Yash Mutatkar --Team Alpha --Task 3

Step-by-Step Procedure: Create an EC2 Instance and Launch Medusa

This document outlines the step-by-step process to create an Amazon EC2 instance and launch the Medusa e-commerce backend on the instance.

Step 1: Log in to AWS Console

1. Navigate to the AWS Management Console (https://aws.amazon.com/console/).

2. Log in with your AWS account credentials.

Step 2: Create an EC2 Instance

1. In the AWS Management Console, go to the EC2 Dashboard by searching for 'EC2'.

2. Click 'Launch Instance' and provide a name for your instance (e.g., 'Medusa-Server').

3. Choose the Amazon Machine Image (AMI):
- Select 'Ubuntu Server 22.04 LTS (HVM), SSD Volume Type'.

4. Choose the instance type:
- Select 't2.micro' (free tier eligible).

5. Configure the Key Pair:
- Create a new key pair or select an existing one for SSH access.

6. Configure the security group:
- Allow HTTP (port 80), HTTPS (port 443), and custom TCP rule for port 9000 (for Medusa) and 7001 (admin panel).

7. Launch the instance.

Step 3: Connect to the EC2 Instance

1. Once the instance is running, select it and click 'Connect'.

2. Use the SSH command provided by AWS to connect to your instance:
```bash
```

```
ssh -i /path/to/key.pem ubuntu@<your-ec2-public-ip>
```

Step 4: Install Required Software

1. Update the package list and install Node.js 18, npm, and Git:
```bash
sudo apt update
sudo apt install nodejs npm git -y
```

2. Verify Node.js and npm installation:
```bash
node -v
npm -v
```

Step 5: Install PostgreSQL

1. Install PostgreSQL by running the following command:
```bash
sudo apt install postgresql postgresql-contrib -y
```

2. Start and enable PostgreSQL service:
```bash
sudo systemctl start postgresql
sudo systemctl enable postgresql
```

3. Set up PostgreSQL user and database for Medusa:
```bash
sudo -i -u postgres
psql
CREATE USER medusa_user WITH PASSWORD 'root1234';
CREATE DATABASE medusa_db;
GRANT ALL PRIVILEGES ON DATABASE medusa_db TO medusa_user;
\q
exit
```

Step 6: Set Up Medusa Project

1. Clone the Medusa repository:
```bash
git clone https://github.com/medusajs/medusa-starter-default.git medusa-server
cd medusa-server
```

2. Install dependencies:
```bash
npm install
```

Step 7: Configure Medusa for PostgreSQL

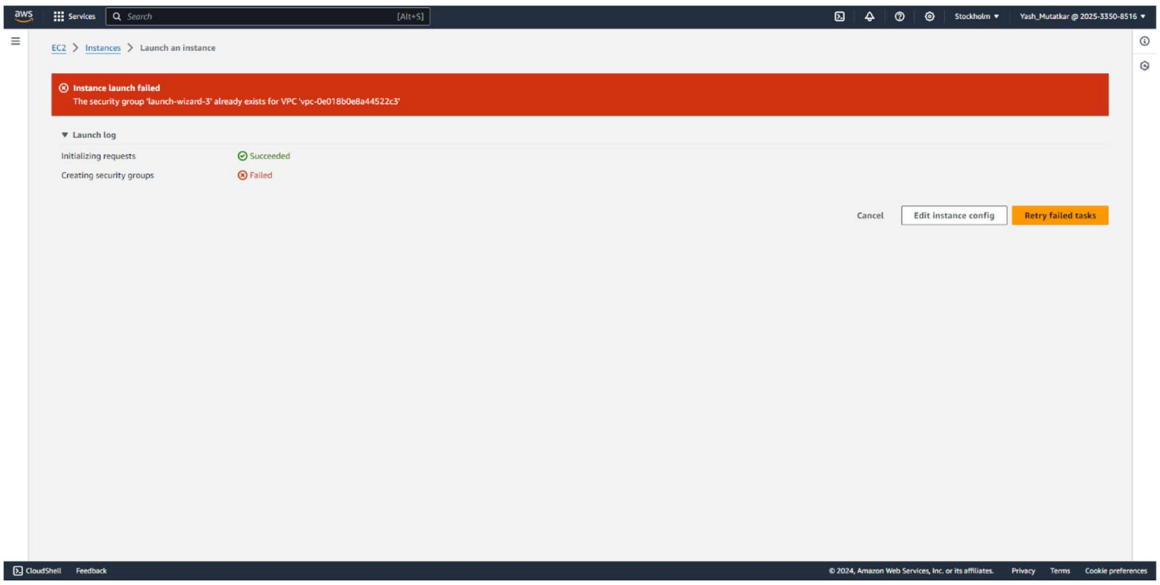
1. Update the `.env` file with the PostgreSQL connection details:
```bash
DATABASE_URL=postgres://medusa_user:root1234@localhost:5432/medusa_db
JWT_SECRET=something
COOKIE_SECRET=somethingelse
ADMIN_CORS=http://localhost:7001
STORE_CORS=http://localhost:9000
```

