**Terraform Case Study- Creating an Architecture using Terraform on AWS**

You work as a DevOps Engineer in a leading software company. You have been asked to build an infrastructure safely and efficiently.

**The company’s requirements:**

1. Use AWS Cloud Provider and the software to be installed is Apache2

2. Use Ubuntu AMI

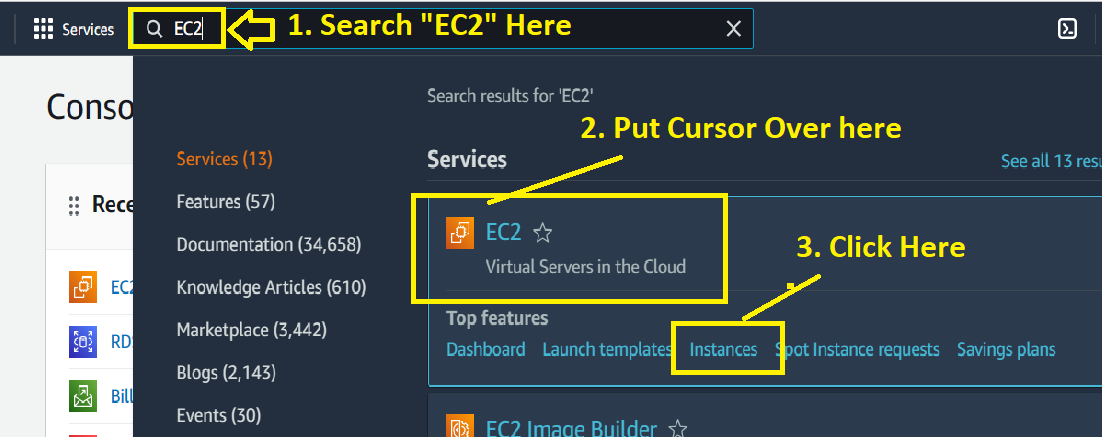
**The company wants the architecture to have the following services:**

1. Create a template with a VPC, 2 subnets and 1 instance in each subnet

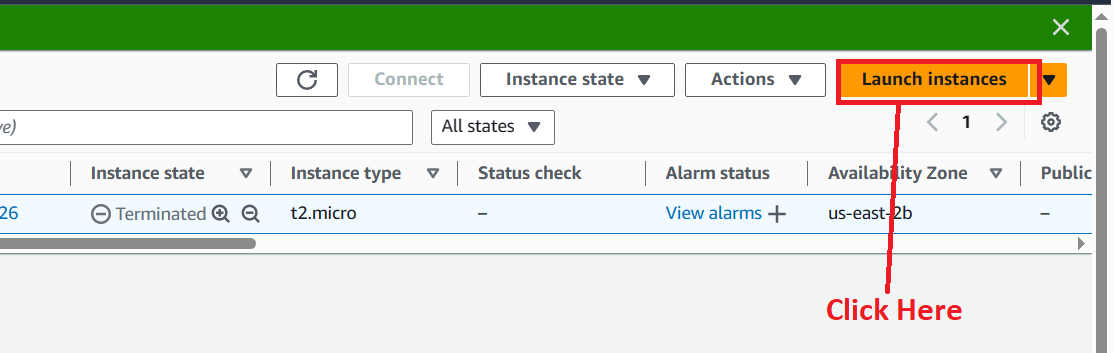
2. Attach security groups, internet gateway and network interface to the instance

**1. Create an EC2 Instance as “Master”**

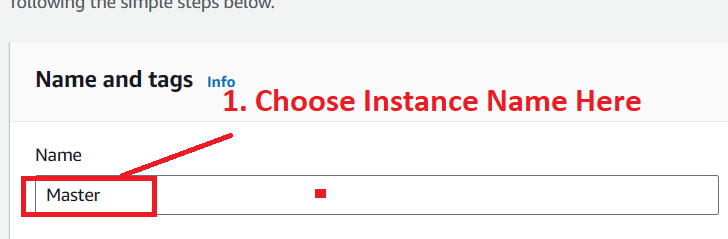
**Step 1: Go** tothe **“Services”** section& **search** the **“EC2”** here.



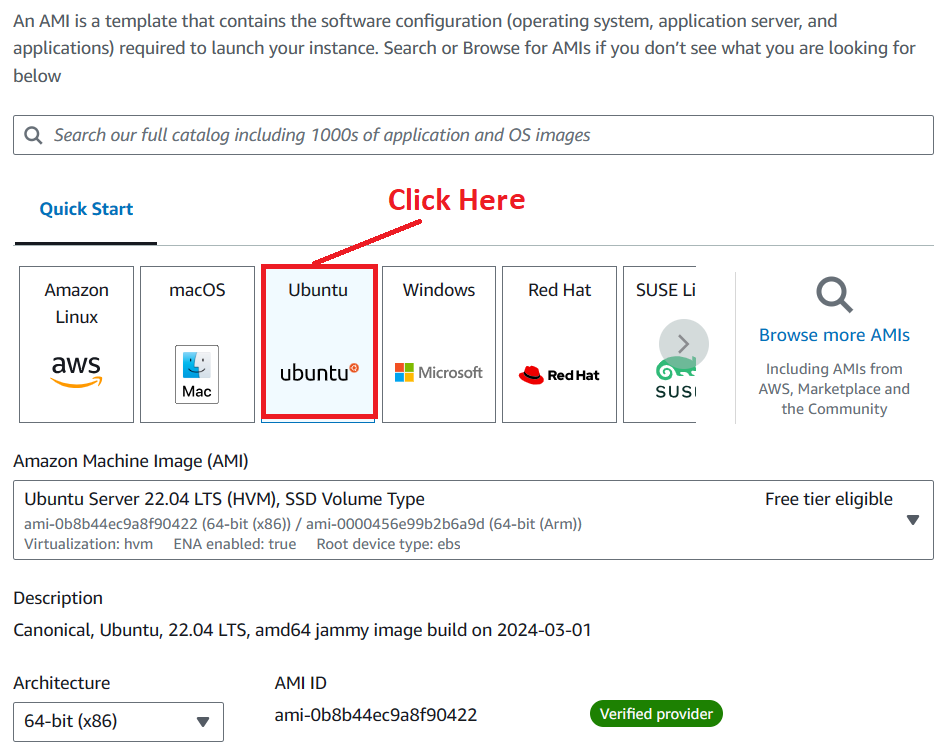
**Step 2: Click** on the **“Launch Instance”.**

