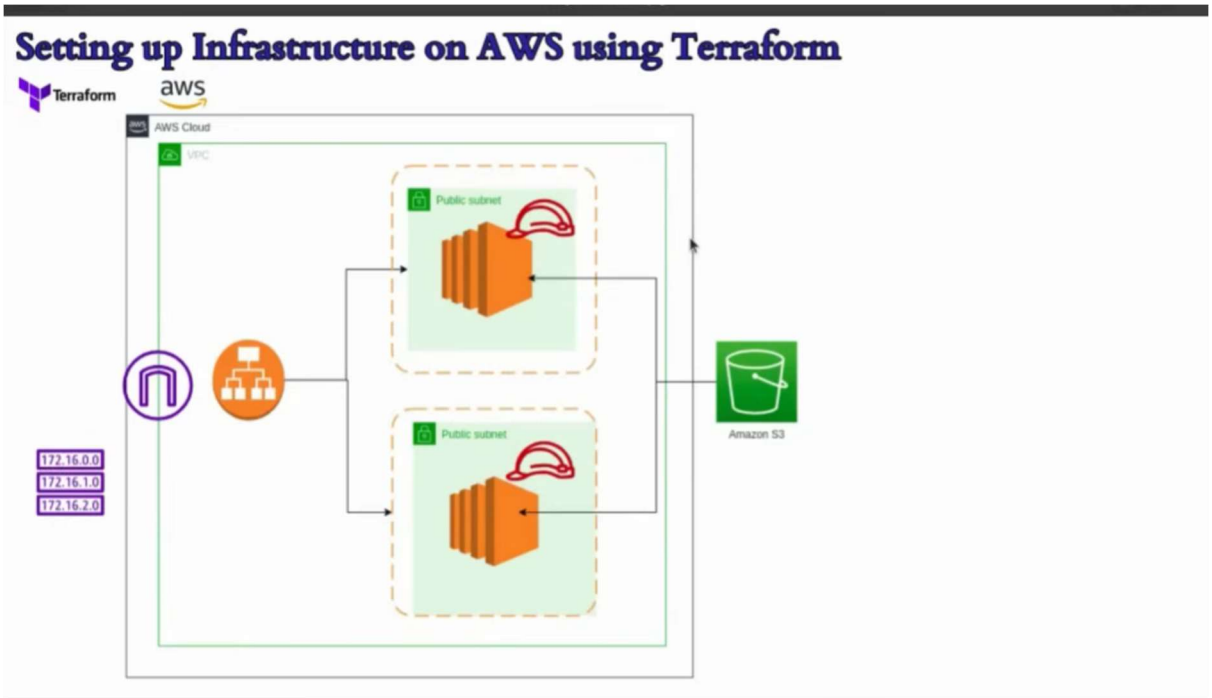
Name:-Roshan Saral Kumar

AWS PROJECT - 3
SETTING UP AWS INFRASTRUCTURE USING TERRAFORM

ARCHITECTURAL DIAGRAM:-



- WHEN USING TERRAFORM FOR THIS ARCHITECTURE WE HAVE TO FIRST THINK ABOUT USING A PROVIDER IN OUR SCRIPT.(PROVDER-> GIVES US ACCESS TO ALL THE SERVICES. EXAMPLE AWS,GCP,AZURE).

- **aws Provider (Classic AWS Provider)**
- **Official Provider:** hashicorp/aws
- **Mature & widely used** — almost all Terraform tutorials and modules use this.
- **Resources:** EC2, S3, VPC, IAM, RDS, Lambda, etc.
- **Syntax Example:**
- provider "aws" {
- region = "us-east-1"
- }
-
- resource "aws_s3_bucket" "my_bucket" {
- bucket = "my-terraform-bucket"
- acl = "private"
- }
- ✅ **Use this if:**
- You want stability and compatibility.
- You're using standard AWS resources.
- You're following tutorials or community modules.
-
- 2️⃣ **awscc Provider (AWS Cloud Control API Provider)**

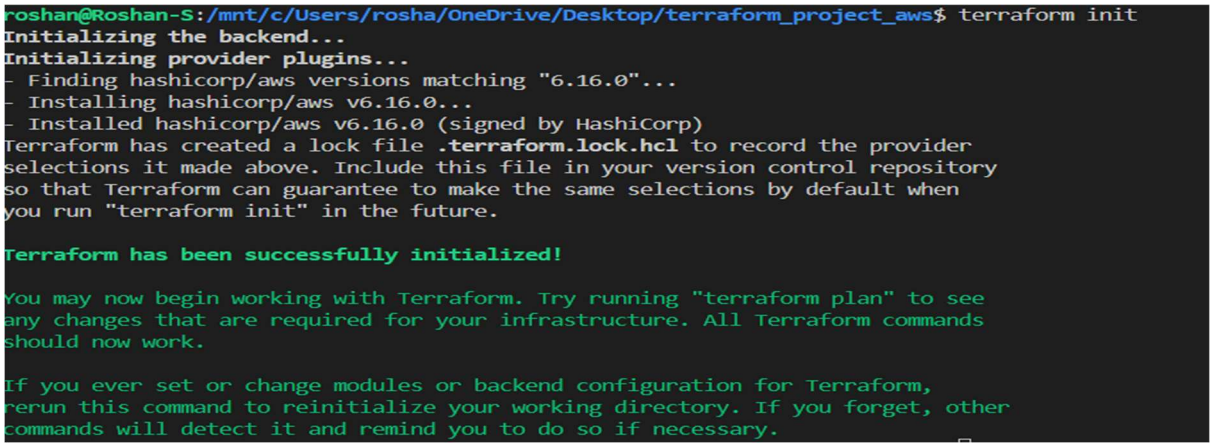
- **Official Provider:** hashicorp/awscc
- **Newer** — built on **AWS Cloud Control API**.
- **Supports some newer AWS services** or resources that aren't fully supported in aws.
- **Syntax Example:**
- provider "awscc" {
- region = "us-east-1"
- }
-
- resource "awscc_s3_bucket" "my_bucket" {
- bucket_name = "my-terraform-bucket"
- }
- ✅ **Use this if:**
- You want to manage **very new AWS services** that the classic provider doesn't support yet.
- You are experimenting with Cloud Control API features.

