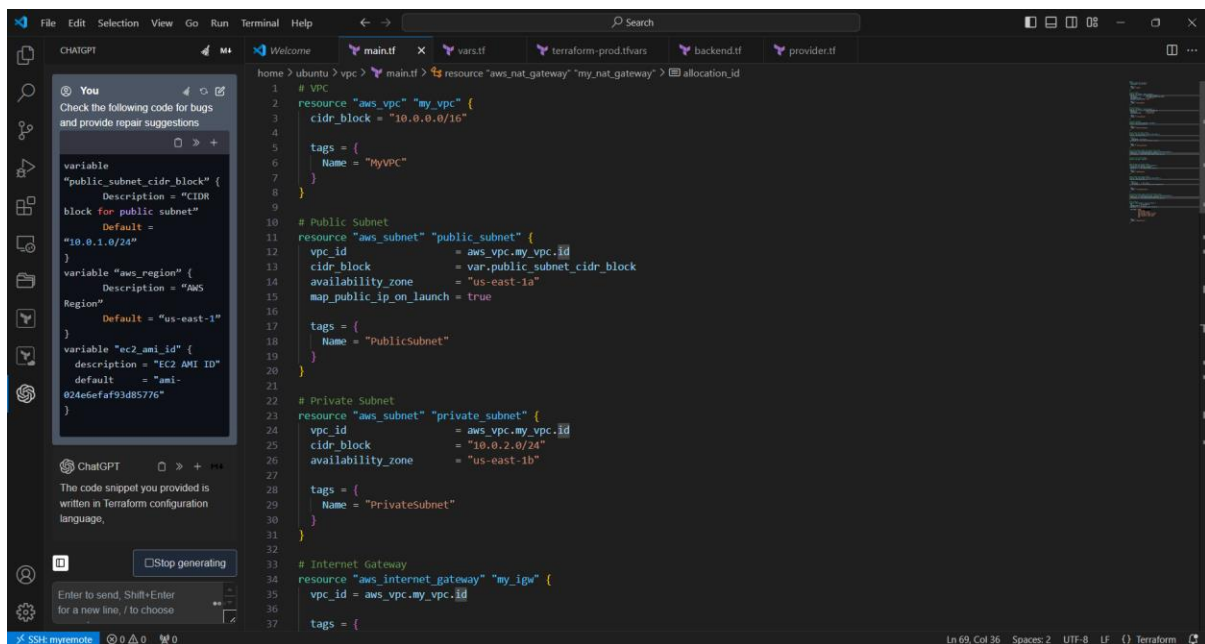


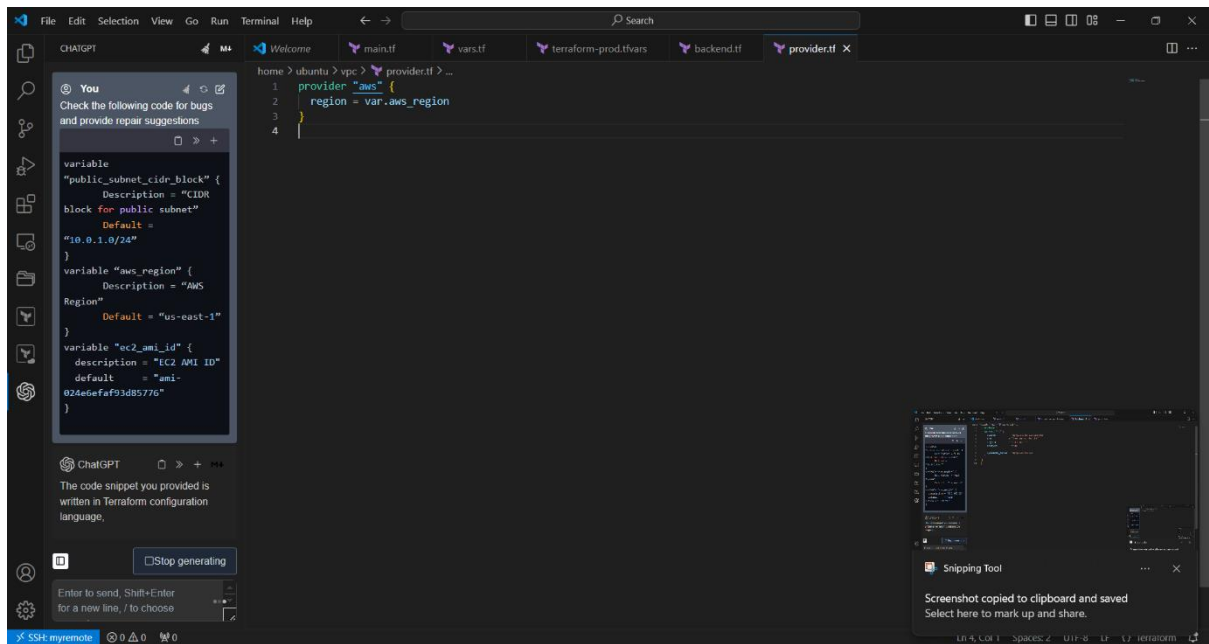
AWS VPC Creation in Terraform

1. Created an own VPC.
2. Created a Public and Private subnet for different Available AZs by assigning different CIDR blocks.
3. Created Internet Gateway & attach it to the VPC.
4. Created Routing table [RT], One as Public & One as Private by associating the appropriate subnets to it.
5. Edited the Public route table's Route alone and map the IGW, not the Private and leave it as it is.
6. Created NAT gateway with new Elastic IP for the internet connection in the Private Subnet. Map it to Private RT.
7. Created an ec2 in public subnet and given the user data to launch an Apache web page in it.
8. Created a s3 bucket to store the terraform statefile in it .
9. Created a DynamoDB table to lock the statefile in s3 bucket.
10. Created the files separately to be reused for dev environment, prod environment etc
 - Main.tf
 - Provider.tf
 - Vars.tf
 - terraform-prod.tfvars

