

**WIPRO CLOUD PRODUCT AND PLATFORM ENGINEERING – 2025**

**Mini Project Report**

**Name of the Student : Rishabh Kumar Singh B**

**Department : Computer Science And Engineering Year / Sem / Section : IV/VII/A**

**Date of Submission : 24-10-2025**

**Scenario - #1**

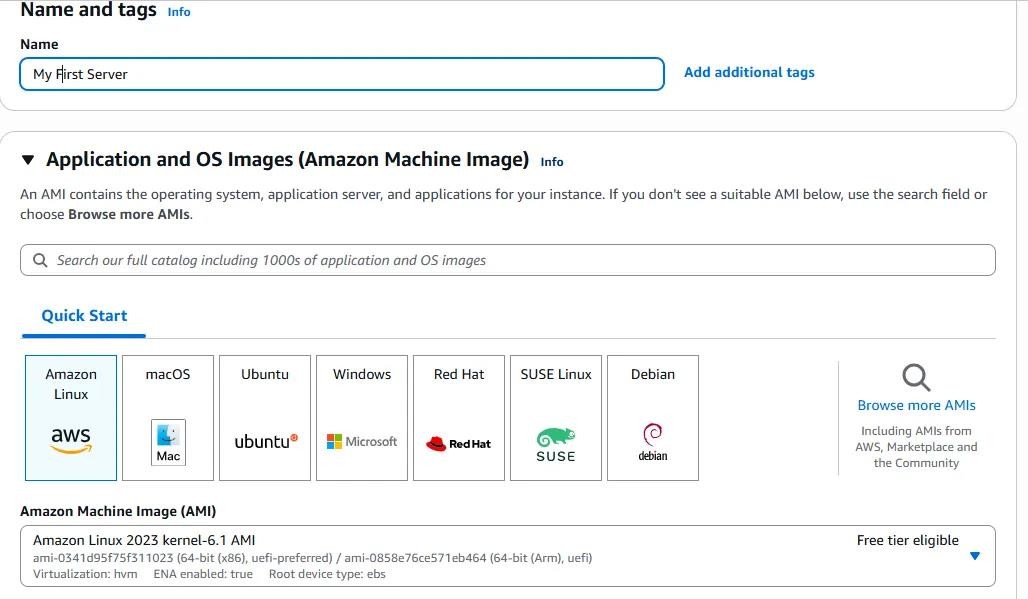
**Title: Host an E-Commerce website on a scalable cloud platform with AI-driven load balancing and data Encryption**

