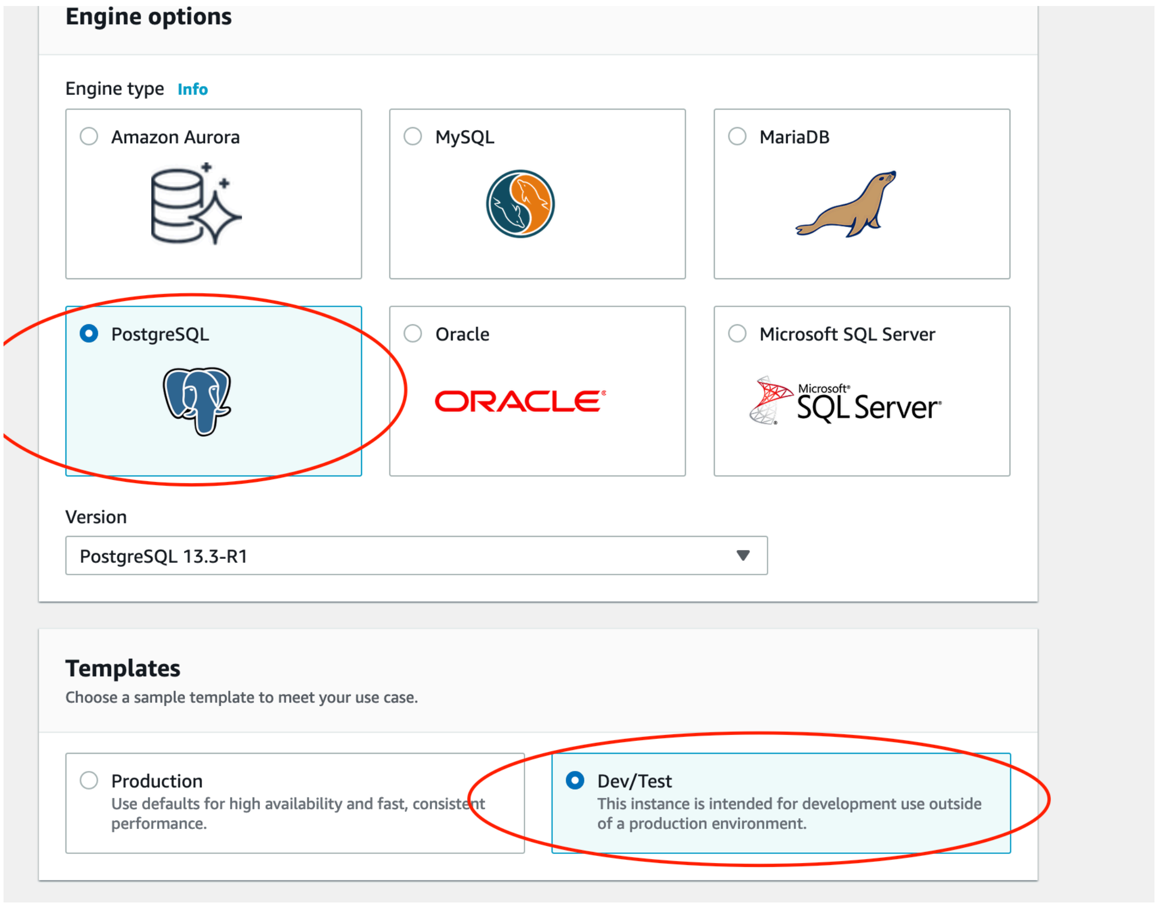
Project work (group) - 3TUs - RDS

1. Create RDS instance in AWS console. Select PostgreSQL as a type, and dev purposes. Remember to make it Publicity accessible.



