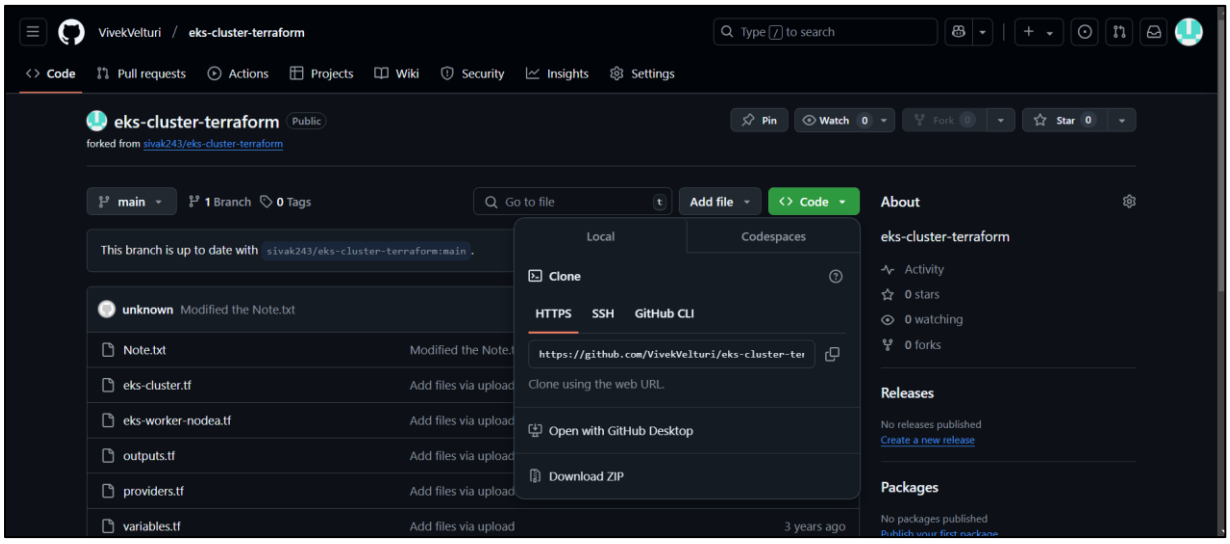
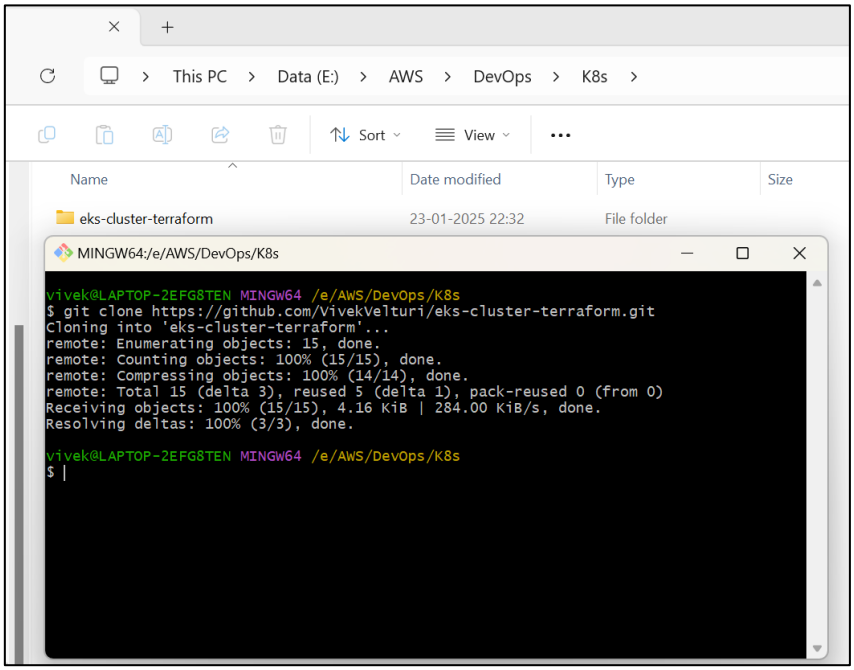
EKS Cluster Setup and Application Deployment with NodePort Access Using Terraform

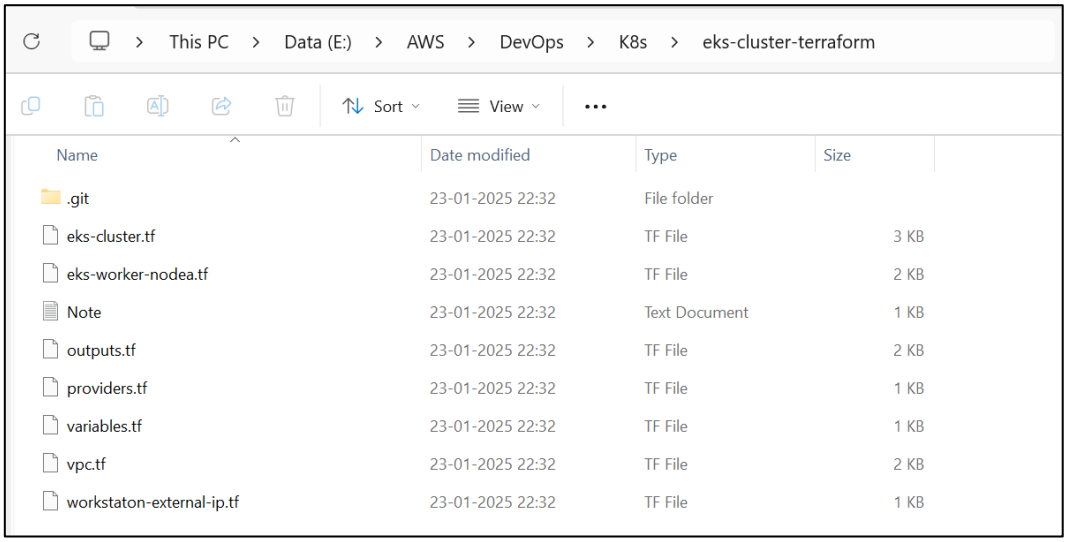
In this Project we will be setting up EKS Cluster using terraform and Deploy Pods with Nodeport Access

- Clone the Terraform scripts from the GitHub.
    - Copy the URL for the Terraform script and clone them into folder in your local computer using the command git clone <URL of the scripts repository> in my case: git clone https://github.com/VivekVelturi/eks-cluster-terraform.git





The copied terraform script is as follows:



eks-cluster.tf

```
#
# EKS Cluster Resources
#  * IAM Role to allow EKS service to manage other AWS services
#  * EC2 Security Group to allow networking traffic with EKS cluster
#  * EKS Cluster
#

resource "aws_iam_role" "demo-cluster" { # Creates an IAM role for the EKS cluster
  name = "terraform-eks-demo-cluster" # Name of the IAM role

  assume_role_policy = <<POLICY
{ # Policy allowing EKS service to assume this IAM role
  "Version": "2012-10-17", # Defines the policy version
  "Statement": [ # List of statements granting permissions
    {
      "Effect": "Allow", # Allows the action
      "Principal": { # The entity that can assume the role
        "Service": "eks.amazonaws.com" # Specifies the EKS service
      },
      "Action": "sts:AssumeRole" # Grants permission to assume this role
    }
  ]
}
POLICY
}

resource "aws_iam_role_policy_attachment" "demo-cluster-
AmazonEKSClusterPolicy" {
```

```
  policy_arn = "arn:aws:iam::aws:policy/AmazonEKSClusterPolicy" # Attaches the EKS Cluster Policy
  role       = aws_iam_role.demo-cluster.name # Associates the policy with the demo-cluster IAM role
}

resource "aws_iam_role_policy_attachment" "demo-cluster-AmazonEKSVPCResourceController" {
  policy_arn = "arn:aws:iam::aws:policy/AmazonEKSVPCResourceController" # Attaches the VPC Resource Controller policy
  role       = aws_iam_role.demo-cluster.name # Associates the policy with the IAM role
}

resource "aws_security_group" "demo-cluster" {
  name        = "terraform-eks-demo-cluster" # Security group name
  description = "Cluster communication with worker nodes" # Description of the security group
  vpc_id      = aws_vpc.demo.id # Associates the security group with a specific VPC

  egress { # Egress rule to allow all outbound traffic
    from_port   = 0 # Start of port range (0 means all)
    to_port     = 0 # Start of port range (0 means all)
    protocol    = "-1" # Protocol "-1" allows all protocols
    cidr_blocks = ["0.0.0.0/0"] # CIDR block for all IPv4 addresses
  }

  tags = { # Tags for identifying the resource
    Name = "terraform-eks-demo" # Name tag for the security group
  }
}

resource "aws_security_group_rule" "demo-cluster-ingress-workstation-https" {
  cidr_blocks       = [local.workstation-external-cidr] # Allows access from the workstation's external IP range
  description       = "Allow workstation to communicate with the cluster API Server" # Description of the rule
  from_port         = 443 # Start of port range (443 is used for HTTPS)
  protocol          = "tcp" # Restricts to TCP protocol
  security_group_id = aws_security_group.demo-cluster.id # Associates the rule with the demo-cluster security group
  to_port           = 443 # End of port range (443 is used for HTTPS).
  type              = "ingress" # Ingress type means traffic coming into the cluster
}

resource "aws_eks_cluster" "demo" {
  name     = var.cluster-name # Name of the cluster, passed as a variable
  role_arn = aws_iam_role.demo-cluster.arn # IAM role to be used by the EKS cluster
```

```
  vpc_config { # VPC networking configuration for the cluster
    security_group_ids = [aws_security_group.demo-cluster.id] # VPC networking configuration for the cluster
    subnet_ids         = aws_subnet.demo[*].id # Subnets for the cluster (referenced from another resource)
  }

  depends_on = [ # Ensures these policies are attached before creating the cluster
    aws_iam_role_policy_attachment.demo-cluster-AmazonEKSClusterPolicy,
    aws_iam_role_policy_attachment.demo-cluster-AmazonEKSVPCResourceController,
  ]
}
```

This Terraform configuration:

1. Creates an IAM role with necessary permissions for EKS.
2. Sets up a security group for secure communication.
3. Deploys an EKS cluster in a VPC with the above configurations.

eks-worker-nodea.tf

```
#
# EKS Worker Nodes Resources
#  * IAM role allowing Kubernetes actions to access other AWS services
#  * EKS Node Group to launch worker nodes
#

resource "aws_iam_role" "demo-node" { # Creates an IAM role for EKS worker nodes
  name = "terraform-eks-demo-node" # Assigns a name to the IAM role

  assume_role_policy = <<POLICY # Policy that allows the EC2 service to assume this IAM role
{
  "Version": "2012-10-17", # Policy version
  "Statement": [ # List of policy statements
    {
      "Effect": "Allow", # Allows the specified action
      "Principal": { # The entity that can assume this role
        "Service": "ec2.amazonaws.com" # Specifies EC2 instances
      },
      "Action": "sts:AssumeRole" # Grants permission to assume this role
    }
  ]
}
POLICY
}
```

```
resource "aws_iam_role_policy_attachment" "demo-node-
AmazonEKSWorkerNodePolicy" {
  policy_arn = "arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy" # Attaches the
Worker Node Policy to the IAM role
  role       = aws_iam_role.demo-node.name # Associates the policy with the demo-node
IAM role
}

resource "aws_iam_role_policy_attachment" "demo-node-AmazonEKS_CNI_Policy" {
  policy_arn = "arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy" # Attaches the
Amazon EKS CNI Policy
  role       = aws_iam_role.demo-node.name # Associates the policy with the demo-node
IAM role
}

resource "aws_iam_role_policy_attachment" "demo-node-
AmazonEC2ContainerRegistryReadOnly" {
  policy_arn = "arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly" #
Attaches a policy to allow read-only access to ECR
  role       = aws_iam_role.demo-node.name # Associates the policy with the demo-node
IAM role
}

resource "aws_eks_node_group" "demo" { # Creates an EKS Node Group for worker nodes
  cluster_name    = aws_eks_cluster.demo.name # Specifies the name of the EKS cluster to
associate with the node group
  node_group_name = "demo" # Assigns a name to the node group
  node_role_arn   = aws_iam_role.demo-node.arn # Specifies the IAM role to be assumed
by the worker nodes
  subnet_ids      = aws_subnet.demo[*].id # Associates the node group with the specified
subnets

  scaling_config { # Configures scaling for the node group
    desired_size = 1 # Sets the desired number of nodes in the group
    max_size     = 1 # Sets the maximum number of nodes
    min_size     = 1 # Sets the minimum number of nodes
  }

  depends_on = [ # Ensures the IAM policies are attached before creating the node group
    aws_iam_role_policy_attachment.demo-node-AmazonEKSWorkerNodePolicy,
    aws_iam_role_policy_attachment.demo-node-AmazonEKS_CNI_Policy,
    aws_iam_role_policy_attachment.demo-node-
AmazonEC2ContainerRegistryReadOnly,
  ]
}
```

This configuration sets up the resources needed for EKS worker nodes:

1. An IAM role (`aws_iam_role.demo-node`) is created, allowing EC2 instances (worker nodes) to assume the role and interact with AWS services.

2. Three managed policies are attached to the role:
    - `AmazonEKSWorkerNodePolicy`: Grants worker nodes the necessary permissions to interact with the EKS cluster.
    - `AmazonEKS_CNI_Policy`: Enables the nodes to manage network interfaces required for Kubernetes pods.
    - `AmazonEC2ContainerRegistryReadOnly`: Provides read-only access to the Amazon Elastic Container Registry (ECR) for pulling container images.
3. An EKS Node Group (`aws_eks_node_group.demo`) is created to launch and manage worker nodes. It uses the IAM role and is associated with specific subnets. The scaling configuration is set to maintain exactly one worker node in the group. The `depends_on` block ensures that the IAM role and its policies are attached before creating the node group, avoiding dependency issues.

outputs.tf

```
#
# Outputs
#

locals {  # Defines reusable local variables for the Terraform configuration
  config_map_aws_auth = <<CONFIGMAPAWSAUTH # Creates a multi-line string for the "aws-auth" ConfigMap in Kubernetes


apiVersion: v1 # Specifies the Kubernetes API version used for the ConfigMap
kind: ConfigMap # Declares the resource type as ConfigMap
metadata: # Metadata section for identifying the resource
  name: aws-auth # Assigns the name "aws-auth" to the ConfigMap
  namespace: kube-system # Places the ConfigMap in the "kube-system" namespace
data: # Contains the actual data for the ConfigMap
  mapRoles: | # Maps IAM roles to Kubernetes users and groups
    - rolearn: ${aws_iam_role.demo-node.arn} # References the ARN of the IAM role for worker nodes
      username: system:node:{{EC2PrivateDNSName}} # Sets the username format for the worker nodes
      groups: # Assigns worker nodes to Kubernetes groups
        - system:bootstrappers # Group responsible for bootstrapping nodes
        - system:nodes # Group for nodes in the Kubernetes cluster
CONFIGMAPAWSAUTH # End of the multi-line string for the ConfigMap

  kubeconfig = <<KUBECONFIG # Creates a multi-line string for the Kubernetes kubeconfig file


apiVersion: v1 # Specifies the Kubernetes API version used for the kubeconfig
clusters: # List of Kubernetes clusters the kubeconfig connects to
- cluster: # Cluster definition block
    server: ${aws_eks_cluster.demo.endpoint} # References the API server endpoint for the EKS cluster
```

```
    certificate-authority-data:
${aws_eks_cluster.demo.certificate_authority[0].data} # Specifies the certificate for
secure communication.
  name: Kubernetes # Assigns a name to the cluster
contexts: # List of contexts defining how to connect to the cluster
- context: # Context definition block
    cluster: Kubernetes # Specifies the cluster name for the context
    user: aws # Specifies the user for authentication
  name: aws # Assigns a name to the context
current-context: aws # Sets the current context to "aws"
kind: Config # Declares the resource type as Config
preferences: {} # Placeholder for user preferences (empty in this case)
users: # List of users for the kubeconfig
- name: aws # Assigns the name "aws" to the user
  user: # User authentication method
    exec: # Executes an external command for authentication
      apiVersion: client.authentication.k8s.io/v1beta1 # API version for the
authentication command
      command: aws-iam-authenticator # Specifies the authentication tool
      args: # List of arguments for the authenticator command
        - "token" # Generates a token for authentication
        - "-i" # Specifies the cluster name as an input parameter
        - "${var.cluster-name}" # References the cluster name variable
KUBECONFIG # End of the multi-line string for the kubeconfig
}

output "config_map_aws_auth" { # Declares an output for the "aws-auth" ConfigMap
  value = local.config_map_aws_auth # Outputs the content of the aws-auth ConfigMap
template
}

output "kubeconfig" { # Declares an output for the kubeconfig file
  value = local.kubeconfig # Outputs the content of the kubeconfig template
}
```

This configuration generates and outputs two essential resources for the EKS cluster:

1. **`config_map_aws_auth`**:
   - A Kubernetes ConfigMap named `aws-auth` is generated. It maps the worker node IAM role (`demo-node`) to Kubernetes users and groups, allowing the nodes to join the cluster and perform their functions.
   - The `system:bootstrappers` and `system:nodes` groups give worker nodes the required permissions in Kubernetes.
2. **`kubeconfig`**:
   - A kubeconfig file is generated, which contains the necessary configuration to connect to the EKS cluster. It includes the API server endpoint, the certificate for secure communication, and the authentication method using `aws-iam-authenticator`.

