Document Created By: Obaidur Rahman
Date: 05/16/2023

Provision an EKS Cluster (AWS) using Terraform:

AWS's Elastic Kubernetes Service (EKS) is a managed service that lets you deploy, manage, and scale containerized applications on Kubernetes.

We will deploy an EKS cluster using Terraform. Then, we will configure **`kubectl`** using Terraform output and verify that our cluster is ready to use.

Why deploy with Terraform?

While you could use the built-in AWS provisioning processes (UI, CLI, CloudFormation) for EKS clusters, Terraform provides you with several benefits:

- **Unified Workflow** - If you already use Terraform to deploy AWS infrastructure, you can use the same workflow to deploy both EKS clusters and applications into those clusters.
- **Full Lifecycle Management** - Terraform creates, updates, and deletes tracked resources without requiring you to inspect an API to identify those resources.
- **Graph of Relationships** - Terraform determines and observes dependencies between resources. For example, if an AWS Kubernetes cluster needs a specific VPC and subnet configurations, Terraform will not attempt to create the cluster if it fails to provision the VPC and subnet first.

Prerequisites:

1. AWS Account with admin access.
2. Install Terraform in local machine Window/Mac/linux. Below is the link.

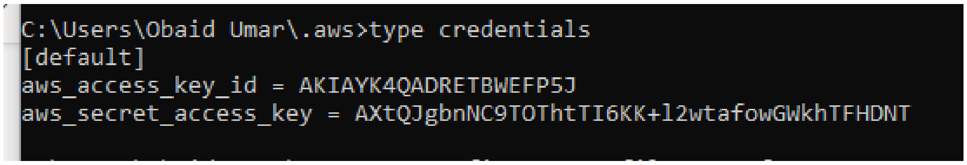
https://developer.hashicorp.com/terraform/downloads

Keep the terraform.exe in a specific folder. And run it from the command prompt to install it.

3. Download AWS CLI for windows and install it.

Once it is installed do the aws configure from command line to set a aws role to a user.

After configuring the aws cli, you will be able to see the credentials as below.

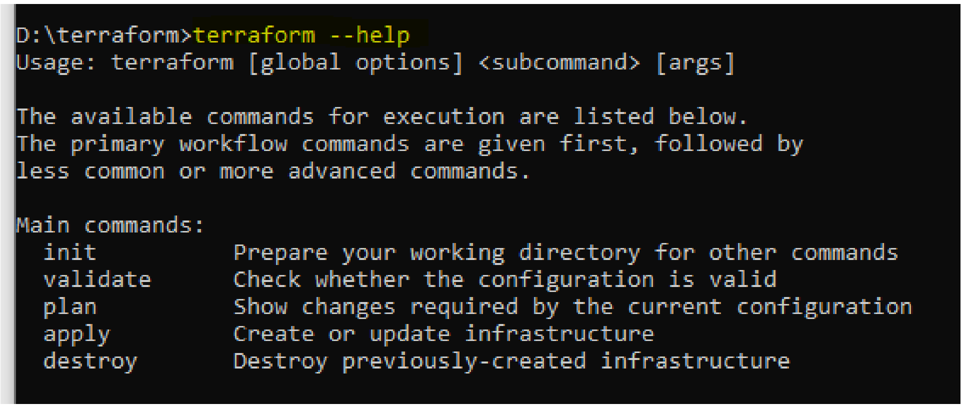


```
C:\Users\Obaid Umar\.aws>type credentials
[default]
aws_access_key_id = AKIAYK4QADRETBWEFP5J
aws_secret_access_key = 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
```

4. Its better to work in a visual studio coder editor for writing the codes. If it is not there download it.

Steps for Spinup the EKS Cluster using terraform:

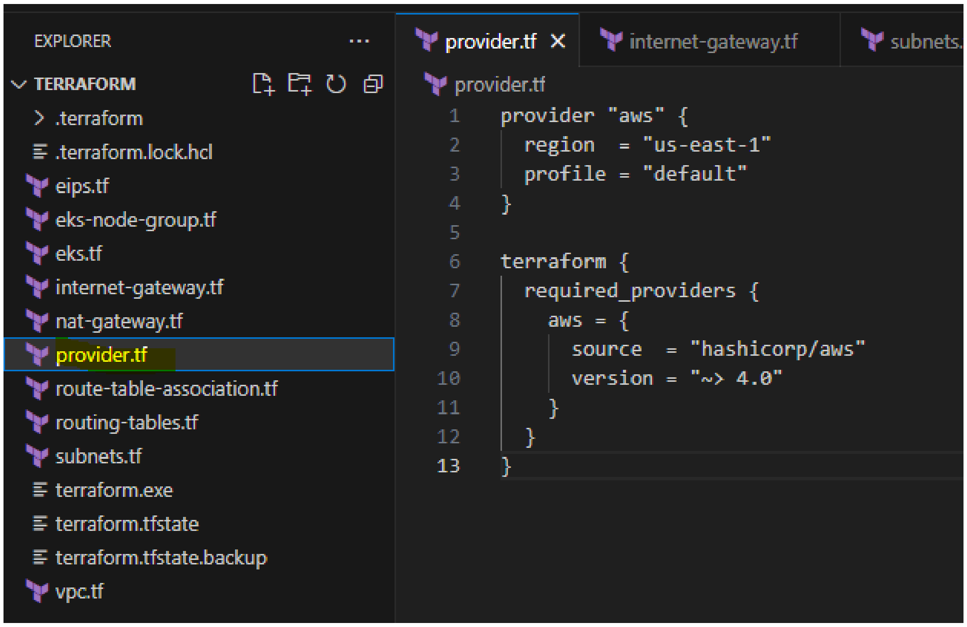
1. Test the terraform it it is working or not before start coding.



```
D:\terraform>terraform --help
Usage: terraform [global options] <subcommand> [args]

The available commands for execution are listed below.
The primary workflow commands are given first, followed by
less common or more advanced commands.

Main commands:
  init          Prepare your working directory for other commands
  validate      Check whether the configuration is valid
  plan          Show changes required by the current configuration
  apply         Create or update infrastructure
  destroy       Destroy previously-created infrastructure
```

