Terraform Assignment – 2
Deploying AWS resources
BY Swapnil Jadhav

A&B is a Leading Business Services & Solution Provider company in the market, for one of their clients is struggling with the infrastructure provisioning and they would like to automate their infrastructure provisioning with the help of Terraform.
You are requested to setup the entire infrastructure using a Terraform Configuration. Following resources need to be deployed:

1. Network Setup

Create a VPC Create an internet gateway

Create a custom Route Table
Create a Subnet
Associate the Subnet with the Route Table

2. Security Group Setup
Create a new security group
Enable ports 22, 80, 443

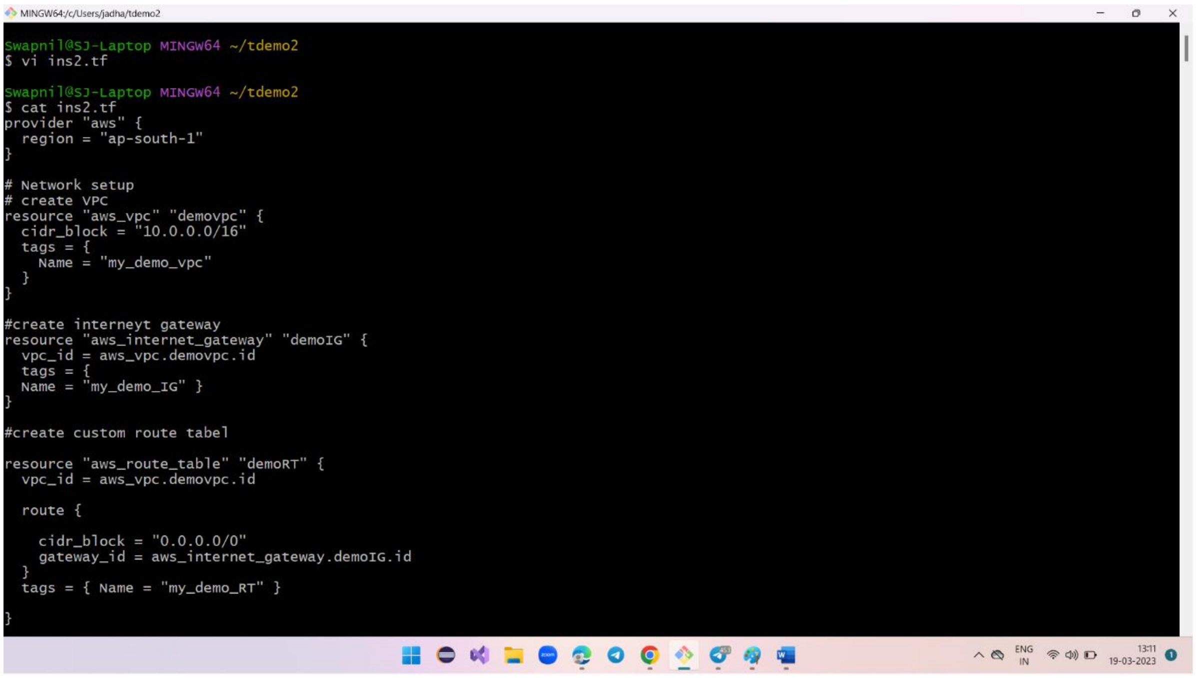
3. Network Interface Setup
Create a new network interface with IP in the previously created subnet
Create an elastic IP associated with the network interface

4. Ec2 instance setup

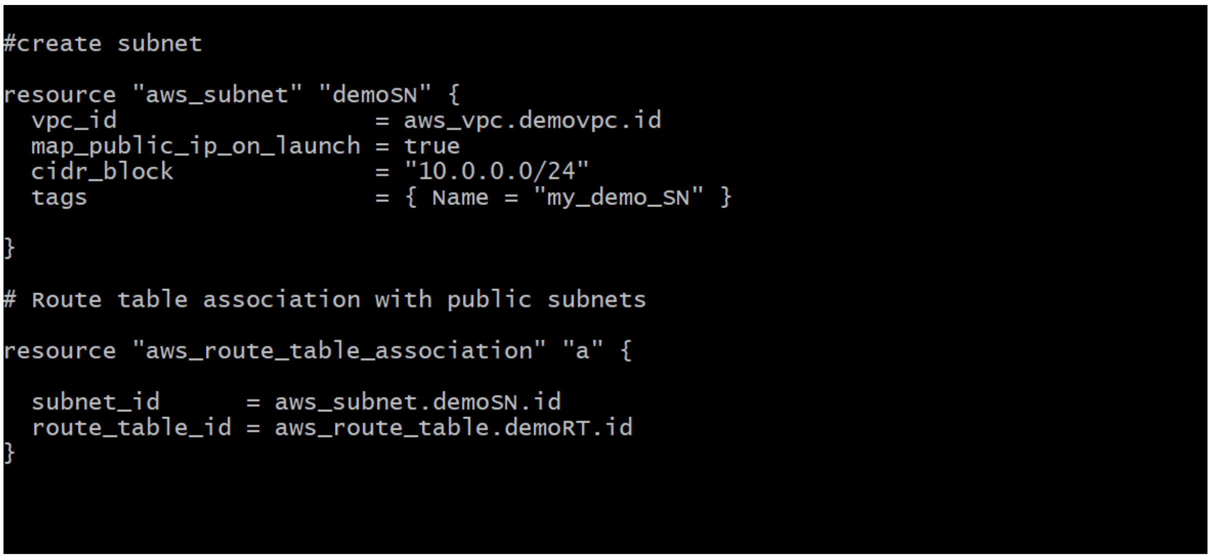
Create a new ubuntu ec2 instance and attach the network interface to it

