## Deploy Three-Tier Architecture in

AWS using Terraform

Terraform :- Terraform is an open-source infrastructure as a code (IAC) tool that allows to create, manage & deploy the production-ready environment. Terraform is a tool for building, changing and versioning infrastructure safely and efficiently.

Prerequisites:

Basic knowledge of AWS & Terraform

AWS Account

IAM User

GitHub Account

AWS Access & Secret Key.

Install terraform process

• Create instance by free service

• Connect the instance into terminal

• Install terraform by using

sudo yum install -y yum-utils shadow-utils

sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo

sudo yum -y install terraform

* Initialize the terraform by using

Terraform init

Step 1: Give permissions for creating infrastructure

#update

#creating vpc using terraform

provider "aws" {

region = "ap-south-1"

access\_key = "AKIATVHK7JK4V2VICJB6**"**

secret\_key = "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"

}

**Step 2: Create a file for the VPC**

* Creating vpc by using below

#creating vpc in terraform

resource "aws\_vpc" "main" {

cidr\_block = "10.0.0.0/16"

instance\_tenancy = "default"

tags = {

Name = "TCS vpc"

}

}

Terraform validate :- The terraform validate command validate the configuration files in adirectory, referring only to the configuration and not accessing any remote services such as remote state, provider APIs etc.

Terraform plan :- the terraform plan command lets you to preview the actions terraform would take to modify your infrastructure, or save a speculative plan which you can apply later.

Terraform apply :- the terraform apply command executes the actions proposed in a

terraform plan. It is used to deploy your infrastructure.

**3:-Creat subnets**

#creating 1st web subnet

resource "aws\_subnet" "web-subnet-1" {

vpc\_id = "vpc-0d5f1f9932e11a9c6"

cidr\_block = "10.0.0.0/24"

map\_public\_ip\_on\_launch = true

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

tags = {

Name = "awb subnet 1"

}

}

#creating 2nd web subnet

resource "aws\_subnet" "web-subnet-2" {

vpc\_id = "vpc-0d5f1f9932e11a9c6"

cidr\_block = "10.0.1.0/24"

map\_public\_ip\_on\_launch = true

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

tags = {

Name = "web subnet 2"

}

}

#creating 1st application subnet

resource "aws\_subnet" "application-subnet-1" {

vpc\_id = "vpc-0d5f1f9932e11a9c6"

cidr\_block = "10.0.2.0/24"

map\_public\_ip\_on\_launch = true

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

tags = {

Name = "application subnet 1"

}

}

#creating 2nd application subnet

resource "aws\_subnet" "application-subnet-2" {

vpc\_id = "vpc-0d5f1f9932e11a9c6"

cidr\_block = "10.0.3.0/24"

map\_public\_ip\_on\_launch = true

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

tags = {

Name = "application subnet 2"

}

}

#creating 1st database subnet

resource "aws\_subnet" "private-subnet-1" {

vpc\_id = "vpc-0d5f1f9932e11a9c6"

cidr\_block = "10.0.4.0/24"

map\_public\_ip\_on\_launch = true

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

tags = {

Name = "database subnet 1"

}

}

#creating 2nd database subnet

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

vpc\_id = "vpc-0d5f1f9932e11a9c6"

cidr\_block = "10.0.5.0/24"

map\_public\_ip\_on\_launch = true

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

tags = {

Name = "database subnet 2"

}

}

**Step 4:- Creating internet gate way**

# Creating Internet gate way by using below code

#creating internet gateway

resource "aws\_internet\_gateway" "IGW" {

vpc\_id = "vpc-0d5f1f9932e11a9c6"

tags = {

Name = "IGW"

}

}

Step 4 :- associating route table with subnets

# creating of route association

resource "aws\_route\_table\_association" "web1" {

subnet\_id = "subnet-03af1c0e38a2b917c"

route\_table\_id = "rtb-0c43e2987cfbb8216"

}

# creating of route association

resource "aws\_route\_table\_association" "web2" {

subnet\_id = "subnet-0f4b02df856238bdf"

route\_table\_id = "rtb-0c43e2987cfbb8216"

}

# creating of route association

resource "aws\_route\_table\_association" "app-sub-1" {

subnet\_id = "subnet-086dea8ce9b3141a3"

route\_table\_id = "rtb-0c43e2987cfbb8216"

}

# creating of route association

resource "aws\_route\_table\_association" "app-sub-2" {

subnet\_id = "subnet-08a06ac7d42ad8f17"

route\_table\_id = "rtb-0c43e2987cfbb8216"

}

# creating of route association

resource "aws\_route\_table\_association" "db1" {

subnet\_id = "subnet-0c1560f777b08cfc8"

route\_table\_id = "rtb-0c43e2987cfbb8216"

