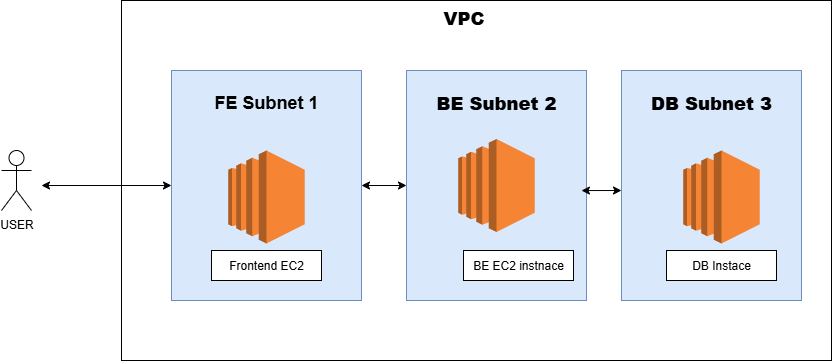
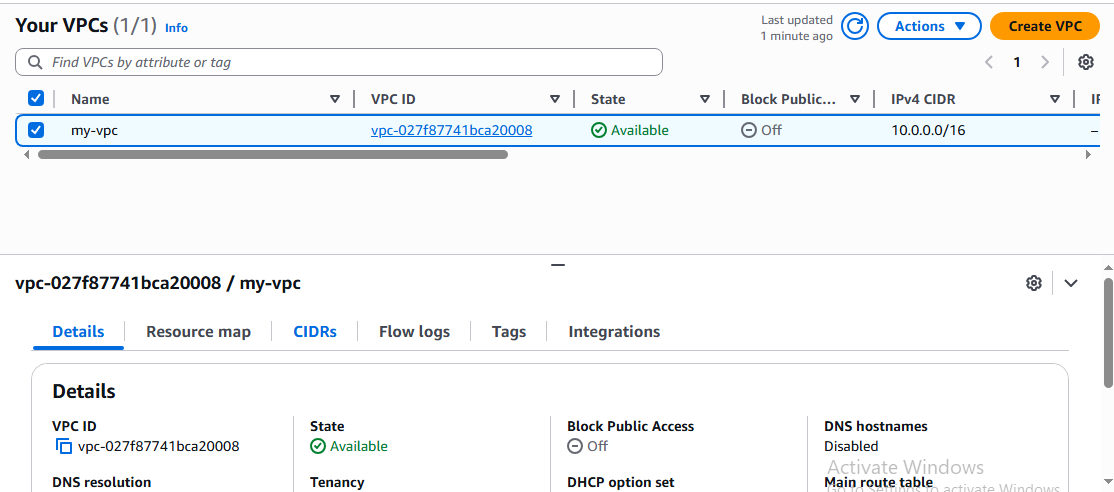
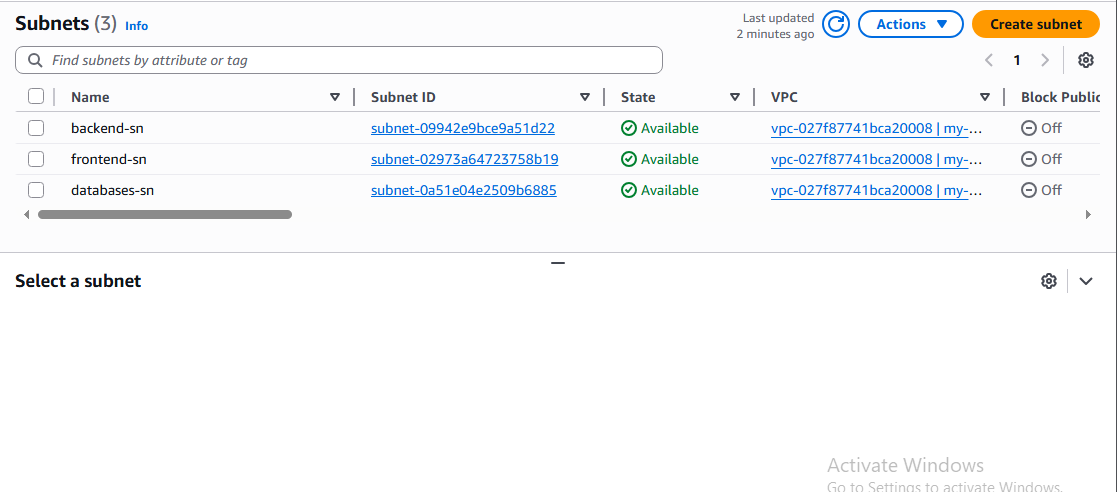
**3-tier deployment setup in AWS**