****

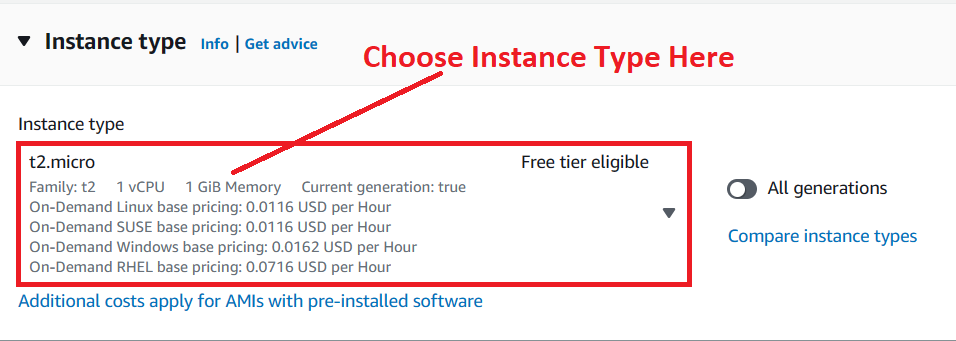
**Step 3: Choose** the **“Name”** as **“Master”** in the **“Name and tags”** section.



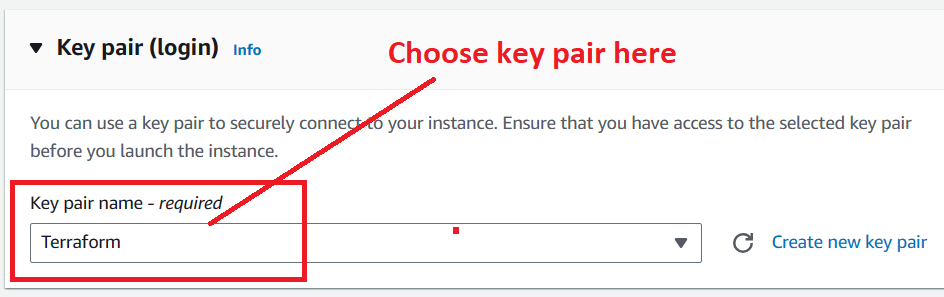
**Step 4: Choose** the **“AMI”** as **“ubuntu”.**

****

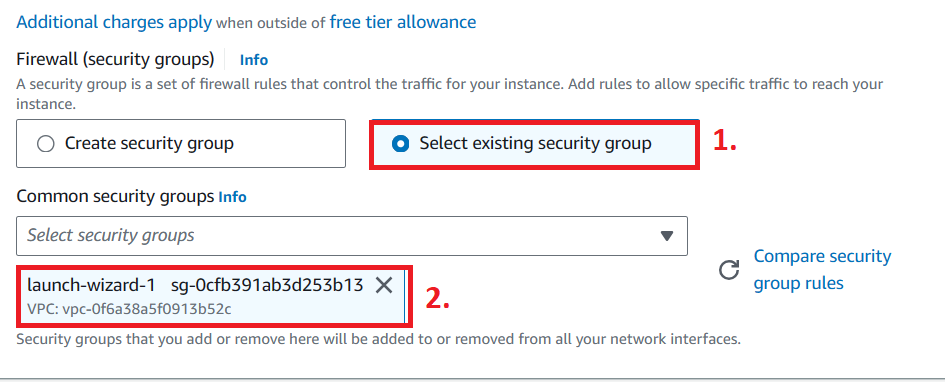
**Step 5: Choose** the **“Instance type”** asthe **“t2.micro”.**

****

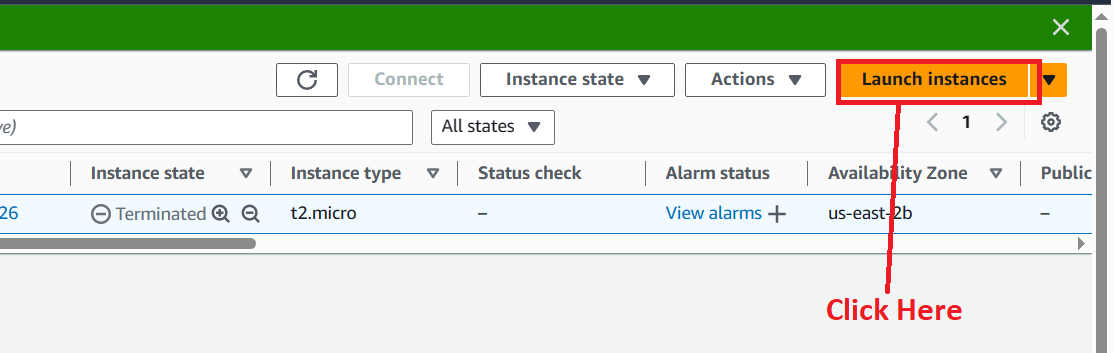
**Step 6: Choose** the **“key pair (login)”** as the **“Terraform”.**

****

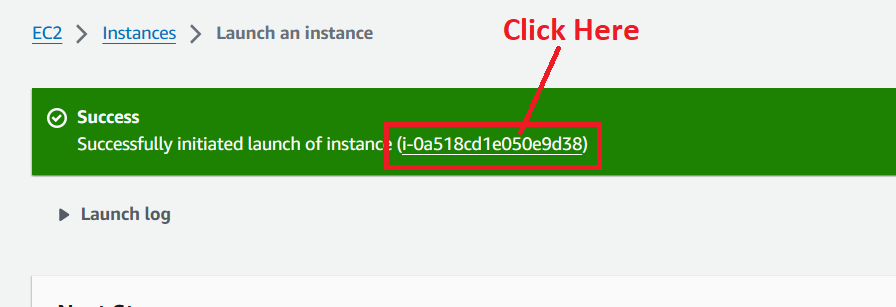
**Step 7: Choose** the **“Select existing security group”** asthe **“launch-wizard-1”.**

