A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A computer screen with text

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect.

A computer screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Config.files:

Provider.tf

provider "aws" {

region = var.aws\_region

}

Variables.tf

variable "aws\_region" {

description = "AWS region to deploy resources"

default = "us-east-1"

}

variable "key\_pair\_name" {

description = "The name of the existing EC2 key pair to use"

type = string

}

variable "public\_key\_path" {

description = "Path to your SSH public key"

type = string

default = "~/.ssh/id\_rsa.pub"

}

variable "instance\_type" {

description = "EC2 instance type"

default = "t2.micro"

}

variable "ami\_id" {

description = "AMI ID for EC2 instance"

default = "ami-0c02fb55956c7d316" # Amazon Linux 2 in us-east-1

}

variable "github\_repo" {

description = "GitHub repository URL to clone"

default = "https://github.com/ikanko1989/SpringBootFootballClub.git"

}

Security\_groups.tf

resource "aws\_security\_group" "allow\_ssh\_ping\_8080" {

name = "allow\_ssh\_ping\_8080"

description = "Allow SSH, ICMP (ping), and TCP 8080"

ingress {

from\_port = 22

to\_port = 22

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

}

ingress {

from\_port = -1

to\_port = -1

protocol = "icmp"

cidr\_blocks = ["0.0.0.0/0"]

}

ingress {

from\_port = 8080

to\_port = 8080

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 = "allow-ssh-ping-8080"

}

}

main.tf

output "public\_ip" {

description = "Public IP of the EC2 instance"

value = aws\_instance.springboot\_app.public\_ip

}

ec2.tf

resource "aws\_instance" "springboot\_app" {

ami = var.ami\_id

instance\_type = var.instance\_type

key\_name = aws\_key\_pair.deployer.key\_name

security\_groups = [aws\_security\_group.allow\_ssh\_ping\_8080.name]

user\_data = file("${path.module}/user\_data.conf")

tags = {

Name = "SpringBootApp"

}

}

user\_data.conf

#!/bin/bash

# Update and install dependencies

yum update -y

yum install -y git curl wget tar

# Install Oracle JDK 21

mkdir -p /usr/lib/jvm

cd /usr/lib/jvm

curl -LO https://download.oracle.com/java/21/latest/jdk-21\_linux-x64\_bin.tar.gz

tar -xzf jdk-21\_linux-x64\_bin.tar.gz

alternatives --install /usr/bin/java java /usr/lib/jvm/jdk-21.0.8/bin/java 2

alternatives --install /usr/bin/javac javac /usr/lib/jvm/jdk-21.0.8/bin/javac 2

alternatives --set java /usr/lib/jvm/jdk-21.0.8/bin/java

alternatives --set javac /usr/lib/jvm/jdk-21.0.8/bin/javac

# Install Maven manually

MAVEN\_VERSION=3.8.8

cd /opt

curl -sLO https://archive.apache.org/dist/maven/maven-3/$MAVEN\_VERSION/binaries/apache-maven-$MAVEN\_VERSION-bin.tar.gz

tar -xzvf apache-maven-$MAVEN\_VERSION-bin.tar.gz

ln -s apache-maven-$MAVEN\_VERSION maven

echo "export M2\_HOME=/opt/maven" >> /etc/profile.d/maven.sh

echo "export PATH=\$M2\_HOME/bin:\$PATH" >> /etc/profile.d/maven.sh

chmod +x /etc/profile.d/maven.sh

source /etc/profile.d/maven.sh

echo "source /etc/profile.d/maven.sh" >> /home/ec2-user/.bash\_profile

chown ec2-user:ec2-user /home/ec2-user/.bash\_profile

# Clone your Spring Boot project

cd /home/ec2-user

git clone https://github.com/ikanko1989/SpringBootFootballClub.git

chown -R ec2-user:ec2-user /home/ec2-user/SpringBootFootballClub

# Run the app as ec2-user

cd /home/ec2-user/SpringBootFootballClub/FootballClub

mvn clean nohup spring-boot:run > /home/ec2-user/springboot.log 2>&1 &Terraform.tfvars

terraform.tfvars

key\_pair\_name = "deployer-key"

public\_key\_path = "~/.ssh/id\_rsa.pub"

keypair.tf

resource "aws\_key\_pair" "deployer" {

public\_key = file(var.public\_key\_path)

key\_name = var.key\_pair\_name

}

variables.tf

variable "aws\_region" {

description = "AWS region to deploy resources"

default = "us-east-1"

