Terraform Task:

===============

1) Create VPC

2) Create Internet gateway

3) Create Custom Route Table

4) Create Subnet

5) Associate subnet with Route Table

6) Create Security Group to allow port 22.80,443

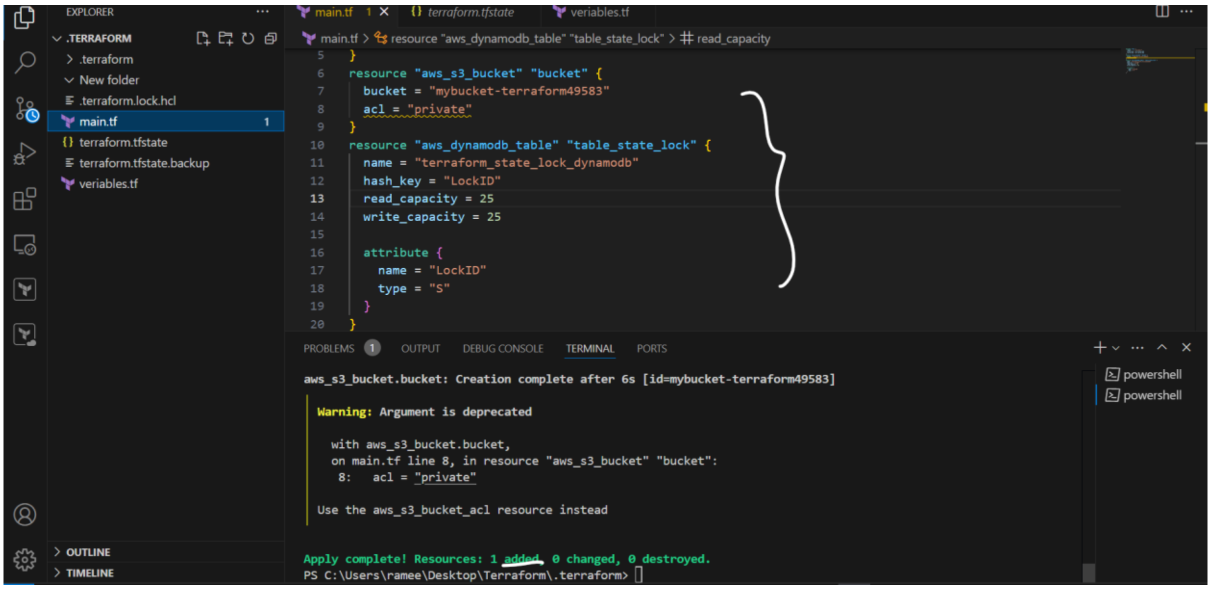
7) Create a network interface with an ip in the subnet that was created in step 4

8) Assign an elastic IP to the network interface created in step 7

9) Create Ubuntu server and install/enable apache

Note:
1) Create single main.tf which will be created the above resources and do not hardcode the id's.
2) Configure s3 as backend and dynamo db locking for multi user execution.

Now here first I am creating first s3 and dynamodb.

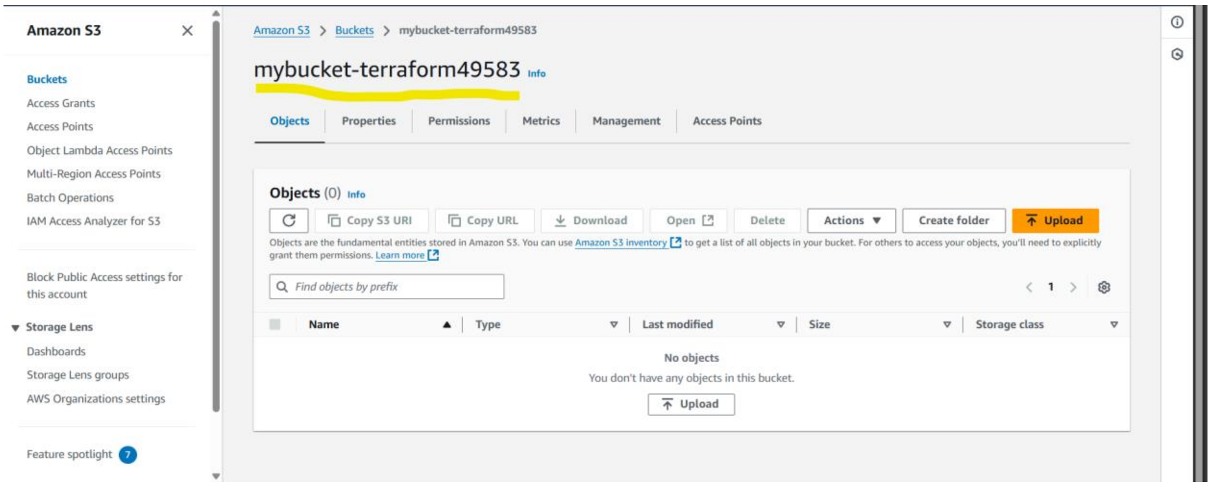


```
}
resource "aws_s3_bucket" "bucket" {
  bucket = "mybucket-terraform49583"
  acl = "private"
}
resource "aws_dynamodb_table" "table_state_lock" {
  name = "terraform_state_lock_dynamodb"
  hash_key = "LockID"
  read_capacity = 25
  write_capacity = 25

  attribute {
    name = "LockID"
    type = "S"
  }
}
```

```
aws_s3_bucket.bucket: Creation complete after 6s [id=mybucket-terraform49583]

  Warning: Argument is deprecated

    with aws_s3_bucket.bucket,
    on main.tf line 8, in resource "aws_s3_bucket" "bucket":
     8:   acl = "private"

  Use the aws_s3_bucket_acl resource instead


Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\ramee\Desktop\Terraform\.terraform>
```

Successfully here changes are apply.

So here my bucket is created.



mybucket-terraform49583

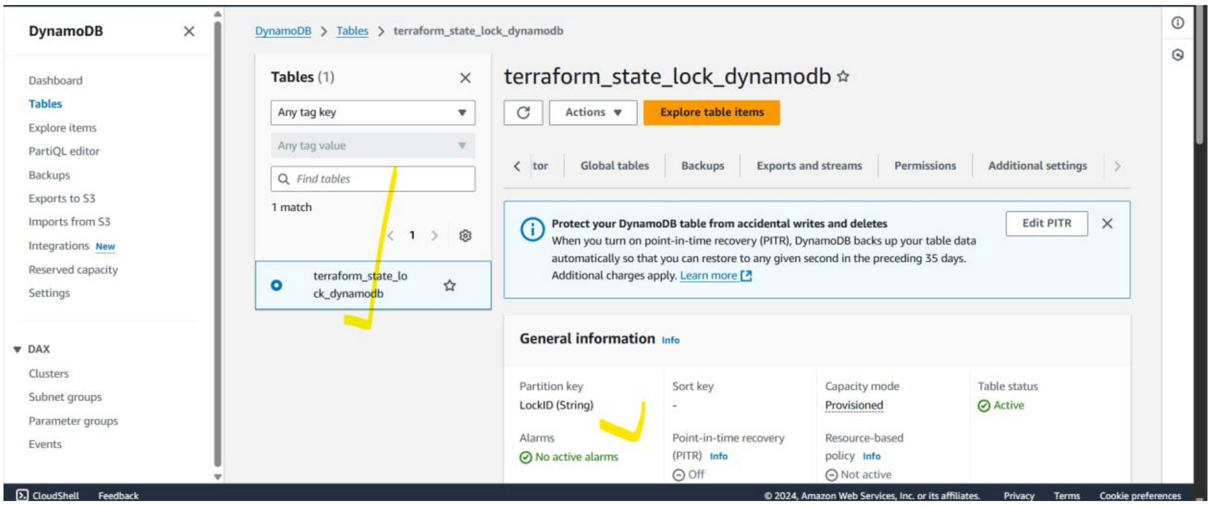
Objects (0)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

No objects

You don't have any objects in this bucket.

Here my dynamodb  table also created.



terraform_state_lock_dynamodb

**Protect your DynamoDB table from accidental writes and deletes**

When you turn on point-in-time recovery (PITR), DynamoDB backs up your table data automatically so that you can restore to any given second in the preceding 35 days. Additional charges apply. Learn more

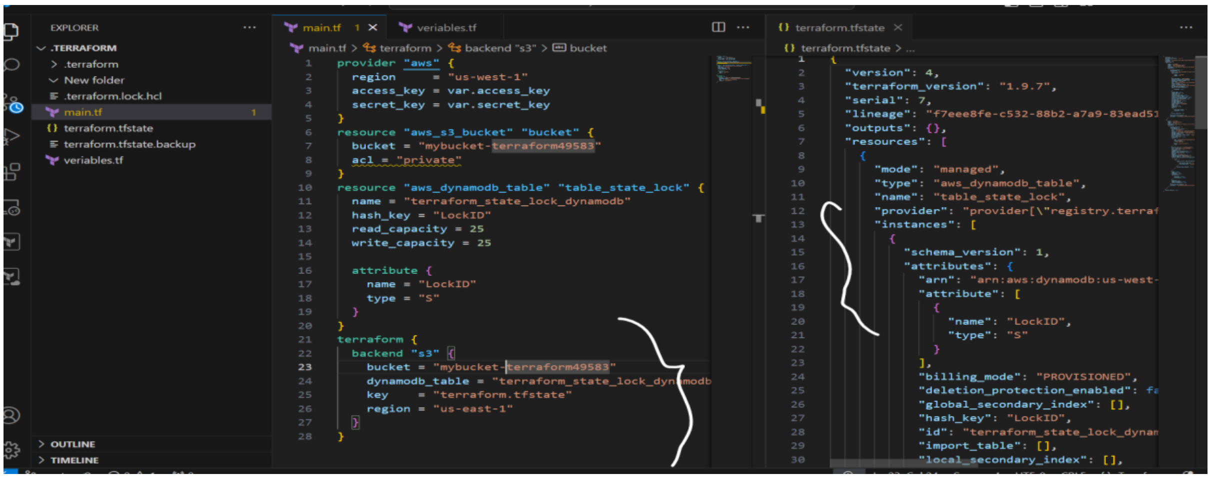
General information

| | | | |
|---|---|---|---|
| Partition key<br>LockID (String) | Sort key<br>- | Capacity mode<br>Provisioned | Table status<br>Active |
| Alarms<br>No active alarms | Point-in-time recovery (PITR)<br>Off | Resource-based policy<br>Not active | |

Now I want to setup for the backend.