Feature	aws	awscc
Maturity	Stable, widely used	New, less tested
Resource Coverage	Most AWS services	Very new / upcoming AWS services
Community Modules	Huge ecosystem	Very limited
Syntax	Classic (aws s3 bucket)	awscc prefixed (awscc s3 bucket)
Recommendation	✅ Most projects	⚡ Only if needed

- 💡 **Recommendation**
- **Default:** Use aws provider for most Terraform projects.
- **Only use awscc** if you need resources not yet supported in aws.

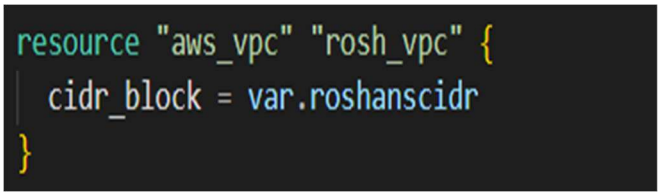
LAWAYS BEFORE STARTING TO USE TERRAFORM IN ANY PROJECT ALWAYS INITIALISE THE TERRAFORM IN THE PROJECT BY USING:-

Command:-
terraform init



after this write ur scripts for the architecture: mentioned previously

step 1:- CREATE THE VPC:- WHICH IS THE NETWORK FOR THE REGION.

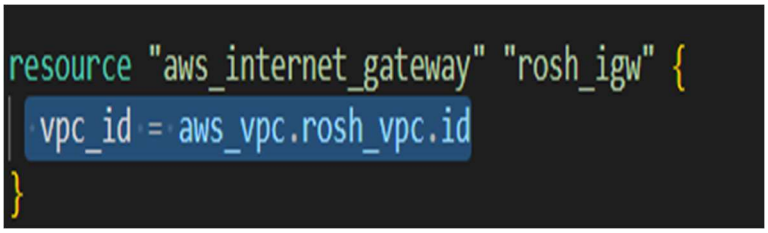


WHEN I SAY THE VALUE LIKE var.roshanscidr it is referencing to the variables.tf file here I have created my variable so to access the variable we use something like var.(the elemnt to access)

Varibales.tf file looks like this:-

STEP 2:- CREATETHE TWO SUBNETS IN 2 DIFFERENT AVAILABILITITY ZONES:-

STEP 3:- IS CREATE THE INTERNET GATEWAY FOR THE VPC:-



STEP 4:- CREATE THE ROUTE TABLE:-

```
resource "aws_route_table" "rosh_RT" {
  vpc_id = aws_vpc.rosh_vpc.id

  route { # we are trying to route the external traffic to the internet gateway
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.rosh_igw.id
  }
}
```

STEP 5;-CREATE ASSOCIATIONS FOR THE ROUTE TABLE OR WE CAN SAY SUBNET ASSOCIATIONS.

```
resource "aws_route_table_association" "rosh_RT_1a" { #for the internet gateway we wa
  subnet_id      = aws_subnet.rosh_sub_1.id
  route_table_id = aws_route_table.rosh_RT.id
}

resource "aws_route_table_association" "rosh_RT_1b" {
  subnet_id      = aws_subnet.rosh_sub_2.id
  route_table_id = aws_route_table.rosh_RT.id
}
```

STEP 6:-WE HAVE TO CREATE SECURITY GROUPS TO ALLOW TRAFFIC TO OUR INSTANCES. SECURITY GROUPS ARE USED AT THE INSTANCE LEVEL. AT THE SUBNET LEVEL OR AT THE NETWROK LEVEL WE CAN USE NACL(NETWORK ACCESS CONTROL LIST).

```
resource "aws_security_group" "roshan_web_sg" { #we are creating a security group which is statefull and it is at the instance level.subnet level we have to go for NACL
  name   = "roshan_web_sg"
  vpc_id = aws_vpc.rosh_vpc.id

  ingress { #inbound rule
    description = "HTTP TRAFFIC FROM VPC"
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress { #inbound rule
    description = "SSH TRAFFIC"
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress { # outbound rule
    description = "OUTBOUND TRAFFIC"
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

  tags = {

    name = "roshan_web_sg"

  }
}
```