**Steps for Cloud Engineering:**

**Create a virtual machine in AWS**

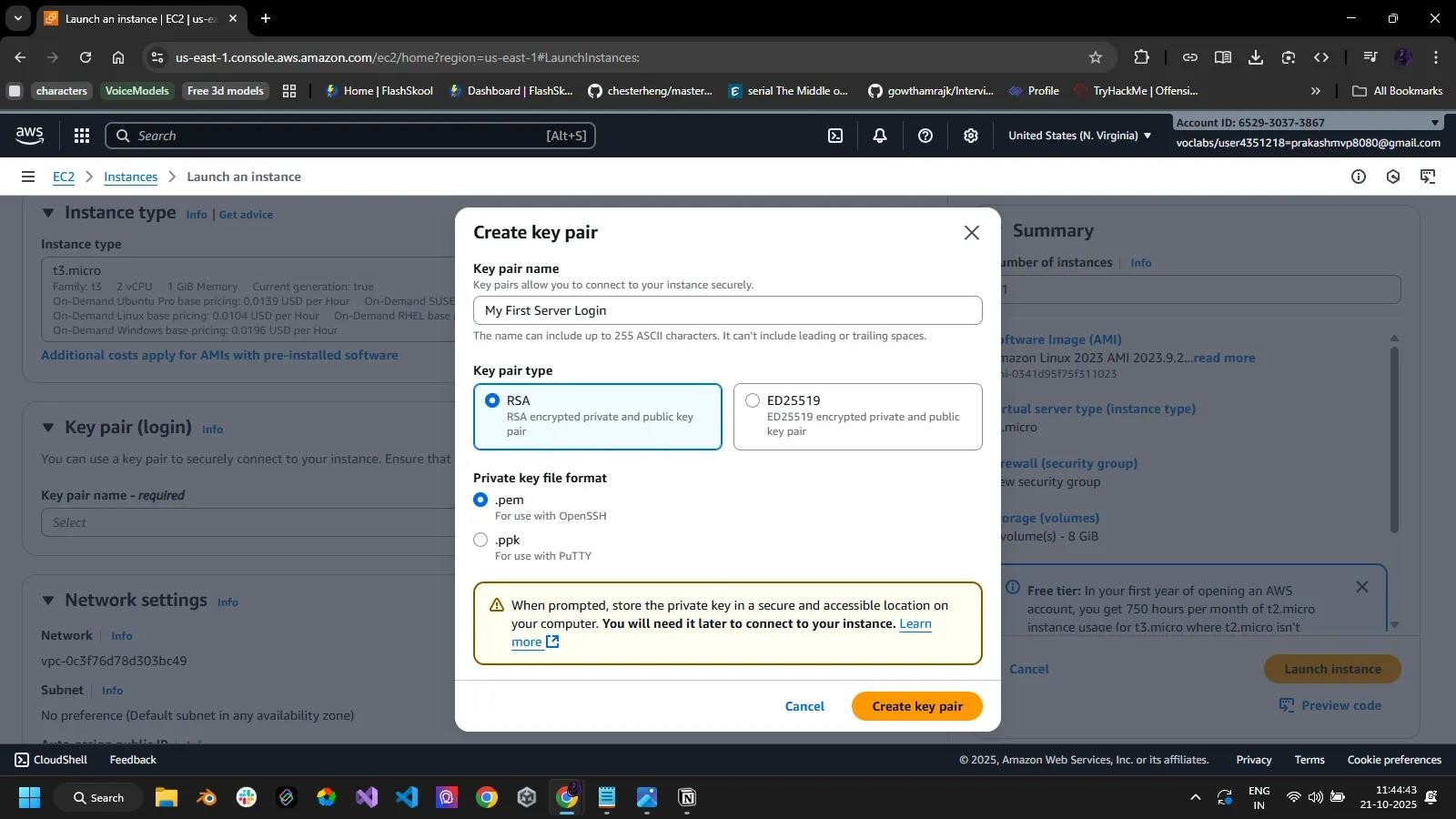
**Step1:**Create an Instance with amazon EC2, Give name for the instance and select the machine image (os) of the instance

# Screenshot 1:

****

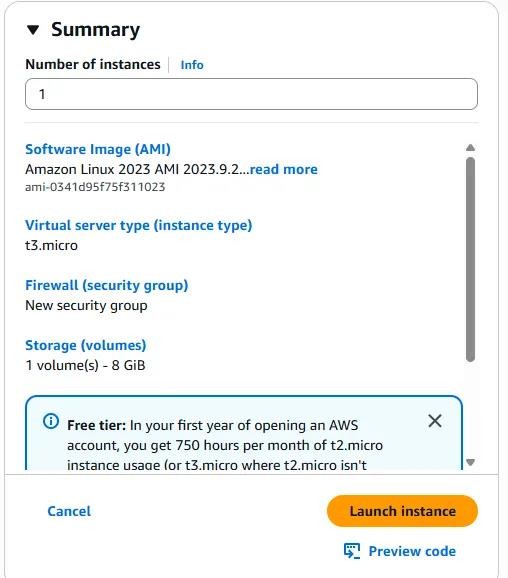
**Step2:** Create a key pair for the instance, this will be used to login to the virtual machine with open ssh connection