To setup the backend I am written the below script.

Checks now we are able to see our state file. Once setup backend we are not able to see the state file.

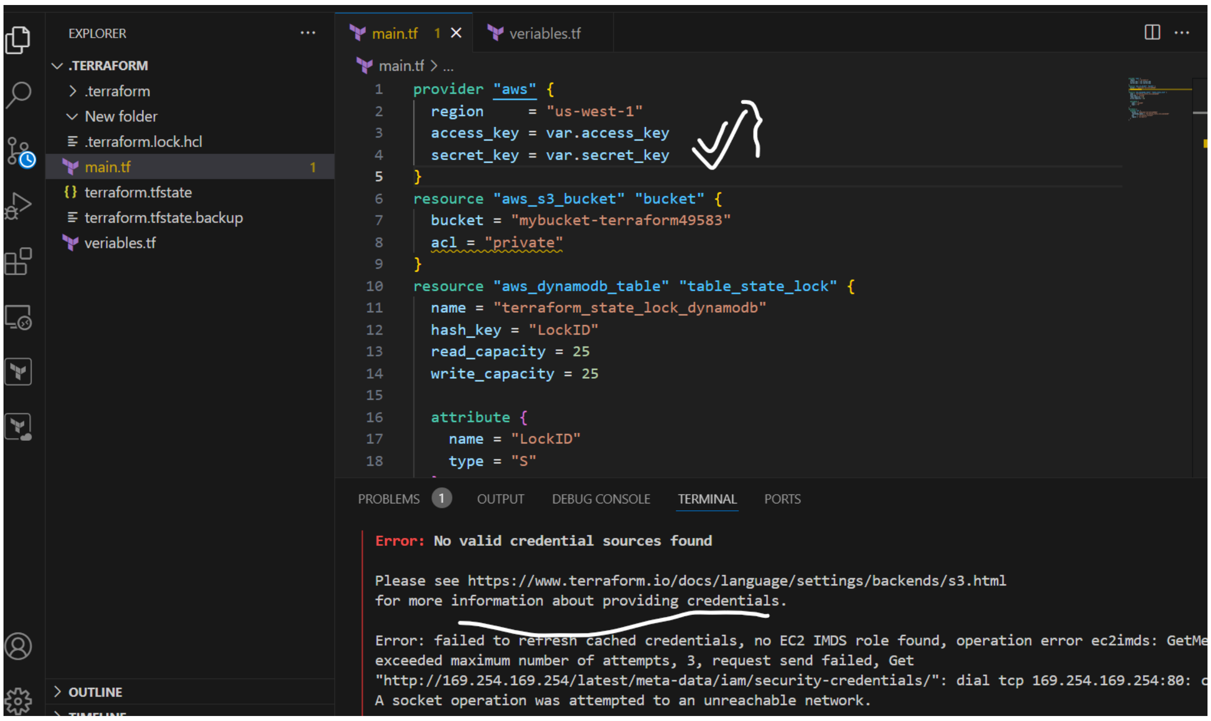


```
provider "aws" {
  region       = "us-west-1"
  access_key = var.access_key
  secret_key = var.secret_key
}
resource "aws_s3_bucket" "bucket" {
  bucket = "mybucket-terraform49583"
  acl = "private"
}
resource "aws_dynamodb_table" "table_state_lock" {
  name = "terraform_state_lock_dynamodb"
  hash_key = "LockID"
  read_capacity = 25
  write_capacity = 25

  attribute {
    name = "LockID"
    type = "S"
  }
}
terraform {
  backend "s3" {
    bucket = "mybucket-terraform49583"
    dynamodb_table = "terraform_state_lock_dynamodb
    key     = "terraform.tfstate"
    region = "us-east-1"
  }
}
```

```
{
  "version": 4,
  "terraform_version": "1.9.7",
  "serial": 7,
  "lineage": "f7eee8fe-c532-88b2-a7a9-83ead51
  "outputs": {},
  "resources": [
    {
      "mode": "managed",
      "type": "aws_dynamodb_table",
      "name": "table_state_lock",
      "provider": "provider[\"registry.terraf
      "instances": [
        {
          "schema_version": 1,
          "attributes": {
            "arn": "arn:aws:dynamodb:us-west-
            "attribute": [
              {
                "name": "LockID",
                "type": "S"
              }
            ],
            "billing_mode": "PROVISIONED",
            "deletion_protection_enabled": fa
            "global_secondary_index": [],
            "hash_key": "LockID",
            "id": "terraform_state_lock_dynam
            "import_table": [],
            "local_secondary_index": [],
```

**Issue:** Even though AWS credentials are specified in the main.tf file, Terraform does not use them for accessing the S3 backend.

**Solution:** To resolve this issue, you need to set the AWS credentials as environment variables in PowerShell using the commands: $env:AWS_ACCESS_KEY_ID = "your-access-key" and $env:AWS_SECRET_ACCESS_KEY = "your-secret-key". This allows Terraform to authenticate properly when interacting with the S3 bucket.

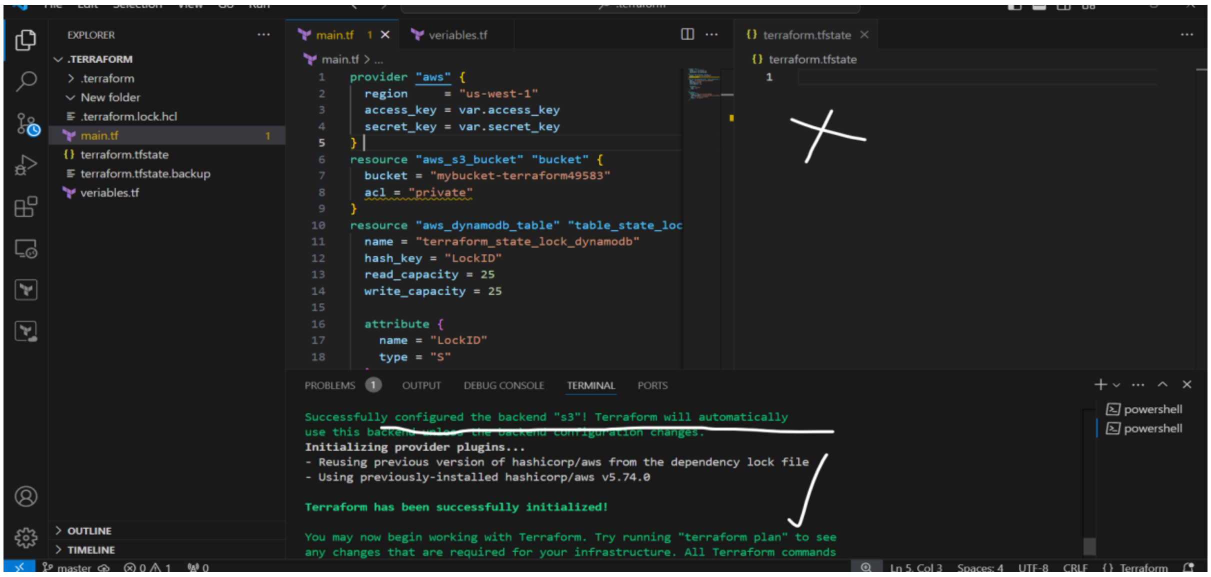


```
provider "aws" {
  region     = "us-west-1"
  access_key = var.access_key
  secret_key = var.secret_key
}
resource "aws_s3_bucket" "bucket" {
  bucket = "mybucket-terraform49583"
  acl = "private"
}
resource "aws_dynamodb_table" "table_state_lock" {
  name = "terraform_state_lock_dynamodb"
  hash_key = "LockID"
  read_capacity = 25
  write_capacity = 25

  attribute {
    name = "LockID"
    type = "S"
```

```
Error: No valid credential sources found

Please see https://www.terraform.io/docs/language/settings/backends/s3.html
for more information about providing credentials.

Error: failed to refresh cached credentials, no EC2 IMDS role found, operation error ec2imds: GetMe
exceeded maximum number of attempts, 3, request send failed, Get
"http://169.254.169.254/latest/meta-data/iam/security-credentials/": dial tcp 169.254.169.254:80: c
A socket operation was attempted to an unreachable network.
```

After successfully configured the backend s3  then we are not able to see the state file.