}

# creating of route association

resource "aws\_route\_table\_association" "db2" {

subnet\_id = "subnet-01811dfcee33943f7"

route\_table\_id = "rtb-0c43e2987cfbb8216"

}

Step 4:- Creating route table

#creating Route Table

resource "aws\_route\_table" "pub-route1" {

vpc\_id = "vpc-0d5f1f9932e11a9c6"

route {

cidr\_block = "0.0.0.0/0"

gateway\_id = "igw-0299f15688cce4a25"

}

tags = {

Name = "pub route 1"

}

}

#creating Route Table

resource "aws\_route\_table" "pvt-route1" {

vpc\_id = "vpc-0d5f1f9932e11a9c6"

route {

cidr\_block = "0.0.0.0/0"

gateway\_id = "igw-0299f15688cce4a25"

}

tags = {

Name = "pvt route 2"

}

}

**Step 5 :- creating security groups**

# Creating security Group

resource "aws\_security\_group" "demo1" {

vpc\_id = "vpc-0d5f1f9932e11a9c6"

# Inbound Rules

#HTTP access from anywhere

ingress {

from\_port = 80

to\_port = 80

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

}

# HTTPS access from anywhere

ingress {

from\_port = 443

to\_port = 443

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

}

#SSH access from anywhere

ingress {

from\_port = 22

to\_port = 22

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

}

# outbound rules

# internet access to anywhere

egress {

from\_port = 0

to\_port = 0

protocol = "-1"

cidr\_blocks = ["0.0.0.0/0"]

}

tags = {

Name = "Web SG"

}

}

**Step 6 :- Creating EC2 instance**

# Creating 1st instance

resource "aws\_instance" "ec2-1" {

ami = "ami-074dc0a6f6c764218"

instance\_type = "t2.micro"

count = "1"

key\_name = "dog"

vpc\_security\_group\_ids = ["sg-03ed81842339deff1"]

subnet\_id = "subnet-03af1c0e38a2b917c"

associate\_public\_ip\_address = true

tags = {

Name = "pub"

}

}

# Creating 2nd instance

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

ami = "ami-074dc0a6f6c764218"

instance\_type = "t2.micro"

count = "1"

key\_name = "dog"

vpc\_security\_group\_ids = ["sg-03ed81842339deff1"]

subnet\_id = "subnet-0c1560f777b08cfc8"

associate\_public\_ip\_address = false

user\_data = "${file("data.sh")}"

tags = {

Name = "pvt"

}

}

**Step 7 :- Creating application LOAD BALANCER**

# Creating External LoadBalancer

resource "aws\_lb" "external-alb" {

name = "External-LB"

internal = false

load\_balancer\_type = "application"

security\_groups = ["sg-0b9021ae2d9b6cc41"]

subnets = ["subnet-086dea8ce9b3141a3","subnet-08a06ac7d42ad8f17"]

}

resource "aws\_lb\_target\_group" "target-elb" {

name = "ALB-TG"

port = 80

protocol = "HTTP"

vpc\_id = "vpc-0d5f1f9932e11a9c6"

}

resource "aws\_lb\_target\_group\_attachment" "attachment" {

target\_group\_arn = aws\_lb\_target\_group.target-elb.arn

target\_id = "i-08d4919fdb1f166c5"

port = 80

}

resource "aws\_lb\_target\_group\_attachment" "attachment1" {

target\_group\_arn = aws\_lb\_target\_group.target-elb.arn

target\_id = "i-07572c8a4a18173b3"

port = 80

}

resource "aws\_lb\_listener" "external-elb" {

load\_balancer\_arn = aws\_lb.external-alb.arn

port = 80

protocol = "HTTP"

default\_action {

type = "forward"

target\_group\_arn = aws\_lb\_target\_group.target-elb.arn

}

**Step 8 :- Creating database security group for RDS**

# Create Database Security Group

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

name = "Database SG"

description = "Allow inbound traffic from application layer"

vpc\_id = "vpc-0d5f1f9932e11a9c6"

ingress {

description = "Allow traffic from application layer"

from\_port = 3306

to\_port = 3306

protocol = "tcp"

security\_groups = ["sg-03ed81842339deff1"]

}

egress {

from\_port = 32768

to\_port = 65535

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

}

tags = {

Name = "Database SG"

}

}

**Step 9 :- creating RDS database instance**

resource "aws\_db\_subnet\_group" "default" {

name = "db"

subnet\_ids = ["subnet-0c1560f777b08cfc8", "subnet-01811dfcee33943f7"]

tags = {

Name = "My DB subnet group"

