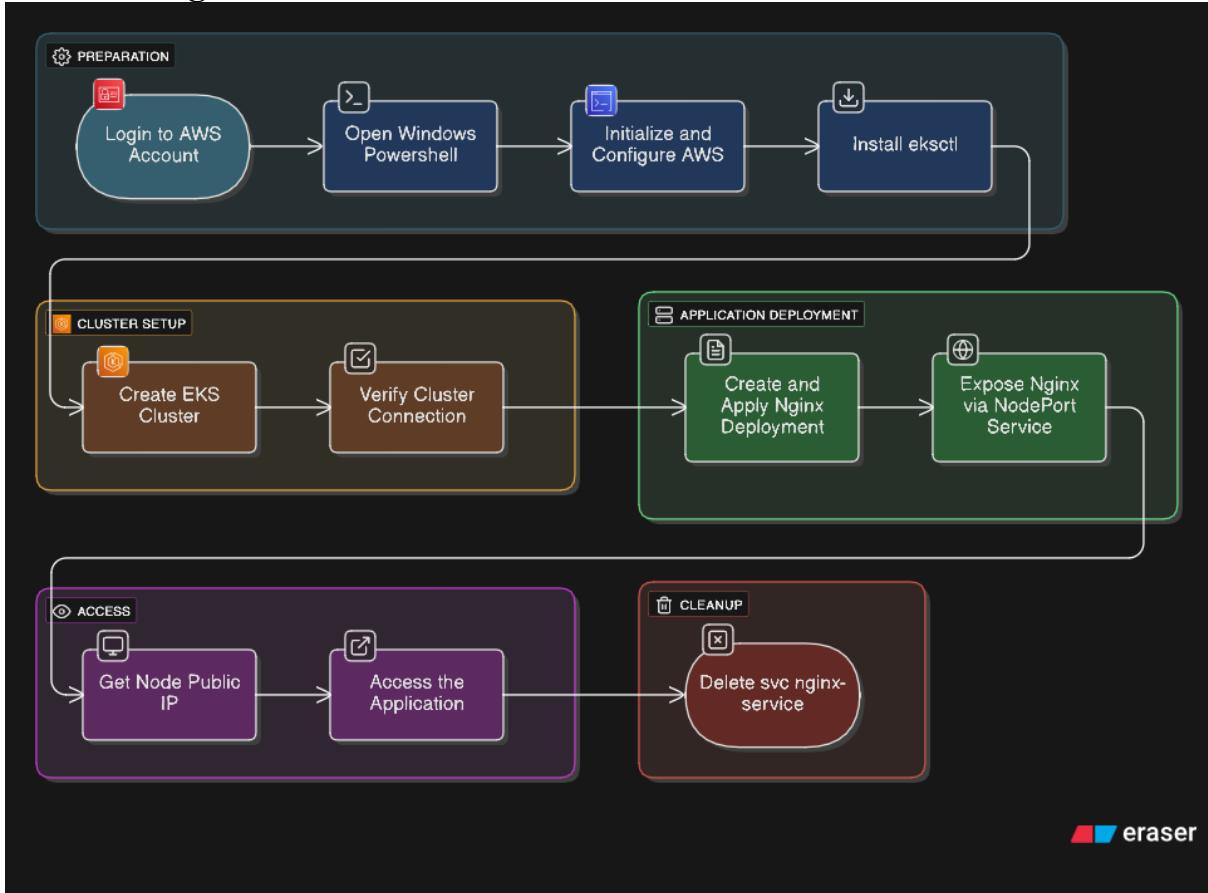


⌚ DEPLOYING A SIMPLE WEB APPLICATION ON KUBERNETES (AWS EKS)

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](https://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>