By outputting these configurations, users can easily deploy them to their Kubernetes cluster and access the cluster securely from their local machine or other tools.

vpc.tf

```
#
# VPC Resources
#  * VPC
#  * Subnets
#  * Internet Gateway
#  * Route Table
#

resource "aws_vpc" "demo" {  # Defines an AWS Virtual Private Cloud (VPC) resource
  cidr_block = "10.0.0.0/16"  # Specifies the CIDR block for the VPC, which provides a large address space (`10.0.0.0/16`)

  tags = tomap({  # Tags for the VPC, for organizational and identification purposes
    "Name"                                      = "terraform-eks-demo-node",
    "kubernetes.io/cluster/${var.cluster-name}" = "shared",
  })
}

resource "aws_subnet" "demo" {  # Defines an AWS Subnet resource
  count = 2  # Creates two subnets (subnet1 and subnet2)

  availability_zone       = data.aws_availability_zones.available.names[count.index]  # Fetches the availability zone names in a loop using count.index.
  cidr_block              = "10.0.${count.index}.0/24"  # Generates CIDR blocks `10.0.0.0/24` and `10.0.1.0/24`
  map_public_ip_on_launch = true  # Maps public IPs to the subnet upon launch (enables internet access)
  vpc_id                  = aws_vpc.demo.id  # Associates the subnet with the previously created VPC (id)

  tags = tomap({  # Tags applied to each subnet for organization and identification
    "Name"                                      = "terraform-eks-demo-node",
    "kubernetes.io/cluster/${var.cluster-name}" = "shared",
  })
}

resource "aws_internet_gateway" "demo" {  # Defines an Internet Gateway (IGW) for the VPC
  vpc_id = aws_vpc.demo.id  # Attaches the internet gateway to the VPC (id)

  tags = {  # Tags to help identify the internet gateway
    Name = "terraform-eks-demo"
  }
}

resource "aws_route_table" "demo" {  # Defines an AWS Route Table
```

```
  vpc_id = aws_vpc.demo.id # The route table is associated with the VPC created earlier

  route { # Creates a route within the route table
    cidr_block = "0.0.0.0/0" # Default route for all traffic
    gateway_id = aws_internet_gateway.demo.id # Route traffic to the internet via the internet gateway
  }
}

resource "aws_route_table_association" "demo" { # Associates the created route table with subnets
  count = 2 # Associates the route table with two subnets (subnet1 and subnet2)

  subnet_id      = aws_subnet.demo.*.id[count.index] # Iterates over both subnets
  route_table_id = aws_route_table.demo.id # Associates the subnets with the route table created earlier
}
```

This set of Terraform resources builds the foundational networking setup for an AWS cluster:

1. **`aws_vpc.demo`**: Creates a VPC with a large CIDR block (`10.0.0.0/16`) and applies organizational tags.
2. **`aws_subnet.demo`**: Creates two subnets (`10.0.0.0/24` and `10.0.1.0/24`), each in separate availability zones, with public IPs mapped at launch.
3. **`aws_internet_gateway.demo`**: Attaches an internet gateway to the VPC to allow internet access.
4. **`aws_route_table.demo`**: Creates a route table for the VPC, routing all traffic to the internet via the internet gateway.
5. **`aws_route_table_association.demo`**: Associates the created route table with both subnets to ensure proper internet connectivity.

This configuration ensures that instances deployed within the VPC have reliable connectivity to the internet, enabling communication with other AWS services or external systems.

providers.tf

```
terraform { # The block to define Terraform settings
  required_version = ">= 0.12" # Ensures that the Terraform version used is 0.12 or higher
}

provider "aws" { # Specifies the AWS provider block for interacting with AWS services
  region = var.aws_region # Sets the AWS region using a variable (`var.aws_region`) for flexibility
}

data "aws_availability_zones" "available" {} # Retrieves the list of available AWS availability zones in the chosen region

# Not required: currently used in conjunction with using
# icanhazip.com to determine local workstation external IP
# to open EC2 Security Group access to the Kubernetes cluster.
```

```
# See workstation-external-ip.tf for additional information.
provider "http" {} # Enables the HTTP provider for making HTTP requests
```

This configuration sets up the foundational elements for Terraform to interact with AWS and other services:

1. **Terraform Block**: Ensures the Terraform version is 0.12 or higher, ensuring compatibility with modern features.
2. **AWS Provider**: Configures the AWS provider to operate in a specified region, which is dynamically set using the `var.aws_region` variable.
3. **Data Source (`aws_availability_zones`)**: Retrieves all available AWS availability zones in the specified region, useful for resource placement and high availability.
4. **HTTP Provider**: Although not directly used in this code, it enables the ability to perform HTTP requests, which can be leveraged by other scripts (e.g., to fetch the local IP for dynamic security configurations).

This setup provides the groundwork for deploying infrastructure on AWS while also enabling advanced configurations such as dynamic Security Group rules based on external IP addresses.

variables.tf

```
variable "aws_region" { # Declares a variable named `aws_region`
  default = "us-west-2" # Sets the default value of the AWS region to `us-west-2`
}

variable "cluster-name" { # Declares a variable named `cluster-name`
  default = "terraform-eks-demo" # Sets the default name for the EKS cluster to `terraform-eks-demo`
  type    = string # Explicitly defines the type of the variable as a string
}
```

This code defines two variables to make the Terraform configuration flexible and reusable:

1. **`aws_region`**: Sets the AWS region for the infrastructure deployment. The default region is `us-west-2` (Oregon), but it can be overridden as needed.
2. **`cluster-name`**: Specifies the name of the EKS cluster to be deployed. The default value is `terraform-eks-demo`, but users can customize it.

By using these variables, the Terraform code becomes dynamic, allowing changes to key configurations (region and cluster name) without modifying the main code. This approach promotes better maintainability and reuse of the Terraform scripts.

workstaton-external-ip.tf

```
#
# Workstation External IP
#
# This configuration is not required and is
# only provided as an example to easily fetch
# the external IP of your local workstation to
```

```
# configure inbound EC2 Security Group access
# to the Kubernetes cluster.
#

data "http" "workstation-external-ip" {  # Defines a data source to make an HTTP GET request
  url = "http://ipv4.icanhazip.com"  # The URL returns the external IP address of the caller
}

# Override with variable or hardcoded value if necessary
locals {
  workstation-external-cidr = "${chomp(data.http.workstation-external-ip.body)}/32"
}
```

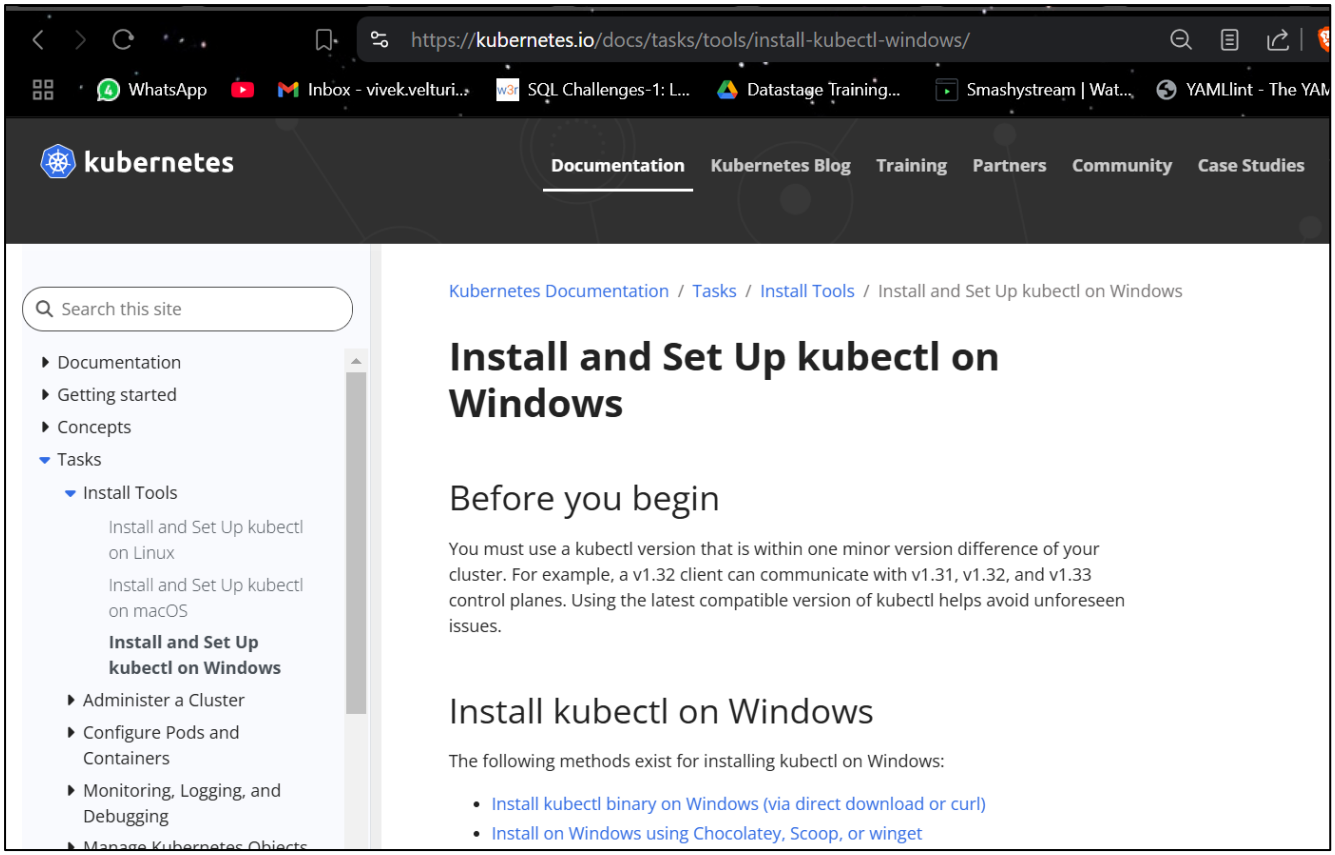
This code dynamically fetches the external IP address of your local workstation using the public API `http://ipv4.icanhazip.com` and formats it in CIDR notation (`<IP>/32`). The local variable `workstation-external-cidr` can be used elsewhere in your Terraform configuration, such as defining inbound Security Group rules to allow access only from your workstation.
Key points:

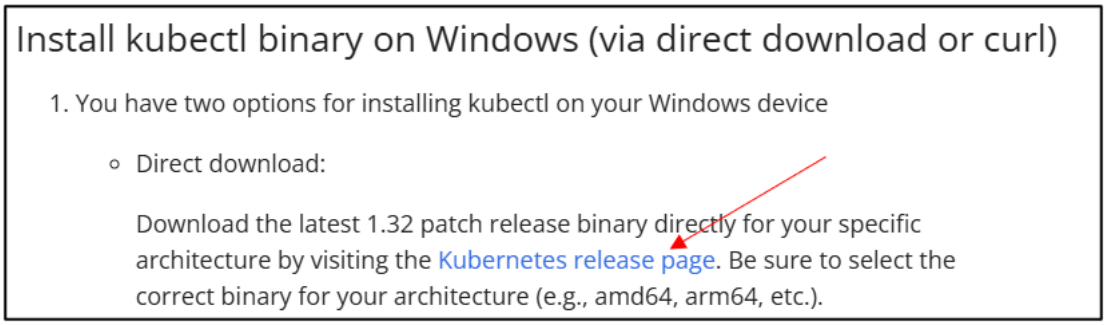
1. **`data.http.workstation-external-ip`**: Makes an HTTP request to fetch the workstation's public IP.
2. **`local.workstation-external-cidr`**: Formats the IP with `/32` to define a network allowing access for only that single IP.

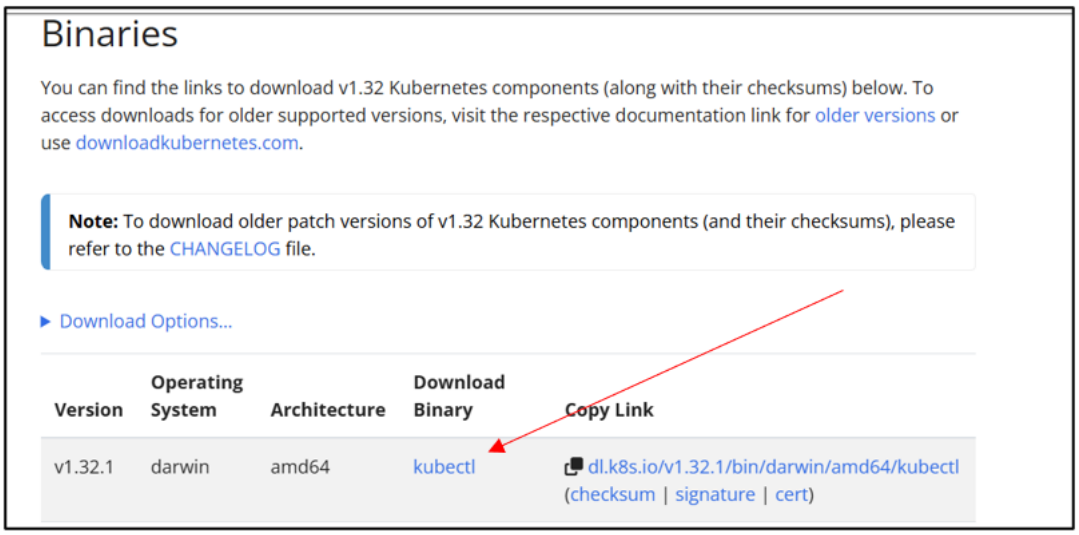
This configuration is optional and useful for securely restricting access to resources from your local machine.

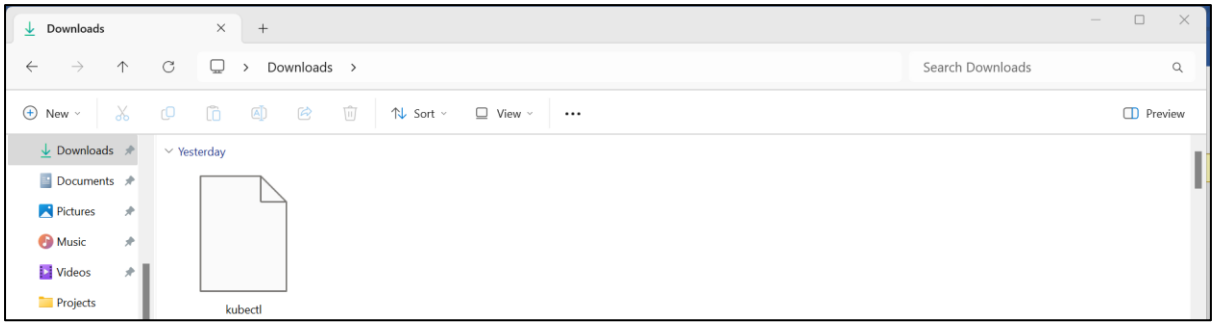
Before we can start working with the EKS cluster created using the Terraform script, we need a tool called kubectl installed on our local computer. Kubectl is a command-line utility that helps us manage and interact with Kubernetes clusters, like the EKS cluster. Since your computer runs on Windows, you'll first need to install kubectl on it. Once installed, you can use it to run commands and communicate with the cluster easily.

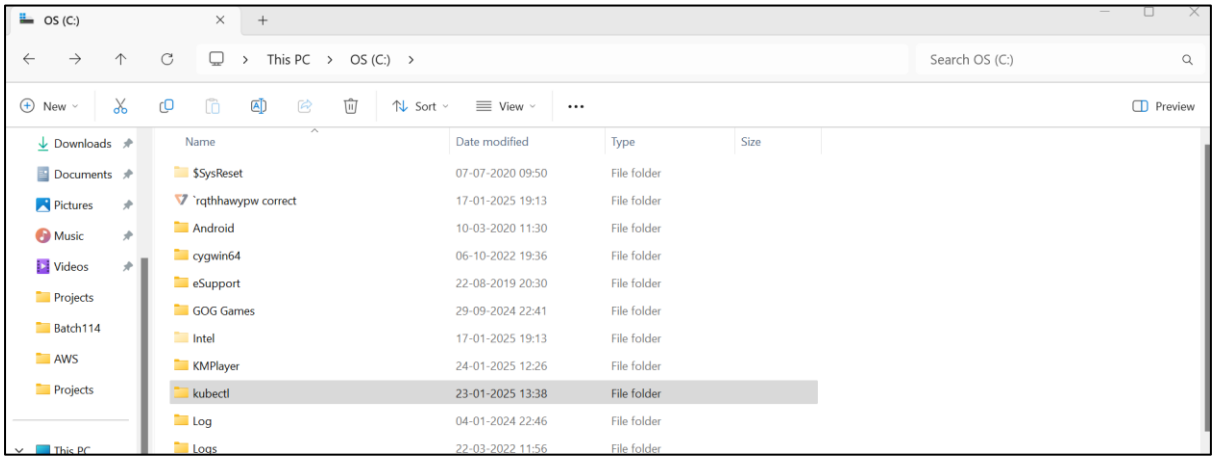
- To install kubectl on a Windows system, visit the official Kubernetes website at: **https://kubernetes.io/docs/tasks/tools/install-kubectl-windows/**
- Under the "Install kubectl on Windows" section, click on "Install kubectl binary on Windows (via direct download or curl)".
- Select the latest version of kubectl for Windows (amd64) and click to download it.
- After downloading the kubectl binary, locate the file in the Downloads folder on your local computer. Copy the file, navigate to the C: Drive, and create a new folder named "kubectl." Paste the copied file into this folder.
- Now go to **Computer Properties**, then click on **Advanced system settings**. In the System Properties window, select the **Environment Variables** button. In the Environment Variables window, find the **Path** variable under System variables, **click Edit, and add C:\kubectl** to the list.
- Now, we have successfully configured kubectl on our local Windows OS computer.