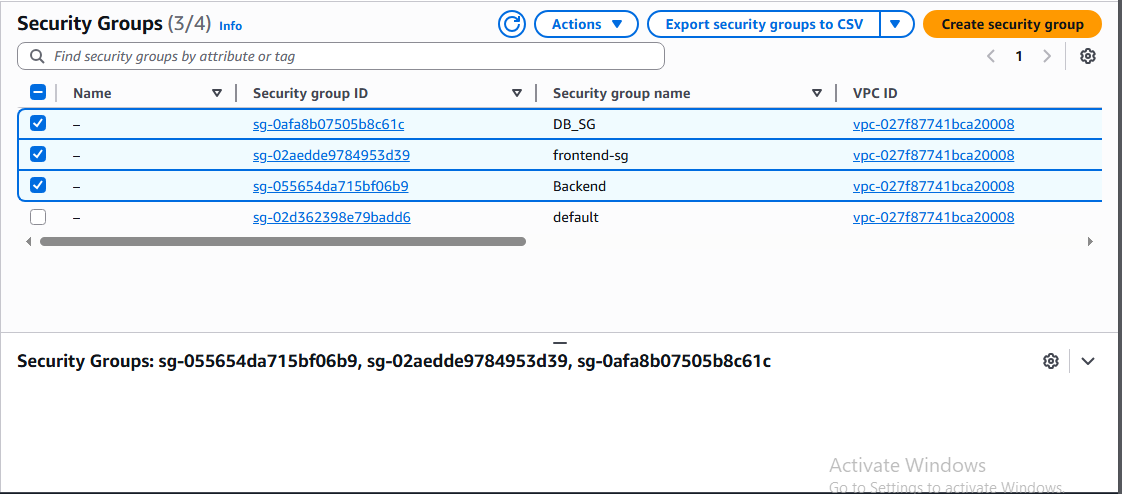
1. **Create the VPC in Any preferred region**



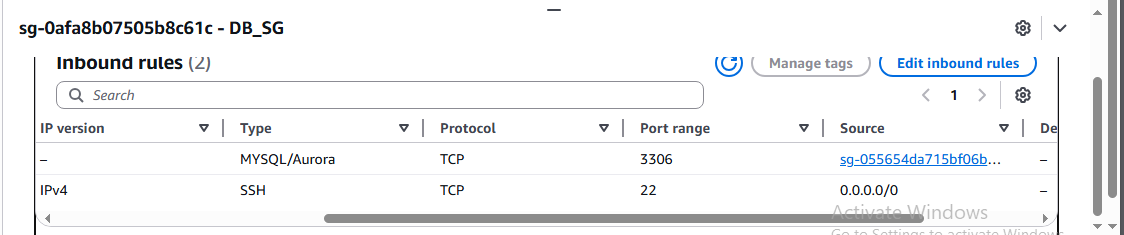
1. **Create 3 Subnets with Proper Naming for each as shown below**



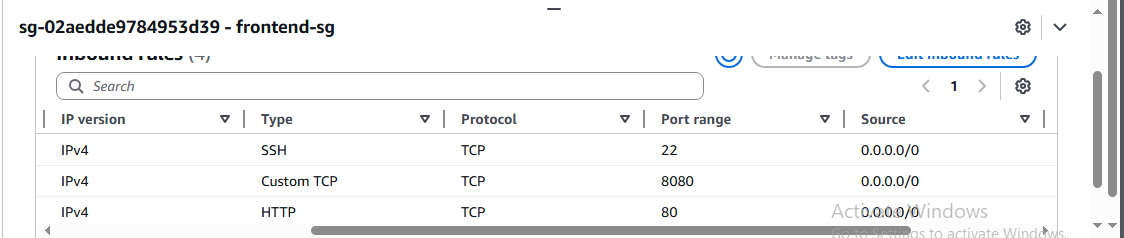
1. **Create SG groups for FE, BE and DB with their respective naming as shown below**