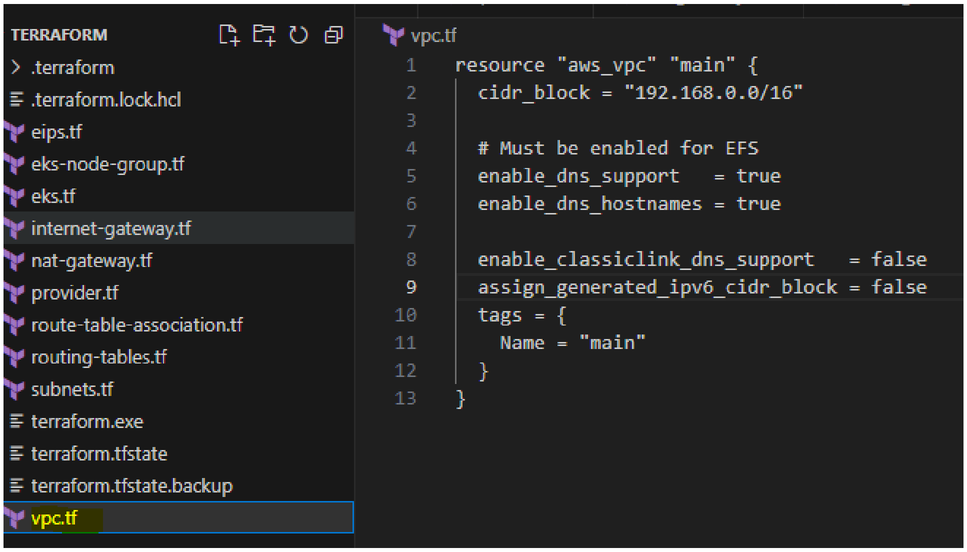
Now got visual studio and start implementing the aws infrastructure, before spinning up the eks we have to setup the networking 1st .

The very 1st thing is to set the provider, we will go ahead and create a file calling **provider.tf** as below.



```
provider "aws" {
  region  = "us-east-1"
  profile = "default"
}

terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 4.0"
    }
  }
}
```

And then will create a terraform file for VPC as **vpc.tf**



```
resource "aws_vpc" "main" {
  cidr_block = "192.168.0.0/16"

  # Must be enabled for EFS
  enable_dns_support   = true
  enable_dns_hostnames = true

  enable_classiclink_dns_support   = false
  assign_generated_ipv6_cidr_block = false
  tags = {
    Name = "main"
  }
}
```

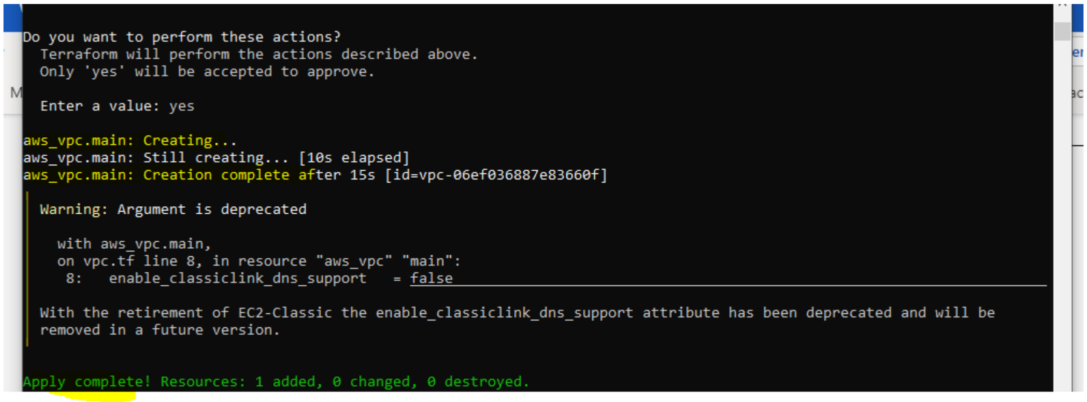
Once these two file s are ready will do to the command line and run the below commands