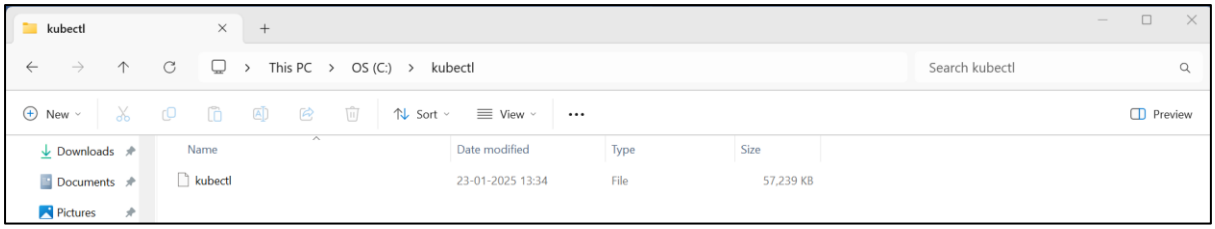

https://kubernetes.io/docs/tasks/tools/install-kubectl-windows/
WhatsApp
Inbox - vivek.velturi...
SQL Challenges-1: L...
Datastage Training...
Smashystream | Wat...
YAMLlint - The YAM
kubernetes
Documentation
Kubernetes Blog
Training
Partners
Community
Case Studies
Search this site
Documentation
Getting started
Concepts
Tasks
Install Tools
Install and Set Up kubectl on Linux
Install and Set Up kubectl on macOS
Install and Set Up kubectl on Windows
Administer a Cluster
Configure Pods and Containers
Monitoring, Logging, and Debugging
Kubernetes Documentation / Tasks / Install Tools / Install and Set Up kubectl on Windows
Install and Set Up kubectl on Windows
Before you begin
You must use a kubectl version that is within one minor version difference of your cluster. For example, a v1.32 client can communicate with v1.31, v1.32, and v1.33 control planes. Using the latest compatible version of kubectl helps avoid unforeseen issues.
Install kubectl on Windows
The following methods exist for installing kubectl on Windows:
Install kubectl binary on Windows (via direct download or curl)
Install on Windows using Chocolatey, Scoop, or winget

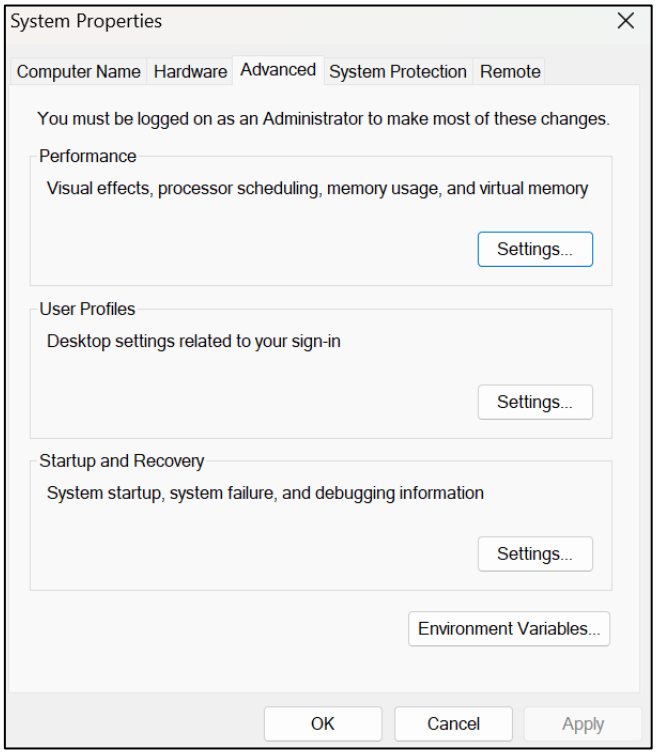

Install kubectl binary on Windows (via direct download or curl)
1. You have two options for installing kubectl on your Windows device
Direct download:
Download the latest 1.32 patch release binary directly for your specific architecture by visiting the Kubernetes release page. Be sure to select the correct binary for your architecture (e.g., amd64, arm64, etc.).

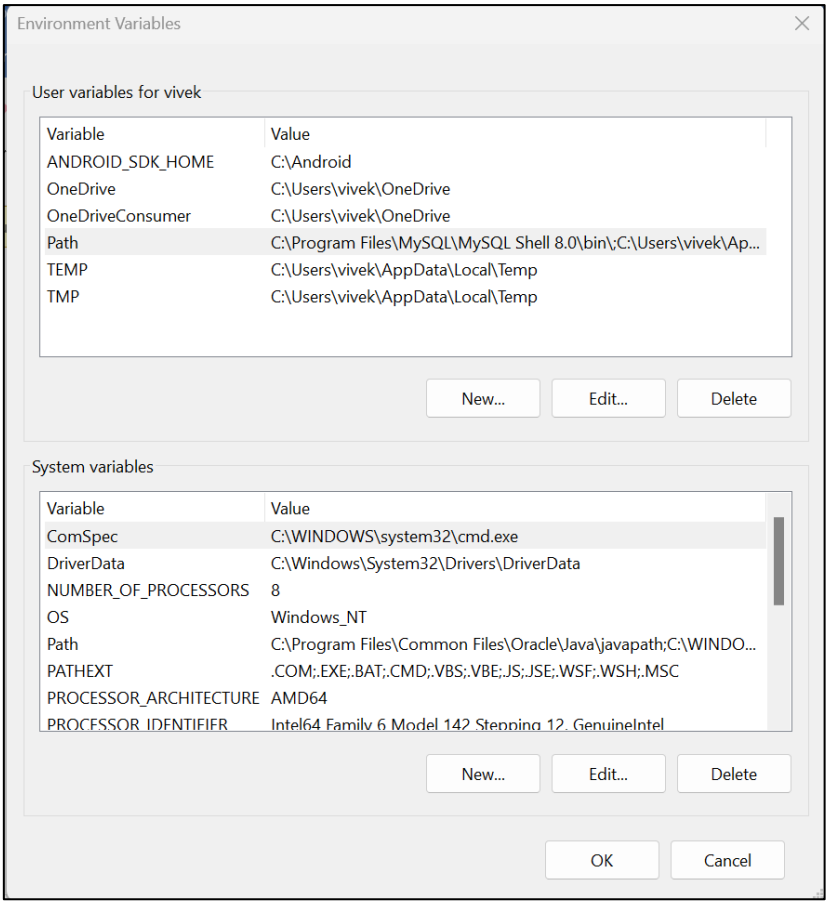

Binaries
You can find the links to download v1.32 Kubernetes components (along with their checksums) below. To access downloads for older supported versions, visit the respective documentation link for older versions or use downloadkubernetes.com.
Note: To download older patch versions of v1.32 Kubernetes components (and their checksums), please refer to the CHANGELOG file.
Download Options...
Version
Operating System
Architecture
Download Binary
Copy Link
v1.32.1
darwin
amd64
kubectl
dl.k8s.io/v1.32.1/bin/darwin/amd64/kubectl
(checksum | signature | cert)

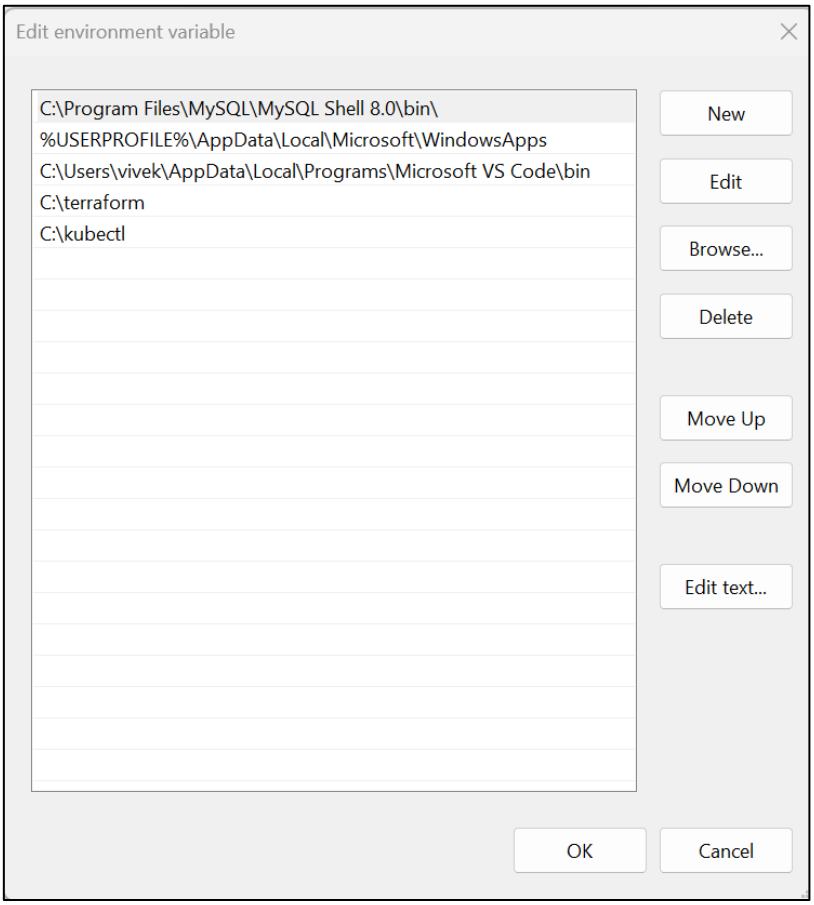

Downloads
Search Downloads
New
Sort
View
Preview
Yesterday
kubectl


OS (C:)
This PC
OS (C:)
Search OS (C:)
Name
Date modified
Type
Size
$SysReset
Android
cygwin64
eSupport
GOG Games
Intel
KMPlayer
kubectl
Log
Logs
Batch114
AWS
Projects

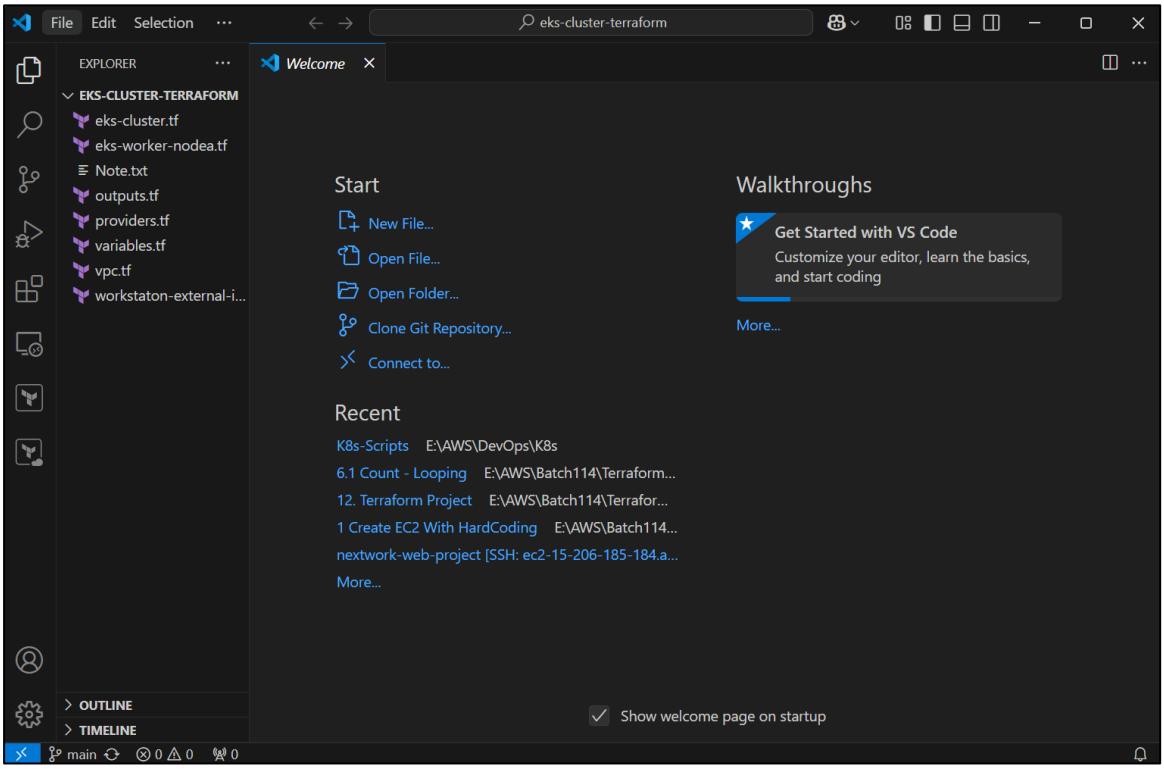

kubectl
This PC
OS (C:)
kubectl
Search kubectl
Name
Date modified
Type
Size
kubectl
23-01-2025 13:34
File
57,239 KB


System Properties
Computer Name
Hardware
Advanced
System Protection
Remote
You must be logged on as an Administrator to make most of these changes.
Performance
Visual effects, processor scheduling, memory usage, and virtual memory
Settings...
User Profiles
Desktop settings related to your sign-in
Settings...
Startup and Recovery
System startup, system failure, and debugging information
Settings...
Environment Variables...
OK
Cancel
Apply

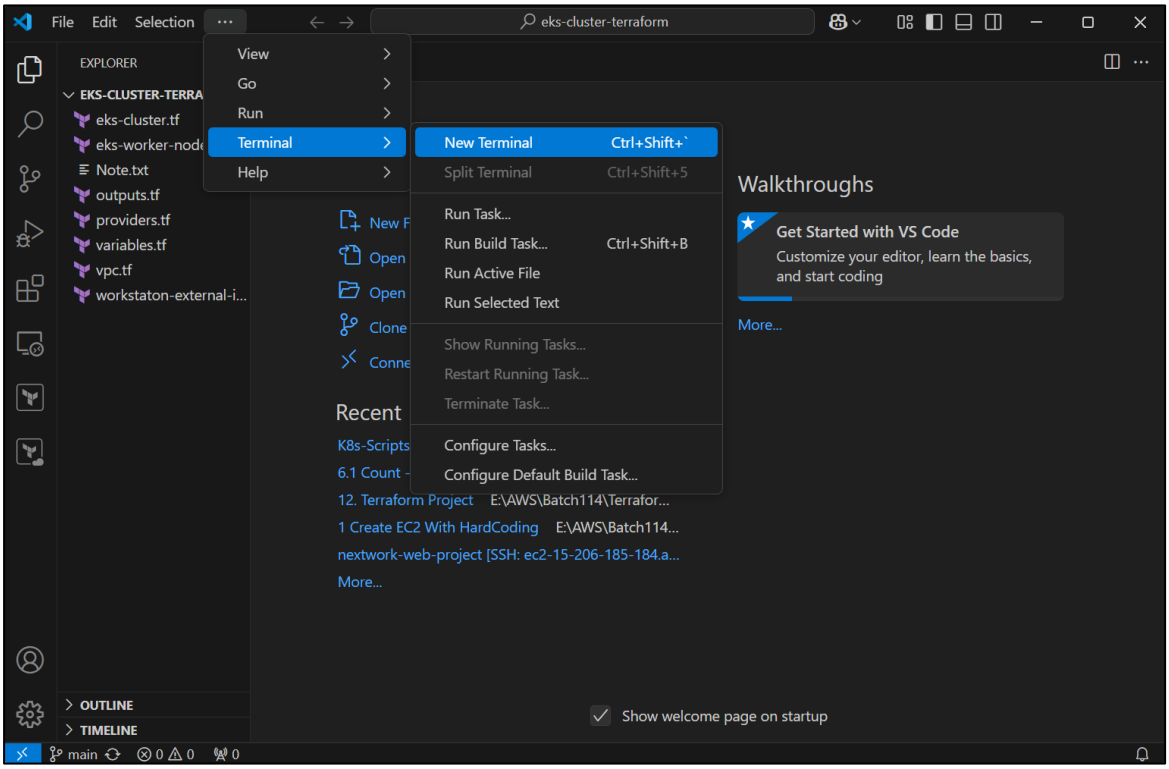

Environment Variables
User variables for vivek
Variable
Value
ANDROID_SDK_HOME
C:\Android
OneDrive
C:\Users\vivek\OneDrive
OneDriveConsumer
C:\Users\vivek\OneDrive
Path
C:\Program Files\MySQL\MySQL Shell 8.0\bin\;C:\Users\vivek\Ap...
TEMP
C:\Users\vivek\AppData\Local\Temp
TMP
C:\Users\vivek\AppData\Local\Temp
New...
Edit...
Delete
System variables
Variable
Value
ComSpec
C:\WINDOWS\system32\cmd.exe
DriverData
C:\Windows\System32\Drivers\DriverData
NUMBER_OF_PROCESSORS
8
OS
Windows_NT
Path
C:\Program Files\Common Files\Oracle\Java\javapath;C:\WINDO...
PATHEXT
.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE
AMD64
PROCESSOR_IDENTIFIER
Intel64 Family 6 Model 142 Stepping 12, GenuineIntel
New...
Edit...
Delete
OK
Cancel


