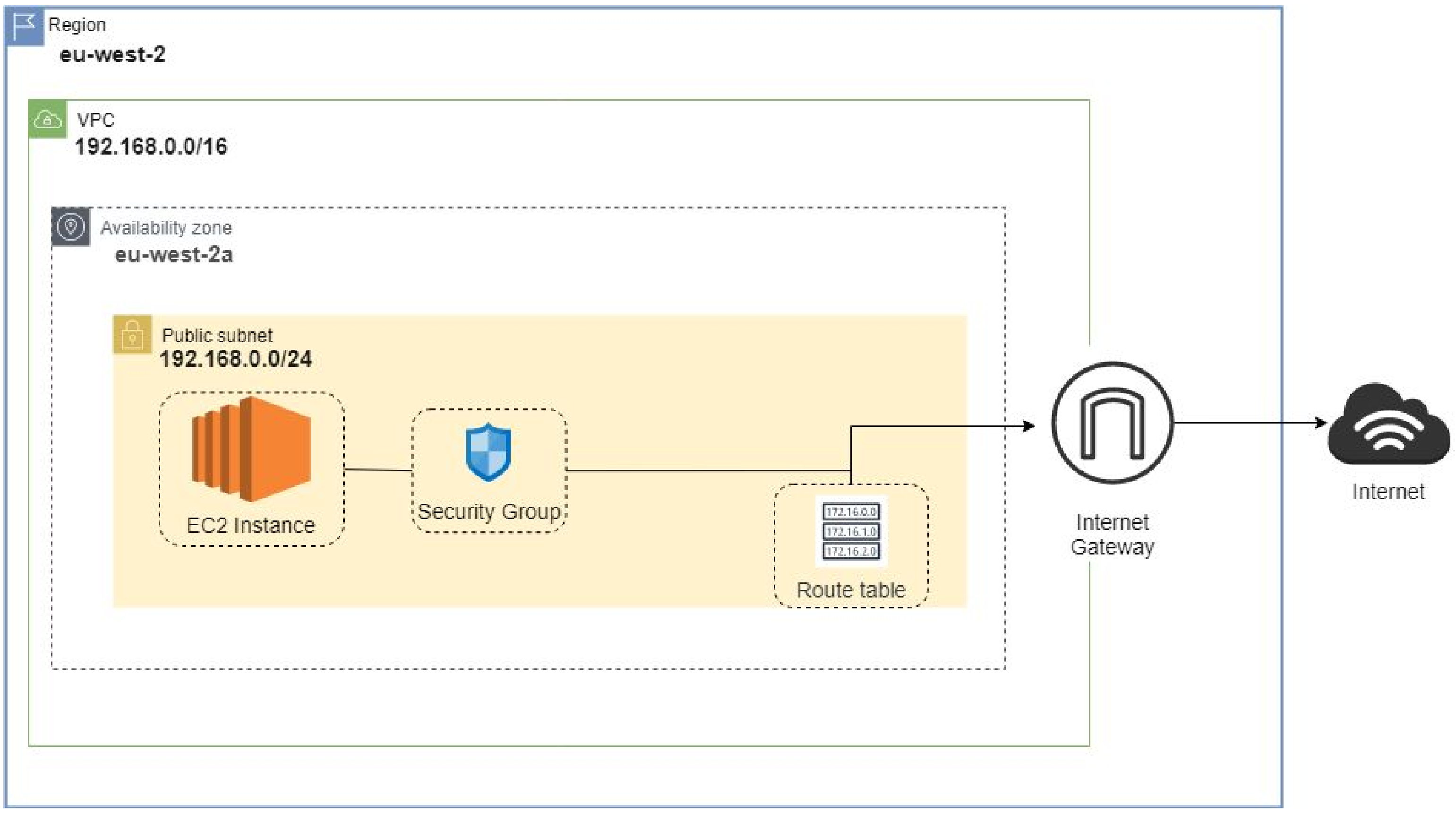
Terraform IaC stack hosting a public web page in a VPC bound environment



Prerequisites:

- AWS CLI
- TerraformAWS Configure profile
- AWS Config profile

Usage:

1. Open IDE terminal or CLI
2. Make sure to select an aws config profile (`setx AWS_PROFILE default`). This stack uses the aws-cli default profile's IAM Credentials.
3. terraform init (initializes terraform)
4. terraform validate (validates the code)
5. terraform plan (shows a preview of changes)
6. terraform apply (runs the code and applies changes)
7. Open website using public ipv4 dns address or public ipv4 address

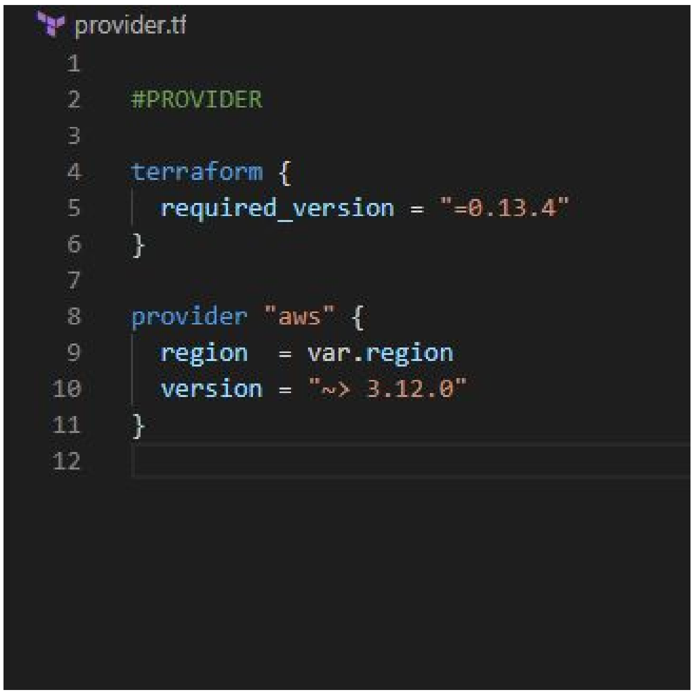
Features

VPC and a public subnet
EC2 instance with user data to install apache web server
Public facing web page with downloaded html page from s3 bucket.
Key pair to ssh into EC2 instance
Route table,security group and Internet gateway for internet access

Code overview

Provider

Specified terraform and aws versions within providers section to make sure there aren't any version inconsistencies. Profile section can be used if you are using a different profile to the default aws profile configured using aws cli.

