(ssh -i "C:\Users\karnam harika\.ssh\investment-banking-backend-key.pem" ec2-user@16.171.40.161) for backend enter in powershell

Deployment in aws

Frontend : ng build -c production generate one Output location: G:\cap\Frontend\Investment-Banking-Frontend\dist\Investment-Banking-Frontend like this

Create an S3 Bucket

1. Log in to **AWS Management Console** → go to **S3**.
2. Click **Create bucket**.
3. Fill in:
   - **Bucket name:** e.g., `investment-banking-frontend` (must be globally unique).
   - **Region:** Choose the nearest region to you.
4. **Block Public Access:** uncheck **"Block all public access"**. AWS will warn you → confirm that you want the bucket to be public.
5. Click **Create bucket**.

Enable Static Website Hosting

1. Click your bucket → go to **Properties** → **Static website hosting**.
2. Select **Enable**.
3. **Hosting type:** Choose **Host a static website**.
4. **Index document:** `index.html`
5. **Error document:** `index.html` (important for Angular routing)
6. Save changes.
7. You'll see a **bucket website endpoint** URL. Keep it; this is your app URL.

(http://investment-banking-frontend.s3-website.eu-north-1.amazonaws.com)

Upload Angular Build Files

1. Go to your bucket → **Upload** → **Add files**.
2. Select **all files** inside:

```
G:\cap\Frontend\Investment-Banking-Frontend\dist\Investment-Banking-
Frontend\Browser
```

3. Click **Upload**.

 Make sure the files are uploaded at the**root of the bucket**, not inside a subfolder.

Add the Bucket Policy Again

1. Go to **Permissions → Bucket policy → Edit**.
2. Paste this JSON (replace `investment-banking-frontend` with your bucket name):

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadGetObject",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::investment-banking-frontend/*"
    }
  ]
}
```

3. Save changes.

Test Website

Open:

```
Properties → Static website hosting → Endpoint
```

(http://investment-banking-frontend.s3-website.eu-north-1.amazonaws.com) link for frontend

Deployment

Backend (add jar)

: Create EC2 Instance

1. Login to **AWS Console**
2. Go to **EC2 → Launch instance**
3. Set:
   - **Name:** `investment-banking-backend`
   - **AMI:** Amazon Linux 2
   - **Instance type:** `t2.micro` (FREE tier)

Key Pair Name (You Can Choose Anything)

Use a **meaningful name** so you remember what it's for.

☐ **Recommended name:**

```
investment-banking-backend-key
```

(Any name is fine, but this is clean and professional.)

☐ Key Pair Type (IMPORTANT)

When AWS asks:

Key pair type:
☐ Select **RSA**

Private key file format:
☐ Select **.pem**

1. **Network settings**
   - Allow **SSH (22)** – My IP
   - Allow **Custom TCP**:
     - Port: `8080`
     - Source: `Anywhere (0.0.0.0/0)`
2. Click **Launch Instance**

☐ EC2 is ready

STEP 2.1: Get EC2 Public IP

1. Go to **AWS Console** → **EC2**
2. Click **Instances**
3. Click your instance
4. Copy:

```
Public IPv4 address
```

Example:

(16.171.40.161)

Move Key File to Correct Location

1. Find your `.pem` file (Downloads folder)
2. Move it to:

```
C:\Users\soft\.ssh\
```

Final path example:

```
C:\Users\soft\.ssh\investment-banking-backend-key.pem
```

 Create `.ssh` folder if it doesn’t exist.

Open PowerShell (IMPORTANT)

1. Press **Windows + R**
2. Type:

```
powershell
```

3. Press Enter

STEP 2.4: Connect Using SSH Command

In PowerShell, run:

```
ssh -i C:\Users\soft\.ssh\investment-banking-backend-key.pem ec2-
user@<PUBLIC-IP>
```

Replace `<PUBLIC-IP>` with your actual IP.

Example:

(ssh -i "C:\Users\karnam harika\.ssh\investment-banking-backend-key.pem" ec2-user@16.171.40.161) enter

First-Time Login Message

If you see:

```
Are you sure you want to continue connecting (yes/no)?
```

Type:

```
yes
```

Press Enter.

SUCCESS MESSAGE (What You SHOULD See)

```
[ec2-user@ip-172-31-xx-xx ~]$
```

□ **You are now connected to EC2!**

Install Java on EC2 (Required for Spring Boot)

sudo yum update –y

sudo yum install java-17-amazon-corretto –y

java –version

Open PowerShell (NOT EC2)

1. Press **Windows + R**
2. Type:

```
powershell
```

3. Press Enter

□□ This must be **your local PowerShell**, not EC2 terminal.

□ Step 3: Run SCP Command (IMPORTANT)

Again open powershell and use that scp--- down one

Because:

- Your path has **spaces**
- Your username has **spaces**

□ Use this **exact format**:

```
scp -i "C:\Users\karnam harika\.ssh\investment-banking-backend-key.pem"
"G:\cap\Backend\Investment-Banking-Backend\target\Investment-Banking-
Backend-0.0.1-SNAPSHOT.jar" ec2-user@16.171.40.161:/home/ec2-user/
```

□ Replace:

- JAR name if different
- IP if different

If fails

Do that jar as zip and repat the process

If fails

Method 2: Upload from GitHub (Very Stable)

1. Push backend project to **GitHub**
2. On EC2:

```
sudo yum install git -y
git clone <your-repo-url>
cd Investment-Banking-Backend
mvn clean package
java -jar target/*.jar
```

(if dependencies in middle fail use this mvn clean install –DskipTests)

(u should get BUILD sucess)

Verify JAR Creation

```
ls target
```

You should see:

```
Investment-Banking-Backend-0.0.1-SNAPSHOT.jar
Investment-Banking-Backend-0.0.1-SNAPSHOT.jar.original
```

Use ONLY the `.jar` (not `.original`)

Create RDS MySQL Instance

Step 1: Create RDS MySQL Instance

1. Go to **AWS Console → RDS → Databases → Create database**
2. **Choose engine**: MySQL
   (or Amazon Aurora MySQL if you want Aurora)
3. **Templates**: Free tier (for testing)
4. **DB instance identifier**: `investment-backend-db`
5. **Credentials**:
   - Master username: `admin`
   - Master password: `YourPassword123`
6. **DB instance size**: `db.t2.micro` (free tier)
7. **Storage**: Default 20GB (free tier)
8. **Connectivity**:

    - VPC: default
    - Public access: Yes (for now, testing)
    - VPC security group: default or create new
        - Add **Inbound rule**: MySQL/Aurora, port **3306**, source **EC2 Security Group**
9. Click **Create database**

- Wait 5–10 minutes until the status is **Available**

Get RDS Endpoint

1. Click on your database instance
2. Find **Endpoint** (something like):

```
investment-backend-db.abcdefg123.us-east-1.rds.amazonaws.com
```

3. Copy this — we'll use it in Spring Boot

Example(investment-backend-db.c98agowiq32f.eu-north-1.rds.amazonaws.com)

Step 3: Configure Spring Boot for RDS

Edit your `application.properties` (or `application.yml`) on EC2:

```
spring.datasource.url=jdbc:mysql://<RDS-ENDPOINT>:3306/<DB-
NAME>?useSSL=false&serverTimezone=UTC
spring.datasource.username=admin
spring.datasource.password=YourPassword123
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```

example(

spring.application.name=Investment-Banking-Backend

================================

# SERVER

================================

server.port=8080

================================

```
# DATABASE (AWS RDS)

# ================================

spring.datasource.url=jdbc:mysql://investment-backend-db.c98agowiq32f.eu-north-1.rds.amazonaws.com:3306/backend?useSSL=false&serverTimezone=UTC

spring.datasource.username=admin

spring.datasource.password=YOUR_RDS_PASSWORD

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver


# ================================

# JPA

# ================================

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=false

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect


# ================================

# JWT

# ================================

jwt.secret=CHANGE_THIS_TO_STRONG_SECRET_IN_PROD

jwt.expiration=86400000


# ================================

# CORS (AWS S3 FRONTEND URL)

# ================================

cors.allowed-origins=http://YOUR-S3-BUCKET-NAME.s3-website.eu-north-1.amazonaws.com)
```

Find EC2 Security Group

1. Go to **AWS Console → EC2**
2. Click **Instances**
3. Click your **backend EC2 instance**
4. Scroll down → **Security**
5. Copy the **Security group name**
   - Example: `launch-wizard-1`
   - Or `backend-ec2-sg`

 Keep this name ex(sg-08430439d02e0fefe)

Go to RDS Security Group

1. Go to **AWS Console → RDS**
2. Click **Databases**
3. Click **investment-backend-db**
4. Scroll to **Connectivity & security**
5. Under **VPC security groups**, click the **security group link**
   - Example: `rds-launch-wizard-1`

 This opens **EC2 → Security Groups page**

Edit Inbound Rules (MOST IMPORTANT)

1. Click **Inbound rules**
2. Click **Edit inbound rules**
3. Click **Add rule**

Fill exactly like this:

Field	Value
Type	MySQL/Aurora
Port	3306 (auto)
Source	**Security group**
Source value	**EC2 security group name** (from Step 1)

How to Identify Correct EC2 Security Group

If you're unsure which SG to select:

1. Go to **EC2 → Instances**
2. Click your backend instance

3. Scroll to **Security**
4. Copy **Security Group ID / Name**
5. Come back here and select **that same SG**