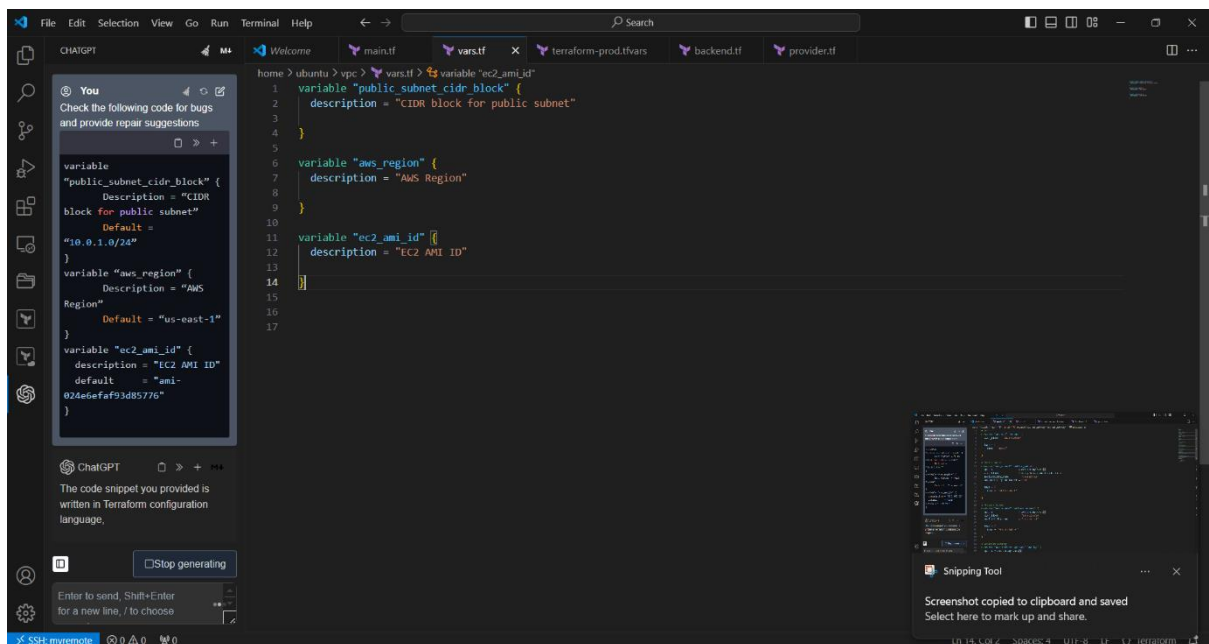


```
1 # VPC
2 resource "aws_vpc" "my_vpc" {
3   cidr_block = "10.0.0.0/16"
4
5   tags = {
6     Name = "MyVpc"
7   }
8 }
9
10 # Public Subnet
11 resource "aws_subnet" "public_subnet" {
12   vpc_id         = aws_vpc.my_vpc.id
13   cidr_block      = var-public_subnet_cidr_block
14   availability_zone = "us-east-1a"
15   map_public_ip_on_launch = true
16
17   tags = {
18     Name = "PublicSubnet"
19   }
20 }
21
22 # Private Subnet
23 resource "aws_subnet" "private_subnet" {
24   vpc_id         = aws_vpc.my_vpc.id
25   cidr_block      = "10.0.2.0/24"
26   availability_zone = "us-east-1b"
27
28   tags = {
29     Name = "PrivateSubnet"
30   }
31 }
32
33 # Internet Gateway
34 resource "aws_internet_gateway" "my_igw" {
35   vpc_id = aws_vpc.my_vpc.id
36
37   tags = {
```

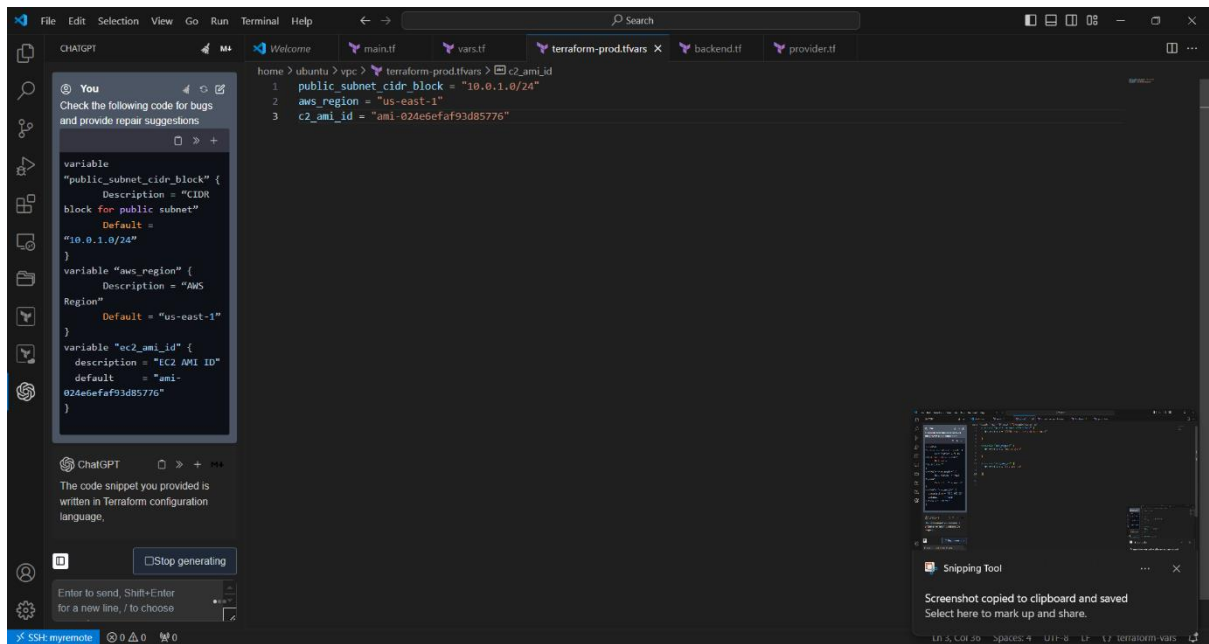
1.created main.tf file with all the resources