Engine options

Engine type Info

- ○ Amazon Aurora
- ○ MySQL
- ○ MariaDB
- ● PostgreSQL
- ○ Oracle
- ○ Microsoft SQL Server

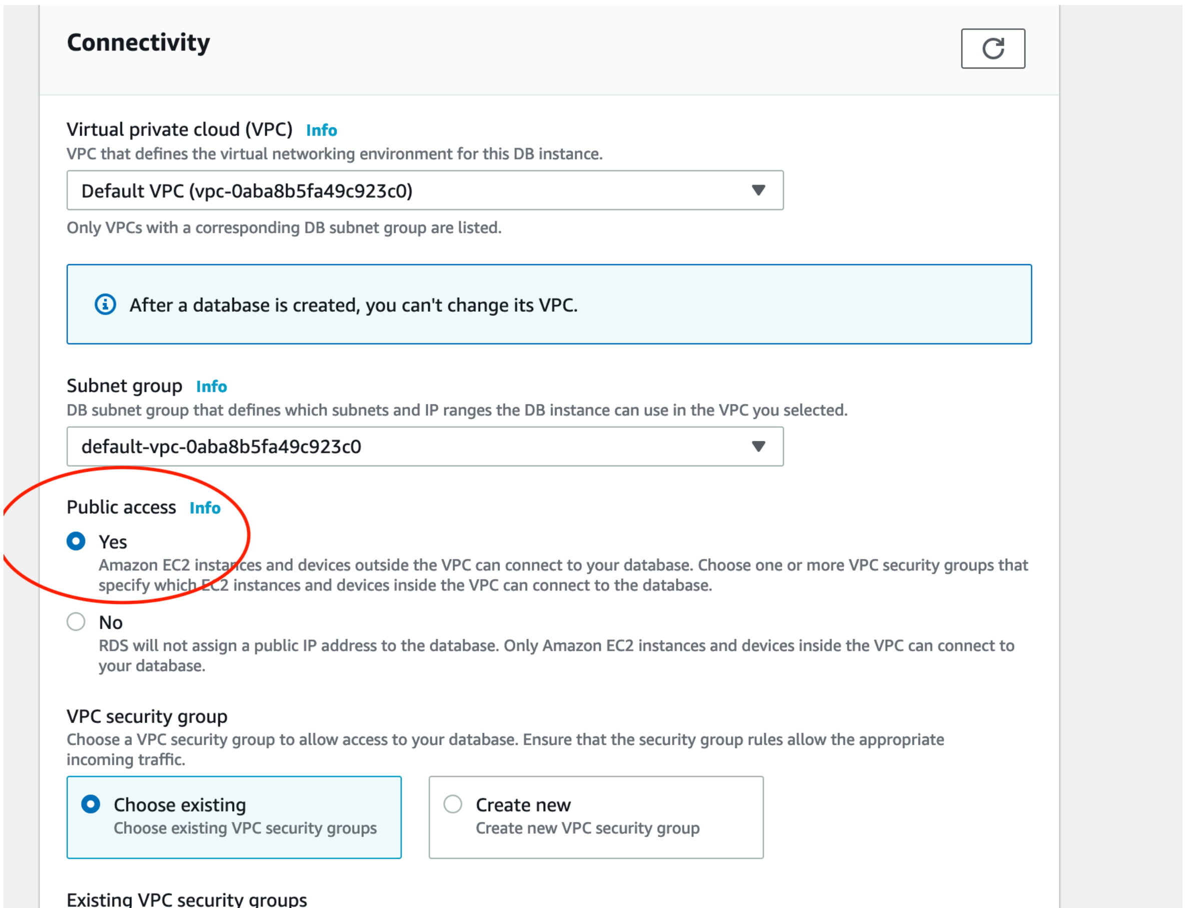
Version

PostgreSQL 13.3-R1

Templates

Choose a sample template to meet your use case.

- ○ Production
  Use defaults for high availability and fast, consistent performance.
- ● Dev/Test
  This instance is intended for development use outside of a production environment.



Connectivity

Virtual private cloud (VPC) Info

VPC that defines the virtual networking environment for this DB instance.

Default VPC (vpc-0aba8b5fa49c923c0)

Only VPCs with a corresponding DB subnet group are listed.

After a database is created, you can't change its VPC.

