< TERRAFORM FOR CLOUD STATE MANAGEMENT CONVERTED >

IMPLEMENTATION PLAN

VERSION 1.0 | 23/06/2022 to 25/06/2022

Purpose

Provision of JAVA App using Terraform automation in AWS Cloud.

About the Project

- *Cloud Management Team*
- *Deploy, Setup and Manage Infrastructure on the Cloud*
- *Heavy usage of Cloud Services*
- *Regular provisioning request and changes*

Problem

- *Infra Setup is complex process*
- *Human Errors in deployment*
- *Difficult to track and not centralized*
- *Resource wastage*
- *Not Repeatable*
- *Managing manually is time consuming task*

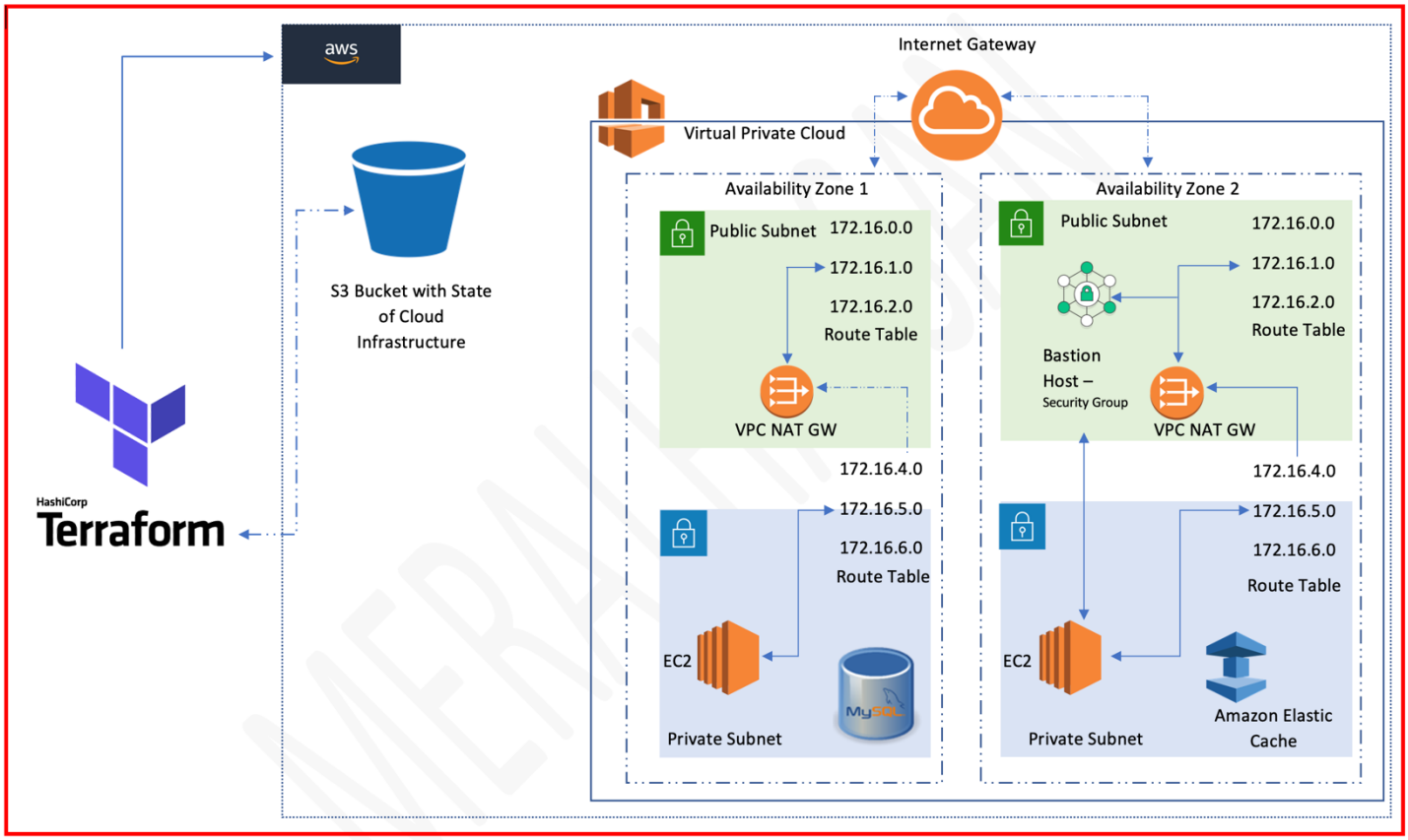
Solution

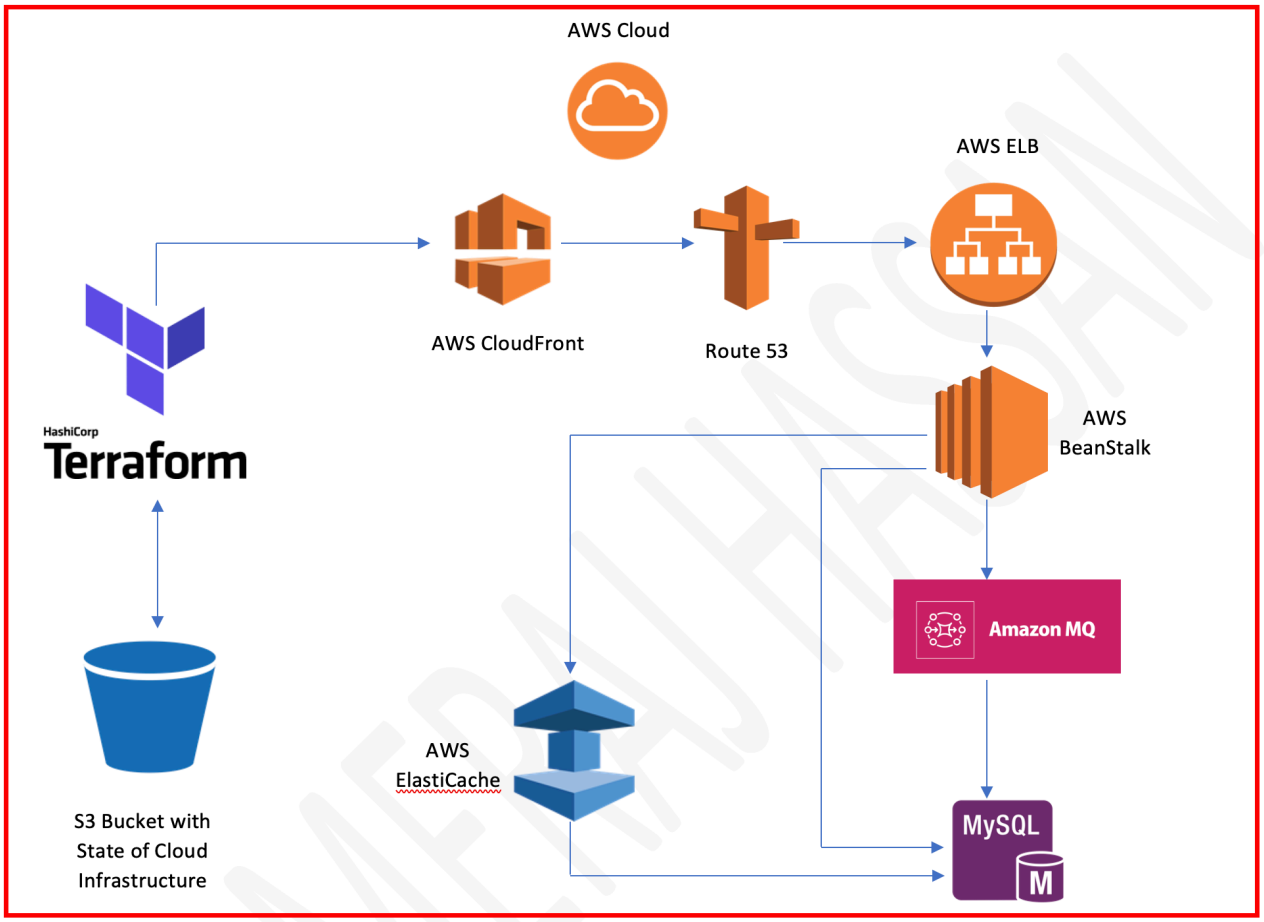
- *Configuration management of infrastructure*
- *Automation setup (No Human Error)*
- *Maintain State of Infrastructure*
- *Version Control [IAAC]*
- *Repeatable*
- *Reusable*

Tools

- *Terraform – Configuration Management AWS Infra*
- *AWS – AWS services [EC2 , VPC, NAT GW, S3 , Aamazon MQ, RDS]*
- *IDE – Intellij*