2.created provide.tf to specify cloud provider

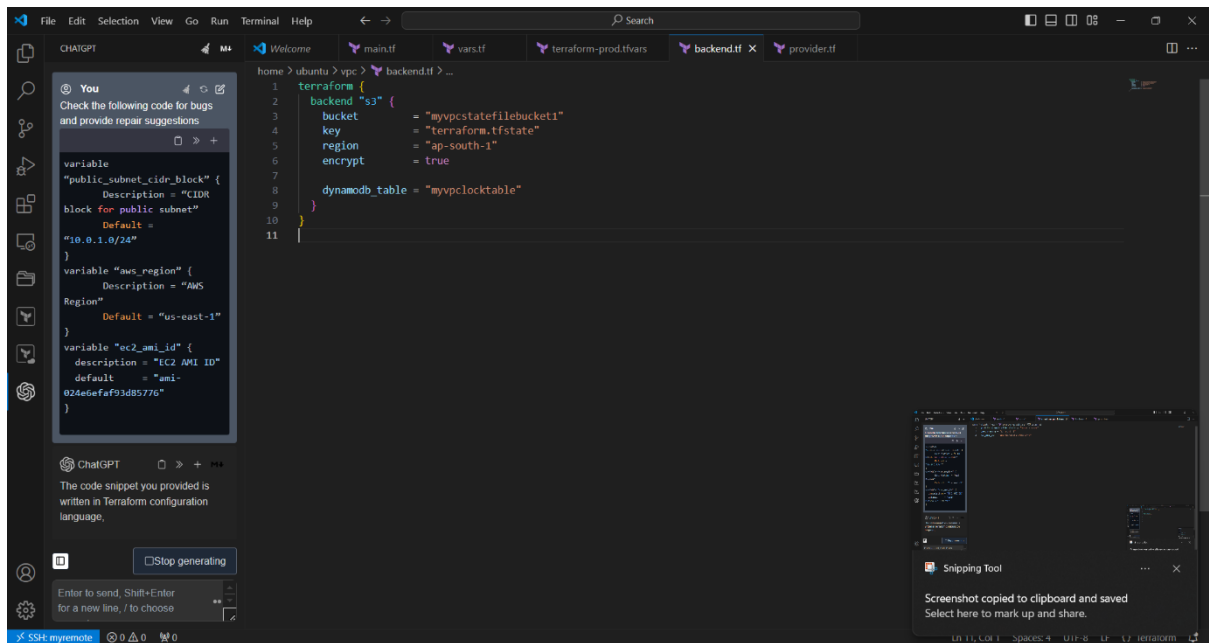


3.created vars.tf to specify variables



4.created terraform-prod. tfvars to specify the variables for reuse the code for different environments

Terraform init -var-file = "terraform-prod. Tfvars" use this command to specify the environment.



5.created backend.tf file to store the statefile in s3 bucket and locked with DynamoDB

The screenshot shows a VS Code editor with a Terraform configuration file named `main.tf`. The configuration defines a VPC, a public subnet, and an EC2 instance. A ChatGPT sidebar on the left provides a code snippet for a public subnet. The terminal window shows the execution of `terraform init`, which initializes the backend and provider plugins. The output indicates that the backend is successfully configured and the provider plugins are initialized.

```
home > ubuntu > vpc > main.tf > resource "aws_instance" "my_ec2_instance" > ami  
1 # VPC  
2 resource "aws_vpc" "my_vpc" {  
3   cidr_block = "10.0.0.0/16"  
4  
5   tags = {  
6     Name = "MyVPC"  
7   }  
8  
9  
10 # Public Subnet  
11 resource "aws_subnet" "public_subnet" {  
12   vpc_id = aws_vpc.my_vpc.id  
13   cidr_block = var.public_subnet_cidr_block  
14   availability_zone = "us-east-1a"  
15   map_public_ip_on_launch = true  
16  
17   tags = {  
18     Name = "PublicSubnet"  
19   }  
20 }
```

ChatGPT sidebar content:

```
variable  
"public_subnet_cidr_block" {  
  Description = "CIDR  
block for public subnet"  
  Default =  
  "10.0.1.0/24"  
}  
variable "aws_region" {  
  Description = "AWS  
Region"  
  Default = "us-east-1"  
}  
variable "ec2_ami_id" {  
  description = "EC2 AMI ID"  
  default = "ami-  
024e6efaf93d85776"  
}
```

Terminal output:

```
ubuntu@ip-172-31-32-20:~$ terraform init  
Initializing the backend...  
Do you want to copy existing state to the new backend?  
Pre-existing state was found while migrating the previous "local" backend to the  
newly configured "s3" backend. No existing state was found in the newly  
configured "s3" backend. Do you want to copy this state to the new "s3"  
backend? Enter "yes" to copy and "no" to start with an empty state.  
  
Enter a value: yes  
  
Successfully configured the backend "s3"! Terraform will automatically  
use this backend unless the backend configuration changes.  
Initializing provider plugins...  
- Reusing previous version of hashicorp/aws from the dependency lock file  
- Using previously-installed hashicorp/aws v5.60.0
```

6.terraform init

The screenshot shows the same VS Code editor with the `main.tf` file. The terminal window now shows the execution of `terraform plan`. The output displays the plan for creating the VPC, subnet, and EC2 instance. The ChatGPT sidebar remains on the left, providing the same code snippet for the public subnet.

```
home > ubuntu > vpc > main.tf > resource "aws_instance" "my_ec2_instance" > ami  
1 # VPC  
2 resource "aws_vpc" "my_vpc" {  
3   cidr_block = "10.0.0.0/16"  
4  
5   tags = {  
6     Name = "MyVPC"  
7   }  
8  
9  
10 # Public Subnet  
11 resource "aws_subnet" "public_subnet" {  
12   vpc_id = aws_vpc.my_vpc.id  
13   cidr_block = var.public_subnet_cidr_block  
14   availability_zone = "us-east-1a"  
15   map_public_ip_on_launch = true  
16  
17   tags = {  
18     Name = "PublicSubnet"  
19   }  
20 }
```

