**Challenge #1**

A 3-tier environment is a common setup. Use a tool of your choosing/familiarity create these

resources on a cloud environment (Azure/AWS/GCP). Please remember we will not be judged

on the outcome but more focusing on the approach, style and reproducibility.

**Solution :**

For Creating a 3-Tier Architecture in AWS follow the below steps -

1. Creating VPC (Virtual Private Cloud)-

Either we can use AWS console or AWS CLI or IAC platform to create VPC

* Using AWS CLI

aws ec2 create-vpc --cidr-block 10.0.0.0/24

* Using Terraform as IAC tool

resource “aws\_vpc” “3tier\_architecture\_vpc” {

cidr\_block = “0.0.0.0/24”

}

1. Creating Subnet groups inside the VPC

* Using AWS CLI

# Public Subnet

aws ec2 create-subnet --vpc-id <vpc id created in first step> --cidr-block 10.0.0.0/25

# Application Subnet and Database subnet

aws ec2 create-subnet --vpc-id <vpc id created in first step> --cidr-block 10.0.0.128/25

* Using Terraform Script

# Create a public subnet for front end application

resource "aws\_subnet" "3\_tier\_public\_subnet" {

vpc\_id = aws\_vpc.3tier\_architecture\_vpc.id

cidr\_block = "10.0.0.0/25”

availability\_zone = "us-east-1a"

map\_public\_ip\_on\_launch = true

}

# Create a private subnet for Application and Database

resource "aws\_subnet" "3\_tier\_private\_subnet" {

vpc\_id = aws\_vpc.3tier\_architecture\_vpc.id

cidr\_block = "10.0.0.128/25"

availability\_zone = "us-east-1a"

map\_public\_ip\_on\_launch = false }

1. Security Groups

* Using AWS CLI

aws ec2 create-security-group --group-name Web-Sg--description "Security Group for Web Tier" --vpc-id your-vpc-id

aws ec2 create-security-group --group-name App-SG --description "Security Group for Application Tier" --vpc-id your-vpc-id

aws ec2 create-security-group --group-name DB-SG --description "Security Group for Database Tier" --vpc-id your-vpc-id

* Using AWS Terraform Script

Security Group for WEB SG - -

resource "aws\_security\_group" "3\_tier\_web\_sg" {

name = "3-tier-web-sg"

description = "SG for web subnet"

vpc\_id = aws\_vpc.3tier\_architecture\_vpc.id

ingress {

from\_port = 80

to\_port = 80

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

}

}

Security Group for APP SG - -

resource "aws\_security\_group" "3\_tier\_app\_sg" {

name = "3-tier-app-sg"

description = "SG for app subnet"

vpc\_id = aws\_vpc.3tier\_architecture\_vpc.id

ingress {

from\_port = 0

to\_port = 65535

protocol = "tcp"

security\_groups = [aws\_security\_group.3\_tier\_web\_sg.id] }

}

Security Group for DB SG - -

resource "aws\_security\_group" "3\_tier\_db\_sg" {

name = "3-tier-db-sg"

description = "SG for db subnet"

vpc\_id = aws\_vpc.3tier\_architecture\_vpc.id

ingress {

from\_port = 0

to\_port = 65535

protocol = "tcp"

security\_groups = [aws\_security\_group.3\_tier\_web\_sg.id] }

}

1. Create EC2 instances for all the tier - i.e. Front End , Application Layer, Backend

* Using AWS CLI

aws ec2 create-security-group --group-name Web-SG --description "Security Group for Web Tier" --vpc-id <vpc id created in first step>

aws ec2 create-security-group --group-name App-SG --description "Security Group for Application Tier" --vpc-id <vpc id created in first step>

aws rds create-db-instance --db-instance-identifier 3-tier-rds-instance --db-instance-class db.t2.micro --engine mysql --master-username mydb --master-user-password 123# --allocated-storage 20 --vpc-security-group-ids <sg id from above step> --availability-zone [us-east-1a,us-east-1b]

* Using Terraform Script

Create EC2 instances for web

resource "aws\_instance" "web\_instance" {

ami = "ami-0c55b159cbfafe1f0"

instance\_type = "t2.micro"

subnet\_id = aws\_subnet.3\_tier\_public\_subnet.id

security\_groups = [aws\_security\_group.3\_tier\_web\_sg.id]

tags = {

Name = "WebInstance"

}

}

Create EC2 instances for app

resource "aws\_instance" "app\_instance" {

ami = "ami-0c55b159cbfafe1f0"

instance\_type = "t2.micro"

subnet\_id = aws\_subnet.3\_tier\_private\_subnet.id

security\_groups = [aws\_security\_group.3\_tier\_app\_sg.id]

tags = {

Name = "AppInstance"

}

}

Create an RDS instance

resource "aws\_db\_instance" "my\_rds" {

allocated\_storage = 20

storage\_type = "gp2"

engine = "mysql"

engine\_version = "5.7"

instance\_class = "db.t2.micro"

name = "mydb"

username = "mydb"

password = "123#"

parameter\_group\_name = "default.mysql5.7"

skip\_final\_snapshot = true

}

This is the basic architecture for 3-Tier application. For scaling and monitoring purpose we can include cloudwatch logs and auto-scaling groups here.