****

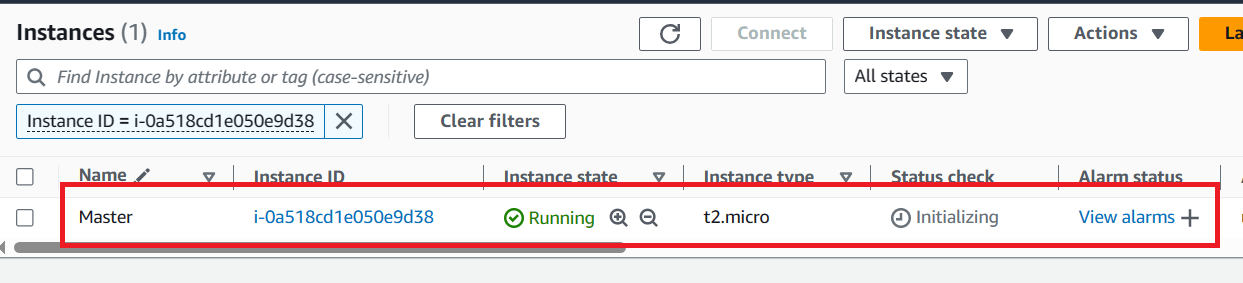
**Step 8: Click** onthe **“Launch Instance”.**

****

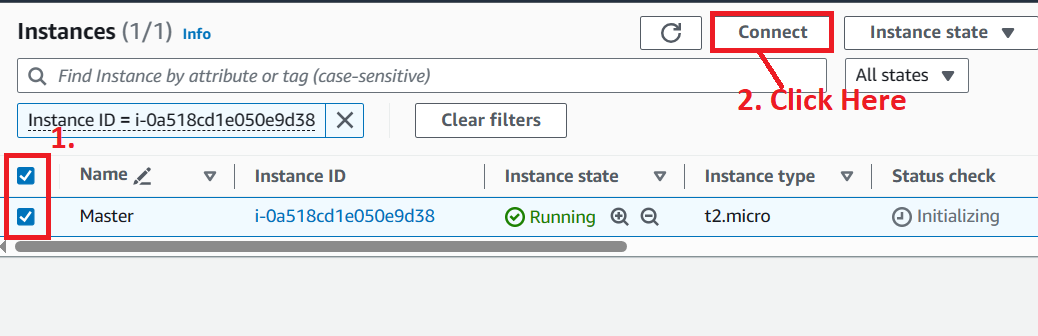
**Step 9: The instance** will be **launched. Click** onthe **“hyperlink”.**

****

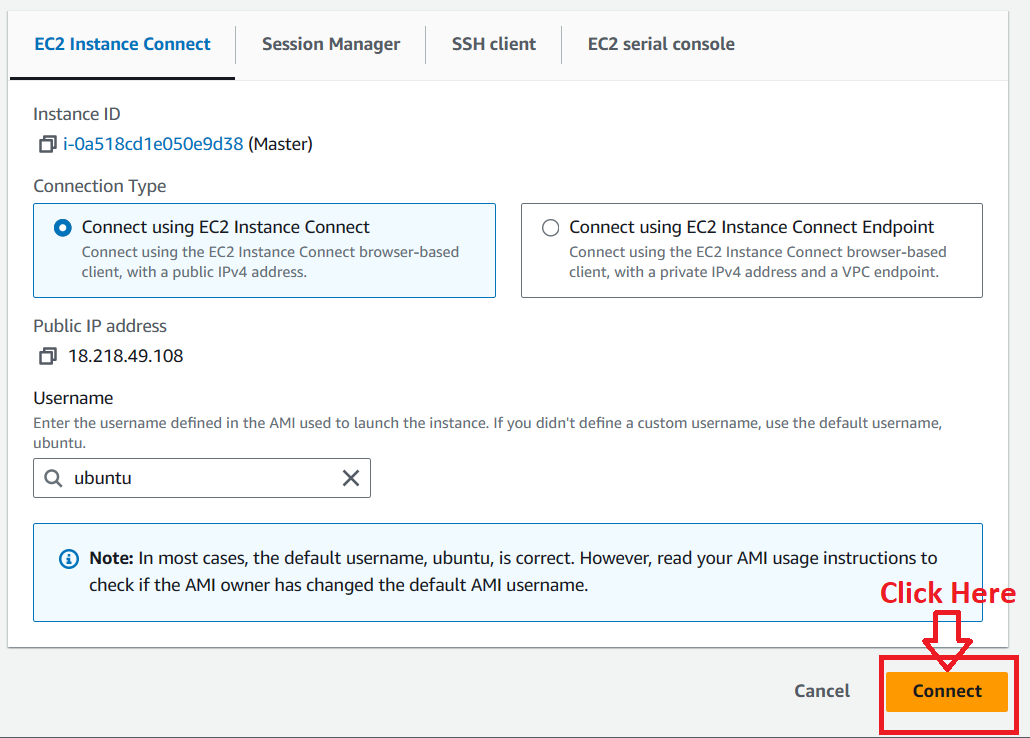
**Step 10: The instance** will be **in** the **“Running” State.**

****

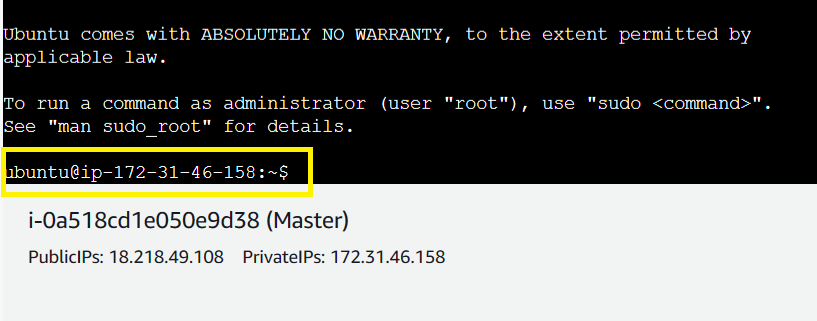
**Step 11: Select** the **instance & click** on the **“Connect”.**