STEP 7:- AFTER CREATING SECURITY GROUPS FORM THE UNIQUE S3 BUCKET.

```
resource "aws_s3_bucket" "rosh_s3" { # Unique bucket creation
  bucket = "roshan20252025-bucketcreation"
}
```

STEP 8:- CREATE THE INSTANCES FOR THE TWO SUBNETS CREATED IN TWO DIFFERENENT AVAILABILITY ZONES.

```
resource "aws_instance" "ec2-instance-1" {
  ami                    = "ami-02d26659fd82cf299"
  instance_type          = "t3.micro"
  vpc_security_group_ids = [aws_security_group.roshan_web_sg.id]
  subnet_id              = aws_subnet.rosh_sub_1.id
  user_data_base64       = base64encode(file("roshansuserdata1.sh")) #make surethat the variable u are using is user_data_base64 not user_data because we will get a warning stating that we have to g
}

resource "aws_instance" "ec2-instance-2" {
  ami                    = "ami-02d26659fd82cf299"
  instance_type          = "t3.micro"
  vpc_security_group_ids = [aws_security_group.roshan_web_sg.id]
  subnet_id              = aws_subnet.rosh_sub_2.id
  user_data_base64       = base64encode(file("roshansuserdata2.sh"))
}
```

STEP 9:- AFTER CREATING THE INSTANCES CREATET HE APPLICATION LOAD BALANCER TO EVENLY DISTRIBUTE THE TRAFFIC AND ALSO MENTION THE SUBNET IDS, SECURITY GROUP IDS USED IN THE VPC.

```
#create alb (Application load balancer)
resource "aws_lb" "roshan-my-alb" { #Elastic load balncing it is a service that we arre using and ima creating an application load balancer(aws_lb)
  name               = "roshan-my-alb"
  internal           = false # because flase means that we are making our alb as public if we want to make it as private then replace false with true.
  load_balancer_type = "application"

  security_groups = [aws_security_group.roshan_web_sg.id]
  subnets         = [aws_subnet.rosh_sub_1.id, aws_subnet.rosh_sub_2.id]

  tags = {
    Name = "ROSHAN_APPLICATION_LOAD_BALANCER_web"
  }
}
          aws_lb_target_group  hashicorp/aws 6.15.0
          Resource Type
resource "aws_lb_target_group" "roshanstg" {
  name     = "roshansTG"
  port     = 80
  protocol = "HTTP"
  vpc_id   = aws_vpc.rosh_vpc.id

  health_check {
    path = "/"
    port = "traffic-port"
  }
}
```

STEP 10:- I WILL CREATE THE TARGET GROUP FOR THE ALB SO THAT TRAFFIC CAN FLOW THROUGH THE INSTANCES OR SERVICES REQUIRED .WE ARE ALLOWING HTTP TRAFFIC FROM PORT 80.WHEN WE SAY PATH = “/” IT MEANS FROM ANY PATH.

```
resource "aws_lb_target_group" "roshanstg" {
  name     = "roshansTG"
  port     = 80
  protocol = "HTTP"
  vpc_id   = aws_vpc.rosh_vpc.id

  health_check {
    path = "/"
    port = "traffic-port"
  }
}
```

STEP 11:- WE HAVE TO ATTACH THE TARGET GROUPS TO THE EC2 INSTANCES IN THE DIFFERENT AVAILABILITY ZONES.

```
resource "aws_lb_target_group_attachment" "roshan_attach1" {
  target_group_arn = aws_lb_target_group.roshanstg.arn
  target_id        = aws_instance.ec2-instance-1.id
  port             = 80
}

resource "aws_lb_target_group_attachment" "roshan_attach2" {
  target_group_arn = aws_lb_target_group.roshanstg.arn
  target_id        = aws_instance.ec2-instance-2.id
  port             = 80
}
```

STEP 12:- JUST BY HAVING A TARGET GROUP AND ALB WE CANNOT ACEES ANYTHING SO THE ALB NEEDS TO LISTEN TO THE REQUESTS BEING MADE BY THE USER SO WE HAVE TO ADD A LISTNER FOR THE ALB.

```
resource "aws_lb_listener" "listener" {
  load_balancer_arn = aws_lb.roshan-my-alb.arn
  port              = 80
  protocol          = "HTTP"

  default_action {
    target_group_arn = aws_lb_target_group.roshanstg.arn
    type             = "forward"
  }
}
```

STEP 13:- TO GET THE DNS OF THE ALB WE CAN DIRECTLY PRINT IT ON OUR TERMINAL BY USING output module like this:-

```
output "loadbalancerdns" {# to get load balancer dns in terminal
  value = aws_lb.roshan-my-alb.dns_name
}
```

STEP 14:- PROVIDER.TF FILE WHICH MENTIONS THE PROVIDERS THAT WE ARE USING TO RUN THE PARTICULAR APPLICATION.

