Please explain steps for setting up of free MySql service on this SpringBoot appln. j hosted on AWS

To set up a **free MySQL service** for your Spring Boot application hosted on AWS, you'll be leveraging the **AWS Free Tier**. This tier offers a limited amount of usage of certain AWS services for free, typically for the first 12 months after you create your AWS account.

For databases, the AWS Free Tier includes **Amazon RDS (Relational Database Service)**, which allows you to run managed instances of databases like MySQL, PostgreSQL, etc., up to certain monthly limits.

Here are the steps to set up a Free Tier eligible MySQL database on AWS RDS and connect your Spring Boot application to it:

**Part 1: Setting up MySQL on AWS RDS (Free Tier Eligible)**

1. **Log in to AWS Management Console:**
   * Go to aws.amazon.com and log in to your account.
2. **Navigate to RDS:**
   * In the search bar at the top, type "RDS" and select "RDS - Relational Database Service".
3. **Create Database:**
   * On the RDS dashboard, click the orange **"Create database"** button.
4. **Choose Database Creation Method:**
   * Select **"Standard create"**. (Easy create might hide some options you need).
5. **Choose Engine Options:**
   * **Engine type:** Select **"MySQL"**.
   * **Version:** Choose a recent stable version (e.g., MySQL 8.0.x).
6. **Choose Templates:**
   * **Crucially, select "Free tier"** under "Templates". This will pre-configure many settings to align with the Free Tier limits.
7. **DB Instance Identifier:**
   * **DB instance identifier:** Give your database instance a unique name (e.g., my-spring-app-db).
8. **Credentials:**
   * **Master username:** Keep the default admin or choose your own.
   * **Master password:** Enter a strong password and confirm it. **REMEMBER THIS!** You'll need it for your Spring Boot app.
9. **DB instance size:**
   * Since you selected "Free tier", it will automatically select a **"Burstable classes (includes t classes)"** instance type (e.g., db.t2.micro or db.t3.micro). **Do not change this** if you want to stay within Free Tier limits.
10. **Storage:**
    * **Storage type:** General Purpose (ssd)
    * **Allocated storage:** It should default to **20 GiB**. Do not increase this if you want to stay within Free Tier limits.
    * **Storage autoscaling:** You can leave this enabled, but be mindful of costs **if it exceeds 20** GiB. For a free tier, you likely want to keep it small.
11. **Connectivity:**
    * **VPC:** Select the **same VPC** where your EC2 instance is running. This is critical for internal network communication.
    * **Subnet group:** Leave as default or select a group that covers subnets your EC2 instance is in.
    * **Public access:** Select **"No"**. You want your Spring Boot application on EC2 to connect to it internally, not expose it directly to the internet.
    * **VPC security groups:**
      + Select **"Create new"** for now, or if you already have a security group for your EC2 instance (e.g., my-spring-app-sg), select that and you'll configure its inbound rules in the next step.
      + If creating a new one, give it a name (e.g., my-rds-db-sg).
12. **Database Authentication:**
    * Select **"Password authentication"**.
13. **Additional Configuration (Optional but review):**
    * **Initial database name:** You can create an initial database here (e.g., acl\_db). If not, you'll create it manually later.
    * **Backup retention period:** Free tier often defaults to 1 day.
    * **Monitoring:** Leave default.
14. **Create database:**
    * Review all settings. The estimated monthly cost should show as **"$0.00"** if you've correctly selected "Free tier" and haven't exceeded any limits.
    * Click **"Create database"**.

*The database creation process will take several minutes (10-20 minutes). Its status will change from "creating" to "available".*

**Part 2: Configure RDS Security Group for Connectivity**

Once your RDS instance status is "available", you need to allow your Spring Boot application (on EC2) to connect to it.