Subnet group Info

DB subnet group that defines which subnets and IP ranges the DB instance can use in the VPC you selected.

default-vpc-0aba8b5fa49c923c0

Public access Info

- ● Yes
  Amazon EC2 instances and devices outside the VPC can connect to your database. Choose one or more VPC security groups that specify which EC2 instances and devices inside the VPC can connect to the database.
- ○ No
  RDS will not assign a public IP address to the database. Only Amazon EC2 instances and devices inside the VPC can connect to your database.

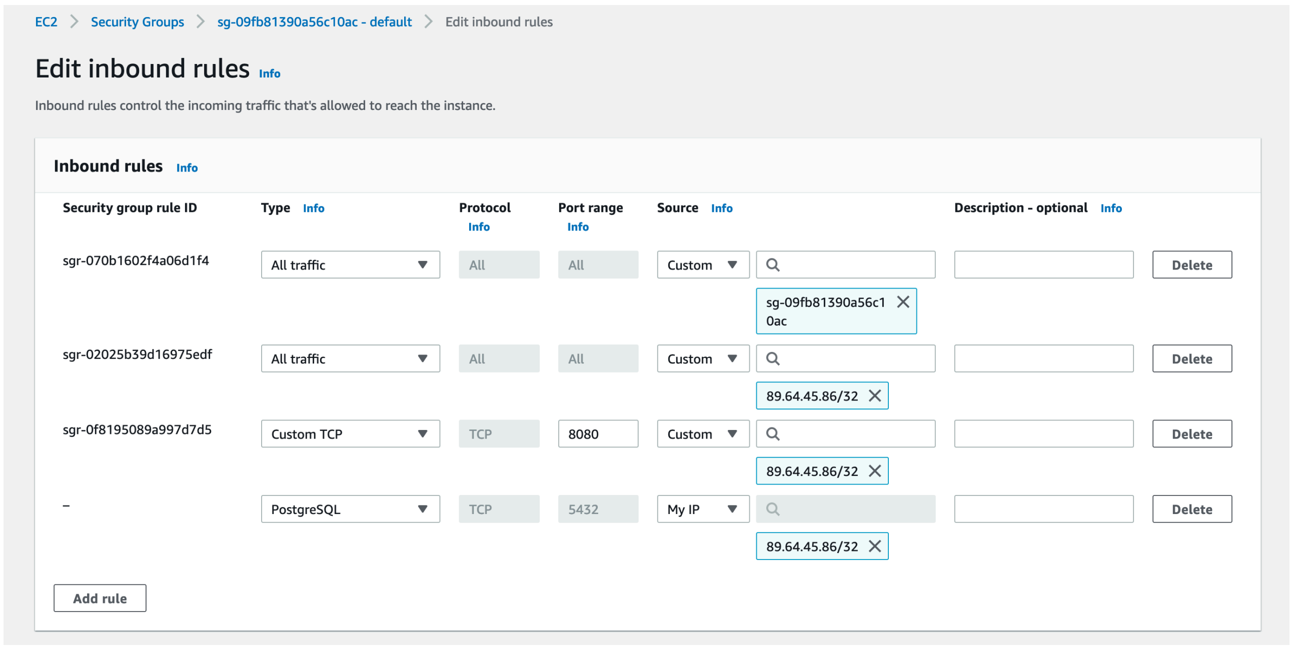
VPC security group

Choose a VPC security group to allow access to your database. Ensure that the security group rules allow the appropriate incoming traffic.

- ● Choose existing
  Choose existing VPC security groups
- ○ Create new
  Create new VPC security group