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "6.16.0"
    }
  }
}

provider "aws" {
  # Configuration options
  region = "ap-south-1"
}
```

STEP 15:-THE USER DATA FILES ARE :- THIS CREATE THE HTML PAGE AND NOTE WE ARE NOT USING ONLY CURL BUT INSTEAD USING IMDSv2 recently we cant use curl for ubuntu 22. + so iam using -X put and getting the SSh token for the session to get the metadata.

```
#!/bin/bash
apt update
apt install -y apache2

# Get the instance ID using the instance metadata here we are using -X put to enforce IMDSV2 this is where i got stuck becz for newer ec2 instances and ubuntu 22 + we need to enforce IMDSV2
TOKEN=$(curl -X PUT "http://169.254.169.254/latest/api/token" \
  -H "X-aws-ec2-metadata-token-ttl-seconds: 21600")
INSTANCE_ID=$(curl -H "X-aws-ec2-metadata-token: $TOKEN" \
  -s http://169.254.169.254/latest/meta-data/instance-id)


# Install the AWS CLI
apt install -y awscli

# Download the images from S3 bucket
#aws s3 cp s3://myterraformprojectbucket2023/project.webp /var/www/html/project.png --acl public-read

# Create a simple HTML file with the portfolio content and display the images
cat <<EOF > /var/www/html/index.html
<!DOCTYPE html>
<html>
<head>
  <title>My Portfolio</title>
  <style>
    /* Add animation and styling for the text */
    @keyframes colorChange {
      0% { color: red; }
      50% { color: green; }
      100% { color: blue; }
    }
    h1 {
      animation: colorChange 2s infinite;
    }
  </style>
</head>
<body>
  <h1>Roshans Terraform Project Server 1 RUNNING AT AP-SOUTH-1A(MUMBAI)</h1>
  <h2>Instance ID: <span style="color:green">$INSTANCE_ID</span></h2>
  <p>Welcome to roshans website in public subnet 1</p>

</body>
</html>
EOF

# Start Apache and enable it on boot
systemctl start apache2
systemctl enable apache2
```

```
TOKEN=$(curl -X PUT "http://169.254.169.254/latest/api/token" \
  -H "X-aws-ec2-metadata-token-ttl-seconds: 21600")
INSTANCE_ID=$(curl -H "X-aws-ec2-metadata-token: $TOKEN" \
  -s http://169.254.169.254/latest/meta-data/instance-id)
```

THIS IS THECHANGE THAT IAM USING TO GET THE METADATA FOR UBUNTU 22.04 AND ABOVE.DONT ONLY USE CURL IF WE USE ONLY CURL THEN WE ARE NOT GETTING THE INSTANCE ID IN THE WEBBROWSER.

STEP 16:- CREATE TWO USER DATAS:-

USER DATA 1:-

```
#!/bin/bash
apt update
apt install -y apache2

# Get the instance ID using the instance metadata here we are using -X put to enforce IMDSV2 this is where i got stuck becz for newer ec2 instances and ubuntu 22 + we need to enforce IMDSv2
TOKEN=$(curl -X PUT "http://169.254.169.254/latest/api/token" \
  -H "X-aws-ec2-metadata-token-ttl-seconds: 21600")
INSTANCE_ID=$(curl -H "X-aws-ec2-metadata-token: $TOKEN" \
  -s http://169.254.169.254/latest/meta-data/instance-id)


# Install the AWS CLI
apt install -y awscli

# Download the images from S3 bucket
#aws s3 cp s3://myterraformprojectbucket2023/project.webp /var/www/html/project.png --acl public-read

# Create a simple HTML file with the portfolio content and display the images
cat <<EOF > /var/www/html/index.html
<!DOCTYPE html>
<html>
<head>
  <title>My Portfolio</title>
  <style>
    /* Add animation and styling for the text */
    @keyframes colorChange {
      0% { color: red; }
      50% { color: green; }
      100% { color: blue; }
    }
    h1 {
      animation: colorChange 2s infinite;
    }
  </style>
</head>
<body>
  <h1>Roshans Terraform Project Server 1 RUNNING AT AP-SOUTH-1A(MUMBAI)</h1>
  <h2>Instance ID: <span style="color:green">$INSTANCE_ID</span></h2>
  <p>Welcome to roshans website in public subnet 1</p>

</body>
</html>
EOF

# Start Apache and enable it on boot
systemctl start apache2
systemctl enable apache2
```

USER DATA 2 :-

roshansuserdata2.sh

```
#!/bin/bash
apt update
apt install -y apache2

# Get the instance ID using the instance metadata here we are using -X put to enforce IMDSV2 this is where i got stuck becz for newer ec2 instances and ubuntu 22 + we need to enforce IMDSv2
TOKEN=$(curl -X PUT "http://169.254.169.254/latest/api/token" \
  -H "X-aws-ec2-metadata-token-ttl-seconds: 21600")
INSTANCE_ID=$(curl -H "X-aws-ec2-metadata-token: $TOKEN" \
  -s http://169.254.169.254/latest/meta-data/instance-id)


# Install the AWS CLI
apt install -y awscli

# Download the images from S3 bucket
#aws s3 cp s3://myterraformprojectbucket2023/project.webp /var/www/html/project.png --acl public-read