Step 8: Start Medusa

1. Run the following command to start the Medusa backend:
```bash
npm run start:custom
```

2. You should now be able to access Medusa via the EC2 public IP at port 9000:
- Medusa Storefront: http://<your-ec2-public-ip>:9000
- Admin Panel: http://<your-ec2-public-ip>:7001

Step 9: Errors encountered and their solution Start Medusa



Steps to resolve the issue:

1. **Check Existing Security Groups**:
    - Go to the **EC2 Dashboard** in the AWS Console.
    - On the left-hand side menu, under **Network & Security**, click on **Security Groups**.
    - Look for a security group with the name launch-wizard-3 in the same VPC (vpc-0e018b0e8a44522c3).
2. **Options to Resolve**:
    - **Use the existing security group**: If the security group launch-wizard-3 already exists and has the required rules, you can select this existing security group during the instance creation process instead of trying to create a new one.
    - **Delete or Rename the existing security group**: If you don't need the existing launch-wizard-3 security group, you can delete it or rename it, then try launching the instance again.
        - To delete it: select the security group and click **Actions → Delete security group**.

        - To rename it, you will have to create a new security group with different naming conventions.

3. **Retry Launching the Instance**:
    - After making the necessary changes, retry the instance launch by either using the existing group or creating a new security group.
    - Used this to solve!!