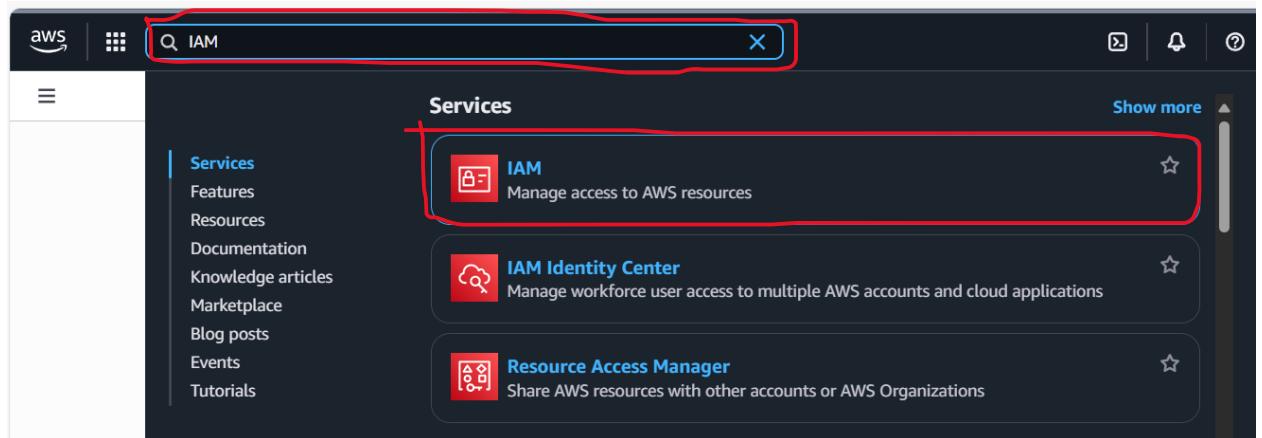
▀ Step 1: Configure AWS IAM User

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

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



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

The screenshot shows the IAM Dashboard. On the left sidebar, under 'Access management', 'Users' is selected. The main area displays 'Security recommendations' with two items: 'Root user has MFA' and 'Root user has no active access keys'. Below that is the 'IAM resources' section, which shows 0 User groups, 1 User, 6 Roles, 1 Policy, and 0 Identity providers. To the right are sections for 'AWS Account' (Account ID: 0482-7014-0599, Sign-in URL: https://048270140599.signin.aws.amazon.com/console) and 'Quick Links' (My security credentials).

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

The screenshot shows the 'Users' page. The left sidebar has 'Users' selected under 'Access management'. The main area lists one user named 'Terraform-Key' with a status of '9 hours ago'. On the far right, there are 'Create user' and 'Delete' buttons, with 'Create user' highlighted by a red box.

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

The screenshot shows the 'Specify user details' step of the IAM User creation wizard. The left sidebar shows the steps: Step 1 (selected), Step 2, Step 3, and Review and create. The main area has a 'User details' section with a 'User name' field containing 'Kubernetes-User-Admin', which is also highlighted by a red box. Below the field is a note: 'The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and * = , . @ _ - (hyphen)'. There is also an optional checkbox for 'Provide user access to the AWS Management Console'. At the bottom, a note says: 'If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user.' A 'Next' button is at the bottom right, also highlighted by a red box.

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

The screenshot shows the 'Set permissions' step in the IAM 'Create user' wizard. The 'Attach policies directly' option is selected, highlighted with a blue border. The other options ('Add user to group' and 'Copy permissions') are shown with smaller borders.

- f) Select **AdministratorAccess**

The screenshot shows the 'Permissions policies' list. The 'AdministratorAccess' policy is selected and highlighted with a red border. Other policies listed include 'AccessAnalyzerServiceRolePolicy', 'AdministratorAccess-Amplify', and 'AdministratorAccess-AWSElasticBeanstalk'. The search bar at the top left contains 'Search'.

- g) Type **EKS** in the search box and select **AmazonEKSClusterPolicy**

The screenshot shows the 'Permissions policies' list with a search bar containing 'eks'. The 'AmazonEKSClusterPolicy' policy is selected and highlighted with a red border. Other policies listed include 'AmazonEKS_CNI_Policy', 'AmazonEKSBlockStoragePolicy', 'AmazonEKSClusterPolicy', 'AmazonEKSComputePolicy', 'AmazonEKSCollectorServiceRolePolicy', and 'AmazonEKSDashboardConsoleReadOnly'. The search bar at the top left contains 'eks'.

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

The screenshot shows the 'Permissions policies' list with a search bar containing 'EKSServicePolicy'. The 'AmazonEKSServicePolicy' policy is selected and highlighted with a red border. Other policies listed include 'AmazonEKSClusterPolicy', 'AmazonEKSComputePolicy', 'AmazonEKSCollectorServiceRolePolicy', and 'AmazonEKSDashboardConsoleReadOnly'. The search bar at the top left contains 'EKSServicePolicy'.

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

Filter by Type

Q: AmazonEC2ContainerRegistryReadOnly

1 match

Policy name ▾ Type Attached entities

AmazonEC2ContainerRegistryReadOnly AWS managed 0

▶ Set permissions boundary - optional

Cancel Previous Next

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

Permissions summary

Name ▾	Type	Used as
AdministratorAccess	AWS managed - job function	Permissions policy
AmazonEC2ContainerRegistryReadOnly	AWS managed	Permissions policy
AmazonEKSClusterPolicy	AWS managed	Permissions policy
AmazonEKSServicePolicy	AWS managed	Permissions policy

Tags - optional

Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.

No tags associated with the resource.

Add new tag

You can add up to 50 more tags.

Cancel Previous Create user

- j) The User has been created

User name	Path	Group:	Last activity	MFA	Password age	Console last sign-in
<input checked="" type="checkbox"/> Kubernetes-User-Admin	/	0	-	-	-	-

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

Kubernetes-User-Admin [Info](#) [Delete](#)

Summary

ARN arn:aws:iam::048270140599:user/Kubernetes-User-Admin	Console access Disabled	Access key 1 Create access key
Created October 12, 2025, 12:36 (UTC+03:00)	Last console sign-in -	

Permissions Groups Tags **Security credentials** Last Accessed

Console sign-in

Console sign-in link
<https://048270140599.signin.aws.amazon.com/console>

Console password
Not enabled

Enable console access

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

Access keys (0)

Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. [Learn more](#)

No access keys. As a best practice, avoid using long-term credentials like access keys. Instead, use tools which provide short term credentials. [Learn more](#)

[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**

Step 1

Access key best practices & alternatives

Step 2 - optional
Set description tag

Step 3
Retrieve access keys

Access key best practices & alternatives [Info](#)

Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.