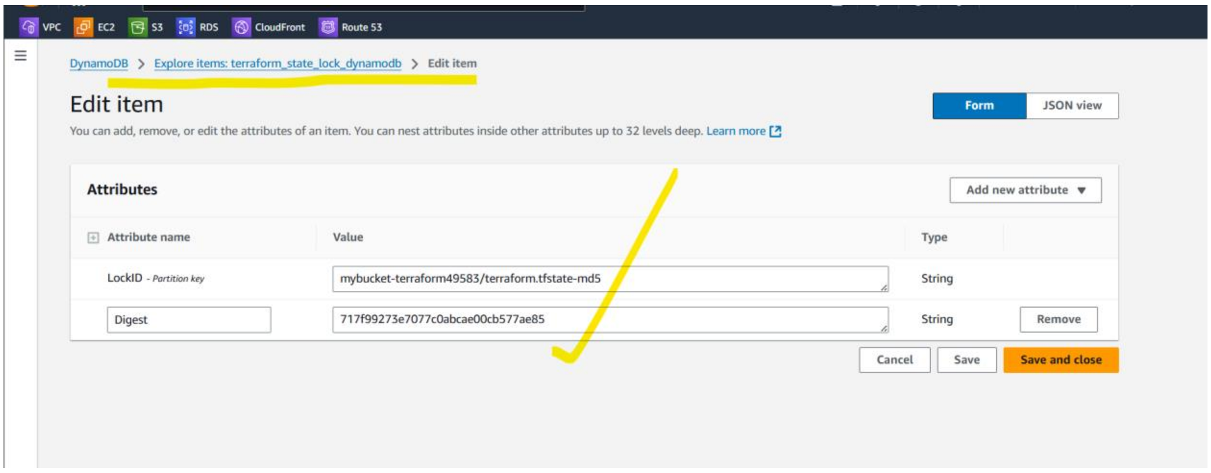
It will store in aws s3.



```
provider "aws" {
  region     = "us-west-1"
  access_key = var.access_key
  secret_key = var.secret_key
}
resource "aws_s3_bucket" "bucket" {
  bucket = "mybucket-terraform49583"
  acl = "private"
}
resource "aws_dynamodb_table" "table_state_loc
  name = "terraform_state_lock_dynamodb"
  hash_key = "LockID"
  read_capacity = 25
  write_capacity = 25

  attribute {
    name = "LockID"
    type = "S"
```

```
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 v5.74.0

Terraform has been successfully initialized!

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

Here created the one lockID.



Edit item

You can add, remove, or edit the attributes of an item. You can nest attributes inside other attributes up to 32 levels deep. Learn more

Attributes

| Attribute name | Value | Type |
|---|---|---|
| LockID - Partition key | mybucket-terraform49583/terraform.tfstate-md5 | String |
| Digest | 717f99273e7077c0abcae00cb577ae85 | String |

Now we have to create resources one by one.

## 1) <u>Create VPC</u>

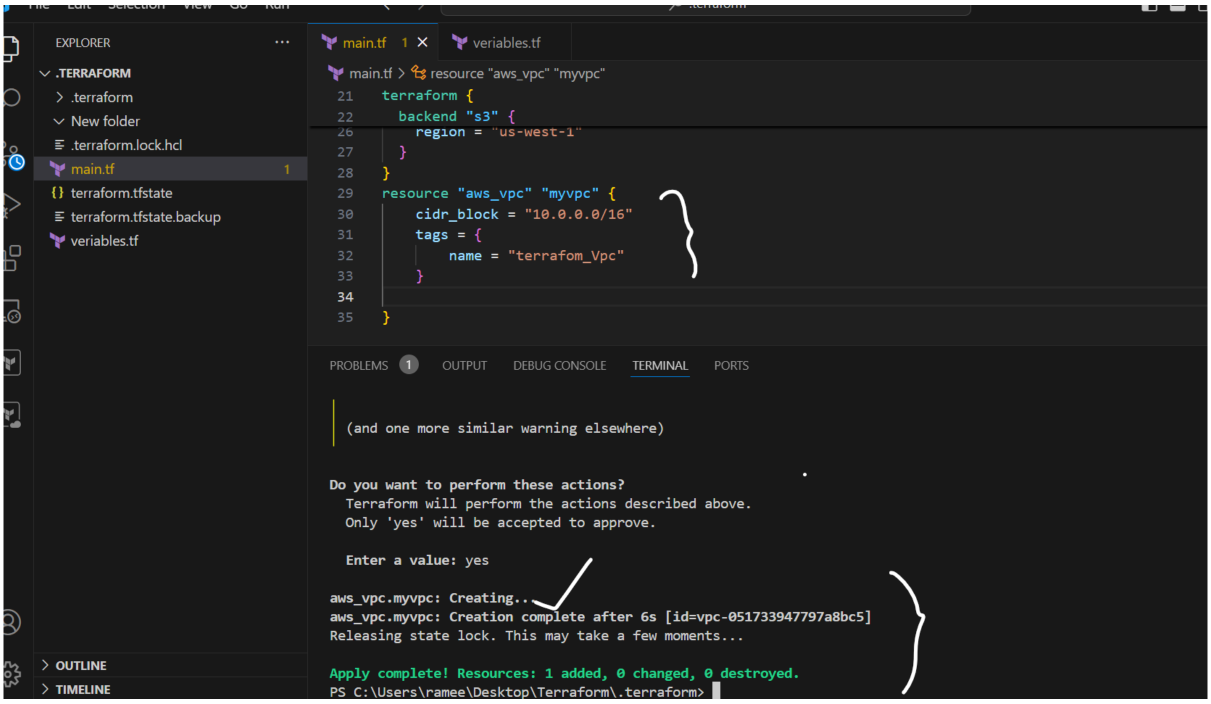
In main.tf file write script to create the vpc

Then you need to perform this command like

CMD: terraform init

CMD: terraform plan

CMD: terraform apply

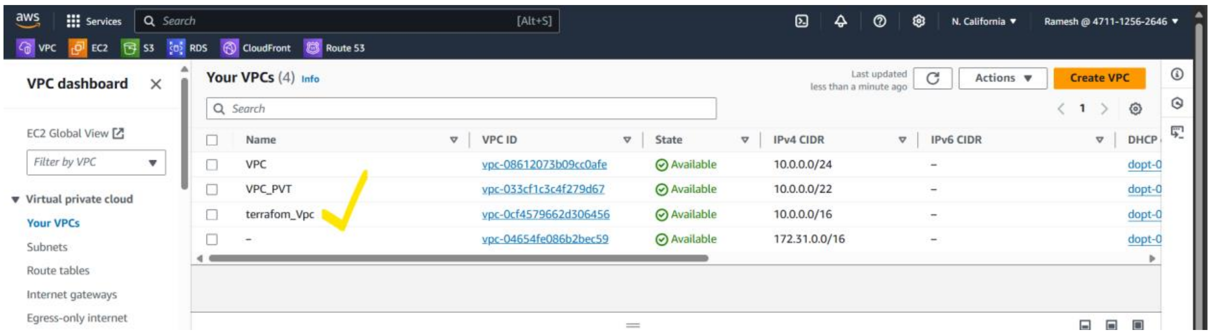


```
terraform {
  backend "s3" {
    region = "us-west-1"
  }
}
resource "aws_vpc" "myvpc" {
    cidr_block = "10.0.0.0/16"
    tags = {
        name = "terrafom_Vpc"
    }

}
```

```
(and one more similar warning elsewhere)

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

  Enter a value: yes

aws_vpc.myvpc: Creating...
aws_vpc.myvpc: Creation complete after 6s [id=vpc-051733947797a8bc5]
Releasing state lock. This may take a few moments...

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\ramee\Desktop\Terraform\.terraform>
```

Now we need to check in  Aws Vpc is created or not.

Here our vpc is available.



Your VPCs (4)

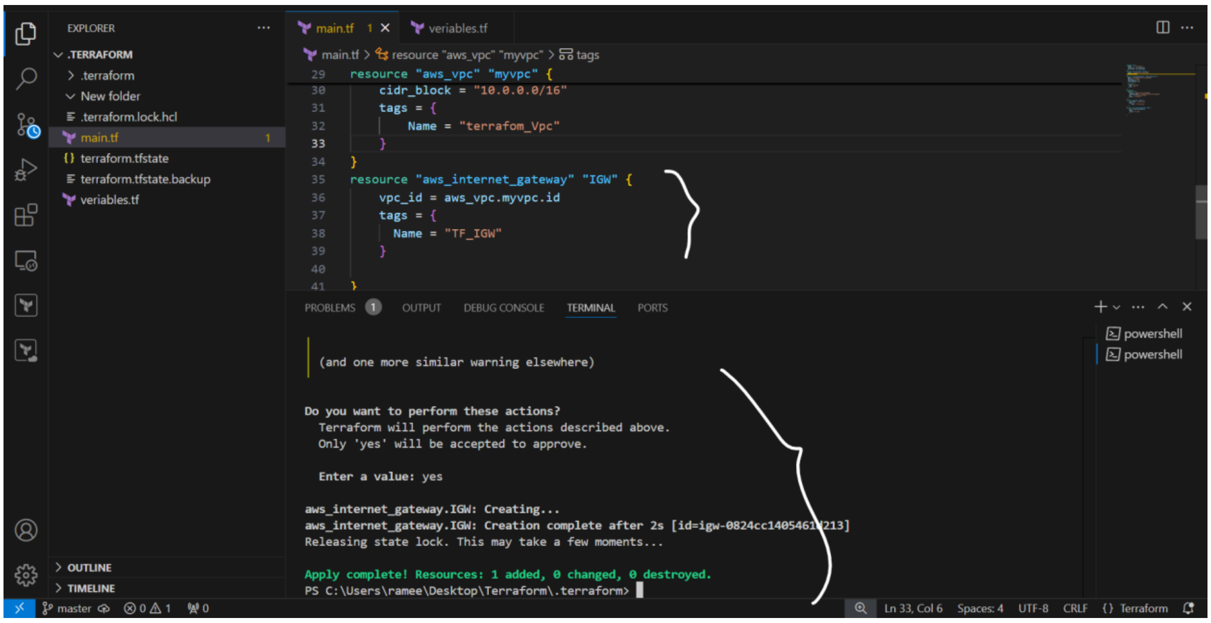
| Name | VPC ID | State | IPv4 CIDR | IPv6 CIDR | DHCP |
|---|---|---|---|---|---|
| VPC | vpc-08612073b09cc0afe | Available | 10.0.0.0/24 | – | dopt-0 |
| VPC_PVT | vpc-033cf1c3c4f279d67 | Available | 10.0.0.0/22 | – | dopt-0 |
| terrafom_Vpc | vpc-0cf4579662d306456 | Available | 10.0.0.0/16 | – | dopt-0 |
| – | vpc-04654fe086b2bec59 | Available | 172.31.0.0/16 | – | dopt-0 |

## <u>Create Internet gateway</u>

In main.tf file write script to create internet gateway.

Then you need to perform this commands like

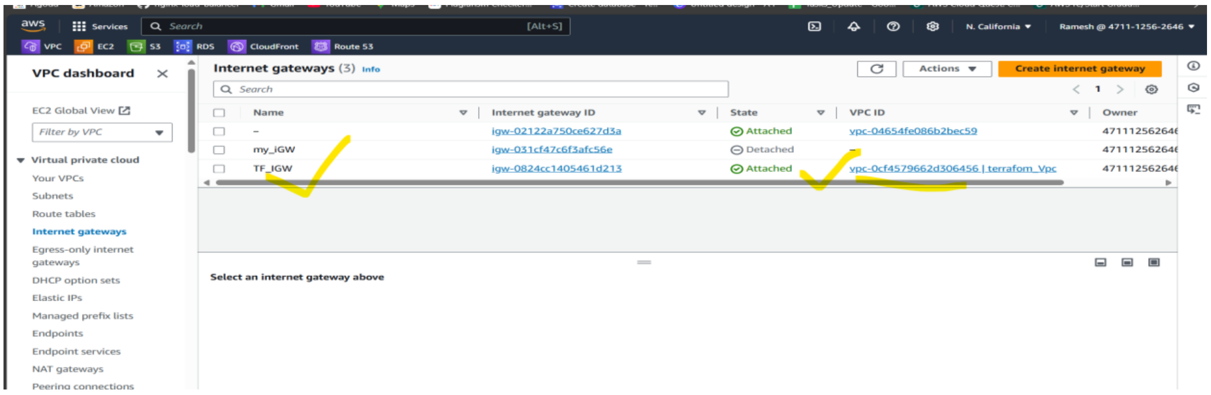
CMD: terraform init, terraform plan,terraform apply



