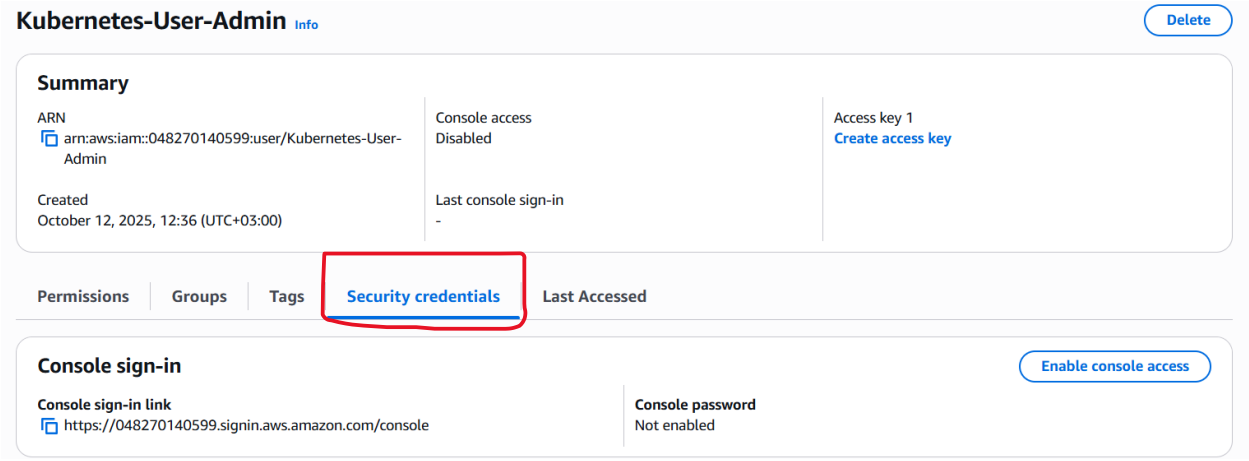
i) You should be able to see a summary of the permissions you have created then click on **Create User**



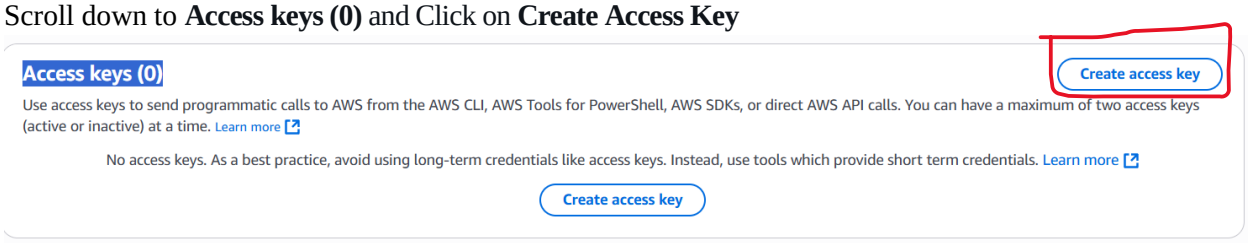
j) The User has been created



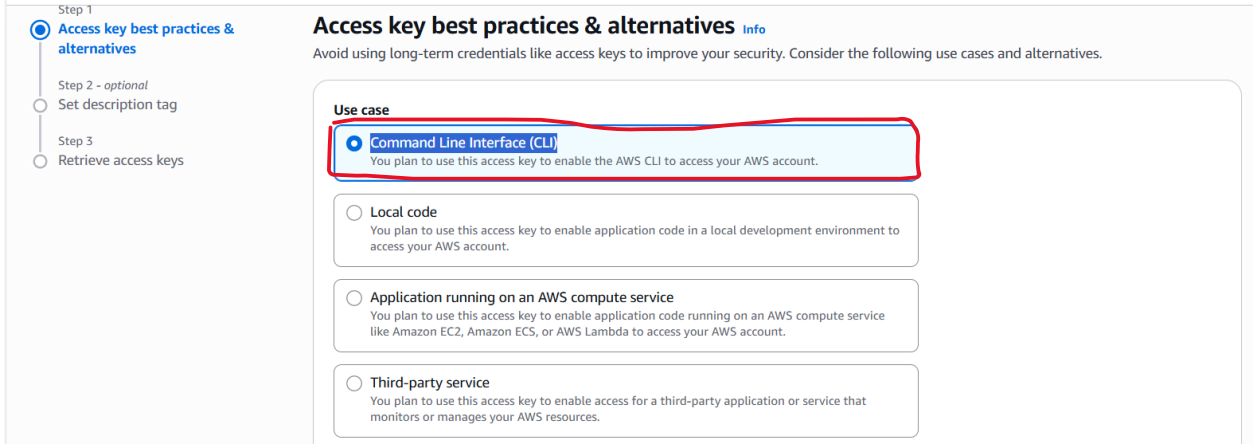
k) Click **Kubernetes-User-Admin** and Click on **Security Credentials** then scroll down