}

}

resource "aws\_db\_instance" "default" {

allocated\_storage = 10

db\_subnet\_group\_name = aws\_db\_subnet\_group.default.id

engine = "mysql"

engine\_version = "5.7"

instance\_class = "db.t2.micro"

multi\_az = true

db\_name = "mydb"

username = "admin"

password = "admin1234"

vpc\_security\_group\_ids = ["sg-03ed81842339deff1"]

}

**Step 10:- creating data.sh**

**Creat data.sh file and in run this commands**

#!/bin/bash

yum update –y

sudo yum –y install git

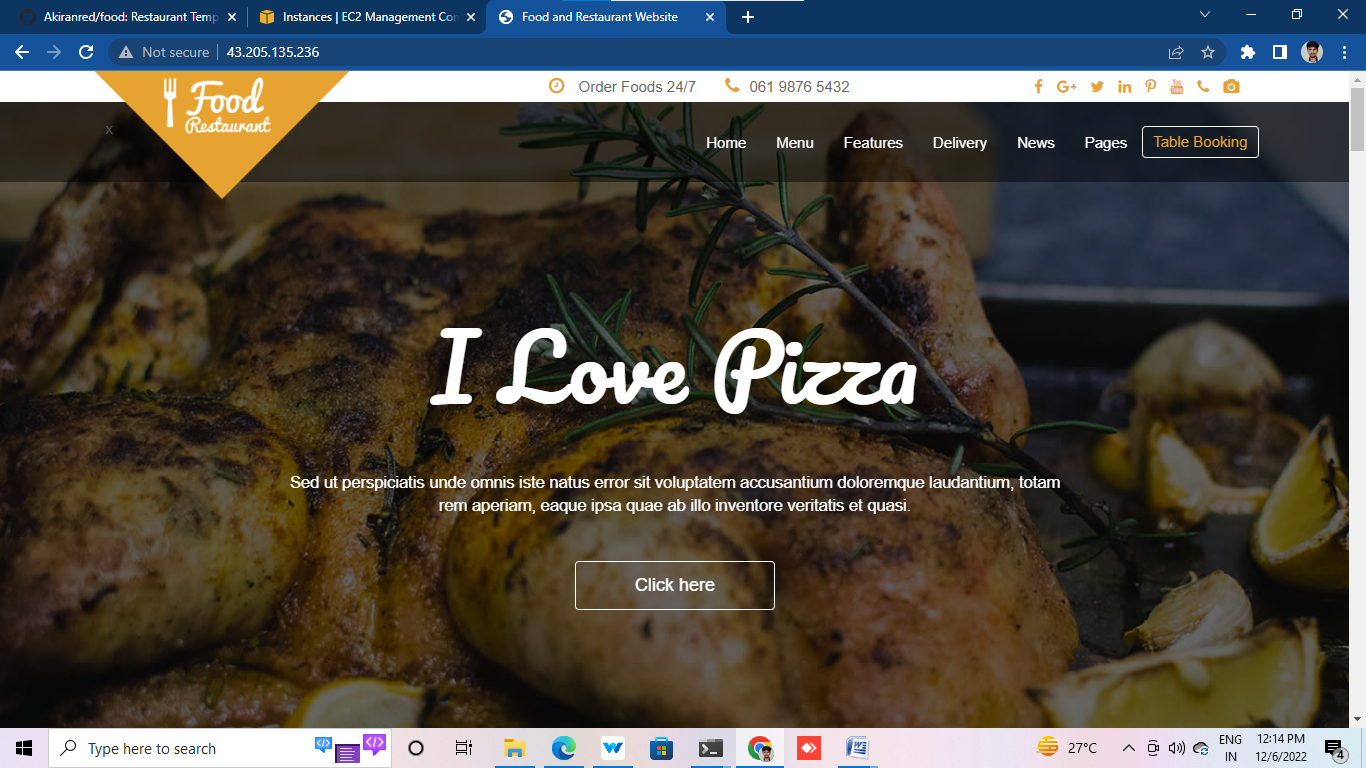
sudo yum install -y httpd

sudo systemctl start httpd

sudo systemctl enable httpd

sudo git clone [https://github.com/Akiranred/food.git /var/www/html/](https://github.com/Akiranred/food.git%20/var/www/html/)

# then brows through public ip+80

 **step 11 :- creating EC2 instance with data.sh**

* Creating EC2 instance with data.sh

# Creating 1st EC2 instance in Public Subnet

resource "aws\_instance" "demoinstance" {

ami = "ami-0beaa649c482330f7"

instance\_type = "t2.micro"

count = 1

key\_name = "joke"

vpc\_security\_group\_ids = ["${aws\_security\_group.demosg.id}"]

subnet\_id = "${aws\_subnet.public\_subnet-1.id}"

associate\_public\_ip\_address = true

user\_data = "${file("data.sh")}"

tags = {

Name = "ec2 instance"

}

}