**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. Terraform is open source infrastructure as code software tool that provides a consistent CLI workflow to manage cloud services.

**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**

* Create main.tf

# update the terraform

# my provider aws

provider "aws" {

region = "us-east-2"

access\_key = "AKIAX4NLJCBA3LM4RC4E"

secret\_key = "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"

}

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

* Create VPC by using below code

# creating vpc in terraform

resource "aws\_vpc" "demovpc" {

cidr\_block = "10.0.0.0/16"

instance\_tenancy = "default"

tags = {

NAME = "demo-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.

**Step 3:- Creating internet gate way**

* Creating Internet gate way by using below code

# creating internet gateway

resource "aws\_internet\_gateway" "demo-gateway" {

vpc\_id = "${aws\_vpc.demovpc.id}"

}

**Step 4 :- creating subnets for vpc**

* Creating subnets by using below code

# creating first web subnet in vpc

resource "aws\_subnet" "public\_subnet-1" {

vpc\_id = "${aws\_vpc.demovpc.id}"

cidr\_block = "10.0.0.0/24"

map\_public\_ip\_on\_launch = true

availability\_zone = "us-east-2a"

tags = {

NAME = "web-subnet-1"

}

}

# creating 2nd web subnet in vpc

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

vpc\_id = "${aws\_vpc.demovpc.id}"

cidr\_block = "10.0.1.0/24"

map\_public\_ip\_on\_launch = true

availability\_zone = "us-east-2b"

tags = {

NAME = "web-subnet-2"

}

}

# creating 1st app subnet in vpc

resource "aws\_subnet" "app\_subnet-1" {

vpc\_id = "${aws\_vpc.demovpc.id}"

cidr\_block = "10.0.2.0/24"

map\_public\_ip\_on\_launch = false

availability\_zone = "us-east-2a"

tags = {

NAME = "app-1"

}

}

# creating 2nd app subnet in vpc

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

vpc\_id = "${aws\_vpc.demovpc.id}"

cidr\_block = "10.0.3.0/24"

map\_public\_ip\_on\_launch = false

availability\_zone = "us-east-2b"

tags = {

NAME = "app-2"

}

}

# creating 1st database subnet in vpc

resource "aws\_subnet" "database\_subnet-1" {

vpc\_id = "${aws\_vpc.demovpc.id}"

cidr\_block = "10.0.4.0/24"

availability\_zone = "us-east-2a"

tags = {

NAME = "db-1"

}

}

# creating 2nd database subnet in vpc

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

vpc\_id = "${aws\_vpc.demovpc.id}"

cidr\_block = "10.0.5.0/24"

availability\_zone = "us-east-2b"

tags = {

NAME = "db-2"

}

}

**Step 5:- Creating route table and associating route table with subnets**

* Creating route table and associating route table with subnets

# creating route table in vpc

resource "aws\_route\_table" "route" {

vpc\_id = "${aws\_vpc.demovpc.id}"

route {

cidr\_block = "0.0.0.0/0"

gateway\_id = "${aws\_internet\_gateway.demo-gateway.id}"

}

tags = {

NAME = "route to internet"

}

}

# associating route table

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

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

route\_table\_id = "${aws\_route\_table.route.id}"

}

# associating route table

resource "aws\_route\_table\_association" "rt-2" {

subnet\_id = "${aws\_subnet.public\_subnet-2.id}"

route\_table\_id = "${aws\_route\_table.route.id}"

}

**Step 6 :- creating security groups**

* Creating security groups by using below code

# creating security group

resource "aws\_security\_group" "demosg" {

vpc\_id = "${aws\_vpc.demovpc.id}"

# inbound rules

# HTTP acces 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 7 :- Creating EC2 instance**

* Creating EC2 instance by using below

# 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

tags = {

Name = "ec2 instance"

}

}

# Creating 1st EC2 instance in Public Subnet

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

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

tags = {

Name = "ec2 instance"

}

}

**Step 8 :- Creating application LOAD BALANCER**

* Creating application load balancer

# Creating External LoadBalancer

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

name = "External-LB"

internal = false

load\_balancer\_type = "application"

security\_groups = [aws\_security\_group.demosg.id]

subnets = [aws\_subnet.public\_subnet-1.id, aws\_subnet.public\_subnet-2.id]

}

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

name = "ALB-TG"

port = 80

protocol = "HTTP"

vpc\_id = aws\_vpc.demovpc.id

}

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

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

target\_id = aws\_instance.instance.id

port = 80

depends\_on = [aws\_instance.instance,]

}

resource "aws\_lb\_target\_group\_attachment" "attachment2" {

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

target\_id = aws\_instance.instance-2.id

port = 80

depends\_on = [aws\_instance.instance-2,]

}

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 9 :- Creating output file for creating DNS id of load balancer**

* Creating DNS for load balancer

# Getting the DNS of load balancer

output "lb\_dns\_name" {

description = "The DNS name of the load balancer"

value = "${aws\_lb.external-alb.dns\_name}"

}

**Step 10 :- Creating database security group for RDS**

* Creating database security groups by using below code

# creating mysql security group

resource "aws\_security\_group" "mysqlsg" {

vpc\_id = "${aws\_vpc.demovpc.id}"

ingress {

from\_port = "3306"

to\_port = "3306"

protocol = "tcp"

}

egress {

from\_port = 32768

to\_port = 65535

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

}

tags = {

NAME = "db sg"

}

}

**Step 11 :- creating RDS database**

* Creating RDS database using below code

# Creating RDS Instance

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

name = "main"

subnet\_ids = [aws\_subnet.database\_subnet-1.id, aws\_subnet.database\_subnet-2.id]

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 = "username"

password = "password"

skip\_final\_snapshot = true

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

}

**Step 12 :- creating data.sh**

* Creating data.sh for execute commands at a time

**#!/bin/bash**

**yum update -y**

**yum install -y httpd.x86\_64**

**systemctl start httpd.service**

**systemctl enable httpd.service**

**echo "Hello World from $(hostname -f)" > /var/www/html/index.html**

**step 13 :- 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"

}

}