* Documentation

Here is the steps that I have done in this project.1.install terraform 2.install AWS CLI and login with the AWS credentials 3.create a directory and create the files to execute the given resources using terraform script with .tf extension Main.tfprovider "aws" {  region = "us-east-1"}# VPCresource "aws\_vpc" "main" {  cidr\_block = var.vpc\_cidr  tags = {    Name = "main-vpc"  }}# Subnetsresource "aws\_subnet" "public" {  count = length(var.public\_subnet\_cidrs)  vpc\_id = aws\_vpc.main.id  cidr\_block = element(var.public\_subnet\_cidrs, count.index)  map\_public\_ip\_on\_launch = true  availability\_zone = element(var.availability\_zones, count.index)  tags = {    Name = "public-subnet-${count.index}"  }}resource "aws\_subnet" "private" {  count = length(var.private\_subnet\_cidrs)  vpc\_id = aws\_vpc.main.id  cidr\_block = element(var.private\_subnet\_cidrs, count.index)  availability\_zone = element(var.availability\_zones, count.index)  tags = {    Name = "private-subnet-${count.index}"  }}# Internet Gatewayresource "aws\_internet\_gateway" "main" {  vpc\_id = aws\_vpc.main.id  tags = {    Name = "main-igw"  }}# Public Route Tableresource "aws\_route\_table" "public" {  vpc\_id = aws\_vpc.main.id  route {    cidr\_block = "0.0.0.0/0"    gateway\_id = aws\_internet\_gateway.main.id  }  tags = {    Name = "public-route-table"  }}# Public Route Table Associationresource "aws\_route\_table\_association" "public" {  count = length(var.public\_subnet\_cidrs)  subnet\_id = element(aws\_subnet.public[\*].id, count.index)  route\_table\_id = aws\_route\_table.public.id}# Security Groupresource "aws\_security\_group" "web\_sg" {  vpc\_id = aws\_vpc.main.id  ingress {    from\_port = 80    to\_port = 80    protocol = "tcp"    cidr\_blocks = ["0.0.0.0/0"]  }  ingress {    from\_port = 22    to\_port = 22    protocol = "tcp"    cidr\_blocks = ["0.0.0.0/0"]  }  egress {    from\_port = 0    to\_port = 0    protocol = "-1"    cidr\_blocks = ["0.0.0.0/0"]  }  tags = {    Name = "web-sg"  }}# EC2 Instancesresource "aws\_instance" "web" {  count = var.instance\_count  ami = var.ami\_id  instance\_type = var.instance\_type  subnet\_id = element(aws\_subnet.public[\*].id, count.index % length(aws\_subnet.public[\*].id))  security\_groups = [aws\_security\_group.web\_sg.name]  tags = {    Name = "web-instance-${count.index}"  }}var.tfvariable "vpc\_cidr" {  description = "The CIDR block for the VPC."  default = "10.0.0.0/16"}variable "public\_subnet\_cidrs" {  description = "The CIDR blocks for the public subnets."  default = ["10.0.1.0/24", "10.0.2.0/24"]}variable "private\_subnet\_cidrs" {  description = "The CIDR blocks for the private subnets."  default = ["10.0.3.0/24", "10.0.4.0/24"]}variable "availability\_zones" {  description = "The availability zones to deploy resources in."  default = ["us-east-1a", "us-east-1b"]}variable "instance\_count" {  description = "Number of EC2 instances to deploy."  default = 2}variable "ami\_id" {  description = "The AMI ID for the EC2 instances."  default = "ami-0c55b159cbfafe1f0" # Example AMI ID for Amazon Linux 2}variable "instance\_type" {  description = "The instance type for the EC2 instances."  default = "t2.micro"}output.tfoutput "vpc\_id" {  description = "The ID of the VPC."  value = aws\_vpc.main.id}output "public\_subnet\_ids" {  description = "The IDs of the public subnets."  value = aws\_subnet.public[\*].id}output "private\_subnet\_ids" {  description = "The IDs of the private subnets."  value = aws\_subnet.private[\*].id}output "instance\_ids" {  description = "The IDs of the EC2 instances."  value = aws\_instance.web[\*].id}output "instance\_public\_ips" {  description = "The public IPs of the EC2 instances."  value = aws\_instance.web[\*].public\_ip}Using this script I have done with making resources using terraform.Execute commands ...>Terraform init>Terraform validate>Terraform plan>Terraform apply4. I have launched a bastion host instance by configuring all essential criterias like ami ,instance type, sg and all .by this instance bastion hosted to both web application instance and postgresdb instance.ssh to the private instances by managing the private keys using bastion hosting 5.install and configure web application in first instance >sudo yum update>sudo yum install httpd>sudo systemctl start httpd>sudo systemctl enable httpdInstall node.js>curl -sL [https://rpm.nodesource.com/setup\_16.x](https://rpm.nodesource.com/setup_16.x#https://rpm.nodesource.com/setup_16.x) | sudo bash ->sudo yum install -y nodejs6.install and configure postgresdb in second instance >sudo yum update>sudo yum install postgresql-server postgresql-contrib>sudo systemctl start postgresql>sudo systemctl enable postgresql7.Use private IP addresses for communication between the web application and PostgreSQL instances within the same VPC to enhance security.8. Setup application load balancer to the web application instance

* configure listeners and target group in order to avoid the network congestion.
* set up a dns name to the application from domain registrar and create dns record using route 53 and attach it with ALB.
* All of these can be monitored by using cloudwatch and also Target group do the health checks.

Thank you