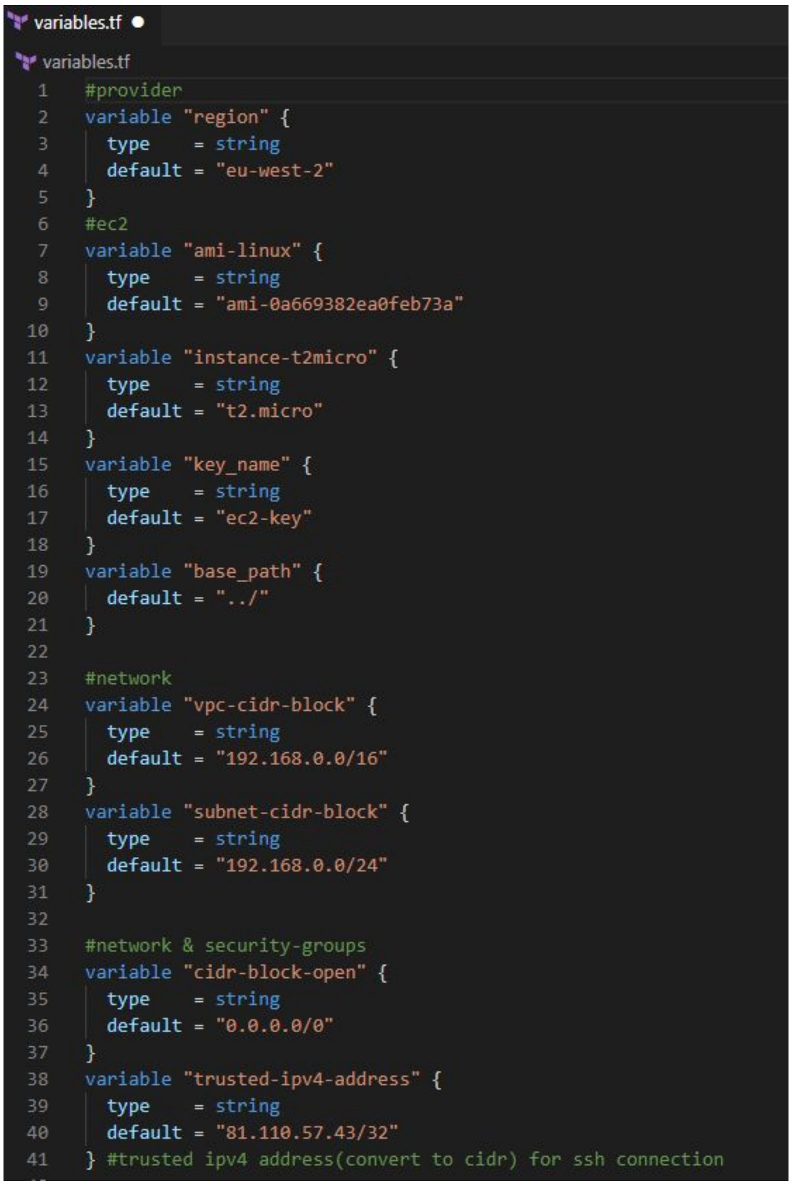


provider.tf

```

#PROVIDER

terraform {
  required_version = "=0.13.4"
}

provider "aws" {
  region  = var.region
  version = "~> 3.12.0"
}

```

Variables

I created the commonly used variables and saved them as strings to be called using *var.variable* within aws resources. I have also labelled the sections of variables using the corresponding .tf file names for improved readability.



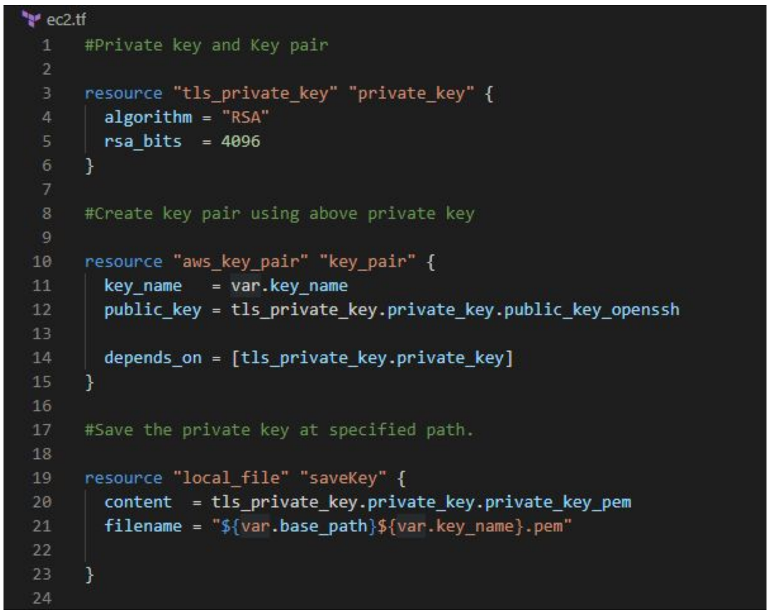
variables.tf ●

variables.tf

```
#provider
variable "region" {
  type    = string
  default = "eu-west-2"
}
#ec2
variable "ami-linux" {
  type    = string
  default = "ami-0a669382ea0feb73a"
}
variable "instance-t2micro" {
  type    = string
  default = "t2.micro"
}
variable "key_name" {
  type    = string
  default = "ec2-key"
}
variable "base_path" {
  default = "../"
}

#network
variable "vpc-cidr-block" {
  type    = string
  default = "192.168.0.0/16"
}
variable "subnet-cidr-block" {
  type    = string
  default = "192.168.0.0/24"
}

#network & security-groups
variable "cidr-block-open" {
  type    = string
  default = "0.0.0.0/0"
}
variable "trusted-ipv4-address" {
  type    = string
  default = "81.110.57.43/32"
} #trusted ipv4 address(convert to cidr) for ssh connection
```

Private key & Key pair for SSH

Private key is created using the RSA algorithm using this private key, an aws_key_pair has been created and saved locally using local_file resource in specified path(var.base_path). This can be used to easily access ec2 instances using ssh.