}

variable "key\_pair\_name" {

description = "The name of the existing EC2 key pair to use"

type = string

}

variable "public\_key\_path" {

description = "Path to your SSH public key"

type = string

default = "~/.ssh/id\_rsa.pub"

}

variable "instance\_type" {

description = "EC2 instance type"

default = "t2.micro"

}

variable "ami\_id" {

description = "AMI ID for EC2 instance"

default = "ami-0c02fb55956c7d316" # Amazon Linux 2 in us-east-1

}

variable "github\_repo" {

description = "GitHub repository URL to clone"

default = "https://github.com/ikanko1989/SpringBootFootballClub.git"

}

Workflow(init,plan,apply):

root@controlplane ~/code via 💠 default on ☁️ (us-east-1) ➜ ll

total 72

drwxr-xr-x 1 root root 4096 Jul 25 07:49 ./

drwx------ 1 root root 4096 Jul 25 07:36 ../

drwxr-xr-x 3 root root 4096 Jul 25 07:37 .terraform/

-rw-r--r-- 1 root root 1404 Jul 25 07:37 .terraform.lock.hcl

-rw-r--r-- 1 root root 530 Jul 4 10:24 README.md

-rw-r--r-- 1 root root 320 Jul 25 07:46 ec2.tf

-rw-r--r-- 1 root root 110 Jul 25 07:46 keypair.tf

-rw-r--r-- 1 root root 114 Jul 25 07:32 main.tf

-rw-r--r-- 1 root root 161 Jul 25 07:11 provider.tf

-rw-r--r-- 1 root root 658 Jul 25 07:25 security\_groups.tf

-rw-r--r-- 1 root root 10920 Jul 25 07:49 terraform.tfstate

-rw-r--r-- 1 root root 180 Jul 25 07:49 terraform.tfstate.backup

-rw-r--r-- 1 root root 66 Jul 25 07:09 terraform.tfvars

-rw-r--r-- 1 root root 1357 Jul 25 07:18 user\_data.conf

-rw-r--r-- 1 root root 735 Jul 25 07:36 variables.tf

root@controlplane ~/code via 💠 default on ☁️ (us-east-1) ➜ terraform init

Initializing the backend...

Initializing provider plugins...

- Reusing previous version of hashicorp/aws from the dependency lock file

- Using previously-installed hashicorp/aws v6.4.0

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.

root@controlplane ~/code via 💠 default on ☁️ (us-east-1) ➜ terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:

+ create

Terraform will perform the following actions:

# aws\_instance.springboot\_app will be created

+ resource "aws\_instance" "springboot\_app" {

+ ami = "ami-0c02fb55956c7d316"

+ arn = (known after apply)

+ associate\_public\_ip\_address = (known after apply)

+ availability\_zone = (known after apply)

+ disable\_api\_stop = (known after apply)

+ disable\_api\_termination = (known after apply)

+ ebs\_optimized = (known after apply)

+ enable\_primary\_ipv6 = (known after apply)

+ get\_password\_data = false

+ host\_id = (known after apply)

+ host\_resource\_group\_arn = (known after apply)

+ iam\_instance\_profile = (known after apply)

+ id = (known after apply)

+ instance\_initiated\_shutdown\_behavior = (known after apply)

+ instance\_lifecycle = (known after apply)

+ instance\_state = (known after apply)

+ instance\_type = "t2.micro"

+ ipv6\_address\_count = (known after apply)

+ ipv6\_addresses = (known after apply)

+ key\_name = "deployer-key"

+ monitoring = (known after apply)

+ outpost\_arn = (known after apply)

+ password\_data = (known after apply)

+ placement\_group = (known after apply)

+ placement\_partition\_number = (known after apply)

+ primary\_network\_interface\_id = (known after apply)

+ private\_dns = (known after apply)

+ private\_ip = (known after apply)

+ public\_dns = (known after apply)

+ public\_ip = (known after apply)

+ region = "us-east-1"

+ secondary\_private\_ips = (known after apply)

+ security\_groups = [

+ "allow\_ssh\_ping\_8080",

]

+ source\_dest\_check = true

+ spot\_instance\_request\_id = (known after apply)

+ subnet\_id = (known after apply)

+ tags = {

+ "Name" = "SpringBootApp"