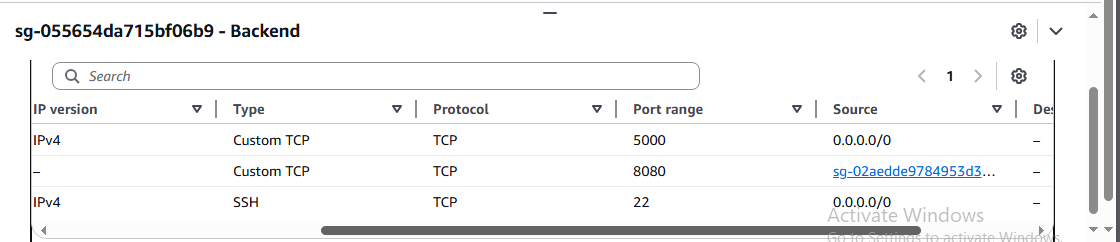
**And The Ports for DB are 3306 from only Backend SG and 22 from 0.0.0.0**

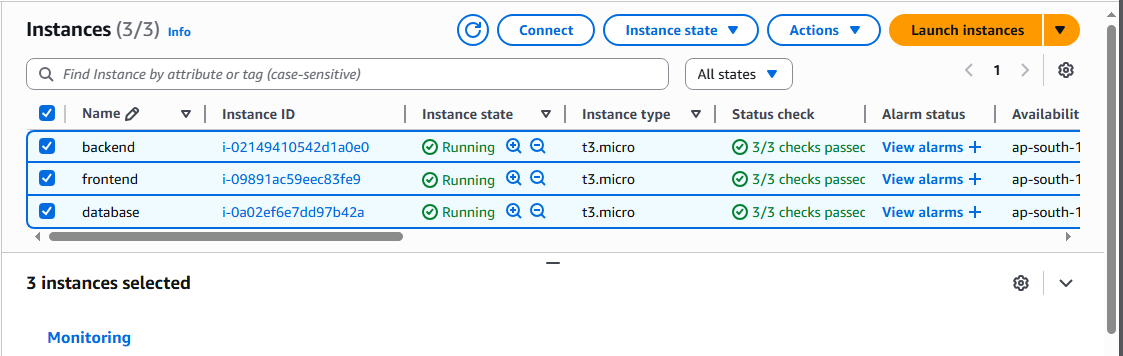


**And The Ports for Frontend are 22, 8080, 80 from 0.0.0.0**



**And The Ports for Backend are 5000, 22 from 0.0.0.0 and Optional 8080 from Frontend**





Perfect — this is a classic **3-tier deployment setup** in AWS:

✅ **Frontend Instance (FE)** – e.g., React/Angular/Next.js  
✅ **Backend Instance (BE)** – e.g., Node.js, Spring Boot, or Grails  
✅ **Database Instance (DB)** – MySQL

Let’s go step-by-step so you can build this **clean and secure AWS setup** 👇

**🧱 1. Architecture Overview**

User → AWS EC2 (Frontend) → AWS EC2 (Backend) → AWS EC2 (MySQL)

* Frontend talks to Backend (via API calls)
* Backend talks to MySQL (via DB connection)
* Public access only to Frontend

**⚙️ 2. Create the EC2 Instances**

Create **3 Ubuntu EC2 instances**:

| **Instance** | **Purpose** | **Example Name** | **Type** | **Network Access** |
| --- | --- | --- | --- | --- |
| FE | Host your frontend | frontend-instance | t2.micro | Public |
| BE | Host your backend APIs | backend-instance | t2.micro | Private |
| DB | Host MySQL | mysql-instance | t2.micro | Private |

🧩 Place them in the **same VPC** and preferably **same subnet (or connected private subnets)**.

**🌐 3. Configure Security Groups**

**Frontend SG (SG-FE)**

* **Inbound**:
  + HTTP (80) or HTTPS (443) — from anywhere (0.0.0.0/0)
  + SSH (22) — from your IP only
* **Outbound**: Allow all
* **Talks to Backend:** Yes (allow FE → BE port)

**Backend SG (SG-BE)**

* **Inbound**:
  + App port (e.g., 8080) — only from **SG-FE**
  + SSH (22) — from 0.0.0.0 or from your IP only (optional)
  + 5000 from anywhere 0.0.0.0
* **Outbound**:
  + Allow traffic to DB (MySQL port 3306)
* **Talks to Database:** Yes (BE → DB)

**MySQL SG (SG-DB)**

* **Inbound**:
  + MySQL (3306) — only from **SG-BE**
  + **SSH 22 from Myip (optional)**
* **Outbound**: Allow all
* No public access!