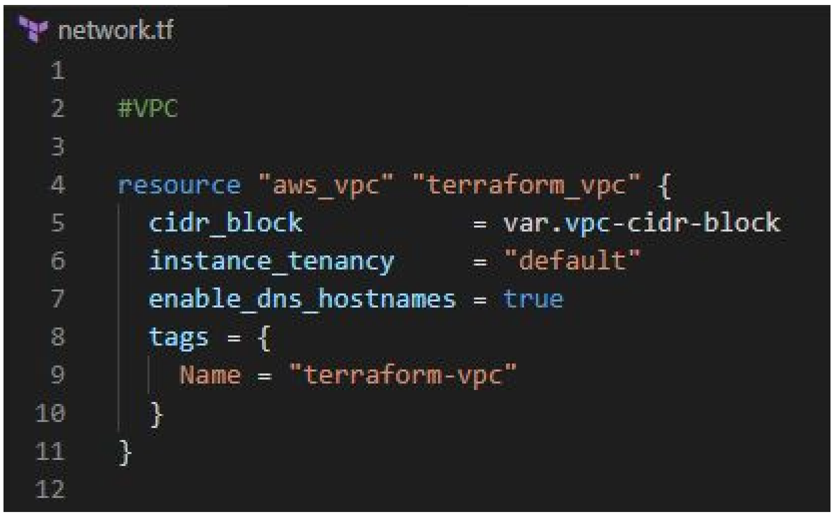


ec2.tf

```
#Private key and Key pair

resource "tls_private_key" "private_key" {
  algorithm = "RSA"
  rsa_bits  = 4096
}

#Create key pair using above private key

resource "aws_key_pair" "key_pair" {
  key_name   = var.key_name
  public_key = tls_private_key.private_key.public_key_openssh

  depends_on = [tls_private_key.private_key]
}

#Save the private key at specified path.

resource "local_file" "saveKey" {
  content  = tls_private_key.private_key.private_key_pem
  filename = "${var.base_path}${var.key_name}.pem"

}
```

VPC

Created a VPC with cidr block 192.168.0.0/16 and set instance tenancy to default. Also enable dns host names to make sure resources within the dns has the same dns host name.

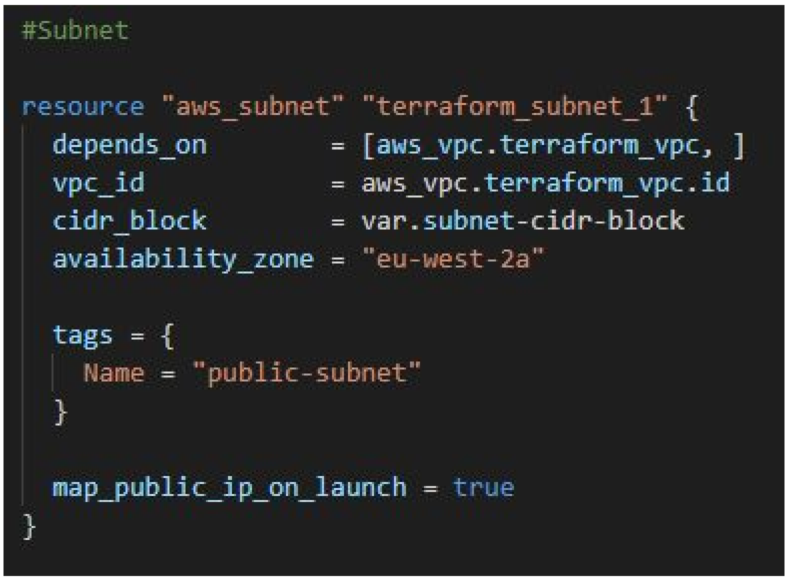


network.tf

```

#VPC

resource "aws_vpc" "terraform_vpc" {
  cidr_block           = var.vpc-cidr-block
  instance_tenancy     = "default"
  enable_dns_hostnames = true
  tags = {
    Name = "terraform-vpc"
  }
}