}

+ tags\_all = {

+ "Name" = "SpringBootApp"

}

+ tenancy = (known after apply)

+ user\_data = <<-EOT

#!/bin/bash

# Update and install dependencies

yum update -y

yum install -y git curl wget tar

# Install Oracle JDK 21

mkdir -p /usr/lib/jvm

cd /usr/lib/jvm

curl -LO https://download.oracle.com/java/21/latest/jdk-21\_linux-x64\_bin.tar.gz

tar -xzf jdk-21\_linux-x64\_bin.tar.gz

alternatives --install /usr/bin/java java /usr/lib/jvm/jdk-21.0.8/bin/java 2

alternatives --install /usr/bin/javac javac /usr/lib/jvm/jdk-21.0.8/bin/javac 2

alternatives --set java /usr/lib/jvm/jdk-21.0.8/bin/java

alternatives --set javac /usr/lib/jvm/jdk-21.0.8/bin/javac

# Install Maven manually

MAVEN\_VERSION=3.8.8

cd /opt

curl -sLO https://archive.apache.org/dist/maven/maven-3/$MAVEN\_VERSION/binaries/apache-maven-$MAVEN\_VERSION-bin.tar.gz

tar -xzvf apache-maven-$MAVEN\_VERSION-bin.tar.gz

ln -s apache-maven-$MAVEN\_VERSION maven

echo "export M2\_HOME=/opt/maven" >> /etc/profile.d/maven.sh

echo "export PATH=\$M2\_HOME/bin:\$PATH" >> /etc/profile.d/maven.sh

chmod +x /etc/profile.d/maven.sh

source /etc/profile.d/maven.sh

# Clone your Spring Boot project

cd /home/ec2-user

git clone https://github.com/ikanko1989/SpringBootFootballClub.git

chown -R ec2-user:ec2-user /home/ec2-user/SpringBootFootballClub

# Run the app as ec2-user

cd /home/ec2-user/SpringBootFootballClub/FootballClub

mvn clean nohup spring-boot:run > /home/ec2-user/springboot.log 2>&1 &

EOT

+ user\_data\_base64 = (known after apply)

+ user\_data\_replace\_on\_change = false

+ vpc\_security\_group\_ids = (known after apply)

}

# aws\_key\_pair.deployer will be created

+ resource "aws\_key\_pair" "deployer" {

+ arn = (known after apply)

+ fingerprint = (known after apply)

+ id = (known after apply)

+ key\_name = "deployer-key"

+ key\_name\_prefix = (known after apply)

+ key\_pair\_id = (known after apply)

+ key\_type = (known after apply)

+ public\_key = "ssh-rsa  root@controlplane"

+ region = "us-east-1"

+ tags\_all = (known after apply)

}

# aws\_security\_group.allow\_ssh\_ping\_8080 will be created

+ resource "aws\_security\_group" "allow\_ssh\_ping\_8080" {

+ arn = (known after apply)

+ description = "Allow SSH, ICMP (ping), and TCP 8080"

+ egress = [

+ {

+ cidr\_blocks = [

+ "0.0.0.0/0",

]

+ description = ""

+ from\_port = 0

+ ipv6\_cidr\_blocks = []

+ prefix\_list\_ids = []

+ protocol = "-1"

+ security\_groups = []

+ self = false

+ to\_port = 0

},

]

+ id = (known after apply)

+ ingress = [

+ {

+ cidr\_blocks = [

+ "0.0.0.0/0",

]

+ description = ""

+ from\_port = -1

+ ipv6\_cidr\_blocks = []

+ prefix\_list\_ids = []

+ protocol = "icmp"

+ security\_groups = []

+ self = false

+ to\_port = -1

},

+ {

+ cidr\_blocks = [

+ "0.0.0.0/0",

]

+ description = ""

+ from\_port = 22

+ ipv6\_cidr\_blocks = []

+ prefix\_list\_ids = []

+ protocol = "tcp"

+ security\_groups = []

+ self = false

+ to\_port = 22

},

+ {

+ cidr\_blocks = [

+ "0.0.0.0/0",

]

+ description = ""

+ from\_port = 8080

+ ipv6\_cidr\_blocks = []

+ prefix\_list\_ids = []

+ protocol = "tcp"

+ security\_groups = []

+ self = false

+ to\_port = 8080

},

]

+ name = "allow\_ssh\_ping\_8080"

+ name\_prefix = (known after apply)

+ owner\_id = (known after apply)

+ region = "us-east-1"

+ revoke\_rules\_on\_delete = false

+ tags = {

+ "Name" = "allow-ssh-ping-8080"

}

+ tags\_all = {

+ "Name" = "allow-ssh-ping-8080"

}

+ vpc\_id = (known after apply)

}

Plan: 3 to add, 0 to change, 0 to destroy.