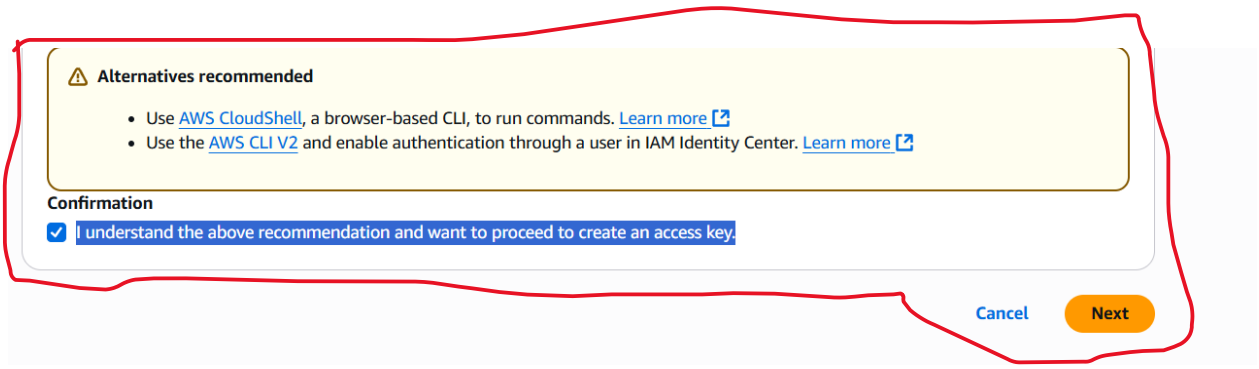


l) Scroll down to **Access keys (0)** and Click on **Create Access Key**

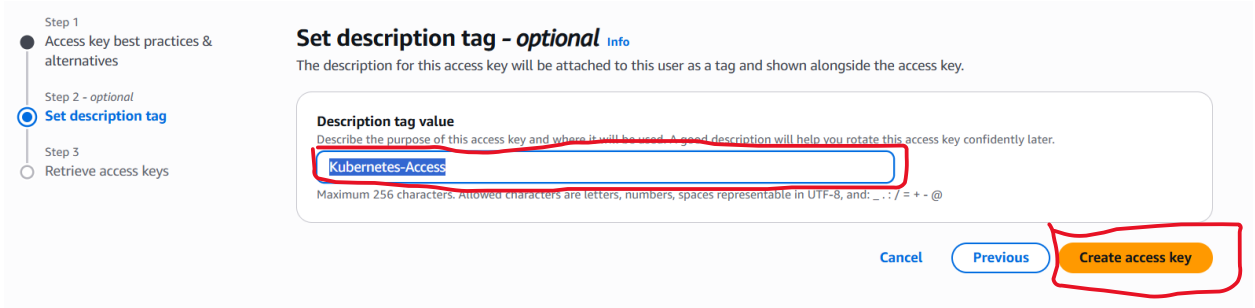


m) Select **Command Line Interface (CLI)** Scroll Down and Click Check box having **I understand the above recommendation and want to proceed to create an access key.** Then Click **Next**

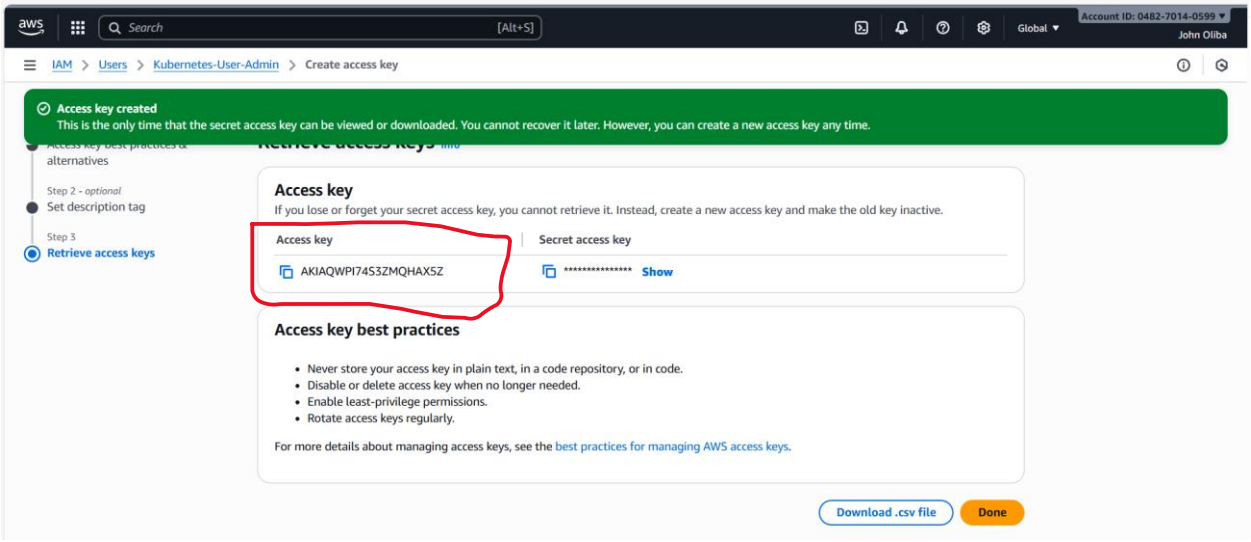


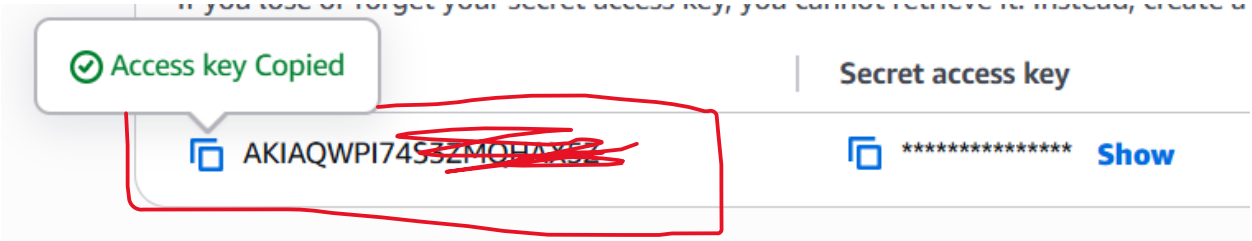


n) Set description tag – optional (Give it a name e.g Kubernetes-Access) and Click on **Create Access Key**

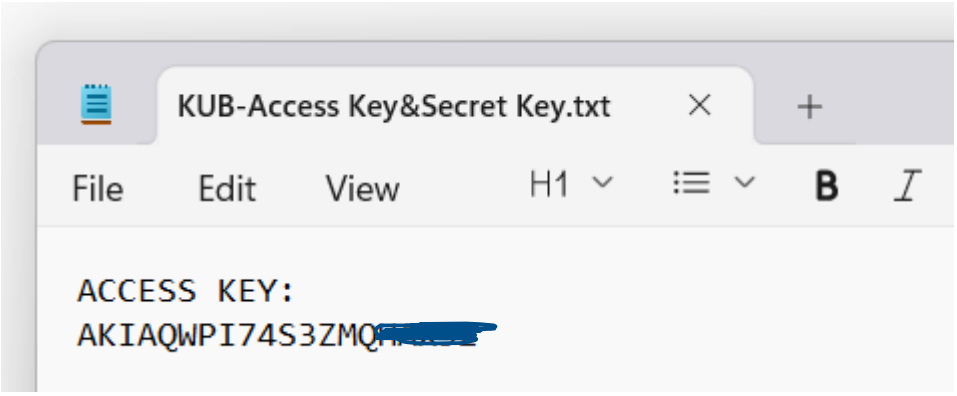


o) Save the Access Key in a sure location on your local machine because it will be necessary for Accessing the AWS CLI using your local machine.