Edit environment variable
C:\Program Files\MySQL\MySQL Shell 8.0\bin\
%USERPROFILE%\AppData\Local\Microsoft\WindowsApps
C:\Users\vivek\AppData\Local\Programs\Microsoft VS Code\bin
C:\terraform
C:\kubectl
New
Edit
Browse...
Delete
Move Up
Move Down
Edit text...
OK
Cancel

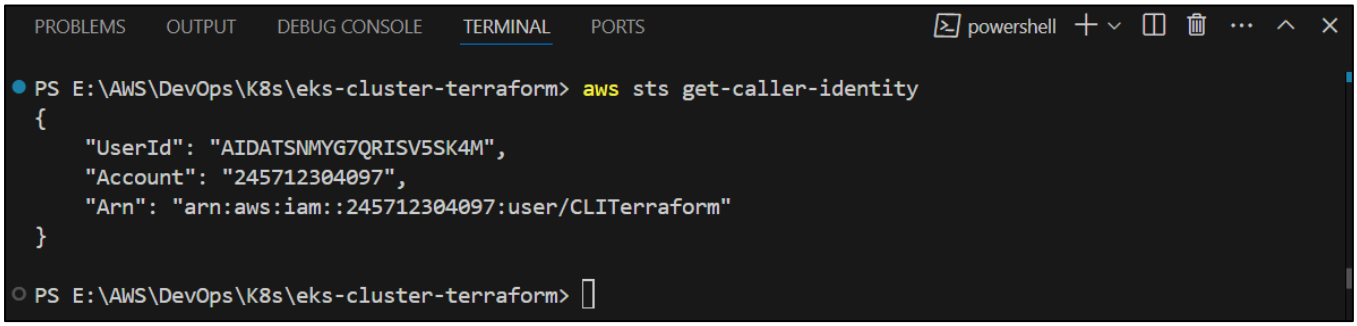
- Now that kubectl is set up on our computer, let's open Visual Studio Code and go to the folder where we cloned the Terraform scripts from GitHub.



- Open a new terminal.



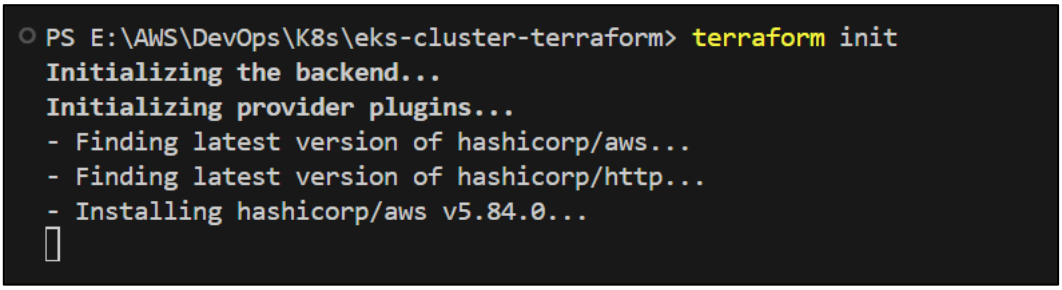
- To check if Visual Studio is connected to our AWS account, run the command: `aws sts get-caller-identity`.

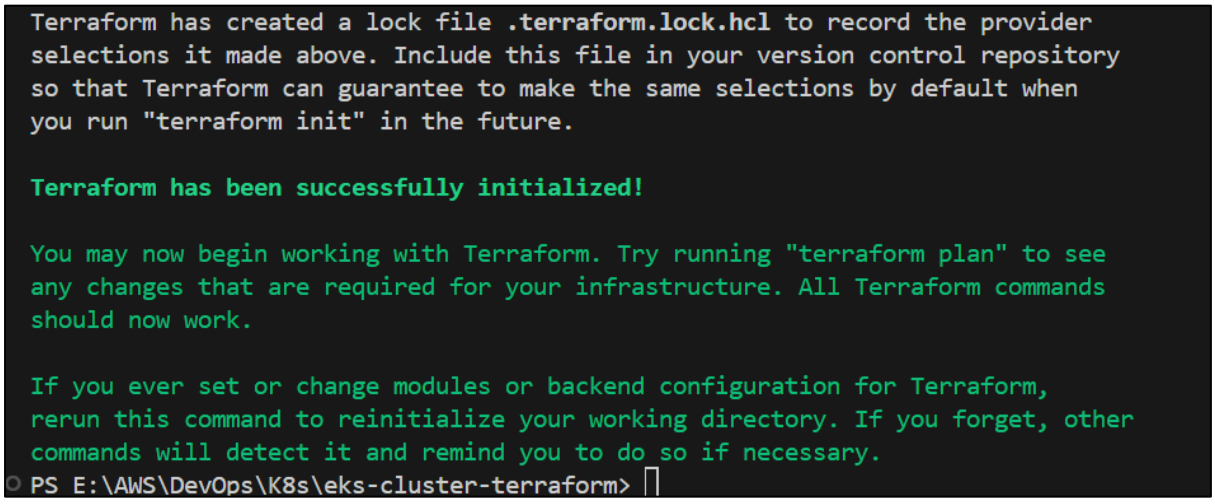


Note: If Visual Studio is not connected to AWS, follow these steps:

Verify AWS CLI Configuration: Ensure the AWS CLI is installed and configured properly. You can configure it by running the command `aws configure` in your command prompt or terminal, and then **enter**ing **your AWS Access Key, Secret Key, region, and output format.**

- Once your AWS is configured, initialize the Terraform script by executing the command: `terraform init`. This will set up the necessary dependencies and prepare the environment for running Terraform.

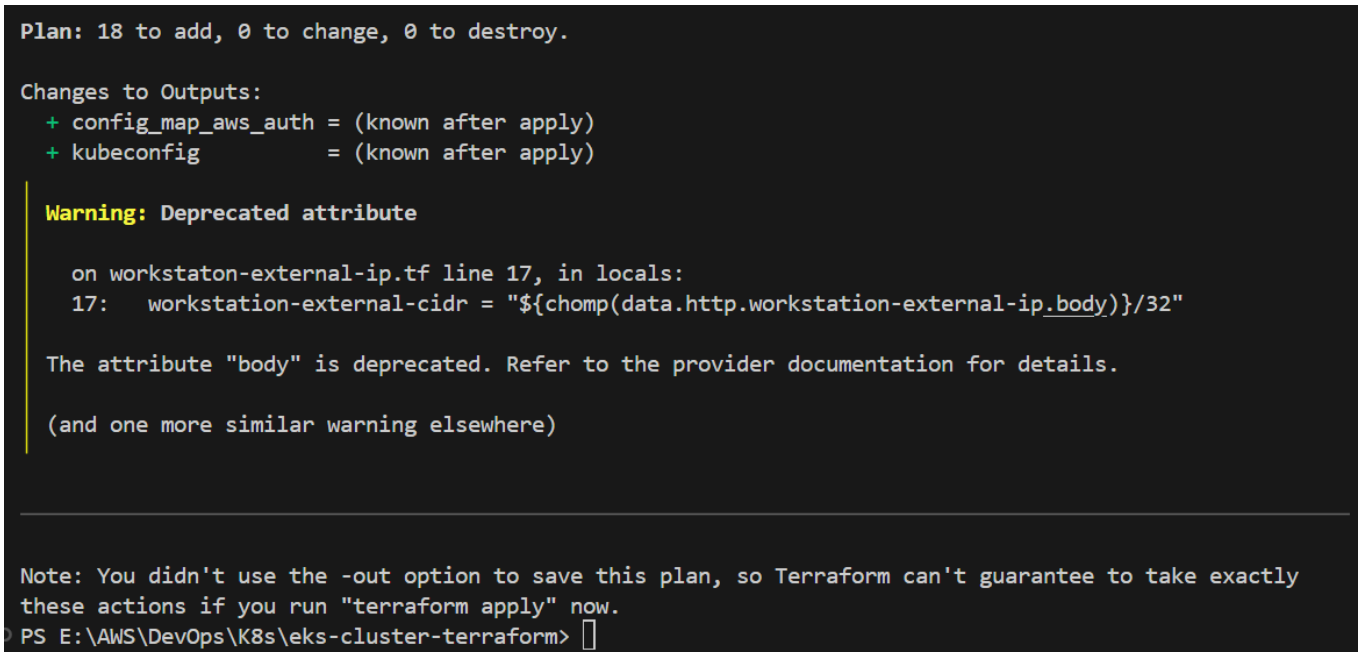




- After running `terraform init`, Terraform creates a lock file named **.terraform.lock.hcl** to record the provider selections it made. This file should be included in your version control system to ensure consistent provider versions are used in future runs. Terraform initialization is now complete, and you can begin working

with your infrastructure. If you modify modules or backend configurations, you should rerun `terraform init` to reinitialize your working directory.

- Now Execute `terraform plan` command. The `terraform plan` command is used to create an execution plan for your infrastructure. It compares the current state of your infrastructure (as defined in your configuration files) with the existing state in your cloud provider (like AWS). The command shows what actions Terraform will take to align the infrastructure with your configuration, such as creating, modifying, or deleting resources. It doesn't make any changes to your infrastructure; it simply provides a preview of what will happen when you run `terraform apply`.



The `terraform plan` output **shows that Terraform will create a total of 18 resources** in your AWS environment, including an **EKS cluster** (`aws_eks_cluster.demo`), **node group** (`aws_eks_node_group.demo`), **IAM roles, security groups, a VPC with subnets, route tables, and an internet gateway.** It also lists 0 resources to be modified or destroyed. The plan outlines the exact resources Terraform will add, along with any associated configurations, such as IAM role policy attachments and subnet associations. There are warnings about deprecated attributes, indicating that some of the configuration elements are outdated and may need to be updated in the future. This plan provides a detailed preview of what will be created when you apply the configuration, ensuring you can review the changes before proceeding.

- Next, we need to use the `terraform apply` command to create the resources outlined in the `terraform plan` output.

```
data.http.workstation-external-ip: Reading...
data.http.workstation-external-ip: Read complete after 0s [id=http://ipv4.icanhazip.com]
data.aws_availability_zones.available: Reading...
data.aws_availability_zones.available: Read complete after 1s [id=us-west-2]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_eks_cluster.demo will be created
  + resource "aws_eks_cluster" "demo" {
      + arn                           = (known after apply)
      + bootstrap_self_managed_addons = true
      + certificate_authority         = (known after apply)
      + cluster_id                    = (known after apply)
      + created_at                    = (known after apply)
      + endpoint                      = (known after apply)
      + id                            = (known after apply)
      + identity                      = (known after apply)
      + name                          = "terraform-eks-demo"
      + platform_version              = (known after apply)
      + role_arn                      = (known after apply)
```

```
      + role_arn                      = (known after apply)
      + status                        = (known after apply)
      + tags_all                      = (known after apply)
      + version                       = (known after apply)

      + access_config (known after apply)

      + kubernetes_network_config (known after apply)

      + upgrade_policy (known after apply)

      + vpc_config {
          + cluster_security_group_id = (known after apply)
          + endpoint_private_access   = false
          + endpoint_public_access    = true
          + public_access_cidrs       = (known after apply)
          + security_group_ids        = (known after apply)
          + subnet_ids                = (known after apply)
          + vpc_id                    = (known after apply)
        }
    }

  # aws_eks_node_group.demo will be created
  + resource "aws_eks_node_group" "demo" {
```

```
  # aws_eks_node_group.demo will be created
  + resource "aws_eks_node_group" "demo" {
      + ami_type               = (known after apply)
      + arn                    = (known after apply)
      + capacity_type          = (known after apply)
      + cluster_name           = "terraform-eks-demo"
      + disk_size              = (known after apply)
      + id                     = (known after apply)
      + instance_types         = (known after apply)
      + node_group_name        = "demo"
      + node_group_name_prefix = (known after apply)
      + node_role_arn          = (known after apply)
      + release_version        = (known after apply)
      + resources              = (known after apply)
      + status                 = (known after apply)
      + subnet_ids             = (known after apply)
      + tags_all               = (known after apply)
      + version                = (known after apply)

      + node_repair_config (known after apply)

      + scaling_config {
```

```
      + scaling_config {
          + desired_size = 1
          + max_size     = 1
          + min_size     = 1
        }

      + update_config (known after apply)
    }

  # aws_iam_role.demo-cluster will be created
  + resource "aws_iam_role" "demo-cluster" {
      + arn                   = (known after apply)
      + assume_role_policy    = jsonencode(
            {
              + Statement = [
                  + {
                      + Action    = "sts:AssumeRole"
                      + Effect    = "Allow"
                      + Principal = {
                          + Service = "eks.amazonaws.com"
                        }
                    },
                ]
              + Version   = "2012-10-17"
            }
        )
      + create_date           = (known after apply)
      + force_detach_policies = false
      + id                    = (known after apply)
      + managed_policy_arns   = (known after apply)
      + max_session_duration  = 3600
      + name                  = "terraform-eks-demo-cluster"
      + name_prefix           = (known after apply)
      + path                  = "/"
      + tags_all              = (known after apply)
      + unique_id             = (known after apply)

      + inline_policy (known after apply)
    }

  # aws_iam_role.demo-node will be created
  + resource "aws_iam_role" "demo-node" {
      + arn                   = (known after apply)
      + assume_role_policy    = jsonencode(
            {
              + Statement = [
                  + {
                      + Action    = "sts:AssumeRole"
                      + Effect    = "Allow"
                      + Principal = {
                          + Service = "ec2.amazonaws.com"
                        }
                    },
                ]
              + Version   = "2012-10-17"
            }
        )
      + create_date           = (known after apply)
      + force_detach_policies = false
      + id                    = (known after apply)
      + managed_policy_arns   = (known after apply)
      + max_session_duration  = 3600
      + name                  = "terraform-eks-demo-node"
      + name_prefix           = (known after apply)
      + path                  = "/"
      + tags_all              = (known after apply)
      + unique_id             = (known after apply)

      + inline_policy (known after apply)
    }
```

```
  }

# aws_iam_role_policy_attachment.demo-cluster-AmazonEKSClusterPolicy will be created
+ resource "aws_iam_role_policy_attachment" "demo-cluster-AmazonEKSClusterPolicy" {
    + id         = (known after apply)
    + policy_arn = "arn:aws:iam::aws:policy/AmazonEKSClusterPolicy"
    + role       = "terraform-eks-demo-cluster"
  }

# aws_iam_role_policy_attachment.demo-cluster-AmazonEKSVPCResourceController will be created
+ resource "aws_iam_role_policy_attachment" "demo-cluster-AmazonEKSVPCResourceController" {
    + id         = (known after apply)
    + policy_arn = "arn:aws:iam::aws:policy/AmazonEKSVPCResourceController"
    + role       = "terraform-eks-demo-cluster"
  }

# aws_iam_role_policy_attachment.demo-node-AmazonEC2ContainerRegistryReadOnly will be created
+ resource "aws_iam_role_policy_attachment" "demo-node-AmazonEC2ContainerRegistryReadOnly" {
    + id         = (known after apply)
    + policy_arn = "arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly"
    + role       = "terraform-eks-demo-node"
  }

# aws_iam_role_policy_attachment.demo-node-AmazonEKSWorkerNodePolicy will be created
```

```
  # aws_iam_role_policy_attachment.demo-node-AmazonEKSWorkerNodePolicy will be created
  + resource "aws_iam_role_policy_attachment" "demo-node-AmazonEKSWorkerNodePolicy" {
      + id         = (known after apply)
      + policy_arn = "arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy"
      + role       = "terraform-eks-demo-node"
    }

  # aws_iam_role_policy_attachment.demo-node-AmazonEKS_CNI_Policy will be created
  + resource "aws_iam_role_policy_attachment" "demo-node-AmazonEKS_CNI_Policy" {
      + id         = (known after apply)
      + policy_arn = "arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy"
      + role       = "terraform-eks-demo-node"
    }

  # aws_internet_gateway.demo will be created
  + resource "aws_internet_gateway" "demo" {
      + arn      = (known after apply)
      + id       = (known after apply)
      + owner_id = (known after apply)
      + tags     = {
          + "Name" = "terraform-eks-demo"
        }
      + tags_all = {
          + "Name" = "terraform-eks-demo"
```

```
          + "Name" = "terraform-eks-demo"
        }
      + vpc_id   = (known after apply)
    }

  # aws_route_table.demo will be created
  + resource "aws_route_table" "demo" {
      + arn              = (known after apply)
      + id               = (known after apply)
      + owner_id         = (known after apply)
      + propagating_vgws = (known after apply)
      + route            = [
          + {
              + cidr_block                = "0.0.0.0/0"
              + gateway_id                = (known after apply)
                # (11 unchanged attributes hidden)
            },
        ]
      + tags_all         = (known after apply)
      + vpc_id           = (known after apply)
    }

  # aws_route_table_association.demo[0] will be created
  + resource "aws_route_table_association" "demo" {
```

```
  + resource "aws_route_table_association" "demo" {
      + id             = (known after apply)
      + route_table_id = (known after apply)
      + subnet_id      = (known after apply)
    }

  # aws_route_table_association.demo[1] will be created
  + resource "aws_route_table_association" "demo" {
      + id             = (known after apply)
      + route_table_id = (known after apply)
      + subnet_id      = (known after apply)
    }

  # aws_security_group.demo-cluster will be created
  + resource "aws_security_group" "demo-cluster" {
      + arn                    = (known after apply)
      + description            = "Cluster communication with worker nodes"
      + egress                 = [
          + {
              + cidr_blocks      = [
                  + "0.0.0.0/0",
                ]
              + from_port        = 0
              + ipv6_cidr_blocks = []
```

```
              + ipv6_cidr_blocks = []
              + prefix_list_ids  = []
              + protocol         = "-1"
              + security_groups  = []
              + self             = false
              + to_port          = 0
                # (1 unchanged attribute hidden)
            },
        ]
      + id                     = (known after apply)
      + ingress                = (known after apply)
      + name                   = "terraform-eks-demo-cluster"
      + name_prefix            = (known after apply)
      + owner_id               = (known after apply)
      + revoke_rules_on_delete = false
      + tags                   = {
          + "Name" = "terraform-eks-demo"
        }
      + tags_all               = {
          + "Name" = "terraform-eks-demo"
        }
      + vpc_id                 = (known after apply)
    }
```

```
      + vpc_id                 = (known after apply)
    }

  # aws_security_group_rule.demo-cluster-ingress-workstation-https will be created
  + resource "aws_security_group_rule" "demo-cluster-ingress-workstation-https" {
      + cidr_blocks              = [
          + "49.204.238.212/32",
        ]
      + description              = "Allow workstation to communicate with the cluster API Server"
      + from_port                = 443
      + id                       = (known after apply)
      + protocol                 = "tcp"
      + security_group_id        = (known after apply)
      + security_group_rule_id   = (known after apply)
      + self                     = false
      + source_security_group_id = (known after apply)
      + to_port                  = 443
      + type                     = "ingress"
    }

  # aws_subnet.demo[0] will be created
  + resource "aws_subnet" "demo" {
      + arn                                            = (known after apply)
      + assign_ipv6_address_on_creation                = false
```

```
    + assign_ipv6_address_on_creation               = false
    + availability_zone                             = "us-west-2a"
    + availability_zone_id                          = (known after apply)
    + cidr_block                                    = "10.0.0.0/24"
    + enable_dns64                                  = false
    + enable_resource_name_dns_a_record_on_launch   = false
    + enable_resource_name_dns_aaaa_record_on_launch = false
    + id                                            = (known after apply)
    + ipv6_cidr_block_association_id                = (known after apply)
    + ipv6_native                                   = false
    + map_public_ip_on_launch                       = true
    + owner_id                                      = (known after apply)
    + private_dns_hostname_type_on_launch           = (known after apply)
    + tags                                          = {
        + "Name"                                    = "terraform-eks-demo-node"
        + "kubernetes.io/cluster/terraform-eks-demo" = "shared"
      }
    + tags_all                                      = {
        + "Name"                                    = "terraform-eks-demo-node"
        + "kubernetes.io/cluster/terraform-eks-demo" = "shared"
      }
    + vpc_id                                        = (known after apply)
  }
```

```
# aws_subnet.demo[1] will be created
+ resource "aws_subnet" "demo" {
    + arn                                           = (known after apply)
    + assign_ipv6_address_on_creation               = false
    + availability_zone                             = "us-west-2b"
    + availability_zone_id                          = (known after apply)
    + cidr_block                                    = "10.0.1.0/24"
    + enable_dns64                                  = false
    + enable_resource_name_dns_a_record_on_launch   = false
    + enable_resource_name_dns_aaaa_record_on_launch = false
    + id                                            = (known after apply)
    + ipv6_cidr_block_association_id                = (known after apply)
    + ipv6_native                                   = false
    + map_public_ip_on_launch                       = true
    + owner_id                                      = (known after apply)
    + private_dns_hostname_type_on_launch           = (known after apply)
    + tags                                          = {
        + "Name"                                    = "terraform-eks-demo-node"
        + "kubernetes.io/cluster/terraform-eks-demo" = "shared"
      }
    + tags_all                                      = {
        + "Name"                                    = "terraform-eks-demo-node"
        + "kubernetes.io/cluster/terraform-eks-demo" = "shared"
```

```
        + "kubernetes.io/cluster/terraform-eks-demo" = "shared"
      }
    + vpc_id                                        = (known after apply)
  }

# aws_vpc.demo will be created
+ resource "aws_vpc" "demo" {
    + arn                                = (known after apply)
    + cidr_block                         = "10.0.0.0/16"
    + default_network_acl_id             = (known after apply)
    + default_route_table_id             = (known after apply)
    + default_security_group_id          = (known after apply)
    + dhcp_options_id                    = (known after apply)
    + enable_dns_hostnames               = (known after apply)
    + enable_dns_support                 = true
    + enable_network_address_usage_metrics = (known after apply)
    + id                                 = (known after apply)
    + instance_tenancy                   = "default"
    + ipv6_association_id                = (known after apply)
    + ipv6_cidr_block                    = (known after apply)
    + ipv6_cidr_block_network_border_group = (known after apply)
    + main_route_table_id                = (known after apply)
    + owner_id                           = (known after apply)
    + tags                               = {
```

```
      + ipv6_cidr_block_network_border_group = (known after apply)
      + main_route_table_id                  = (known after apply)
      + owner_id                             = (known after apply)
      + tags                                 = {
          + "Name"                                    = "terraform-eks-demo-node"
          + "kubernetes.io/cluster/terraform-eks-demo" = "shared"
        }
      + tags_all                             = {
          + "Name"                                    = "terraform-eks-demo-node"
          + "kubernetes.io/cluster/terraform-eks-demo" = "shared"
        }
    }

Plan: 18 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + config_map_aws_auth = (known after apply)
  + kubeconfig          = (known after apply)
```

```
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
```

