**DevOps-Task**

**Assigned task**

**-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 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:latest

EXPOSE 8080

CMD ["nginx", "-g", "daemon off;"]

**Commands:**

* **For starting docker Service**

sudo systemctl start docker

* **Authenticate Docker with your ECR repository**

aws ecr get-login-password --region ap-south-1 | docker login --username AWS --password-stdin 078307860773.dkr.ecr.ap-south-1.amazonaws.com

* **Build the docker image**

sudo docker build -t final-1 .

* **Image to Container**

sudo docker run -d -it final-1

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

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

#aws is a provider

provider "aws" {

region = "ap-south-1"

access\_key = "AKIAREO4DEESTW3JTPEB"

secret\_key = "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" }

#we creating a vpc

resource "aws\_vpc" "my\_vpc-2" {

cidr\_block = "[10.0.0.0/16](http://10.0.0.0/16" \t "https://chat.google.com/u/0/_blank)"

}

#we are creating a public subnet

resource "aws\_subnet" "public\_subnet-2" {

vpc\_id     = [aws\_vpc.my\_vpc-2.id](http://aws_vpc.my_vpc-2.id/" \t "https://chat.google.com/u/0/_blank)

cidr\_block = "[10.0.1.0/24](http://10.0.1.0/24" \t "https://chat.google.com/u/0/_blank)"

availability\_zone= "ap-south-1a"

map\_public\_ip\_on\_launch = true

}

#we are creating a private subnet

resource "aws\_subnet" "private\_subnet-2" {

vpc\_id     = [aws\_vpc.my\_vpc-2.id](http://aws_vpc.my_vpc-2.id/" \t "https://chat.google.com/u/0/_blank)

cidr\_block = "[10.0.2.0/24](http://10.0.2.0/24" \t "https://chat.google.com/u/0/_blank)"

availability\_zone       = "ap-south-1a"

}

#we are creating security groups for public subnet

resource "aws\_security\_group" "public\_sg-2" {

vpc\_id = [aws\_vpc.my\_vpc-2.id](http://aws_vpc.my_vpc-2.id/" \t "https://chat.google.com/u/0/_blank)

ingress {

from\_port   = 22

to\_port     = 22

protocol    = "tcp"

cidr\_blocks = ["[0.0.0.0/0](http://0.0.0.0/0" \t "https://chat.google.com/u/0/_blank)"]

}

ingress {

from\_port   = 80

to\_port     = 80

protocol    = "tcp"

cidr\_blocks = ["[0.0.0.0/0](http://0.0.0.0/0" \t "https://chat.google.com/u/0/_blank)"]

}

egress {

from\_port   = 0

to\_port     = 0

protocol    = "-1"

cidr\_blocks = ["[0.0.0.0/0](http://0.0.0.0/0" \t "https://chat.google.com/u/0/_blank)"]

}

}

#we are creating security for private subnet

resource "aws\_security\_group" "private\_sg-2" {

vpc\_id = [aws\_vpc.my\_vpc-2.id](http://aws_vpc.my_vpc-2.id/" \t "https://chat.google.com/u/0/_blank)

ingress {

from\_port   = 22

to\_port     = 22

protocol    = "tcp"

cidr\_blocks = ["[0.0.0.0/0](http://0.0.0.0/0" \t "https://chat.google.com/u/0/_blank)"]

}

egress {

from\_port   = 0

to\_port     = 0

protocol    = "-1"

cidr\_blocks = ["[0.0.0.0/0](http://0.0.0.0/0" \t "https://chat.google.com/u/0/_blank)"]

}

}

#we are creating an instance

resource "aws\_instance" "ec2\_instance-2" {

ami           = "ami-0f5ee92e2d63afc18"

instance\_type = "t2.micro"

subnet\_id     = [aws\_subnet.public\_subnet-2.id](http://aws_subnet.public_subnet-2.id/" \t "https://chat.google.com/u/0/_blank)

vpc\_security\_group\_ids = [[aws\_security\_group.public\_sg-2.id](http://aws_security_group.public_sg-2.id/" \t "https://chat.google.com/u/0/_blank)]

user\_data = <<-EOF

#!/bin/bash

echo "Starting user data script"

sudo apt-get update

sudo apt-get install -y [docker.io](http://docker.io/" \t "https://chat.google.com/u/0/_blank)

sudo aws ecr get-login-password --region ap-south-1 |sudo docker login --username AWS --password-stdin [078307860773.dkr.ecr.ap-south-1.amazonaws.com](http://078307860773.dkr.ecr.ap-south-1.amazonaws.com/" \t "https://chat.google.com/u/0/_blank)

sudo docker pull [078307860773.dkr.ecr.ap-south-1.amazonaws.com/final-assessment-1:latest](http://078307860773.dkr.ecr.ap-south-1.amazonaws.com/final-assessment-1:latest" \t "https://chat.google.com/u/0/_blank)

sudo docker run -d -p 80:80 [078307860773.dkr.ecr.ap-south-1.amazonaws.com/final-assessment-1:latest](http://078307860773.dkr.ecr.ap-south-1.amazonaws.com/final-assessment-1:latest" \t "https://chat.google.com/u/0/_blank) EOF } output "public\_ip" {   value = aws\_instance.ec2\_instance-2.public\_ip }

**Output:**

**Commands for executing main.tf**

* terraform init
* terraform plan
* terraform apply

All the resources are created in AWS console

**Step 4:**

**Pulling image from ECR**

* **Pulling image to docker From ECR**

sudo docker pull 078307860773.dkr.ecr.ap-south-1.amazonaws.com/final-assessment-1

* **After pulling this image i have created a docker container**

docker run -d -it 0e9af366c15d(container is formed)

**Step 5:**

**Starting Nginx Server**

**Commands**

Sudo docker exec -it 8dc2b86721c5 bash(we get inside bash -- with the help container id)

Service nginx status(to check nginx server -- exit to come out

sudo docker run -it -p 8080:80 -d final-1(to run it one port)