Use case

Command Line Interface (CLI)
You plan to use this access key to enable the AWS CLI to access your AWS account.

Local code
You plan to use this access key to enable application code in a local development environment to access your AWS account.

Application running on an AWS compute service
You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.

Third-party service
You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.

⚠ Alternatives recommended

- Use [AWS CloudShell](#), a browser-based CLI, to run commands. [Learn more](#)
- Use the [AWS CLI V2](#) and enable authentication through a user in IAM Identity Center. [Learn more](#)

Confirmation

I understand the above recommendation and want to proceed to create an access key.

[Cancel](#) [Next](#)

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

Step 1

Access key best practices & alternatives

Set description tag

Step 3
Retrieve access keys

Set description tag - optional [Info](#)

The description for this access key will be attached to this user as a tag and shown alongside the access key.

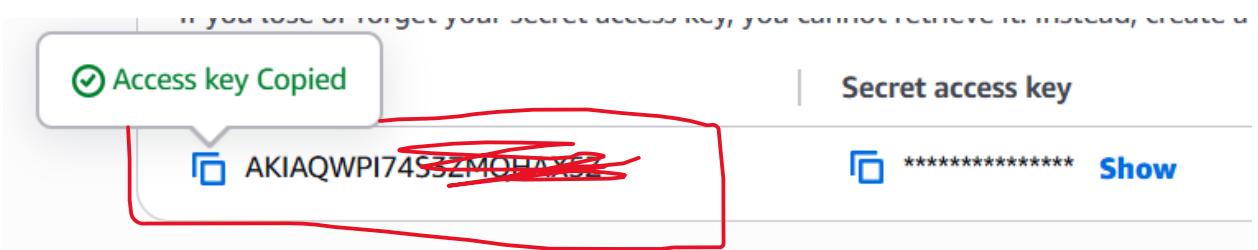
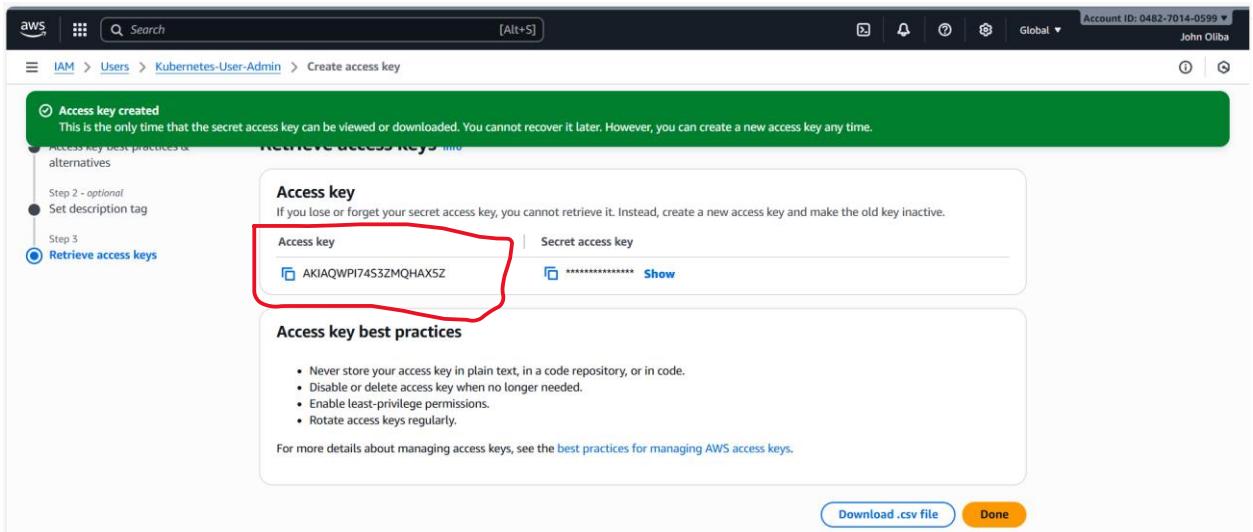
Description tag value
Describe the purpose of this access key and where it will be used. A good description will help you rotate this access key confidently later.

Kubernetes-Access

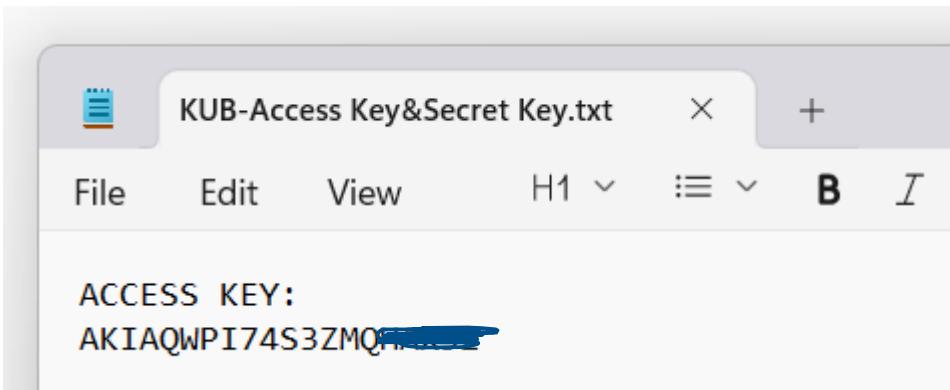
Maximum 256 characters. Allowed characters are letters, numbers, spaces representable in UTF-8, and: _ . / = + - @