- Enter yes to confirm the creation of cluster in your AWS.

```
aws_iam_role.demo-node: Creating...
aws_iam_role.demo-cluster: Creating...
aws_vpc.demo: Creating...
aws_iam_role.demo-cluster: Creation complete after 2s [id=terraform-eks-demo-cluster]
aws_iam_role_policy_attachment.demo-cluster-AmazonEKSVPCResourceController: Creating...
aws_iam_role_policy_attachment.demo-cluster-AmazonEKSClusterPolicy: Creating...
aws_iam_role.demo-node: Creation complete after 2s [id=terraform-eks-demo-node]
aws_iam_role_policy_attachment.demo-node-AmazonEC2ContainerRegistryReadOnly: Creating...
aws_iam_role_policy_attachment.demo-node-AmazonEKS_CNI_Policy: Creating...
aws_iam_role_policy_attachment.demo-node-AmazonEKSWorkerNodePolicy: Creating...
aws_iam_role_policy_attachment.demo-cluster-AmazonEKSClusterPolicy: Creation complete after 1s [id=terraform-eks-demo-cluster-20250124140048
196700000001]
aws_iam_role_policy_attachment.demo-node-AmazonEC2ContainerRegistryReadOnly: Creation complete after 1s [id=terraform-eks-demo-node-20250124
140048507900000003]
aws_iam_role_policy_attachment.demo-cluster-AmazonEKSVPCResourceController: Creation complete after 1s [id=terraform-eks-demo-cluster-202501
24140048333000000002]
aws_iam_role_policy_attachment.demo-node-AmazonEKS_CNI_Policy: Creation complete after 1s [id=terraform-eks-demo-node-2025012414004867220000
0004]
aws_iam_role_policy_attachment.demo-node-AmazonEKSWorkerNodePolicy: Creation complete after 1s [id=terraform-eks-demo-node-20250124140048819
100000005]
aws_vpc.demo: Creation complete after 5s [id=vpc-015116bd437a4b354]
aws_internet_gateway.demo: Creating...
aws_subnet.demo[0]: Creating...
aws_subnet.demo[1]: Creating...
```

```
aws_subnet.demo[1]: Creating...
aws_security_group.demo-cluster: Creating...
aws_internet_gateway.demo: Creation complete after 2s [id=igw-0231b873f9979a7a0]
aws_route_table.demo: Creating...
aws_route_table.demo: Creation complete after 3s [id=rtb-0001f16ac9418e37c]
aws_security_group.demo-cluster: Creation complete after 6s [id=sg-077344b207c384866]
aws_security_group_rule.demo-cluster-ingress-workstation-https: Creating...
aws_security_group_rule.demo-cluster-ingress-workstation-https: Creation complete after 1s [id=sgrule-827502948]
aws_subnet.demo[0]: Still creating... [10s elapsed]
aws_subnet.demo[1]: Still creating... [10s elapsed]
aws_subnet.demo[0]: Creation complete after 13s [id=subnet-00cef28410ec7ae21]
aws_subnet.demo[1]: Creation complete after 13s [id=subnet-02c1b14a21b7b6648]
aws_route_table_association.demo[0]: Creating...
aws_route_table_association.demo[1]: Creating...
aws_eks_cluster.demo: Creating...
aws_route_table_association.demo[1]: Creation complete after 1s [id=rtbassoc-0fbc31687274e8f0a]
aws_route_table_association.demo[0]: Creation complete after 1s [id=rtbassoc-0cd23c50cb05fde9d]
aws_eks_cluster.demo: Still creating... [10s elapsed]
aws_eks_cluster.demo: Still creating... [20s elapsed]
aws_eks_cluster.demo: Still creating... [30s elapsed]
aws_eks_cluster.demo: Still creating... [40s elapsed]
aws_eks_cluster.demo: Still creating... [50s elapsed]
aws_eks_cluster.demo: Still creating... [1m0s elapsed]
```

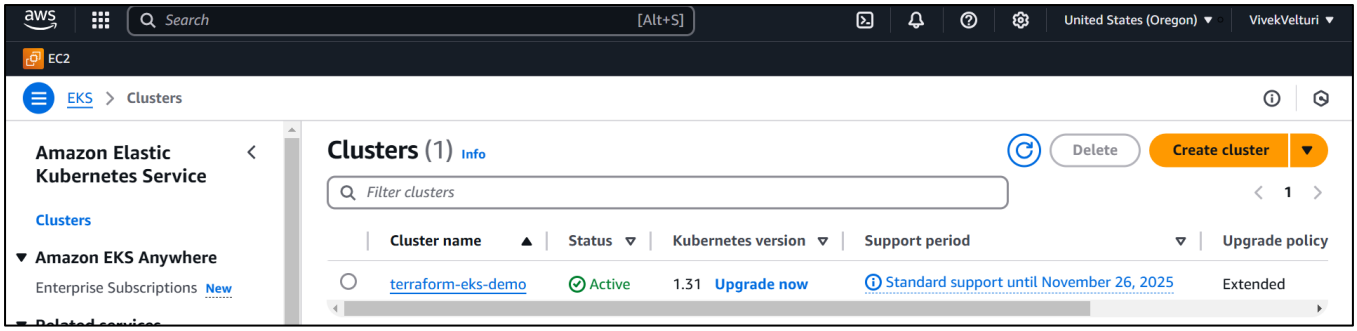
```
aws_eks_cluster.demo: Still creating... [7m50s elapsed]
aws_eks_cluster.demo: Still creating... [8m0s elapsed]
aws_eks_cluster.demo: Still creating... [8m10s elapsed]
aws_eks_cluster.demo: Still creating... [8m20s elapsed]
aws_eks_cluster.demo: Creation complete after 8m30s [id=terraform-eks-demo]
aws_eks_node_group.demo: Creating...
aws_eks_node_group.demo: Still creating... [10s elapsed]
aws_eks_node_group.demo: Still creating... [20s elapsed]
aws_eks_node_group.demo: Still creating... [30s elapsed]
aws_eks_node_group.demo: Still creating... [40s elapsed]
aws_eks_node_group.demo: Still creating... [50s elapsed]
aws_eks_node_group.demo: Still creating... [1m0s elapsed]
aws_eks_node_group.demo: Still creating... [1m10s elapsed]
aws_eks_node_group.demo: Still creating... [1m20s elapsed]
aws_eks_node_group.demo: Still creating... [1m30s elapsed]
aws_eks_node_group.demo: Still creating... [1m40s elapsed]
aws_eks_node_group.demo: Still creating... [1m50s elapsed]
aws_eks_node_group.demo: Creation complete after 1m52s [id=terraform-eks-demo:demo]
```

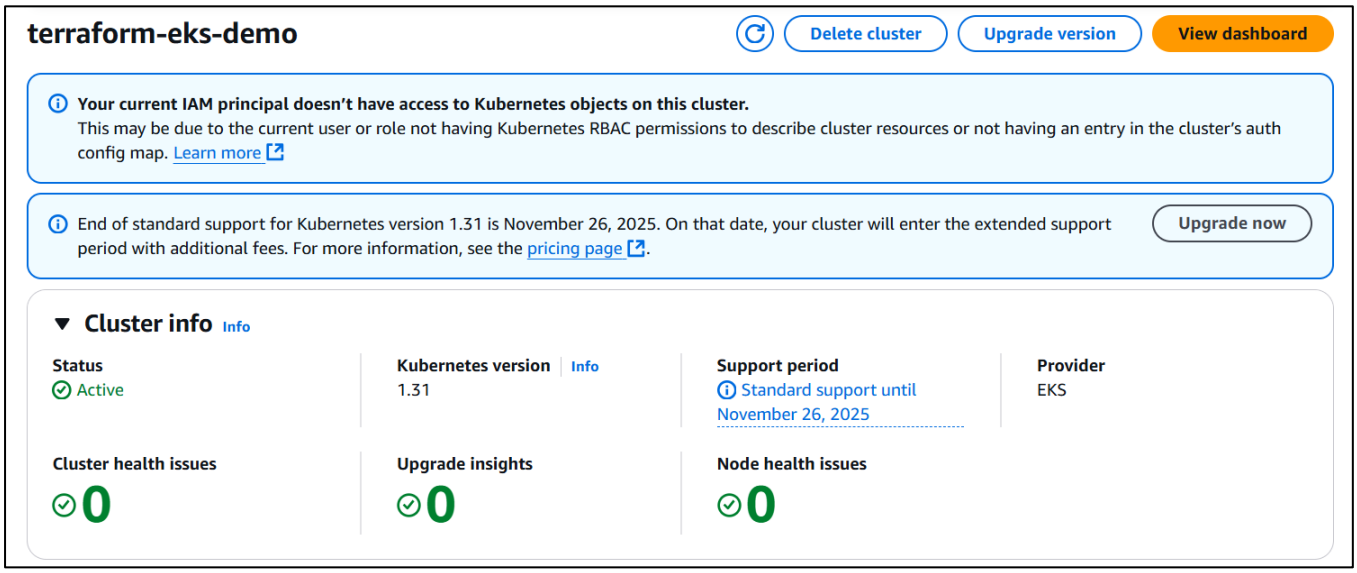
```
Apply complete! Resources: 18 added, 0 changed, 0 destroyed.

Outputs:

config_map_aws_auth = <<EOT


apiVersion: v1
kind: ConfigMap
metadata:
  name: aws-auth
  namespace: kube-system
data:
  mapRoles: |
    - rolearn: arn:aws:iam::245712304097:role/terraform-eks-demo-node
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes

EOT
kubeconfig = <<EOT
```