Architecture of Terraform Setup





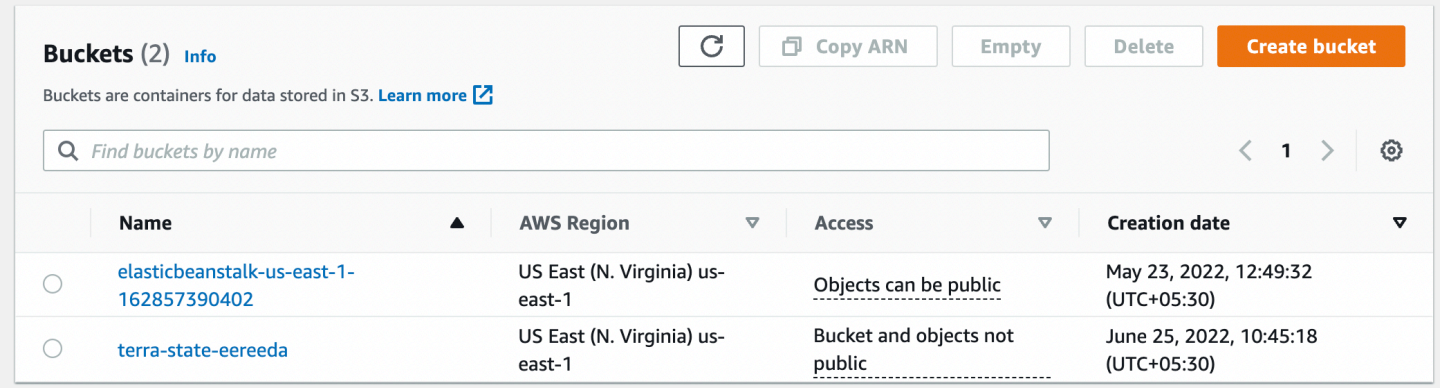
Flow of Execution

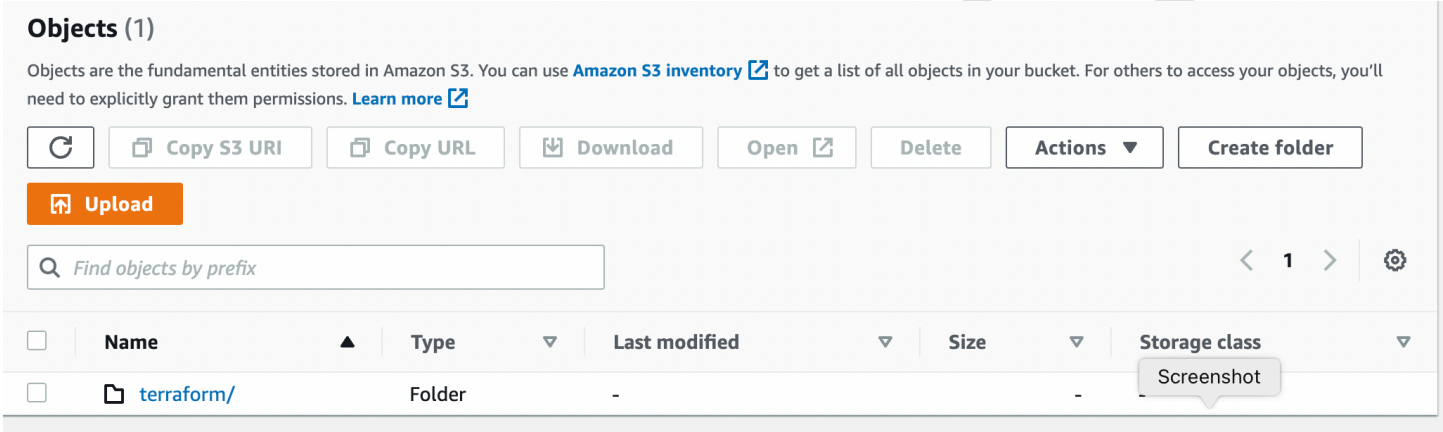
- *Setup Terraform with backend*
- *Setup VPC [Secure & HA]*
- *Provision BeanStalk Environment*
- *Provision Backend Services*
    - *RDS*
    - *Elastic Cache*
    - *Active MQ*
- *Security host, Key Pairs, Bastion host, etc.*

Prerequisite

- *AWS account*
- *GITHUB Accounts*

1. S3 for Backend





backend.tf

```
terraform {
  backend "s3" {
    bucket = "terra-state-eereeda"
    key = "terraform/backend"
    region = "us-east-1"
  }
}
```

$ terraform init

```
merajhassan@MERAJs-MacBook-Air DevOps_Projects % terraform init

Initializing the backend...

Successfully configured the backend "s3"! Terraform will automatically
use this backend unless the backend configuration changes.

Initializing provider plugins...

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

2. Variables & Providers

provider.tf

```
provider "aws" {
  region = var.AWS_REGION
}
```

vars.tf

```
variable AWS_REGION {
  default = "us-east-1"
}
variable AMIS {
  type = map
  default = {
    us-east-1 = "ami-08d4ac5b634553e16"
    us-east-2 = "ami-03e57de632660544c"
  }
}
variable PRIV_KEY_PATH {
  default = "eereeda-terraform-key"
}
variable PUB_KEY_PATH {
  default = "eereeda-terraform-key.pub"
}
variable USERNAME {
  default = "ubuntu"
}
variable MYIP {
  default = "183.83.39.124/32"
}

variable rmquser {
  default = "rabbit"
}
variable rmqpass {
  default = "eereeda@terraform12345"
}