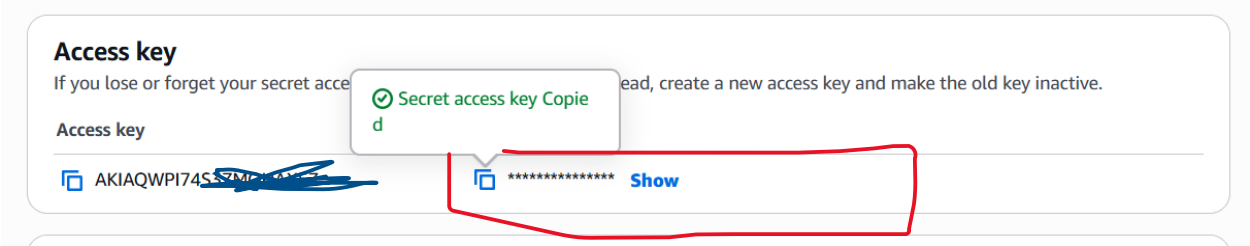


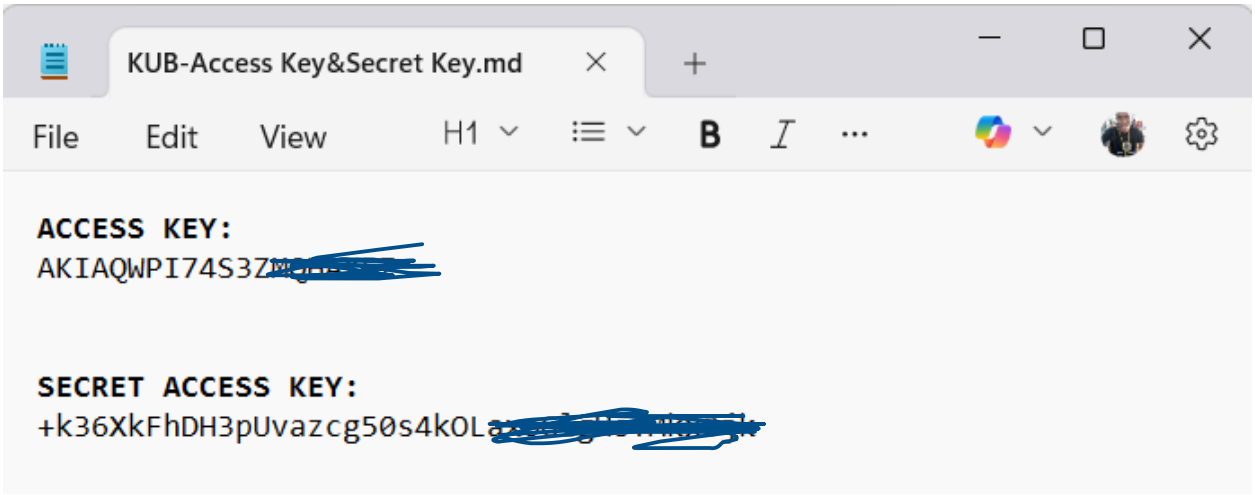


Open Note Pad, Type **ACCESS KEY** and paste the Access Key as illustrated in the Image below:

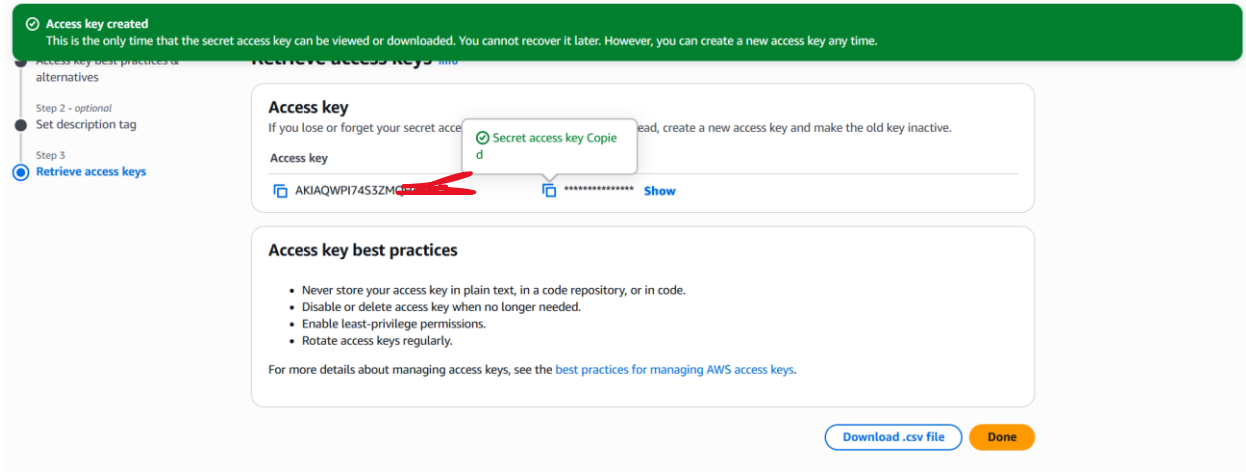


p) Type on the same notepad page **SECRET ACCESS KEY** then copy and paste it as seen below:





q) Download the CSV File As well and Save it in a secure location on your local machine.

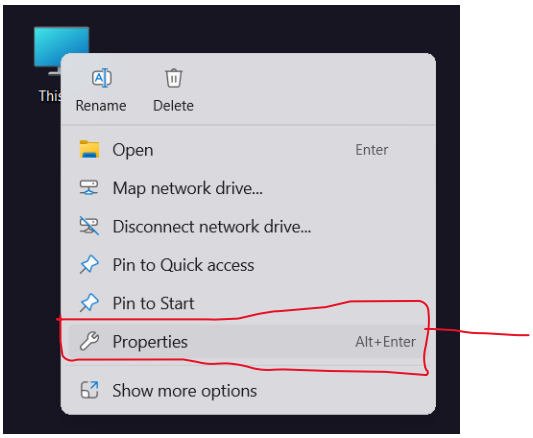


A C.S.V file will be download, make sure it is saved in a secure location

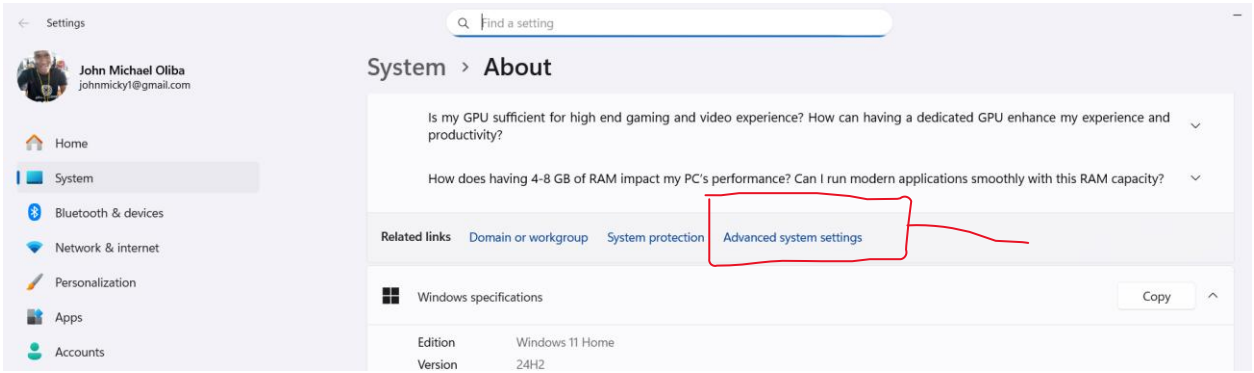
---

□ **Step 2:** Install the Tool for managing EKS clusters (eksctl)

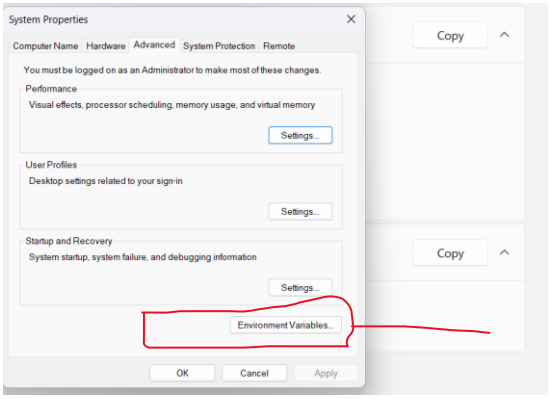
- Right-click This PC or Computer and select Properties.