```
resource "aws_vpc" "myvpc" {
    cidr_block = "10.0.0.0/16"
    tags = {
        Name = "terrafom_Vpc"
    }
}
resource "aws_internet_gateway" "IGW" {
    vpc_id = aws_vpc.myvpc.id
    tags = {
      Name = "TF_IGW"
    }

}
```

```
│ (and one more similar warning elsewhere)

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

  Enter a value: yes

aws_internet_gateway.IGW: Creating...
aws_internet_gateway.IGW: Creation complete after 2s [id=igw-0824cc1405461d213]
Releasing state lock. This may take a few moments...

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\ramee\Desktop\Terraform\.terraform>
```

Here created internet gateway and also attached with our Vpc.



Internet gateways (3)

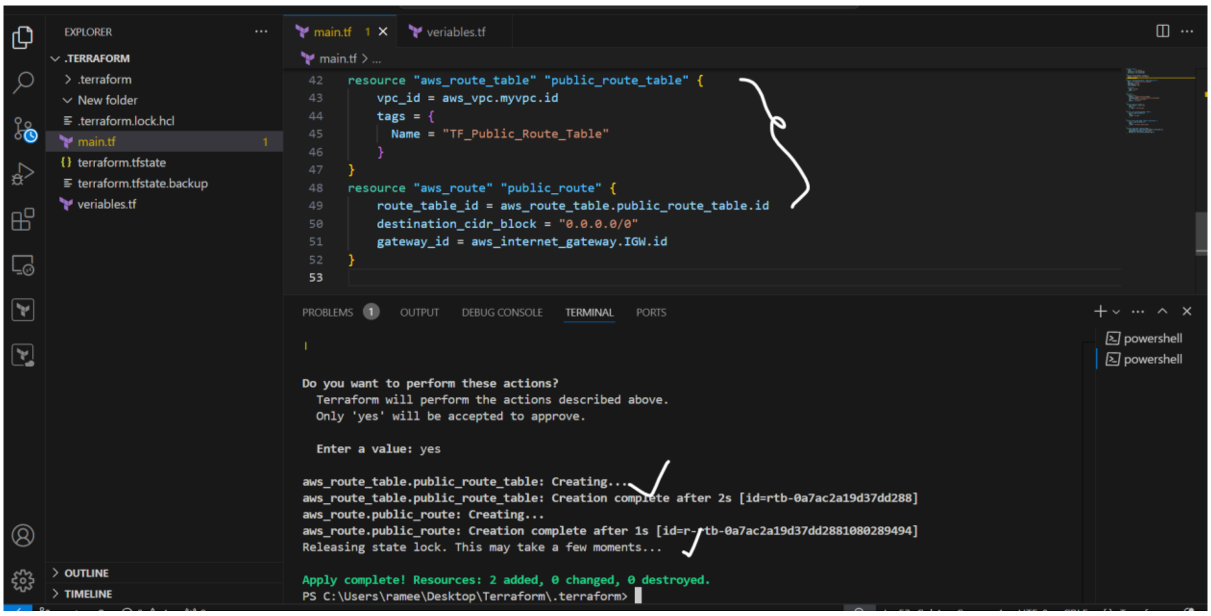
| Name | Internet gateway ID | State | VPC ID | Owner |
|---|---|---|---|---|
| – | igw-02122a750ce627d3a | Attached | vpc-04654fe086b2bec59 | 47111256264 |
| my_IGW | igw-031cf47c6f3afc56e | Detached | – | 47111256264 |
| TF_IGW | igw-0824cc1405461d213 | Attached | vpc-0cf4579662d306456 \| terrafom_Vpc | 47111256264 |

Select an internet gateway above

## Create Custom Route Table

Here I am writed the script for create route table and attached the internet gateway to route table.

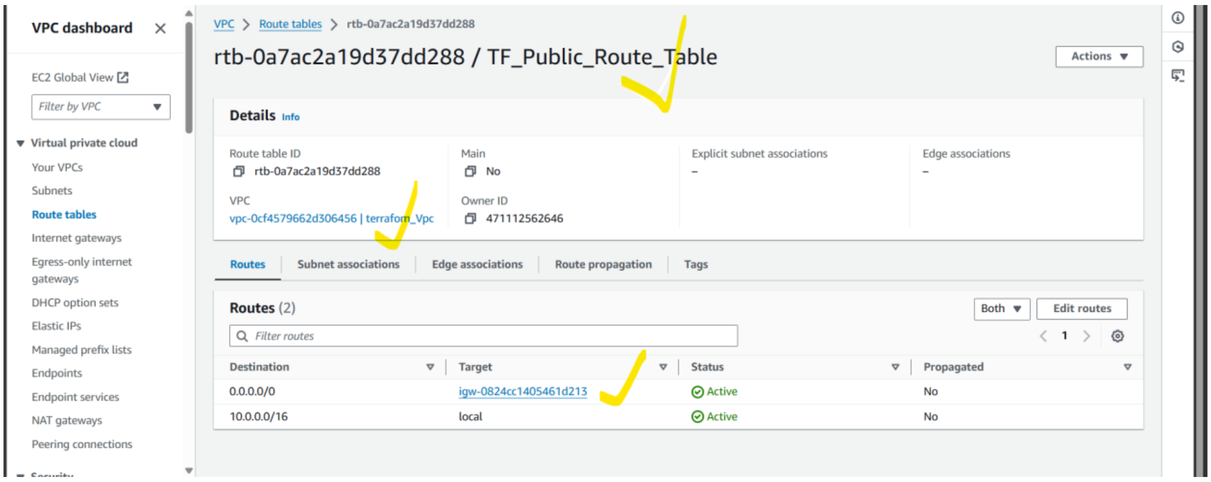
After you need to perform the three commands.



```
resource "aws_route_table" "public_route_table" {
    vpc_id = aws_vpc.myvpc.id
    tags = {
      Name = "TF_Public_Route_Table"
    }
}
resource "aws_route" "public_route" {
    route_table_id = aws_route_table.public_route_table.id
    destination_cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.IGW.id
}
```

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

  Enter a value: yes

aws_route_table.public_route_table: Creating...
aws_route_table.public_route_table: Creation complete after 2s [id=rtb-0a7ac2a19d37dd288]
aws_route.public_route: Creating...
aws_route.public_route: Creation complete after 1s [id=r-rtb-0a7ac2a19d37dd2881080289494]
Releasing state lock. This may take a few moments...

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
PS C:\Users\ramee\Desktop\Terraform\.terraform>
```

Here our route table is created and attached internet gateway.



VPC > Route tables > rtb-0a7ac2a19d37dd288

rtb-0a7ac2a19d37dd288 / TF_Public_Route_Table

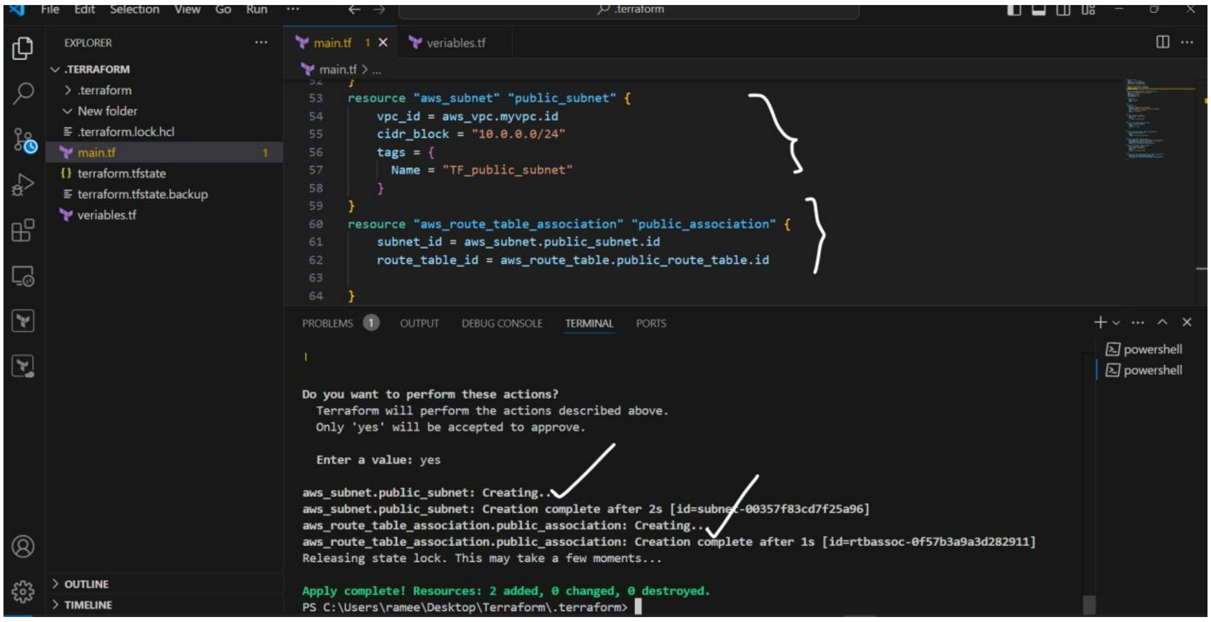
**Details**

| | | | |
|---|---|---|---|
| Route table ID | Main | Explicit subnet associations | Edge associations |
| rtb-0a7ac2a19d37dd288 | No | – | – |
| VPC | Owner ID | | |
| vpc-0cf4579662d306456 \| terrafom_Vpc | 471112562646 | | |

Routes (2)

| Destination | Target | Status | Propagated |
|---|---|---|---|
| 0.0.0.0/0 | igw-0824cc1405461d213 | Active | No |
| 10.0.0.0/16 | local | Active | No |

Here I am creating Public subnet and Public subnet associate with Route table.