1. **Go to RDS dashboard** -> **Databases** -> Select your new database instance.
2. In the **"Connectivity & security"** tab, scroll down to **"Security group"**.
3. Click on the **VPC security group** associated with your RDS instance (e.g., my-rds-db-sg or the one you selected).
4. In the Security Group details, go to **"Inbound rules"** -> **"Edit inbound rules"**.
5. **Add a rule:**
   * **Type:** MySQL/Aurora (Port 3306)
   * **Source:** Select **"Custom"** and then paste the **Security Group ID** of your **EC2 instance** (e.g., sg-xxxxxxxxxxxxx). This is the **most secure way** to allow traffic from your EC2 application.
     + *Alternatively, for quick testing (less secure), you could use "Anywhere-IPv4" (0.0.0.0/0) but* ***do not use this in production****.*
   * Click **"Save rules"**.

**Part 3: Connect Your Spring Boot Application to RDS**

1. **Get RDS Endpoint:**
   * Go back to your RDS dashboard -> **Databases** -> Select your database instance.
   * In the **"Connectivity & security"** tab, find the **"Endpoint & port"** section. Copy the **Endpoint** URL (it will look something like my-spring-app-db.xxxxxxxx.ap-south-1.rds.amazonaws.com).
2. **Update Your Spring Boot application.properties:**
   * Modify your application.properties (or application.yml) to point to your new RDS MySQL database.

Properties

# --- MySQL RDS Database Configuration ---

spring.datasource.url=jdbc:mysql://<YOUR\_RDS\_ENDPOINT>:3306/<INITIAL\_DATABASE\_NAME>?useSSL=FALSE&allowPublicKeyRetrieval=TRUE

spring.datasource.username=<YOUR\_MASTER\_USERNAME> # e.g., admin

spring.datasource.password=<YOUR\_MASTER\_PASSWORD> # The password you set in RDS

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

# --- JPA / Hibernate Configuration ---

spring.jpa.hibernate.ddl-auto=update # Use 'create' for first run, then 'update' or 'none' for existing data

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect # Use MySQL8Dialect if MySQL 8.0+

# Optional: If you had H2 console enabled, disable it for production or external DBs

spring.h2.console.enabled=false

* + **Replace the placeholders:**
    - <YOUR\_RDS\_ENDPOINT>: The endpoint you copied from RDS.
    - <INITIAL\_DATABASE\_NAME>: The name of the database you created in RDS (e.g., acl\_db). If you didn't specify one, you might need to connect manually first and create one via a MySQL client.
    - <YOUR\_MASTER\_USERNAME>
    - <YOUR\_MASTER\_PASSWORD>
  + **useSSL=FALSE**: For local development, this is often set to FALSE to avoid SSL setup headaches, but for production, you should configure SSL (useSSL=TRUE and proper certificates).
  + **allowPublicKeyRetrieval=TRUE**: Sometimes needed for MySQL 8.0+ driver.

1. **Add MySQL JDBC Driver Dependency:**
   * In your pom.xml (if using Maven):

XML

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<scope>runtime</scope> </dependency>

* + Or build.gradle (if using Gradle):

Gradle

dependencies {

runtimeOnly 'mysql:mysql-connector-java'

}

Make sure you mvn clean install or gradle build after adding the dependency.

1. **Redeploy Your Spring Boot Application to EC2:**
   * Build a new JAR with the updated application.properties.
   * Copy the new JAR to your EC2 instance (e.g., using scp).
   * Stop your old application process and start the new one (e.g., using nohup java -jar your-app.jar > app.log 2>&1 &).

Your Spring Boot application should now connect to the MySQL database hosted on AWS RDS.

**Important Free Tier Limits & Considerations:**

* **db.t2.micro / db.t3.micro:** You get 750 hours per month across *all* Free Tier eligible instances. If you run multiple t2.micro databases, hours combine.
* **20 GB General Purpose (SSD) Storage:** You get 20 GiB free.
* **20 GB Backup Storage:** For database backups.
* **10 Million I/Os:** Input/Output operations.
* **Exceeding Limits:** If you go over these limits, you *will* be charged at standard rates.
* **Free Tier Expiration:** The free tier usually lasts for 12 months from your AWS account creation. After that, you'll be charged standard rates for all services.
* **Not for Production:** The Free Tier is for learning and development. It does not offer high availability, strong performance guarantees, or the robustness required for production workloads.