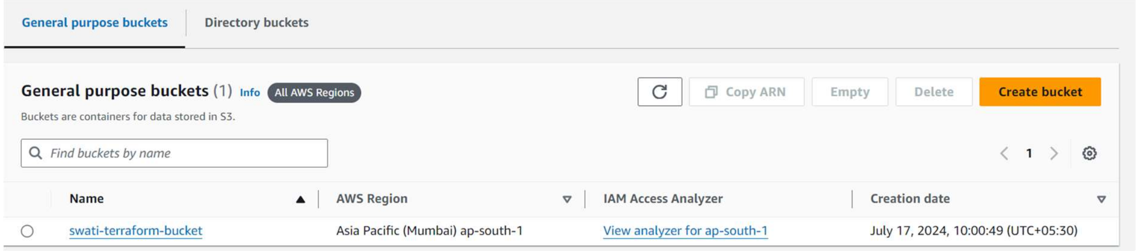
AWS EKS Cluster Automation with Terraform

Objective: Create an AWS EKS cluster using Terraform. Build VPC, IAM user, roles and Worker nodes. Use S3 and DynamoDB for locking.

Step 1: Create an IAM user "terraform-user". Allow S3FullAccess and attach the following policies to it:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket",
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject",
        "s3:GetBucketLocation"
      ],
      "Resource": [
        "arn:aws:s3:::my-terraform-state-bucket",
        "arn:aws:s3:::my-terraform-state-bucket/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:PutItem",
        "dynamodb:DeleteItem",
        "dynamodb:GetItem",
        "dynamodb:Scan",
        "dynamodb:Query",
        "dynamodb:UpdateItem"
      ],
      "Resource": "arn:aws:dynamodb:us-west-
2:YOUR_AWS_ACCOUNT_ID:table/terraform-lock"
    }
  ]
}
```

Step 2: Create an S3 bucket

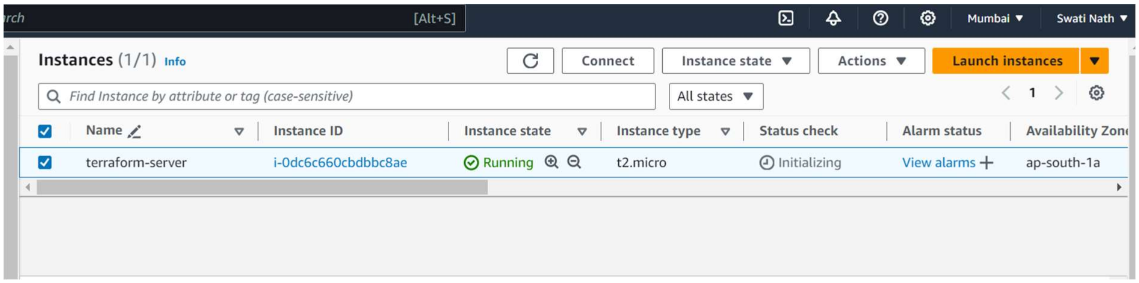


General purpose buckets (1)

Buckets are containers for data stored in S3.

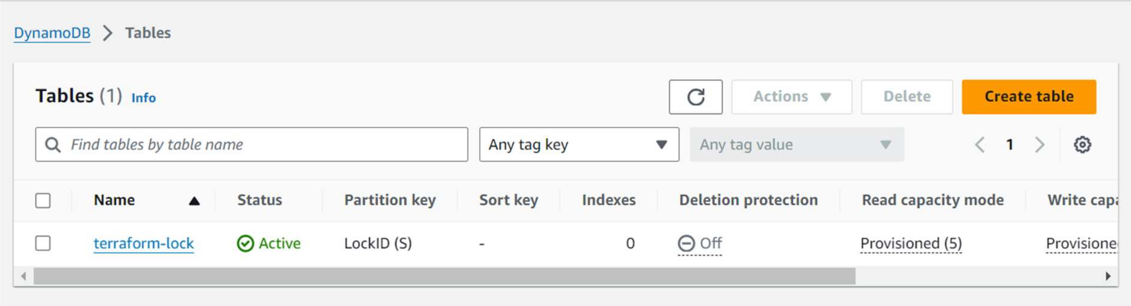
	Name	AWS Region	IAM Access Analyzer	Creation date
○	swati-terraform-bucket	Asia Pacific (Mumbai) ap-south-1	View analyzer for ap-south-1	July 17, 2024, 10:00:49 (UTC+05:30)

Step 3: Launch an ec2 instance as "terraform-server"



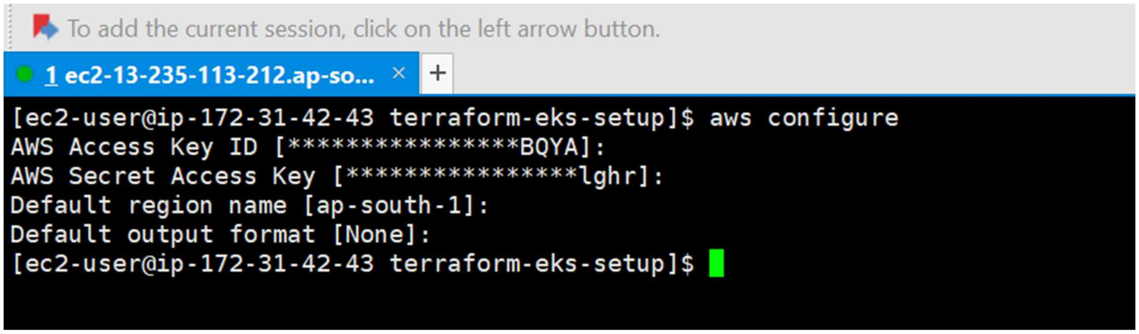
☑	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zon
☑	terraform-server	i-0dc6c660cbdbbc8ae	✓ Running	t2.micro	Initializing	View alarms +	ap-south-1a

Step 4: Create a table in DynamoDB as "terraform-lock". (We can also skip this step and create the table using terraform script)



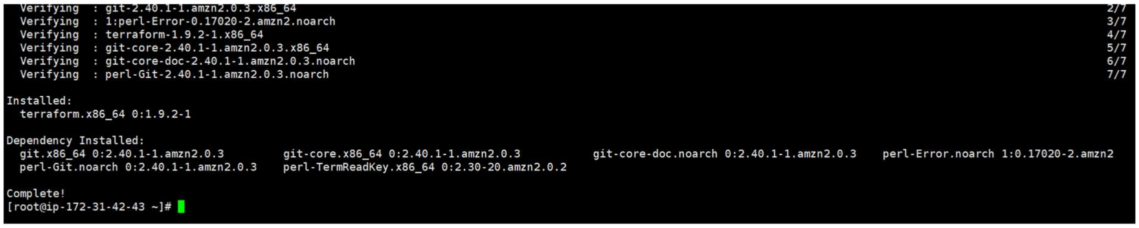