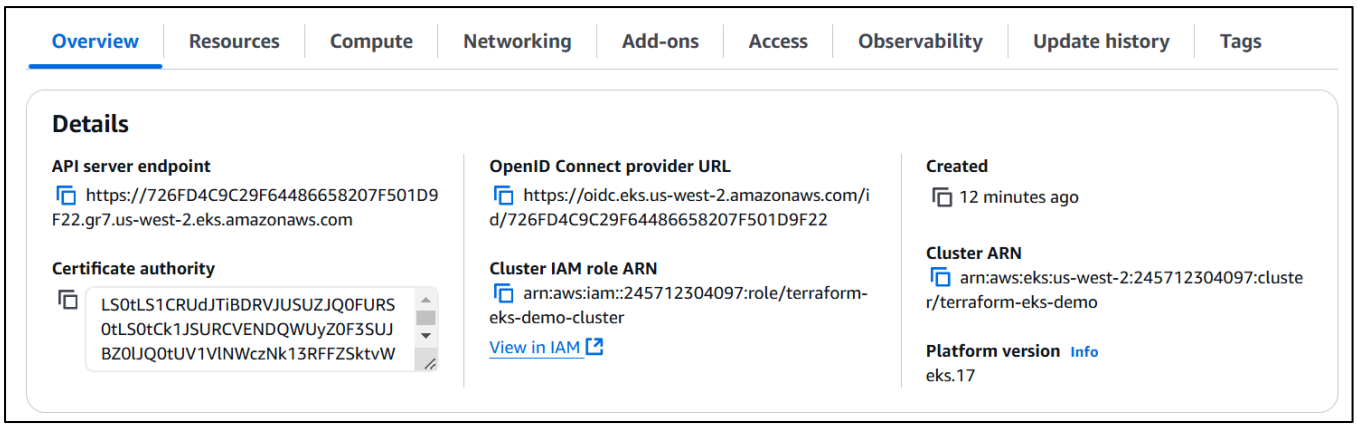
```
apiVersion: v1
clusters:
- cluster:
    server: https://726FD4C9C29F64486658207F501D9F22.gr7.us-west-2.eks.amazonaws.com
    certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURCVENDQWUyZ0F3SUJBZ0lJQ0tUV1VlNWczNk13RFFZSktvWklodmNOQVFFTEJRQXdG
VEVUTUJFR0ExVUUKQXhNS2EzVmlaWEp1WlhSbGN6QWVGdzB5TlRBeE1qUXhOREF4TURsYUZ3MHpOVEF4TWpJeE5EQTJNRGxhTUJVeApFekFSQmdOVkJBTVRDbXQxWW1WeWJtVjBaWE13
Z2dFaU1BMEdDU3FHU0liM0RRRUJBUVVBQTRJQkR3QXdnZ0VLCkFvSUJBUUROTDh2KzUxcVRPd25uQ0tnaGNQTks0WXI5cFZBbStRdUF4QnJLdS9uclBvWit0QnM4SVBFNkZ3ejUKcHU2
NlpTUDQwcjlCZC9RdlBVSGtVMGFSYnUzN2FIMFc3YStYTis0T0tEbUhHdW9hdHNBTy9qd1M2Z1dmVlhkQgo3WUJhRHpuelVLRkZVK2JrNUJnSmxHV2Q4QUhVTjRvSHZ3NTRIWG9FbFZy
dUliV21yM2dGRXZ1bkJmMW9FbVhHCklOTjFVYmJHSjlJZU0zb2NMd2QrbjYxa0dvSGNjNEFZREx0c2hVemhBZlNzcFhUSTlENHdyMXFOZzNWTEJYcUMKcUtiemhjNEZkYklSVlNsRHdR
WW1nU2xBOFFTY2ZrNTlxVE53SHpyZ2t5dndwWHhDSmNEbzNWdzBmb1lvT0xlcwpXVisrNW83cldMZGx5VmxPV2E0SDM3LzdnVks1QWdNQkFBR2pXVEJYTUE0R0ExVWREd0VCL3dRRUF3
SUNwREFQCkJnTlZIUk1CQWY4RUJUQURBUUgvTUIwR0ExVWREZ1FXQkJTT2trT2Nsb0NqVlh4RHRTTU42Qjgxbm05NEZEQVYKQmdOVkhSRUVEakFNZ2dwcmRXSmxjbTVsZEdWek1BMEdD
U3FHU0liM0RRRUJDd1VBQTRJQkFRQjg5N3J5d0dLUgplYngzZWM4c0FZeU1oWGp4eUhkYU5CeVNqQXFlMXB5VC9ZM1FmU3RjY08reEV2eGt4VEU5NEVCbDlpdERlTi9JCndmeVRVQ21O
TFZWcEJTSmRzWEdHUVFpdFRWakVrVUpVQjBRRGFzRFBjc2xlcDVHa0Y4eDBtTzJjbHFuUXlNcUsKOTJmWWlCOGgvck1DbUdhOXJSQlBNc0YycXBPMFNvbEQwakFlQ1FncEFseWdKZWhp
dTh4cURjOGhOQ2FVTjVLLwp3YXBCYnFKTndKUEkxR1ptT0NDTmE4NTdXcFBZVHdPdjNvZnhscU5CRkpUaHBhdFdFYUtDaEpWdFAvZGxtcXl2ClFINmlOODN5WGgwTWh1Sm9lMEtubDNx
RXhBeE5HYlRMQzM0am1sa1lEUGs4c3NLZy9WWU9ScW1uUXptSnA0WlAKY2FBK2J5cnlUcGRNCi0tLS0tRU5EIENFUlRJRklDQVRFLS0tLS0K
  name: kubernetes
contexts:
- context:
    cluster: kubernetes
    user: aws
  name: aws
current-context: aws
```

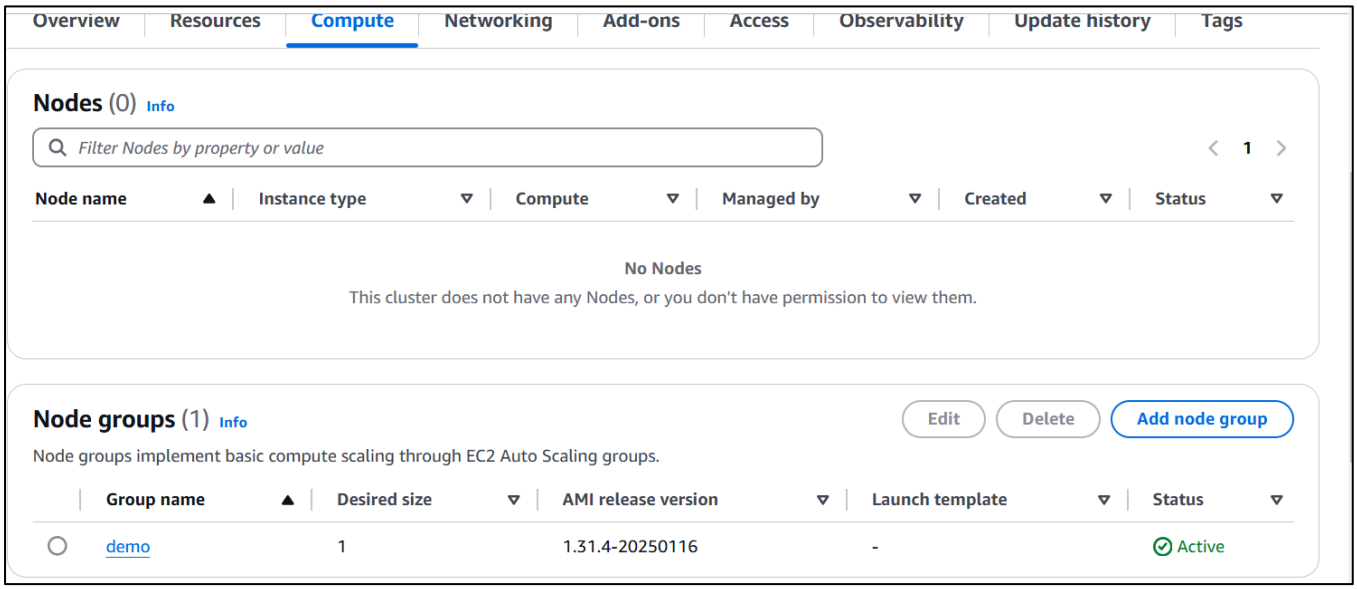
```
current-context: aws
kind: Config
preferences: {}
users:
- name: aws
  user:
    exec:
      apiVersion: client.authentication.k8s.io/v1beta1
      command: aws-iam-authenticator
      args:
        - "token"
        - "-i"
        - "terraform-eks-demo"

EOT
PS E:\AWS\DevOps\K8s\eks-cluster-terraform>
```

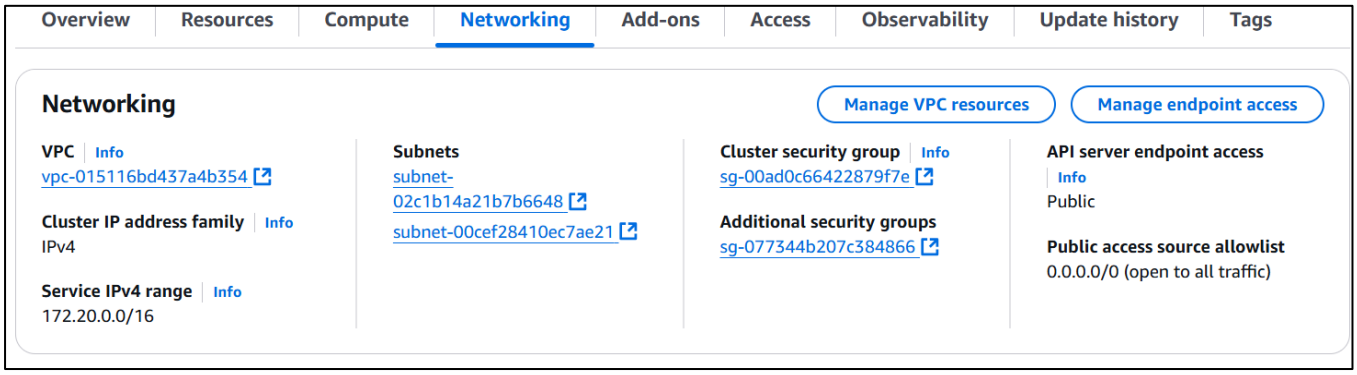
- After running the `terraform apply` command, log in to your AWS Management Console and navigate to the region where the resources were deployed using the Terraform script. Verify that an EKS cluster has been created and check the EC2 dashboard to confirm that an instance has been launched for the node.

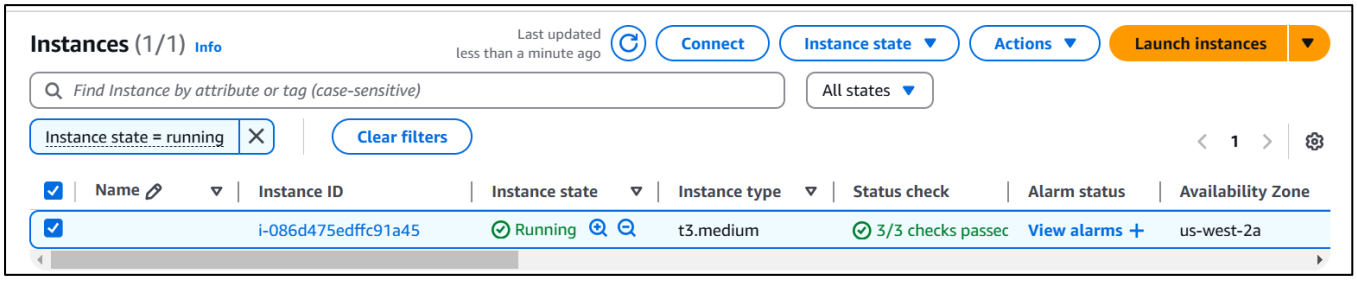


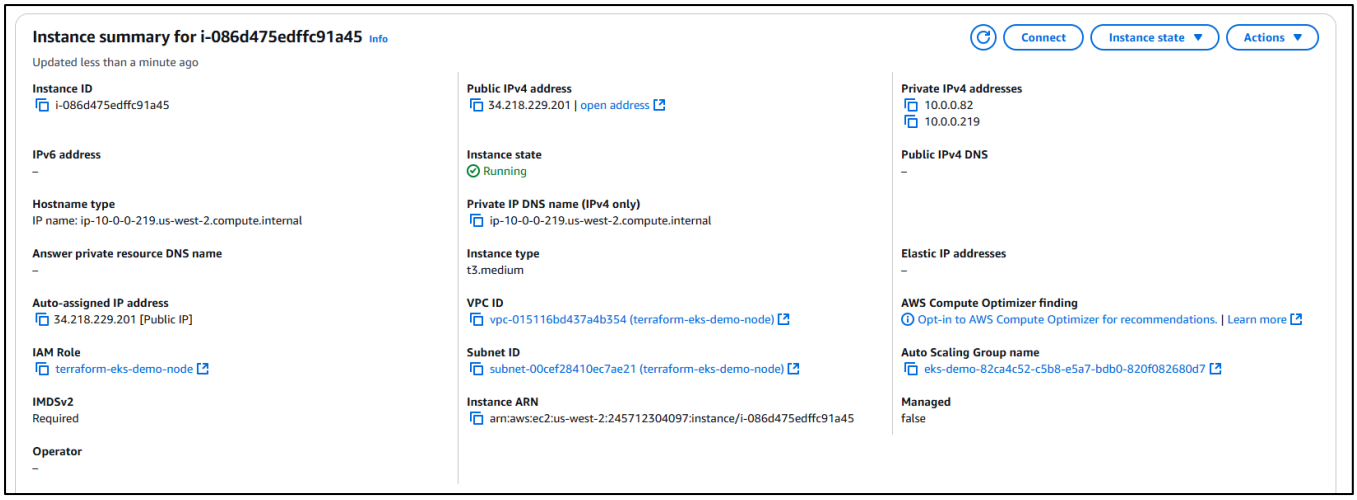


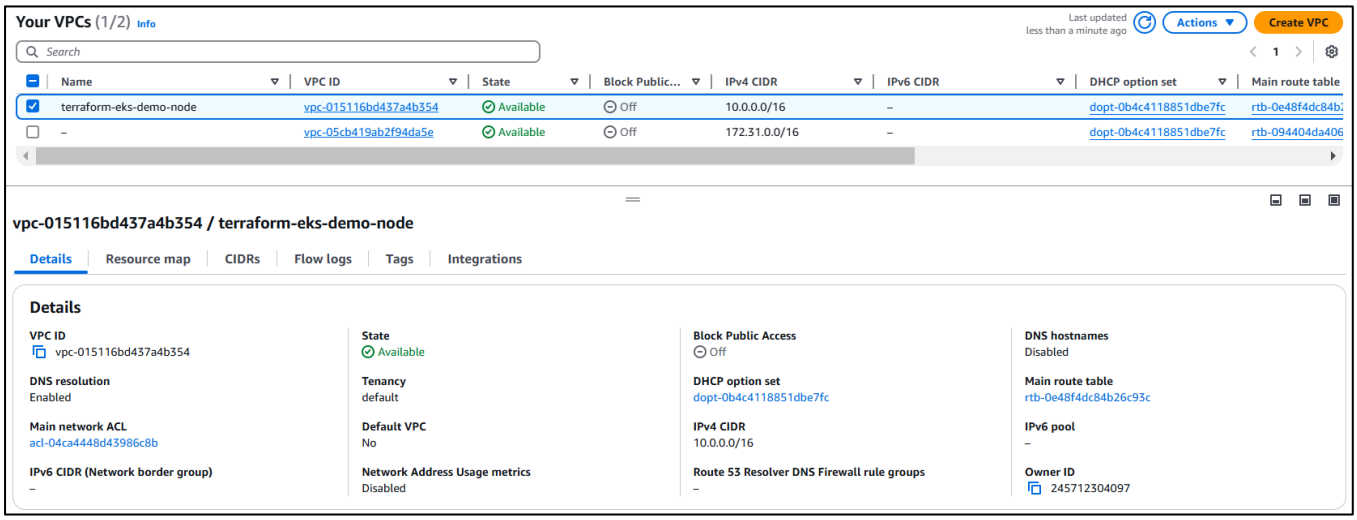

Overview
Resources
Compute
Networking
Add-ons
Access
Observability
Update history
Tags
Details
API server endpoint
https://726FD4C9C29F64486658207F501D9F22.gr7.us-west-2.eks.amazonaws.com
Certificate authority
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURCVENDQWUyZ0F3SUJBZ0lJQ0tUV1VlNWczNk13RFFZSktvW
OpenID Connect provider URL
https://oidc.eks.us-west-2.amazonaws.com/id/726FD4C9C29F64486658207F501D9F22
Cluster IAM role ARN
arn:aws:iam::245712304097:role/terraform-eks-demo-cluster
View in IAM
Created
12 minutes ago
Cluster ARN
arn:aws:eks:us-west-2:245712304097:cluster/terraform-eks-demo
Platform version Info
eks.17

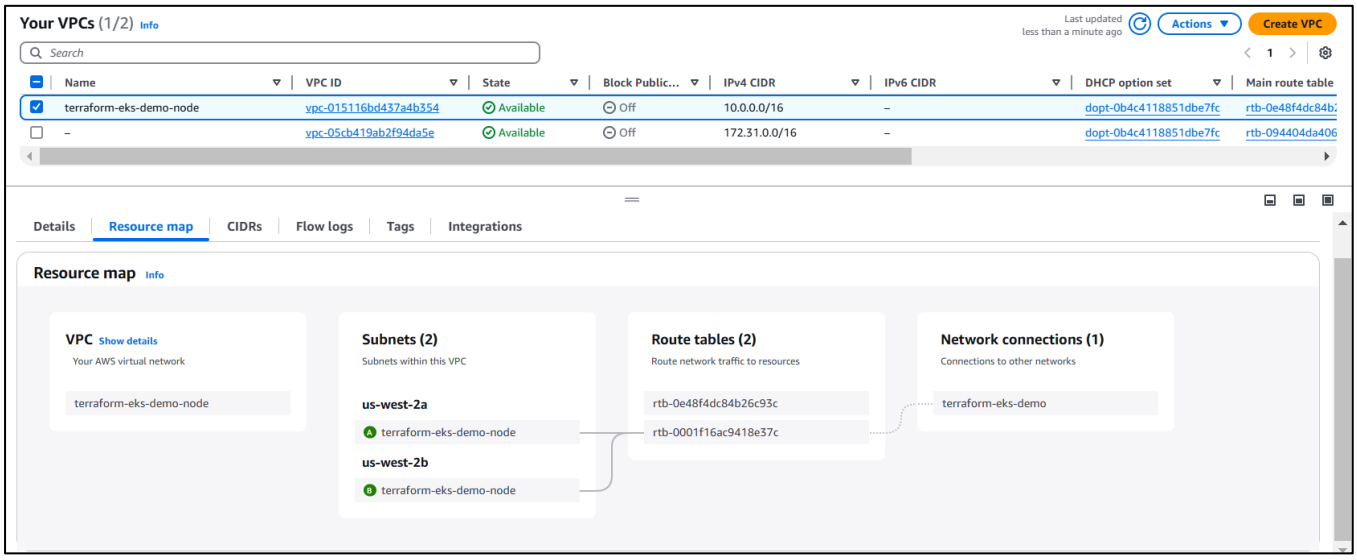

Overview
Resources
Compute
Networking
Add-ons
Access
Observability
Update history
Tags
Nodes (0) Info
Filter Nodes by property or value
Node name
Instance type
Compute
Managed by
Created
Status
No Nodes
This cluster does not have any Nodes, or you don't have permission to view them.
Node groups (1) Info
Edit
Delete
Add node group
Node groups implement basic compute scaling through EC2 Auto Scaling groups.
Group name
Desired size
AMI release version
Launch template
Status
demo
1
1.31.4-20250116
-
Active