**🗄️ 4. Setup MySQL DB Instance**

1. SSH into the DB instance

sudo apt update

sudo apt install mysql-server -y

1. Edit config to listen on private IP

sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf

Change:

bind-address = 0.0.0.0

1. Restart MySQL

sudo systemctl restart mysql

1. Create DB and user:
2. sudo mysql -u root

**CREATE DATABASE appdb;**

**CREATE USER 'appuser'@'%' IDENTIFIED BY 'StrongPassword123';**

**GRANT ALL PRIVILEGES ON appdb.\* TO 'appuser'@'%';**

**FLUSH PRIVILEGES;**

**🖥️ 5. Setup Backend Instance**

1. SSH into backend instance
2. Install Node.js / Java / Grails / etc.
3. Clone your code or deploy it from GitHub / Docker
4. In .env or config file, point DB connection to private IP of DB instance:

DB\_HOST=10.0.1.25 # Example private IP of DB EC2

DB\_USER=appuser

DB\_PASS=StrongPassword123

1. Run backend:

npm start # or ./gradlew run or catalina.sh run

1. Ensure backend is listening on 0.0.0.0:8080.

**🖼️ 6. Setup Frontend Instance**

1. SSH into frontend instance
2. Install Nginx (to serve your frontend build)

sudo apt update

sudo apt install nginx -y

1. Build your frontend (React/Angular etc.)

npm run build

1. Copy the build output to /var/www/html

sudo cp -r build/\* /var/www/html/

1. Edit API URLs in frontend (example):

const API\_BASE\_URL = "http://<backend-private-ip>:8080";

✅ **Better:** Create a public DNS name for the backend through a Load Balancer later.

1. Restart Nginx:

sudo systemctl restart nginx

**🔗 7. Test End-to-End**

* From your laptop browser →  
  http://<frontend-public-ip>
* The frontend → calls backend (on backend’s private IP)
* Backend → connects to MySQL (on DB’s private IP)

Linux Part

1. **MYSQL DB INSTANCE (Private Subnet)**

sudo apt update -y && sudo apt upgrade -y

sudo apt install -y mysql-server

sudo mysql\_secure\_installation

# Log in to MySQL

sudo mysql -u root –p

Allow remote connections (from backend only):

sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf

**Change this line:**

bind-address = 127.0.0.1

to:

bind-address = 0.0.0.0

**Restart MySQL:**

sudo systemctl restart mysql

sudo systemctl enable mysql

**Create DB and user:**

sudo mysql -u root

CREATE DATABASE appdb;

CREATE USER 'appuser'@'%' IDENTIFIED BY 'StrongPassword123';

GRANT ALL PRIVILEGES ON appdb.\* TO 'appuser'@'%';

FLUSH PRIVILEGES;

1. **BACKEND INSTANCE (Public Subnet)**

sudo apt update

sudo apt install -y curl

curl -fsSL https://deb.nodesource.com/setup\_20.x | sudo -E bash -

sudo apt install -y nodejs

node -v

npm -v

git clone https://github.com/jaiswamy/1.Nodejs.git

npm start

NOTE: Check or Create **.env** file

root@ip-10-0-1-28:~# cat .env

PORT=5000

DB\_NAME= appdb

DB\_HOST= <db private ip>

DB\_USER= appuser

DB\_PASS= StrongPassword123

1. **FRONTEND INSTANCE (Public Subnet)**

apt-get update

sudo apt install nginx -y

npm run build

git clone <https://github.com/jaiswamy/1.Nodejs.git>

**(You have to add the index.html and style.css into /var/www/html/ )**

systemctl restart nginx

NOTE: In the index.html file make sure you add the <public ip:port>, shown below

**2. In the index.html make sure to add the**

<script>

const API\_URL = 'http://<backend\_pub\_ip>:5000/api/items';

…

…

</script>