## 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

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

}

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

cidr\_block = var.cidr\_block

tags = {

Name = "task-vpc"

}

}

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

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

tags = {

Name = "my-igw"

}

}

resource "aws\_route\_table" "rt-table" {

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

route {

cidr\_block = "0.0.0.0/0"

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

}

}

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

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

cidr\_block = "10.0.1.0/24"

availability\_zone = var.availability\_zone

map\_public\_ip\_on\_launch = false

tags = {

Name = "task-private-subnet"

}

}

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

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

cidr\_block = "10.0.2.0/24"

availability\_zone = var.availability\_zone

map\_public\_ip\_on\_launch = true

tags = {

Name = "task-public-subnet"

}

}

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

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

route\_table\_id = aws\_route\_table.rt-table.id

}

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

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

tags = {

Name = "sg-ft"

}

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 = 0

to\_port = 65535

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

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

}

egress {

from\_port = 0

to\_port = 0

protocol = "-1"

}

}

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

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

private\_ips = ["10.0.1.50"]

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

}

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]

}

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 get-login-password --region ap-south-1 | sudo docker login --username AWS --password-stdin 543066576745.dkr.ecr.ap-south-1.amazonaws.com

sudo docker pull 543066576745.dkr.ecr.ap-south-1.amazonaws.com/kavya-ft:latest

sudo docker run -d -p 8080:80 docker push 543066576745.dkr.ecr.ap-south-1.amazonaws.com/kavya-ft:latest

EOF

tags = {

Name = "my-ec2-final"

}

}

Varible.tf

variable "region" {

type = string

default = "ap-south-1"

}

variable "access\_key" {

type = string

default = "AKIAX44KKK5U5PJCUHAL"

}

variable "secret\_key" {

type = string

default = "3ELVqmM/jRSUM+0U4QN1ok4hEt2lzxa5LZaAun7H"

}

variable "cidr\_block" {

type = string

default = "10.0.0.0/16"

}

variable "availability\_zone" {

type = string

default = "ap-south-1a"

}

variable "ami" {

type = string

default = "ami-08e5424edfe926b43"

}

variable "instance\_type" {

type = string

default = "t2.micro"

}

variable "key\_name" {

type = string

default = "kavya-ft"

}

— 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.

Step5:

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

Pushing and pulling nginx image.

— https://github.com/kavyareddych/kavya-final.git