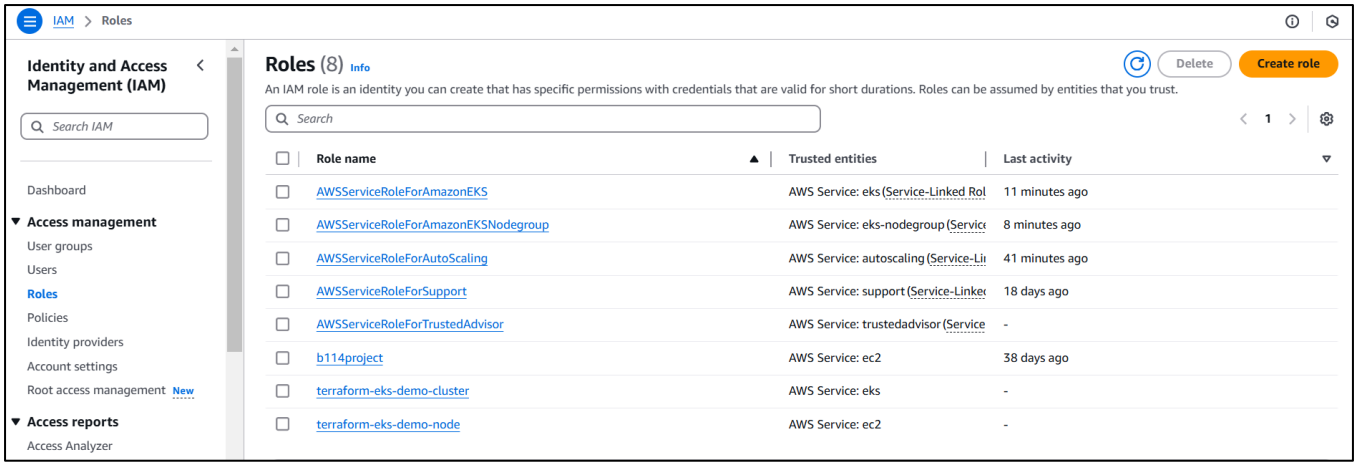

Overview
Resources
Compute
Networking
Add-ons
Access
Observability
Update history
Tags
Networking
Manage VPC resources
Manage endpoint access
VPC Info
vpc-015116bd437a4b354
Cluster IP address family Info
IPv4
Service IPv4 range Info
172.20.0.0/16
Subnets
subnet-02c1b14a21b7b6648
subnet-00cef28410ec7ae21
Cluster security group Info
sg-00ad0c66422879f7e
Additional security groups
sg-077344b207c384866
API server endpoint access Info
Public
Public access source allowlist
0.0.0.0/0 (open to all traffic)


Instances (1/1) Info
Last updated less than a minute ago
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states
Instance state = running
Clear filters
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
i-086d475edffc91a45
Running
t3.medium
3/3 checks passed
View alarms
us-west-2a


Instance summary for i-086d475edffc91a45 Info
Updated less than a minute ago
Connect
Instance state
Actions
Instance ID
i-086d475edffc91a45
IPv6 address
–
Hostname type
IP name: ip-10-0-0-219.us-west-2.compute.internal
Answer private resource DNS name
–
Auto-assigned IP address
34.218.229.201 [Public IP]
IAM Role
terraform-eks-demo-node
IMDSv2
Required
Operator
–
Public IPv4 address
34.218.229.201 | open address
Instance state
Running
Private IP DNS name (IPv4 only)
ip-10-0-0-219.us-west-2.compute.internal
Instance type
t3.medium
VPC ID
vpc-015116bd437a4b354 (terraform-eks-demo-node)
Subnet ID
subnet-00cef28410ec7ae21 (terraform-eks-demo-node)
Instance ARN
arn:aws:ec2:us-west-2:245712304097:instance/i-086d475edffc91a45
Private IPv4 addresses
10.0.0.82
10.0.0.219
Public IPv4 DNS
–
Elastic IP addresses
–
AWS Compute Optimizer finding
Opt-in to AWS Compute Optimizer for recommendations. | Learn more
Auto Scaling Group name
eks-demo-82ca4c52-c5b8-e5a7-bdb0-820f082680d7
Managed
false


Your VPCs (1/2) Info
Last updated less than a minute ago
Actions
Create VPC
Search
Name
VPC ID
State
Block Public...
IPv4 CIDR
IPv6 CIDR
DHCP option set
Main route table
terraform-eks-demo-node
vpc-015116bd437a4b354
Available
Off
10.0.0.0/16
–
dopt-0b4c4118851dbe7fc
rtb-0e48f4dc84b2
–
vpc-05cb419ab2f94da5e
Available
Off
172.31.0.0/16
–
dopt-0b4c4118851dbe7fc
rtb-094404da406
vpc-015116bd437a4b354 / terraform-eks-demo-node
Details
Resource map
CIDRs
Flow logs
Tags
Integrations
Details
VPC ID
vpc-015116bd437a4b354
State
Available
Block Public Access
Off
DNS hostnames
Disabled
DNS resolution
Enabled
Tenancy
default
DHCP option set
dopt-0b4c4118851dbe7fc
Main route table
rtb-0e48f4dc84b26c93c
Main network ACL
acl-04ca4448d43986c8b
Default VPC
No
IPv4 CIDR
10.0.0.0/16
IPv6 pool
–
IPv6 CIDR (Network border group)
–
Network Address Usage metrics
Disabled
Route 53 Resolver DNS Firewall rule groups
–
Owner ID
245712304097


Your VPCs (1/2) Info
Last updated less than a minute ago
Actions
Create VPC
Search
Name
VPC ID
State
Block Public...
IPv4 CIDR
IPv6 CIDR
DHCP option set
Main route table
terraform-eks-demo-node
vpc-015116bd437a4b354
Available
Off
10.0.0.0/16
–
dopt-0b4c4118851dbe7fc
rtb-0e48f4dc84b
–
vpc-05cb419ab2f94da5e
Available
Off
172.31.0.0/16
–
dopt-0b4c4118851dbe7fc
rtb-094404da406
Details
Resource map
CIDRs
Flow logs
Tags
Integrations
Resource map Info
VPC Show details
Your AWS virtual network
terraform-eks-demo-node
Subnets (2)
Subnets within this VPC
us-west-2a
terraform-eks-demo-node
us-west-2b
terraform-eks-demo-node
Route tables (2)
Route network traffic to resources
rtb-0e48f4dc84b26c93c
rtb-0001f16ac9418e37c
Network connections (1)
Connections to other networks
terraform-eks-demo



- The `kubectl` utility, which we installed earlier on our local computer, serves as the command-line interface to communicate with Kubernetes clusters. Once the EKS cluster has been successfully created in AWS, `kubectl` allows us to manage the cluster and its resources. This includes tasks such as deploying applications, scaling services, monitoring workloads, and managing configurations. By connecting `kubectl` to the cluster, we can execute commands directly from our local machine to interact with the Kubernetes environment running in AWS.
- To verify the connection between `kubectl` and the EKS cluster, use the following command: `kubectl get pods`

```
PS E:\AWS\DevOps\K8s\eks-cluster-terraform> kubectl get pods
E0124 19:48:39.644463    9912 memcache.go:265] "Unhandled Error" err="couldn't get current server API group list: Get \"http://localhost:808
0/api?timeout=32s\": dial tcp [::1]:8080: connectex: No connection could be made because the target machine actively refused it."
E0124 19:48:39.654812    9912 memcache.go:265] "Unhandled Error" err="couldn't get current server API group list: Get \"http://localhost:808
0/api?timeout=32s\": dial tcp [::1]:8080: connectex: No connection could be made because the target machine actively refused it."
E0124 19:48:39.661757    9912 memcache.go:265] "Unhandled Error" err="couldn't get current server API group list: Get \"http://localhost:808
0/api?timeout=32s\": dial tcp [::1]:8080: connectex: No connection could be made because the target machine actively refused it."
E0124 19:48:39.665025    9912 memcache.go:265] "Unhandled Error" err="couldn't get current server API group list: Get \"http://localhost:808
0/api?timeout=32s\": dial tcp [::1]:8080: connectex: No connection could be made because the target machine actively refused it."
E0124 19:48:39.679772    9912 memcache.go:265] "Unhandled Error" err="couldn't get current server API group list: Get \"http://localhost:808
0/api?timeout=32s\": dial tcp [::1]:8080: connectex: No connection could be made because the target machine actively refused it."
Unable to connect to the server: dial tcp [::1]:8080: connectex: No connection could be made because the target machine actively refused it.
PS E:\AWS\DevOps\K8s\eks-cluster-terraform>
```

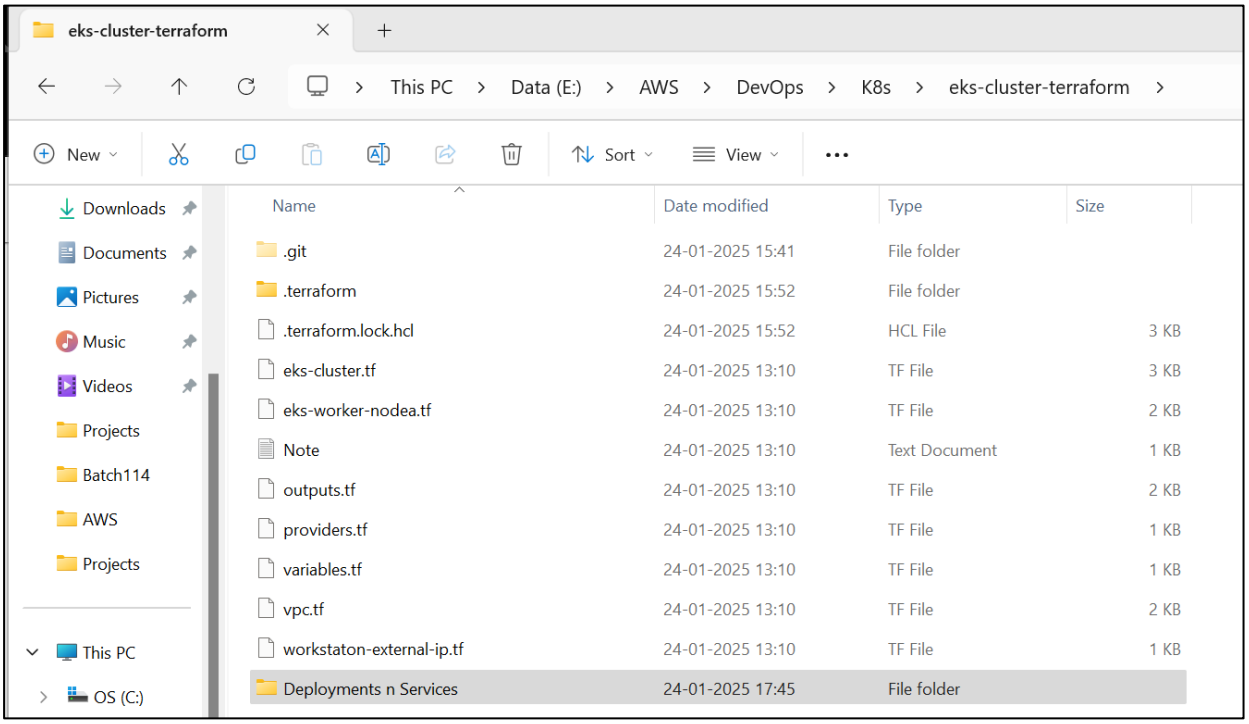
- If you encounter an error such as **`unable to connect to the server`**, it indicates that `kubectl` is not yet configured to interact with the cluster. To establish this connection, use the following command:

  `aws eks --region <region> update-kubeconfig --name <cluster-name>`

  For example, if your cluster is in the `us-west-2` region and its name is `terraform-eks-demo`, the command would be: `aws eks --region us-west-2 update-kubeconfig --name terraform-eks-demo`

```
PS E:\AWS\DevOps\K8s\eks-cluster-terraform> aws eks --region us-west-2 update-kubeconfig --name terraform-eks-demo
Added new context arn:aws:eks:us-west-2:245712304097:cluster/terraform-eks-demo to C:\Users\vivek\.kube\config
PS E:\AWS\DevOps\K8s\eks-cluster-terraform>
```

- Next, let's set up a deployment. Begin by creating a folder named **Deployment n Services** inside the **eks-cluster-terraform** directory, where your Terraform scripts are stored. Inside this folder, create a YAML file called **deployment.yml**. Obtain the deployment script from the official Kubernetes website, paste it into the file, and save it using **Ctrl+S**. This YAML file will define a deployment configuration with a replica set of 3, which will create and manage pods within the cluster.

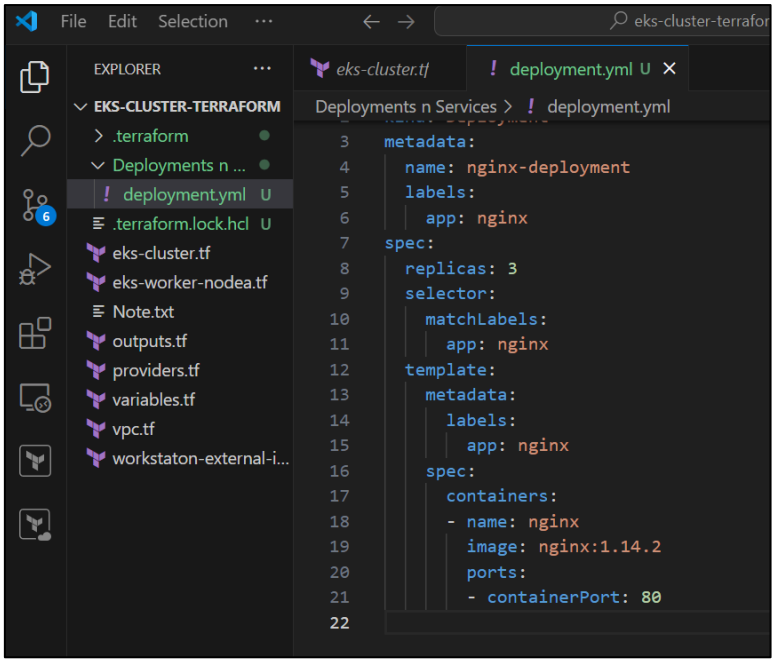


Creating a Deployment

The following is an example of a Deployment. It creates a ReplicaSet to bring up three `nginx` Pods:

controllers/nginx-deployment.yaml

```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
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```



- After saving the deployment file, run the command `kubectl apply -f deployment.yml` to create the pods. Once the command is executed, you can verify the pods have been created by running `kubectl get pods`.