```

Subnet

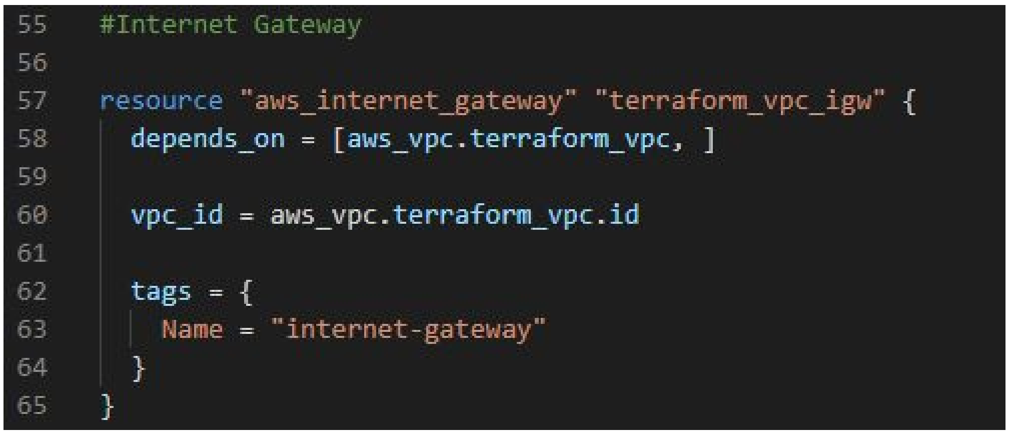
Created a public subnet inside the vpc with the cidr block of 192.168.0.0/24 in eu-west-2a availability zone.I am also using map_public_ip_on_launch to make sure I have a public ip address when the subnet is created.



```
#Subnet

resource "aws_subnet" "terraform_subnet_1" {
  depends_on        = [aws_vpc.terraform_vpc, ]
  vpc_id            = aws_vpc.terraform_vpc.id
  cidr_block        = var.subnet-cidr-block
  availability_zone = "eu-west-2a"

  tags = {
    Name = "public-subnet"
  }

  map_public_ip_on_launch = true
}
```

Internet Gateway

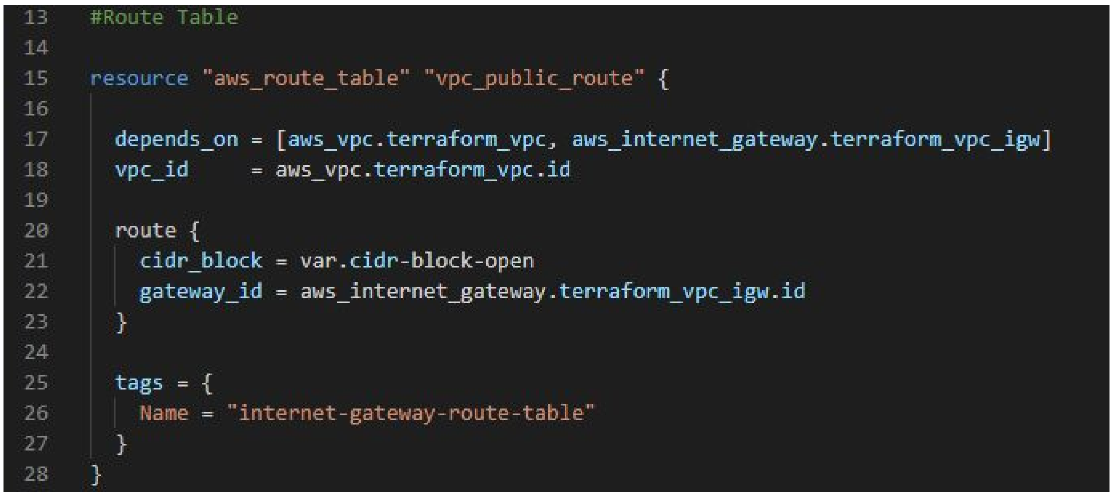
Created an internet gateway to target the vpc for internet traffic and also to provide NAT support for instances that have a public ipv4 address. This gives the vpc access to the internet with the help of route tables.