Install httpd server on it

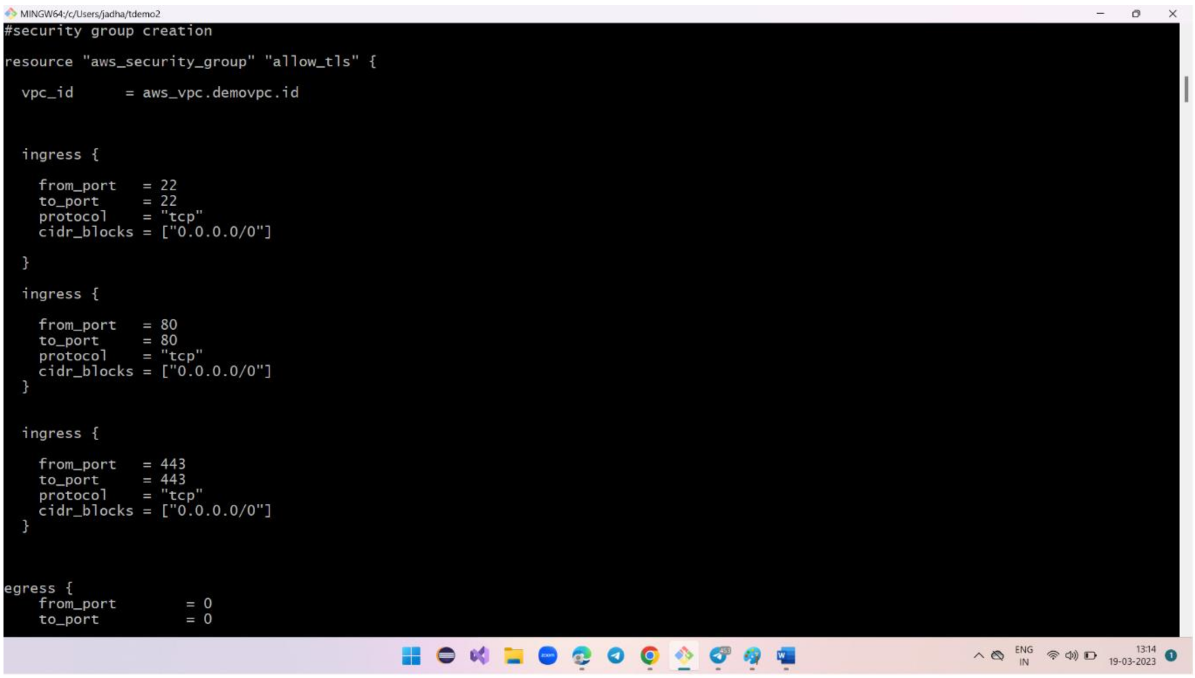
All Configuration code has been taken from:
https://registry.terraform.io/providers/hashicorp/aws/latest/docs