# Create a simple HTML file with the portfolio content and display the images
cat <<EOF > /var/www/html/index.html
<!DOCTYPE html>
<html>
<head>
  <title>My Portfolio</title>
  <style>
    /* Add animation and styling for the text */
    @keyframes colorChange {
      0% { color: red; }
      50% { color: green; }
      100% { color: blue; }
    }
    h1 {
      animation: colorChange 2s infinite;
    }
  </style>
</head>
<body>
  <h1>Roshans Terraform Project Server 2 RUNNING AT AP-SOUTH-1B(MUMBAI)</h1>
  <h2>Instance ID: <span style="color:green">$INSTANCE_ID</span></h2>
  <p>Welcome to roshans website in public subnet 1</p>

</body>
</html>
EOF

# Start Apache and enable it on boot
systemctl start apache2
systemctl enable apache2
```

AFTER THIS WE HAVE TO RUN THE FOLLOWING COMMANDS AND HAVE TO WAIT FOR SOMETIME WHILE USING terraform apply –auto-approve.

terraform plan
terraform validate
terraform apply –auto-approve

AFTER APPROVING AND SEEING UR WEBSITE PLS DELETE THE RESOURCES BECAUSE COST IS A FACTOR.TO DELETE ALL THE RESOURCES USE THE BELOW COMMAND:-

terraform destroy –auto-approve

when I do terraform plan in my terminal I get output as 16 resources to add

```
      + vpc_id                                     = (known after apply)
    }

  # aws_vpc.rosh_vpc will be created
  + resource "aws_vpc" "rosh_vpc" {
      + arn                                  = (known after apply)
      + cidr_block                           = "10.0.0.0/16"
      + default_network_acl_id               = (known after apply)
      + default_route_table_id               = (known after apply)
      + default_security_group_id            = (known after apply)
      + dhcp_options_id                      = (known after apply)
      + enable_dns_hostnames                 = (known after apply)
      + enable_dns_support                   = true
      + enable_network_address_usage_metrics = (known after apply)
      + id                                   = (known after apply)
      + instance_tenancy                     = "default"
      + ipv6_association_id                  = (known after apply)
      + ipv6_cidr_block                      = (known after apply)
      + ipv6_cidr_block_network_border_group = (known after apply)
      + main_route_table_id                  = (known after apply)
      + owner_id                             = (known after apply)
      + region                               = "ap-south-1"
      + tags_all                             = (known after apply)
    }

Plan: 16 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + loadbalancerdns = (known after apply)
```

THEN I USED terraform validate to check if there is any syntax error so this is very imp because we nee dto be carefull when we are making an infrastructure. So always use these commands first and then use the terraform apply –auto-approve command. We can also use terraform apply but will ask yes or no options and wates time instead u can add –auto-approve.

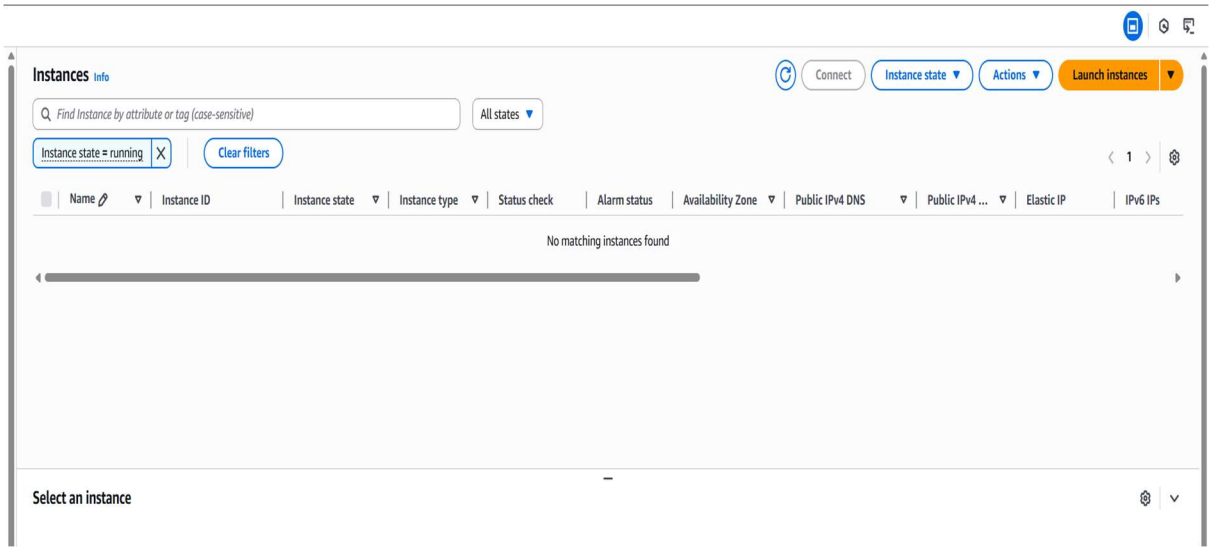
Output for terraform validate.