****

**Step 12: Click** onthe **“Connect” again.**

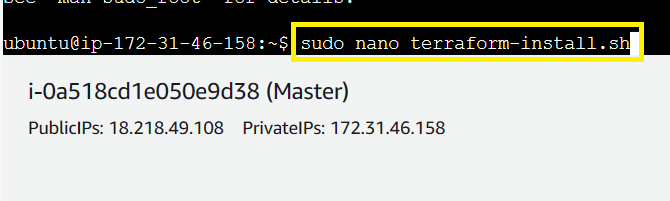
****

**Step 13: The instance** will be **successfully connected.**

****

**2. Install the “Terraform” on “Master”**

**Step 1: Install** the **Terraform, create** an **“terraform-install.sh” file** to **install Terraform using** this **command: sudo nano terraform\_install.sh**

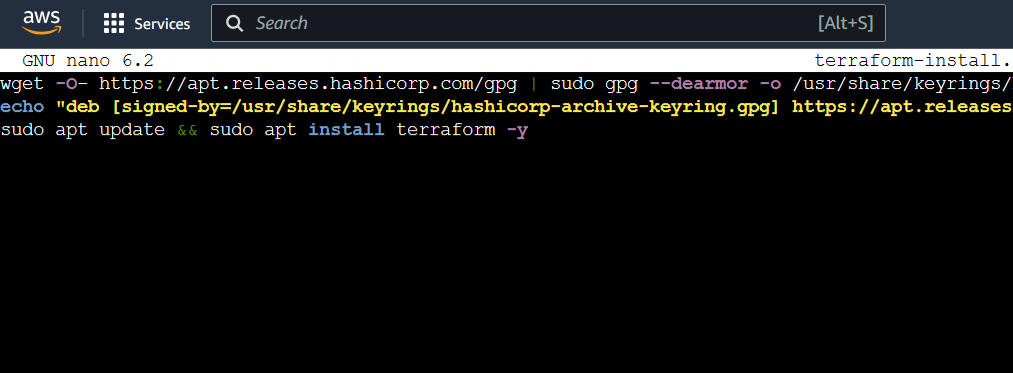
****

**Step 2: Paste this Script here:**

**wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg**

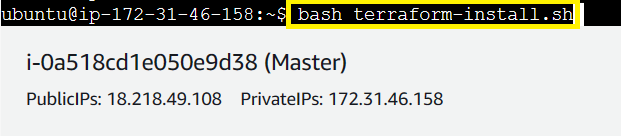
**echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb\_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list**

**sudo apt update && sudo apt install terraform -y**

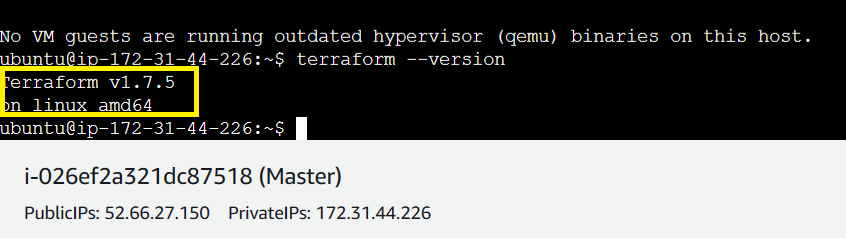
****

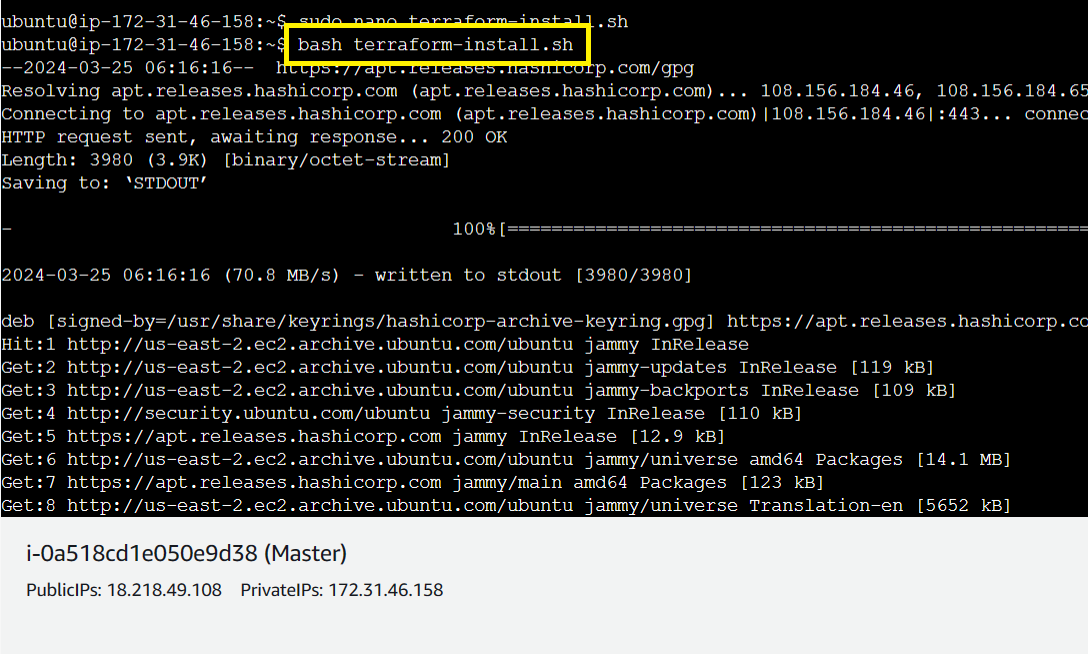
**Save & exit** from the **file. Do** the **CTRL+X** to **exit** & **press “Yes”** to **save. Press “enter”** to **completely exit** from the **file.**

**Step 3: Run** the **“bash terraform-install.sh” command** to **install** the **terraform**

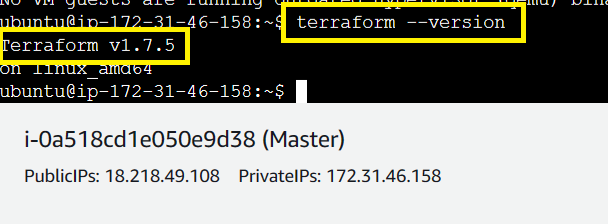
****

**Step 4: The terraform** will be **successfully installed.**

****

****

**Step 5: Run** this **command: terraform --version** to **check** the **“Terraform Version”.**

****

**3. Create a Script for VPC, 2 Subnets, Security Group, Elastic Network Interface & Internet Gateway with 2 Instances.**