variable dbuser {
  default = "admin"
}
variable dbpass {
  default = "admin123"
}
```

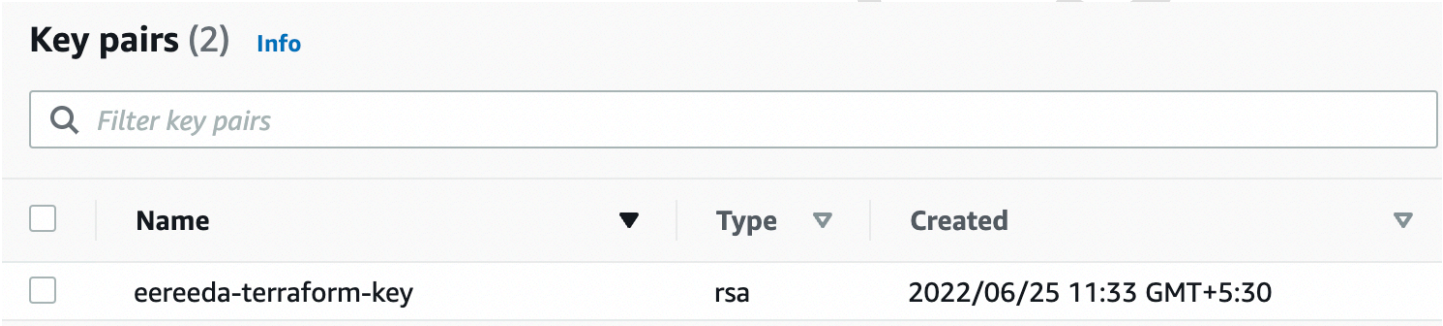
```
variable dbname {
  default = "accounts"
}

variable instance_count {
  default = "1"
}

variable VPC_NAME {
  default = "eereeda-VPC"
}

variable Zone1 {
  default = "us-east-1a"
}
variable Zone2 {
  default = "us-east-1b"
}
variable Zone3 {
  default = "us-east-1c"
}

variable VpcCIDR {
  default = "172.21.0.0/16"
}

variable PubSub1CIDR {
  default = "172.21.1.0/24"
}
variable PubSub2CIDR {
  default = "172.21.2.0/24"
}
variable PubSub3CIDR {
  default = "172.21.3.0/24"
}

variable privSub1CIDR {
  default = "172.21.4.0/24"
}
variable privSub2CIDR {
  default = "172.21.5.0/24"
}
variable privSub3CIDR {
  default = "172.21.6.0/24"
}
```

3. Key Pairs

```
merajhassan@MERAJs-MacBook-Air DevOps_Projects % ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/merajhassan/.ssh/id_rsa): eereeda-terraform-key

merajhassan@MERAJs-MacBook-Air DevOps_Projects % ls
backend-s3.tf                   eereeda-terraform-key.pub       vars.tf
eereeda-terraform-key           providers.tf
```

keypair.tf

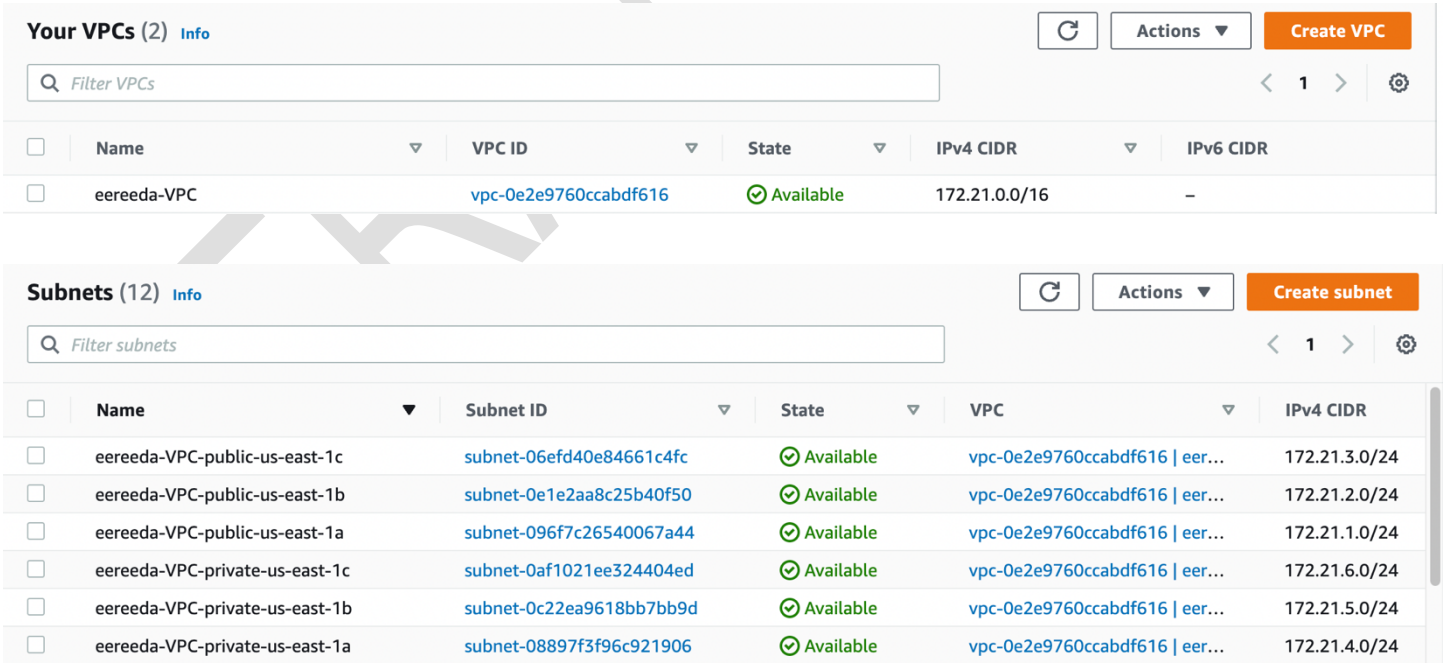
```
resource "aws_key_pair" "eereedakey" {
  key_name   = "eereeda-terraform-key"
  public_key = file(var.PUB_KEY_PATH)
}
```



4. VPC Module & Setup

URL for Terraform Registry: https://registry.terraform.io/
Module: terraform-aws-modules/vpc

```
module "vpc" {
  source = "terraform-aws-modules/vpc/aws"

  name             = var.VPC_NAME
  cidr             = var.VpcCIDR
  azs              = [var.Zone1, var.Zone2, var.Zone3]
  private_subnets  = [var.PrivSub1CIDR, var.PrivSub2CIDR, var.PrivSub3CIDR]
  public_subnets   = [var.PubSub1CIDR, var.PubSub2CIDR, var.PubSub3CIDR]

  enable_nat_gateway   = true
  single_nat_gateway   = true
  enable_dns_hostnames = true
  enable_dns_support   = true

  tags = {
    Terraform   = "true"
    Environment = "Prod"
  }
  vpc_tags = {
    Name = var.VPC_NAME
  }
}
```



Route tables (4) Info

Actions | Create route table

Filter route tables

1

Name	Route table ID	Explicit subnet associat...	Edge associations	Main	VPC
eereeda-VPC-public	rtb-082721215c988d23d	3 subnets	–	No	vpc-0e2e9760ccabdf616 | eer
eereeda-VPC-private	rtb-0c53259b14ce1f679	3 subnets	–	No	vpc-0e2e9760ccabdf616 | eer

rtb-082721215c988d23d / eereeda-VPC-public

Details | Routes | Subnet associations | Edge associations

Explicit subnet associations (3)

Find subnet association

Subnet ID	IPv4 CIDR
subnet-06efd40e84661c4fc / eereeda-VPC-public-us-east-1c	172.21.3.0/24
subnet-0e1e2aa8c25b40f50 / eereeda-VPC-public-us-east-1b	172.21.2.0/24
subnet-096f7c26540067a44 / eereeda-VPC-public-us-east-1a	172.21.1.0/24

rtb-0c53259b14ce1f679 / eereeda-VPC-private

Details | Routes | Subnet associations | Edge associations

Explicit subnet associations (3)

Find subnet association

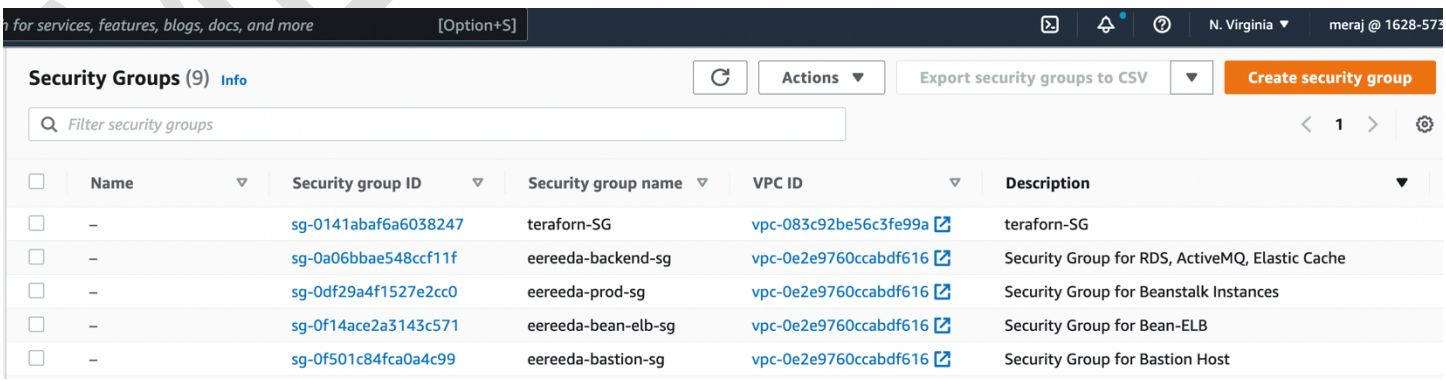
Subnet ID	IPv4 CIDR
subnet-08897f3f96c921906 / eereeda-VPC-private-us-east-1a	172.21.4.0/24
subnet-0af1021ee324404ed / eereeda-VPC-private-us-east-1c	172.21.6.0/24
subnet-0c22ea9618bb7bb9d / eereeda-VPC-private-us-east-1b	172.21.5.0/24

5. **Security Group Setup**

secgrp.tf

```
resource "aws_security_group" "eereeda-bean-elb-sg" {
  name        = "eereeda-bean-elb-sg"
  description = "Security Group for Bean-ELB"
  vpc_id      = module.vpc.vpc_id
  egress {
    from_port   = 0
    protocol    = "-1"
    to_port     = 0
    cidr_blocks = ["0.0.0.0/0"]
  }
  ingress {
    from_port   = 80
    protocol    = "tcp"
    to_port     = 80
    cidr_blocks = ["0.0.0.0/0"]
  }
}

resource "aws_security_group" "eereeda-bastion-sg" {
  name        = "eereeda-bastion-sg"
  description = "Security Group for Bastion Host"
  vpc_id      = module.vpc.vpc_id
  egress {
    from_port   = 0
    protocol    = "-1"
    to_port     = 0
    cidr_blocks = ["0.0.0.0/0"]
  }
  ingress {
    from_port   = 22
    protocol    = "tcp"
    to_port     = 22
    cidr_blocks = ["0.0.0.0/0"]
  }
}

resource "aws_security_group" "eereeda-prod-sg" {
  name        = "eereeda-prod-sg"
  description = "Security Group for Beanstalk Instances"
  vpc_id      = module.vpc.vpc_id
  egress {
    from_port   = 0
    protocol    = "-1"
    to_port     = 0
    cidr_blocks = ["0.0.0.0/0"]
```

```
  }
  ingress {
    from_port       = 22
    protocol        = "tcp"
    to_port         = 22
    security_groups = [aws_security_group.eereeda-bastion-sg.id]
  }
}

resource "aws_security_group" "eereeda-backend-sg" {
  name        = "eereeda-backend-sg"
  description = "Security Group for RDS, ActiveMQ, Elastic Cache"
  vpc_id      = module.vpc.vpc_id
  egress {
    from_port   = 0
    protocol    = "-1"
    to_port     = 0
    cidr_blocks = ["0.0.0.0/0"]
  }
  ingress {
    from_port       = 0
    protocol        = "-1"
    to_port         = 0
    security_groups = [aws_security_group.eereeda-prod-sg.id]
  }
}

resource "aws_security_group_rule" "sec_group_allow_itself" {
  from_port                = 0
  protocol                 = "tcp"
  security_group_id        = aws_security_group.eereeda-backend-sg.id
  source_security_group_id = aws_security_group.eereeda-backend-sg.id
  to_port                  = 65535
  type                     = "ingress"
}
```



6. **RDS, ElasticCache & AmazonMQ**

```
resource "aws_db_subnet_group" "eereeda-rds-subgrp" {
  name = "eereeda-rds-subgrp"
  subnet_ids =
[module.vpc.private_subnets[0],module.vpc.private_subnets[1],,mo
dule.vpc.private_subnets[2]]
  tags = {
    Name = "Subnet group for RDS"
  }
}

resource "aws_elasticache_subnet_group" "eereeda-ecache-subgrp"
{
  name       = "aws_elasticache_subnet_group"
  subnet_ids = [module.vpc.private_subnets[0],
module.vpc.private_subnets[1], ,module.vpc.private_subnets[2]]
  tags       = {
    Name = "Subnet group for ECACHE"
  }
}

resource "aws_db_instance" "eereeda-rds" {
  instance_class = "db.t2.micro"
  allocated_storage      = 20
  storage_type           = "gp2"
  engine                 = "mysql"
  engine_version         = "5.6.34"
  name                 = var.dbname
  username               = var.dbuser
  password               = var.dbpass
  parameter_group_name   = "default.mysql5.6"
  multi_az               = "false"
  publicly_accessible    = "false"
  skip_final_snapshot    = true
  db_subnet_group_name = aws_db_subnet_group.eereeda-rds-
subgrp.name
  vpc_security_group_ids = [aws_security_group.eereeda-backend-
sg.id]
}

resource "aws_elasticache_cluster" "eereeda-cache" {
  cluster_id = "eereeda-cache"
  engine               = "memcached"
```

```
  node_type            = "cache.t2.micro"
  num_cache_nodes      = 1
  parameter_group_name = "default.memcached1.5"
  port                 = 11211
  security_group_ids   = [aws_security_group.eereeda-backend-
sg.id]
  subnet_group_name    = aws_elasticache_subnet_group.eereeda-
ecache-subgrp.name
}

resource "aws_mq_broker" "eereeda-rmq" {
  broker_name        = "eereeda-rmq"
  engine_type        = "ActiveMQ"
  engine_version     = "5.15.0"
  host_instance_type = "mq.t2.micro"
  security_groups = [aws_security_group.eereeda-backend-sg.id]
  subnet_ids = [module.vpc.private_subnet[0]]
  user {
    password = var.rmquser
    username = var.rmqpass
  }
}
```

7. **BeanStalk Environment Setup**

bean-app.tf

```
resource "aws_elastic_beanstalk_application" "eereeda-prod" {
  name = "eereeda-prod"
}
```

bean-env.tf

```
resource "aws_elastic_beanstalk_environment" "eereeda-bean-prod" {
  name                = "eereeda-bean-prod"
  application         = aws_elastic_beanstalk_application.eereeda-
prod.name
  solution_stack_name = "64bit Amazon Linux 2 v4.1.1 running Tomcat 8.5
Corretto 11"
  cname_prefix        = "eereeda-bean-prod-domain"
  setting {
    name      = "VPCId"
    namespace = "aws:ec2:vpc"
    value     = module.vpc.vpc_id
  }
  setting {
    namespace = "aws:autoscaling:launchconfiguration"
    name      = "IamInstanceProfile"
    value     = "aws-elasticbeanstalk-ec2-role"
  }
  setting {
    namespace = "aws:ec2:vpc"
    name      = "AssociatePublicIpAddress"
    value     = "false"
  }

  setting {
    namespace = "aws:ec2:vpc"
    name      = "Subnets"
    value     = join(",", [module.vpc.private_subnets[0],
module.vpc.private_subnets[1], module.vpc.private_subnets[2]])
  }
  setting {
    namespace = "aws:ec2:vpc"
    name      = "ELBSubnets"
    value     = join(",", [module.vpc.public_subnets[0],
module.vpc.public_subnets[1], module.vpc.public_subnets[2]])
  }

  setting {
    namespace = "aws:autoscaling:launchconfiguration"
    name      = "InstanceType"
    value     = "t2.micro"
  }
```

```
  setting {
    namespace = "aws:autoscaling:launchconfiguration"
    name      = "EC2KeyName"
    value     = aws_key_pair.eereedakey.key_name
  }

  setting {
    namespace = "aws:autoscaling:asg"
    name      = "Availability Zones"
    value     = "Any 3"
  }
  setting {
    namespace = "aws:autoscaling:asg"
    name      = "MinSize"
    value     = "1"
  }
  setting {
    namespace = "aws:autoscaling:asg"
    name      = "MaxSize"
    value     = "4"
  }

  setting {
    namespace = "aws:elasticbeanstalk:application:environment"
    name      = "environment"
    value     = "prod"
  }
  setting {
    namespace = "aws:elasticbeanstalk:application:environment"
    name      = "LOGGING_APPENDER"
    value     = "GRAYLOG"
  }
  setting {
    namespace = "aws:elasticbeanstalk:healthreporting:system"
    name      = "SystemType"
    value     = "enhanced"
  }
  setting {
    namespace = "aws:autoscaling:updatepolicy:rollingupdate"
    name      = "RollingUpdateEnabled"
    value     = "true"
  }
  setting {
    namespace = "aws:autoscaling:updatepolicy:rollingupdate"
    name      = "RollingUpdateType"
    value     = "Health"
  }

  setting {
    namespace = "aws:autoscaling:updatepolicy:rollingupdate"
    name      = "MaxBatchSize"
    value     = "1"
```

```
  }
  setting {
    namespace = "aws:elb:loadbalancer"
    name      = "CrossZone"
    value     = "true"
  }

  setting {
    name      = "StickinessEnabled"
    namespace = "aws:elasticbeanstalk:environment:process:default"
    value     = "true"
  }

  setting {
    namespace = "aws:elasticbeanstalk:command"
    name      = "BatchSizeType"
    value     = "Fixed"
  }

  setting {
    namespace = "aws:elasticbeanstalk:command"
    name      = "BatchSize"
    value     = "1"
  }
  setting {
    namespace = "aws:elasticbeanstalk:command"
    name      = "DeploymentPolicy"
    value     = "Rolling"
  }

  setting {
    namespace = "aws:autoscaling:launchconfiguration"
    name      = "SecurityGroups"
    value     = aws_security_group.eereeda-prod-sg.id
  }

  setting {
    namespace = "aws:elbv2:loadbalancer"
    name      = "SecurityGroups"
    value     = aws_security_group.eereeda-bean-elb-sg.id
  }

  depends_on = [aws_security_group.eereeda-bean-elb-sg,
aws_security_group.eereeda-prod-sg]

}
```

8. **Bastion Host & DB Initialization**

db-deploy.tmpl

```
sudo apt update
sudo apt install git mysql-client -y
git clone -b vp-rem
https://github.com/merajafnan/DevOps_Projects.git
mysql -h ${rds-endpoint} -u ${dbuser} --password=${dbpass}
accounts --ssl-mode=DISABLED <
/home/ubuntu/DevOps_Projects/src/main/resources/db_backup.sql
```

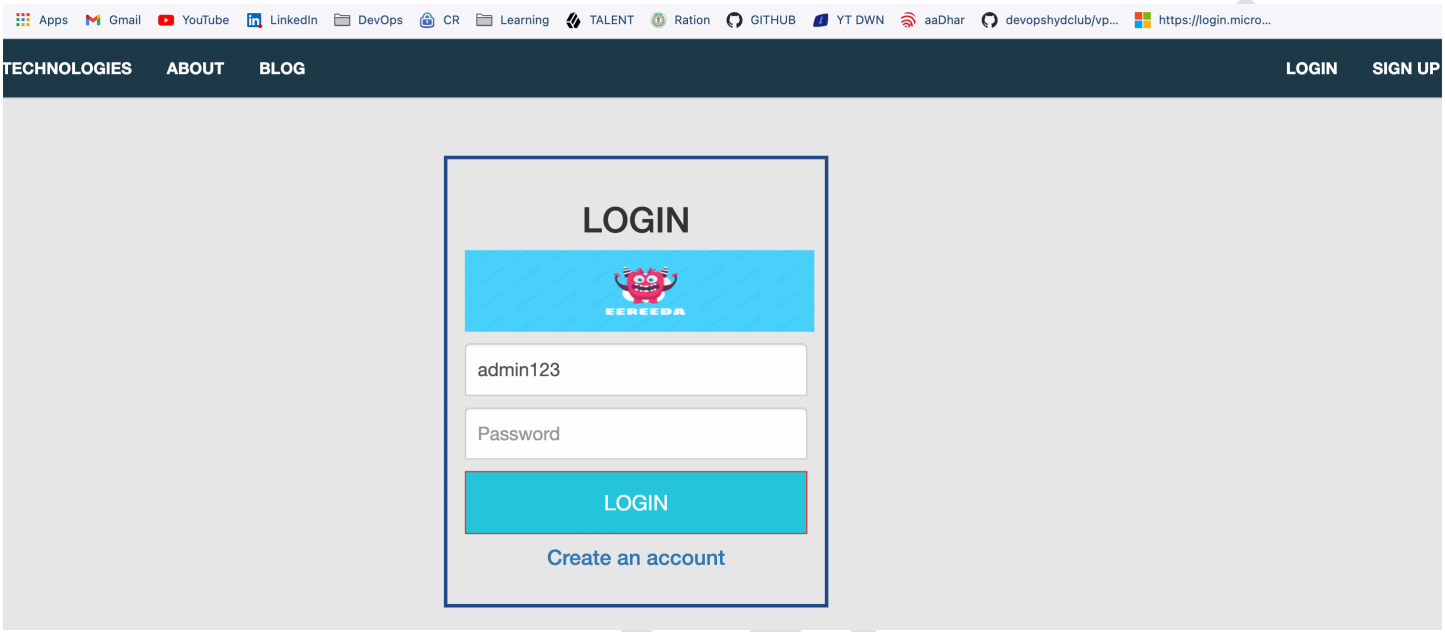
bastion-host.tf

```
resource "aws_instance" "eereeda-bastion" {
  ami                    = lookup(var.AMIS, var.AWS_REGION)
  instance_type          = "t2.micro"
  key_name               = aws_key_pair.eereedakey.key_name
  subnet_id              = module.vpc.public_subnets[0]
  count                  = var.instance_count
  vpc_security_group_ids = [aws_security_group.eereeda-bastion-sg.id]

  tags = {
    Name    = "eereeda-bastion"
    PROJECT = "eereeda"
  }

  provisioner "file" {
    content     = templatefile("templates/db-deploy.tmpl", { rds-endpoint
= aws_db_instance.eereeda-rds.address, dbuser = var.dbuser, dbpass =
var.dbpass })
    destination = "/tmp/eereeda-dbdeploy.sh"
  }

  provisioner "remote-exec" {
    inline = [
      "chmod +x /tmp/eereeda-dbdeploy.sh",
      "sudo /tmp/eereeda-dbdeploy.sh"
    ]
  }

  connection {
    user        = var.USERNAME
    private_key = file(var.PRIV_KEY_PATH)
    host        = self.public_ip
  }
  depends_on = [aws_db_instance.eereeda-rds]
}
```

9. Validate & Summarize

$ mvn Install

Elastic Beanstalk > Environment > eereeda-bean-prod > Upload and Deploy > eereeda-v2.war > Deploy



11. References

The following table summarizes the documents referenced in this plan.

DOCUMENT NAME	INSTRUCTOR	LOCATION
DevOps Beginners to Advanced	Imran Teli	https://www.udemy.com/course/decodingdevops/learn/lecture/28273912?start=0#overview