ChatGPT sidebar content:

```
variable  
"public_subnet_cidr_block" {  
  Description = "CIDR  
block for public subnet"  
  Default =  
  "10.0.1.0/24"  
}  
variable "aws_region" {  
  Description = "AWS  
Region"  
  Default = "us-east-1"  
}  
variable "ec2_ami_id" {  
  description = "EC2 AMI ID"  
  default = "ami-  
024e6efaf93d85776"  
}
```

Terminal output:

```
ubuntu@ip-172-31-32-20:~$ terraform plan  
aws_eip.my_eip: Refreshing state... [id=eipalloc-0ff7f9cbbd494d6b6]  
aws_vpc.my_vpc: Refreshing state... [id=vpc-0be34ee976148384]  
aws_subnet.private_subnet: Refreshing state... [id=subnet-0b21c8138955acba0]  
aws_subnet.public_subnet: Refreshing state... [id=subnet-05c796056396b3ae]  
aws_internet_gateway.my_igw: Refreshing state... [id=igw-0e01e523c50c832c]  
aws_instance.my_ec2_instance: Refreshing state... [id=i-04f60f13b82c2e7d0]  
aws_nat_gateway.my_nat_gateway: Refreshing state... [id=nat-09fb7426e2061804d]  
aws_route_table.private_route_table: Refreshing state... [id=rtb-0e1811a8dcbcf78e8]  
aws_route_table.public_route_table: Refreshing state... [id=rtbassoc-0e31d40e0eb5cb0c]  
aws_route_table_association.private_subnet_association: Refreshing state... [id=rtbassoc-0bd4d8754b17eee05]
```

7.terraform plan

```
home > ubuntu > vpc > main.tf > resource "aws_instance" "my_ec2_instance" > ami
1 # VPC
2 resource "aws_vpc" "my_vpc" {
3   cidr_block = "10.0.0.0/16"
4
5   tags = {
6     Name = "MyVPC"
7   }
8 }
9
10 # Public Subnet
11 resource "aws_subnet" "public_subnet" {
12   vpc_id = aws_vpc.my_vpc.id
13   cidr_block = var.public_subnet_cidr_block
14 }

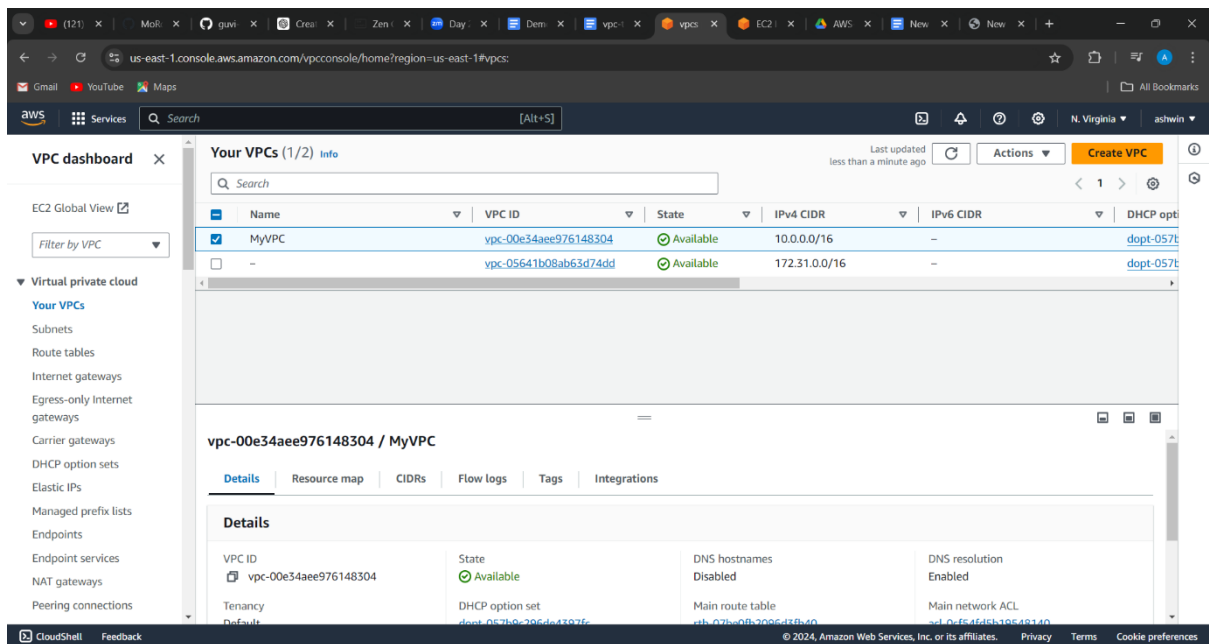
No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.
ubuntu@ip-172-31-32-20:~$ terraform apply
aws_eip.my_eip: Refreshing state... [id=eipalloc-0ff7f9cbbd494db6]
aws_vpc.my_vpc: Refreshing state... [id=vpc-00e34aee976148304]
aws_subnet.public_subnet: Refreshing state... [id=subnet-05c7960563966b3ae]
aws_subnet.private_subnet: Refreshing state... [id=subnet-0b21c8138955acba9]
aws_internet_gateway.my_igw: Refreshing state... [id=igw-0e01e5223c50c832e]
aws_route_table.public_route_table: Refreshing state... [id=rtb-0076a228b8b22370a]
aws_instance.my_ec2_instance: Refreshing state... [id=i-04f60f13b82c2e7d0]
aws_nat_gateway.my_nat_gateway: Refreshing state... [id=nat-09fb7426e206180ad]
aws_route_table_association.public_subnet_association: Refreshing state... [id=rtbassoc-0831d4050ebc5cbec]
aws_route_table.private_route_table: Refreshing state... [id=rtb-0e1811a0dcfcf78e8]
aws_route_table_association.private_subnet_association: Refreshing state... [id=rtbassoc-0bd4d8754b17ee05]

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.
Apply complete! Resources: 0 added, 0 changed, 0 destroyed.
ubuntu@ip-172-31-32-20:~$
```

