

Name: Yash Deepak Wagh

Team: Cloud

Date: 18/07/2024

Task: Deploy an Amazon EKS Cluster using Terraform (Major Project).

Solution:

Step 1: Install Terraform on Windows

1) Download Terraform:

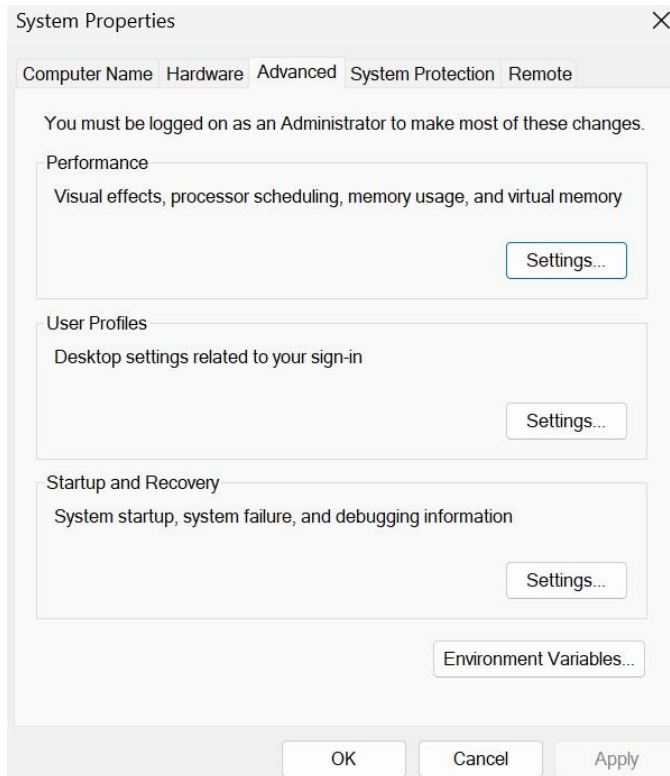
□ Go to the [Terraform downloads page](#). □
Download the appropriate version for Windows.

2) Install Terraform:

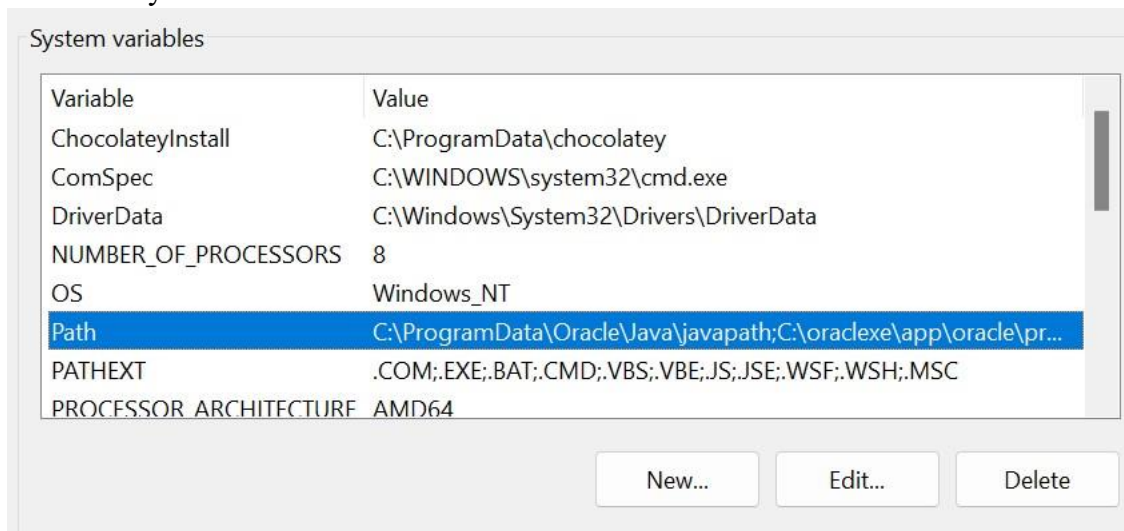
- Extract the downloaded zip file.
- Move the `terraform.exe` file to a directory included in your system's PATH (e.g., `C:\Program Files\Terraform`).
- Copy (e.g., `C:\Program Files\Terraform`) Path to your system “Environment variable”.

Set terraform path in Environment variable using below images.