□	Name ▲	Status	Partition key	Sort key	Indexes	Deletion protection	Read capacity mode	Write capa
□	terraform-lock	✓ Active	LockID (S)	-	0	Off	Provisioned (5)	Provisione

Step 5: Connect to the terraform-server using SSH. Configure AWS on the server using "aws configure". Input the access key and secret key generated for the terraform-user. We are using ap-south-1 as our default region.


To add the current session, click on the left arrow button.
1 ec2-13-235-113-212.ap-so...

```
[ec2-user@ip-172-31-42-43 terraform-eks-setup]$ aws configure
AWS Access Key ID [****************BQYA]:
AWS Secret Access Key [****************lghr]:
Default region name [ap-south-1]:
Default output format [None]:
[ec2-user@ip-172-31-42-43 terraform-eks-setup]$
```

Step 6: Install terraform on the server (refer official documentation)



```
  Verifying  : git-2.40.1-1.amzn2.0.3.x86_64                                                                                                   2/7
  Verifying  : 1:perl-Error-0.17020-2.amzn2.noarch                                                                                             3/7
  Verifying  : terraform-1.9.2-1.x86_64                                                                                                        4/7
  Verifying  : git-core-2.40.1-1.amzn2.0.3.x86_64                                                                                              5/7
  Verifying  : git-core-doc-2.40.1-1.amzn2.0.3.noarch                                                                                          6/7
  Verifying  : perl-Git-2.40.1-1.amzn2.0.3.noarch                                                                                              7/7