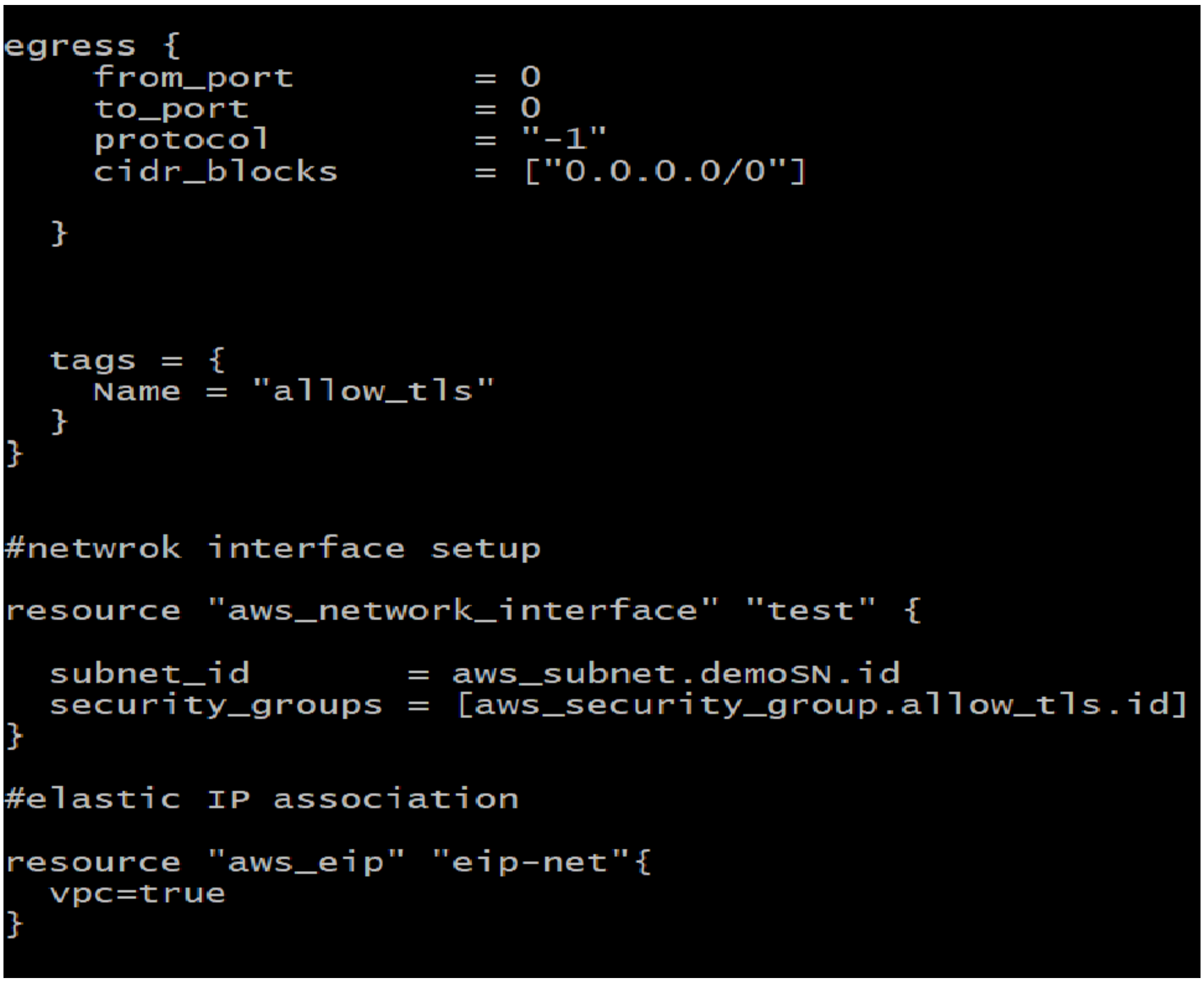
```
Swapnil@SJ-Laptop MINGW64 ~/tdemo2
$ vi ins2.tf

Swapnil@SJ-Laptop MINGW64 ~/tdemo2
$ cat ins2.tf
provider "aws" {
  region = "ap-south-1"
}

# Network setup
# create VPC
resource "aws_vpc" "demovpc" {
  cidr_block = "10.0.0.0/16"
  tags = {
    Name = "my_demo_vpc"
  }
}

#create interneyt gateway
resource "aws_internet_gateway" "demoIG" {
  vpc_id = aws_vpc.demovpc.id
  tags = {
  Name = "my_demo_IG" }
}

#create custom route tabel

resource "aws_route_table" "demoRT" {
  vpc_id = aws_vpc.demovpc.id

  route {

    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.demoIG.id
  }
  tags = { Name = "my_demo_RT" }

}
```



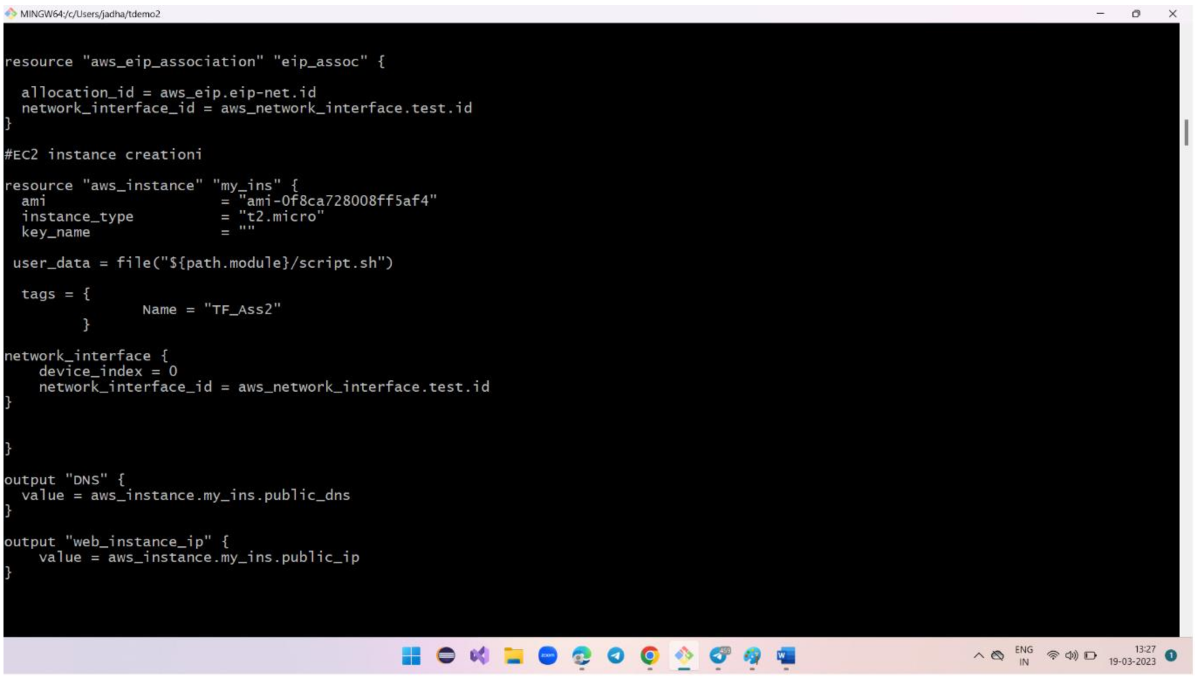
```
#create subnet

resource "aws_subnet" "demoSN" {
  vpc_id                  = aws_vpc.demovpc.id
  map_public_ip_on_launch = true
  cidr_block              = "10.0.0.0/24"
  tags                    = { Name = "my_demo_SN" }

}

# Route table association with public subnets

resource "aws_route_table_association" "a" {

  subnet_id      = aws_subnet.demoSN.id
  route_table_id = aws_route_table.demoRT.id
}
```