# Screenshot2:



**Step 3:** select the instance type , storage size requited and create a new security group for the instance

# Screenshot 3:

****

**Step 4:** Connecting to the instance with ssh using the .pem key file

# Screenshot 4:

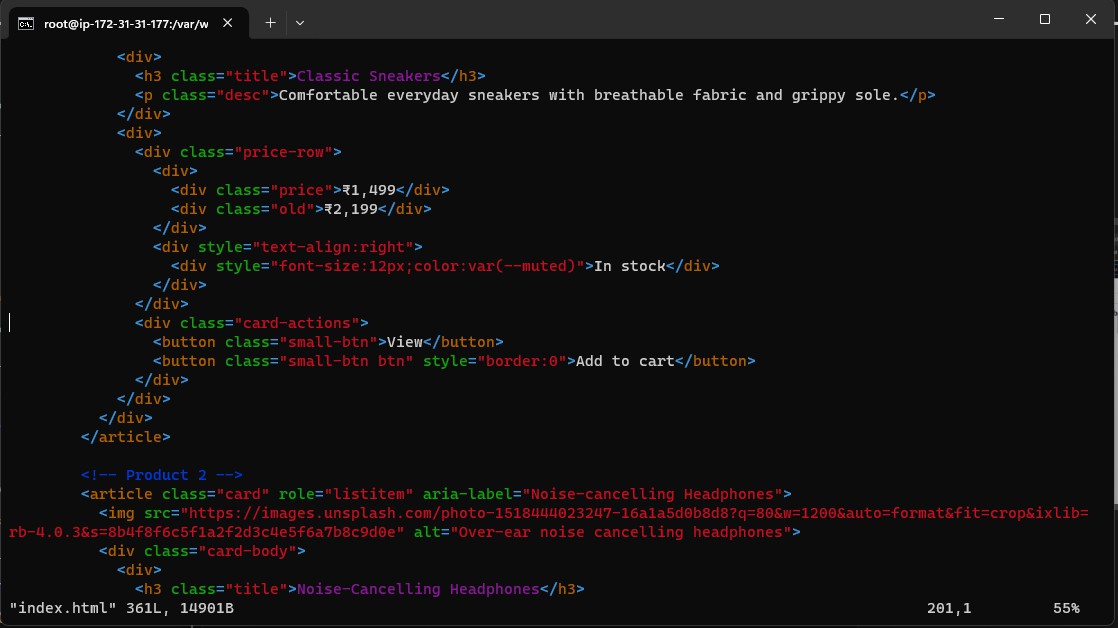
**Step 5:** install httpd and update the system to latest version using yum update -y and yum install httpd -y

