Que 1 →

➔ Create below resources using Terraform using count * count.index.

- 3 IAM Users (yourname_0 , yourname_1 , yourname_2)
- 3 IAM Groups (dev_0 , dev_1 , dev_2)

➔ Map these IAM users in IAM groups that you have created using **aws_iam_user_group_membership** Terraform resource.

➔ Note :-

- yourname_0 should be part of dev_0 group.
- yourname_1 should be part of dev_1 group.
- yourname_2 should be part of dev_2 group.

Que 2 →

➔ Create below resources using Terraform based on conditions.

➔ Create a variable name ENV with any of these values (DEV/QA).

- If ENV is DEV then
  - Create 2 EC2 instances.
  - 1 Security Group and allow traffic from port 22,80,443.
  - Map Security Group to EC2 Instance.
- If ENV is QA then.
  - Create 1 EC2 instance.
  - 1 Security Group and allow traffic from port 22,8080,3306.
  - Map Security Group to EC2 Instance.

➔ Note :-

- Use data sources to fetch the AMI ID for Dev and QA instances.
- Dev instance should be created with filter "ubuntu/images/hvm-ssd/ubuntu-focal-20.04-amd64-server-*"
- QA instance should be created with filter "ubuntu/images/hvm-ssd/ubuntu-bionic-18.04-amd64-server-"
- Use Dynamic block for security group ingress & egress to add rules.
- Also Make sure these resources must be created in your own VPC.

Que 3 →

➔ Create below resources using Terraform.

- Create VPC with 2 Public and 2 Private subnets including IGW / NAT GW / RT & Subnet association.
- Create 1 Security Group named alb-sg. Allow traffic on port 80 & 443 from 0.0.0.0/0
- Create 1 Application Load Balancer in the public Subnet of your VPC & attach alb-sg security group to ALB.
- Create 1 Security Group named web-sg. Allow traffic on port 80 from 0.0.0.0/0
- Create 1 EC2 instance named web-ec2 in public Subnet & attach web-sg security group to EC2 instance.

➔ Note :-

- Use data sources to fetch the AMI ID.
- Use Dynamic block for security group ingress & egress to add rules.
- Generate ssh key using aws_key_pair resource & map to EC2 instances.

Que 4 →

➔ Create below resources using Terraform.

- Create VPC with 2 Public and 2 Private subnets including IGW / NAT GW / RT & Subnet association.
- Create 1 Security Group named app-sg. Allow traffic on port 8080 from 0.0.0.0/0.
- Create 1 EC2 instance named app-ec2 in private Subnet & attach app-sg security group to EC2 instance.
- Create 1 Security Group named rds-sg. Allow traffic on port 3306 from 0.0.0.0/0.
- Create 1 MySQL RDS instance with aws_db_instance with type db.t3.medium. Attach rds-sg to RDS instance.

➔ Note :-

- Use data sources to fetch the AMI ID.
- Use Dynamic block for security group ingress & egress to add rules.
- Generate ssh key using aws_key_pair resource & map to EC2 instances.

Que 5 →

➔ Create three tier application architecture using Terraform.

➔ Three tier application architecture will require below resources.

VPC

- VPC with 2 Public and 2 Private subnets including IGW / NAT GW / RT & Subnet association.

ALB

- Create 1 Security Group named alb-sg. Allow traffic on port 80 & 443 from 0.0.0.0/0
- Create 1 Application Load Balancer in the public Subnet of your VPC & attach alb-sg security group to ALB.

WEB-SERVER

- Create 1 Security Group named web-sg. Allow traffic on port 80 from 0.0.0.0/0
- Create 1 EC2 instance named web-ec2 in public Subnet & attach web-sg security group to EC2 instance.

APPLICATION

- Create 1 Security Group named app-sg. Allow traffic on port 8080 from 0.0.0.0/0.
- Create 1 EC2 instance named app-ec2 in private Subnet & attach app-sg security group to EC2 instance.

DATABASE

- Create 1 Security Group named rds-sg. Allow traffic on port 3306 from 0.0.0.0/0.
- Create 1 MySQL RDS instance with aws_db_instance with type db.t3.medium. Attach rds-sg to RDS instance.

S3 BUCKET

- Create an S3 bucket to store the data.

➔ Note :-

- Use data sources to fetch the AMI ID.
- Use Dynamic block for security group ingress & egress to add rules.
- Generate ssh key using aws_key_pair resource & map to EC2 instances.

Thank You