Installed:
  terraform.x86_64 0:1.9.2-1

Dependency Installed:
  git.x86_64 0:2.40.1-1.amzn2.0.3         git-core.x86_64 0:2.40.1-1.amzn2.0.3          git-core-doc.noarch 0:2.40.1-1.amzn2.0.3    perl-Error.noarch 1:0.17020-2.amzn2
  perl-Git.noarch 0:2.40.1-1.amzn2.0.3    perl-TermReadKey.x86_64 0:2.30-20.amzn2.0.2

Complete!
[root@ip-172-31-42-43 ~]#
```

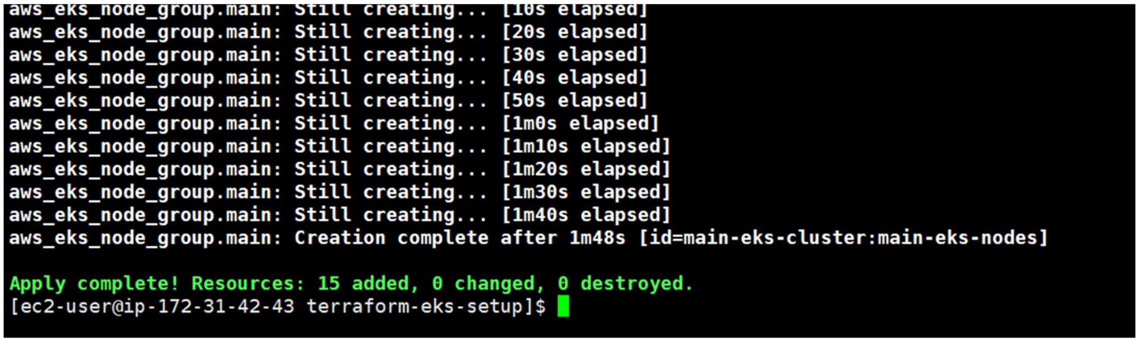
Step 7: Create a new directory: mkdir terraform-eks-setup

Navigate into the directory: cd terraform-eks-setup

Create a main.tf file and use the following script:

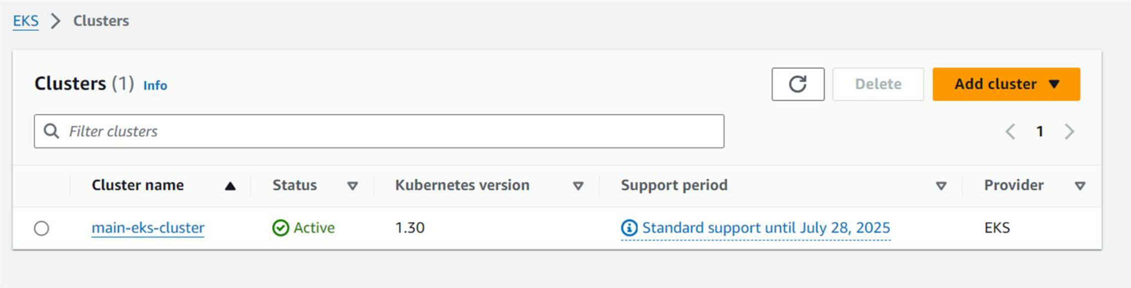
```
# Configure the AWS provider
provider "aws" {
  region = "ap-south-1"  # Replace with your desired AWS region
}

# Create a VPC
resource "aws_vpc" "main" {
  cidr_block = "10.0.0.0/16"

  tags = {
    Name = "main-vpc"
  }
}

# Create Public Subnets
resource "aws_subnet" "public" {
  count             = 2
  vpc_id            = aws_vpc.main.id
  cidr_block        = cidrsubnet(aws_vpc.main.cidr_block, 8,
count.index)
  availability_zone = element(["ap-south-1a", "ap-south-1c"],
count.index)
map_public_ip_on_launch = true

  tags = {
    Name = "public-subnet-${count.index}"
  }
}

# Create an Internet Gateway
resource "aws_internet_gateway" "main" {
  vpc_id = aws_vpc.main.id

  tags = {
    Name = "main-igw"
  }
}

# Create a Route Table for Public Subnets
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-route-table"
  }
}

# Associate the Route Table with Public Subnets
resource "aws_route_table_association" "public" {
  count          = 2
  subnet_id      = element(aws_subnet.public.*.id, count.index)
  route_table_id = aws_route_table.public.id
}

