using aws terraform code to create an EC2 instance and deploy sample application

Step1:

— create an server with using aws cloud

— in the server install docker for creating docker file

— and the same server install the terraform also for creating vpc

And subnets and security groups and Ec2-instance.

— in the server install the aws-cli also for connecting with aws

Console.

— write the Dockerfile in server for creating the nginx image

Dockerfile

FROM nginx:latest

WORKDIR /app

COPY . .

CMD ["echo","hello world"]

— after that give the aws credentials with using ‘aws configure’

Command. If enter this command its ask aws credentials id

Aws access key and aws region name. Then successful login

aws.

— now the nginx image pushed to aws ECR with using ECR(Elastic

Container registry) commands.

Step2:

— now with using terraform create the vpc , subnetids,

security groups and ec2-instance.

Main.tf

provider "aws" {

region = var.region

access\_key = var.access\_key

secret\_key = var.secret\_key

}

# step-1 creating VPC

resource "aws\_vpc" "my-vpc-1" {

cidr\_block = var.cidr\_block

tags = {

Name = "my-vpc"

}

}

# step-2 creating internet gateway

resource "aws\_internet\_gateway" "my-igw-1" {

vpc\_id = aws\_vpc.my-vpc-1.id

tags = {

Name = "my-igw"

}

}

#step-3 creating route table

resource "aws\_route\_table" "my-rt-1" {

vpc\_id = aws\_vpc.my-vpc-1.id

route {

cidr\_block = "0.0.0.0/0"

gateway\_id = aws\_internet\_gateway.my-igw-1.id

}

tags = {

Name = "my-rt"

}

}

# step-4 creating subnet

resource "aws\_subnet" "my-subnet" {

vpc\_id = aws\_vpc.my-vpc-1.id

cidr\_block = "10.0.1.0/24"

availability\_zone = var.availability\_zone

tags = {

Name = "my-subnet-1"

}

}

#step-5: associate subnet with route table

resource "aws\_route\_table\_association" "my-rt-association" {

subnet\_id = aws\_subnet.my-subnet.id

route\_table\_id = aws\_route\_table.my-rt-1.id

}

#step-6 : creating security group and allow ports 22,80,443

resource "aws\_security\_group" "my-sg-1" {

vpc\_id = aws\_vpc.my-vpc-1.id

ingress {

description = "SSH"

from\_port = 22

to\_port = 22

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

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

}

ingress {

description = "HTTP"

from\_port = 80

to\_port = 80

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

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

}

ingress {

description = "HTTPS"

from\_port = 443

to\_port = 443

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

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

}

egress {

from\_port = 0

to\_port = 0

protocol = "-1"

cidr\_blocks = ["0.0.0.0/0"]

}

tags = {

Name = "my-sg"

}

}

#step-7 : create network interface with an IP in subnet created in step-4

resource "aws\_network\_interface" "my-ni" {

subnet\_id = aws\_subnet.my-subnet.id

private\_ips = ["10.0.1.50"]

security\_groups = [aws\_security\_group.my-sg-1.id]

}

#step-8: assigning elastic ip to network interface

resource "aws\_eip" "name" {

vpc = true

network\_interface = aws\_network\_interface.my-ni.id

associate\_with\_private\_ip = "10.0.1.50"

depends\_on = [aws\_internet\_gateway.my-igw-1]

}

#step-9: creating ubuntu instance and downloading nginx

resource "aws\_instance" "ubuntu" {

ami = var.ami

instance\_type = var.instance\_type

availability\_zone = var.availability\_zone

key\_name = var.key\_name

network\_interface {

device\_index = 0

network\_interface\_id = aws\_network\_interface.my-ni.id

}

user\_data = <<-EOF

#! /bin/bash

sudo su -

sudo apt-get update -y

sudo apt 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 | sudo docker login --username AWS --password-stdin public.ecr.aws/x4l2t1n6

sudo docker pull public.ecr.aws/x4l2t1n6/mkreddy:latest

sudo docker run -d -p 8080:80 public.ecr.aws/x4l2t1n6/mkreddy@sha256:9ac1131ddaab7fded0f61d509642ed09268256c5afe52aad46000b9167c7c683 i

EOF

tags = {

Name = "tf-ubuntu"

}

}

— after that give ‘terraform init’ its intilizing the code

— after that give ‘terraform validate’ its verifying the code

— after that give ‘terraform plan’ its show the plan what are all will

be Created.

— after that ‘terraform apply’ its created automatically what ever

i give in terraform file.

Step3:

— the terraform file i am mentioned user date

— the user date will be install the docker and aws-cli on instance.

and pull the nginx image form aws ECR. and create the

Container with using nginx .

Step4:

— now explose the nginx image in web using public ip address

and container port.

— <http://13.231.171.29:8000/>

Step5:

— now push the all the files to github whatever i created for

Pushing and pulling nginx image.

— https://github.com/reddymodi/FinalTask.git