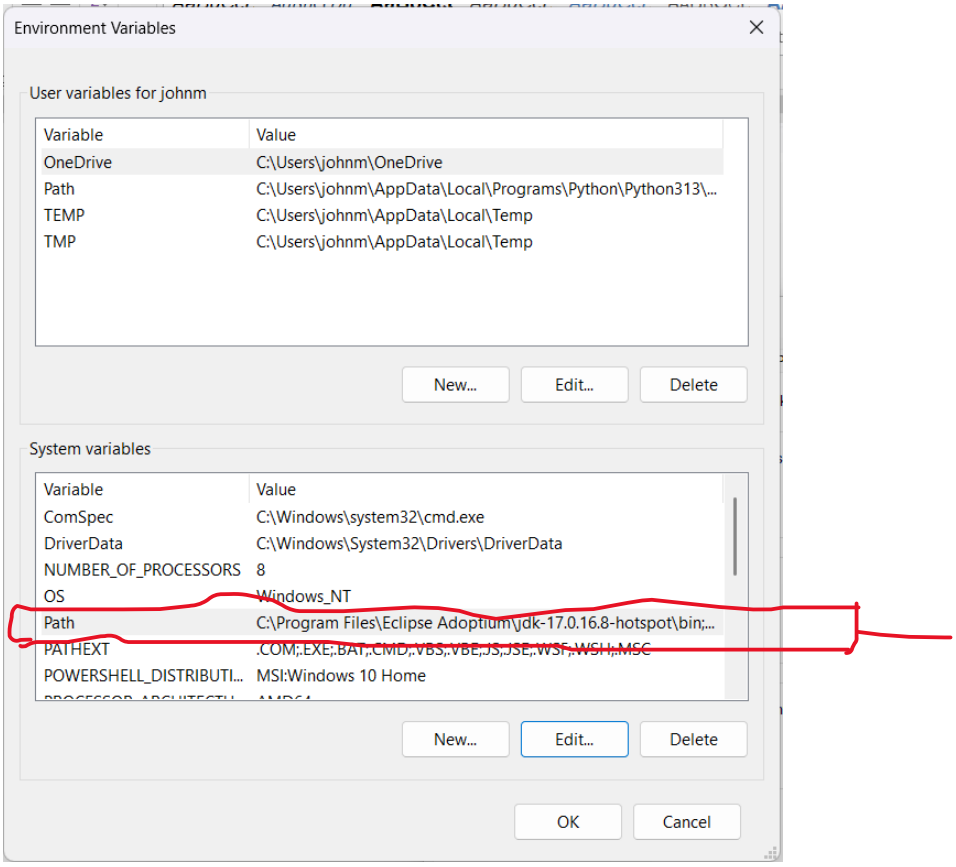
- Click on Advanced system settings on the left side.



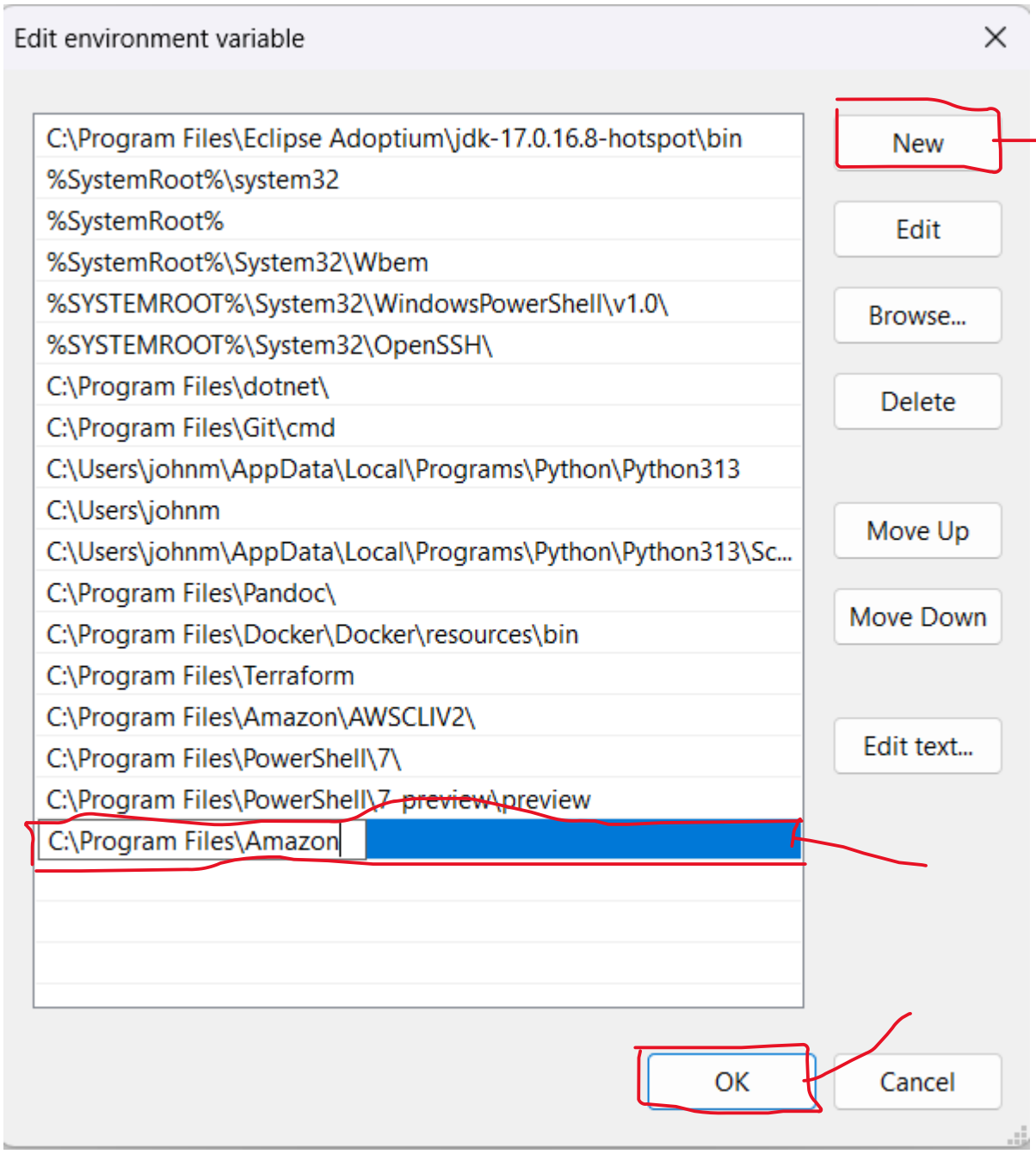
- Click on Environment Variables.



