AWS RDS Production Setup Guide

This guide details the steps to set up a production-ready PostgreSQL database on AWS RDS and connect it to your Elastic Beanstalk (EB) environment.

1. Create AWS RDS Instance

1. **Go to AWS Console** -> **RDS** -> **Create database**.
2. **Choose a database creation method**: Standard create.
3. **Engine options**: PostgreSQL.
4. **Version**: Select **PostgreSQL 14** or higher (match your local development version if possible).
5. **Templates**: Select **Production** (for high availability) or **Free tier** (for testing/low cost).
6. **Settings**:
    - **DB instance identifier**: `driversklub-prod-db`
    - **Master username**: `postgres` (or your choice)
    - **Master password**: *Generate a strong password and save it securely.*
7. **Instance configuration**:
    - **DB instance class**: `db.t3.micro` (Free tier) or `db.t3.small/medium` (Production).
8. **Storage**:
    - **Storage type**: gp3 (General Purpose SSD).
    - **Allocated storage**: 20 GiB (start small, autoscaling covers growth).
9. **Connectivity**:
    - **Compute resource**: Don't connect to an EC2 compute resource.
    - **VPC**: Select the **Default VPC** (same as your Elastic Beanstalk).
    - **Public access**: **No** (Recommended for security). *If `Yes`, restrict strictly by IP.*
    - **VPC Security Group**: Create new named `rds-postgres-sg`.
10. **Database authentication**: Password authentication.
11. **Create database**.

---

2. Configure Network Security

Allow your Elastic Beanstalk instances to talk to the RDS database.

1. **Go to EC2 Dashboard** -> **Security Groups**.
2. Find the **Security Group used by your Elastic Beanstalk environment** (usually named like `awseb-e-...-stack-AWSEBSecurityGroup...`). Copy its **Security Group ID** (starts with `sg-`).
3. Find the **RDS Security Group** you created (`rds-postgres-sg`).
4. **Edit Inbound rules** for the *RDS Security Group*:
    - **Type**: PostgreSQL (TCP/5432).
    - **Source**: Paste the **Elastic Beanstalk Security Group ID**.
    - *This allow traffic ONLY from your backend servers.*

---

3. Configure Environment Variables in Elastic Beanstalk

1. **Go to Elastic Beanstalk Console** -> Your Environment (`driversklub-backend-env`).
2. **Configuration** -> **Updates, monitoring, and logging** -> **Software** -> **Edit**.

3. Scroll to **Environment properties**.
4. Add the `DATABASE_URL` property:

   Format:

   ```
   postgresql://<username>:<password>@<endpoint>:5432/postgres?
   schema=public&connection_limit=10
   ```

   Example:

   ```
   postgresql://postgres:MyStrongPass123@driversklub-prod-db.cxyz.ap-south-
   1.rds.amazonaws.com:5432/postgres?schema=public&connection_limit=10
   ```

   - `<endpoint>` : Found in RDS Console -> Connectivity & security -> Endpoint.
   - `<username>` / `<password>` : Defined in Step 1.
5. **Apply** changes. The environment will update (rolling restart).

---

4. Production Migrations

To ensure your production database schema matches your code, you need to run migrations. The best way in Elastic Beanstalk is using `.ebextensions`.

Method A: Automated Migrations (Recommended)

1. Create a folder `.ebextensions` in your project root if it doesn't exist.
2. Create a file `.ebextensions/01_migration.config` :

```
container_commands:
  01_migrate:
    command: "npx prisma migrate deploy --schema packages/database/prisma/schema.prisma"
    leader_only: true
```

- `leader_only: true` : Ensures the migration runs only on one instance during deployment, preventing race conditions.
- **Deploy**: Commit and push this file. The next deployment will automatically apply pending migrations.

Method B: Manual Migrations (Control Freak)

If you prefer to migrate manually:

1. Install the **AWS CLI** and configure it.
2. Use **EB CLI** to SSH into an instance:

```
eb ssh driversklub-backend-env
```

3. Navigate to the app directory (usually `/var/app/current`).
4. Run the migration:

```
export DATABASE_URL=... # (If not already set in shell)
npx prisma migrate deploy --schema packages/database/prisma/schema.prisma
```

---

5. Verification

1. Deploy your application.
2. Check the **EB Health**.
3. Check **Logs** (`eb logs`) to verify the database connection was successful and migrations ran (search for `01_migrate`).
4. Use the App/Dashboard to verify data can be read/written.