Changes to Outputs:

+ public\_ip = (known after apply)

────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply"

now.

root@controlplane ~/code via 💠 default on ☁️ (us-east-1) ➜ terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:

+ create

Terraform will perform the following actions:

# aws\_instance.springboot\_app will be created

+ resource "aws\_instance" "springboot\_app" {

+ ami = "ami-0c02fb55956c7d316"

+ arn = (known after apply)

+ associate\_public\_ip\_address = (known after apply)

+ availability\_zone = (known after apply)

+ disable\_api\_stop = (known after apply)

+ disable\_api\_termination = (known after apply)

+ ebs\_optimized = (known after apply)

+ enable\_primary\_ipv6 = (known after apply)

+ get\_password\_data = false

+ host\_id = (known after apply)

+ host\_resource\_group\_arn = (known after apply)

+ iam\_instance\_profile = (known after apply)

+ id = (known after apply)

+ instance\_initiated\_shutdown\_behavior = (known after apply)

+ instance\_lifecycle = (known after apply)

+ instance\_state = (known after apply)

+ instance\_type = "t2.micro"

+ ipv6\_address\_count = (known after apply)

+ ipv6\_addresses = (known after apply)

+ key\_name = "deployer-key"

+ monitoring = (known after apply)

+ outpost\_arn = (known after apply)

+ password\_data = (known after apply)

+ placement\_group = (known after apply)

+ placement\_partition\_number = (known after apply)

+ primary\_network\_interface\_id = (known after apply)

+ private\_dns = (known after apply)

+ private\_ip = (known after apply)

+ public\_dns = (known after apply)

+ public\_ip = (known after apply)

+ region = "us-east-1"

+ secondary\_private\_ips = (known after apply)

+ security\_groups = [

+ "allow\_ssh\_ping\_8080",

]

+ source\_dest\_check = true

+ spot\_instance\_request\_id = (known after apply)

+ subnet\_id = (known after apply)

+ tags = {

+ "Name" = "SpringBootApp"

}

+ tags\_all = {

+ "Name" = "SpringBootApp"

}

+ tenancy = (known after apply)

+ user\_data = <<-EOT

#!/bin/bash

# Update and install dependencies

yum update -y

yum install -y git curl wget tar

# Install Oracle JDK 21

mkdir -p /usr/lib/jvm

cd /usr/lib/jvm

curl -LO https://download.oracle.com/java/21/latest/jdk-21\_linux-x64\_bin.tar.gz

tar -xzf jdk-21\_linux-x64\_bin.tar.gz

alternatives --install /usr/bin/java java /usr/lib/jvm/jdk-21.0.8/bin/java 2

alternatives --install /usr/bin/javac javac /usr/lib/jvm/jdk-21.0.8/bin/javac 2

alternatives --set java /usr/lib/jvm/jdk-21.0.8/bin/java

alternatives --set javac /usr/lib/jvm/jdk-21.0.8/bin/javac

# Install Maven manually

MAVEN\_VERSION=3.8.8

cd /opt

curl -sLO https://archive.apache.org/dist/maven/maven-3/$MAVEN\_VERSION/binaries/apache-maven-$MAVEN\_VERSION-bin.tar.gz

tar -xzvf apache-maven-$MAVEN\_VERSION-bin.tar.gz

ln -s apache-maven-$MAVEN\_VERSION maven

echo "export M2\_HOME=/opt/maven" >> /etc/profile.d/maven.sh

echo "export PATH=\$M2\_HOME/bin:\$PATH" >> /etc/profile.d/maven.sh

chmod +x /etc/profile.d/maven.sh

source /etc/profile.d/maven.sh

echo "source /etc/profile.d/maven.sh" >> /home/ec2-user/.bash\_profile

chown ec2-user:ec2-user /home/ec2-user/.bash\_profile

chmod +x /etc/profile.d/maven.sh

source /etc/profile.d/maven.sh

# Clone your Spring Boot project

cd /home/ec2-user

git clone https://github.com/ikanko1989/SpringBootFootballClub.git

chown -R ec2-user:ec2-user /home/ec2-user/SpringBootFootballClub

# Run the app as ec2-user

sudo -u ec2-user bash -c "

