**Terraform Script:**

provider "aws" {

region = "us-west-2"

}

**# Create VPC**

resource "aws\_vpc" "my\_vpc" {

cidr\_block = "10.0.0.0/16"

}

**# Create Private Subnet**

resource "aws\_subnet" "my\_subnet" {

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

cidr\_block = "10.0.1.0/24"

map\_public\_ip\_on\_launch = false

}

**# Create Security Group**

resource "aws\_security\_group" "ecs\_tasks" {

name\_prefix = "ecs-tasks-"

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

}

**# Create ECS Cluster**

resource "aws\_ecs\_cluster" "my\_cluster" {

name = "my-cluster"

}

**# Create ECS Task Definition**

resource "aws\_ecs\_task\_definition" "my\_task" {

family = "my-family"

network\_mode = "awsvpc"

requires\_compatibilities = ["FARGATE"]

cpu = "256"

memory = "512"

execution\_role\_arn = aws\_iam\_role.ecs\_execution\_role.arn

container\_definitions = jsonencode([{

name = "my-container"

image = "my-ecr-repo/my-image:latest"

portMappings = [{

containerPort = 80

hostPort = 80

}]

}])

}

**# Create IAM Role for ECS Execution**

resource "aws\_iam\_role" "ecs\_execution\_role" {

name = "ecs\_execution\_role"

assume\_role\_policy = jsonencode({

Version = "2012-10-17",

Statement = [{

Action = "sts:AssumeRole",

Effect = "Allow",

Principal = {

Service = "ecs-tasks.amazonaws.com"

}

}]

})

}

**# Create ECS Service**

resource "aws\_ecs\_service" "my\_service" {

name = "my-service"

cluster = aws\_ecs\_cluster.my\_cluster.id

task\_definition = aws\_ecs\_task\_definition.my\_task.arn

launch\_type = "FARGATE"

desired\_count = 2

network\_configuration {

subnets = [aws\_subnet.my\_subnet.id]

}

}

**# Create Load Balancer**

resource "aws\_lb" "my\_lb" {

name = "my-lb"

internal = false

load\_balancer\_type = "application"

security\_groups = [aws\_security\_group.ecs\_tasks.id]

subnets = [aws\_subnet.my\_subnet.id]

}

**# Output Load Balancer DNS**

output "load\_balancer\_dns" {

value = aws\_lb.my\_lb.dns\_name

}

---------------------------------------------------

Architecture Diagram

1. AWS VPC encapsulates ECS Cluster, RDS, and S3.

2. ECS Cluster contains multiple ECS Services.

3. ECS Services connect to RDS and S3.

4. Load Balancer routes traffic to ECS Services.

1. AWS VPC: The central component that encapsulates all other services.

- Private Subnet: Houses ECS Cluster, RDS instance.

- Public Subnet: Houses the Load Balancer.

2. AWS ECS Cluster:

- ECS Services: Runs the Docker containers.

- Task Definition: Specifies the Docker image and resources.

- IAM Role for ECS: Allows ECS tasks to interact with other AWS services.

3. AWS RDS (Postgres):

- Directly connected to ECS Services.

- Hosted within the private subnet of the VPC.

4. AWS S3 Bucket:

- Stores application data.

- Accessed by ECS services via IAM roles.

5. AWS Load Balancer (ELB):

- Routes incoming traffic to ECS services.

- Located in the public subnet.

6. AWS ECR:

- Stores the Docker image used by ECS.

- Connected to your CI/CD pipeline for image updates.

7. GitHub Repository:

- Source code and build instructions.

- Connected to a CI/CD pipeline that pushes Docker images to ECR.

**Data Flow**

1. User requests hit the Load Balancer.

2. Load Balancer routes the traffic to one of the ECS Services.

3. ECS Service processes the request, interacting with RDS for database operations and S3 for data storage.

**Deployment Steps**

1. Fork the GitHub repository.

2. Set up CI/CD to build Docker image and push to ECR.

3. Run **terraform init** and **terraform apply** to deploy the infrastructure.

**Configurations**

- ECS Task Definition: CPU=256, Memory=512

- RDS: Postgres, db.t2.micro

- S3: Standard storage class

.