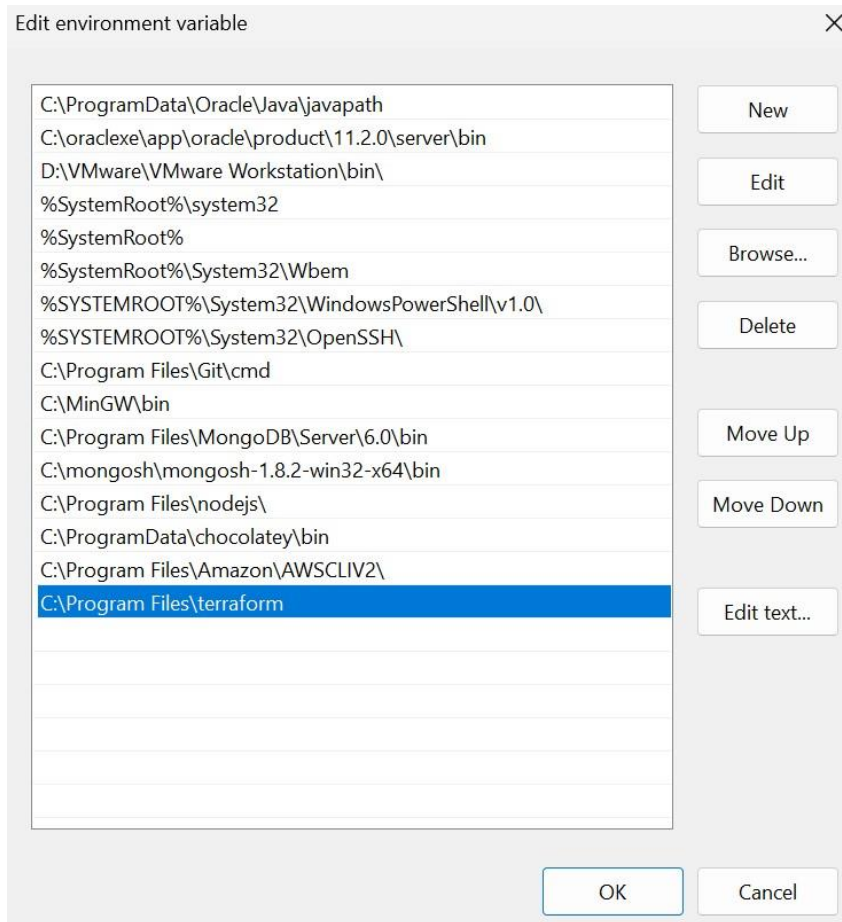
- Search Environment variable in windows search bar or else in settings.
- Click on Environment variable.



- In system variable Click on “Path” and then click “Edit”.



- Add new path



Click “ok” and environment variable path is set for terraform.

3) To check Terraform is install in the system:

☐ Search cmd and Run as administrator ☐

☐ cd C:\Program Files\Terraform

☐ terraform --version

```
C:\Windows\System32>cd C:\Program Files\Terraform  
  
C:\Program Files\terraform>terraform --version  
Terraform v1.9.0  
on windows_amd64
```

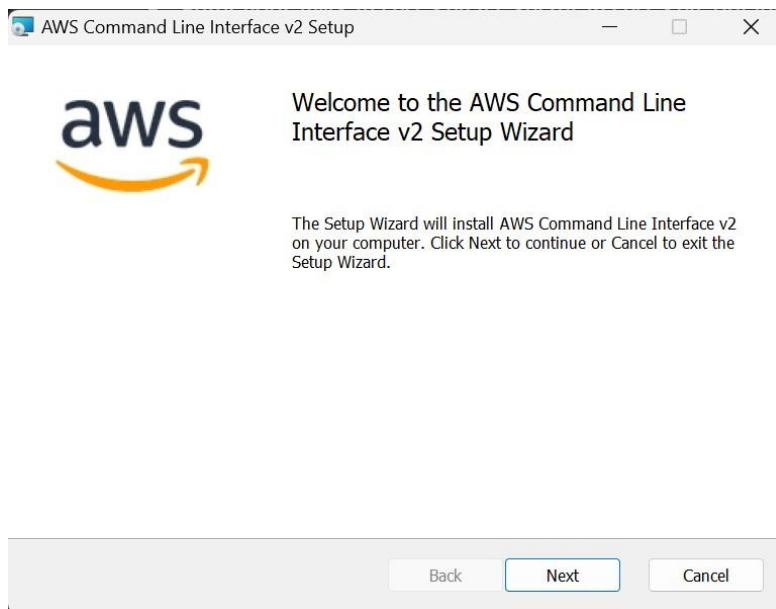
Step 2: Configure AWS Credentials for Terraform.