# Run the app as ec2-user

cd /home/ec2-user/SpringBootFootballClub/FootballClub

mvn clean nohup spring-boot:run > /home/ec2-user/springboot.log 2>&1 &

EOT

+ user\_data\_base64 = (known after apply)

+ user\_data\_replace\_on\_change = false

+ vpc\_security\_group\_ids = (known after apply)

}

# aws\_key\_pair.deployer will be created

+ resource "aws\_key\_pair" "deployer" {

+ arn = (known after apply)

+ fingerprint = (known after apply)

+ id = (known after apply)

+ key\_name = "deployer-key"

+ key\_name\_prefix = (known after apply)

+ key\_pair\_id = (known after apply)

+ key\_type = (known after apply)

+ public\_key = "ssh-rsa  root@controlplane"

+ region = "us-east-1"

+ tags\_all = (known after apply)

}

# aws\_security\_group.allow\_ssh\_ping\_8080 will be created

+ resource "aws\_security\_group" "allow\_ssh\_ping\_8080" {

+ arn = (known after apply)

+ description = "Allow SSH, ICMP (ping), and TCP 8080"

+ egress = [

+ {

+ cidr\_blocks = [

+ "0.0.0.0/0",

]

+ description = ""

+ from\_port = 0

+ ipv6\_cidr\_blocks = []

+ prefix\_list\_ids = []

+ protocol = "-1"

+ security\_groups = []

+ self = false

+ to\_port = 0

},

]

+ id = (known after apply)

+ ingress = [

+ {

+ cidr\_blocks = [

+ "0.0.0.0/0",

]

+ description = ""

+ from\_port = -1

+ ipv6\_cidr\_blocks = []

+ prefix\_list\_ids = []

+ protocol = "icmp"

+ security\_groups = []

+ self = false

+ to\_port = -1

},

+ {

+ cidr\_blocks = [

+ "0.0.0.0/0",

]

+ description = ""

+ from\_port = 22

+ ipv6\_cidr\_blocks = []

+ prefix\_list\_ids = []

+ protocol = "tcp"

+ security\_groups = []

+ self = false

+ to\_port = 22

},

+ {

+ cidr\_blocks = [

+ "0.0.0.0/0",

]

+ description = ""

+ from\_port = 8080

+ ipv6\_cidr\_blocks = []

+ prefix\_list\_ids = []

+ protocol = "tcp"

+ security\_groups = []

+ self = false

+ to\_port = 8080

},

]

+ name = "allow\_ssh\_ping\_8080"

+ name\_prefix = (known after apply)

+ owner\_id = (known after apply)

+ region = "us-east-1"

+ revoke\_rules\_on\_delete = false

+ tags = {

+ "Name" = "allow-ssh-ping-8080"

}

+ tags\_all = {

+ "Name" = "allow-ssh-ping-8080"

}

+ vpc\_id = (known after apply)

}

Plan: 3 to add, 0 to change, 0 to destroy.

Changes to Outputs:

+ public\_ip = (known after apply)

Do you want to perform these actions?

Terraform will perform the actions described above.

Only 'yes' will be accepted to approve.

Enter a value: yes

aws\_key\_pair.deployer: Creating...

aws\_security\_group.allow\_ssh\_ping\_8080: Creating...

aws\_key\_pair.deployer: Creation complete after 1s [id=deployer-key]

aws\_security\_group.allow\_ssh\_ping\_8080: Creation complete after 3s [id=sg-07200bfd5463e5791]

aws\_instance.springboot\_app: Creating...

aws\_instance.springboot\_app: Still creating... [10s elapsed]

aws\_instance.springboot\_app: Creation complete after 14s [id=i-0f93898e794f1a8ea]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.