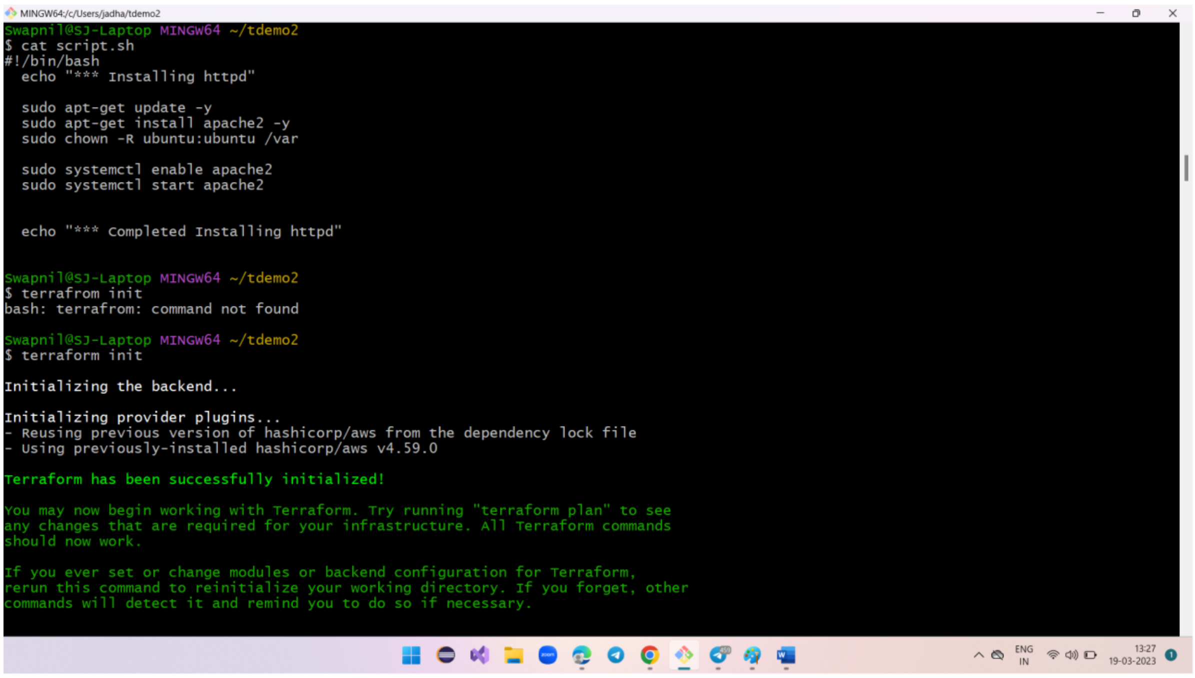
```
#security group creation

resource "aws_security_group" "allow_tls" {

  vpc_id      = aws_vpc.demovpc.id


  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 {
    from_port        = 0
    to_port          = 0
```