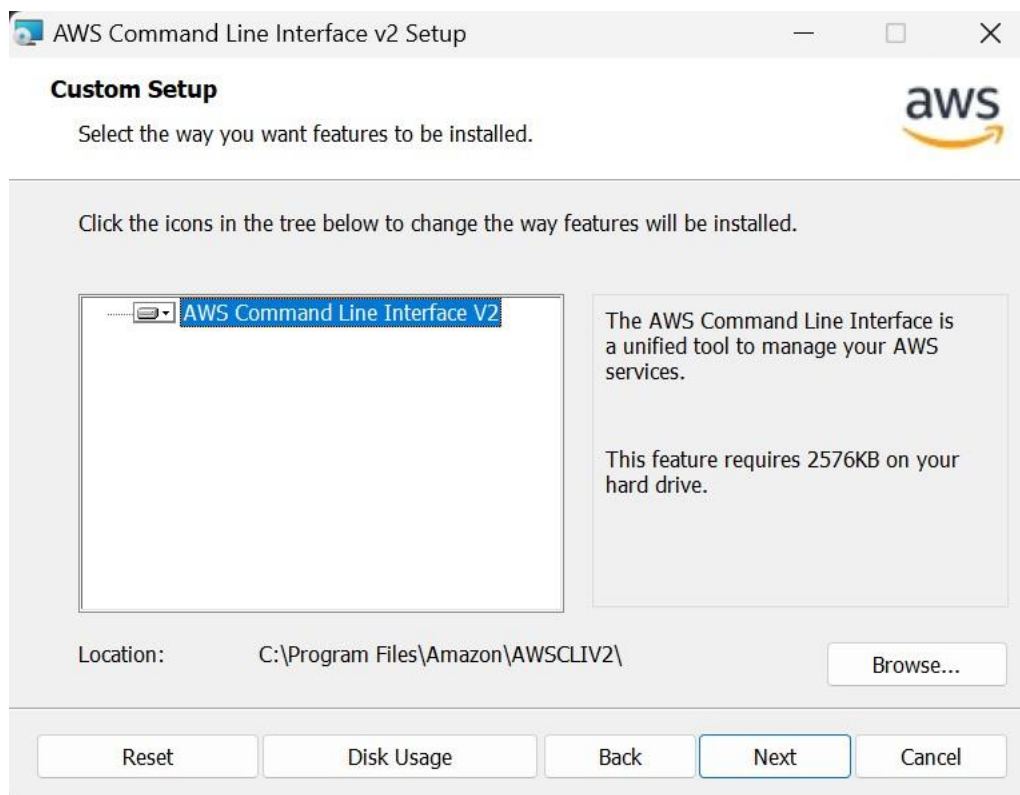
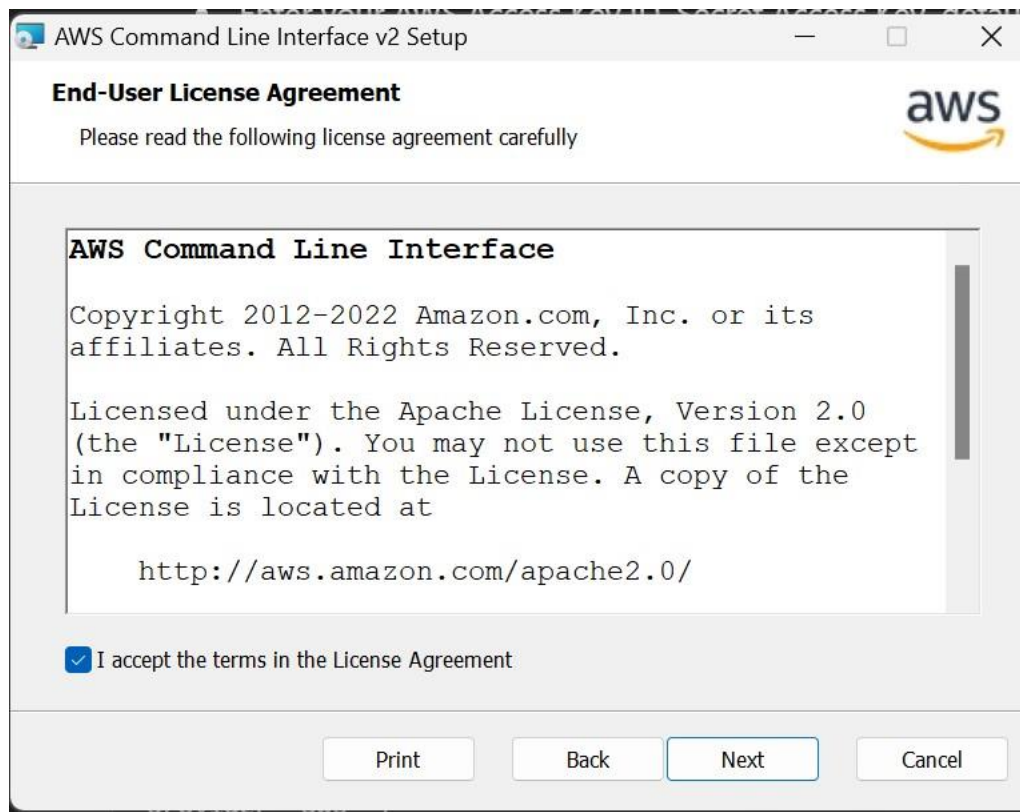
1) **Install AWS CLI** (if not already installed): □ Download the AWS CLI from the [AWS CLI installation page](https://awscli.amazonaws.com/AWSCLIV2.msi).

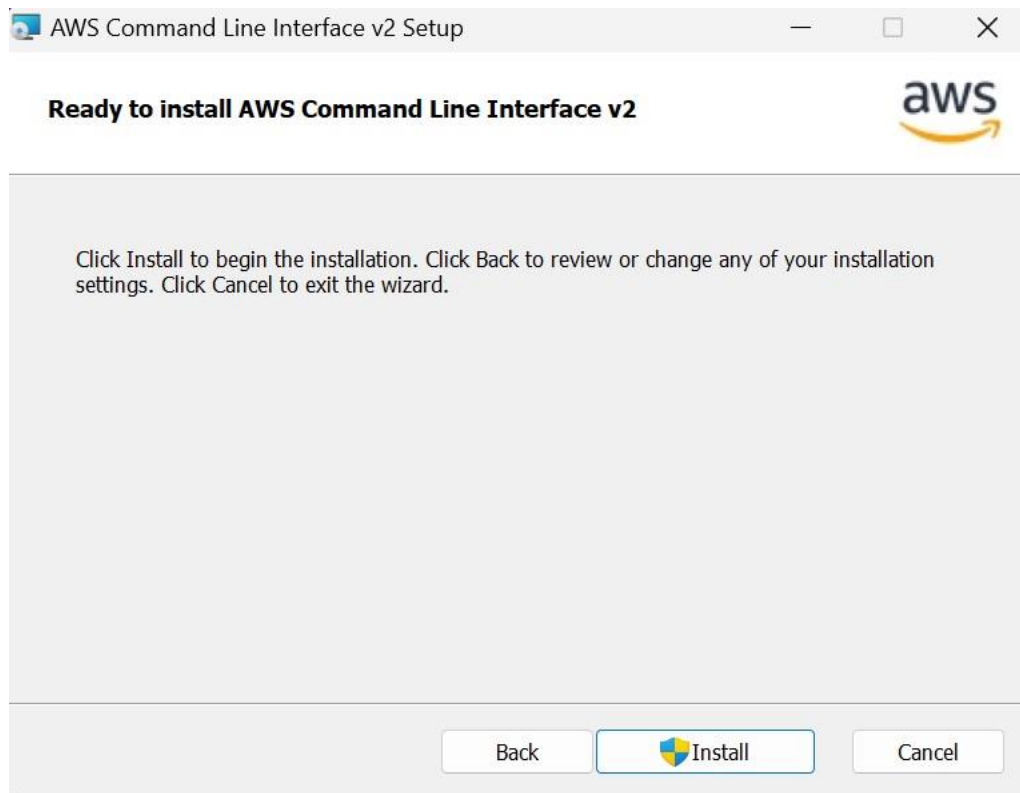
□ Follow the installation instructions.

```
C:\Program Files>cd..
```

```
C:\>msiexec.exe /i https://awscli.amazonaws.com/AWSCLIV2.msi
```