$ terraform fmt #It will format the terafrom files

$ terraform init # to initialize the terraform

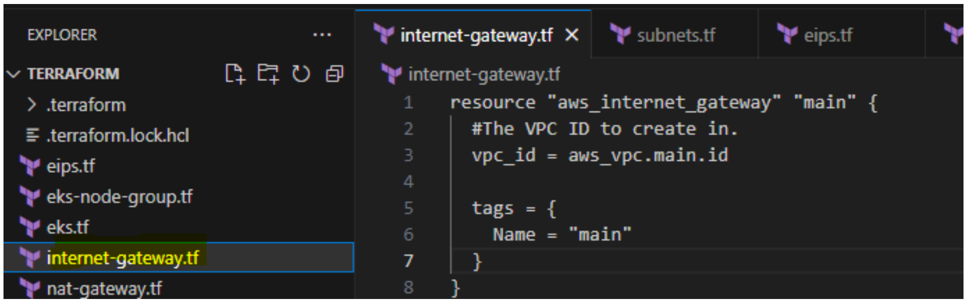
$ terraform plan # this for Dry run

$ terraform apply # Now it will implement the requested resources.



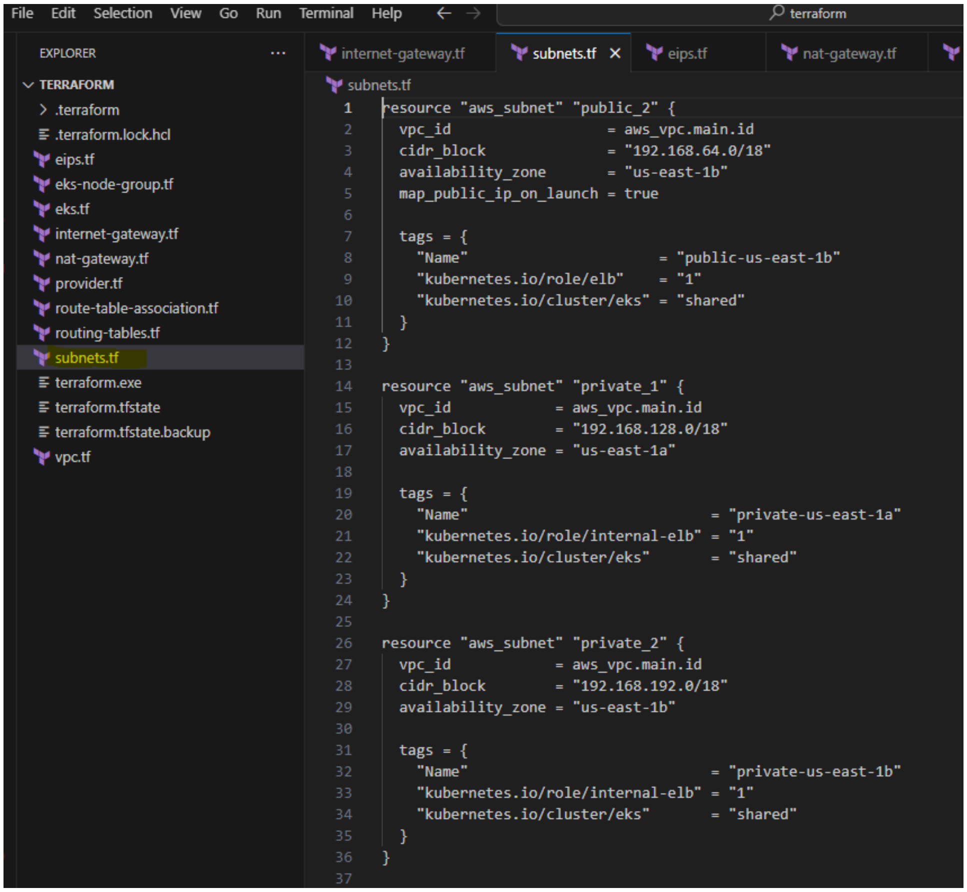
```
Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_vpc.main: Creating...
aws_vpc.main: Still creating... [10s elapsed]
aws_vpc.main: Creation complete after 15s [id=vpc-06ef036887e83660f]

  Warning: Argument is deprecated

    with aws_vpc.main,
    on vpc.tf line 8, in resource "aws_vpc" "main":
     8:   enable_classiclink_dns_support   = false

  With the retirement of EC2-Classic the enable_classiclink_dns_support attribute has been deprecated and will be
  removed in a future version.


Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

Now similarly we have to create **Internet Gateway, Subnets, NAT Gateway, Routing Tables** in sequence as per the terraform script.



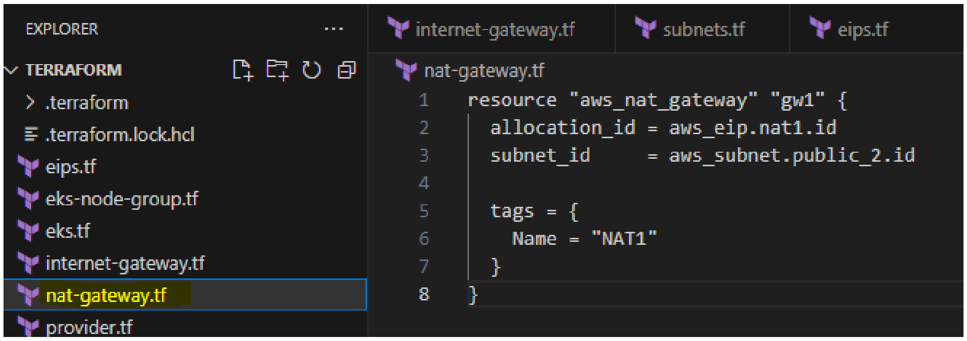
```
resource "aws_internet_gateway" "main" {
  #The VPC ID to create in.
  vpc_id = aws_vpc.main.id