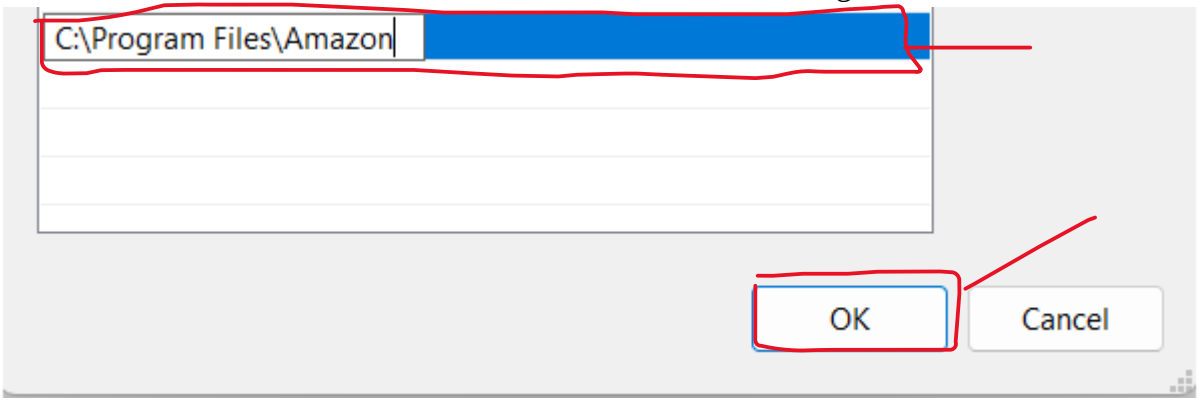
- Under System Variables, scroll down and find the Path variable, then click Edit.



- Click New and enter C:\Program Files\Amazon.



- Click OK to close all the windows and Restart the PC for the changes to take effect



□ Step 3: Prepare Configuration Files

Create a folder named `kub-config-files` on your Desktop.
Inside it, create three files:

1 cluster-config.yaml

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: us-east-1
  version: "1.32"

managedNodeGroups:
  - name: my-nodes
    instanceType: t3.small
    desiredCapacity: 2
    volumeSize: 20
    ssh:
      allow: false
```

---

2 nginx-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx-container
        image: nginx:latest
        ports:
        - containerPort: 80
        resources:
          limits:
            memory: "128Mi"
            cpu: "500m"
```

---

3 nginx-service.yaml

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

These three files are the core components used to provision the entire EKS cluster and deploy your application onto it.

Configuration File Explanations

I. `cluster-config.yaml` (Infrastructure Definition)

This file is read by the **`eksctl`** tool and is the blueprint for the underlying AWS cloud infrastructure.

- **Primary Function:** Tells AWS *how* to build the EKS cluster control plane and the worker nodes.
- **Key Settings:**
  - **`name: my-cluster`**: The name of the EKS cluster.
  - **`region: us-east-1`**: Where the cluster will be deployed geographically.
  - **`managedNodeGroups`**: Defines the worker nodes (the virtual machines where your Nginx application runs). It requests **2** instances of the **`t3.small`** type.

II. `nginx-deployment.yaml` (Application Definition)

This file is read by **`kubectl`** and defines the desired state of your application within the Kubernetes cluster.

- **Primary Function:** Ensures that **three** identical **Nginx** Pods are constantly running and manages their lifecycle (e.g., handles restarts if a Pod fails).
- **Key Settings:**
  - **`kind: Deployment`**: The type of Kubernetes object being created.
  - **`replicas: 3`**: The system must always maintain three copies of the Nginx application.
  - **`image: nginx:latest`**: Specifies the Docker image to use for the container.

III. `nginx-service.yaml` (Exposure Definition)

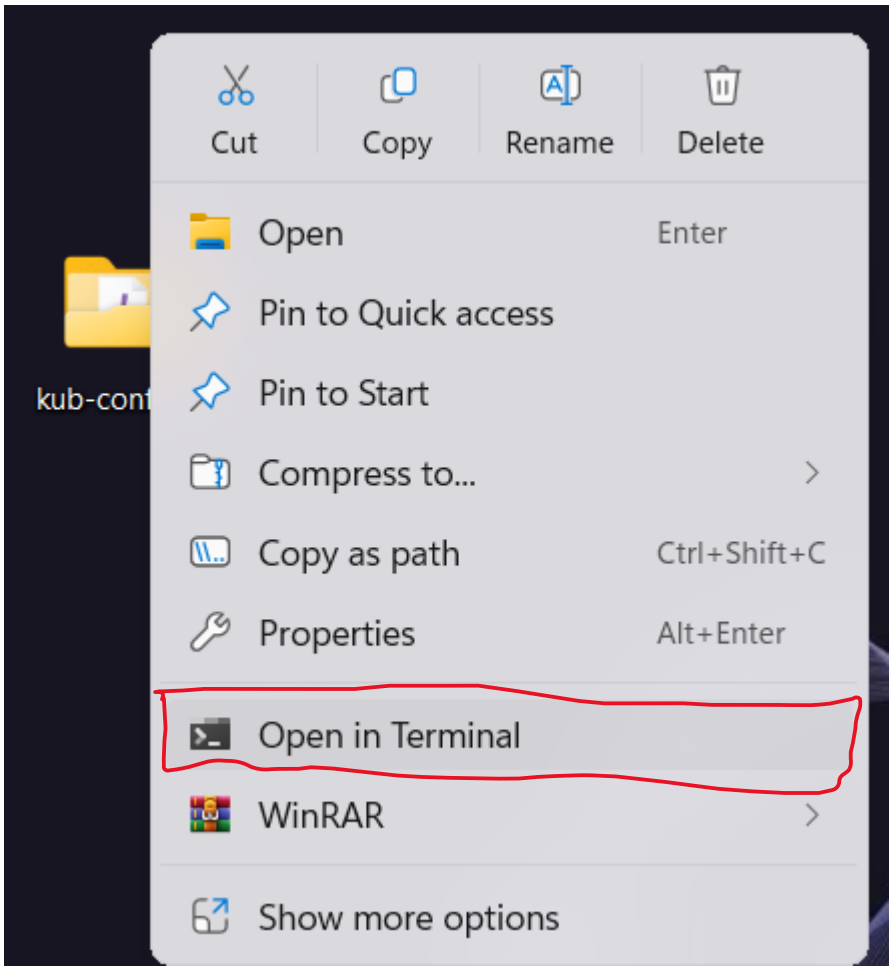
This file is read by **`kubectl`** and defines how external traffic can reach your Nginx Pods.