```
Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.
roshan@Roshan-S:/mnt/c/Users/rosha/OneDrive/Desktop/terraform_project_aws$ terraform validate
Success! The configuration is valid.

roshan@Roshan-S:/mnt/c/Users/rosha/OneDrive/Desktop/terraform_project_aws$
```

NOW IIAM GOING TO USE terraform apply –auto-approve and this will make surew that all my resources are created in the aws management console without me goijg to the aws management console don’t stop anything we have to wait for sometime in order to create the infrastructure . don’t stop anything. Depending on the network speed the infrastructure can be created fast.

```
roshan@Roshan-S:/mnt/c/Users/rosha/OneDrive/Desktop/terraform_project_aws$ terraform apply --auto-approve
```

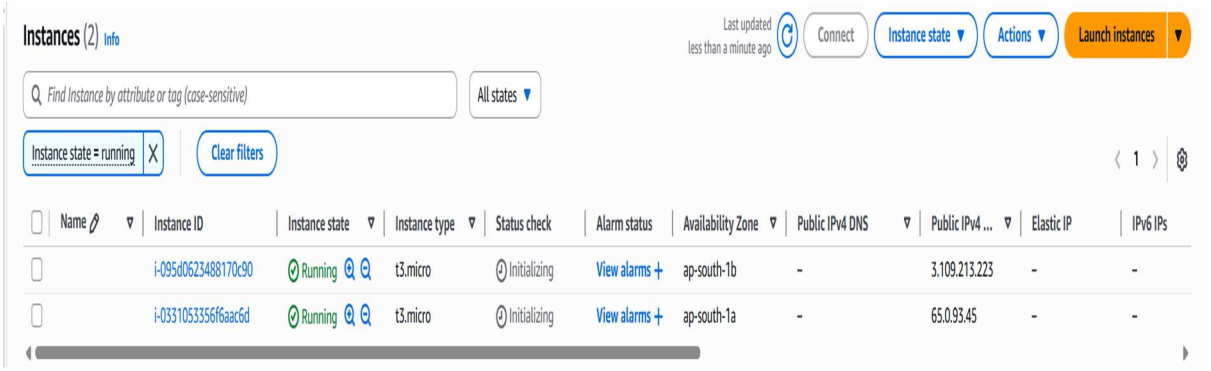
INITIALLY WE DON’T HAVE ANY EC2 INSTANCES RUNNING AND IT IS NOW EMPTY ONCE WE USE THIS COMMAND WE CAN SEE THAT THE EC2 INSTANCES WILL BE FORMED IN THE CONSOLE AND THE ENTIRE INFRASTRUCTURE IS CREATED WITHIN SECONDS.BUT WE NEED TO WAIT UNTIL THE ENTIRE SETUP IS MADE.




```
aws_vpc.rosh_vpc: Creating...
aws_s3_bucket.rosh_s3: Creating...
aws_vpc.rosh_vpc: Creation complete after 1s [id=vpc-04943a7677c462f04]
aws_internet_gateway.rosh_igw: Creating...
aws_subnet.rosh_sub_2: Creating...
aws_subnet.rosh_sub_1: Creating...
aws_lb_target_group.roshanstg: Creating...
aws_security_group.roshan_web_sg: Creating...
aws_s3_bucket.rosh_s3: Creation complete after 2s [id=roshan20252025-bucketcreation]
aws_internet_gateway.rosh_igw: Creation complete after 1s [id=igw-0a0fd2ca95f38866d]
aws_route_table.rosh_RT: Creating...
aws_lb_target_group.roshanstg: Creation complete after 1s [id=arn:aws:elasticloadbalancing:ap-south-1:518286664533:targetgroup/roshansTG/d25dee4ade8e5a31]
aws_route_table.rosh_RT: Creation complete after 1s [id=rtb-0d507f2e160e80b7d]
aws_security_group.roshan_web_sg: Creation complete after 3s [id=sg-097d241aad984e88e]
```

IT WILL SHOW LIKE THE ABOVE FIGURE IN THE TERMINAL.

ONCE THIS HAPPENS MY ENTIRE INFRASTRUCTURE IS CREATED AND WE CAN SEE THAT THE EC2 INSTANCES ARE CREATED INTIALLY IT WAS EMPTY AND NOW IT IS CREATED WITHIN SECONDS.