2) Create IAM user to get AWS Access Key ID, Secret Access Key.

- ☐ Go to IAM in AWS console.
- ☐ Create user by following below images.

- We have attach administrator access for the demonstration purpose in the production scenario it is not the best practice.

- **In the terraform script we have added the IAM role and policy so the process should be done in the secure and independent way.**

- Select **“Attach policies directly”**
- ☐ Select **“Administrator Access”**

Permissions options

☐ Add user to group
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

☐ Copy permissions
Copy all group memberships, attached managed policies, and inline policies from an existing user.

☒ Attach policies directly
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

Permissions policies (1/1221)

Choose one or more policies to attach to your new user.

Filter by Type

All types ▼

272 matches

< 1 2 3 4 5 6 7 ... 14 >

	Policy name	Type	Attached entities
<input checked="" type="checkbox"/>	AdministratorAccess	AWS managed - job function	0

Review and create the user.

- Click on Created user
- At top in Summary click **“create access key”** and follow below step.

[IAM](#) > [Users](#) > [terraform-user](#) > Create access key

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.

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.

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

[Cancel](#)

[Previous](#)

[Create access key](#)

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](#)

Download .csv file for further steps.

Create password for IAM user

- **Click on created user.**
- **Go to security credentials and set user password.**

Enable console access

×

Enable console access for terraform-user.

Console password

☐ Autogenerated password

☒ Custom password

.....

- Must be at least 8 characters long
- Must include at least three of the following mix of character types: uppercase letters (A-Z), lowercase letters (a-z), numbers (0-9), and symbols ! @ # \$ % ^ & * () _ + - (hyphen) = [] { } | ' "

☐ Show password

☐ User must create new password at next sign-in
Users automatically get the [IAMUserChangePassword](#) policy to allow them to change their own password.

Cancel

Enable console access

Console password

×

✓

You have successfully enabled the user's new password.

This is the only time you can view this password. After you close this window, if the password is lost, you must create a new one.

Console sign-in URL

<https://637423437444.signin.aws.amazon.com/console>

User name

tanmay

Console password

***** [Show](#)

Download .csv file

Close

Download the .csv file to take user access.

3) Configure AWS CLI:

- Open Command Prompt (cmd) and run “**aws configure**”.
- Enter your AWS Access Key ID, Secret Access Key, default region name (e.g., us-east2), and default output format (e.g., json).

```
C:\Windows\System32>aws configure
AWS Access Key ID [*****QBMP]: 
AWS Secret Access Key [*****uCUf]: 
Default region name [us-east-2]: 
Default output format [json]:
```

For default region name (choose your region)

For default output format Press “enter”.

Step 3: Create a Terraform Configuration for the EKS-Cluster.

1) Create a Directory for Terraform Files:

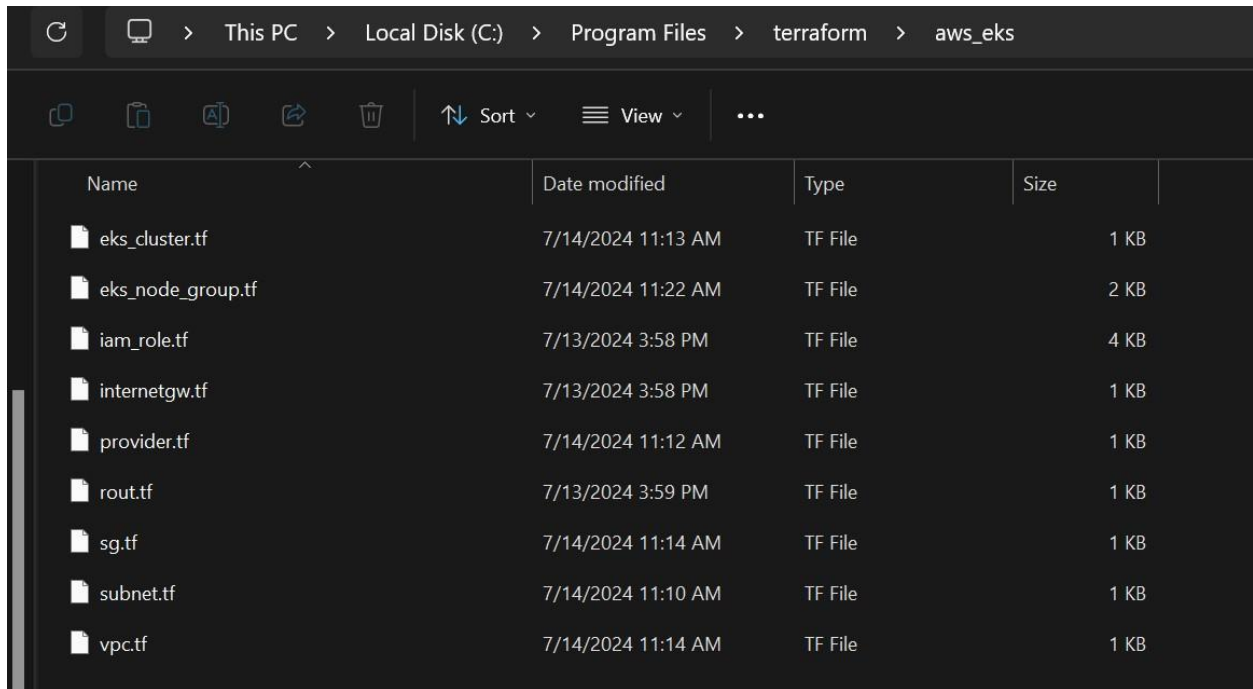
- Create a new directory for your Terraform project, e.g., C:\terraform\aws_eks.