- **Primary Function:** Creates an external access point to your three Nginx Pods, abstracting away their internal network addresses.
- **Key Settings:**
  - `kind: Service`: The type of Kubernetes object.
  - `type: LoadBalancer`: This is the most important part; it instructs the underlying AWS EKS platform to provision an **AWS Elastic Load Balancer (ELB)** and assign it a publicly accessible DNS hostname.
  - `port: 80` / `targetPort: 80`: Exposes the service externally on port 80 and routes that traffic to the containers listening internally on port 80.

In short, `cluster-config.yaml` builds the house, `nginx-deployment.yaml` puts the workers inside, and `nginx-service.yaml` builds the front door.

💻 Step 4: Open Terminal in Configuration Folder

Right-click the folder → **Open in Terminal**.
Ensure you are in the correct directory by checking:



```
dir .
```

```
PS C:\Users\johnm\OneDrive\Desktop\kub-config-files> dir


    Directory: C:\Users\johnm\OneDrive\Desktop\kub-config-files


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         10/17/2025   1:27 PM            328 cluster-config.yaml
-a----         10/17/2025   1:55 PM            573 nginx-deployment.yaml
-a----         10/17/2025   2:16 PM            557 nginx-service.yaml
```

You should see the three YAML files.

□ Step 5: Install and Verify Tools

- Type:
  aws configure

  This shows output

```
PS C:\Users\johnm\OneDrive\Desktop\kub-config-files> aws configure
File association not found for extension .py
AWS Access Key ID [****************AX5Z]:
```

- Enter your AWS Access Key and press Enter
  This shows output

```
File association not found for extension .py
AWS Access Key ID [****************AX5Z]: AKIAQWPI[redacted]
```

- Enter Secret Access Key and press Enter
  This shows output

```
AWS Secret Access Key [****************x3jk]: +k36XkFhDH3pUvazcg50s4kOLaxoG[redacted]
```

- Select Default Region Name e.g (us-east-1) and press Enter
  This shows output

```
Default region name [us-east-1]: us-east-1
```

- For Default output format, type json and press Enter
  This shows output

```
Default output format [json]: json
```

Type aws --version and press **Enter**

aws --version

This shows output

```
PS C:\Users\johnm\OneDrive\Desktop\kub-config-files> aws --version
File association not found for extension .py
aws-cli/1.42.52 Python/3.13.6 Windows/11 botocore/1.40.52
```

Setting up kubectl Kubernetes command-line tool typing command

kubectl version --client

This shows output

```
PS C:\Users\johnm\OneDrive\Desktop\kub-config-files> kubectl version --client
Client Version: v1.32.2
Kustomize Version: v5.5.0
```

`eksctl version

You will be able to get the output below

```
PS C:\Users\johnm\OneDrive\Desktop\kub-config-files> eksctl version
0.215.0
```

🏗️ Step 6: Create Kubernetes Cluster on AWS EKS

Run:

```
eksctl create cluster -f cluster-config.yaml
```

🕒 This process takes **10–15 minutes**.
The output confirms creation of:

- Control Plane
- Worker Nodes
- Networking components

Once the installation is complete, this will show the output below.

```
2025-10-17 16:53:30 [ℹ]  creating addon: vpc-cni
2025-10-17 16:53:30 [ℹ]  successfully created addon: vpc-cni
2025-10-17 16:53:31 [ℹ]  creating addon: kube-proxy
2025-10-17 16:53:32 [ℹ]  successfully created addon: kube-proxy
2025-10-17 16:55:38 [ℹ]  building managed nodegroup stack "eksctl-my-cluster-nodegroup-my-nodes"
2025-10-17 16:55:40 [ℹ]  deploying stack "eksctl-my-cluster-nodegroup-my-nodes"
2025-10-17 16:55:40 [ℹ]  waiting for CloudFormation stack "eksctl-my-cluster-nodegroup-my-nodes"
2025-10-17 16:56:12 [ℹ]  waiting for CloudFormation stack "eksctl-my-cluster-nodegroup-my-nodes"
2025-10-17 16:57:12 [ℹ]  waiting for CloudFormation stack "eksctl-my-cluster-nodegroup-my-nodes"
2025-10-17 16:57:55 [ℹ]  waiting for CloudFormation stack "eksctl-my-cluster-nodegroup-my-nodes"
2025-10-17 16:59:00 [ℹ]  waiting for CloudFormation stack "eksctl-my-cluster-nodegroup-my-nodes"
2025-10-17 16:59:01 [ℹ]  waiting for the control plane to become ready
2025-10-17 16:59:02 [✔]  saved kubeconfig as "C:\\Users\\johnm\\.kube\\config"
2025-10-17 16:59:02 [ℹ]  no tasks
2025-10-17 16:59:02 [✔]  all EKS cluster resources for "my-cluster" have been created
2025-10-17 16:59:04 [ℹ]  nodegroup "my-nodes" has 2 node(s)
2025-10-17 16:59:04 [ℹ]  node "ip-192-168-17-97.ec2.internal" is ready
2025-10-17 16:59:04 [ℹ]  node "ip-192-168-40-218.ec2.internal" is ready
2025-10-17 16:59:04 [ℹ]  waiting for at least 2 node(s) to become ready in "my-nodes"
2025-10-17 16:59:04 [ℹ]  nodegroup "my-nodes" has 2 node(s)
2025-10-17 16:59:04 [ℹ]  node "ip-192-168-17-97.ec2.internal" is ready
2025-10-17 16:59:04 [ℹ]  node "ip-192-168-40-218.ec2.internal" is ready
2025-10-17 16:59:04 [✔]  created 1 managed nodegroup(s) in cluster "my-cluster"
2025-10-17 16:59:07 [ℹ]  kubectl command should work with "C:\\Users\\johnm\\.kube\\config", try 'kubectl get nodes'
2025-10-17 16:59:07 [✔]  EKS cluster "my-cluster" in "us-east-1" region is ready
```

Verify Cluster

kubectl get nodes

```
PS C:\Users\johnm\OneDrive\Desktop\kub-config-files> kubectl get nodes
File association not found for extension .py
NAME                                STATUS   ROLES    AGE    VERSION
ip-192-168-17-97.ec2.internal       Ready    <none>   3m4s   v1.32.9-eks-113cf36
ip-192-168-40-218.ec2.internal      Ready    <none>   3m4s   v1.32.9-eks-113cf36
```

☑ You should see nodes with `STATUS = Ready`.