8.terraform apply



9.vpc has created

us-east-1.console.aws.amazon.com/vpconsole/home?region=us-east-1#subnets:

VPC dashboard

Subnets (8) Info

Find resources by attribute or tag

Last updated less than a minute ago

Actions Create subnet

Name	Subnet ID	State	VPC	IPv4 CIDR
-	subnet-ucc9d23u58g8838ec	Available	vpc-05641b08ab63d74dd	172.31.1.0/20
-	subnet-0a33c7f30343f1660	Available	vpc-05641b08ab63d74dd	172.31.32.0/20
-	subnet-0dd7321cae7e0afef	Available	vpc-05641b08ab63d74dd	172.31.80.0/20
PublicSubnet	subnet-05c7960563966b3ae	Available	vpc-00e34aee976148304 MyV...	10.0.1.0/24
-	subnet-02c76f85ed4796c23	Available	vpc-05641b08ab63d74dd	172.31.48.0/20
PrivateSubnet	subnet-0b21c8138955acba9	Available	vpc-00e34aee976148304 MyV...	10.0.2.0/24
-	subnet-09fef97899a9a9bdc	Available	vpc-05641b08ab63d74dd	172.31.64.0/20

Select a subnet

CloudShell Feedback

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

10 created subnets public and private in different availability zones

us-east-1.console.aws.amazon.com/vpconsole/home?region=us-east-1#RouteTables:

VPC dashboard

Route tables (4) Info

Find resources by attribute or tag

Last updated 1 minute ago

Actions Create route table

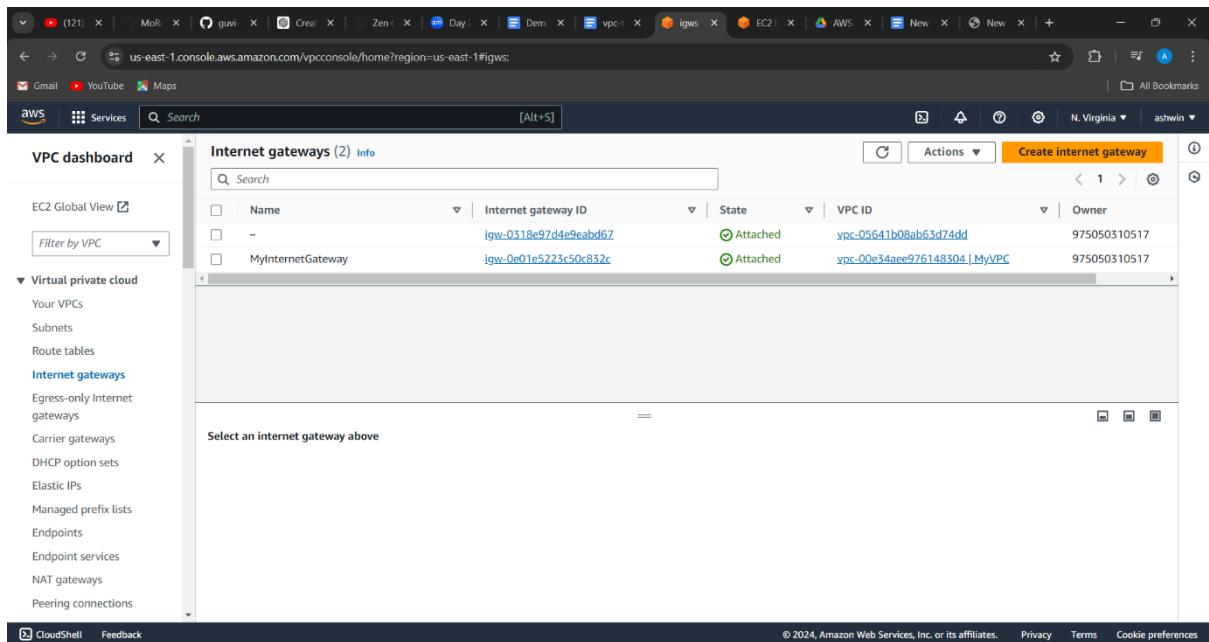
Name	Route table ID	Explicit subnet associ...	Edge associations	Main	VPC
PublicRouteTable	rtb-0076a228b5b22370a	subnet-05c7960563966b...	-	No	vpc-00e34aee976148304 M
PrivateRouteTable	rtb-0e1811a0dcbcf78e8	subnet-0b21c8138955ac...	-	No	vpc-00e34aee976148304 M
-	rtb-01308c186c41a93f2	-	-	Yes	vpc-05641b08ab63d74dd
-	rtb-07be0fb2096d3fb40	-	-	Yes	vpc-00e34aee976148304 M