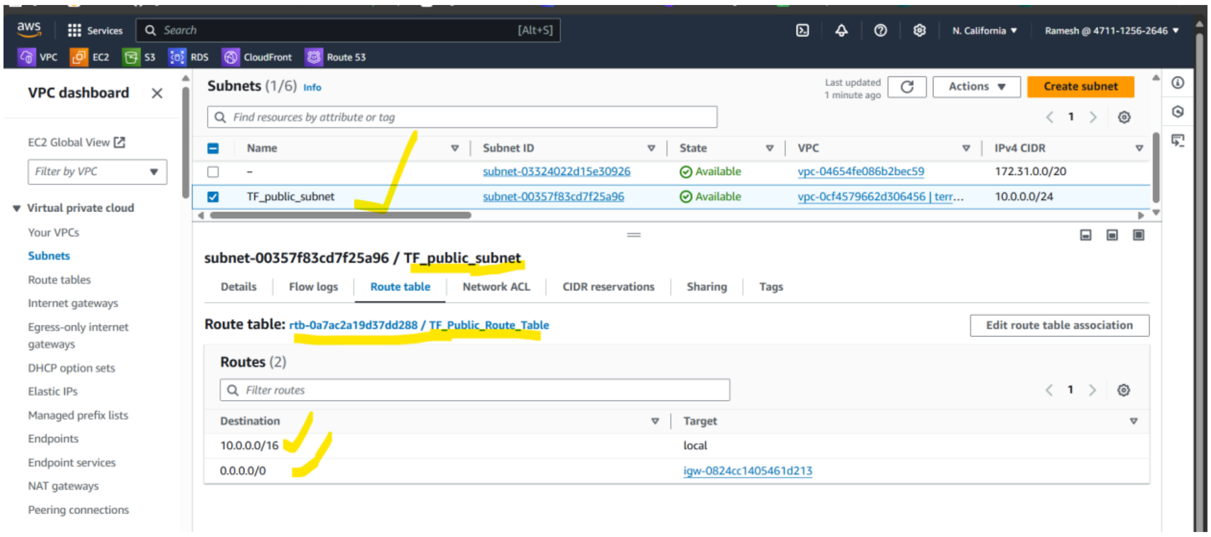


```
resource "aws_subnet" "public_subnet" {
    vpc_id = aws_vpc.myvpc.id
    cidr_block = "10.0.0.0/24"
    tags = {
      Name = "TF_public_subnet"
    }
}
resource "aws_route_table_association" "public_association" {
    subnet_id = aws_subnet.public_subnet.id
    route_table_id = aws_route_table.public_route_table.id

}
```

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

  Enter a value: yes

aws_subnet.public_subnet: Creating...
aws_subnet.public_subnet: Creation complete after 2s [id=subnet-00357f83cd7f25a96]
aws_route_table_association.public_association: Creating...
aws_route_table_association.public_association: Creation complete after 1s [id=rtbassoc-0f57b3a9a3d282911]
Releasing state lock. This may take a few moments...

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
PS C:\Users\ramee\Desktop\Terraform\.terraform>
```

Here our public subnet associate with public route table.

Now it’s Done.



Subnets (1/6)

| | Name | Subnet ID | State | VPC | IPv4 CIDR |
|---|---|---|---|---|---|
| ☐ | - | subnet-03324022d15e30926 | Available | vpc-04654fe086b2bec59 | 172.31.0.0/20 |
| ☑ | TF_public_subnet | subnet-00357f83cd7f25a96 | Available | vpc-0cf4579662d306456 \| terr... | 10.0.0.0/24 |

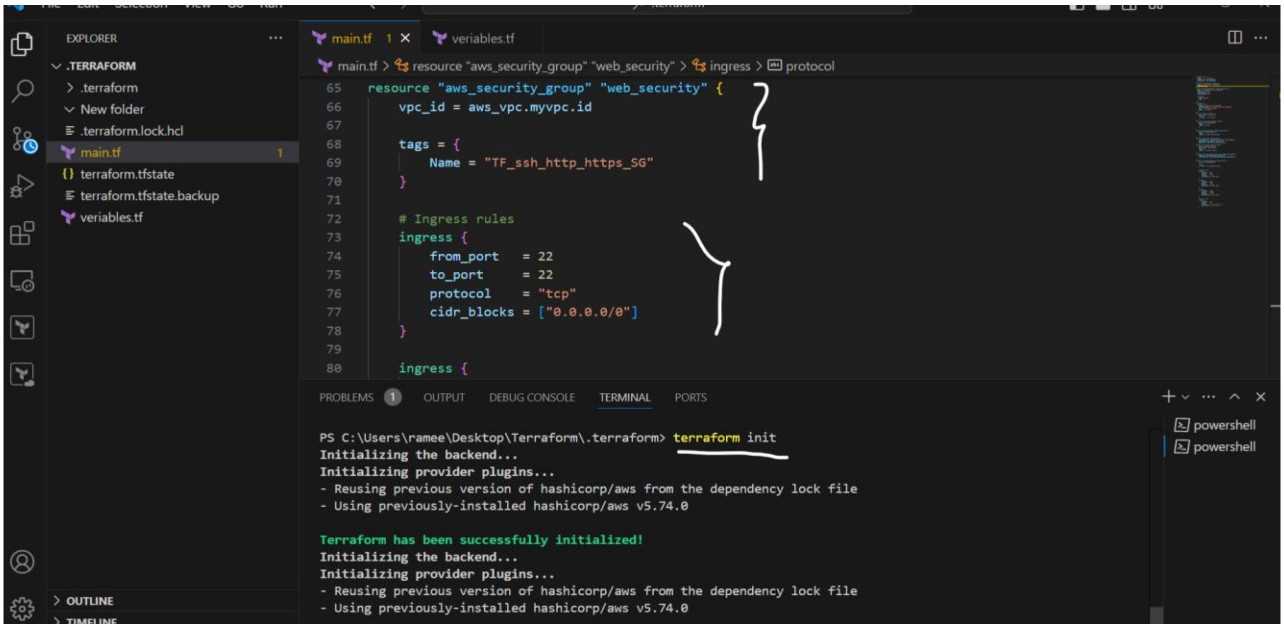
subnet-00357f83cd7f25a96 / TF_public_subnet

Route table: rtb-0a7ac2a19d37dd288 / TF_Public_Route_Table

Routes (2)

| Destination | Target |
|---|---|
| 10.0.0.0/16 | local |
| 0.0.0.0/0 | igw-0824cc1405461d213 |

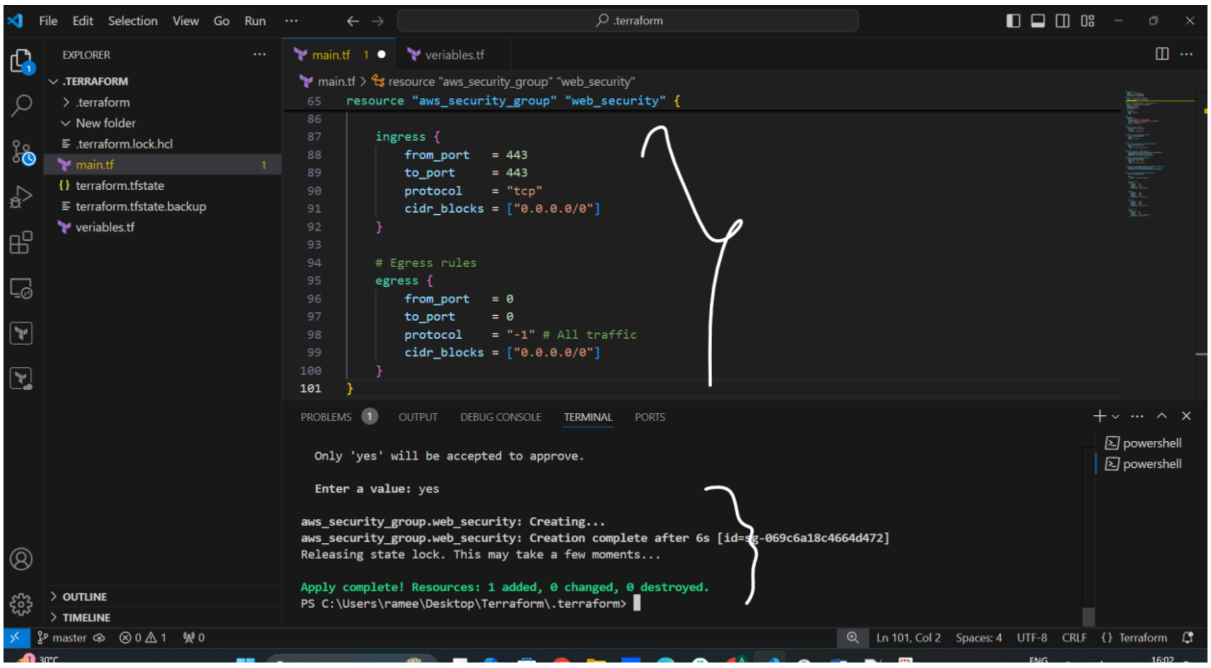
## Here I am write script for security group and added the rules.