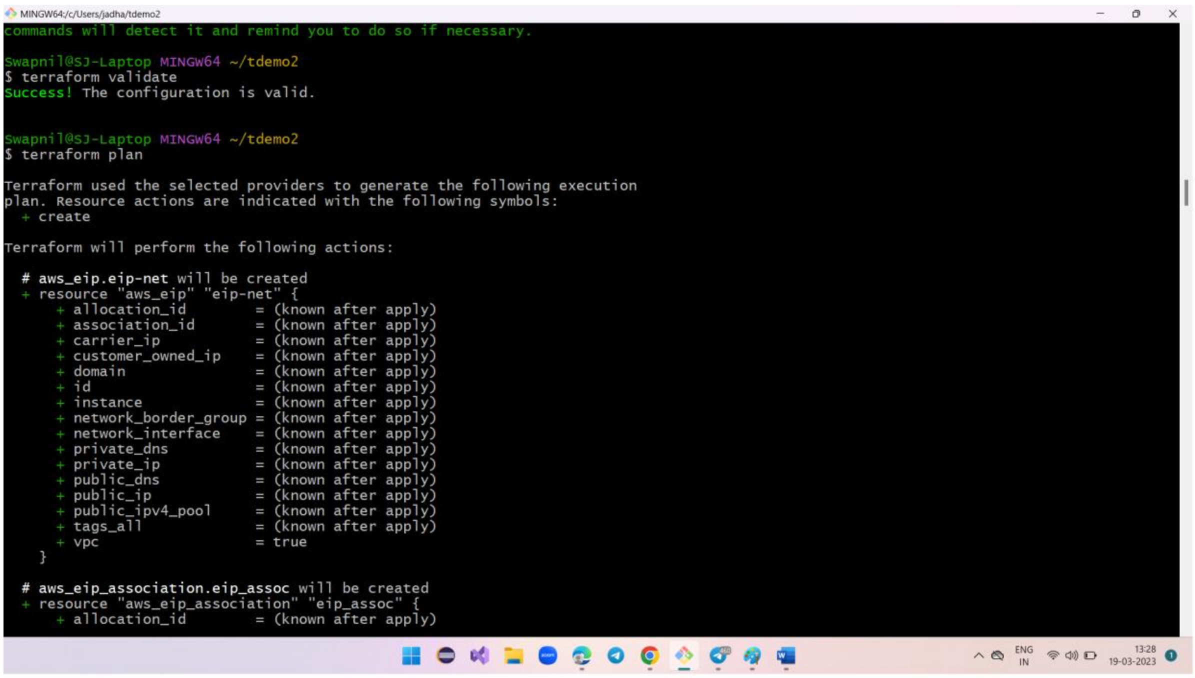
```
egress {
    from_port        = 0
    to_port          = 0
    protocol         = "-1"
    cidr_blocks      = ["0.0.0.0/0"]

  }



  tags = {
    Name = "allow_tls"
  }
}


#netwrok interface setup

resource "aws_network_interface" "test" {

  subnet_id       = aws_subnet.demoSN.id
  security_groups = [aws_security_group.allow_tls.id]
}

#elastic IP association

resource "aws_eip" "eip-net"{
  vpc=true
}
```


MINGW64:/c/Users/jadha/tdemo2

```
resource "aws_eip_association" "eip_assoc" {

  allocation_id = aws_eip.eip-net.id
  network_interface_id = aws_network_interface.test.id
}

#EC2 instance creationi

resource "aws_instance" "my_ins" {
  ami                     = "ami-0f8ca728008ff5af4"
  instance_type           = "t2.micro"
  key_name                = ""

 user_data = file("${path.module}/script.sh")

  tags = {
                Name = "TF_Ass2"
         }

network_interface {
    device_index = 0
    network_interface_id = aws_network_interface.test.id
}


}

output "DNS" {
  value = aws_instance.my_ins.public_dns
}

output "web_instance_ip" {
    value = aws_instance.my_ins.public_ip
}
```