# Create IAM Roles and Policies for EKS
resource "aws_iam_role" "eks_cluster" {
  name = "eks-cluster-role"

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

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

resource "aws_iam_role" "eks_worker" {
  name = "eks-worker-role"

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

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

resource "aws_iam_role_policy_attachment"
"eks_worker_AmazonEC2ContainerRegistryReadOnly" {
  policy_arn =
"arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly"
  role       = aws_iam_role.eks_worker.name
}

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

# Create the EKS Cluster
resource "aws_eks_cluster" "main" {
  name     = "main-eks-cluster"
  role_arn = aws_iam_role.eks_cluster.arn

  vpc_config {
    subnet_ids = aws_subnet.public.*.id
  }

  tags = {
    Name = "main-eks-cluster"
  }
}

# Create Worker Nodes
resource "aws_eks_node_group" "main" {
  cluster_name    = aws_eks_cluster.main.name
  node_group_name = "main-eks-nodes"
  node_role_arn   = aws_iam_role.eks_worker.arn
  subnet_ids      = aws_subnet.public.*.id

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

  instance_types = ["t3.medium"]
  tags = {
    Name = "main-eks-nodes"
  }
}

# Configure the S3 Backend for Terraform State
terraform {
  backend "s3" {
    bucket         = "my-terraform-state-bucket"  # Replace with your
bucket name
    key            = "eks-cluster/terraform.tfstate"
    region         = "us-east-1"
    dynamodb_table = "terraform-lock"
    encrypt        = true
  }
}
```

Step 8: Save and exit the file. Use the following commands to create the resources: terraform init, terraform plan, terraform apply



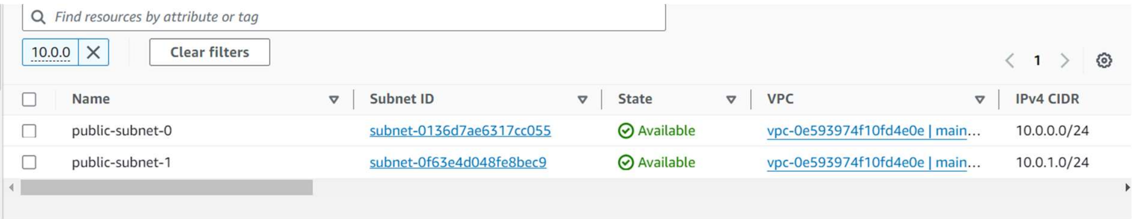
```
aws_eks_node_group.main: Still creating... [10s elapsed]
aws_eks_node_group.main: Still creating... [20s elapsed]
aws_eks_node_group.main: Still creating... [30s elapsed]
aws_eks_node_group.main: Still creating... [40s elapsed]
aws_eks_node_group.main: Still creating... [50s elapsed]
aws_eks_node_group.main: Still creating... [1m0s elapsed]
aws_eks_node_group.main: Still creating... [1m10s elapsed]
aws_eks_node_group.main: Still creating... [1m20s elapsed]
aws_eks_node_group.main: Still creating... [1m30s elapsed]
aws_eks_node_group.main: Still creating... [1m40s elapsed]
aws_eks_node_group.main: Creation complete after 1m48s [id=main-eks-cluster:main-eks-nodes]