```
resource "aws_security_group" "web_security" {
    vpc_id = aws_vpc.myvpc.id

    tags = {
        Name = "TF_ssh_http_https_SG"
    }

    # Ingress rules
    ingress {
        from_port   = 22
        to_port     = 22
        protocol    = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }

    ingress {
```

```
PS C:\Users\ramee\Desktop\Terraform\.terraform> terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.74.0

Terraform has been successfully initialized!
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.74.0
```

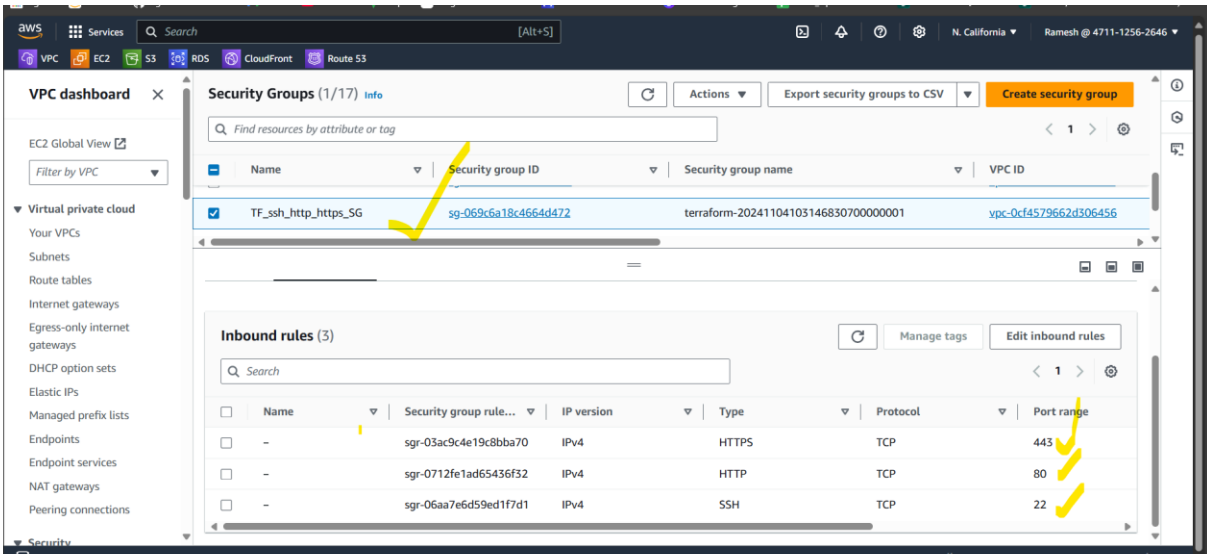
## The apply is successful.



```
resource "aws_security_group" "web_security" {

    ingress {
        from_port   = 443
        to_port     = 443
        protocol    = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }

    # Egress rules
    egress {
        from_port   = 0
        to_port     = 0
        protocol    = "-1" # All traffic
        cidr_blocks = ["0.0.0.0/0"]
    }
}
```

```
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_security_group.web_security: Creating...
aws_security_group.web_security: Creation complete after 6s [id=sg-069c6a18c4664d472]
Releasing state lock. This may take a few moments...

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\ramee\Desktop\Terraform\.terraform>
```

Here our security group created with inbound rules 443, 80, 22.



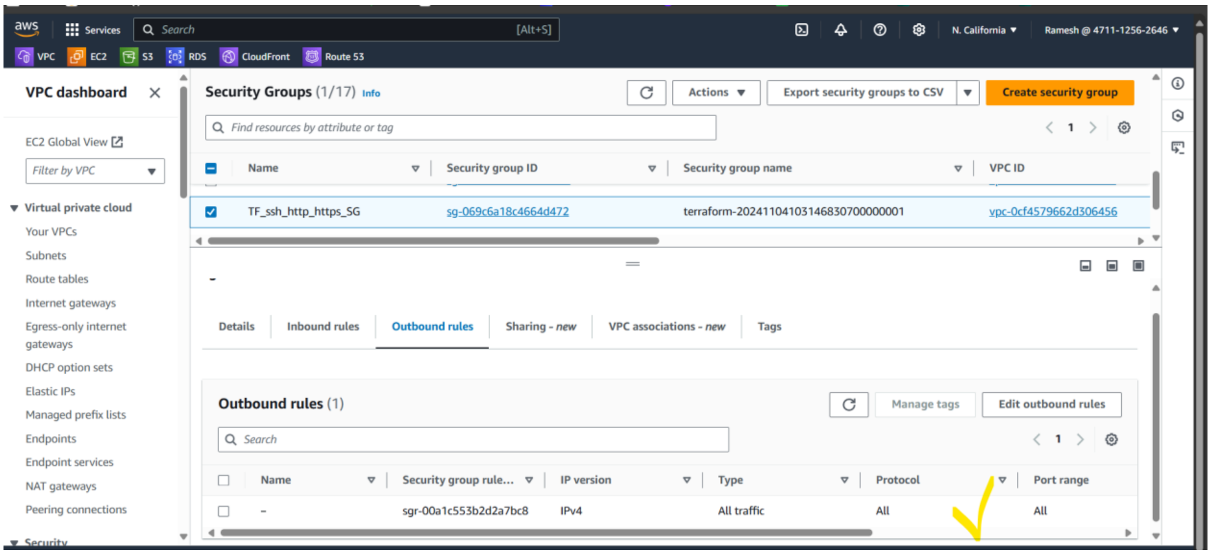
Security Groups (1/17) Info

| Name | Security group ID | Security group name | VPC ID |
|---|---|---|---|
| TF_ssh_http_https_SG | sg-069c6a18c4664d472 | terraform-20241104103146830700000001 | vpc-0cf4579662d306456 |

Inbound rules (3)

| Name | Security group rule... | IP version | Type | Protocol | Port range |
|---|---|---|---|---|---|
| - | sgr-03ac9c4e19c8bba70 | IPv4 | HTTPS | TCP | 443 |
| - | sgr-0712fe1ad65436f32 | IPv4 | HTTP | TCP | 80 |
| - | sgr-06aa7e6d59ed1f7d1 | IPv4 | SSH | TCP | 22 |

**Here outbound rule also created.**



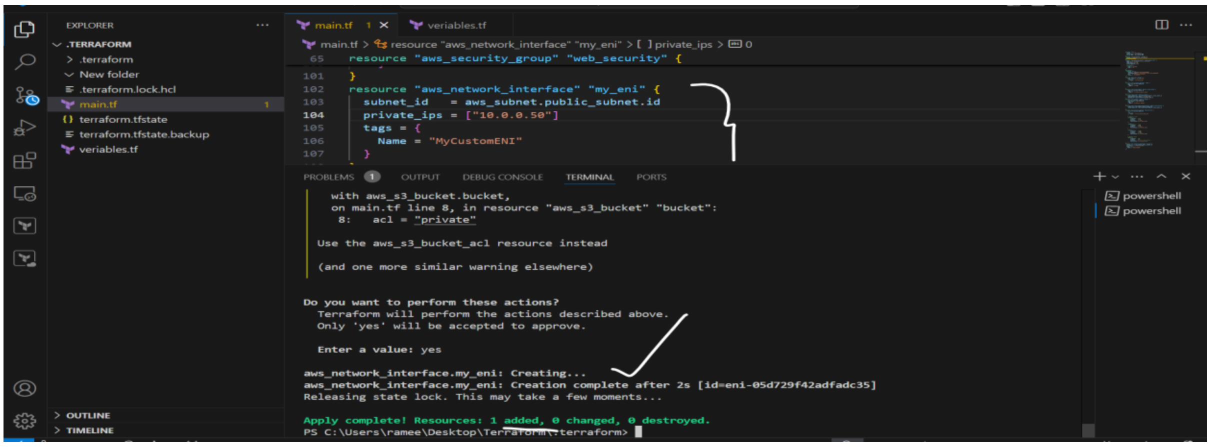
Security Groups (1/17) Info

| Name | Security group ID | Security group name | VPC ID |
|---|---|---|---|
| TF_ssh_http_https_SG | sg-069c6a18c4664d472 | terraform-20241104103146830700000001 | vpc-0cf4579662d306456 |

Outbound rules (1)

| Name | Security group rule... | IP version | Type | Protocol | Port range |
|---|---|---|---|---|---|
| - | sgr-00a1c553b2d2a7bc8 | IPv4 | All traffic | All | All |

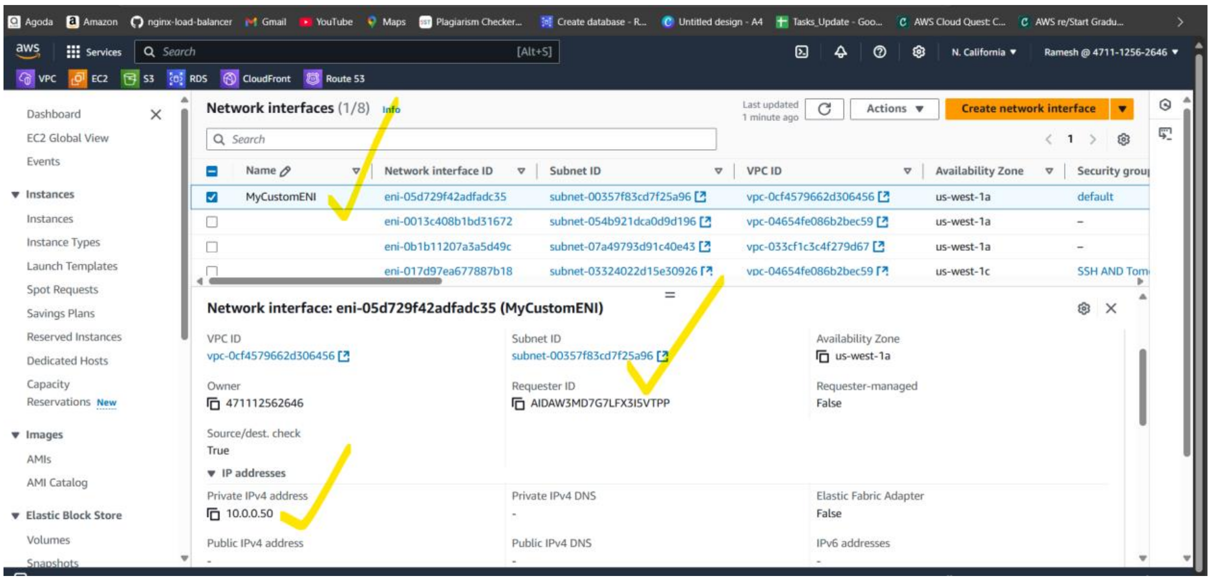
**Here we creating the interface id.**



```
resource "aws_security_group" "web_security" {
}
resource "aws_network_interface" "my_eni" {
  subnet_id    = aws_subnet.public_subnet.id
  private_ips = ["10.0.0.50"]
  tags = {
    Name = "MyCustomENI"
  }
```

```
   with aws_s3_bucket.bucket,
   on main.tf line 8, in resource "aws_s3_bucket" "bucket":
    8:   acl = "private"

  Use the aws_s3_bucket_acl resource instead

  (and one more similar warning elsewhere)

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

  Enter a value: yes

aws_network_interface.my_eni: Creating...
aws_network_interface.my_eni: Creation complete after 2s [id=eni-05d729f42adfadc35]
Releasing state lock. This may take a few moments...