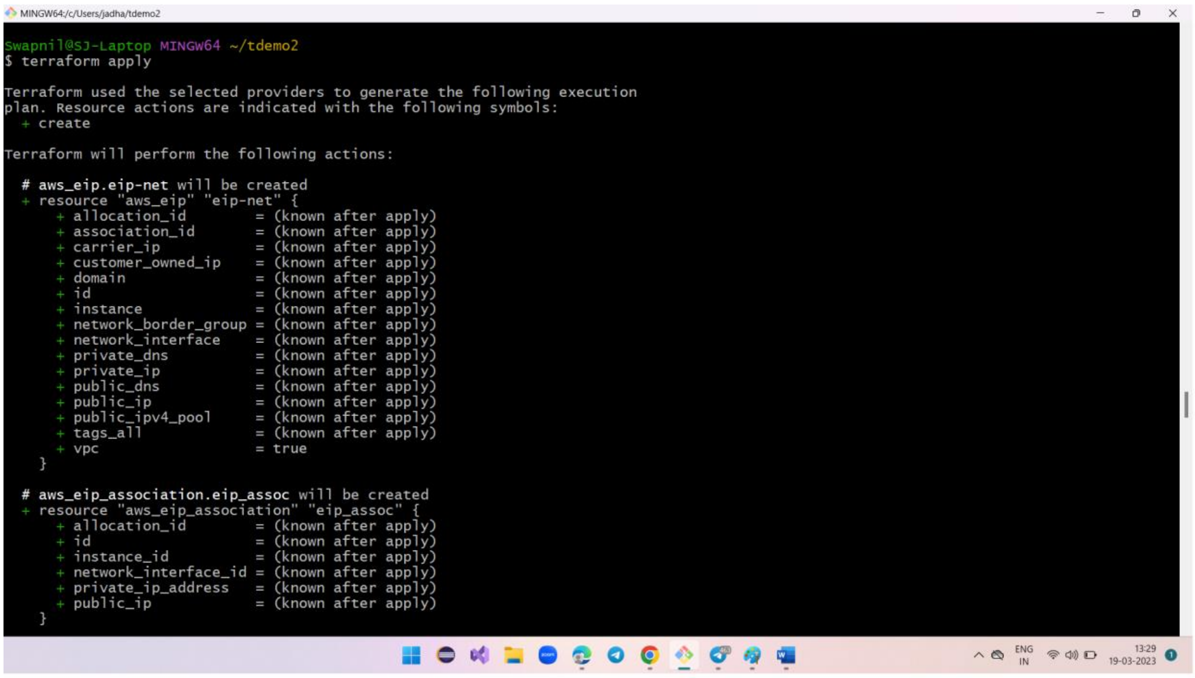

MINGW64:/c/Users/jadha/tdemo2

```
Swapnil@SJ-Laptop MINGW64 ~/tdemo2
$ cat script.sh
#!/bin/bash
  echo "*** Installing httpd"

  sudo apt-get update -y
  sudo apt-get install apache2 -y
  sudo chown -R ubuntu:ubuntu /var

  sudo systemctl enable apache2
  sudo systemctl start apache2


  echo "*** Completed Installing httpd"


Swapnil@SJ-Laptop MINGW64 ~/tdemo2
$ terrafrom init
bash: terrafrom: command not found

Swapnil@SJ-Laptop MINGW64 ~/tdemo2
$ terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.59.0

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
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```



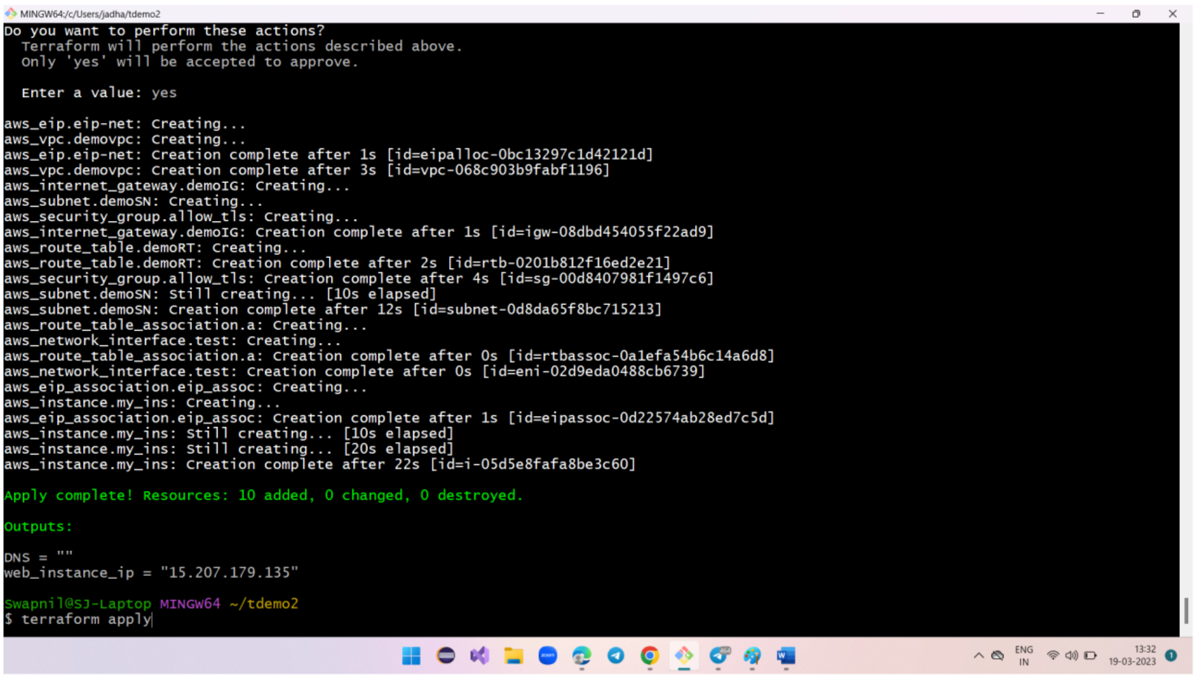
MINGW64:/c/Users/jadha/tdemo2

```
commands will detect it and remind you to do so if necessary.

Swapnil@SJ-Laptop MINGW64 ~/tdemo2
$ terraform validate
Success! The configuration is valid.