Apply complete! Resources: 15 added, 0 changed, 0 destroyed.
[ec2-user@ip-172-31-42-43 terraform-eks-setup]$ 
```

Step 9: Verify the resources hence created. Check the cluster created on EKS

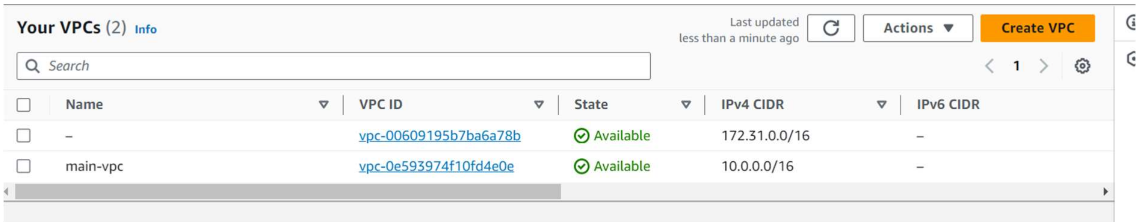


	Cluster name ▲	Status ▽	Kubernetes version ▽	Support period ▽	Provider ▽
○	main-eks-cluster	✓ Active	1.30	Standard support until July 28, 2025	EKS

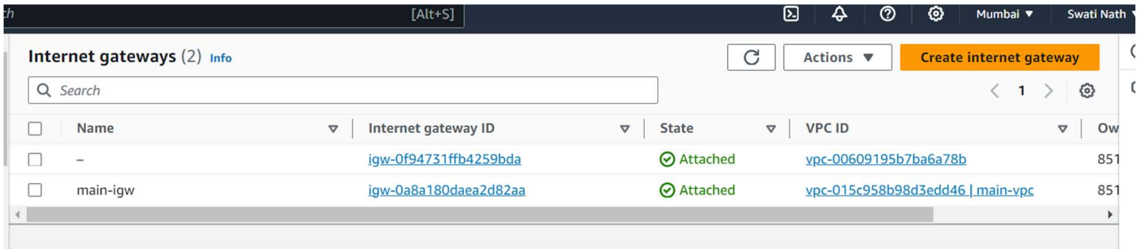
Find the VPC, subnets and internet gateway created



Name	Subnet ID	State	VPC	IPv4 CIDR
public-subnet-0	subnet-0136d7ae6317cc055	Available	vpc-0e593974f10fd4e0e | main…	10.0.0.0/24
public-subnet-1	subnet-0f63e4d048fe8bec9	Available	vpc-0e593974f10fd4e0e | main…	10.0.1.0/24

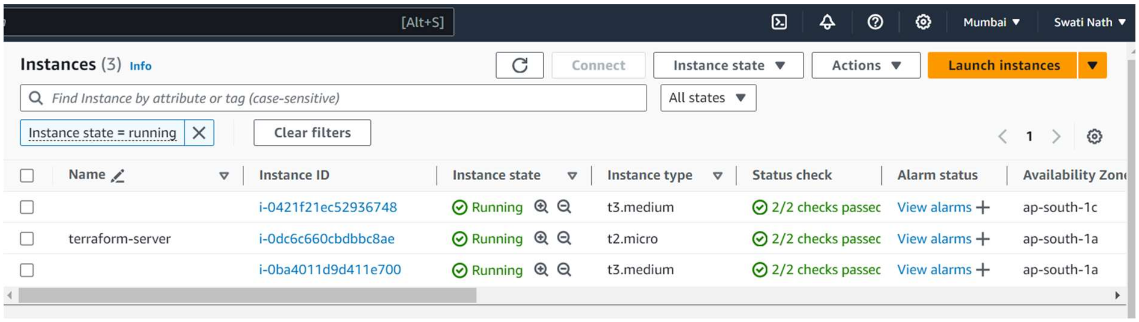


Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR
–	vpc-00609195b7ba6a78b	Available	172.31.0.0/16	–
main-vpc	vpc-0e593974f10fd4e0e	Available	10.0.0.0/16	–



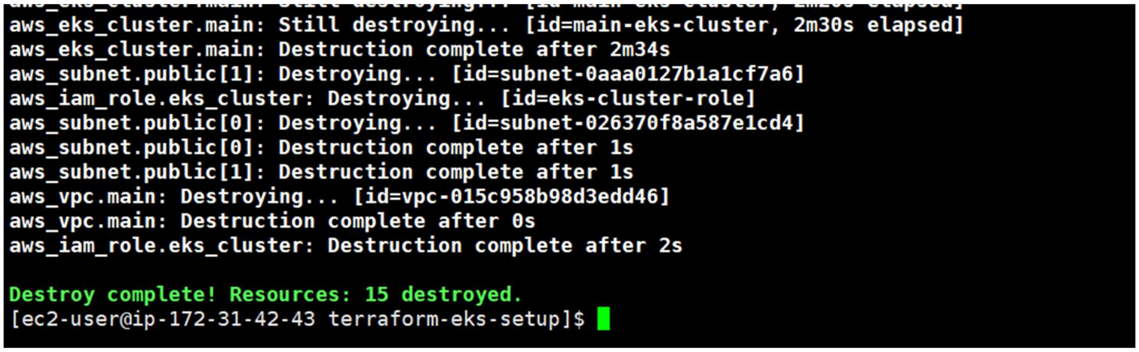
Name	Internet gateway ID	State	VPC ID	Ow
–	igw-0f94731ffb4259bda	Attached	vpc-00609195b7ba6a78b	851
main-igw	igw-0a8a180daea2d82aa	Attached	vpc-015c958b98d3edd46 | main-vpc	851

We have 2 instances launched on the EC2 console as specified in the main.tf script



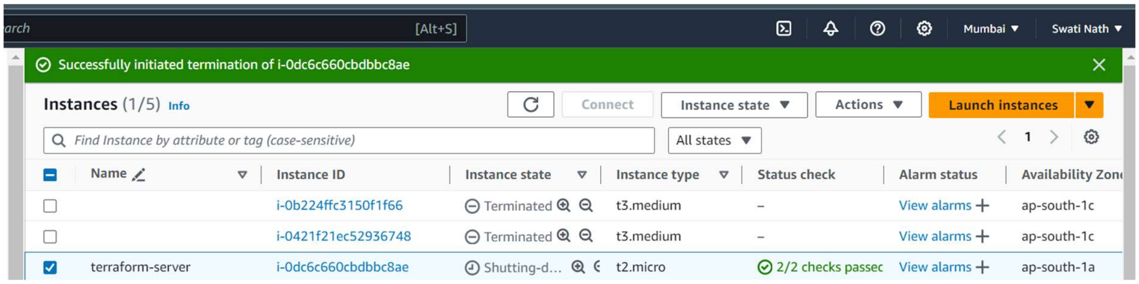
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zon
	i-0421f21ec52936748	Running	t3.medium	2/2 checks passec	View alarms +	ap-south-1c
terraform-server	i-0dc6c660cbdbbc8ae	Running	t2.micro	2/2 checks passec	View alarms +	ap-south-1a
	i-0ba4011d9d411e700	Running	t3.medium	2/2 checks passec	View alarms +	ap-south-1a

Step 10: Use “terraform destroy” to delete all the resources created