**A. Create a VPC**

# Create a VPC

**resource "aws\_vpc" "testvpc" {**

**cidr\_block = "10.0.0.0/16"**

**tags = {**

**Name = "testvpc"**

**}**

**}**

**B. Create a Public & Private Subnet**

# Create a Public Subnet

**resource "aws\_subnet" "testsbnt1" {**

**vpc\_id = aws\_vpc.testvpc.id**

**cidr\_block = "10.0.1.0/24"**

**map\_public\_ip\_on\_launch = "true"**

**availability\_zone = "us-east-2a"**

**tags = {**

**Name = "testsbnt1"**

**}**

**}**

# Create a Private Subnet

**resource "aws\_subnet" "testsbnt2" {**

**vpc\_id = aws\_vpc.testvpc.id**

**cidr\_block = "10.0.2.0/24"**

**map\_public\_ip\_on\_launch = "false"**

**availability\_zone = "us-east-2b"**

**tags = {**

**Name = "testsbnt2"**

**}**

**}**

**C. Create an Internet Gateway**

# Create an Internet Gateway

**resource "aws\_internet\_gateway" "testigw" {**

**vpc\_id = aws\_vpc.testvpc.id**

**tags = {**

**Name = "testigw"**

**}**

**}**

**D. Create Route Tables for Public & Private Subnets**

# Create a Route Table for Public Subnet

**resource "aws\_route\_table" "testrtb1" {**

**vpc\_id = aws\_vpc.testvpc.id**

**route {**

**cidr\_block = "0.0.0.0/0"**

**gateway\_id = aws\_internet\_gateway.testigw.id**

**}**

**route {**

**ipv6\_cidr\_block = "::/0"**

**gateway\_id = aws\_internet\_gateway.testigw.id**

**}**

**}**

# Associate Public Route Table with Public Subnet

**resource "aws\_route\_table\_association" "testassoc1" {**

**subnet\_id = aws\_subnet.testsbnt1.id**

**route\_table\_id = aws\_route\_table.testrtb1.id**

**}**

# Create a Private Route Table for Subnet 2

**resource "aws\_route\_table" "testrtb2" {**

**vpc\_id = aws\_vpc.testvpc.id**

**route {**

**cidr\_block = "0.0.0.0/0"**

**gateway\_id = aws\_nat\_gateway.nat.id**

**}**

**tags = {**

**Name = "testrtb2"**

**}**

**}**

# Associate Route Table with Private Subnet

**resource "aws\_route\_table\_association" "testassoc2" {**

**subnet\_id = aws\_subnet.testsbnt2.id**

**route\_table\_id = aws\_route\_table.testrtb2.id**

**}**

**E. Create Elastic Network Interface with Elastic IP**

# Assign ENI with IP

**resource "aws\_network\_interface" "testeni1" {**

**subnet\_id = aws\_subnet.testsbnt1.id**

**private\_ips = ["10.0.1.10"]**

**security\_groups = [aws\_security\_group.testsg.id]**

**}**

**resource "aws\_network\_interface" "testeni2" {**

**subnet\_id = aws\_subnet.testsbnt2.id**

**private\_ips = ["10.0.2.10"]**

**security\_groups = [aws\_security\_group.testsg.id]**

**}**

# Assign Elastic IP to ENI

**resource "aws\_eip" "testeip1" {**

**domain = "vpc"**

**network\_interface = aws\_network\_interface.testeni1.id**

**associate\_with\_private\_ip = "10.0.1.10"**

**depends\_on= [aws\_internet\_gateway.testigw, aws\_instance.Instance1]**

**tags = {**

**Name = "testeip1"**

**}**

**}**

# Create an Elastic IP Address for NAT Gateway

**resource "aws\_eip" "testeip2" {**