📦 Step 7: Deploy Nginx Application

Apply the deployment:

kubectl apply -f nginx-deployment.yaml

```
PS C:\Users\johnm\OneDrive\Desktop\kub-config-files> kubectl apply -f nginx-deployment.yaml
File association not found for extension .py
deployment.apps/nginx-deployment created
```

Check deployment:

kubectl get deployments

```
PS C:\Users\johnm\OneDrive\Desktop\kub-config-files> kubectl get deployments
File association not found for extension .py
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   3/3     3            3           41s
```

kubectl get pods

```
PS C:\Users\johnm\OneDrive\Desktop\kub-config-files> kubectl get pods
File association not found for extension .py
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-5f486b54c7-72n7q   1/1     Running   0          88s
nginx-deployment-5f486b54c7-btth9   1/1     Running   0          88s
nginx-deployment-5f486b54c7-pbghb   1/1     Running   0          88s
```

☑ You should see:

```
nginx-deployment   3/3   Running   1m
```

🌍 Step 8: Expose the Application

A Service is necessary to provide stable networking and access to the Pods. The NodePort type exposes the service on a static port on all worker nodes.

- ❖ **Apply the service (using nginx-service.yaml):**

kubectl apply -f nginx-service.yaml

```
PS C:\Users\johnm\OneDrive\Desktop\kub-config-files> kubectl apply -f nginx-service.yaml
File association not found for extension .py
service/nginx-service created
```

- ❖ **Check service details:**

kubectl get svc

```
PS C:\Users\johnm\OneDrive\Desktop\kub-config-files> kubectl get svc
File association not found for extension .py
NAME            TYPE           CLUSTER-IP      EXTERNAL-IP                                                                PORT(S)        AGE
kubernetes      ClusterIP      10.100.0.1      <none>                                                                     443/TCP        14m
nginx-service   LoadBalancer   10.100.66.151   ab68f9736265e41b799786bfb0ddfadb-343796880.us-east-1.elb.amazonaws.com   80:32307/TCP   51s
PS C:\Users\johnm\OneDrive\Desktop\kub-config-files>
```

🌐 Step 9: Access Your Web Application

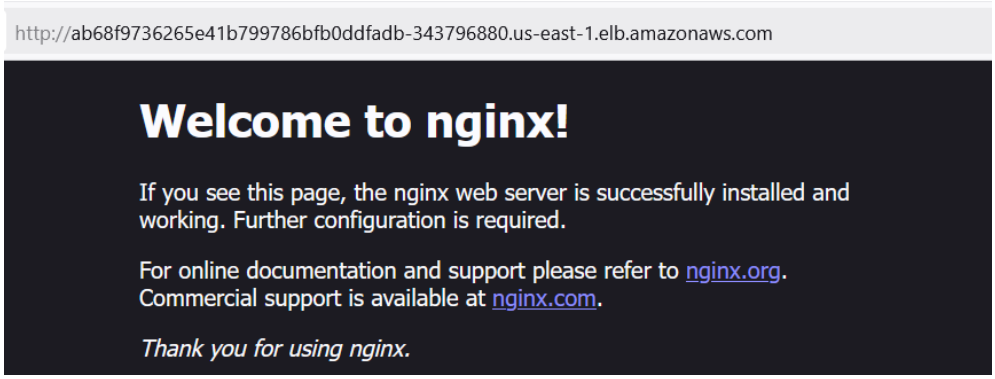
```
PS C:\Users\johnm\OneDrive\Desktop\kub-config-files> kubectl get svc
File association not found for extension .py
NAME            TYPE           CLUSTER-IP      EXTERNAL-IP                                                               PORT(S)        AGE
kubernetes      ClusterIP      10.100.0.1      <none>                                                                    443/TCP        14m
nginx-service   LoadBalancer   10.100.66.151   ab68f9736265e41b799786bfb0ddfadb-343796880.us-east-1.elb.amazonaws.com   80:32307/TCP   51s
```

In your browser, open:
`http://<EXTERNAL-IP>`

For example:

http://ab68f9736265e41b799786bfb0ddfadb-343796880.us-east-1.elb.amazonaws.com

☑ Expected Result:
You should see the **Nginx Welcome Page** —
"Welcome to nginx!"



🧹 Step 10: Clean Up AWS Resources

a) Deleting resources

After testing, delete resources to avoid charges.

Delete Kubernetes Objects:

```
kubectl delete -f nginx-service.yaml
```

```
PS C:\Users\johnm\OneDrive\Desktop\kub-config-files> kubectl delete -f nginx-service.yaml
File association not found for extension .py
service "nginx-service" deleted
```

```
kubectl delete -f nginx-deployment.yaml
```

```
PS C:\Users\johnm\OneDrive\Desktop\kub-config-files> kubectl delete -f nginx-deployment.yaml
File association not found for extension .py
deployment.apps "nginx-deployment" deleted
```

Delete EKS Cluster:

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

(Optional) elete CloudFormation Stack:

```
aws cloudformation delete-stack --stack-name eksctl-my-cluster-cluster --region us-east-1
```

```
PS C:\Users\johnm\OneDrive\Desktop\kub-config-files> aws cloudformation delete-stack --stack-name eksctl-my-cluster-cluster --region us-east-1
File association not found for extension .py
PS C:\Users\johnm\OneDrive\Desktop\kub-config-files>
```

b) Confirming Deletion of resources

To confirm everything is gone, run the following commands. The output should show that no resources are found:

kubectl get deployments

```
PS C:\Users\johnm\OneDrive\Desktop\kub-config-files> kubectl get deployments
File association not found for extension .py
No resources found in default namespace.
```

kubectl get services

```
PS C:\Users\johnm\OneDrive\Desktop\kub-config-files> kubectl get services
File association not found for extension .py
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   29m
```

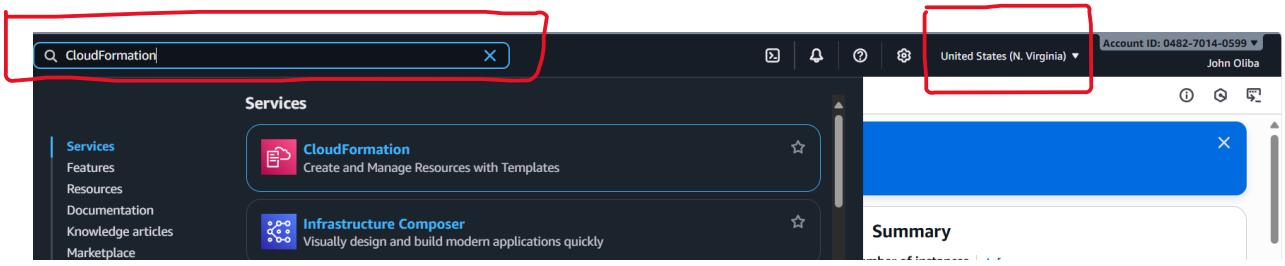
kubectl get pods -l app=nginx

```
PS C:\Users\johnm\OneDrive\Desktop\kub-config-files> kubectl get pods -l app=nginx
File association not found for extension .py
No resources found in default namespace.
```

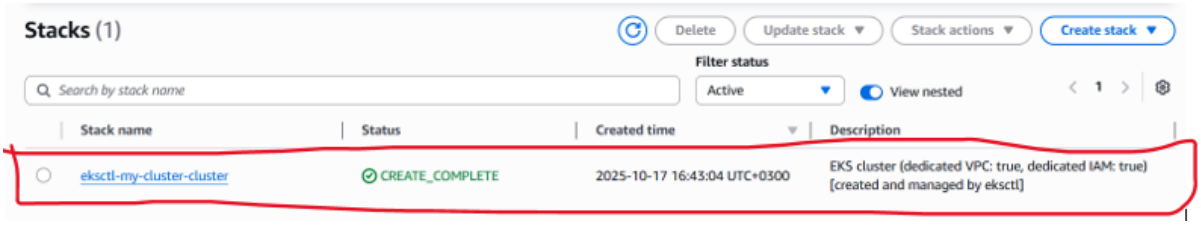
c) Manual Deletion Steps (Via AWS Console)

Step 1: Delete the Node Group Stack (The Dependency)

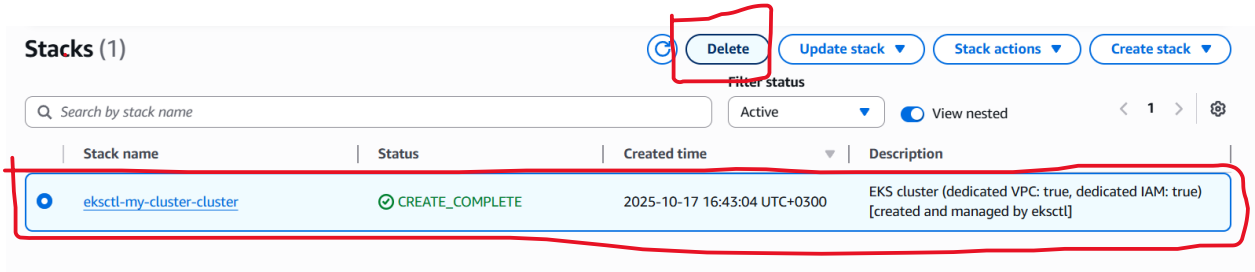
1. Log into your **AWS Management Console**.
2. Make sure you are in **US East (N. Virginia - us-east-1)** region and type **CloudFormation** service in the search bar



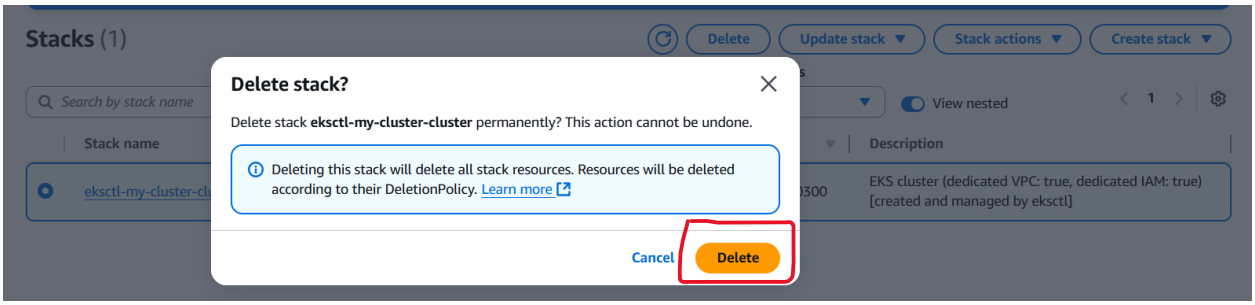
3. In the list of stacks, find the stack named: eksctl-my-cluster-nodegroup-my-nodes



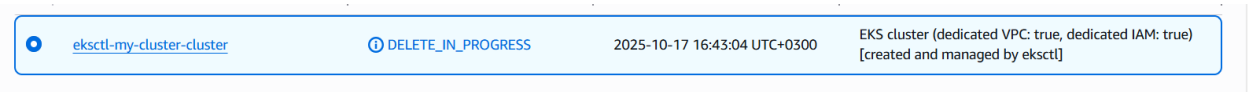
4. Click the **Delete** button at the top of the page.

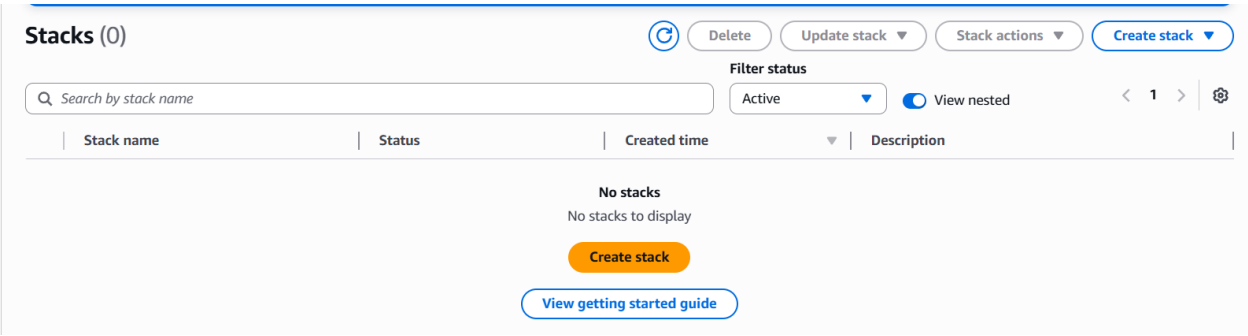


5. Confirm the deletion.



Wait for Completion (Crucial): The status of this stack will change to **DELETE_IN_PROGRESS**. You must wait until it disappears from the list, or the status shows **DELETE_COMPLETE**. This usually takes 5-10 minutes as the EC2 instances are terminated.





Confirm that the stacks nologer exist

```
PS C:\Users\johnm\OneDrive\Desktop\kub-config-files> aws cloudformation describe-stacks
File association not found for extension .py
{
    "Stacks": []
}
PS C:\Users\johnm\OneDrive\Desktop\kub-config-files>
```

- ☑ **Summary**

Step	Description
1	Configured AWS IAM user with EKS permissions
2	Installed and verified AWS CLI, kubectl, and eksctl
3	Created Kubernetes manifests (Cluster, Deployment, Service)
4	Launched an EKS cluster and verified node readiness
5	Deployed and exposed Nginx using a LoadBalancer
6	Accessed application externally
7	Cleaned up resources

💡 Best Practices

- Use **IAM roles** for secure automation instead of static keys.
- Always enable **EKS cluster logging** for observability.
- Use **CloudFront** for HTTPS traffic and caching.
- Set up **auto-scaling** for production workloads.
- Regularly clean unused clusters and LoadBalancers to reduce cost.

Author Information

Prepared by: John Michael Oliba
Passionate about;
- DevOps
- Systems Engineering
- Administration
- Graphics Design
- Accounts & Finance

Date: October 17, 2025
Version: 1.0

GitHub Link: git@github.com:johnmicky1/Nginx-Web-App-Deployment-on-Kubernetes-AWS-EKS-.git