```
aws_eks_cluster.main: Still destroying... [id=main-eks-cluster, 2m20s elapsed]
aws_eks_cluster.main: Still destroying... [id=main-eks-cluster, 2m30s elapsed]
aws_eks_cluster.main: Destruction complete after 2m34s
aws_subnet.public[1]: Destroying... [id=subnet-0aaa0127b1a1cf7a6]
aws_iam_role.eks_cluster: Destroying... [id=eks-cluster-role]
aws_subnet.public[0]: Destroying... [id=subnet-026370f8a587e1cd4]
aws_subnet.public[0]: Destruction complete after 1s
aws_subnet.public[1]: Destruction complete after 1s
aws_vpc.main: Destroying... [id=vpc-015c958b98d3edd46]
aws_vpc.main: Destruction complete after 0s
aws_iam_role.eks_cluster: Destruction complete after 2s

Destroy complete! Resources: 15 destroyed.
[ec2-user@ip-172-31-42-43 terraform-eks-setup]$
```

Step 11: Once all the resources have been destroyed, terminate the instance “terraform-server”



Successfully initiated termination of i-0dc6c660cbdbbc8ae

Instances (1/5) Info

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
□		i-0b224ffc3150f1f66	Terminated	t3.medium	–	View alarms +	ap-south-1c
□		i-0421f21ec52936748	Terminated	t3.medium	–	View alarms +	ap-south-1c
☑	terraform-server	i-0dc6c660cbdbbc8ae	Shutting-d...	t2.micro	2/2 checks passec	View alarms +	ap-south-1a