```
aws_route_table.rosh_RT: Creating...
aws_lb_target_group.roshanstg: Creation complete after 1s [id=arn:aws:elasticloadbalancing:ap-south-1:518286664533:targetgroup/roshansTG/d25dee4ade
aws_route_table.rosh_RT: Creation complete after 1s [id=rtb-0d507f2e160e80b7d]
aws_security_group.roshan_web_sg: Creation complete after 3s [id=sg-097d241aad984e88e]
aws_subnet.rosh_sub_2: Still creating... [10s elapsed]
aws_subnet.rosh_sub_1: Still creating... [10s elapsed]
aws_subnet.rosh_sub_2: Creation complete after 11s [id=subnet-0aee0dd71179db5ec]
aws_subnet.rosh_sub_1: Creation complete after 11s [id=subnet-0e054c31c087e1253]
aws_route_table_association.rosh_RT_1b: Creating...
aws_route_table_association.rosh_RT_1a: Creating...
aws_lb.roshan-my-alb: Creating...
aws_instance.ec2-instance-2: Creating...
aws_instance.ec2-instance-1: Creating...
aws_route_table_association.rosh_RT_1b: Creation complete after 1s [id=rtbassoc-0d5ad195f5a540044]
aws_route_table_association.rosh_RT_1a: Creation complete after 0s [id=rtbassoc-0bd5b75d224c9a6d7]
aws_lb.roshan-my-alb: Still creating... [10s elapsed]
aws_instance.ec2-instance-2: Still creating... [10s elapsed]
aws_instance.ec2-instance-1: Still creating... [10s elapsed]
aws_instance.ec2-instance-1: Creation complete after 12s [id=i-0331053356f6aac6d]
aws_instance.ec2-instance-2: Creation complete after 12s [id=i-095d0623488170c90]
aws_lb_target_group_attachment.roshan_attach1: Creating...
aws_lb_target_group_attachment.roshan_attach2: Creating...
aws_lb_target_group_attachment.roshan_attach1: Creation complete after 0s [id=arn:aws:elasticloadbalancing:ap-south-1:518286664533:targetgroup/rosh
aws_lb_target_group_attachment.roshan_attach2: Creation complete after 0s [id=arn:aws:elasticloadbalancing:ap-south-1:518286664533:targetgroup/rosh
aws_lb.roshan-my-alb: Still creating... [20s elapsed]
aws_lb.roshan-my-alb: Still creating... [30s elapsed]
aws_lb.roshan-my-alb: Still creating... [40s elapsed]
aws_lb.roshan-my-alb: Still creating... [50s elapsed]
aws_lb.roshan-my-alb: Still creating... [1m0s elapsed]
aws_lb.roshan-my-alb: Still creating... [1m10s elapsed]
aws_lb.roshan-my-alb: Still creating... [1m20s elapsed]
aws_lb.roshan-my-alb: Still creating... [1m30s elapsed]
aws_lb.roshan-my-alb: Still creating... [1m40s elapsed]
aws_lb.roshan-my-alb: Still creating... [1m50s elapsed]
```

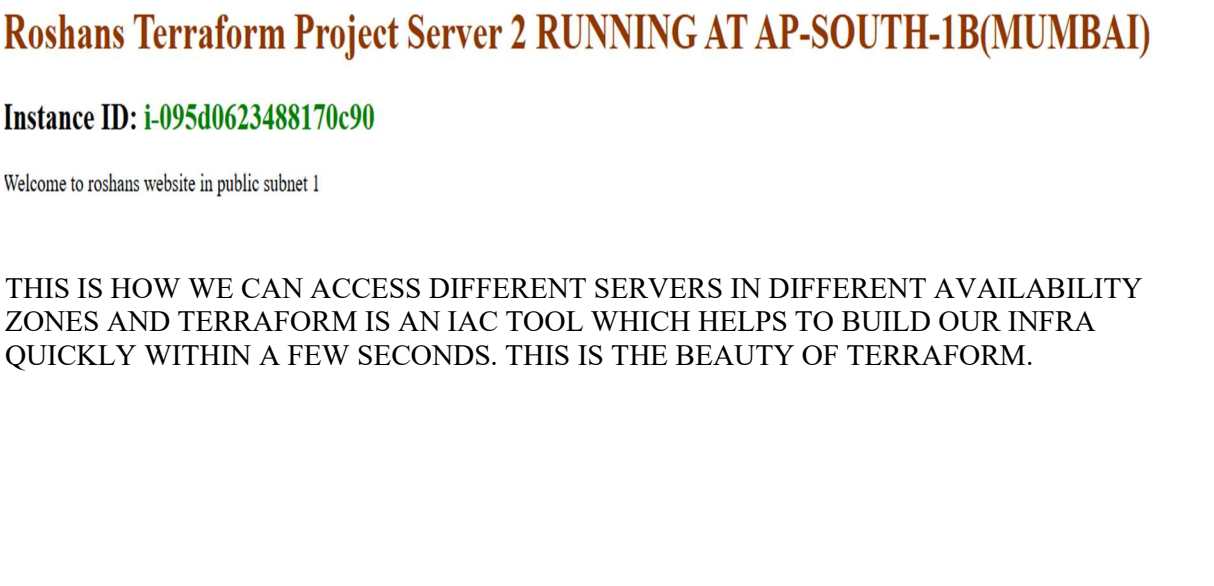
ALB WILLL TAKE SOME TIME TO CREATE. SO WE HAVE TO WAIT FOR THE ALB.NOW AFTER EVERYTHING IS OVER WE GET THIS MESSAGE THAT IS THE BELOW MESSAGE FROM THE TERMINAL:-

SO NOW IAM GOING TO ACCESS MY LOAD BALANCER DNS FROM THE TERMINAL ITSELF COPY THE URL AND PASTE IN A NEW TAB OR BROWSER. SO THAT WE CAN SEE THE APPLICATION