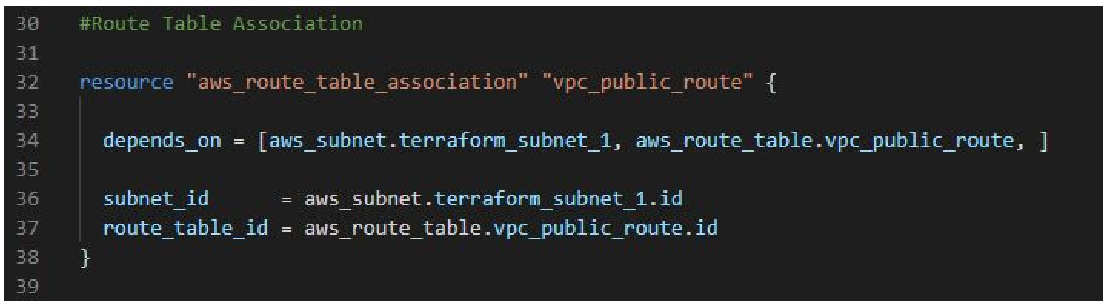
```
#Internet Gateway

resource "aws_internet_gateway" "terraform_vpc_igw" {
  depends_on = [aws_vpc.terraform_vpc, ]

  vpc_id = aws_vpc.terraform_vpc.id

  tags = {
    Name = "internet-gateway"
  }
}
```

Route table and Subnet association

Route table contains the routes(rules) that are used to determine where network traffic from the subnet is directed. Aws_route_table_association is used to associate the route of the public subnet to the route table.



```
#Route Table

resource "aws_route_table" "vpc_public_route" {

  depends_on = [aws_vpc.terraform_vpc, aws_internet_gateway.terraform_vpc_igw]
  vpc_id     = aws_vpc.terraform_vpc.id

  route {
    cidr_block = var.cidr-block-open
    gateway_id = aws_internet_gateway.terraform_vpc_igw.id
  }

  tags = {
    Name = "internet-gateway-route-table"
  }
}
```

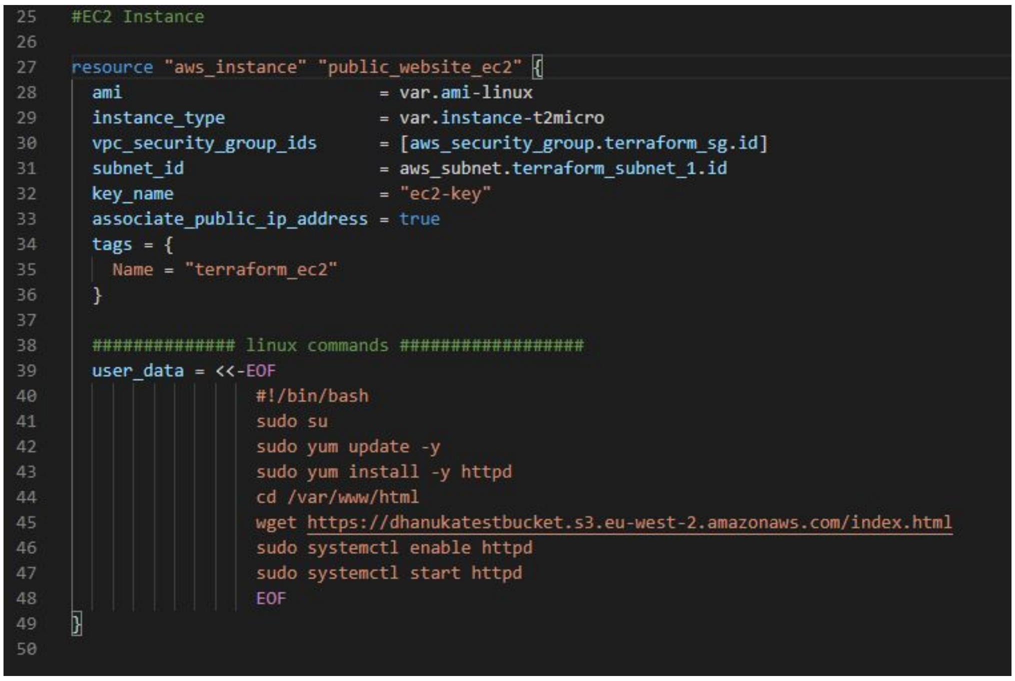


```
#Route Table Association

resource "aws_route_table_association" "vpc_public_route" {

  depends_on = [aws_subnet.terraform_subnet_1, aws_route_table.vpc_public_route, ]

  subnet_id      = aws_subnet.terraform_subnet_1.id
  route_table_id = aws_route_table.vpc_public_route.id
}

```

EC2 web server

Created an EC2 instance within the public subnet with associated public ip address.I also user the user_data section to provide the linux cli commands to

1. Get administrator privileges
2. Software update
3. Install apache
4. Move to /var/www/html directory
5. Copy index.html from s3 bucket
6. Enable and start apache webserver.

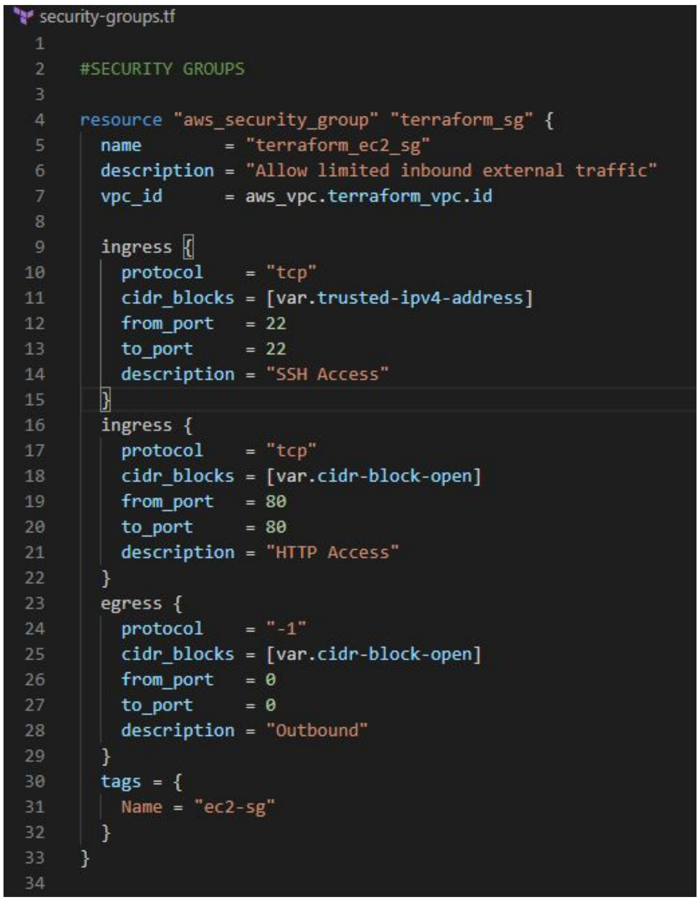


```
#EC2 Instance

resource "aws_instance" "public_website_ec2" {
  ami                         = var.ami-linux
  instance_type               = var.instance-t2micro
  vpc_security_group_ids      = [aws_security_group.terraform_sg.id]
  subnet_id                   = aws_subnet.terraform_subnet_1.id
  key_name                    = "ec2-key"
  associate_public_ip_address = true
  tags = {
    Name = "terraform_ec2"
  }

  ############## linux commands #################
  user_data = <<-EOF
                #!/bin/bash
                sudo su
                sudo yum update -y
                sudo yum install -y httpd
                cd /var/www/html
                wget https://dhanukatestbucket.s3.eu-west-2.amazonaws.com/index.html
                sudo systemctl enable httpd
                sudo systemctl start httpd
                EOF
}
```

Security Groups

For the security groups I am allowing port 80 for usage of http traffic from anywhere and port 22 for ssh using the keypair that is already made. Also adding egress traffic from all ports is allowed as it is public facing.

.