[Cancel](#) [Previous](#) [Create access key](#)

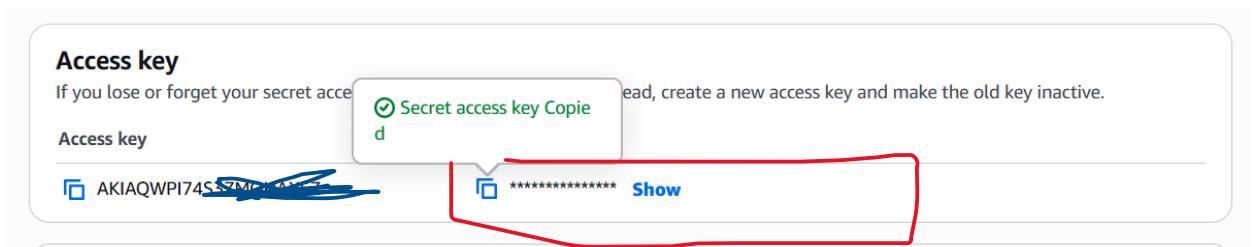
- o) Save the Access Key in a sure location on your local machine because it will be necessary for Accessing the AWS CLI using your local machine.

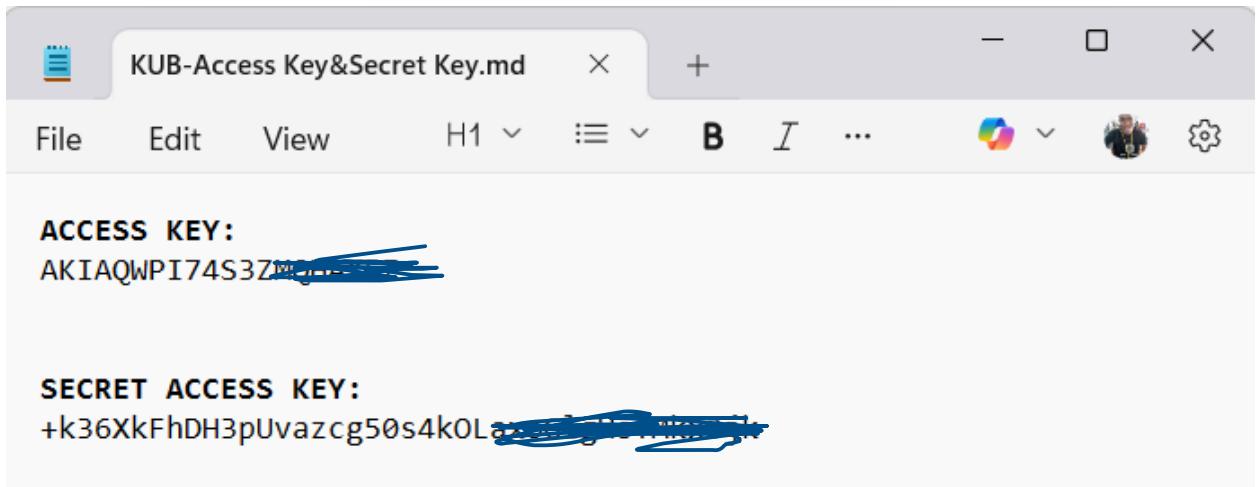


Open Note Pad, Type **ACCESS KEY** and paste the Access Key as illustrated in the Image below:



- 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.

Access key created
This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.

Access key best practices

Step 2 - optional
Set description tag
Step 3
Retrieve access keys

Access key
If you lose or forget your secret access key, you can't use it again. Instead, create a new access key and make the old key inactive.

Access key AKIAQWPI74S3ZM...

Secret access key Copied

Access key best practices

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

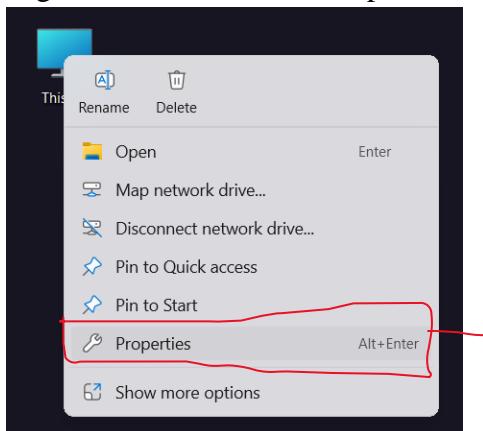
For more details about managing access keys, see the [best practices for managing AWS access keys](#).