**domain = "vpc"**

**associate\_with\_private\_ip = "10.0.2.10"**

**depends\_on= [aws\_internet\_gateway.testigw]**

**tags = {**

**Name = "testeip2"**

**}**

**}**

**F. Create a NAT Gateway for VPC**

# Create a NAT Gateway for VPC

**resource "aws\_nat\_gateway" "nat" {**

**allocation\_id = aws\_eip.testeip2.id**

**subnet\_id = aws\_subnet.testsbnt2.id**

**tags = {**

**Name = "nat"**

**}**

**}**

**G. Create a Security Group for Instance**

# Create a Security Group

**resource "aws\_security\_group" "testsg" {**

**description = "Allow limited inbound external traffic"**

**vpc\_id = aws\_vpc.testvpc.id**

**name = "testsg"**

**ingress {**

**protocol = "tcp"**

**cidr\_blocks = ["0.0.0.0/0"]**

**from\_port = 22**

**to\_port = 22**

**}**

**ingress {**

**protocol = "tcp"**

**cidr\_blocks = ["0.0.0.0/0"]**

**from\_port = 80**

**to\_port = 80**

**}**

**ingress {**

**protocol = "tcp"**

**cidr\_blocks = ["0.0.0.0/0"]**

**from\_port = 443**

**to\_port = 443**

**}**

**egress {**

**from\_port = 0**

**to\_port = 0**

**protocol = "-1"**

**cidr\_blocks = ["0.0.0.0/0"]**

**}**

**tags = {**

**Name = "testsg"**

**}**

**}**

**H. Create Linux Server & Install/Enable Apache2 on Both Instances**

# Create Linux Server & Install/Enable Apache2 (Instance 1)

**resource "aws\_instance" "Instance1" {**

**ami = "ami-0b8b44ec9a8f90422"**

**instance\_type = "t2.micro"**

**availability\_zone = "us-east-2a"**

**key\_name = "Terraform"**

**network\_interface {**

**device\_index = 0**

**network\_interface\_id = aws\_network\_interface.testeni1.id**

**}**

**user\_data = <<-EOF**

**#!/bin/bash**

**sudo apt update -y**

**sudo apt install apache2 -y**

**sudo systemctl start apache2**

**sudo systemctl enable apache2**

**EOF**

**tags = {**

**Name = "Instance1"**

**}**

**}**

# Create Linux Server & Install/Enable Apache2 Here (Instance 2)

**resource "aws\_instance" "Instance2" {**

**ami = "ami-0b8b44ec9a8f90422"**

**instance\_type = "t2.micro"**

**availability\_zone = "us-east-2b"**

**key\_name = "Terraform"**

**network\_interface {**

**device\_index = 0**

**network\_interface\_id = aws\_network\_interface.testeni2.id**

**}**

**user\_data = <<-EOF**

**#!/bin/bash**

**sudo apt update -y**

**sudo apt install apache2 -y**

**sudo systemctl start apache2**

**sudo systemctl enable apache2**

**EOF**

**tags = {**

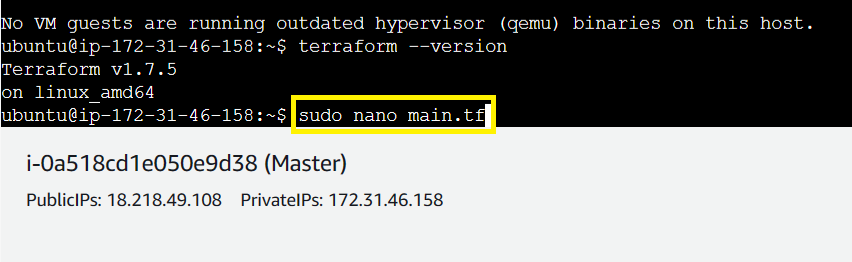
**Name = "Instance2"**

**}**

**}**

**4. Run the Created Terraform Script**

**Step 1: Create a “main.tf” file using the command: sudo nano main.tf**

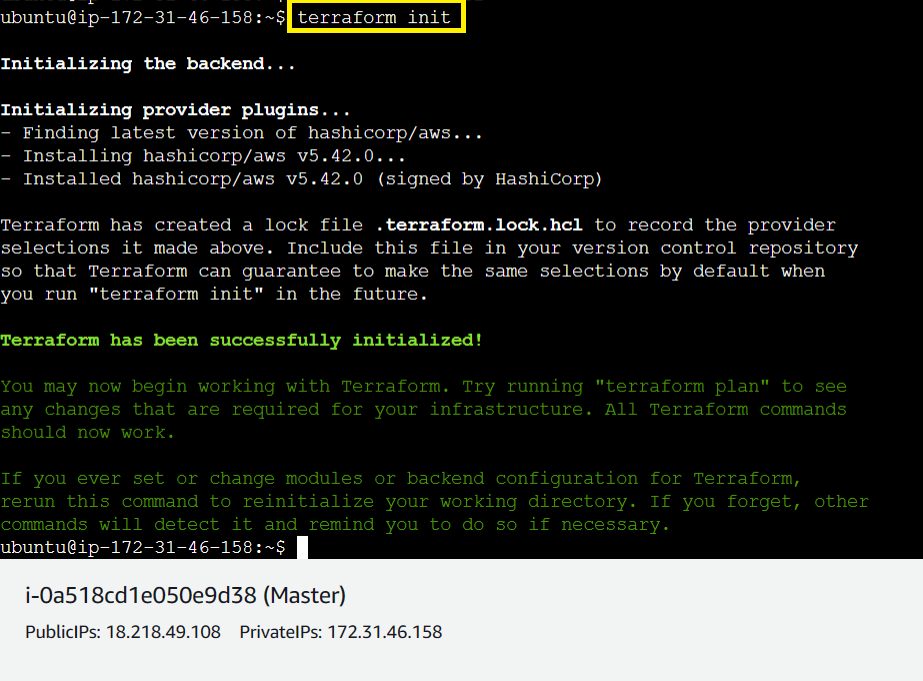
****

**Step 2: Paste the Created Script here:**

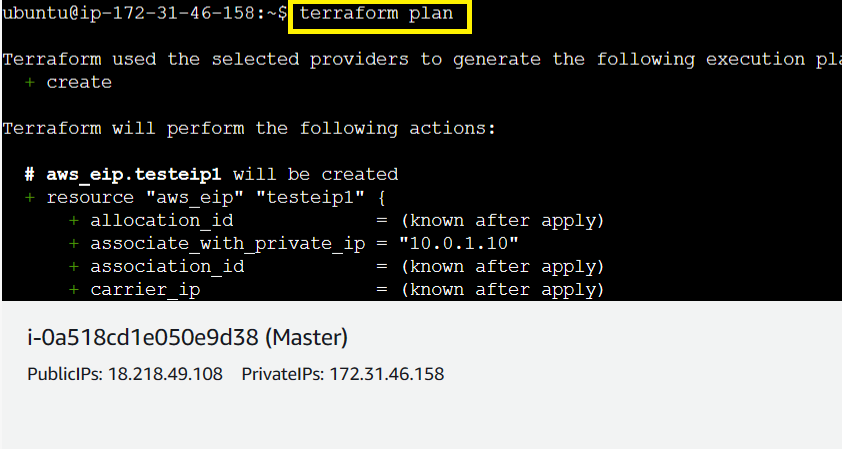
****

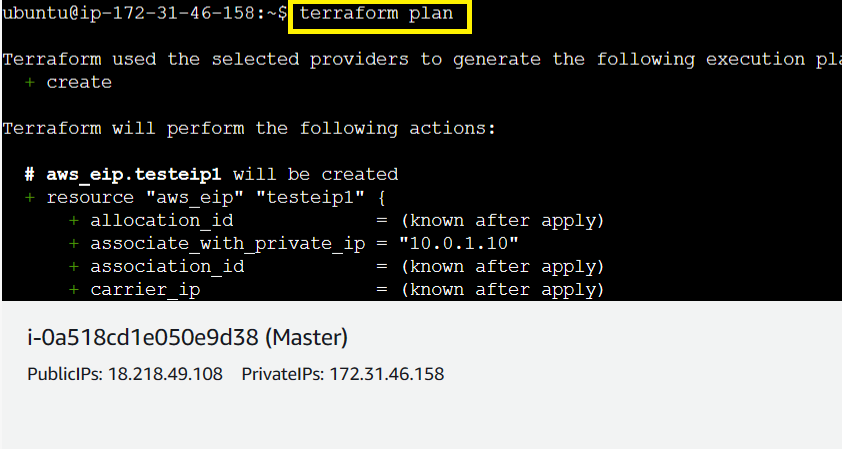
**Save & exit from the file. Do “CTRL+X” to exit** & **press “Yes”** to **save. Press “enter”** to **completely exit** from the **file.**