```
PS E:\AWS\DevOps\K8s\eks-cluster-terraform\DeploymentsnServices> kubectl apply -f deployment.yml
deployment.apps/nginx-deployment created
PS E:\AWS\DevOps\K8s\eks-cluster-terraform\DeploymentsnServices>
```

```
PS E:\AWS\DevOps\K8s\eks-cluster-terraform\DeploymentsnServices> kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-92q4w    1/1     Running   0          30s
nginx-deployment-d556bf558-cbrg8    1/1     Running   0          30s
nginx-deployment-d556bf558-qr2fw    1/1     Running   0          30s
PS E:\AWS\DevOps\K8s\eks-cluster-terraform\DeploymentsnServices>
```

```
PS E:\AWS\DevOps\K8s\eks-cluster-terraform\DeploymentsnServices> kubectl get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/nginx-deployment-d556bf558-92q4w   1/1     Running   0          63s
pod/nginx-deployment-d556bf558-cbrg8   1/1     Running   0          63s
pod/nginx-deployment-d556bf558-qr2fw   1/1     Running   0          63s

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   172.20.0.1   <none>        443/TCP   30m

NAME                               READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx-deployment   3/3     3            3           65s

NAME                                          DESIRED   CURRENT   READY   AGE
replicaset.apps/nginx-deployment-d556bf558    3         3         3       65s
PS E:\AWS\DevOps\K8s\eks-cluster-terraform\DeploymentsnServices>
```

- If we delete any pods created through the deployment, new pods will automatically be created to maintain the specified replica set defined in the deployment configuration.

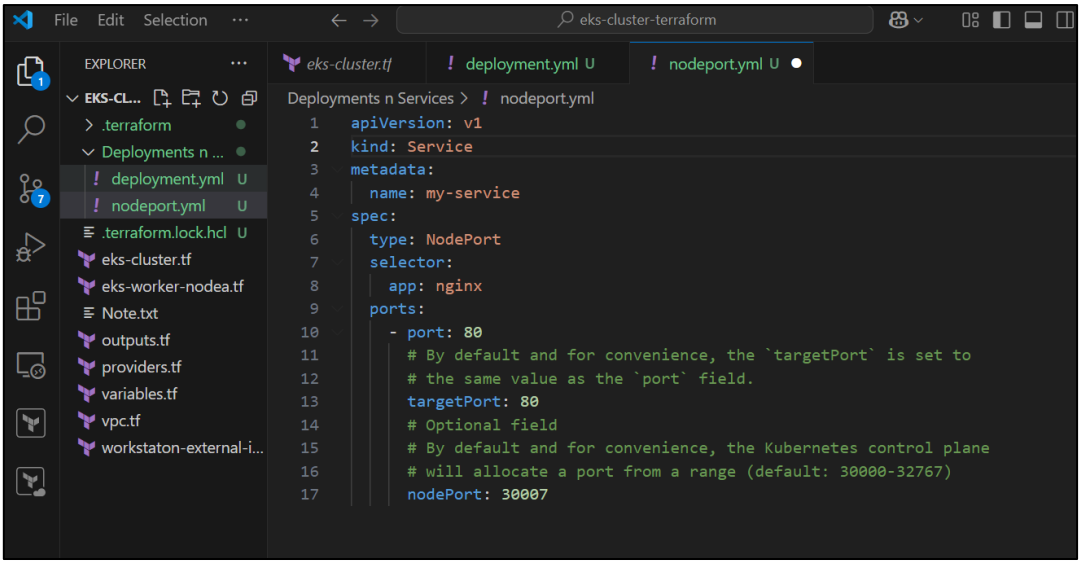
```
PS E:\AWS\DevOps\K8s\eks-cluster-terraform\DeploymentsnServices> kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-92q4w    1/1     Running   0          6m31s
nginx-deployment-d556bf558-cbrg8    1/1     Running   0          6m31s
nginx-deployment-d556bf558-qr2fw    1/1     Running   0          6m31s
PS E:\AWS\DevOps\K8s\eks-cluster-terraform\DeploymentsnServices> kubectl delete pod nginx-deployment-d556bf558-92q4w
pod "nginx-deployment-d556bf558-92q4w" deleted
PS E:\AWS\DevOps\K8s\eks-cluster-terraform\DeploymentsnServices> kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-cbrg8    1/1     Running   0          6m56s
nginx-deployment-d556bf558-qr2fw    1/1     Running   0          6m56s
nginx-deployment-d556bf558-vkj9v    1/1     Running   0          10s
PS E:\AWS\DevOps\K8s\eks-cluster-terraform\DeploymentsnServices>
```

- Next, visit the official Kubernetes website and download the YAML script for creating a NodePort service. Create a new file named **nodeport.yml** inside the **Deployments n Services** folder and paste the downloaded script into it.

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  type: NodePort
  selector:
    app.kubernetes.io/name: MyApp
  ports:
    - port: 80
      # By default and for convenience, the `targetPort` is set to
      # the same value as the `port` field.
      targetPort: 80
      # Optional field
      # By default and for convenience, the Kubernetes control plane
      # will allocate a port from a range (default: 30000-32767)
      nodePort: 30007
```

- In the NodePort script, you will notice the following selector:
  `selector:`
  `app.kubernetes.io/name: MyApp`
  However, in our deployment, the pods are created with the label `app: nginx`. To ensure the NodePort service targets the pods created by our deployment,

  replace:
  `app.kubernetes.io/name: MyApp`
  with:
  `app: nginx`
  This change will allow the NodePort service to apply correctly to the pods managed by the deployment.



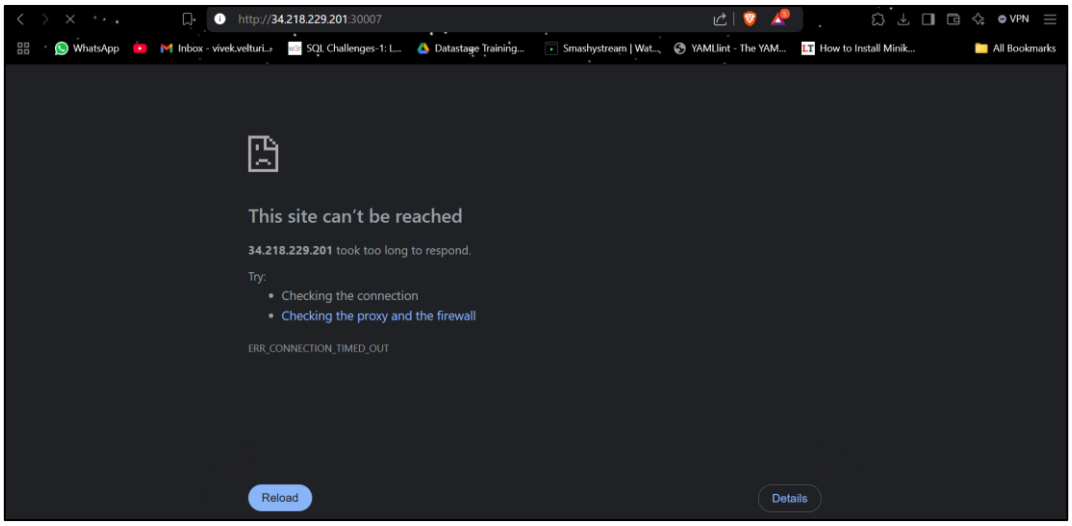
- After saving the nodeport.yml file, execute the command `kubectl apply -f nodeport.yml` to create the NodePort service. Once the service is successfully created, you can verify its status and details by using either `kubectl get svc` to list all services or `kubectl get all` to view all resources in the cluster, including the newly created service. These commands will confirm that the NodePort service has been applied and is ready to route traffic to the pods managed by your deployment.

```
PS E:\AWS\DevOps\K8s\eks-cluster-terraform\DeploymentsnServices> kubectl apply -f nodeport.yml
service/my-service created
PS E:\AWS\DevOps\K8s\eks-cluster-terraform\DeploymentsnServices>
```

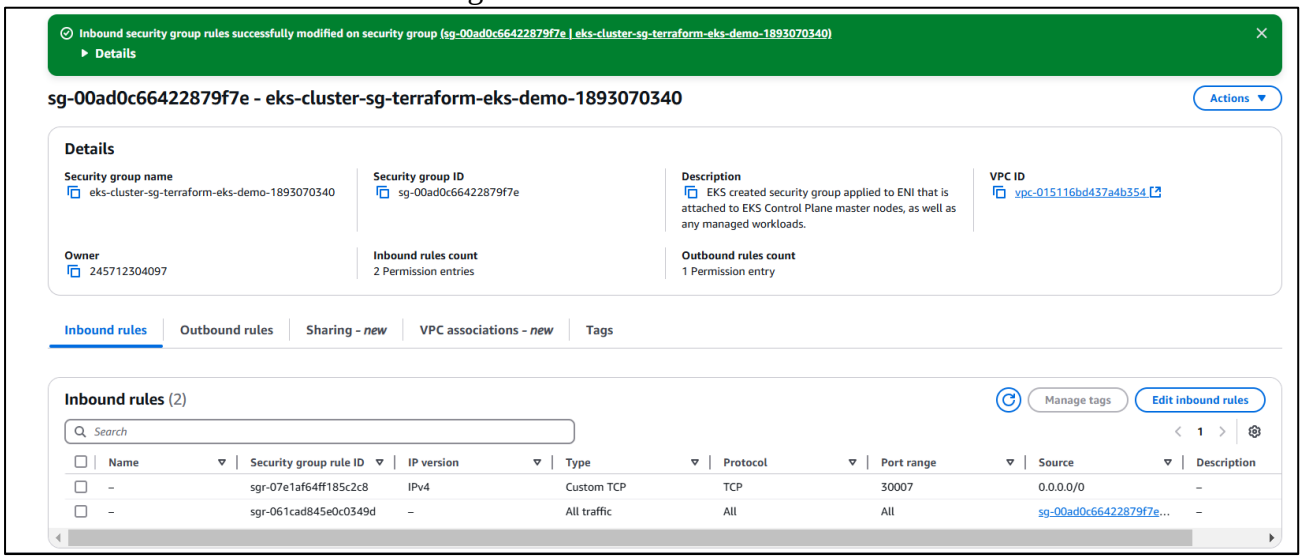
```
PS E:\AWS\DevOps\K8s\eks-cluster-terraform\DeploymentsnServices> kubectl get svc
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   172.20.0.1      <none>        443/TCP        31m
my-service   NodePort    172.20.23.213   <none>        80:30007/TCP   27s
PS E:\AWS\DevOps\K8s\eks-cluster-terraform\DeploymentsnServices>
```

```
PS E:\AWS\DevOps\K8s\eks-cluster-terraform\DeploymentsnServices> kubectl get all
NAME                                    READY   STATUS    RESTARTS   AGE
pod/nginx-deployment-d556bf558-92q4w    1/1     Running   0          2m58s
pod/nginx-deployment-d556bf558-cbrg8    1/1     Running   0          2m58s
pod/nginx-deployment-d556bf558-qr2fw    1/1     Running   0          2m58s

NAME                 TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
service/kubernetes   ClusterIP   172.20.0.1      <none>        443/TCP        31m
service/my-service   NodePort    172.20.23.213   <none>        80:30007/TCP   52s

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx-deployment    3/3     3            3           3m

NAME                                           DESIRED   CURRENT   READY   AGE
replicaset.apps/nginx-deployment-d556bf558     3         3         3       3m
PS E:\AWS\DevOps\K8s\eks-cluster-terraform\DeploymentsnServices>
```

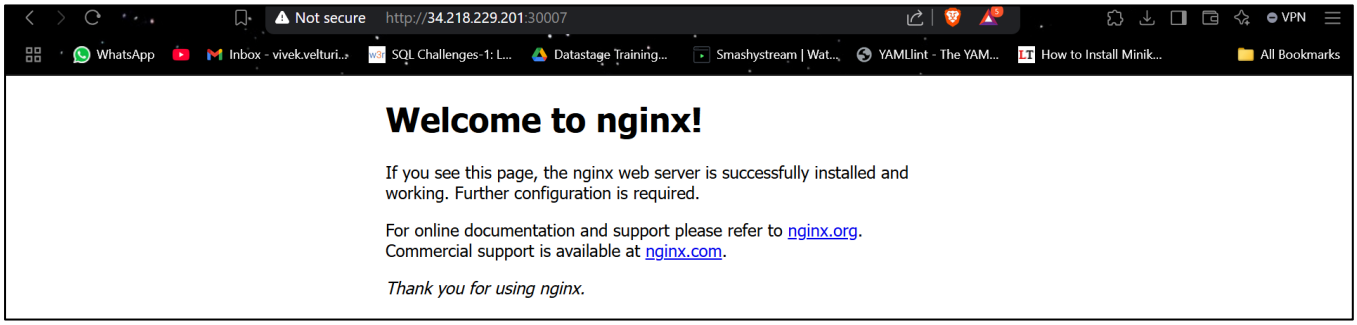
- In the NodePort script, we have specified port 80 for the host and port 30007 for the client. To verify the service, open your web browser and enter the public IP address of your node instance followed by `:30007`. For example, if your node instance's public IP is `34.218.229.201`, you would enter `http://34.218.229.201:30007` in the browser to check if the service is accessible.



- If the service is not accessible, it's likely because port `30007` has not been added to the security group associated with the node instance.
- Go to the AWS Management Console.
    - Open the **EC2 Dashboard** and locate the node instance created as part of your cluster.
    - **Modify Security Group**:
        - Select the **Security Group** attached to the node instance.
        - Click on **Edit Inbound Rules** to modify the access rules.
        - **Add Inbound Rule**:
            - Add a new rule:
                - **Type**: Custom TCP
                - **Port Range**: `30007`
                - **Source**: `0.0.0.0/0` (to allow access from any IP address).
            - Save the changes.



- **Test the Service Again,** Open your browser and enter the public IP address of the node followed by `:30007`.
  For example: `http:// 34.218.229.201:30007`.
- You should now be able to access the NGINX webpage.



- To delete the EKS cluster and all associated resources created using Terraform, follow these steps:

1. Open the **terminal** in the same directory where you previously executed the Terraform commands (`terraform init`, `terraform plan`, and `terraform apply`).
2. Run the following command to start the deletion process: `terraform destroy`
3. Terraform will display a list of resources it plans to delete. Carefully review the plan, and when prompted, confirm the deletion by typing `yes`.

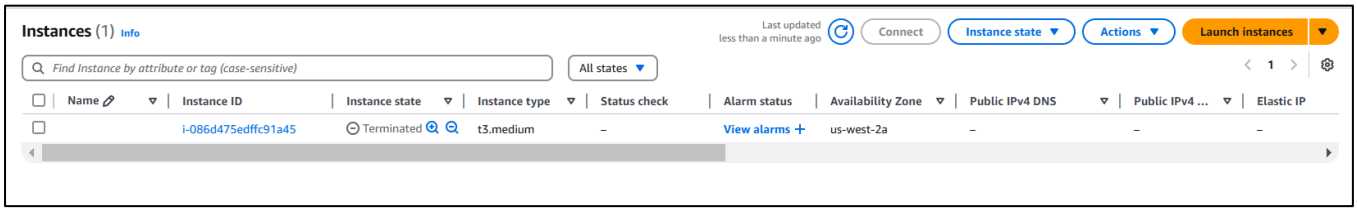
```
PS E:\AWS\DevOps\K8s\eks-cluster-terraform> terraform destroy
data.http.workstation-external-ip: Reading...
```

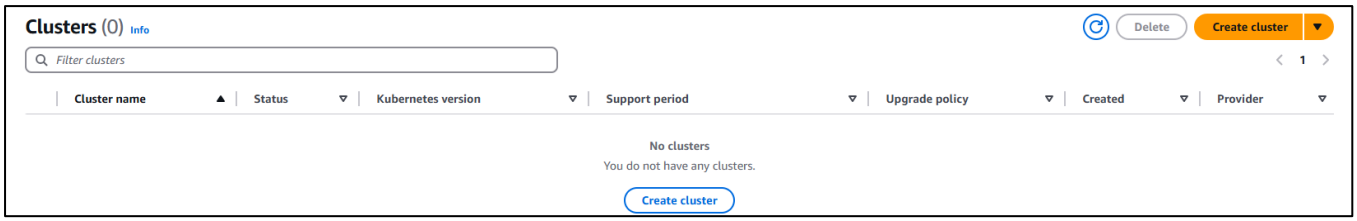
```
        } -> null
        # (4 unchanged attributes hidden)
    }

Plan: 0 to add, 0 to change, 18 to destroy.

Changes to Outputs:
  - config_map_aws_auth = <<-EOT
        apiVersion: v1
        kind: ConfigMap
```

```
Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes
```

```
aws_iam_role_policy_attachment.demo-cluster-AmazonEKSClusterPolicy: Destroying... [id=terraform-eks-demo-cluster-20250124140048196700000001]
aws_iam_role_policy_attachment.demo-cluster-AmazonEKSVPCResourceController: Destroying... [id=terraform-eks-demo-cluster-20250124140048333000000002]
aws_subnet.demo[1]: Destroying... [id=subnet-02c1b14a21b7b6648]
aws_subnet.demo[0]: Destroying... [id=subnet-00cef28410ec7ae21]
aws_security_group.demo-cluster: Destroying... [id=sg-077344b207c384866]
aws_iam_role_policy_attachment.demo-cluster-AmazonEKSClusterPolicy: Destruction complete after 1s
aws_iam_role_policy_attachment.demo-cluster-AmazonEKSVPCResourceController: Destruction complete after 1s
aws_iam_role.demo-cluster: Destroying... [id=terraform-eks-demo-cluster]
aws_iam_role.demo-cluster: Destruction complete after 0s
aws_subnet.demo[0]: Destruction complete after 2s
aws_subnet.demo[1]: Destruction complete after 2s
aws_security_group.demo-cluster: Destruction complete after 2s
aws_vpc.demo: Destroying... [id=vpc-015116bd437a4b354]
aws_vpc.demo: Destruction complete after 1s

Destroy complete! Resources: 18 destroyed.
PS E:\AWS\DevOps\K8s\eks-cluster-terraform>
```

4. Wait for the process to complete. Terraform will remove all the resources it managed, including the EKS cluster, EC2 instances, and networking components.
5. After completion, you can verify the deletion by logging into the AWS Management Console and checking the **EKS** and **EC2** sections to ensure no resources remain.





This process ensures that all AWS resources created by your Terraform scripts are properly removed, avoiding unnecessary charges.