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

Internet Gateway



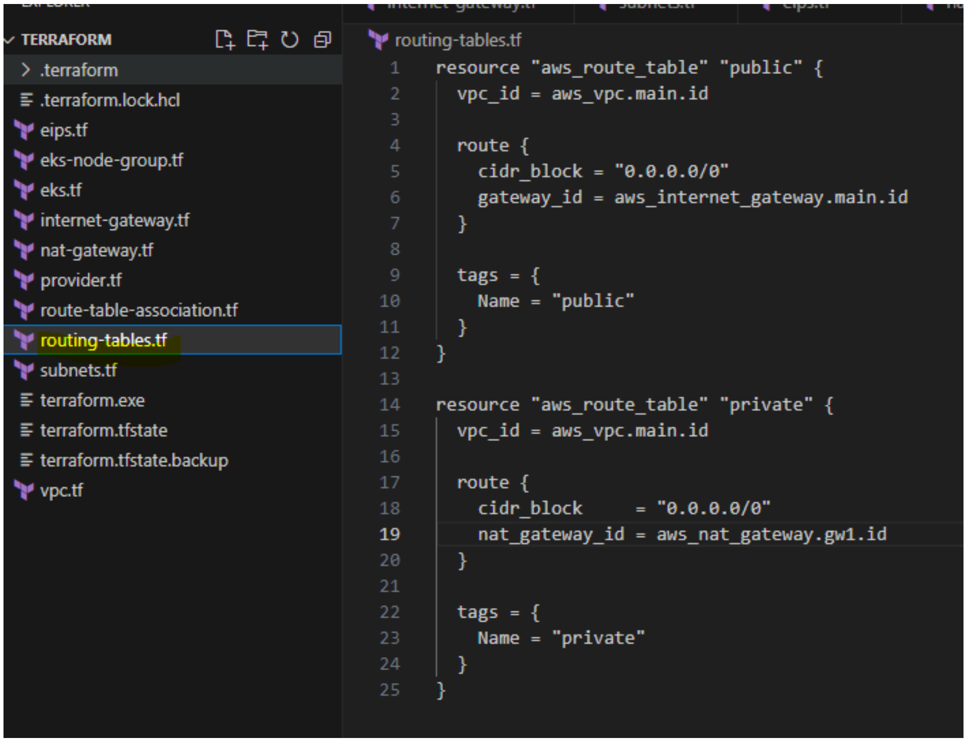
```
resource "aws_subnet" "public_2" {
  vpc_id                  = aws_vpc.main.id
  cidr_block              = "192.168.64.0/18"
  availability_zone       = "us-east-1b"
  map_public_ip_on_launch = true

  tags = {
    "Name"                        = "public-us-east-1b"
    "kubernetes.io/role/elb"      = "1"
    "kubernetes.io/cluster/eks" = "shared"
  }
}

resource "aws_subnet" "private_1" {
  vpc_id            = aws_vpc.main.id
  cidr_block        = "192.168.128.0/18"
  availability_zone = "us-east-1a"

  tags = {
    "Name"                              = "private-us-east-1a"
    "kubernetes.io/role/internal-elb" = "1"
    "kubernetes.io/cluster/eks"       = "shared"
  }
}

resource "aws_subnet" "private_2" {
  vpc_id            = aws_vpc.main.id
  cidr_block        = "192.168.192.0/18"
  availability_zone = "us-east-1b"

  tags = {
    "Name"                              = "private-us-east-1b"
    "kubernetes.io/role/internal-elb" = "1"
    "kubernetes.io/cluster/eks"       = "shared"
  }
}
```

Subnetting.



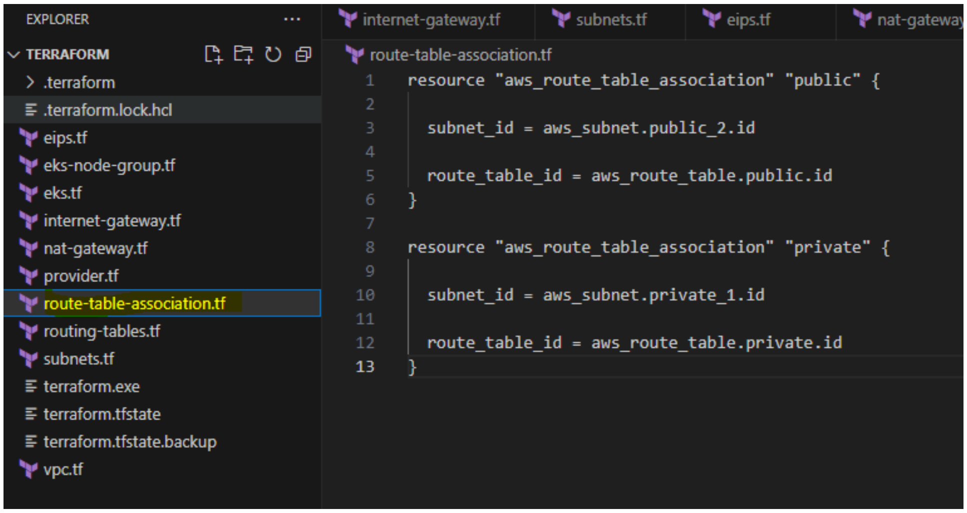
```
resource "aws_nat_gateway" "gw1" {
  allocation_id = aws_eip.nat1.id
  subnet_id     = aws_subnet.public_2.id

  tags = {
    Name = "NAT1"
  }
}
```

NAT Gatway.



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

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

  tags = {
    Name = "public"
  }
}

resource "aws_route_table" "private" {
  vpc_id = aws_vpc.main.id

  route {
    cidr_block     = "0.0.0.0/0"
    nat_gateway_id = aws_nat_gateway.gw1.id
  }

  tags = {
    Name = "private"
  }
}
```

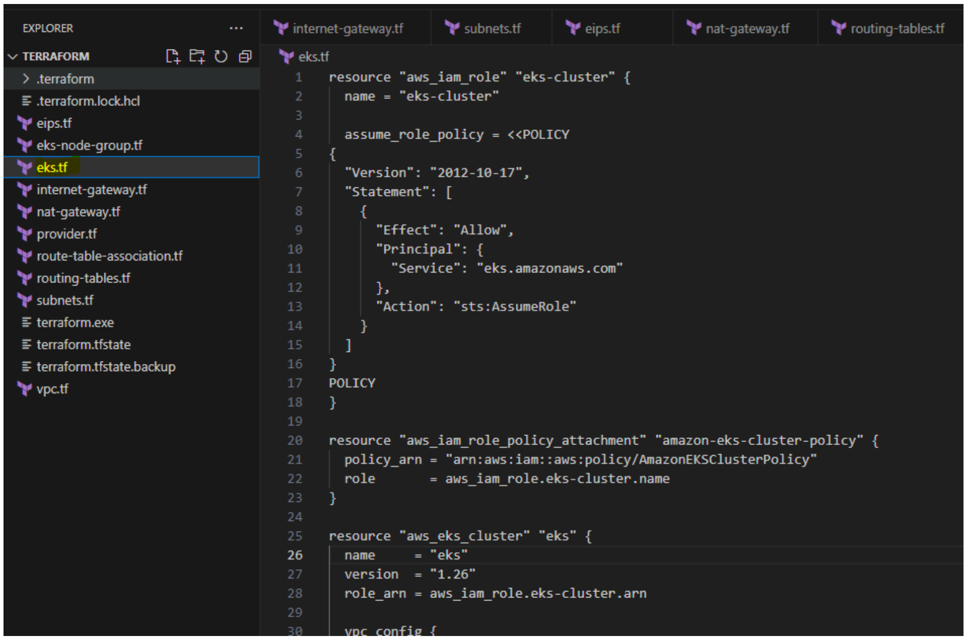
Routing tables



```
resource "aws_route_table_association" "public" {

  subnet_id = aws_subnet.public_2.id

  route_table_id = aws_route_table.public.id
}

resource "aws_route_table_association" "private" {

  subnet_id = aws_subnet.private_1.id

  route_table_id = aws_route_table.private.id
}
```

Route Table association

And Finally We have to create 2 file for EKS Cluster spin up one we can call it **eks.tf** in whichwe have to se some Policy and assume roles to be used for createing the cluster.



```
resource "aws_iam_role" "eks-cluster" {
  name = "eks-cluster"

  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" "amazon-eks-cluster-policy" {
  policy_arn = "arn:aws:iam::aws:policy/AmazonEKSClusterPolicy"
  role       = aws_iam_role.eks-cluster.name
}

resource "aws_eks_cluster" "eks" {
  name     = "eks"
  version  = "1.26"
  role_arn = aws_iam_role.eks-cluster.arn

  vpc_config {
```

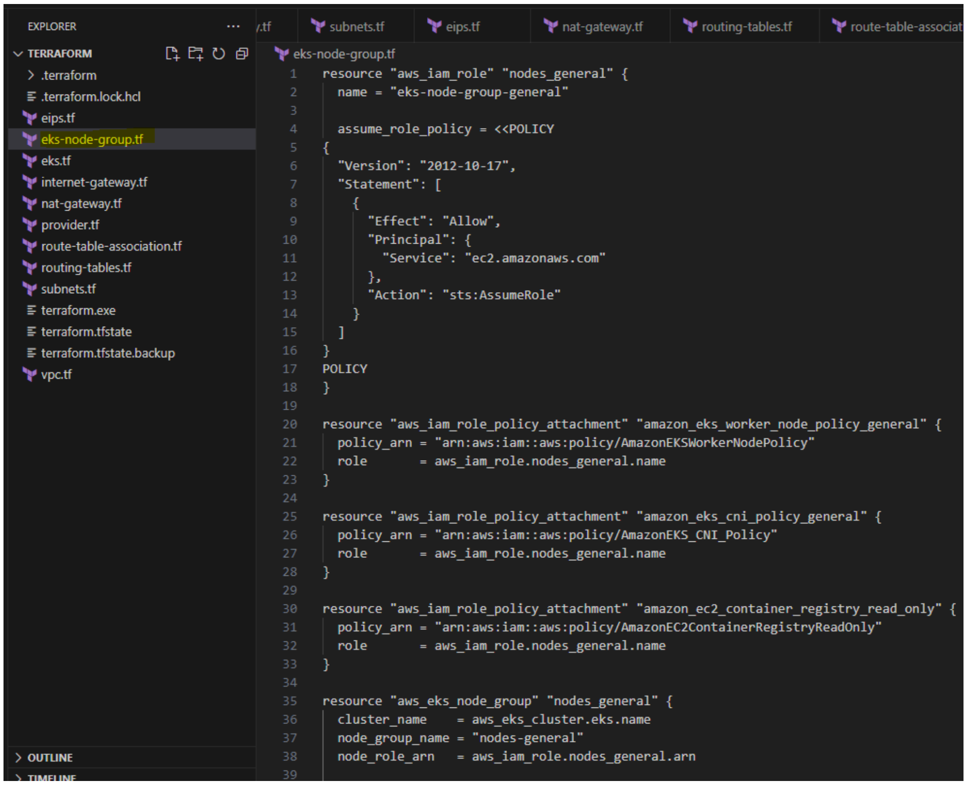
And after that we have to create eks-node-group.tf file which is the most imposrtant file in which we have set the Additional assume roles as below.

AmazonEKSWorkerNodePolicy,

AmazonEKS_CNI_Policy,

AmazonEC2ContainerRegistryReadOnly.

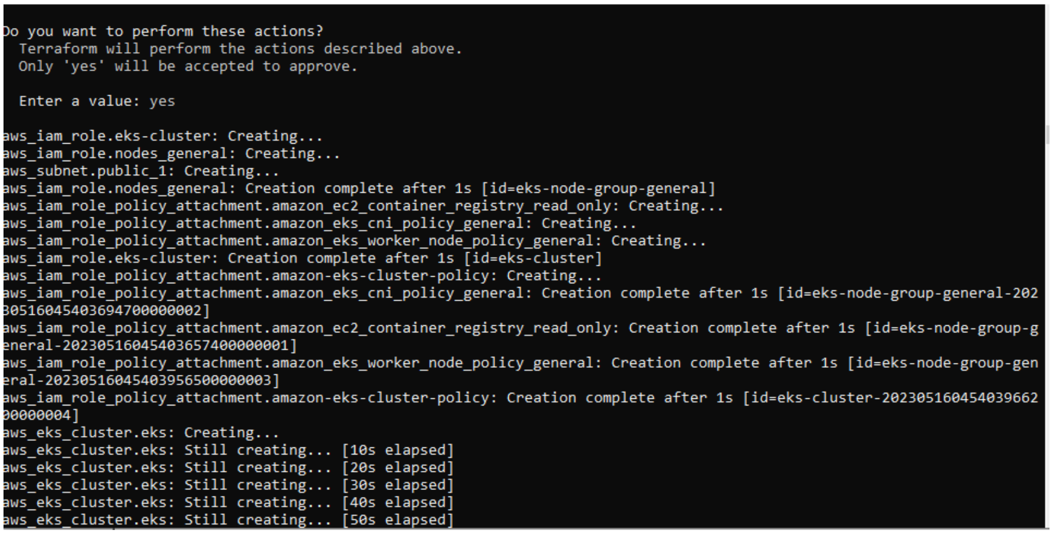
And also EcC2 instance resource type size and version of EKS cluster every thing is setop in these two files.



```
resource "aws_iam_role" "nodes_general" {
  name = "eks-node-group-general"

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

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

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

resource "aws_iam_role_policy_attachment" "amazon_ec2_container_registry_read_only" {
  policy_arn = "arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly"
  role       = aws_iam_role.nodes_general.name
}

resource "aws_eks_node_group" "nodes_general" {
  cluster_name    = aws_eks_cluster.eks.name
  node_group_name = "nodes-general"
  node_role_arn   = aws_iam_role.nodes_general.arn
```

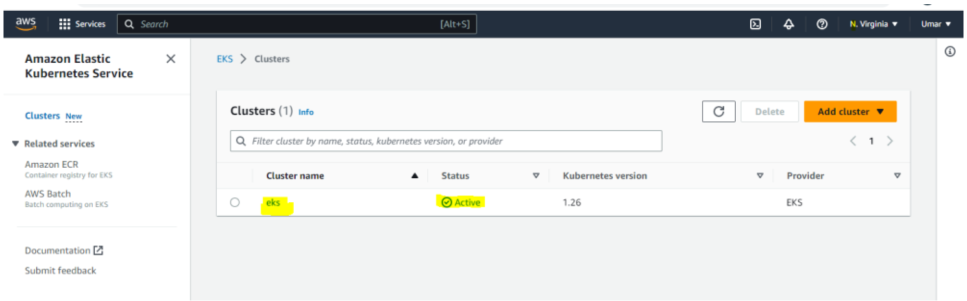
Once this two files ae ready will go to the terraform window and hit the same commands as below.

```
$ terraform fmt          #It will format the terafrom files

$ terraform init         # to initialize the terraform

$ terraform plan         # this for Dry run

$ terraform apply        # Now it will implement the requested resources.
```

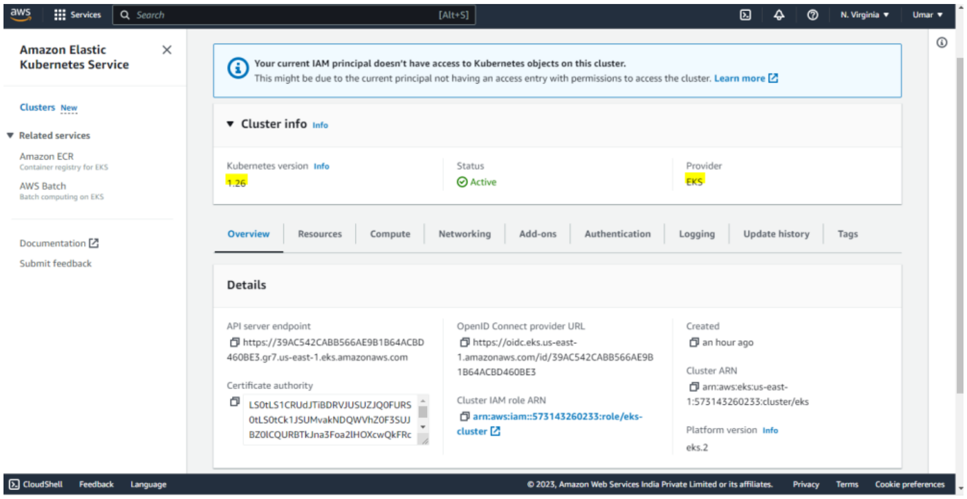