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\ramee\Desktop\Terraform\.terraform>
```

## Here  check the details of network interface.



**Network interfaces** (1/8)

| | Name | Network interface ID | Subnet ID | VPC ID | Availability Zone | Security group |
|---|---|---|---|---|---|---|
| ✓ | MyCustomENI | eni-05d729f42adfadc35 | subnet-00357f83cd7f25a96 | vpc-0cf4579662d306456 | us-west-1a | default |
| | | eni-0013c408b1bd31672 | subnet-054b921dca0d9d196 | vpc-04654fe086b2bec59 | us-west-1a | – |
| | | eni-0b1b11207a3a5d49c | subnet-07a49793d91c40e43 | vpc-033cf1c3c4f279d67 | us-west-1a | – |
| | | eni-017d97ea677887b18 | subnet-03324022d15e30926 | vpc-04654fe086b2bec59 | us-west-1c | SSH AND Tom |

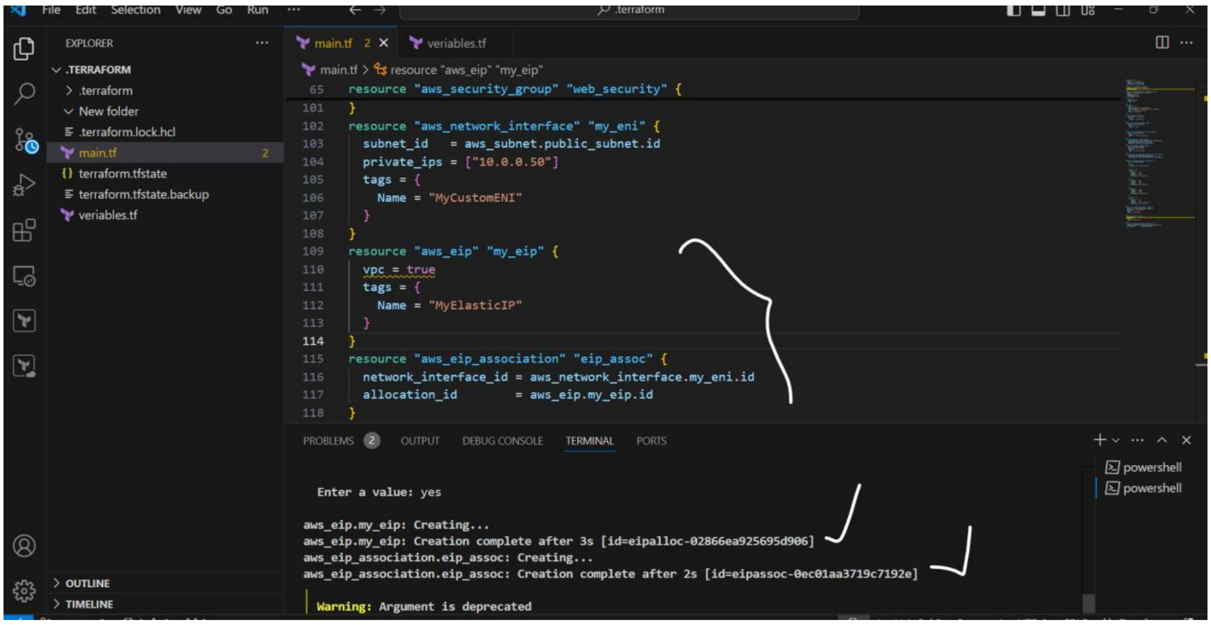
**Network interface: eni-05d729f42adfadc35 (MyCustomENI)**

| | | |
|---|---|---|
| VPC ID<br>vpc-0cf4579662d306456 | Subnet ID<br>subnet-00357f83cd7f25a96 | Availability Zone<br>us-west-1a |
| Owner<br>471112562646 | Requester ID<br>AIDAW3MD7G7LFX3I5VTPP | Requester-managed<br>False |
| Source/dest. check<br>True | | |

IP addresses

| | | |
|---|---|---|
| Private IPv4 address<br>10.0.0.50 | Private IPv4 DNS<br>- | Elastic Fabric Adapter<br>False |
| Public IPv4 address<br>- | Public IPv4 DNS<br>- | IPv6 addresses<br>- |

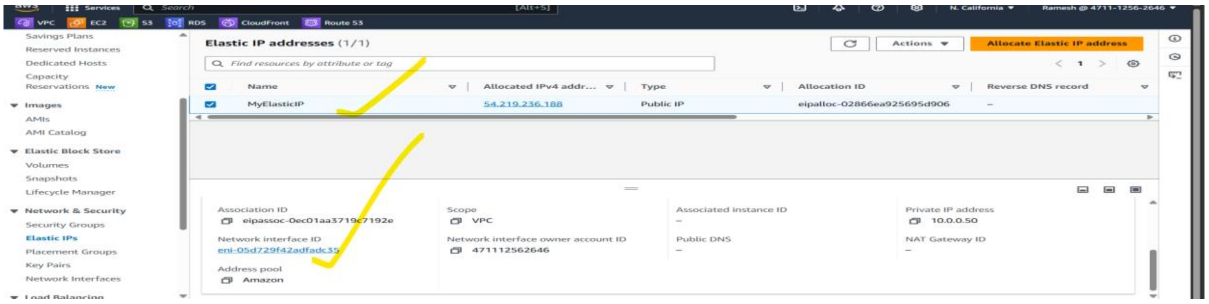
## Here I am creating the elastic ip and attach with network interface.



```
main.tf > resource "aws_eip" "my_eip"
resource "aws_security_group" "web_security" {
}
resource "aws_network_interface" "my_eni" {
  subnet_id    = aws_subnet.public_subnet.id
  private_ips = ["10.0.0.50"]
  tags = {
    Name = "MyCustomENI"
  }
}
resource "aws_eip" "my_eip" {
  vpc = true
  tags = {
    Name = "MyElasticIP"
  }
}
resource "aws_eip_association" "eip_assoc" {
  network_interface_id = aws_network_interface.my_eni.id
  allocation_id        = aws_eip.my_eip.id
}
```

```
  Enter a value: yes

aws_eip.my_eip: Creating...
aws_eip.my_eip: Creation complete after 3s [id=eipalloc-02866ea925695d906]
aws_eip_association.eip_assoc: Creating...
aws_eip_association.eip_assoc: Creation complete after 2s [id=eipassoc-0ec01aa3719c7192e]

Warning: Argument is deprecated
```

## The below you can check it’s attached or not.

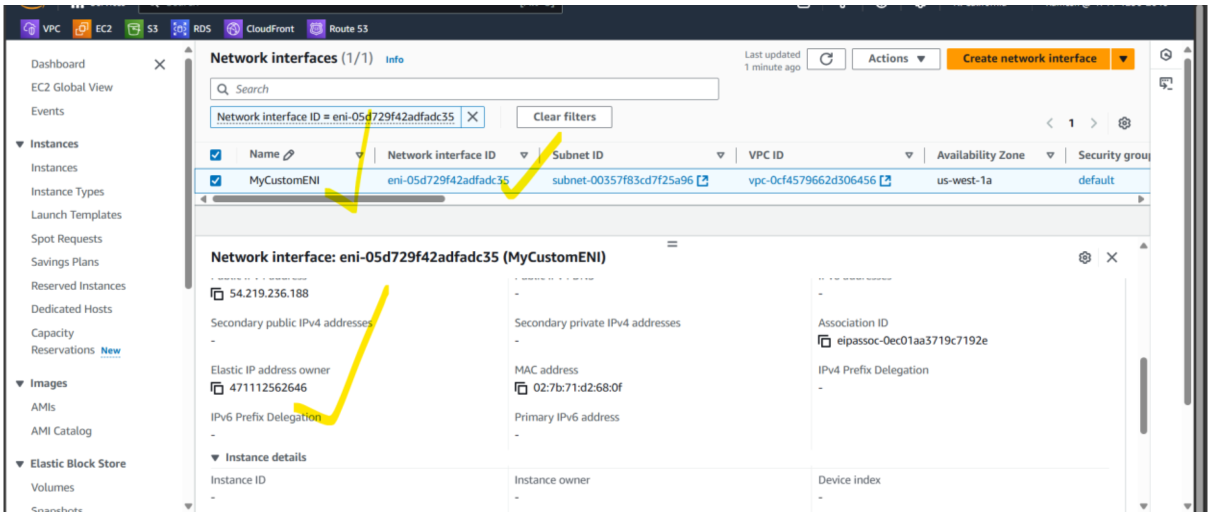


**Elastic IP addresses** (1/1)

| | Name | Allocated IPv4 addr... | Type | Allocation ID | Reverse DNS record |
|---|---|---|---|---|---|
| ✓ | MyElasticIP | 54.219.236.188 | Public IP | eipalloc-02866ea925695d906 | – |

| | | | |
|---|---|---|---|
| Association ID<br>eipassoc-0ec01aa3719c7192e | Scope<br>VPC | Associated instance ID<br>– | Private IP address<br>10.0.0.50 |
| Network interface ID<br>eni-05d729f42adfadc35 | Network interface owner account ID<br>471112562646 | Public DNS<br>– | NAT Gateway ID<br>– |
| Address pool<br>Amazon | | | |

## Here see the details.



Network interfaces (1/1)

Network interface ID = eni-05d729f42adfadc35

| Name | Network interface ID | Subnet ID | VPC ID | Availability Zone | Security group |
|---|---|---|---|---|---|
| MyCustomENI | eni-05d729f42adfadc35 | subnet-00357f83cd7f25a96 | vpc-0cf4579662d306456 | us-west-1a | default |

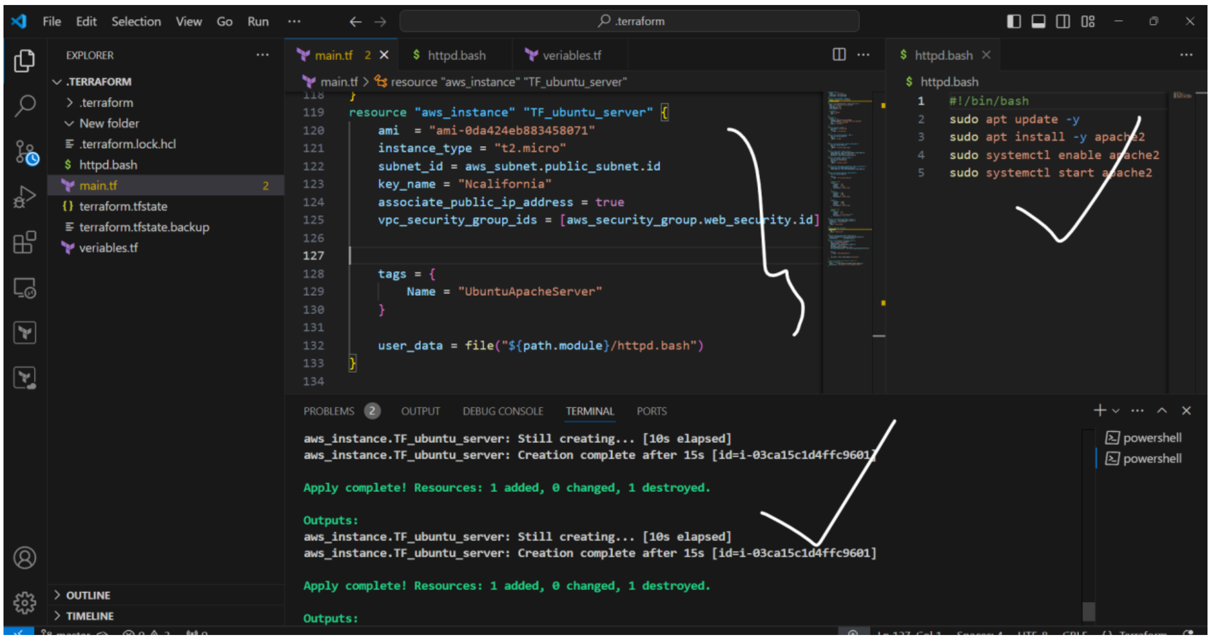
Network interface: eni-05d729f42adfadc35 (MyCustomENI)

| | | |
|---|---|---|
| 54.219.236.188 | - | - |
| Secondary public IPv4 addresses<br>- | Secondary private IPv4 addresses<br>- | Association ID<br>eipassoc-0ec01aa3719c7192e |
| Elastic IP address owner<br>471112562646 | MAC address<br>02:7b:71:d2:68:0f | IPv4 Prefix Delegation<br>- |
| IPv6 Prefix Delegation<br>- | Primary IPv6 address<br>- | |

Instance details

| Instance ID | Instance owner | Device index |
|---|---|---|
| - | - | - |

## Here I am writing the script for launch Ubuntu instance with appache2 .



main.tf > resource "aws_instance" "TF_ubuntu_server"

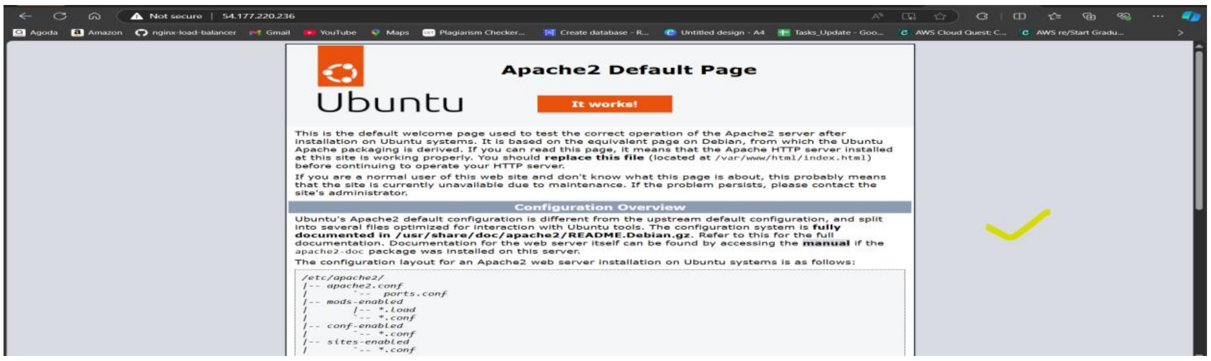
```
resource "aws_instance" "TF_ubuntu_server" {
    ami  = "ami-0da424eb883458071"
    instance_type = "t2.micro"
    subnet_id = aws_subnet.public_subnet.id
    key_name = "Ncalifornia"
    associate_public_ip_address = true
    vpc_security_group_ids = [aws_security_group.web_security.id]


    tags = {
        Name = "UbuntuApacheServer"
    }

    user_data = file("${path.module}/httpd.bash")
}
```

httpd.bash

```
#!/bin/bash
sudo apt update -y
sudo apt install -y apache2
sudo systemctl enable apache2
sudo systemctl start apache2
```

```
aws_instance.TF_ubuntu_server: Still creating... [10s elapsed]
aws_instance.TF_ubuntu_server: Creation complete after 15s [id=i-03ca15c1d4ffc9601]

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

Outputs:
aws_instance.TF_ubuntu_server: Still creating... [10s elapsed]
aws_instance.TF_ubuntu_server: Creation complete after 15s [id=i-03ca15c1d4ffc9601]

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

Outputs:
```

## Here successfully we are seen apache2 server.



Not secure | 54.177.220.236

Apache2 Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at /var/www/html/index.html) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the apache2-doc package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

**The below script:**

```
provider "aws" {
  region     = "us-west-1"
  access_key = var.access_key
  secret_key = var.secret_key
}
resource "aws_s3_bucket" "bucket" {
  bucket = "mybucket-terraform49583"
  acl = "private"
}
resource "aws_dynamodb_table" "table_state_lock" {
  name = "terraform_state_lock_dynamodb"
  hash_key = "LockID"
  read_capacity = 25
  write_capacity = 25

  attribute {
    name = "LockID"
    type = "S"
  }
}
terraform {
  backend "s3" {
    bucket = "mybucket-terraform49583"
     dynamodb_table = "terraform_state_lock_dynamodb"
    key    = "terraform.tfstate"
    region = "us-west-1"
  }
}
resource "aws_vpc" "myvpc" {
    cidr_block = "10.0.0.0/16"
    tags = {
        Name = "terrafom_Vpc"
    }
}
resource "aws_internet_gateway" "IGW" {
    vpc_id = aws_vpc.myvpc.id
    tags = {
      Name = "TF_IGW"
    }

}
resource "aws_route_table" "public_route_table" {
    vpc_id = aws_vpc.myvpc.id
    tags = {
      Name = "TF_Public_Route_Table"
    }
```

```
}
resource "aws_route" "public_route" {
    route_table_id = aws_route_table.public_route_table.id
    destination_cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.IGW.id
}
resource "aws_subnet" "public_subnet" {
    vpc_id = aws_vpc.myvpc.id
    cidr_block = "10.0.0.0/24"
    tags = {
      Name = "TF_public_subnet"
    }
}
resource "aws_route_table_association" "public_association" {
    subnet_id = aws_subnet.public_subnet.id
    route_table_id = aws_route_table.public_route_table.id

}
resource "aws_security_group" "web_security" {
    vpc_id = aws_vpc.myvpc.id

    tags = {
        Name = "TF_ssh_http_https_SG"
    }

    # Ingress rules
    ingress {
        from_port   = 22
        to_port     = 22
        protocol    = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }

    ingress {
        from_port   = 80
        to_port     = 80
        protocol    = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }

    ingress {
        from_port   = 443
        to_port     = 443
        protocol    = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }

    # Egress rules
    egress {
        from_port   = 0
        to_port     = 0
        protocol    = "-1" # All traffic
        cidr_blocks = ["0.0.0.0/0"]
    }
}
resource "aws_network_interface" "my_eni" {
  subnet_id    = aws_subnet.public_subnet.id
  private_ips = ["10.0.0.50"]
  tags = {
    Name = "MyCustomENI"
  }
}
resource "aws_eip" "my_eip" {
  vpc = true
  tags = {
    Name = "MyElasticIP"
  }
}
resource "aws_eip_association" "eip_assoc" {
  network_interface_id = aws_network_interface.my_eni.id
  allocation_id        = aws_eip.my_eip.id
}
resource "aws_instance" "TF_ubuntu_server" {
    ami  = "ami-0da424eb883458071"
    instance_type = "t2.micro"
    subnet_id = aws_subnet.public_subnet.id
    key_name = "Ncalifornia"
    associate_public_ip_address = true
    vpc_security_group_ids = [aws_security_group.web_security.id]


    tags = {
        Name = "UbuntuApacheServer"
    }

    user_data = file("${path.module}/httpd.bash")
}

# Output to retrieve the instance's public IP
output "instance_ip" {
  value       = aws_instance.TF_ubuntu_server.public_ip
  description = "The public IP of the Ubuntu server"
}
```