Swapnil@SJ-Laptop MINGW64 ~/tdemo2
$ terraform plan

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_eip.eip-net will be created
  + resource "aws_eip" "eip-net" {
      + allocation_id        = (known after apply)
      + association_id       = (known after apply)
      + carrier_ip           = (known after apply)
      + customer_owned_ip    = (known after apply)
      + domain               = (known after apply)
      + id                   = (known after apply)
      + instance             = (known after apply)
      + network_border_group = (known after apply)
      + network_interface    = (known after apply)
      + private_dns          = (known after apply)
      + private_ip           = (known after apply)
      + public_dns           = (known after apply)
      + public_ip            = (known after apply)
      + public_ipv4_pool     = (known after apply)
      + tags_all             = (known after apply)
      + vpc                  = true
    }

  # aws_eip_association.eip_assoc will be created
  + resource "aws_eip_association" "eip_assoc" {
      + allocation_id        = (known after apply)
```



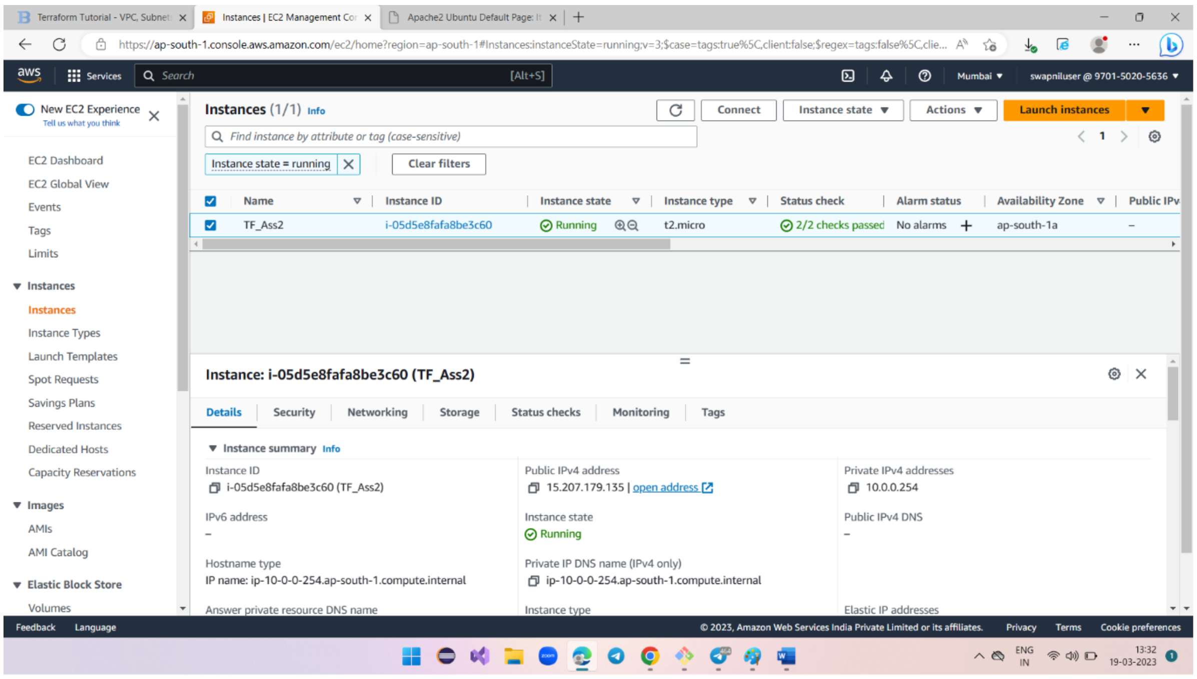
MINGW64:/c/Users/jadha/tdemo2

```
Swapnil@SJ-Laptop MINGW64 ~/tdemo2
$ terraform apply

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_eip.eip-net will be created
  + resource "aws_eip" "eip-net" {
      + allocation_id        = (known after apply)
      + association_id       = (known after apply)
      + carrier_ip           = (known after apply)
      + customer_owned_ip    = (known after apply)
      + domain               = (known after apply)
      + id                   = (known after apply)
      + instance             = (known after apply)
      + network_border_group = (known after apply)
      + network_interface    = (known after apply)
      + private_dns          = (known after apply)
      + private_ip           = (known after apply)
      + public_dns           = (known after apply)
      + public_ip            = (known after apply)
      + public_ipv4_pool     = (known after apply)
      + tags_all             = (known after apply)
      + vpc                  = true
    }

  # aws_eip_association.eip_assoc will be created
  + resource "aws_eip_association" "eip_assoc" {
      + allocation_id        = (known after apply)
      + id                   = (known after apply)
      + instance_id          = (known after apply)
      + network_interface_id = (known after apply)
      + private_ip_address   = (known after apply)
      + public_ip            = (known after apply)
    }
```



MINGW64:/c/Users/jadha/tdemo2