2) Create main.tf File:

- Inside the project directory, create a nine files and add the following Terraform configuration:

1. eks_cluster.tf
2. eks_node_group.tf
3. iam_role.tf
4. internetgw.tf
5. provider.tf
6. rout.tf
7. sg.tf
8. subnet.tf
9. vpc.t

- We have provision the eks cluster in ohio (us-east-2) region and done the configuration according to that.
- So you can choose your region and do the necessary changes in the below configuration files.



Name	Date modified	Type	Size
eks_cluster.tf	7/14/2024 11:13 AM	TF File	1 KB
eks_node_group.tf	7/14/2024 11:22 AM	TF File	2 KB
iam_role.tf	7/13/2024 3:58 PM	TF File	4 KB
internetgw.tf	7/13/2024 3:58 PM	TF File	1 KB
provider.tf	7/14/2024 11:12 AM	TF File	1 KB
rout.tf	7/13/2024 3:59 PM	TF File	1 KB
sg.tf	7/14/2024 11:14 AM	TF File	1 KB
subnet.tf	7/14/2024 11:10 AM	TF File	1 KB
vpc.tf	7/14/2024 11:14 AM	TF File	1 KB

eks_cluster file

```
resource "aws_eks_cluster" "eks" {  
  name      = "eks-cluster"  role_arn =  
  aws_iam_role.master.arn  
  
  vpc_config {  subnet_ids = [aws_subnet.public-1.id,  
aws_subnet.public-2.id]  
  }  
  
  depends_on = [  
aws_iam_role_policy_attachment.AmazonEKSClusterPolicy,  
aws_iam_role_policy_attachment.AmazonEKSServicePolicy,  
aws_iam_role_policy_attachment.AmazonEKSVPCResourceController,  
aws_iam_role_policy_attachment.AmazonEKSVPCResourceController,  
    #aws_subnet.pub_sub1,  
#aws_subnet.pub_sub2,  
  ]  
}
```

eks_node_group.tf file(part 1)

```

resource "aws_instance" "kubectl-server" {

    key_name          = "mykey" //replace with your created key
    ami              = "ami-0862be96e41dcbf74" //replace with your instance ami id
    instance_type    = "t2.micro"
    associate_public_ip_address = true
    subnet_id        =
aws_subnet.public-1.id

    vpc_security_group_ids = [aws_security_group.allow_tls.id]

    tags = {
        Name = "kubectl"
    }

}

resource "aws_eks_node_group" "node-grp" {

    cluster_name = aws_eks_cluster.eks.name

    node_group_name = "pc-node-group"

    node_role_arn = aws_iam_role.worker.arn

    subnet_ids = [aws_subnet.public-1.id, aws_subnet.public-2.id]

    capacity_type = "ON_DEMAND"
    disk_size
= "20"
    instance_types = ["t2.small"]

    remote_access {

        ec2_ssh_key = "mykey" //Create your own key in aws key management and enter the name
    }
}

```

```
source_security_group_ids = [aws_security_group.allow_tls.id]
```

```
}
```

```
labels = tomap({ env = "dev" })

scaling_config {
  desired_size = 2
  max_size    = 3  min_size
= 1
}

update_config {
  max_unavailable = 1
}

depends_on = [
  aws_iam_role_policy_attachment.AmazonEKSWorkerNodePolicy,
  aws_iam_role_policy_attachment.AmazonEKS_CNI_Policy,
  aws_iam_role_policy_attachment.AmazonEC2ContainerRegistryReadOnly,
  #aws_subnet.pub_sub1,
  #aws_subnet.pub_sub2,
]
}
```

iam_role.tf file (part1)

```

resource "aws_iam_role" "master" {
  name = "ed-eks-master"

  assume_role_policy = <<POLICY
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
POLICY
}

resource "aws_iam_role_policy_attachment" "AmazonEKSClusterPolicy"
{ policy_arn = "arn:aws:iam::aws:policy/AmazonEKSClusterPolicy" role
= aws_iam_role.master.name
}

resource "aws_iam_role_policy_attachment" "AmazonEKSServicePolicy"
{ policy_arn = "arn:aws:iam::aws:policy/AmazonEKSServicePolicy" role
= aws_iam_role.master.name
}

```

iam_role.tf (part2)


```

resource "aws_iam_role_policy_attachment" "AmazonEKSVPCResourceController"
{
  policy_arn = "arn:aws:iam::aws:policy/AmazonEKSVPCResourceController"
  role      = aws_iam_role.master.name
}

resource "aws_iam_role" "worker" {
  name = "ed-eks-worker"

  assume_role_policy = <<POLICY
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
POLICY
}

resource "aws_iam_policy" "autoscaler" {
  name      = "ed-eks-autoscaler-policy"
  policy = <<EOF

```

iam_role.tf (part3)

```

{
  "Version": "2012-10-17",
  "Statement": [

```

```

{
  "Action": [
    "autoscaling:DescribeAutoScalingGroups",
    "autoscaling:DescribeAutoScalingInstances",
    "autoscaling:DescribeTags",
    "autoscaling:DescribeLaunchConfigurations",
    "autoscaling:SetDesiredCapacity",
    "autoscaling:TerminateInstanceInAutoScalingGroup",
    "ec2:DescribeLaunchTemplateVersions"
  ],
  "Effect": "Allow",
  "Resource": "*"
}
]
}
EOF

}

resource "aws_iam_role_policy_attachment" "AmazonEKSWorkerNodePolicy" {
  policy_arn = "arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy"
  role       = aws_iam_role.worker.name
}

```

iam_role.tf (part4)

```

resource "aws_iam_role_policy_attachment" "AmazonEKS_CNI_Policy"
{
  policy_arn = "arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy"
  role      = aws_iam_role.worker.name
}

resource "aws_iam_role_policy_attachment" "AmazonSSMManagedInstanceCore"
{
  policy_arn = "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore"
  role      = aws_iam_role.worker.name
}

resource "aws_iam_role_policy_attachment" "AmazonEC2ContainerRegistryReadOnly"
{
  policy_arn = "arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly"
  role      = aws_iam_role.worker.name
}

resource "aws_iam_role_policy_attachment" "x-ray" {
  policy_arn = "arn:aws:iam::aws:policy/AWSXRayDaemonWriteAccess"
  role      = aws_iam_role.worker.name
}

resource "aws_iam_role_policy_attachment" "s3" {
  policy_arn = "arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess"
  role      = aws_iam_role.worker.name
}

resource "aws_iam_role_policy_attachment" "autoscaler"
{
  policy_arn = aws_iam_policy.autoscaler.arn
  role      = aws_iam_role.worker.name
}

resource "aws_iam_instance_profile" "worker"
{
  depends_on = [aws_iam_role.worker]
  name      = "ed-eks-worker-new-profile"
  role      = aws_iam_role.worker.name
}

```

```
resource "aws_internet_gateway" "gw" { vpc_id = aws_vpc.main.id

tags = {
  Name = "main"
}
}
```

provider.tf file

```
terraform {
  required_providers { aws =
    { source = "hashicorp/aws"
  }
}
```

```
provider "aws" {
  region = "us-east-2" //replace with your region
}
```

rout.tf

```
resource "aws_route_table" "rtb" {
  vpc_id = aws_vpc.main.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.gw.id
  }

  tags = {
    Name = "MyRoute"
  }
}

resource "aws_route_table_association" "a-1" {
  subnet_id   = aws_subnet.public-1.id
  route_table_id = aws_route_table.rtb.id
}

resource "aws_route_table_association" "a-2" {
  subnet_id   = aws_subnet.public-2.id
  route_table_id = aws_route_table.rtb.id
}
```

```
resource "aws_security_group" "allow_tls"
{
  name      = "allow_tls"
  description =
    "Allow TLS inbound traffic"
  vpc_id    =
    aws_vpc.main.id

  ingress {
    description = "TLS
from VPC"
    from_port = 22
    to_port   = 22
    protocol  =
      "tcp"
    cidr_blocks =
      ["0.0.0.0/0"]
  }

  egress {
    from_port = 0
    to_port   = 0
    protocol  =
      "-1"
    cidr_blocks =
      ["0.0.0.0/0"]
  }

  tags = {
    Name = "allow_tls"
  }
}
```

```
resource "aws_subnet" "public-1" {  
  vpc_id      = aws_vpc.main.id cidr_block  
  = "10.0.1.0/24"  
  availability_zone    = "us-east-2a" //replace with your zone  
  map_public_ip_on_launch = true  
  
  tags = {  
    Name = "public-sub-1"  
  }  
}  
  
resource "aws_subnet" "public-2" {  
  vpc_id      = aws_vpc.main.id cidr_block  
  = "10.0.2.0/24"  
  availability_zone    = "us-east-2b" //replace with your zone  
  map_public_ip_on_launch = true  
  
  tags = {  
    Name = "public-sub-2"  
  }  
}
```

vpc.tf

```

resource "aws_vpc" "main" {
  cidr_block = "10.0.0.0/16"

  tags = {
    Name = "VPC-main" //replace with your
name
  }
}

```

3) Initialize Terraform:

- Open Command Prompt and navigate to your project directory.
- Run “**terraform init**” to initialize the project.

```

C:\Program Files\terraform\aws-eks>terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.57.0...
- Installed hashicorp/aws v5.57.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

```

4) Apply Terraform Configuration:

- Run “**terraform apply**”.
- Review the plan output and confirm the apply by typing “**yes**”.


```

C:\Program Files\terraform\aws_eks>terraform apply
aws_iam_policy.autoscaler: Refreshing state... [id=arn:aws:iam::637423437444:policy/ed-eks-autoscaler-policy]
aws_iam_role.worker: Refreshing state... [id=ed-eks-worker]
aws_iam_role.master: Refreshing state... [id=ed-eks-master]

Note: Objects have changed outside of Terraform

Terraform detected the following changes made outside of Terraform since the last "terraform apply" which may have affected this plan:

# aws_iam_policy.autoscaler has changed
~ resource "aws_iam_policy" "autoscaler" {
+   arn      = "arn:aws:iam::637423437444:policy/ed-eks-autoscaler-policy"
    id       = "arn:aws:iam::637423437444:policy/ed-eks-autoscaler-policy"
    name     = "ed-eks-autoscaler-policy"
    # (2 unchanged attributes hidden)
}

# aws_iam_role.master has changed
~ resource "aws_iam_role" "master" {
+   arn      = "arn:aws:iam::637423437444:role/ed-eks-master"
    id       = "ed-eks-master"
    name     = "ed-eks-master"
    # (5 unchanged attributes hidden)
}

aws_eks_cluster.eks: Creation complete after 7m1s [id=eks-cluster]
aws_eks_node_group.node-grp: Creating...
aws_eks_node_group.node-grp: Still creating... [10s elapsed]
aws_eks_node_group.node-grp: Still creating... [20s elapsed]
aws_eks_node_group.node-grp: Still creating... [30s elapsed]
aws_eks_node_group.node-grp: Still creating... [40s elapsed]
aws_eks_node_group.node-grp: Still creating... [50s elapsed]
aws_eks_node_group.node-grp: Still creating... [1m0s elapsed]
aws_eks_node_group.node-grp: Still creating... [1m10s elapsed]
aws_eks_node_group.node-grp: Still creating... [1m20s elapsed]
aws_eks_node_group.node-grp: Still creating... [1m30s elapsed]
aws_eks_node_group.node-grp: Still creating... [1m40s elapsed]
aws_eks_node_group.node-grp: Still creating... [1m50s elapsed]
aws_eks_node_group.node-grp: Creation complete after 1m51s [id=eks-cluster:pc-node-group]

Apply complete! Resources: 25 added, 0 changed, 3 destroyed.

```

Step 4: Verify the EKS-Cluster Creation in AWS Console.

- Log in to your AWS Management Console.

- Navigate to the EKS dashboard.
- Verify that a new “EKS-cluster” is created in “us-east-2” region.

Below is the EKS-Cluster:

The screenshot shows the AWS EKS Clusters dashboard. At the top, there's a navigation bar with the EKS logo and a breadcrumb 'Clusters'. Below this, there's a section titled 'Clusters (1)' with an 'Info' link. A search bar labeled 'Filter clusters' is present. To the right of the search bar are buttons for 'Refresh', 'Delete', and 'Add cluster'. Below the search bar is a table with the following columns: Cluster name, Status, Kubernetes version, Support period, and Provider. The table contains one entry: 'eks-cluster' with a status of 'Active', Kubernetes version '1.30', support period 'Standard support until July 28, 2025', and provider 'EKS'.

Cluster name	Status	Kubernetes version	Support period	Provider
eks-cluster	Active	1.30	Standard support until July 28, 2025	EKS

Cluster nodes and node groups:

The screenshot shows the AWS EKS Compute tab. At the top, there's a navigation bar with tabs for Overview, Resources, Compute (selected), Networking, Add-ons, Access, Observability, Upgrade insights, and Up. Below this, there's a section titled 'Nodes (2)' with an 'Info' link. A search bar labeled 'Filter Nodes by property or value' is present. To the right of the search bar are buttons for 'Refresh', 'Delete', and 'Add node group'. Below the search bar is a table with the following columns: Node name, Instance type, Node group, Created, and Status. The table contains two entries: 'ip-10-0-1-112.us-east-2.compute.internal' and 'ip-10-0-2-253.us-east-2.compute.internal', both with instance type 't2.small', node group 'pc-node-group', and status 'Ready'. Below this table is a section titled 'Node groups (1)' with an 'Info' link. A search bar labeled 'Filter Node groups by property or value' is present. To the right of the search bar are buttons for 'Edit', 'Delete', and 'Add node group'. Below the search bar is a table with the following columns: Group name, Desired size, AMI release version, Launch template, and Status. The table contains one entry: 'pc-node-group' with a desired size of 2, AMI release version '1.30.0-20240703', launch template '-', and status 'Active'.

Node name	Instance type	Node group	Created	Status
ip-10-0-1-112.us-east-2.compute.internal	t2.small	pc-node-group	Created 26 minutes ago	Ready
ip-10-0-2-253.us-east-2.compute.internal	t2.small	pc-node-group	Created 25 minutes ago	Ready

Group name	Desired size	AMI release version	Launch template	Status
pc-node-group	2	1.30.0-20240703	-	Active

Pods:

<

Overview

Resources

Compute

Networking

Add-ons

Access

Observability

Upgrade insights

Up >

Resource types

▼ Workloads

PodTemplates

Pods

ReplicaSets

Deployments

StatefulSets

DaemonSets

Jobs

CronJobs

PriorityClasses

HorizontalPodAutoscalers

► Cluster

Workloads: Pods (6)

View details

Pod is the smallest and simplest Kubernetes object. A Pod represents a set of running containers on your cluster.

Learn more

All Namespaces

Filter Pods by name

< 1 >

	Name	Age
<input type="radio"/>	aws-node-48xdv	Created 25 minutes ago
<input type="radio"/>	aws-node-nwst5	Created 25 minutes ago
<input type="radio"/>	coredns-858457f4f6-fk9rh	Created 27 minutes ago

Overview of cluster:

<

Overview

Resources

Compute

Networking

Add-ons

Access

Observability

Upgrade insights

Up >

Details

API server endpoint

https://A5BACF9642F37BD42AFA677E9BC07F03.gr7.us-east-2.eks.amazonaws.com

OpenID Connect provider URL

https://oidc.eks.us-east-2.amazonaws.com/id/A5BACF9642F37BD42AFA677E9BC07F03

Created

an hour ago

Certificate authority

LS0tLS1CRUdJTlBDRVJUSUZJQ0FURSOtLS0tCk1JSURCVENDQWUyZ0F3SUJBZ0lUSnVGa2FxTlAyclF3RFFZSkt

Cluster IAM role ARN

arn:aws:iam::637423437444:role/eks-cluster-cluster-20240712051310780200000002

View in IAM

Cluster ARN

arn:aws:eks:us-east-2:637423437444:cluster/eks-cluster

Platform version

Info

eks.5

Vpc, Subnets and security groups:

Navigation: Overview | Resources | Compute | **Networking** | Add-ons | Access | Observability | Upgrade insights | Up >

Networking

[Manage VPC resources](#)
[Manage endpoint access](#)

VPC Info vpc-08eed8b9ace9eb4d2 ↗ Cluster IP address family Info IPv4 Service IPv4 range Info 172.20.0.0/16	Subnets subnet-079f2e4cb0f995231 ↗ subnet-067f271067cc189e9 ↗	Cluster security group Info sg-0be2e1bb4afe0a7f ↗ Additional security groups None	API server endpoint access Info Public Public access source allowlist 0.0.0.0/0 (open to all traffic)
--	--	---	---

This are the three instances

- **Instances are Autoscaled: 2**
- **Main instance: 1**

Instances (3) [Info](#) [Refresh](#) [Connect](#) [Instance state](#) [Actions](#) [Launch instances](#)

Find Instance by attribute or tag (case-sensitive) [All states](#)

[Instance state = running](#) [Clear filters](#)

<input type="checkbox"/>	Name ↗	Instance ID	Instance state ↕	Instance type ↕	Status check	Alarm status	Availability Zone ↕
<input type="checkbox"/>		i-08e124e1cce314121	Running 🔍 🔍	t2.small	2/2 checks passed	View alarms +	us-east-2b
<input type="checkbox"/>	kubectl	i-0a82a0f71e37c514b	Running 🔍 🔍	t2.micro	2/2 checks passed	View alarms +	us-east-2a
<input type="checkbox"/>		i-0190ac907408fb35b	Running 🔍 🔍	t2.small	2/2 checks passed	View alarms +	us-east-2a

➤ **Autoscaling for kubernetes instances**

Auto Scaling groups (1) [Info](#) [Refresh](#) [Launch configurations](#) [Launch templates](#) [Actions](#) [Create Auto Scaling group](#)

Search your Auto Scaling groups

<input type="checkbox"/>	Name	Launch template/configuration ↗	Instances ↕	Status ↕	Desired capacity ↕
<input type="checkbox"/>	eks-pc-node-group-26c857f1-e0a6-2b6f-5def-0096259660e	eks-26c857f1-e0a6-2b6f-5def-00962596	2	-	2

To Check the kubernetes cluster in the cmd.

1) Update the context

- **aws eks update-kubeconfig --region "your region" --name "cluster-name"**

```
C:\Program Files\terraform\aws_eks>aws eks update-kubeconfig --region us-east-2 --name eks-cluster
Updated context arn:aws:eks:us-east-2:637423437444:cluster/eks-cluster in C:\Users\HP\.kube\config
```

2) Install kubernetes in windows cmd using below command:

- **curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.23.17/2024-0419/bin/windows/amd64/kubect.exe**

```
C:\Program Files\terraform\aws_eks>curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.23.17/2024-04-19/bin/windows/amd64/kubect.exe
% Total    % Received % Xferd  Average Speed   Time    Time     Current
   Dload  Upload   Total   Spent    Left     Speed
100 44.3M  100 44.3M    0     0  1638k      0  0:00:27  0:00:27 --:--:-- 1276k
```

3) Check the version of kubernetes

- **kubect version --client**

```
C:\Program Files\terraform\aws_eks>kubect version --client
Client Version: version.Info{Major:"1", Minor:"23+", GitVersion:"v1.23.17-eks-ae9a62a", GitCommit:"d1eaa64fc31d03fc013272320eb765297d183bd8", GitTreeState:"clean", BuildDate:"2024-04-11T19:00:23Z", GoVersion:"go1.21.8", Compiler:"gc", Platform:"windows/amd64"}
```

4) Check the nodes present in the cluster that we have created just.

- **kubectl get nodes**

```
C:\Program Files\terraform\aws_eks>kubectl get nodes
NAME                                STATUS    ROLES    AGE    VERSION
ip-10-0-1-112.us-east-2.compute.internal Ready    <none>    17m    v1.30.0-eks-036c24b
ip-10-0-2-253.us-east-2.compute.internal Ready    <none>    17m    v1.30.0-eks-036c24b
```

5) Get list services:

- **kubectl get svc**

```
C:\Program Files\terraform\aws_eks>kubectl get svc
NAME            TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
kubernetes      ClusterIP     172.20.0.1    <none>          443/TCP     21m
```