Select a route table

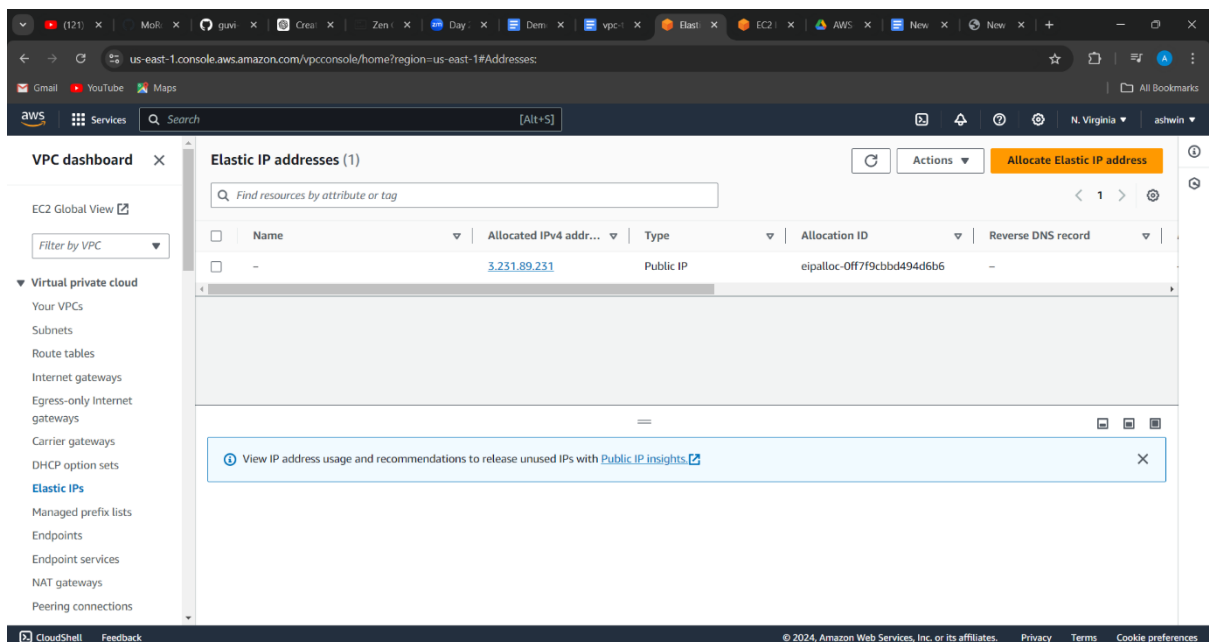
CloudShell Feedback

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

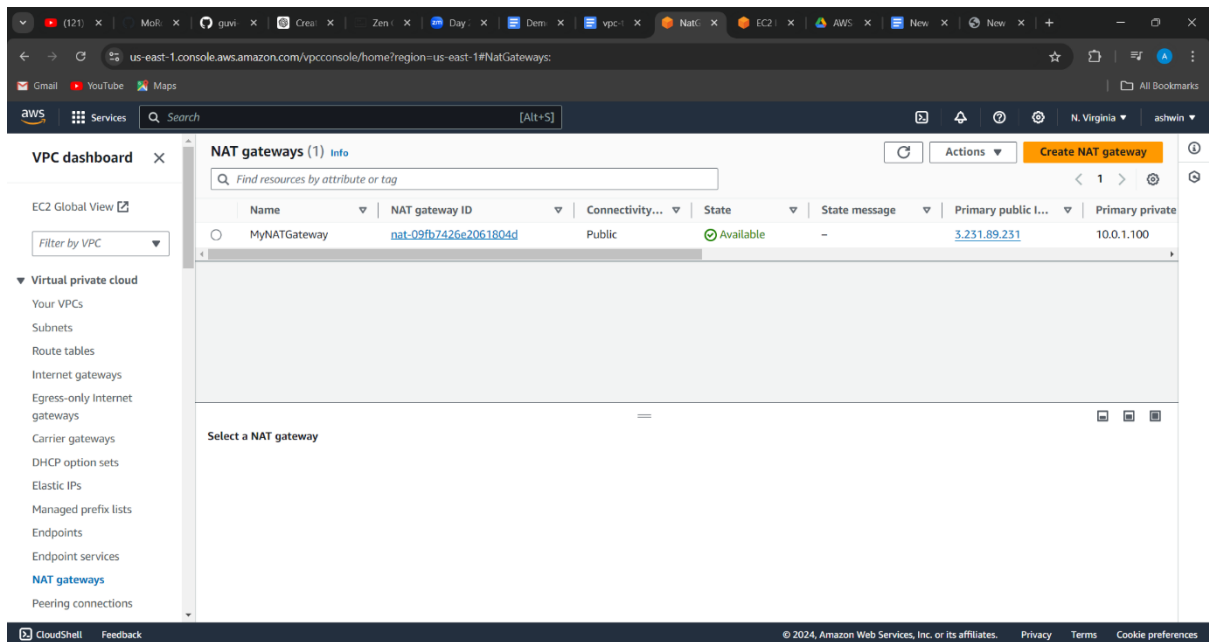
11.created route tables for public and private.



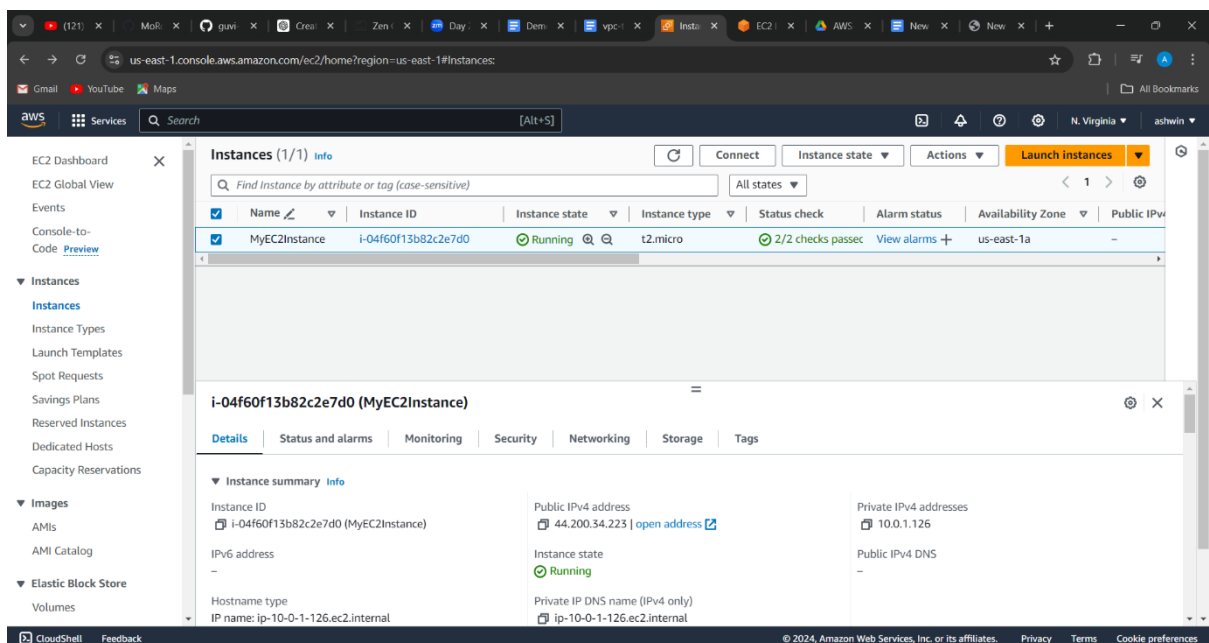
12.created igw and attached to vpc and public route table .



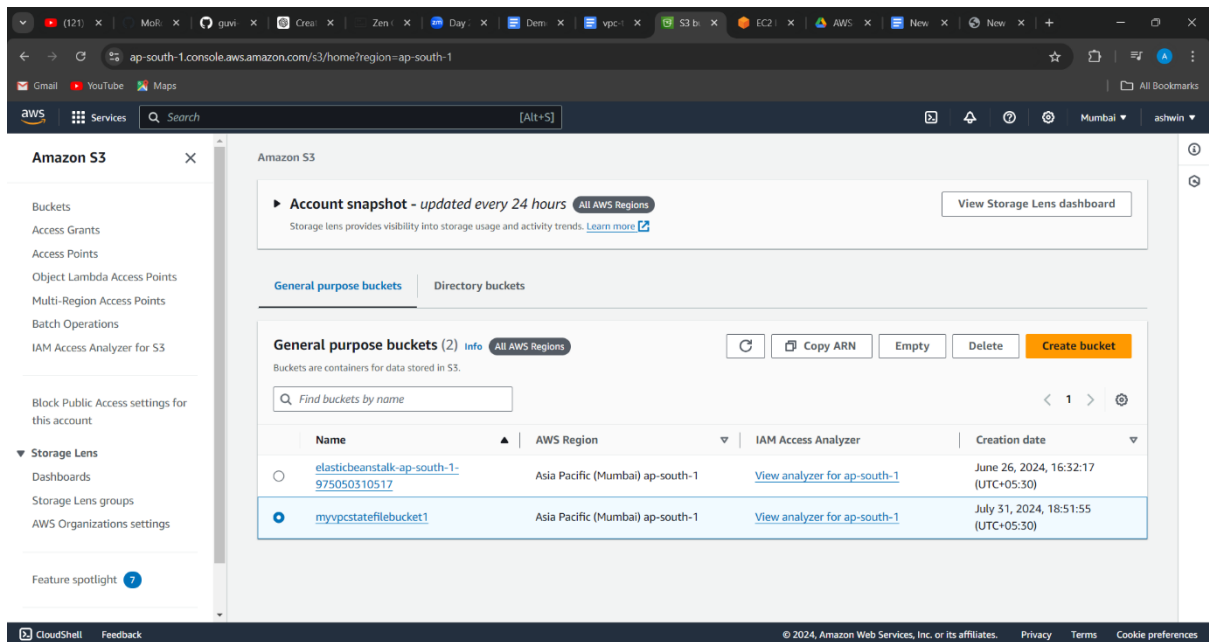
13.created a elastic ip to create NAT gateway



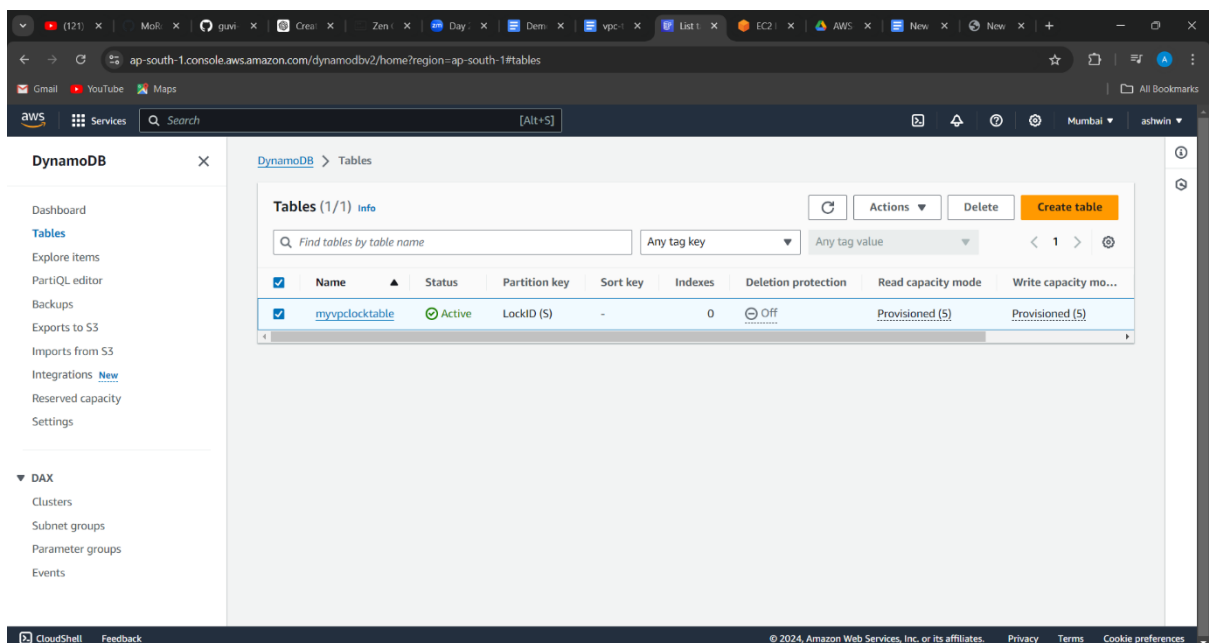
14. created a nat gateway and attached to private route table