**Step 3: Initialize** the **“Terraform” using** the **command: terraform init.**

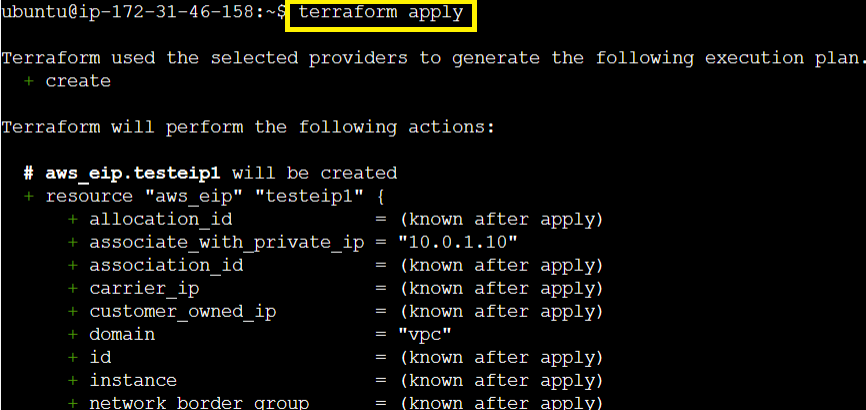
****

**Step 4: Run** the **“terraform plan” command** to **execute the plan.**

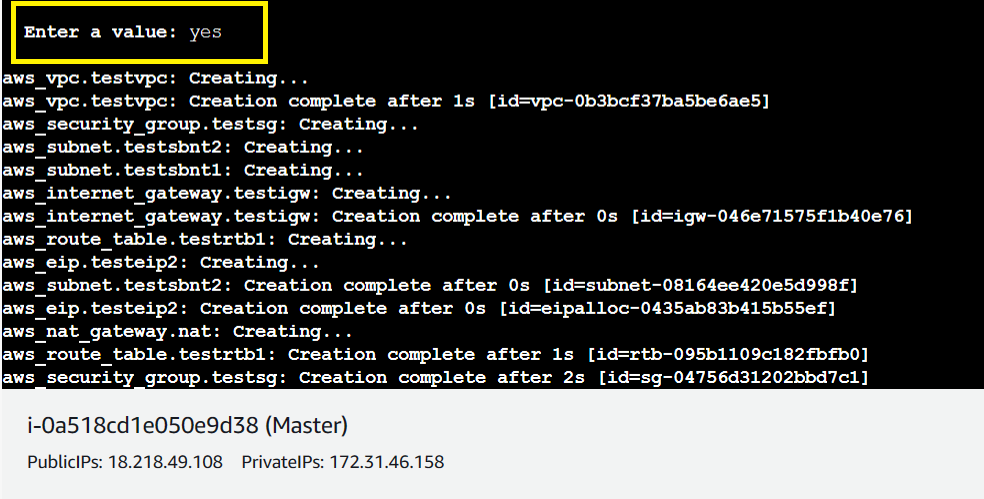
****

****

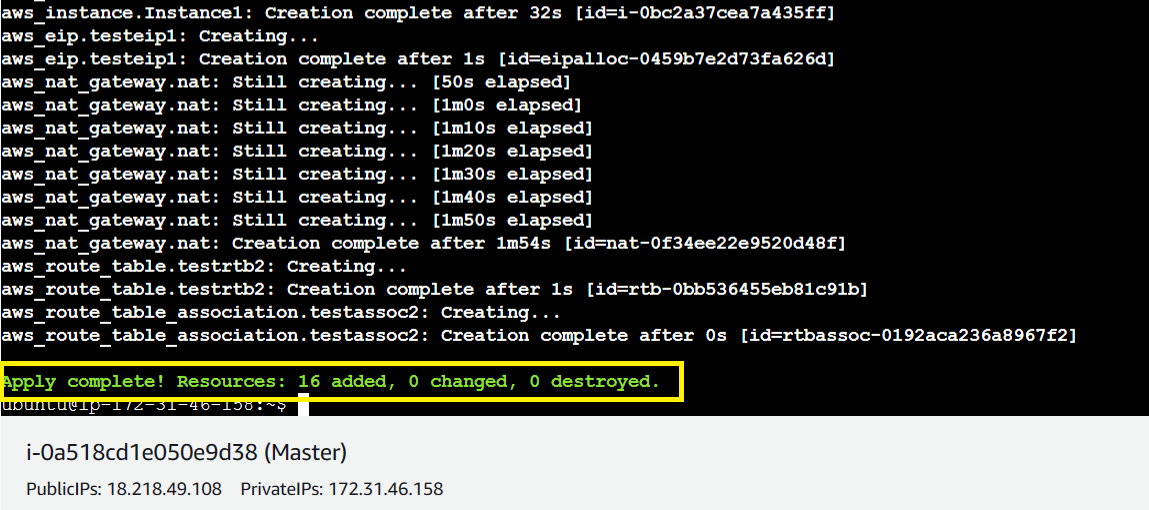
**Step 5: Now, run** the **“terraform apply” command** to **create** the **infrastructure.**