Existing VPC security groups

2. Modify your default Security Group to allow Inbound traffic of type PostgreSQL from IP of your computer.



EC2 > Security Groups > sg-09fb81390a56c10ac - default > Edit inbound rules

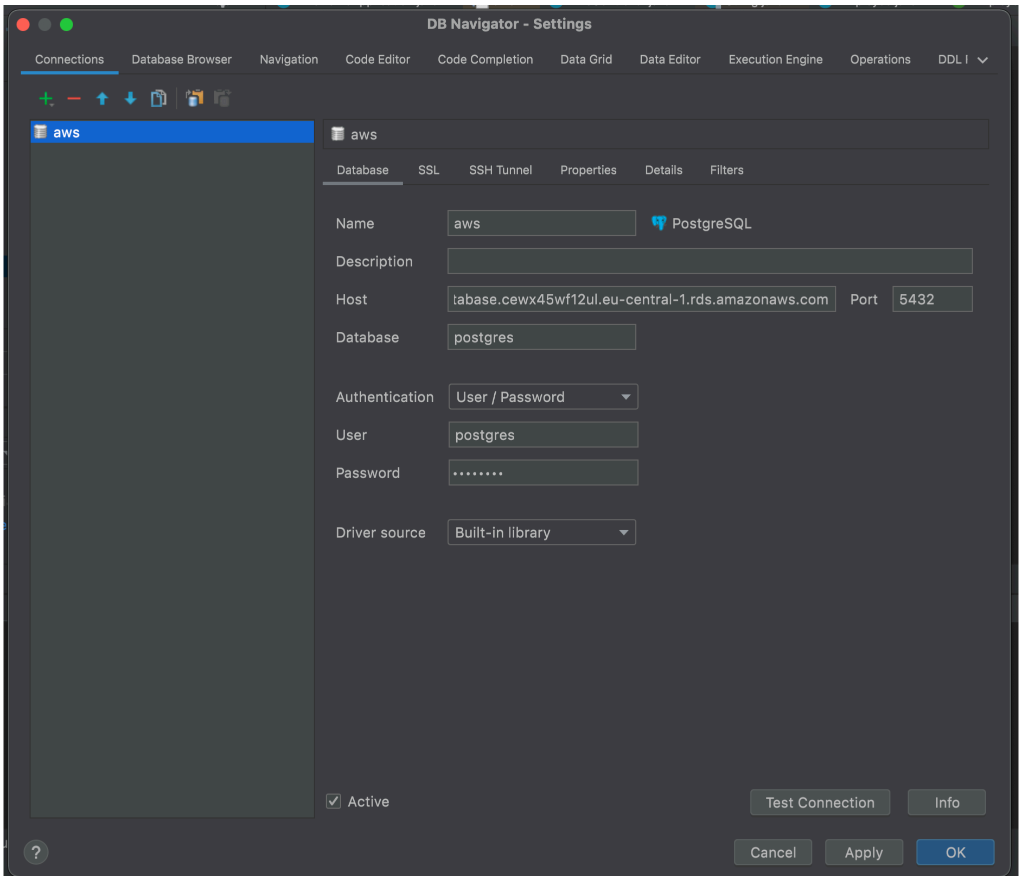
Edit inbound rules Info

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules Info

Security group rule ID	Type	Protocol	Port range	Source		Description - optional
sgr-070b1602f4a06d1f4	All traffic	All	All	Custom	sg-09fb81390a56c10ac	
sgr-02025b39d16975edf	All traffic	All	All	Custom	89.64.45.86/32	
sgr-0f8195089a997d7d5	Custom TCP	TCP	8080	Custom	89.64.45.86/32	
–	PostgreSQL	TCP	5432	My IP	89.64.45.86/32	

2. Test instance connection via IntelliJ (DB Navigator) or another database client (e.g. DBeaver). Use the same username/password that you set in RDS creator.



