**Using aws terraform code to deploy sample nginx/python application.**

* **Build docker image for sample nginx/python and push to aws elastic container registry(ECR) using docker cli.**
* **Create VPC with 1 public and private subnet, Security Groups and EC2 instance using terraform.**
* **Deploy Nginx Application as Docker Container using User Data script using terraform. Image should be pulled from ECR.**
* **Outcome is able to access the nginx website with url** [Error! Hyperlink reference not valid.](http://<public_ip>:<port>)**.**
* **Keep the code stuff into git repository. - Document all the execution steps**

**Services Used:**

* IAM
* ECR
* EC2 instance

**Tools used :**

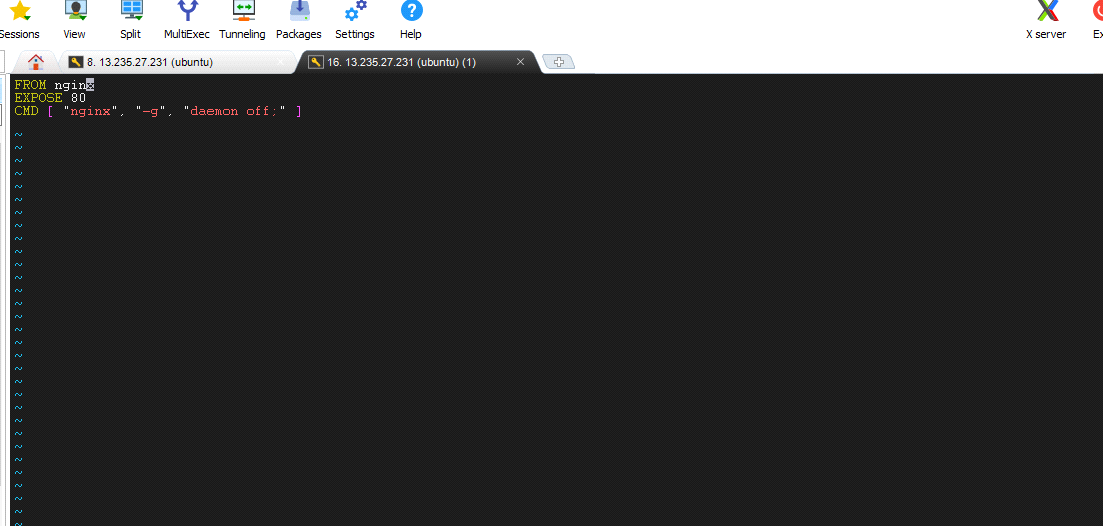
* Docker
* Terraform
* Git

**Step 1:**

* we created 1 EC2 instances installed Docker and Terraform and AWS\_CLI
* Next we create IAM user for AWS credentials and AWS ECR for storing image

**Next we create Dockerfile**

FROM nginx EXPOSE 8080



**Commands:**

* **For starting docker Service**

sudo systemctl start docker

* **Authenticate Docker with your ECR repository**

aws ecr-public get-login-password --region us-east-1 | sudo docker login -- username AWS --password-stdin public.ecr.aws/o1m3s8m0

* **Build the docker image**

sudo docker build -t student .

* **Image to Container**

sudo docker run -d -it student

* **Tag the Docker image with the ECR repository**

docker tag student:latest public.ecr.aws/o1m3s8m0/ahmed:latest

* **Push the Docker image to ECR**

Sudo docker push 078307860773.dkr.ecr.ap-south- 1.amazonaws.com/final-assessment-1:latest

Image is pushed to the ECR

**Step 2:**

Code: main.tf

provider "aws" {

region = "ap-south-1"

}

resource "aws\_vpc" "myVpc1" {

cidr\_block = "10.0.0.0/24"

}

data "aws\_availability\_zones" "available\_zones" {}

resource "aws\_subnet" "publicSubnet1" {

vpc\_id = aws\_vpc.myVpc1.id

cidr\_block = "10.0.0.0/25"

availability\_zone = data.aws\_availability\_zones.available\_zones.names[0]

tags = {

Name = "publicSubnet1"

}

}

resource "aws\_subnet" "privateSubnet1" {

vpc\_id = aws\_vpc.myVpc1.id

cidr\_block = "10.0.0.128/25"

availability\_zone = data.aws\_availability\_zones.available\_zones.names[1]

tags = {

Name = "privateSubnet1"

}

}

resource "aws\_internet\_gateway" "myIGW1" {

vpc\_id = aws\_vpc.myVpc1.id

tags = {

Name = "myIGW1"

}

}

resource "aws\_route\_table" "myPublicRoute" {

vpc\_id = aws\_vpc.myVpc1.id

route {

cidr\_block = "0.0.0.0/0"

gateway\_id = aws\_internet\_gateway.myIGW1.id

}

tags = {

Name = "myRoute"

}

}

// Associate subnet with the route table

resource "aws\_route\_table\_association" "myPublicRouteAssociate" {

subnet\_id = aws\_subnet.publicSubnet1.id

route\_table\_id = aws\_route\_table.myPublicRoute.id

}

resource "aws\_security\_group" "mySecureGrp" {

name = "mySecureGrp"

vpc\_id = aws\_vpc.myVpc1.id

ingress {

from\_port = 22

to\_port = 22

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

}

ingress {

from\_port = 8080

to\_port = 8080

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

protocol = "-1"

cidr\_blocks = ["0.0.0.0/0"]

ipv6\_cidr\_blocks = ["::/0"]

}

tags = {

Name = "mySecureGrp"

}

}

resource "aws\_instance" "Project" {

ami = "ami-0f5ee92e2d63afc18"

instance\_type = "t2.micro"

key\_name = "fyroz"

subnet\_id = aws\_subnet.publicSubnet1.id

vpc\_security\_group\_ids = [aws\_security\_group.mySecureGrp.id]

associate\_public\_ip\_address = true

user\_data = <<-EOF

#!/bin/bash

echo "hello world!" > hello.txt

sudo apt-get update -y

sudo apt-get install docker.io -y

curl "<https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip>" -o "awscliv2.zip"

unzip awscliv2.zip

sudo ./aws/install

aws ecr-public get-login-password --region us-east-1 | docker login --username AWS --password-stdin public.ecr.aws/e6v4n7j9

sudo docker pull public.ecr.aws/e6v4n7j9/ecr:latest

sudo docker run -d -p 8080:80 public.ecr.aws/e6v4n7j9/ecr:latest

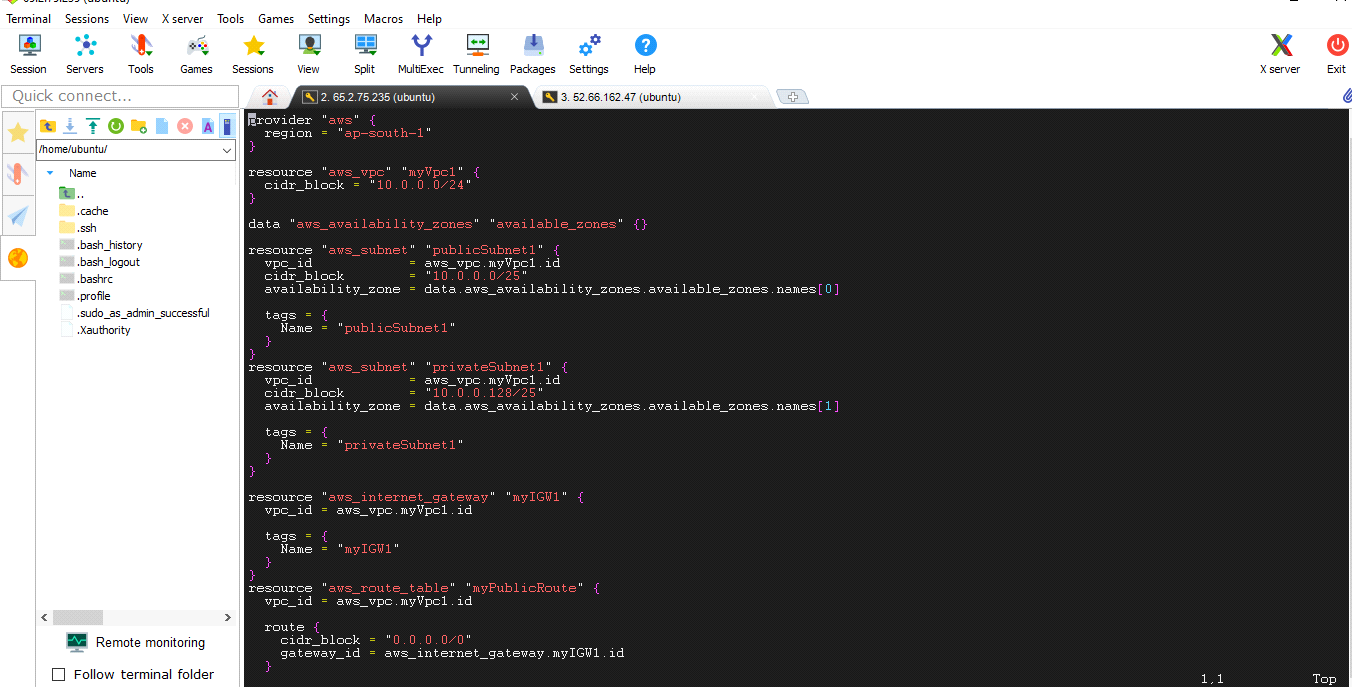
EOF

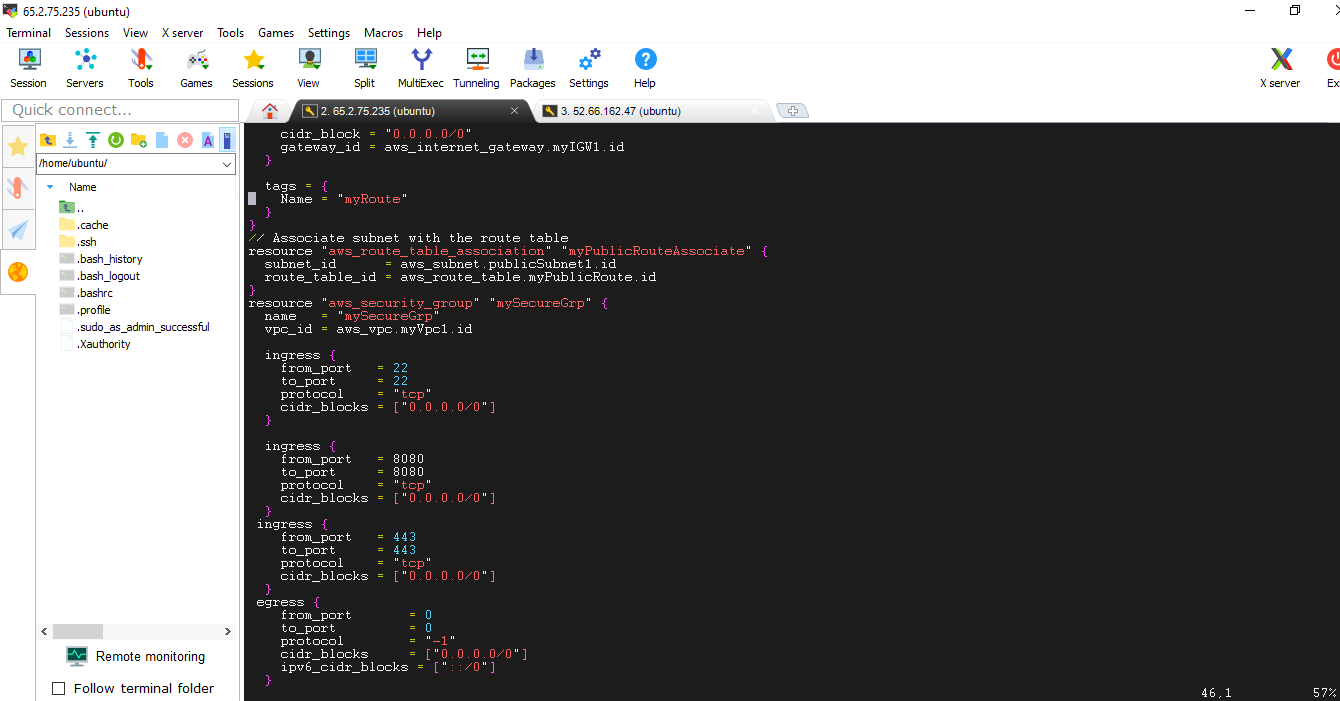
tags = {

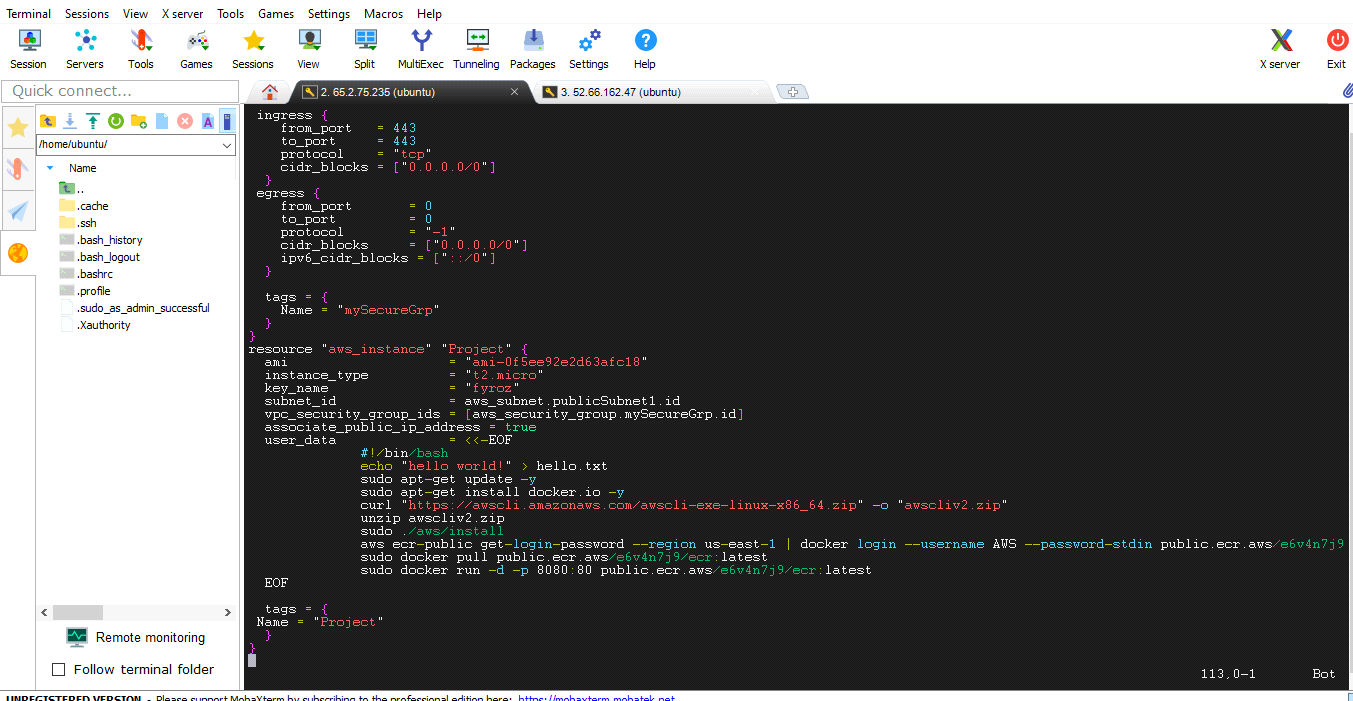
Name = "Project"

}

}







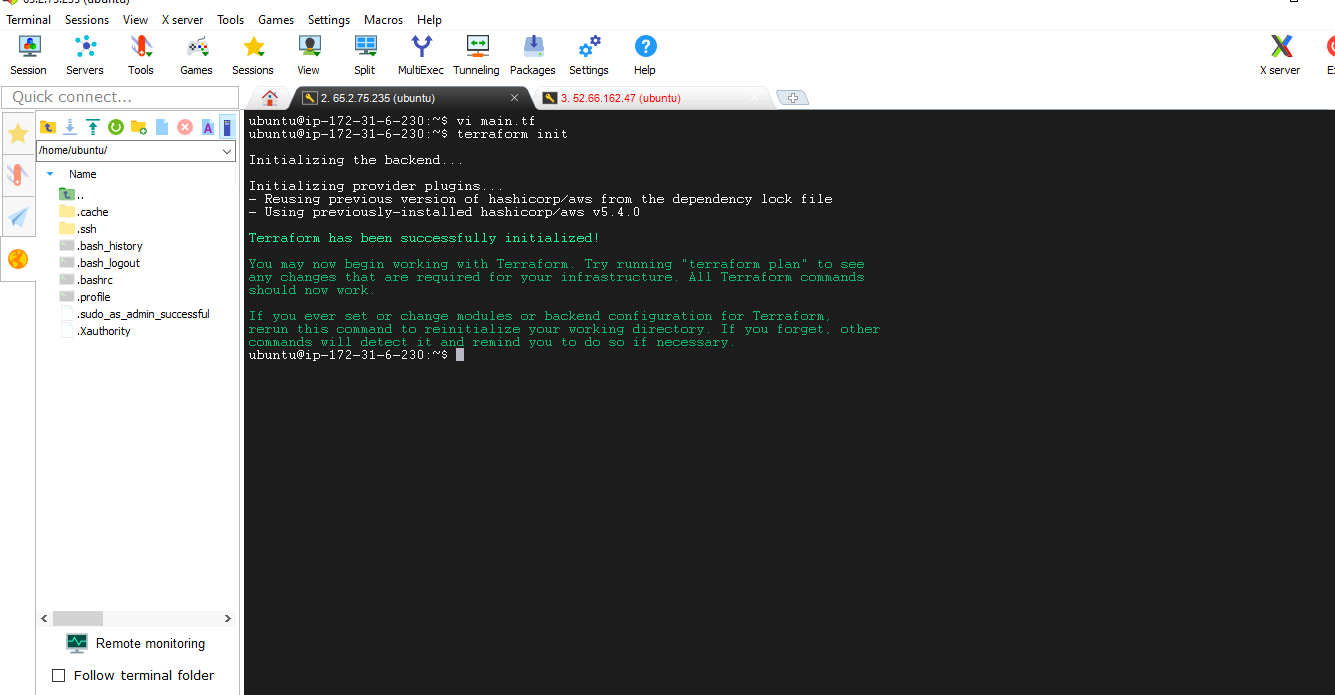
**Output:**

**Commands for executing main.tf**

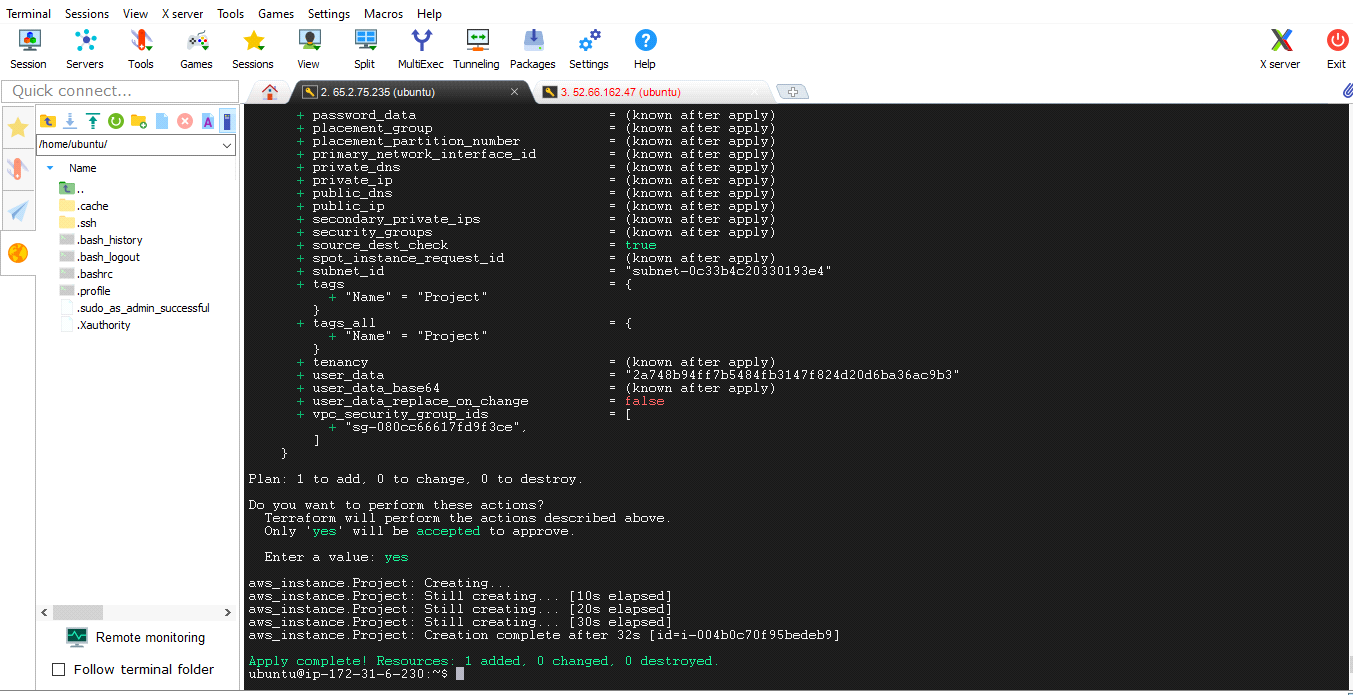
* terraform init
* terraform plan
* terraform apply

All the resources are created in AWS console

* terraform init



* **terraform plan**
* **terraform apply**

****

**Step 5:**

**in aws instances we can see that new instance has been created with public\_ip**

**Step 6:**

* We can expose our nginx application by using instance public ip address with port number 8080