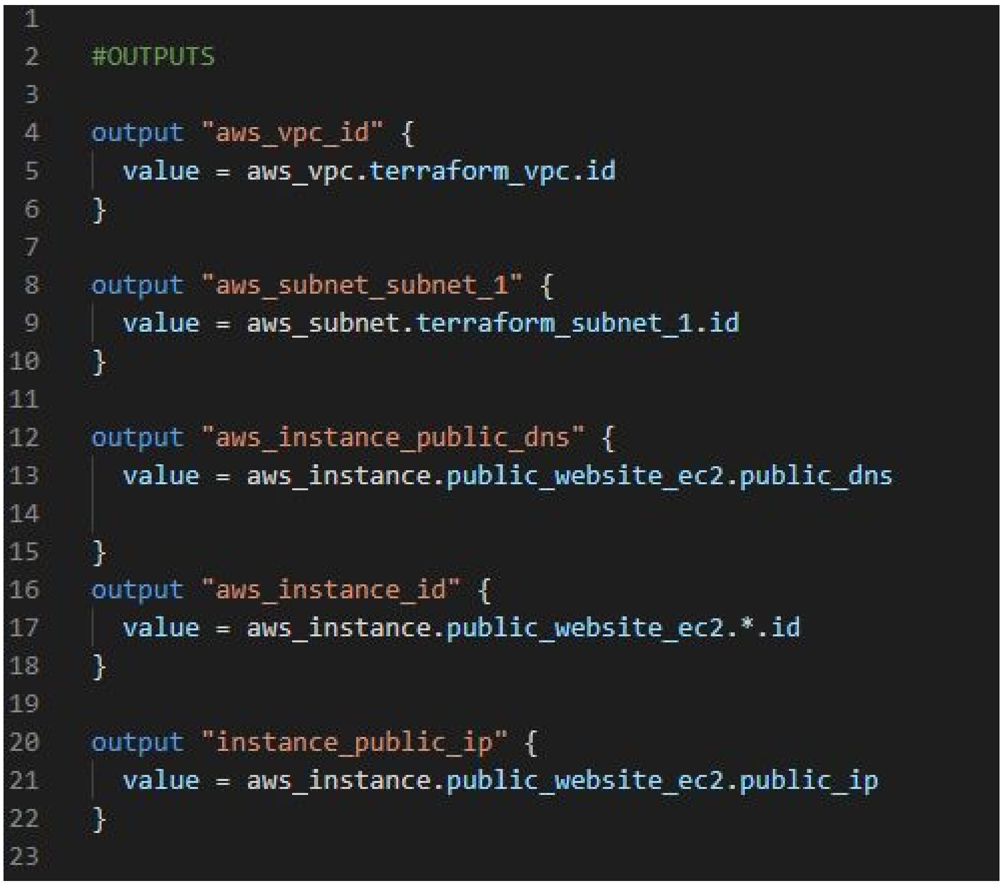


security-groups.tf

```

#SECURITY GROUPS

resource "aws_security_group" "terraform_sg" {
  name        = "terraform_ec2_sg"
  description = "Allow limited inbound external traffic"
  vpc_id      = aws_vpc.terraform_vpc.id

  ingress {
    protocol    = "tcp"
    cidr_blocks = [var.trusted-ipv4-address]
    from_port   = 22
    to_port     = 22
    description = "SSH Access"
  }
  ingress {
    protocol    = "tcp"
    cidr_blocks = [var.cidr-block-open]
    from_port   = 80
    to_port     = 80
    description = "HTTP Access"
  }
  egress {
    protocol    = "-1"
    cidr_blocks = [var.cidr-block-open]
    from_port   = 0
    to_port     = 0
    description = "Outbound"
  }
  tags = {
    Name = "ec2-sg"
  }
}

```

Outputs

This contains all the information that the script will output when I use the command “terraform apply”. I am using this section to make sure the resources I wantare being created and also to easily access the public ip addresses of the infrastructure I want to access through the internet.



```

#OUTPUTS

output "aws_vpc_id" {
  value = aws_vpc.terraform_vpc.id
}

output "aws_subnet_subnet_1" {
  value = aws_subnet.terraform_subnet_1.id
}

output "aws_instance_public_dns" {
  value = aws_instance.public_website_ec2.public_dns

}
output "aws_instance_id" {
  value = aws_instance.public_website_ec2.*.id
}

output "instance_public_ip" {
  value = aws_instance.public_website_ec2.public_ip
}

```