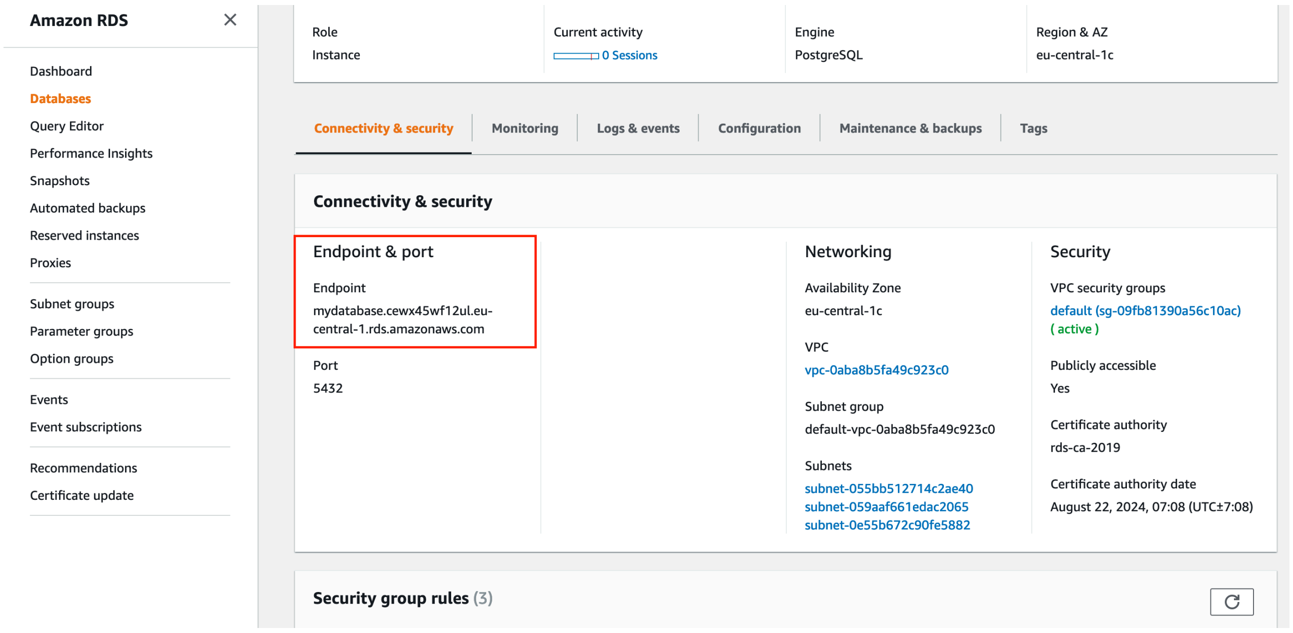
DB Navigator - Settings

Connections: aws

Name	aws (PostgreSQL)
Description	
Host	tabase.cewx45wf12ul.eu-central-1.rds.amazonaws.com
Port	5432
Database	postgres
Authentication	User / Password
User	postgres
Password	••••••••
Driver source	Built-in library

✓ Active

You can find host as an endpoint under Connectivity & Security section of your RDS instance:



Role: Instance
Current activity: 0 Sessions
Engine: PostgreSQL
Region & AZ: eu-central-1c

Connectivity & security

Endpoint & port
Endpoint: mydatabase.cewx45wf12ul.eu-central-1.rds.amazonaws.com
Port: 5432

Networking
Availability Zone: eu-central-1c
VPC: vpc-0aba8b5fa49c923c0
Subnet group: default-vpc-0aba8b5fa49c923c0
Subnets: subnet-055bb512714c2ae40, subnet-059aaf661edac2065, subnet-0e55b672c90fe5882

Security
VPC security groups: default (sg-09fb81390a56c10ac) (active)
Publicly accessible: Yes
Certificate authority: rds-ca-2019
Certificate authority date: August 22, 2024, 07:08 (UTC±7:08)

4. Modify your last project:
- Add dependencies: **Spring Boot Data JPA and Postgresql**

- Add in **application.properties** database related props:

spring.datasource.url=jdbc:postgresql://endpoint/postgres
spring.datasource.username=your_username
spring.datasource.password=your_password
spring.jpa.hibernate.ddl-auto=create-drop

your_username and your_password are the ones that you set when creating RDS instance endpoint - you can find it under Connectivity & Security section of your RDS instance (step above).

- Create entity: **Employee** with fields: **id** (long, use proper adnotation with strategy), **name** (string), **role** (string)
- Create **EmployeeRepository** by extending Spring **JpaRepository**
- Create rest endpoint and expose 2 methods: to **add a new employee** and to **list existing employees**

You can find example of a project in src folder.

5. Run your application locally and test exposed endpoints, use Postman to test POST method that creates a new employee
6. Upload your modified jar to EC2 using steps from the last projects
7. Test your new endpoints on EC2