```
aws_instance.ec2-instance-2: Creation complete after 12s [id=i-095d0623488170c90]
aws_lb_target_group_attachment.roshan_attach1: Creating...
aws_lb_target_group_attachment.roshan_attach2: Creating...
aws_lb_target_group_attachment.roshan_attach1: Creation complete after 0s [id=arn:aws:elasticloadbalancing:ap-south-1:518286664533:targetgroup/roshansTG/d25dee4ade8e5a31-20251026134645096300000004]
aws_lb_target_group_attachment.roshan_attach2: Creation complete after 0s [id=arn:aws:elasticloadbalancing:ap-south-1:518286664533:targetgroup/roshansTG/d25dee4ade8e5a31-20251026134645202100000005]
aws_lb.roshan-my-alb: Still creating... [20s elapsed]
aws_lb.roshan-my-alb: Still creating... [30s elapsed]
aws_lb.roshan-my-alb: Still creating... [40s elapsed]
aws_lb.roshan-my-alb: Still creating... [50s elapsed]
aws_lb.roshan-my-alb: Still creating... [1m0s elapsed]
aws_lb.roshan-my-alb: Still creating... [1m10s elapsed]
aws_lb.roshan-my-alb: Still creating... [1m20s elapsed]
aws_lb.roshan-my-alb: Still creating... [1m30s elapsed]
aws_lb.roshan-my-alb: Still creating... [1m40s elapsed]
aws_lb.roshan-my-alb: Still creating... [1m50s elapsed]
aws_lb.roshan-my-alb: Still creating... [2m0s elapsed]
aws_lb.roshan-my-alb: Still creating... [2m10s elapsed]
aws_lb.roshan-my-alb: Still creating... [2m20s elapsed]
aws_lb.roshan-my-alb: Still creating... [2m30s elapsed]
aws_lb.roshan-my-alb: Still creating... [2m40s elapsed]
aws_lb.roshan-my-alb: Still creating... [2m50s elapsed]
aws_lb.roshan-my-alb: Still creating... [3m0s elapsed]
aws_lb.roshan-my-alb: Still creating... [3m10s elapsed]
aws_lb.roshan-my-alb: Creation complete after 3m12s [id=arn:aws:elasticloadbalancing:ap-south-1:518286664533:loadbalancer/app/roshan-my-alb/a802d4752250f760]
aws_lb_listener.listener: Creating...
aws_lb_listener.listener: Creation complete after 0s [id=arn:aws:elasticloadbalancing:ap-south-1:518286664533:listener/app/roshan-my-alb/a802d4752250f760/70d714362cee2240]

Apply complete! Resources: 16 added, 0 changed, 0 destroyed.

Outputs:

loadbalancerdns = "roshan-my-alb-160937813.ap-south-1.elb.amazonaws.com"
```

OUTPUT IS:-

ONCE WE REFRESH THE PAGE WE GET FOR SERVER 2 AND AGAIN WHEN WE REFRESH WE GET SERVER 1 WHICH IS IN A ROUND ROBIN FASHION.



THIS IS HOW WE CAN ACCESS DIFFERENT SERVERS IN DIFFERENT AVAILABILITY ZONES AND TERRAFORM IS AN IAC TOOL WHICH HELPS TO BUILD OUR INFRA QUICKLY WITHIN A FEW SECONDS. THIS IS THE BEAUTY OF TERRAFORM.

NOW I WILL DELETE MY RESOURCES USING THE COMMAND.

terraform destroy –auto-approve

it will clean up all the resources to save the cost.

We will get this as the output

```
aws_vpc.rosh_vpc: Destroying... [id=vpc-04943a7677c462f04]
aws_vpc.rosh_vpc: Destruction complete after 0s

Destroy complete! Resources: 16 destroyed.
```

CONCLUSION :- WHENVER WE HAVE COMPLEX INFRASTRUCTURES GO FOR TERRAFORM AS AN IAC TOOL IT S VERY HELPFUL AND HASHICORP HAS PROVIDED DOCUMNETATIONS WHICH HELPS US TO BUILD INFRASTRUCTURES.

CHALLENGES WAS THAT :-

WE HAD TO USE IMDSV2 AND CURL COMMAND WAS NOT WORKING SO FOR ANYTHING ABOVE UBUNTU 22. + WE HAVE TO ENFORCE IMDSV2 OTHERWISE WE CANNOT COLLECT THE META DATA THAT IS LET US SAY INSTANCE ID OF THE PARTICULAR INSTANCE IN DIFFERENT AVAILABILITY ZONES.

DIFFERENCE BETWEEN IMDSV2 AND IMDSV1:-

Feature / Aspect	**IMDSv1**	**IMDSv2**
Release Year	2010	2019
Request Type	Simple HTTP GET	Two-step process: HTTP PUT (token) + HTTP GET
Token Requirement	❌ No token required	✅ Requires session token (TTL-based)
Security Against SSRF Attacks	❌ Vulnerable	✅ Strong protection via token-based access
Default Behavior (as of 2023)	Disabled or restricted in many environments	Enabled and recommended by AWS
Terraform Compatibility	May work without extra config	Requires metadata_options block with http_tokens = "required"

Feature / Aspect	IMDSv1	IMDSv2
Command Example (curl)	curl http://169.254.169.254/latest/meta-data/	TOKEN=$(curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-metadata-token-ttl-seconds: 21600") curl -H "X-aws-ec2-metadata-token: $TOKEN" http://169.254.169.254/latest/meta-data/
Best Practice	Deprecated for new workloads	Recommended for all production environments