****

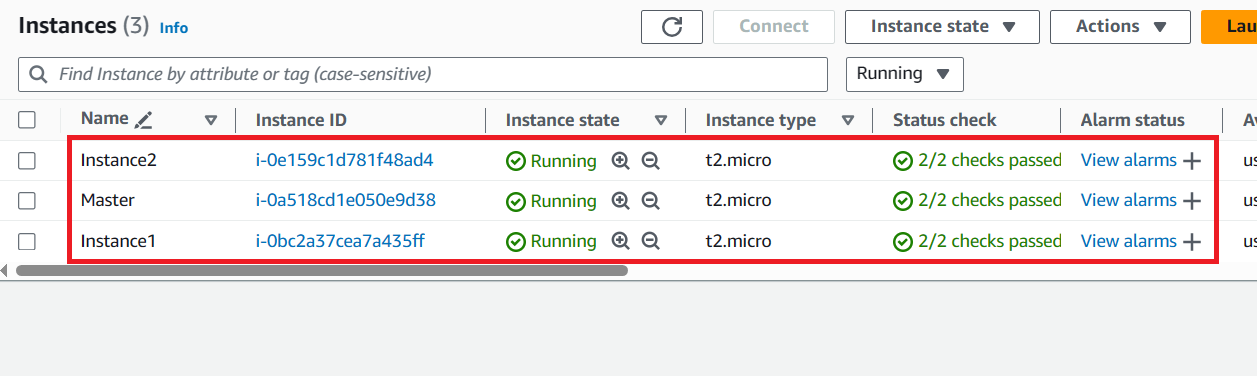
**Step 6: Type “yes”** to **continue.**

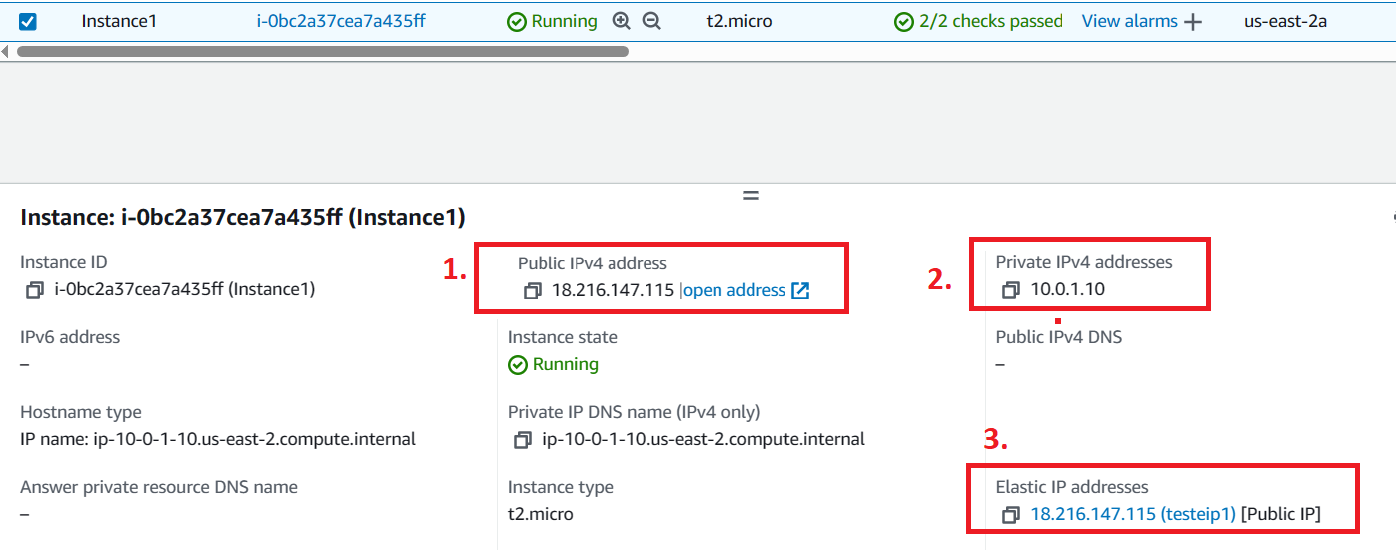
****

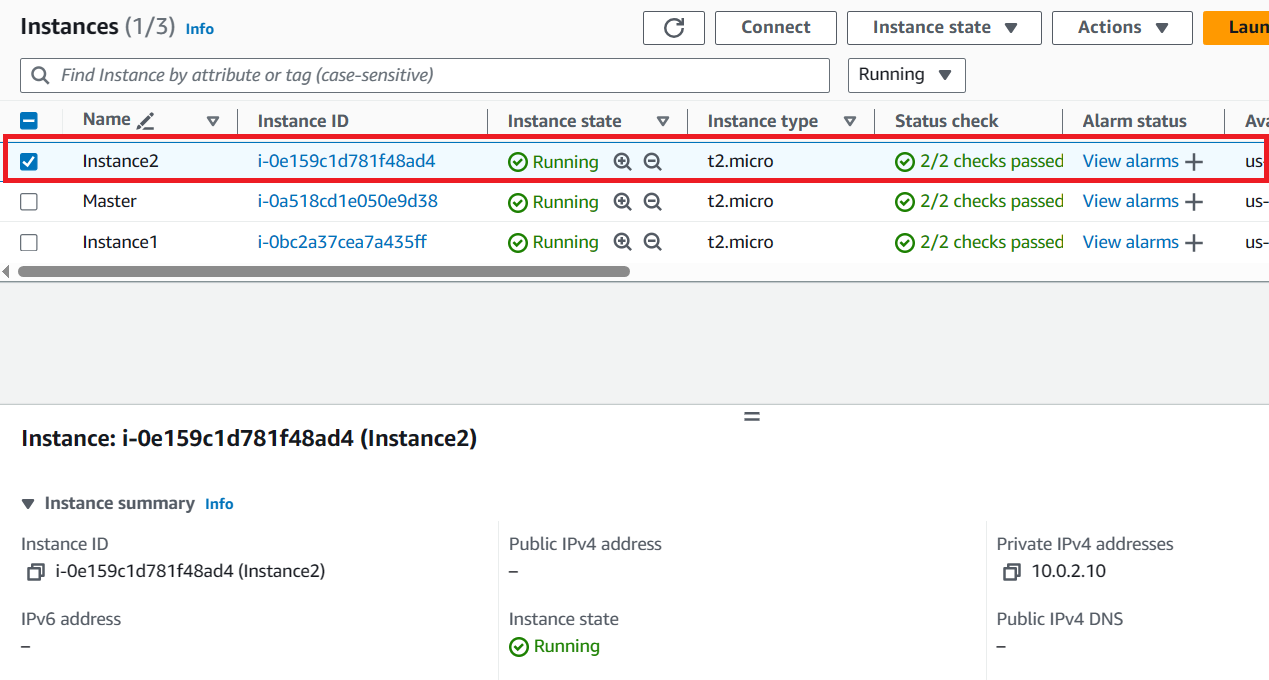
**Step 7: All** the **resources** will be **successfully created.**

****

**Step 8: All** the **instances** will be **successfully created.**

****

****

****

**Step 9: Click** onthe **“Open Address”** inthe **“Public IP Address (18.216.147.115)”.**

**The “Apache2”** will be **successfully installed”.**

****

**While, on** other **instance, “Apache2” page** will not **be shown because** of **“Public IP Address”** is not **assigned.**

**🡨---------------------------Terraform Case Study Solution Completed--------🡪**