Outputs:

public\_ip = "3.80.49.220"

root@controlplane ~/code via 💠 default on ☁️ (us-east-1) ➜ terraform show

# aws\_instance.springboot\_app:

resource "aws\_instance" "springboot\_app" {

ami = "ami-0c02fb55956c7d316"

arn = "arn:aws:ec2:us-east-1:381491834499:instance/i-0f93898e794f1a8ea"

associate\_public\_ip\_address = true

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

disable\_api\_stop = false

disable\_api\_termination = false

ebs\_optimized = false

get\_password\_data = false

hibernation = false

id = "i-0f93898e794f1a8ea"

instance\_initiated\_shutdown\_behavior = "stop"

instance\_state = "running"

instance\_type = "t2.micro"

ipv6\_address\_count = 0

ipv6\_addresses = []

key\_name = "deployer-key"

monitoring = false

placement\_partition\_number = 0

primary\_network\_interface\_id = "eni-0dff93301c1e99e30"

private\_dns = "ip-172-31-19-110.ec2.internal"

private\_ip = "172.31.19.110"

public\_dns = "ec2-3-80-49-220.compute-1.amazonaws.com"

public\_ip = "3.80.49.220"

region = "us-east-1"

secondary\_private\_ips = []

security\_groups = [

"allow\_ssh\_ping\_8080",

]

source\_dest\_check = true

subnet\_id = "subnet-00cb5868a544013aa"

tags = {

"Name" = "SpringBootApp"

}

tags\_all = {

"Name" = "SpringBootApp"

}

tenancy = "default"

user\_data = <<-EOT

#!/bin/bash

# Update and install dependencies

yum update -y

yum install -y git curl wget tar

# Install Oracle JDK 21

mkdir -p /usr/lib/jvm

cd /usr/lib/jvm

curl -LO https://download.oracle.com/java/21/latest/jdk-21\_linux-x64\_bin.tar.gz

tar -xzf jdk-21\_linux-x64\_bin.tar.gz

alternatives --install /usr/bin/java java /usr/lib/jvm/jdk-21.0.8/bin/java 2

alternatives --install /usr/bin/javac javac /usr/lib/jvm/jdk-21.0.8/bin/javac 2

alternatives --set java /usr/lib/jvm/jdk-21.0.8/bin/java

alternatives --set javac /usr/lib/jvm/jdk-21.0.8/bin/javac

# Install Maven manually

MAVEN\_VERSION=3.8.8

cd /opt

curl -sLO https://archive.apache.org/dist/maven/maven-3/$MAVEN\_VERSION/binaries/apache-maven-$MAVEN\_VERSION-bin.tar.gz

tar -xzvf apache-maven-$MAVEN\_VERSION-bin.tar.gz

ln -s apache-maven-$MAVEN\_VERSION maven

echo "export M2\_HOME=/opt/maven" >> /etc/profile.d/maven.sh

echo "export PATH=\$M2\_HOME/bin:\$PATH" >> /etc/profile.d/maven.sh

chmod +x /etc/profile.d/maven.sh

source /etc/profile.d/maven.sh

chmod +x /etc/profile.d/maven.sh

source /etc/profile.d/maven.sh

# Clone your Spring Boot project

cd /home/ec2-user

git clone https://github.com/ikanko1989/SpringBootFootballClub.git

chown -R ec2-user:ec2-user /home/ec2-user/SpringBootFootballClub

# Run the app as ec2-user

cd /home/ec2-user/SpringBootFootballClub/FootballClub

mvn clean nohup spring-boot:run > /home/ec2-user/springboot.log 2>&1 &

EOT

user\_data\_replace\_on\_change = false

vpc\_security\_group\_ids = [

"sg-07200bfd5463e5791",

]

capacity\_reservation\_specification {

capacity\_reservation\_preference = "open"

}

cpu\_options {

core\_count = 1

threads\_per\_core = 1

}

credit\_specification {

cpu\_credits = "standard"

}

enclave\_options {

enabled = false

}

maintenance\_options {

auto\_recovery = "default"

}

metadata\_options {

http\_endpoint = "enabled"

http\_protocol\_ipv6 = "disabled"

http\_put\_response\_hop\_limit = 1

http\_tokens = "optional"

instance\_metadata\_tags = "disabled"

}

private\_dns\_name\_options {

enable\_resource\_name\_dns\_a\_record = false

enable\_resource\_name\_dns\_aaaa\_record = false

hostname\_type = "ip-name"

}

root\_block\_device {

delete\_on\_termination = true

device\_name = "/dev/xvda"

encrypted = false

iops = 100

tags = {}

tags\_all = {}

throughput = 0

volume\_id = "vol-090169513fc26d6b1"

volume\_size = 8

volume\_type = "gp2"

}

}

# aws\_key\_pair.deployer:

resource "aws\_key\_pair" "deployer" {

arn = "arn:aws:ec2:us-east-1:381491834499:key-pair/deployer-key"

fingerprint = "51:8f:7f:dc:ab:d0:a1:1e:16:42:29:d0:be:94:8f:ce"

id = "deployer-key"

key\_name = "deployer-key"

key\_pair\_id = "key-09456f5a04b49bcc8"

key\_type = "rsa"

public\_key = "ssh-rsa  root@controlplane"

region = "us-east-1"

tags\_all = {}

}

# aws\_security\_group.allow\_ssh\_ping\_8080:

resource "aws\_security\_group" "allow\_ssh\_ping\_8080" {

arn = "arn:aws:ec2:us-east-1:381491834499:security-group/sg-07200bfd5463e5791"

description = "Allow SSH, ICMP (ping), and TCP 8080"

egress = [

{

cidr\_blocks = [

"0.0.0.0/0",

]

description = ""

from\_port = 0

ipv6\_cidr\_blocks = []

prefix\_list\_ids = []

protocol = "-1"

security\_groups = []

self = false

to\_port = 0

},

]

id = "sg-07200bfd5463e5791"

ingress = [

{

cidr\_blocks = [

"0.0.0.0/0",

]

description = ""

from\_port = -1

ipv6\_cidr\_blocks = []

prefix\_list\_ids = []

protocol = "icmp"

security\_groups = []

self = false

to\_port = -1

},

{

cidr\_blocks = [

"0.0.0.0/0",

]

description = ""

from\_port = 22

ipv6\_cidr\_blocks = []

prefix\_list\_ids = []

protocol = "tcp"

security\_groups = []

self = false

to\_port = 22

},

{

cidr\_blocks = [

"0.0.0.0/0",

]

description = ""

from\_port = 8080

ipv6\_cidr\_blocks = []

prefix\_list\_ids = []

protocol = "tcp"

security\_groups = []

self = false

to\_port = 8080

},

]

name = "allow\_ssh\_ping\_8080"

owner\_id = "381491834499"

region = "us-east-1"

revoke\_rules\_on\_delete = false

tags = {

"Name" = "allow-ssh-ping-8080"

}

tags\_all = {

"Name" = "allow-ssh-ping-8080"

}

vpc\_id = "vpc-06f9b49013f2e0d5b"

}

Outputs:

public\_ip = "3.80.49.220"

root@controlplane ~/code via 💠 default on ☁️ (us-east-1) ➜ terraform state list

aws\_instance.springboot\_app

aws\_key\_pair.deployer

aws\_security\_group.allow\_ssh\_ping\_8080

root@controlplane ~/code via 💠 default on ☁️ (us-east-1) ➜ ping 3.80.49.220

PING 3.80.49.220 (3.80.49.220) 56(84) bytes of data.

64 bytes from 3.80.49.220: icmp\_seq=1 ttl=61 time=28.6 ms

64 bytes from 3.80.49.220: icmp\_seq=2 ttl=61 time=28.3 ms

64 bytes from 3.80.49.220: icmp\_seq=3 ttl=61 time=28.0 ms

^C

--- 3.80.49.220 ping statistics ---

3 packets transmitted, 3 received, 0% packet loss, time 2002ms

rtt min/avg/max/mdev = 28.008/28.298/28.612/0.247 ms

root@controlplane ~/code via 💠 default on ☁️ (us-east-1) ➜ ssh -i ~/.ssh/id\_rsa ec2-user@3.80.49.220

The authenticity of host '3.80.49.220 (3.80.49.220)' can't be established.

ED25519 key fingerprint is SHA256:do66K7naakfqrLx9RtOPnHFYmSGTWK8LCMYuzBFIPvI.

This key is not known by any other names.

Are you sure you want to continue connecting (yes/no/[fingerprint])? yes

Warning: Permanently added '3.80.49.220' (ED25519) to the list of known hosts.

\_\_| \_\_|\_ )

\_| ( / Amazon Linux 2 AMI

\_\_\_|\\_\_\_|\_\_\_|

https://aws.amazon.com/amazon-linux-2/

54 package(s) needed for security, out of 98 available

Run "sudo yum update" to apply all updates.

[ec2-user@ip-172-31-19-110 ~]$ ls

[ec2-user@ip-172-31-19-110 ~]$ ll

total 0

[ec2-user@ip-172-31-19-110 ~]$