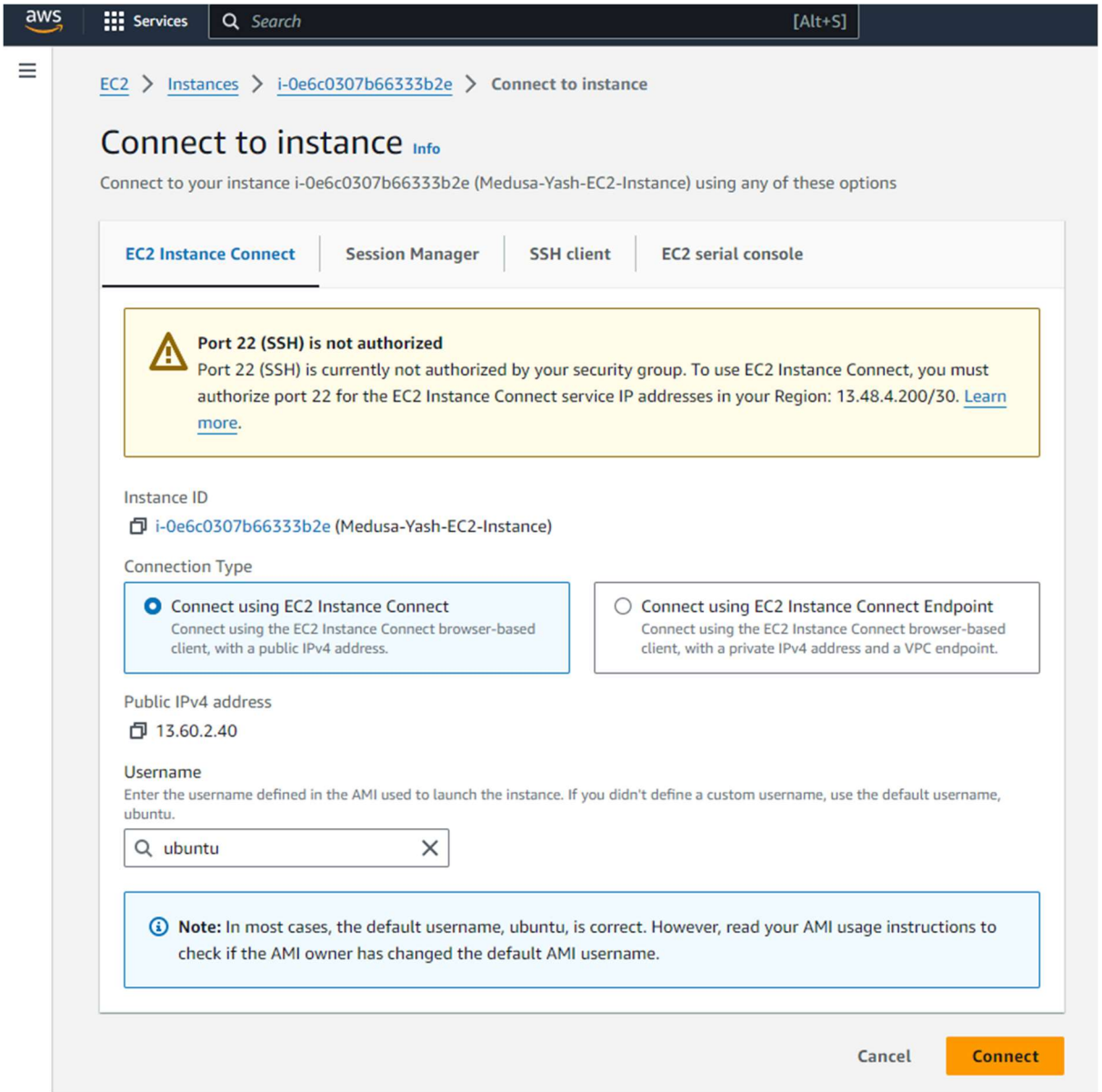
Best Approach:

- **Create a new security group with a different name**: You can create a new security group under your name or a relevant name that fits your company's naming conventions. This way, you avoid interfering with any existing configurations that might rely on the current security group.

Here's how you can create a new security group:

1. **Go to the EC2 Dashboard**.
2. On the left-hand side menu, click **Security Groups** under **Network & Security**.
3. Click **Create Security Group**.
4. Provide a new name, such as launch-wizard-yourname, and configure the required inbound and outbound rules.
5. Select the appropriate **VPC**.
6. After creating it, select this new security group when launching your instance.

This approach ensures you won't disrupt any ongoing processes or configurations that depend on the existing security group.



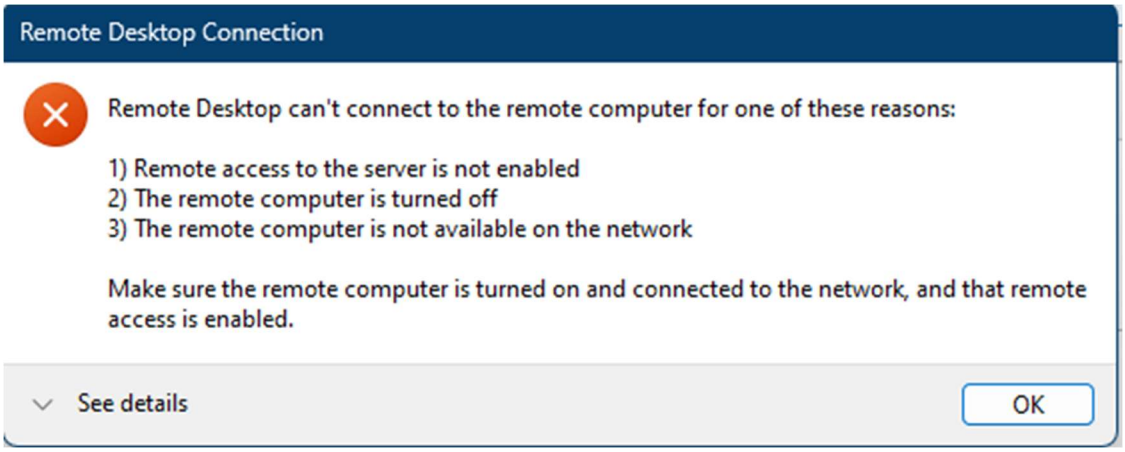
To fix this:

1. **Edit Security Group to Allow Port 22**:
    - Go to the **EC2 Dashboard**.
    - On the left-hand side, click on **Security Groups** under **Network & Security**.
    - Find and select the security group associated with your EC2 instance.
    - Click **Edit inbound rules**.
    - Add a new inbound rule for **SSH (port 22)**:
        - **Type**: SSH

        - **Protocol**: TCP
        - **Port Range**: 22
        - **Source**: Choose either:
            - My IP to allow SSH only from your current IP address.
            - Anywhere (0.0.0.0/0) to allow SSH from any IP address (not recommended unless necessary).
    - Save the changes.
2. **Retry Connection**:
    - Once the rule is added, go back to the EC2 instance connect page and retry connecting via EC2 Instance Connect.

By authorizing port 22 in the security group, you'll be able to establish an SSH connection to your instance.

Solved using user as ubuntu



Steps to Resolve:

1. Check RDP (Remote Desktop Protocol) Port (3389) in Security Group:

- Ensure that **port 3389** (RDP) is open in your EC2 instance's security group.
- Go to **EC2 Dashboard** → **Security Groups** → Select your security group → **Edit inbound rules**.
- Add a new rule:
  - **Type**: RDP
  - **Protocol**: TCP
  - **Port Range**: 3389
  - **Source**: Choose My IP for secure access from your machine, or 0.0.0.0/0 for open access (be cautious with this).
- Save the changes and try connecting again.

2. Enable Remote Access on the Instance:

- If it's a Windows instance, make sure that **Remote Desktop** is enabled inside the instance.
  - Login via **EC2 Instance Connect** or **SSH** if possible and go to:
    - **Control Panel** → **System** → **Remote settings**.
    - Under **Remote Desktop**, ensure the option to allow remote connections is enabled.

3. Ensure the Instance is Running:

- Check the **EC2 Dashboard** to make sure your instance is in a **Running** state.

4. Verify the Public IP Address:

- Make sure you're connecting to the correct **public IP address** of the instance.
- You can find the public IP under **EC2 Instance Details** in the AWS console.

5. Check for Network Issues:

- Make sure the instance is not facing network issues by checking the **Network ACL** rules.
- Ensure that there are no outbound rules blocking traffic on port 3389.

6. Reboot the Instance:

- If you're still unable to connect, try **rebooting the instance** from the AWS console and then attempt to reconnect.

It seems like you're trying to connect to an **Ubuntu EC2 instance** using **Remote Desktop (RDP)**, but RDP is not available for Linux-based instances like Ubuntu.

Ubuntu EC2 instances typically use SSH for remote access, not RDP. Here's how you can connect:

1. Connect via SSH:

- You are already connected to the instance using SSH (visible in the terminal in your screenshot). This is the correct method for managing Ubuntu servers.
- You can perform all administrative tasks via the SSH connection.

2. If you need a graphical interface on Ubuntu (like RDP):

If you need a graphical desktop environment (RDP-like) on your Ubuntu instance, you can install a desktop environment and an RDP server (like **xrdp**) to enable remote desktop access.

Steps to enable RDP on Ubuntu (Optional):

1. **Install XFCE (Desktop environment)**:

bash

Copy code

sudo apt update

sudo apt install xfce4 xfce4-goodies

2. **Install xrdp (RDP server)**:

bash

Copy code

sudo apt install xrdp

3. **Start and enable the xrdp service**:

bash

Copy code

sudo systemctl enable xrdp

sudo systemctl start xrdp

4. **Allow RDP through the firewall (if applicable)**:

bash

Copy code

sudo ufw allow 3389/tcp

5. **Connect via RDP**:
    - After this, you can use your **Remote Desktop Connection** app (from Windows) to connect to your Ubuntu instance using its public IP address.

However, if SSH access is sufficient, I recommend continuing to manage the instance through your existing SSH connection.