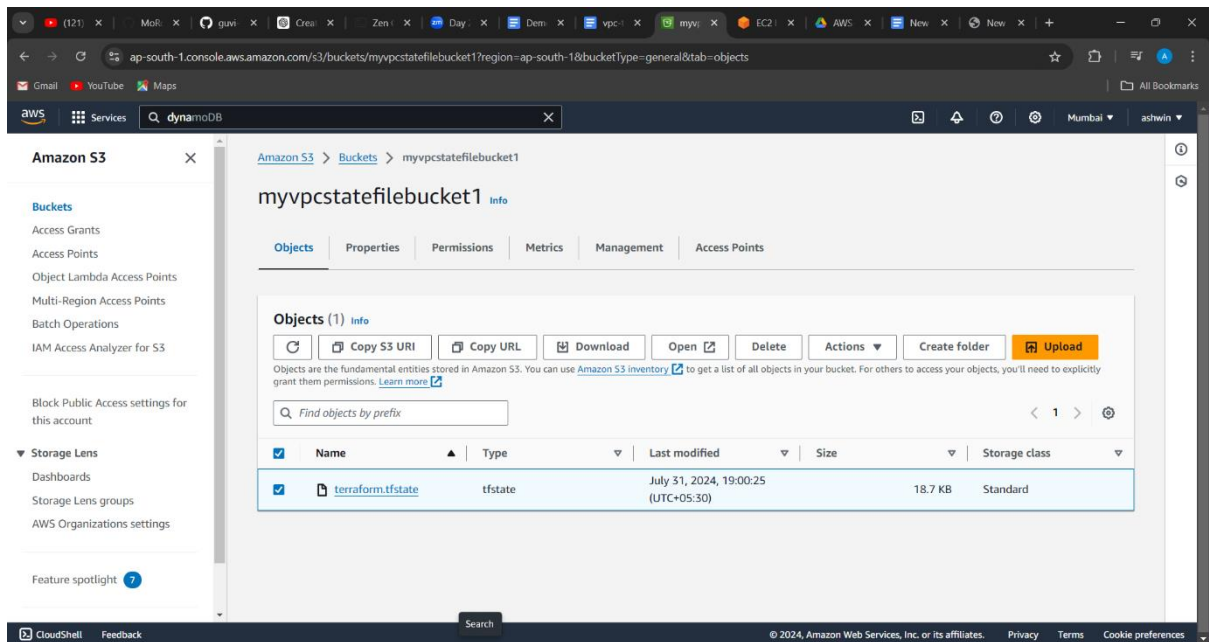
15. A ec2 machine has created in public subnet of the vpc



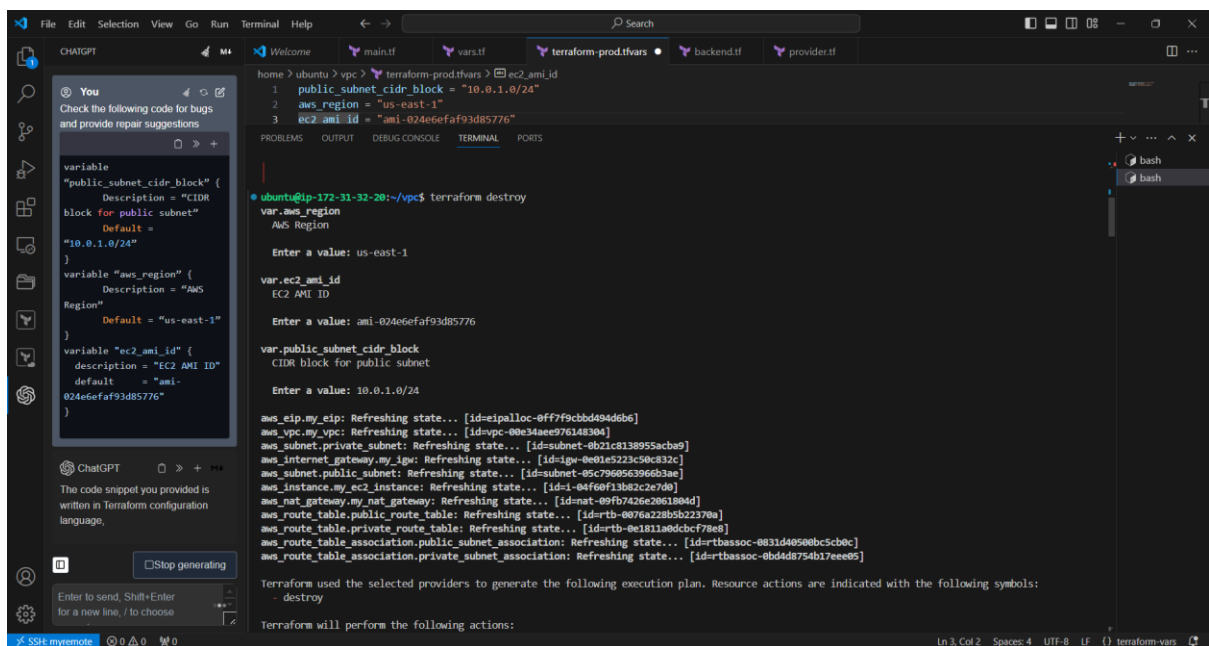
16. s3 bucket is created to store statefile



17.A dynamodb table has been created to lock the statefile



18.terraform statefile has been pushed to s3 bucket



19.terraform destroy it has asking for some variables to be destroyed because of DynamoDB lock