```
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_eip.eip-net: Creating...
aws_vpc.demovpc: Creating...
aws_eip.eip-net: Creation complete after 1s [id=eipalloc-0bc13297c1d42121d]
aws_vpc.demovpc: Creation complete after 3s [id=vpc-068c903b9fabf1196]
aws_internet_gateway.demoIG: Creating...
aws_subnet.demoSN: Creating...
aws_security_group.allow_tls: Creating...
aws_internet_gateway.demoIG: Creation complete after 1s [id=igw-08dbd454055f22ad9]
aws_route_table.demoRT: Creating...
aws_route_table.demoRT: Creation complete after 2s [id=rtb-0201b812f16ed2e21]
aws_security_group.allow_tls: Creation complete after 4s [id=sg-00d8407981f1497c6]
aws_subnet.demoSN: Still creating... [10s elapsed]
aws_subnet.demoSN: Creation complete after 12s [id=subnet-0d8da65f8bc715213]
aws_route_table_association.a: Creating...
aws_network_interface.test: Creating...
aws_route_table_association.a: Creation complete after 0s [id=rtbassoc-0a1efa54b6c14a6d8]
aws_network_interface.test: Creation complete after 0s [id=eni-02d9eda0488cb6739]
aws_eip_association.eip_assoc: Creating...
aws_instance.my_ins: Creating...
aws_eip_association.eip_assoc: Creation complete after 1s [id=eipassoc-0d22574ab28ed7c5d]
aws_instance.my_ins: Still creating... [10s elapsed]
aws_instance.my_ins: Still creating... [20s elapsed]
aws_instance.my_ins: Creation complete after 22s [id=i-05d5e8fafa8be3c60]

Apply complete! Resources: 10 added, 0 changed, 0 destroyed.

Outputs:

DNS = ""
web_instance_ip = "15.207.179.135"

Swapnil@SJ-Laptop MINGW64 ~/tdemo2
$ terraform apply
```



https://ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#Instances:instanceState=running;v=3;$case=tags:true%5C,client:false;$regex=tags:false%5C,clie...

Instances (1/1) Info

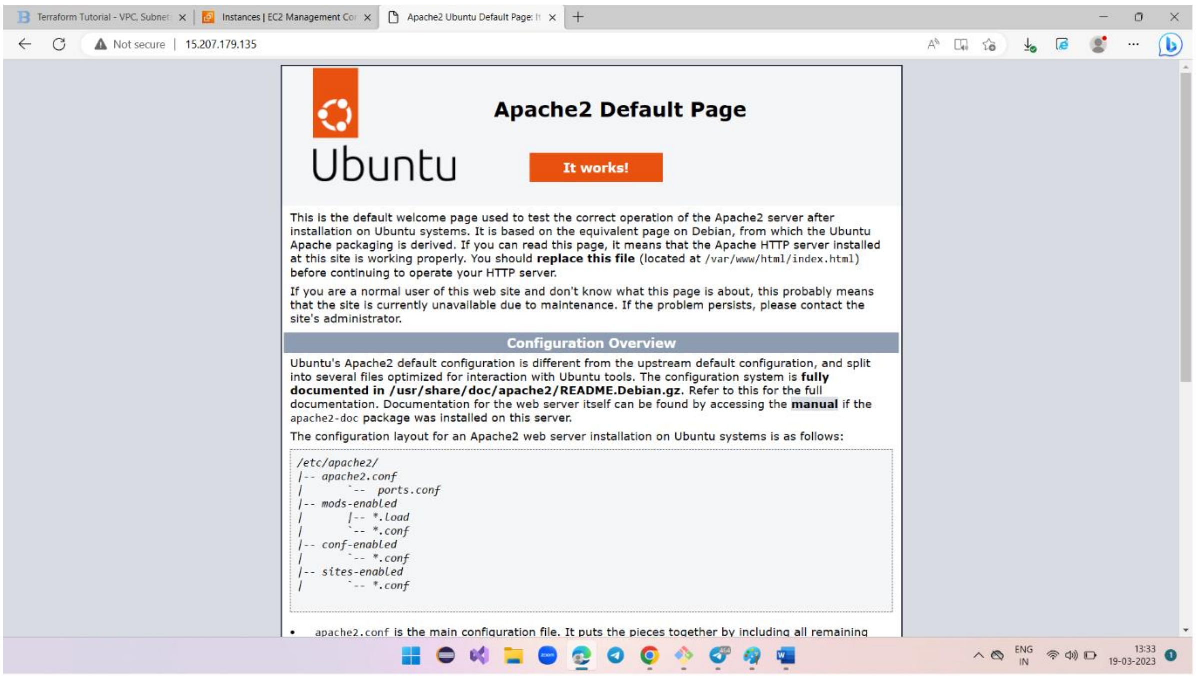
Instance state = running

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv
TF_Ass2	i-05d5e8fafa8be3c60	Running	t2.micro	2/2 checks passed	No alarms	ap-south-1a	–

Instance: i-05d5e8fafa8be3c60 (TF_Ass2)

Instance summary

Instance ID: i-05d5e8fafa8be3c60 (TF_Ass2)	Public IPv4 address: 15.207.179.135 | open address	Private IPv4 addresses: 10.0.0.254
IPv6 address: –	Instance state: Running	Public IPv4 DNS: –
Hostname type: IP name: ip-10-0-0-254.ap-south-1.compute.internal	Private IP DNS name (IPv4 only): ip-10-0-0-254.ap-south-1.compute.internal	
Answer private resource DNS name	Instance type	Elastic IP addresses


Terraform Tutorial - VPC, Subnet
Instances | EC2 Management Co
Apache2 Ubuntu Default Page: It
Not secure | 15.207.179.135
ENG IN
13:33
19-03-2023

Ubuntu

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
```

- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining