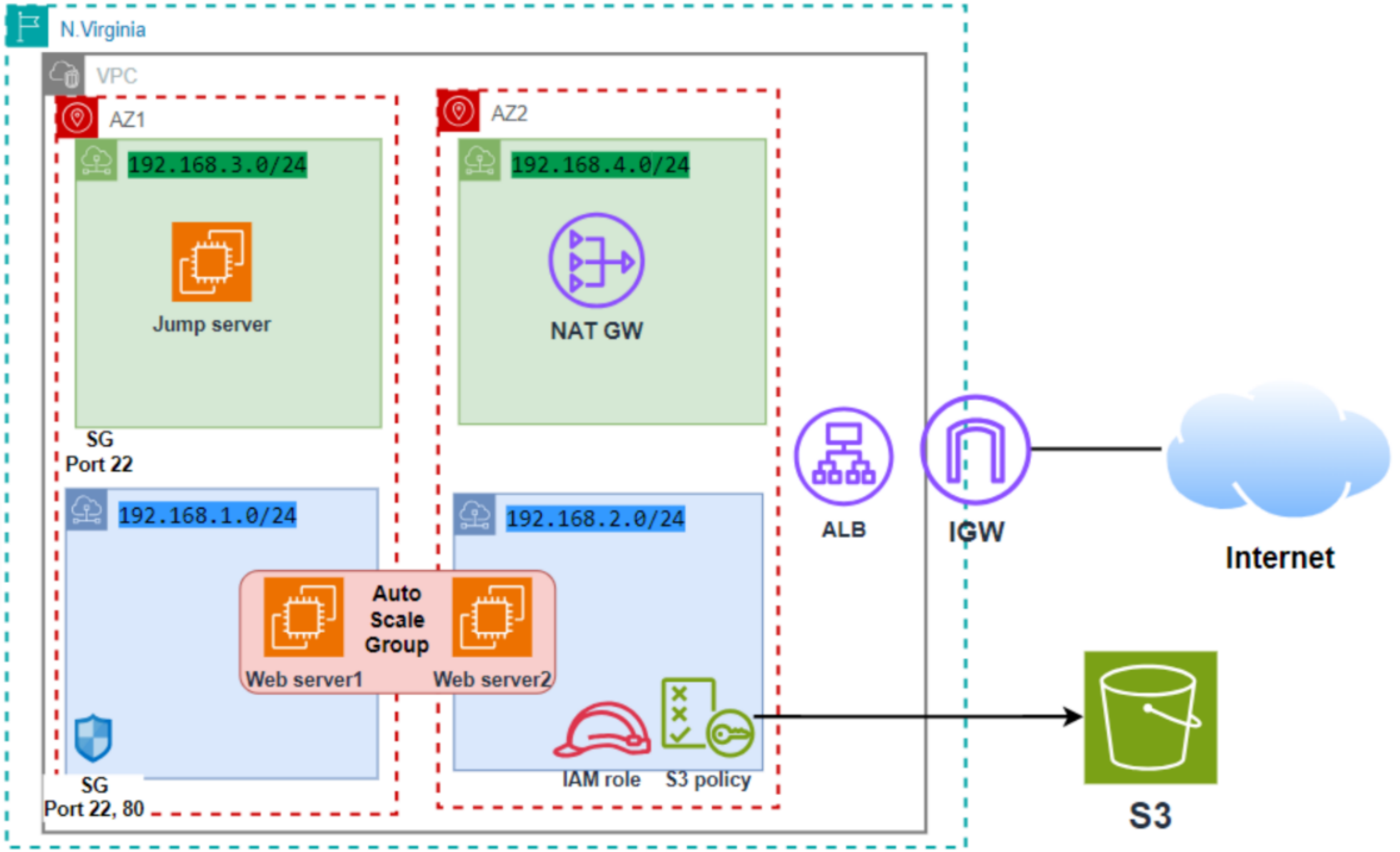
Public Routing Table

Network	Next hop
0.0.0.0/0	IGW

Private Routing Table

Network	Next hop
0.0.0.0/0	NAT GW



WITH

MOHAMED RIZK

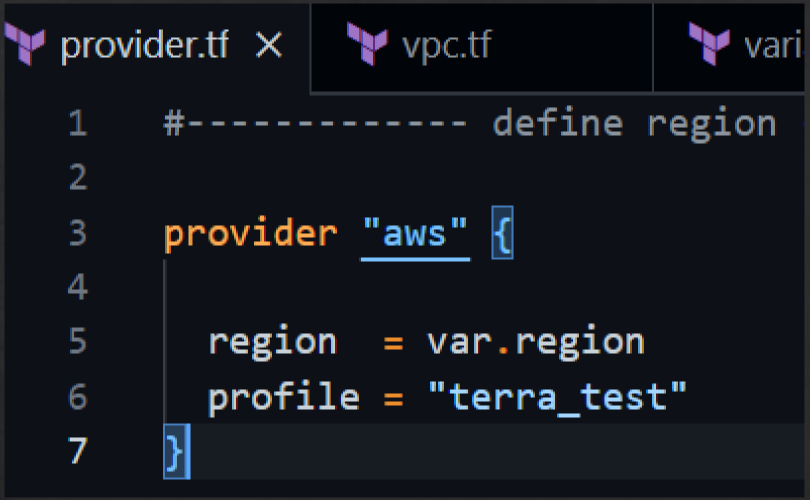
LAMA ALMASSRY

Archticture WITH TERRAFORM

First understanding this archticture and create your vpc after define your provider

1. AWS Provider:

o Configured for the us-east-1 region to deploy the infrastructure.

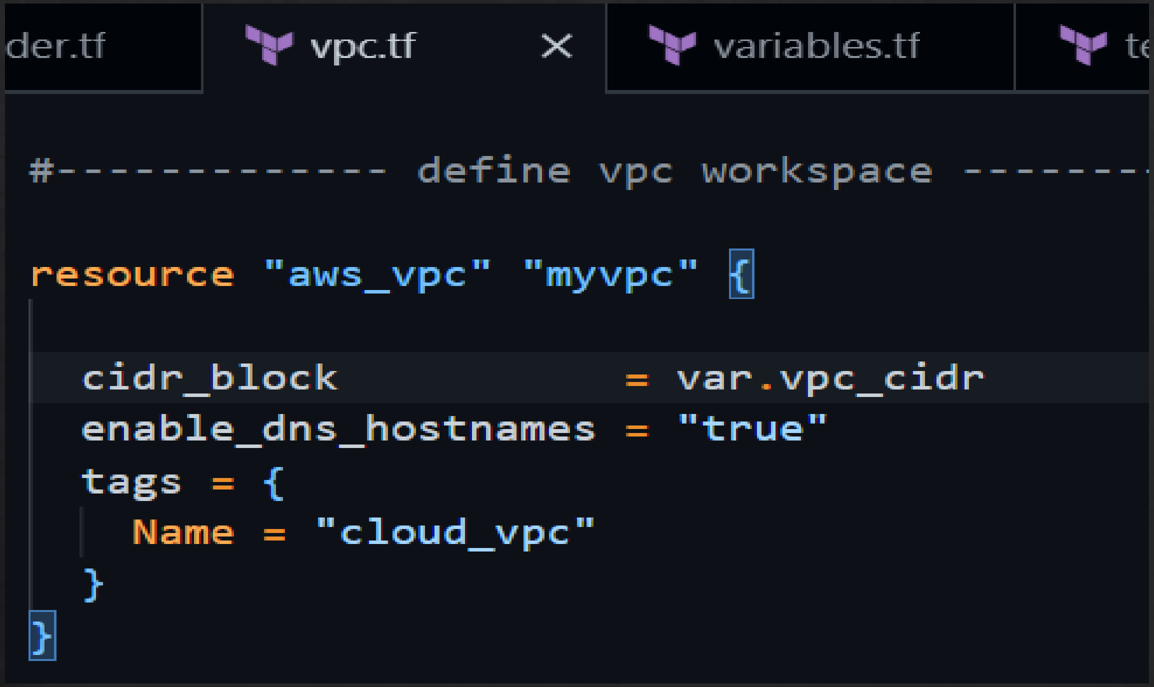


```
#------------ define region

provider "aws" {

  region  = var.region
  profile = "terra_test"
}
```

2. VPC and Subnets:

o A VPC with CIDR block 192.168.0.0/16 is created.

o Four subnets are defined: two for public resources (e.g., ALB, bastion host) and two for private instances (e.g., ASG instances).

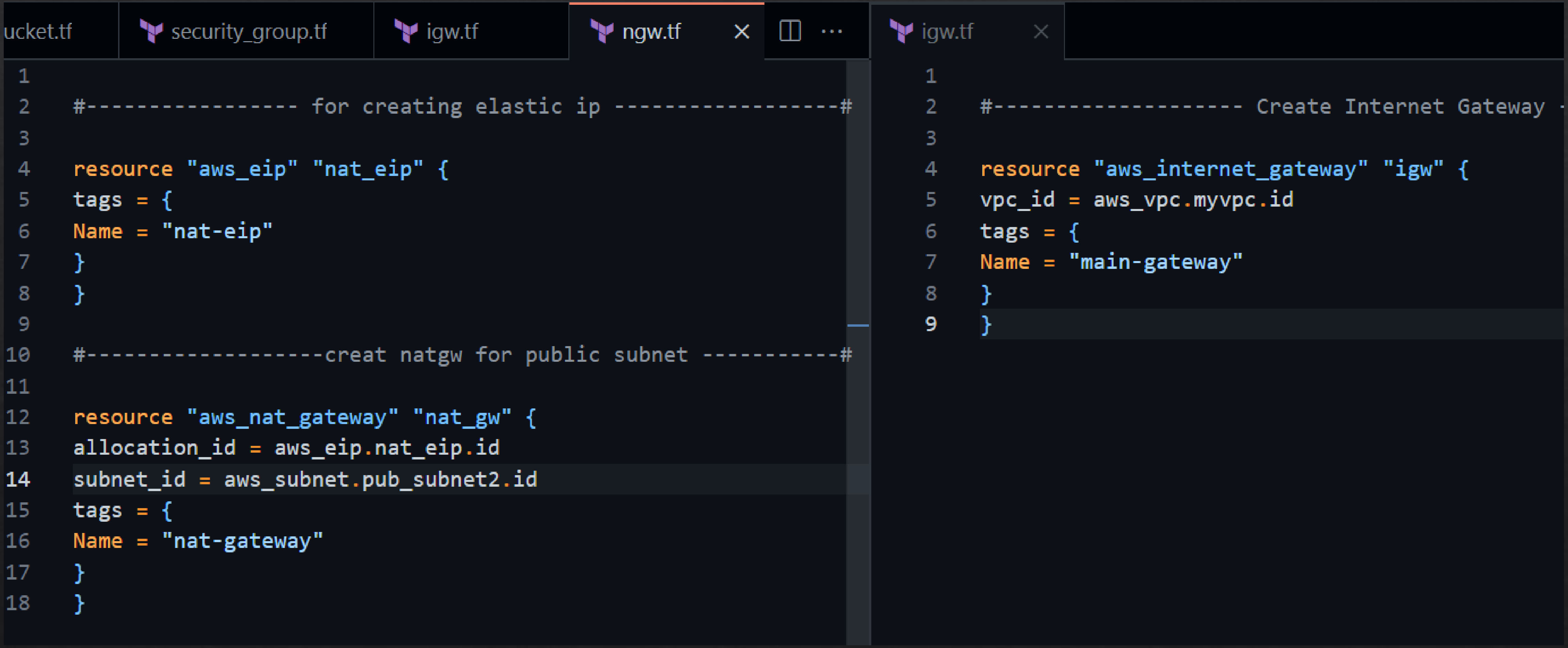


```
#------------- define vpc workspace --------

resource "aws_vpc" "myvpc" {

  cidr_block           = var.vpc_cidr
  enable_dns_hostnames = "true"
  tags = {
    Name = "cloud_vpc"
  }
}
```

3. Internet Gateway & NAT Gateway:

o An Internet Gateway allows public internet access for the public subnets.

o A NAT Gateway provides outbound internet access to private instances via the NAT gateway in the public subnet.



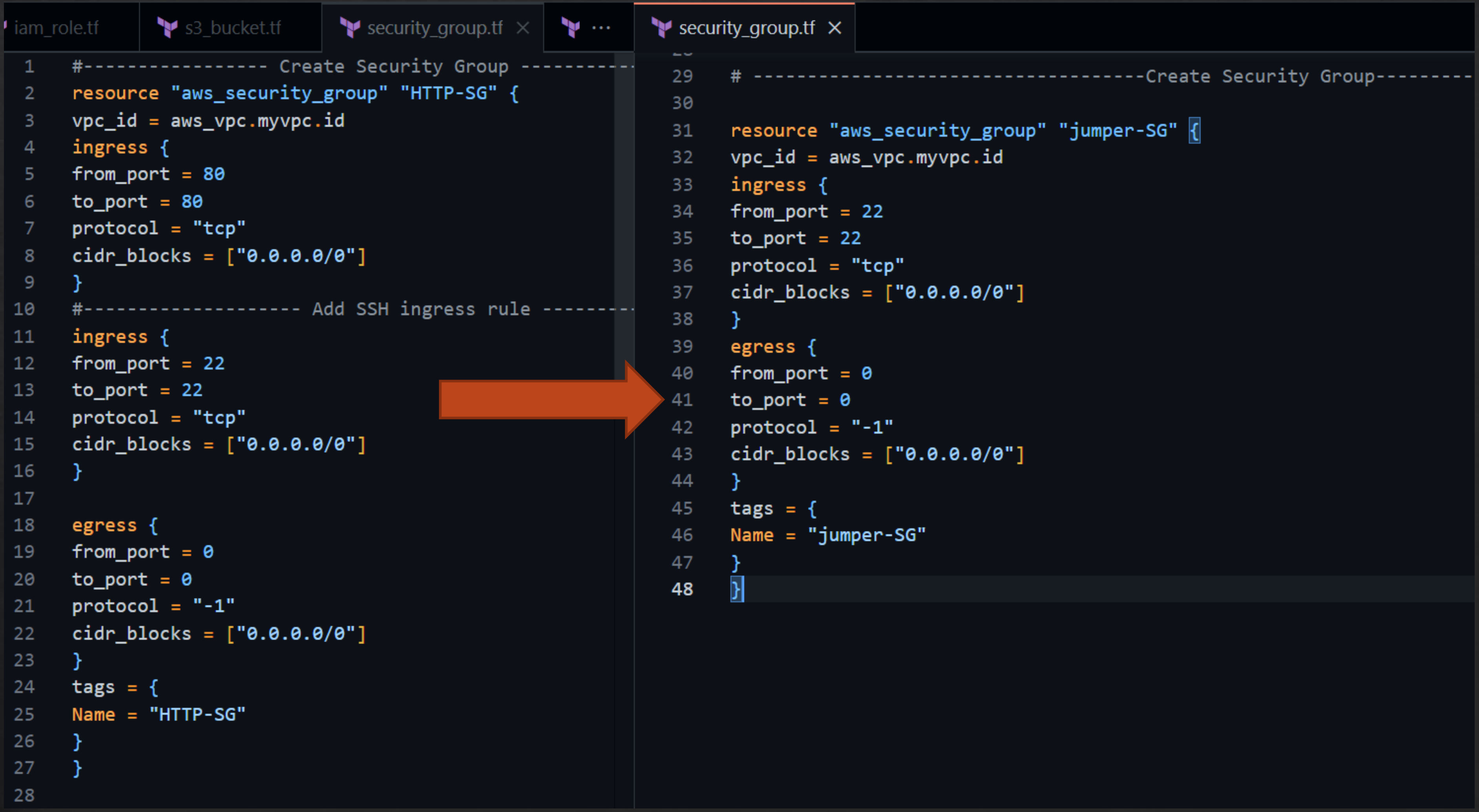
```
#---------------- for creating elastic ip -----------------#

resource "aws_eip" "nat_eip" {
tags = {
Name = "nat-eip"
}
}

#------------------creat natgw for public subnet -----------#

resource "aws_nat_gateway" "nat_gw" {
allocation_id = aws_eip.nat_eip.id
subnet_id = aws_subnet.pub_subnet2.id
tags = {
Name = "nat-gateway"
}
}
```

```
#------------------- Create Internet Gateway -

resource "aws_internet_gateway" "igw" {
vpc_id = aws_vpc.myvpc.id
tags = {
Name = "main-gateway"
}
}
```

4. Security Groups:

o Security group for ALB and instances allows HTTP (port 80) and SSH (port 22) traffic from the internet.

o A dedicated security group for the bastion host allows SSH access, enabling the management of private instances through the bastion host.

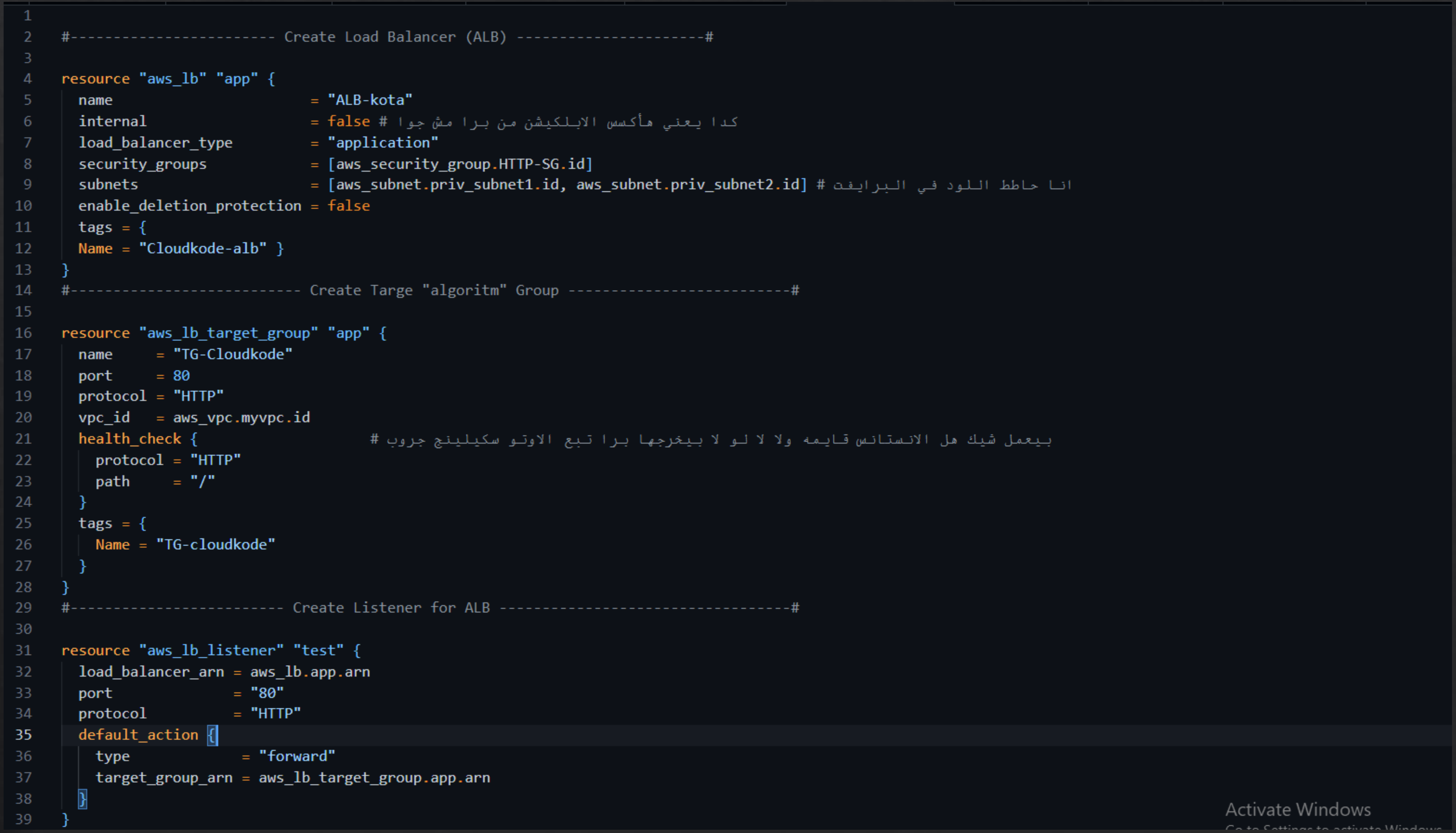


```
#---------------- Create Security Group ---------
resource "aws_security_group" "HTTP-SG" {
vpc_id = aws_vpc.myvpc.id
ingress {
from_port = 80
to_port = 80
protocol = "tcp"
cidr_blocks = ["0.0.0.0/0"]
}
#------------------ Add SSH ingress rule --------
ingress {
from_port = 22
to_port = 22
protocol = "tcp"
cidr_blocks = ["0.0.0.0/0"]
}

egress {
from_port = 0
to_port = 0
protocol = "-1"
cidr_blocks = ["0.0.0.0/0"]
}
tags = {
Name = "HTTP-SG"
}
}

```

```
# -----------------------------------Create Security Group--------

resource "aws_security_group" "jumper-SG" {
vpc_id = aws_vpc.myvpc.id
ingress {
from_port = 22
to_port = 22
protocol = "tcp"
cidr_blocks = ["0.0.0.0/0"]
}
egress {
from_port = 0
to_port = 0
protocol = "-1"
cidr_blocks = ["0.0.0.0/0"]
}
tags = {
Name = "jumper-SG"
}
}
```

5. Application Load Balancer (ALB):

o The ALB listens on port 80 and forwards traffic to an Auto Scaling Group (ASG) through a target group. Health checks are performed on the root path to ensure instance availability.

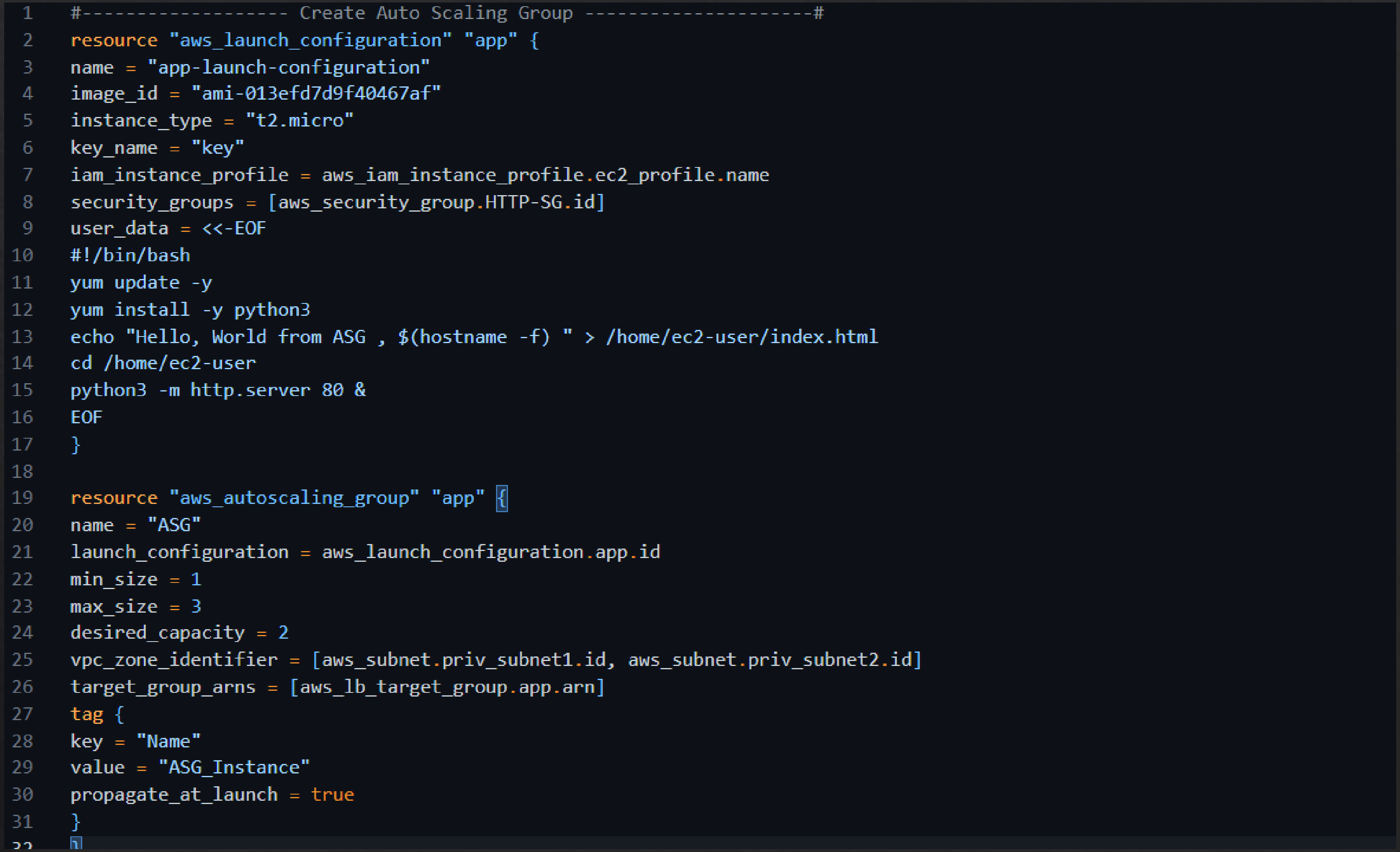


```
#----------------------- Create Load Balancer (ALB) ---------------------#

resource "aws_lb" "app" {
  name                       = "ALB-kota"
  internal                   = false # كدا يعني هأكسس الابلكيشن من برا مش جوا
  load_balancer_type         = "application"
  security_groups            = [aws_security_group.HTTP-SG.id]
  subnets                    = [aws_subnet.priv_subnet1.id, aws_subnet.priv_subnet2.id] # انا حاطط اللود في البرايفت
  enable_deletion_protection = false
  tags = {
  Name = "Cloudkode-alb" }
}
#-------------------------- Create Targe "algoritm" Group -------------------------#

resource "aws_lb_target_group" "app" {
  name     = "TG-Cloudkode"
  port     = 80
  protocol = "HTTP"
  vpc_id   = aws_vpc.myvpc.id
  health_check {                    # بيعمل شيك هل الانستانس قايمه ولا لا لو لا بيخرجها برا تبع الاوتو سكيلينج جروب
    protocol = "HTTP"
    path     = "/"
  }
  tags = {
    Name = "TG-cloudkode"
  }
}
#----------------------- Create Listener for ALB --------------------------------#

resource "aws_lb_listener" "test" {
  load_balancer_arn = aws_lb.app.arn
  port              = "80"
  protocol          = "HTTP"
  default_action {
    type             = "forward"
    target_group_arn = aws_lb_target_group.app.arn
  }
}
```

6. Auto Scaling Group (ASG):

o The ASG dynamically scales between 1 to 3 instances, ensuring high availability. Each instance hosts a simple Python web server that serves an "Hello, World" page.

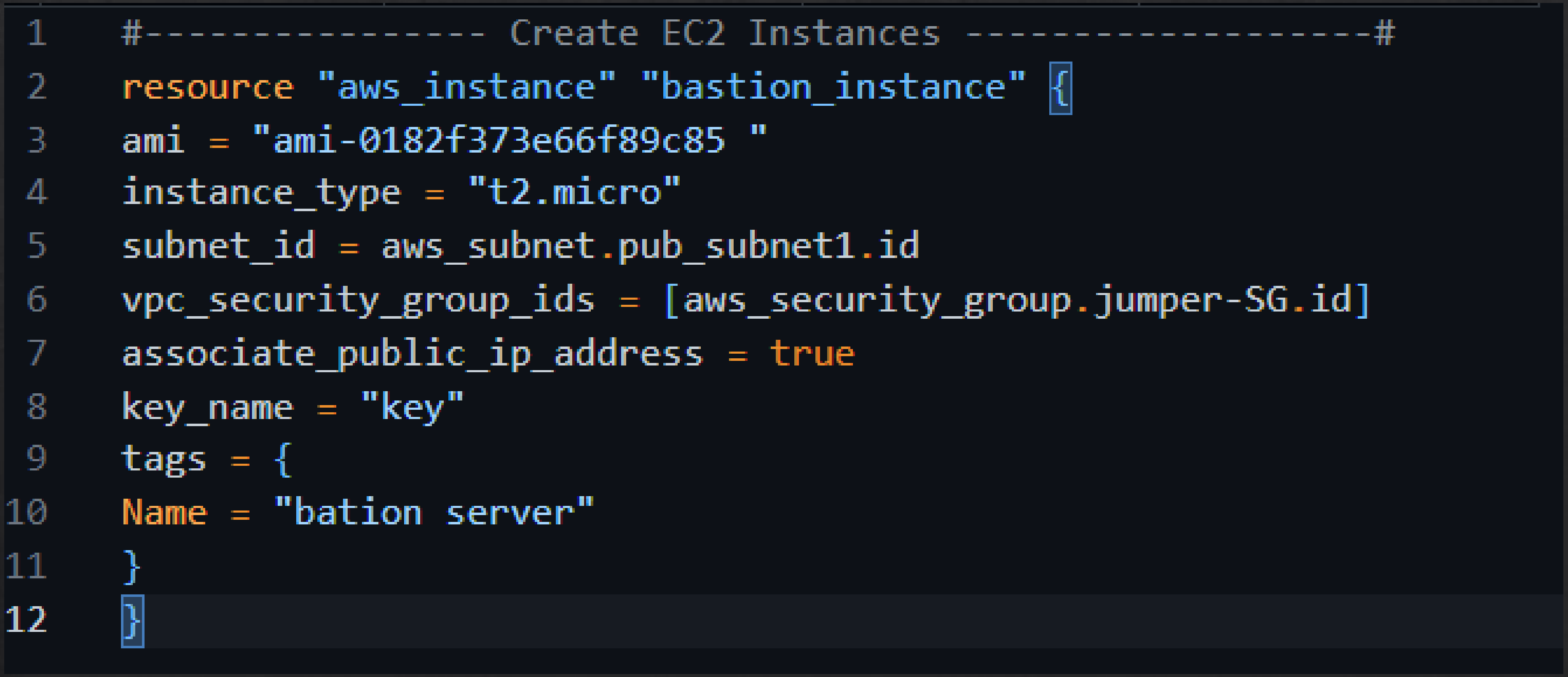
o The ASG is configured to use an EC2 Launch Configuration, which installs necessary software, including Python.



```
#------------------- Create Auto Scaling Group ---------------------#
resource "aws_launch_configuration" "app" {
name = "app-launch-configuration"
image_id = "ami-013efd7d9f40467af"
instance_type = "t2.micro"
key_name = "key"
iam_instance_profile = aws_iam_instance_profile.ec2_profile.name
security_groups = [aws_security_group.HTTP-SG.id]
user_data = <<-EOF
#!/bin/bash
yum update -y
yum install -y python3
echo "Hello, World from ASG , $(hostname -f) " > /home/ec2-user/index.html
cd /home/ec2-user
python3 -m http.server 80 &
EOF
}

resource "aws_autoscaling_group" "app" {
name = "ASG"
launch_configuration = aws_launch_configuration.app.id
min_size = 1
max_size = 3
desired_capacity = 2
vpc_zone_identifier = [aws_subnet.priv_subnet1.id, aws_subnet.priv_subnet2.id]
target_group_arns = [aws_lb_target_group.app.arn]
tag {
key = "Name"
value = "ASG_Instance"
propagate_at_launch = true
}
}
```

7. Bastion Host:

- A bastion host (EC2 instance) is deployed in the public subnet to securely SSH into the private EC2 instances.
- Users upload an SSH key to the bastion host for connecting to private instances.

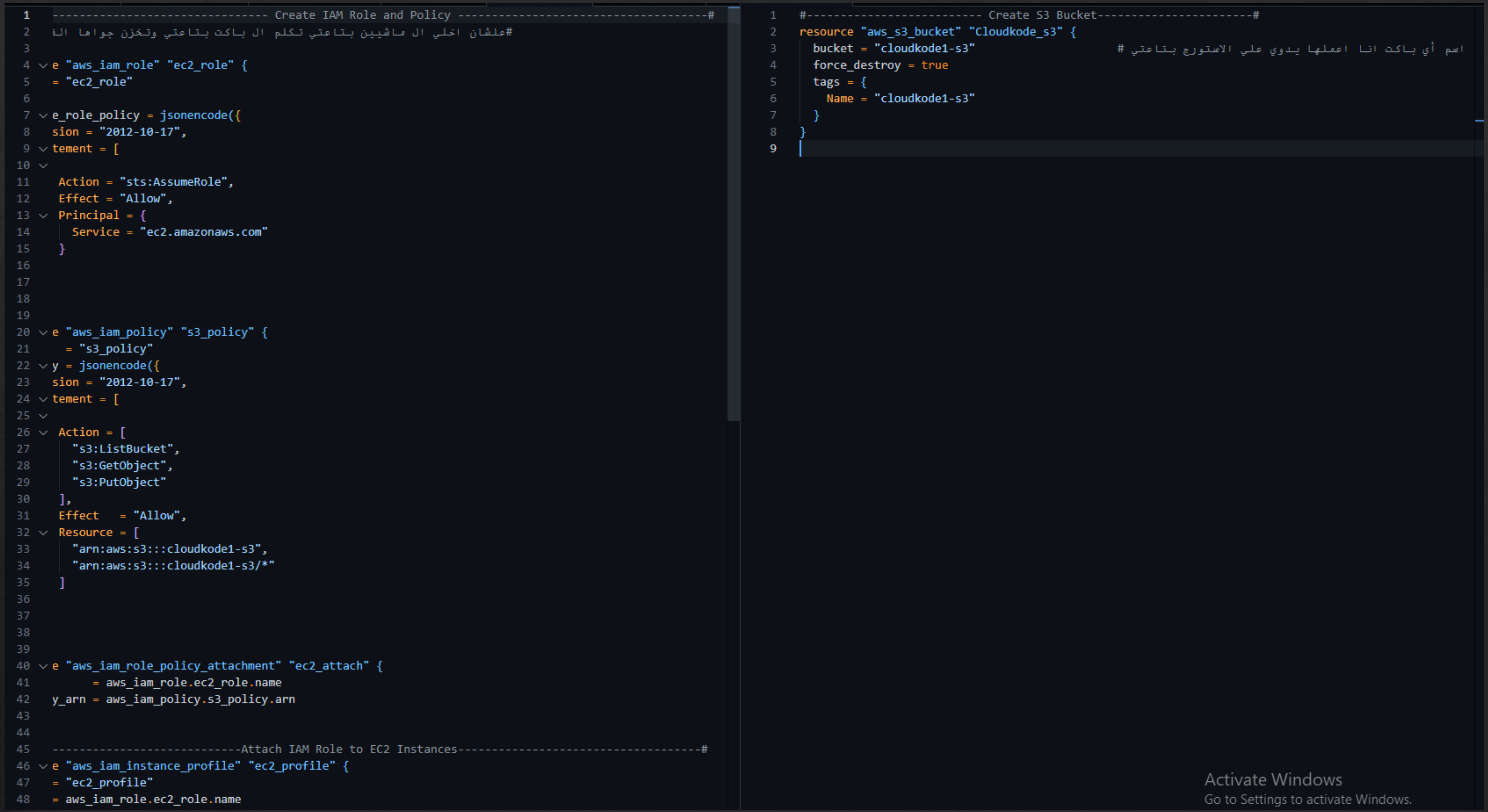


```
#---------------- Create EC2 Instances -------------------#
resource "aws_instance" "bastion_instance" {
ami = "ami-0182f373e66f89c85 "
instance_type = "t2.micro"
subnet_id = aws_subnet.pub_subnet1.id
vpc_security_group_ids = [aws_security_group.jumper-SG.id]
associate_public_ip_address = true
key_name = "key"
tags = {
Name = "bation server"
}
}
```

8. IAM Roles and S3 Integration:

o An IAM role is assigned to the EC2 instances, allowing them to access the S3 bucket.

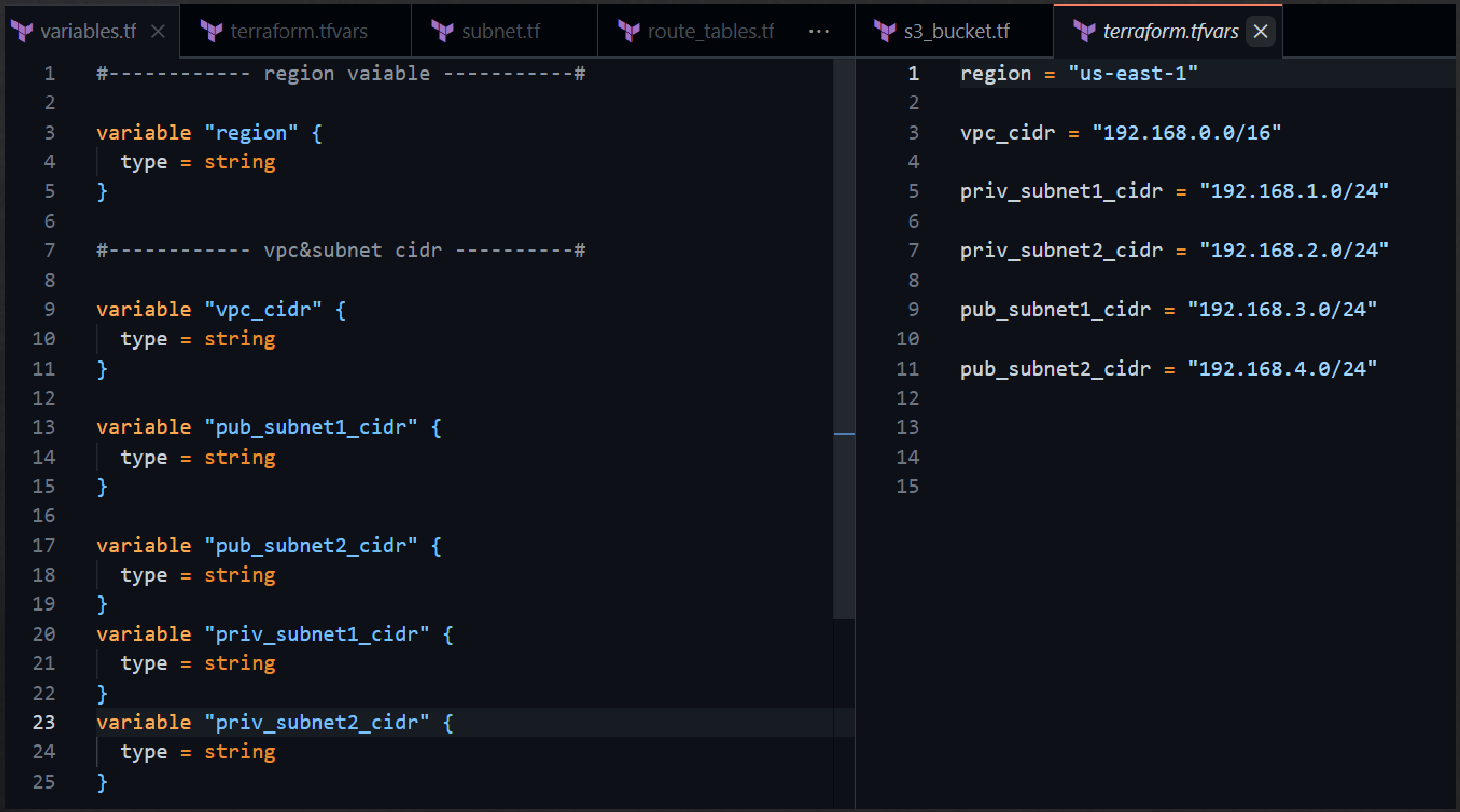
o Instances can read and write to the Cloudkode-s3 S3 bucket for backups or other file storage purposes.



```
------------------------------- Create IAM Role and Policy ------------------------------------#
#علشان اخلي ال ماشيين بتاعتي تكلم ال باكت بتاعتي وتخزن جواها ال

e "aws_iam_role" "ec2_role" {
= "ec2_role"

e_role_policy = jsonencode({
sion = "2012-10-17",
tement = [

 Action = "sts:AssumeRole",
 Effect = "Allow",
 Principal = {
   Service = "ec2.amazonaws.com"
 }




e "aws_iam_policy" "s3_policy" {
  = "s3_policy"
y = jsonencode({
sion = "2012-10-17",
tement = [

 Action = [
   "s3:ListBucket",
   "s3:GetObject",
   "s3:PutObject"
 ],
 Effect   = "Allow",
 Resource = [
   "arn:aws:s3:::cloudkode1-s3",
   "arn:aws:s3:::cloudkode1-s3/*"
 ]




e "aws_iam_role_policy_attachment" "ec2_attach" {
      = aws_iam_role.ec2_role.name
y_arn = aws_iam_policy.s3_policy.arn


--------------------------Attach IAM Role to EC2 Instances-----------------------------------#
e "aws_iam_instance_profile" "ec2_profile" {
= "ec2_profile"
= aws_iam_role.ec2_role.name
```

```
#-------------------------- Create S3 Bucket---------------------#
resource "aws_s3_bucket" "Cloudkode_s3" {
  bucket = "cloudkode1-s3"                       # اسم أي باكت انا اعملها يدوي علي الاستورج بتاعتي
  force_destroy = true
  tags = {
    Name = "cloudkode1-s3"
  }
}

```

9. Some benefit usese variables like an example



```
#----------- region vaiable ----------#

variable "region" {
  type = string
}

#----------- vpc&subnet cidr ---------#

variable "vpc_cidr" {
  type = string
}

variable "pub_subnet1_cidr" {
  type = string
}

variable "pub_subnet2_cidr" {
  type = string
}
variable "priv_subnet1_cidr" {
  type = string
}
variable "priv_subnet2_cidr" {
  type = string
}
```

```
region = "us-east-1"

vpc_cidr = "192.168.0.0/16"

priv_subnet1_cidr = "192.168.1.0/24"

priv_subnet2_cidr = "192.168.2.0/24"

pub_subnet1_cidr = "192.168.3.0/24"

pub_subnet2_cidr = "192.168.4.0/24"
```

> “ At the end I want to thanks Mohamed rizk and lama almassry with this huge experiences and encourage them to keep going away and take us . ”

Thank you