Download .csv file Done

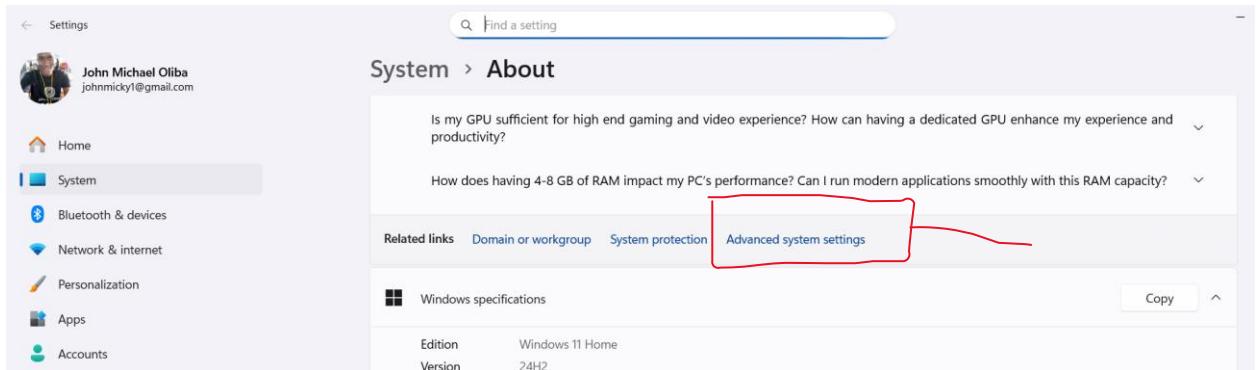
A C.S.V file will be download, make sure it is saved in a secure location

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

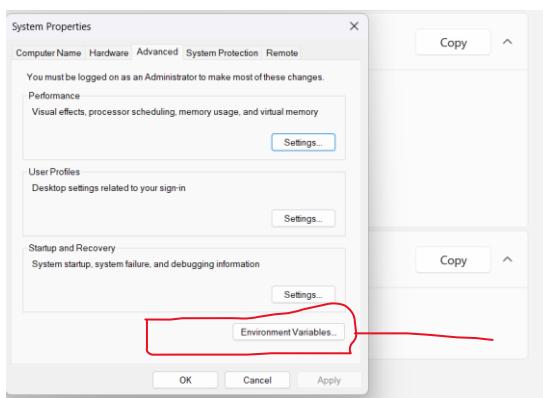
- Right-click This PC or Computer and select Properties.



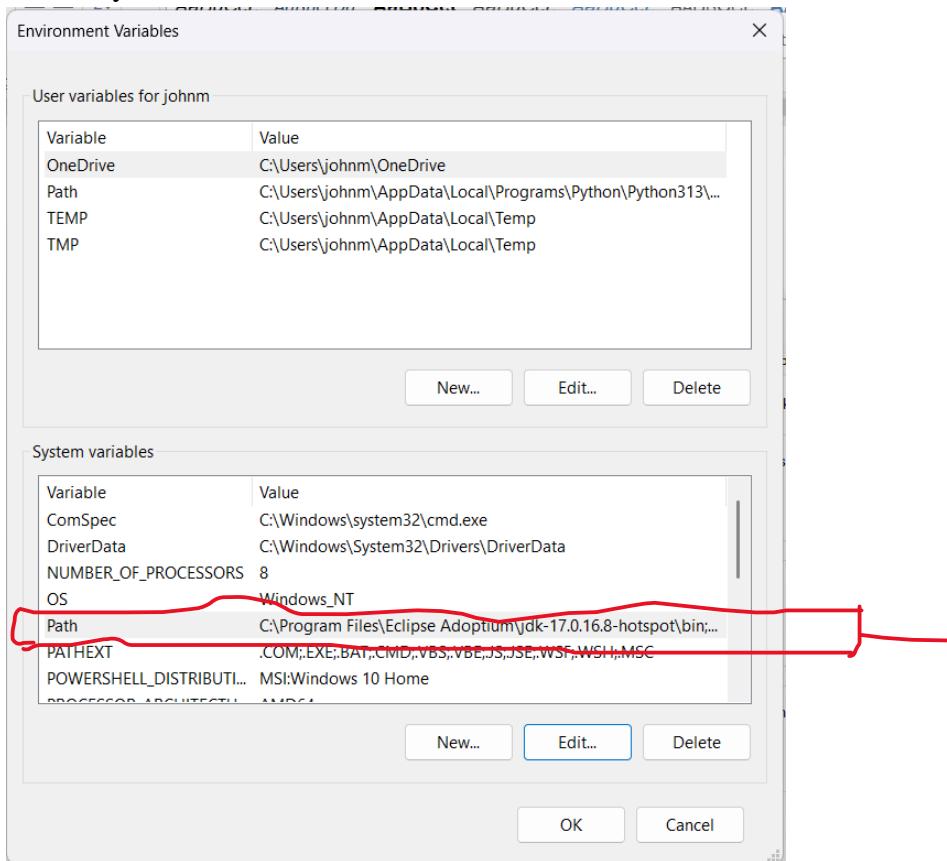
- Click on Advanced system settings on the left side.