```
Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_iam_role.eks-cluster: Creating...
aws_iam_role.nodes_general: Creating...
aws_subnet.public_1: Creating...
aws_iam_role.nodes_general: Creation complete after 1s [id=eks-node-group-general]
aws_iam_role_policy_attachment.amazon_ec2_container_registry_read_only: Creating...
aws_iam_role_policy_attachment.amazon_eks_cni_policy_general: Creating...
aws_iam_role_policy_attachment.amazon_eks_worker_node_policy_general: Creating...
aws_iam_role.eks-cluster: Creation complete after 1s [id=eks-cluster]
aws_iam_role_policy_attachment.amazon-eks-cluster-policy: Creating...
aws_iam_role_policy_attachment.amazon_eks_cni_policy_general: Creation complete after 1s [id=eks-node-group-general-202
30516045403694700000002]
aws_iam_role_policy_attachment.amazon_ec2_container_registry_read_only: Creation complete after 1s [id=eks-node-group-g
eneral-20230516045403657400000001]
aws_iam_role_policy_attachment.amazon_eks_worker_node_policy_general: Creation complete after 1s [id=eks-node-group-gen
eral-20230516045403956500000003]
aws_iam_role_policy_attachment.amazon-eks-cluster-policy: Creation complete after 1s [id=eks-cluster-202305160454039662
00000004]
aws_eks_cluster.eks: Creating...
aws_eks_cluster.eks: Still creating... [10s elapsed]
aws_eks_cluster.eks: Still creating... [20s elapsed]
aws_eks_cluster.eks: Still creating... [30s elapsed]
aws_eks_cluster.eks: Still creating... [40s elapsed]
aws_eks_cluster.eks: Still creating... [50s elapsed]
```

After completion of this terraform apply the EKS cluster will be ready.



	Cluster name	Status	Kubernetes version	Provider
○	eks	Active	1.26	EKS



Your current IAM principal doesn't have access to Kubernetes objects on this cluster.
This might be due to the current principal not having an access entry with permissions to access the cluster. Learn more

▼ Cluster info Info

Kubernetes version Info	Status	Provider
1.26	✓ Active	EKS

Details

API server endpoint
https://39AC542CABB566AE9B1B64ACBD460BE3.gr7.us-east-1.eks.amazonaws.com

Certificate authority
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUMvakNDQWVhZ0F3SUJBZ0lCQURBTkJna3Foa2lHOXcwQkFRc

OpenID Connect provider URL
https://oidc.eks.us-east-1.amazonaws.com/id/39AC542CABB566AE9B1B64ACBD460BE3

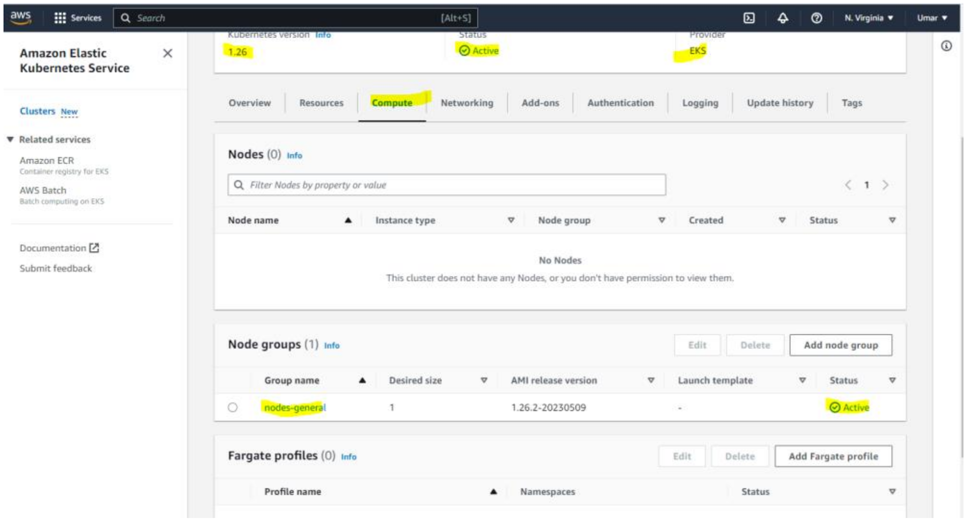
Cluster IAM role ARN
arn:aws:iam::573143260233:role/eks-cluster

Created
an hour ago

Cluster ARN
arn:aws:eks:us-east-1:573143260233:cluster/eks

Platform version Info
eks.2

CloudShell
Feedback
Language
© 2023, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences



Kubernetes version Info	Status	Provider
1.26	✓ Active	EKS

Nodes (0) Info

Filter Nodes by property or value

Node name	Instance type	Node group	Created	Status

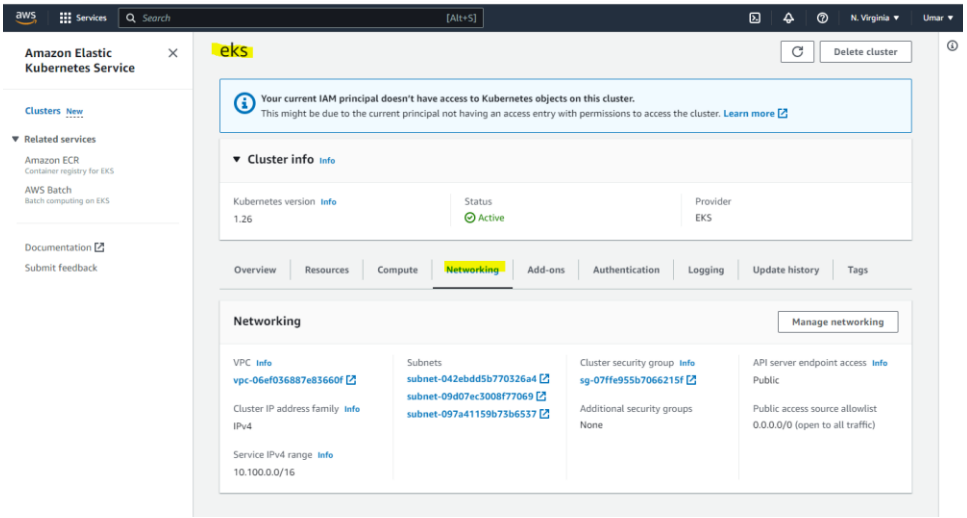
No Nodes
This cluster does not have any Nodes, or you don't have permission to view them.

Node groups (1) Info

	Group name	Desired size	AMI release version	Launch template	Status
○	nodes-general	1	1.26.2-20230509	-	✓ Active

Fargate profiles (0) Info

	Profile name	Namespaces	Status



eks

Your current IAM principal doesn't have access to Kubernetes objects on this cluster.
This might be due to the current principal not having an access entry with permissions to access the cluster. Learn more

▾ Cluster info Info

Kubernetes version Info
1.26

Status
Active

Provider
EKS

Networking

VPC Info
vpc-06ef036887e83660f

Cluster IP address family Info
IPv4

Service IPv4 range Info
10.100.0.0/16

Subnets
subnet-042ebdd5b770326a4
subnet-09d07ec3008f77069
subnet-097a41159b73b6537

Cluster security group Info
sg-07ffe955b7066215f

Additional security groups
None

API server endpoint access Info
Public

Public access source allowlist
0.0.0.0/0 (open to all traffic)

All the tl File will be shared view Once dirve folder.