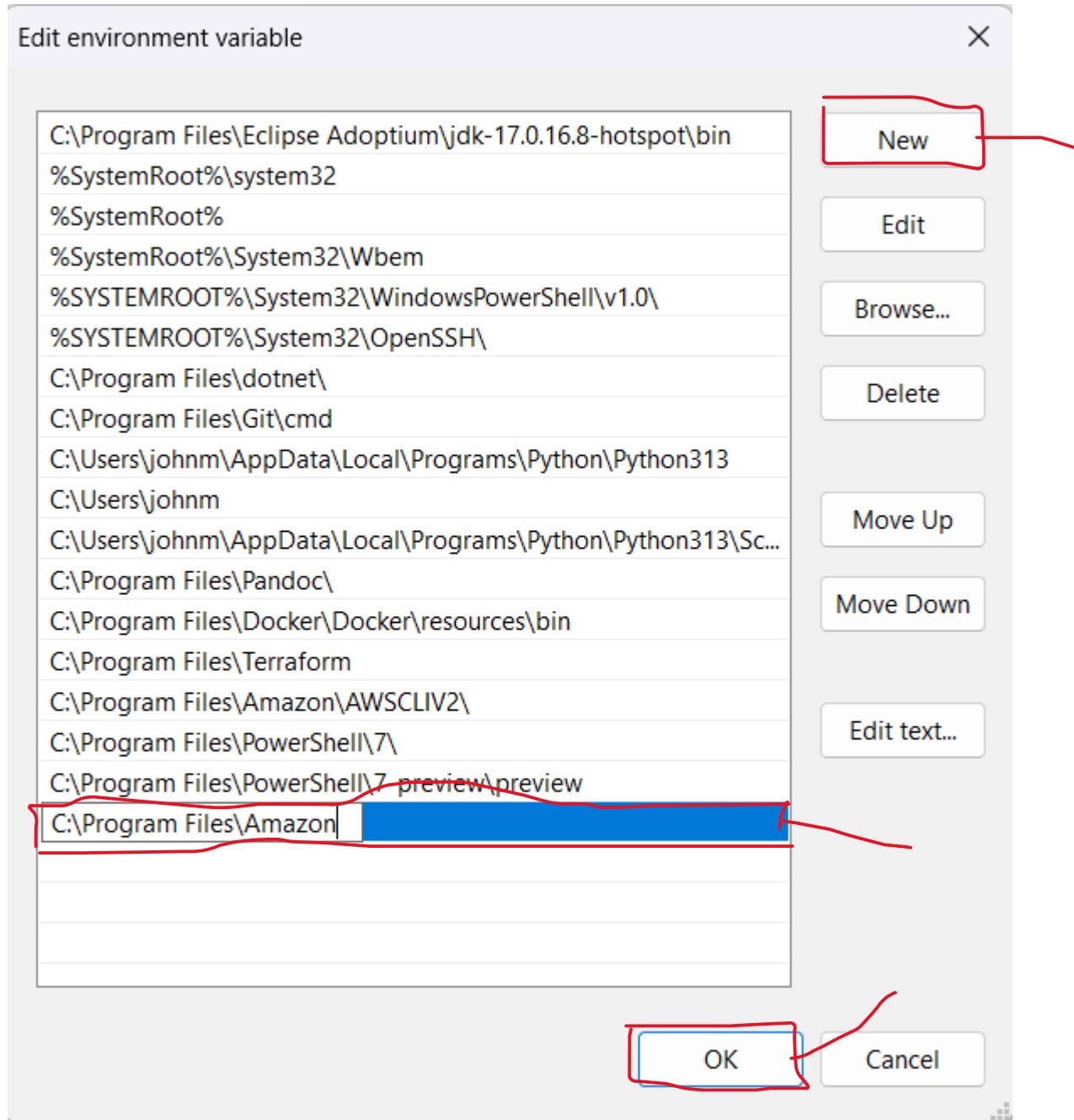
- Click on Environment Variables.



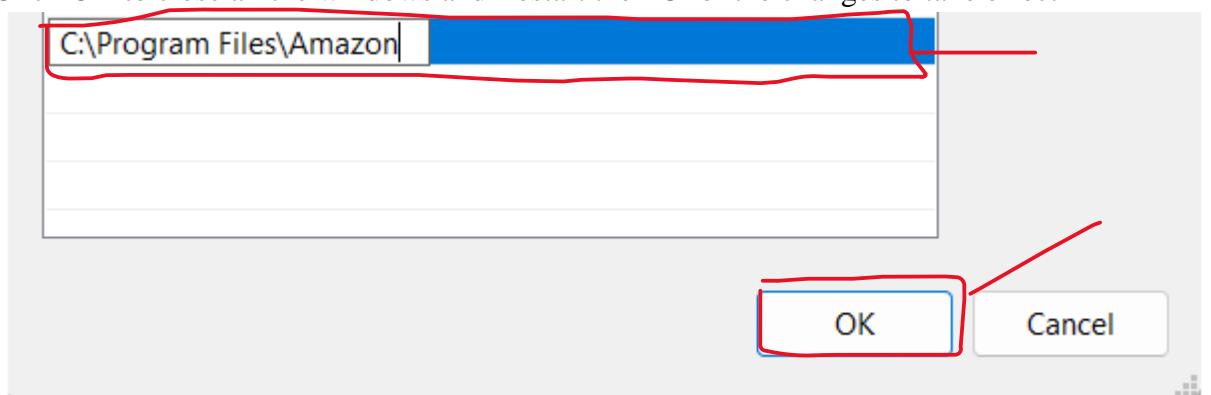
- Under System Variables, scroll down and find the Path variable, then click Edit.



- 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.

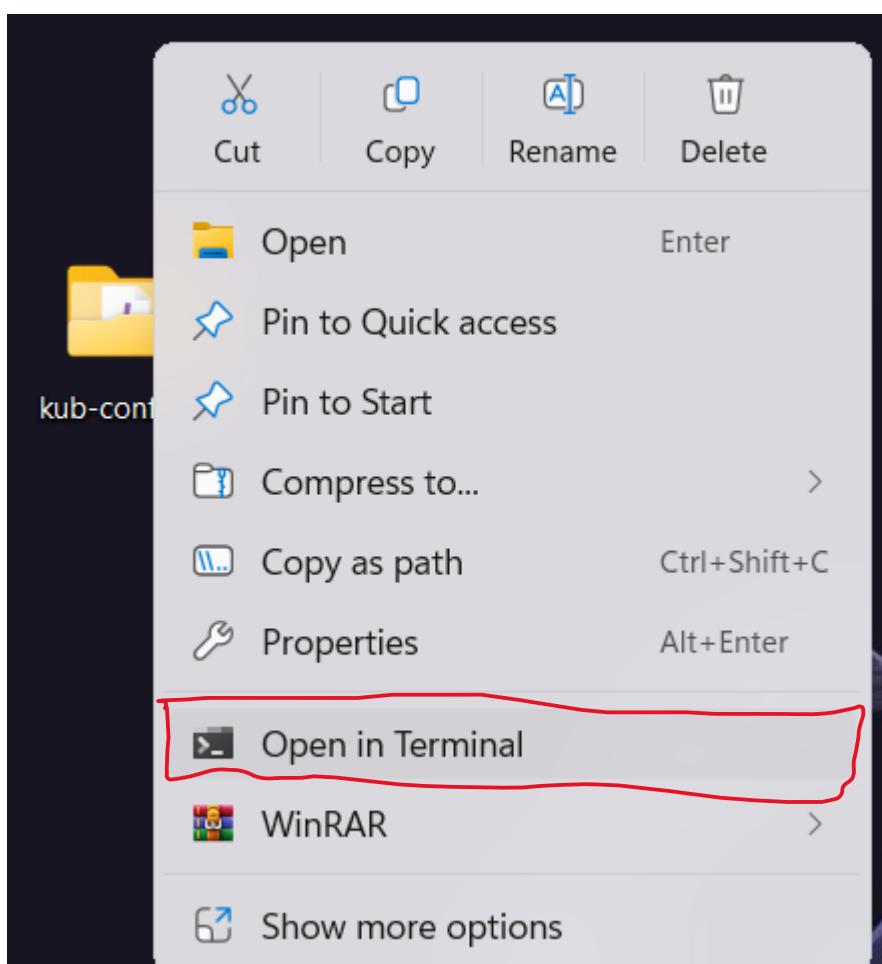
- **Primary Function:** Creates an external access point to your three Nginx Pods, abstracting away their internal network addresses.
- **Key Settings:**
 - **kind: Service**: The type of Kubernetes object.
 - **type: LoadBalancer**: This is the most important part; it instructs the underlying AWS EKS 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 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]: AKIAQWPIT7HSG3*****AX5Z
```

- Enter Secret Access Key and press Enter
This shows output

```
AWS Secret Access Key [*****x3jk]: +k36XkFhDH3pUvazcg50s4k0LaxoGlgy*****
```

- 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
```

E 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 [✓] creating addon: vpc-cni
2025-10-17 16:53:30 [✓] successfully created addon: vpc-cni
2025-10-17 16:53:31 [✓] creating addon: kube-proxy
2025-10-17 16:53:32 [✓] successfully created addon: kube-proxy
2025-10-17 16:55:38 [✓] building managed nodegroup stack "eksctl-my-cluster-nodegroup-my-nodes"
2025-10-17 16:55:40 [✓] deploying stack "eksctl-my-cluster-nodegroup-my-nodes"
2025-10-17 16:55:40 [✓] waiting for CloudFormation stack "eksctl-my-cluster-nodegroup-my-nodes"
2025-10-17 16:56:12 [✓] waiting for CloudFormation stack "eksctl-my-cluster-nodegroup-my-nodes"
2025-10-17 16:57:12 [✓] waiting for CloudFormation stack "eksctl-my-cluster-nodegroup-my-nodes"
2025-10-17 16:57:55 [✓] waiting for CloudFormation stack "eksctl-my-cluster-nodegroup-my-nodes"
2025-10-17 16:59:00 [✓] waiting for CloudFormation stack "eksctl-my-cluster-nodegroup-my-nodes"
2025-10-17 16:59:01 [✓] 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 [✓] no tasks
2025-10-17 16:59:02 [✓] all EKS cluster resources for "my-cluster" have been created
2025-10-17 16:59:04 [✓] nodegroup "my-nodes" has 2 node(s)
2025-10-17 16:59:04 [✓] node "ip-192-168-17-97.ec2.internal" is ready
2025-10-17 16:59:04 [✓] node "ip-192-168-40-218.ec2.internal" is ready
2025-10-17 16:59:04 [✓] waiting for at least 2 node(s) to become ready in "my-nodes"
2025-10-17 16:59:04 [✓] nodegroup "my-nodes" has 2 node(s)
2025-10-17 16:59:04 [✓] node "ip-192-168-17-97.ec2.internal" is ready
2025-10-17 16:59:04 [✓] 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 [✓] 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
```

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
```

File association not found for extension .py				
deployment.apps/nginx-deployment created				

Check deployment:

```
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
```

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
```

File association not found for extension .py				
service/nginx-service created				

- ❖ Check service details:

```
kubectl get svc
```

File association not found for extension .py					
NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
nginx-service	LoadBalancer	10.100.66.151	ab68f9736265e41b799786bf0ddfad-343796880.us-east-1.elb.amazonaws.com	443/TCP 80:32307/TCP	14m 51s

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
kubernetes     ClusterIP  10.100.0.1    <none>
nginx-service  LoadBalancer 10.100.66.151 ab68f9736265e41b799786bfb0ddfadb-343796880.us-east-1.elb.amazonaws.com
                                         PORT(S)        AGE
                                         443/TCP      14m
                                         80:32307/TCP 51s
```

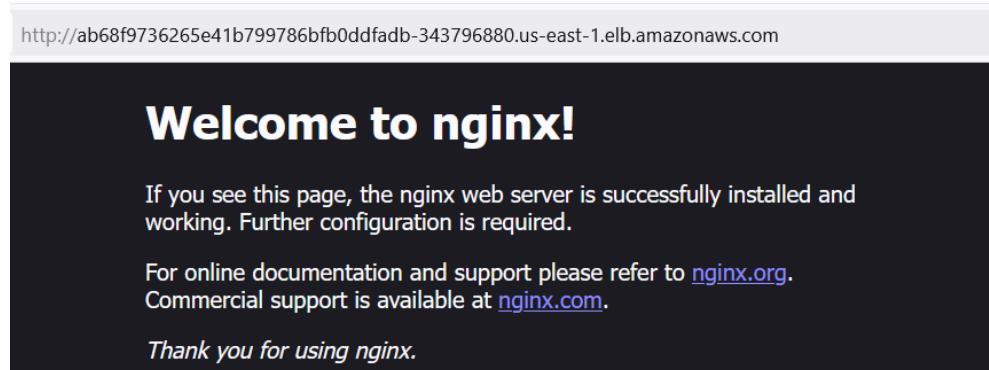
In your browser, open:
<http://<EXTERNAL-IP>>

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

Delete EKS Cluster:

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

(Optional) Delete 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
```

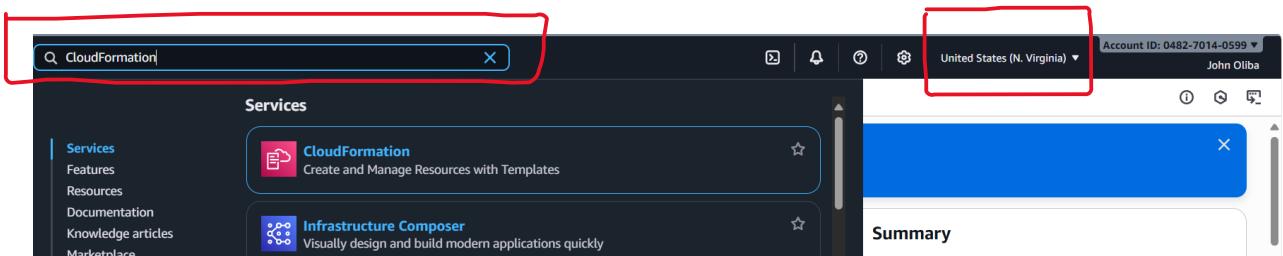
```
kubectl get pods -l app=nginx
```

```
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

Stacks (1)			
Stack name	Status	Created time	Description
eksctl-my-cluster-nodegroup-my-nodes	CREATE_COMPLETE	2025-10-17 16:43:04 UTC+0300	EKS cluster (dedicated VPC: true, dedicated IAM: true) [created and managed by eksctl]

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

Stacks (1)

Stack name	Status	Created time	Description
eksctl-my-cluster-cluster	CREATE_COMPLETE	2025-10-17 16:43:04 UTC+0300	EKS cluster (dedicated VPC: true, dedicated IAM: true) [created and managed by eksctl]

- Confirm the deletion.

Delete stack?

Delete stack [eksctl-my-cluster-cluster](#) permanently? This action cannot be undone.

Deleting this stack will delete all stack resources. Resources will be deleted according to their [DeletionPolicy](#).

[Cancel](#) [Delete](#)

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.

Stacks (1)

Stack name	Status	Created time	Description
eksctl-my-cluster-cluster	DELETE_IN_PROGRESS	2025-10-17 16:43:04 UTC+0300	EKS cluster (dedicated VPC: true, dedicated IAM: true) [created and managed by eksctl]

Stacks (0)

Stack name	Status	Created time	Description
------------	--------	--------------	-------------

No stacks

No stacks to display

[Create stack](#)

[View getting started guide](#)

Confirm that the stacks no longer 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.
-

aticon 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](https://github.com/johnmicky1/Nginx-Web-App-Deployment-on-Kubernetes-AWS-EKS-.git)