# Screenshot 5:

**Deploy a simple web server on the created instance**

**Step 6:** Create a index.html file containing the code for the website

# Screenshot 6:

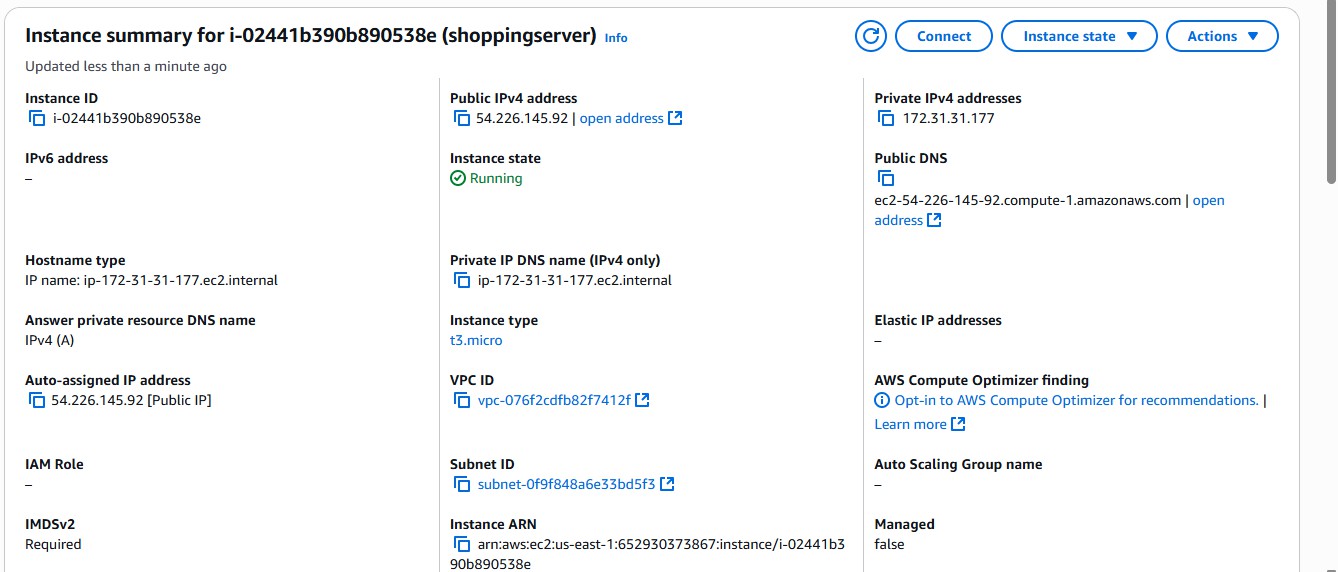
****

**Step 7:** start the htdp service to run the index.html and host in on the created instance in aws

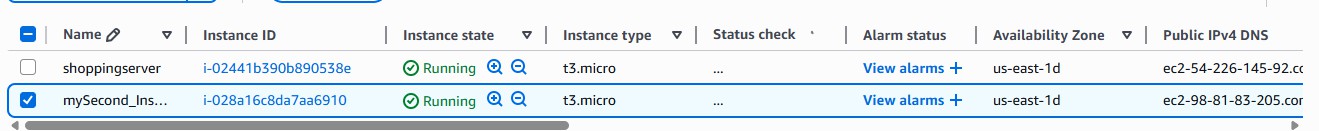
-service httpd start

**Step 8:** Use the public ip in the instance to launch the E commerce site

# Screenshot 8:

****

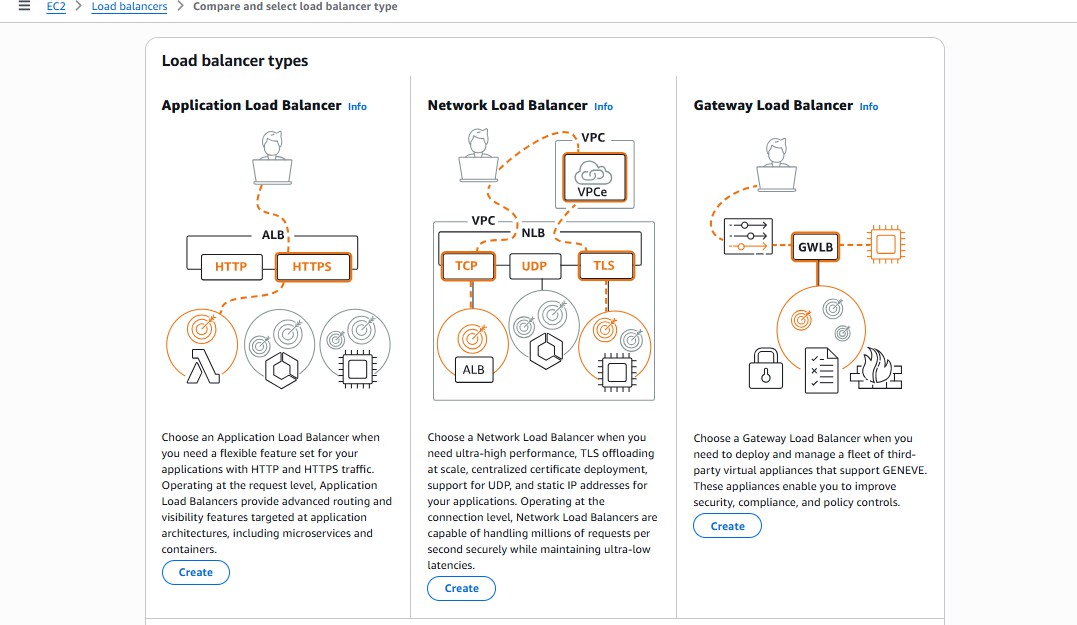
**Step 9:** Create another EC2 instance form the EC2 dashboard to create a load balancer



# Create a load balancer Screenshot 9:

**Step 10:** go to load balancer dashboard and create a new application load balancer

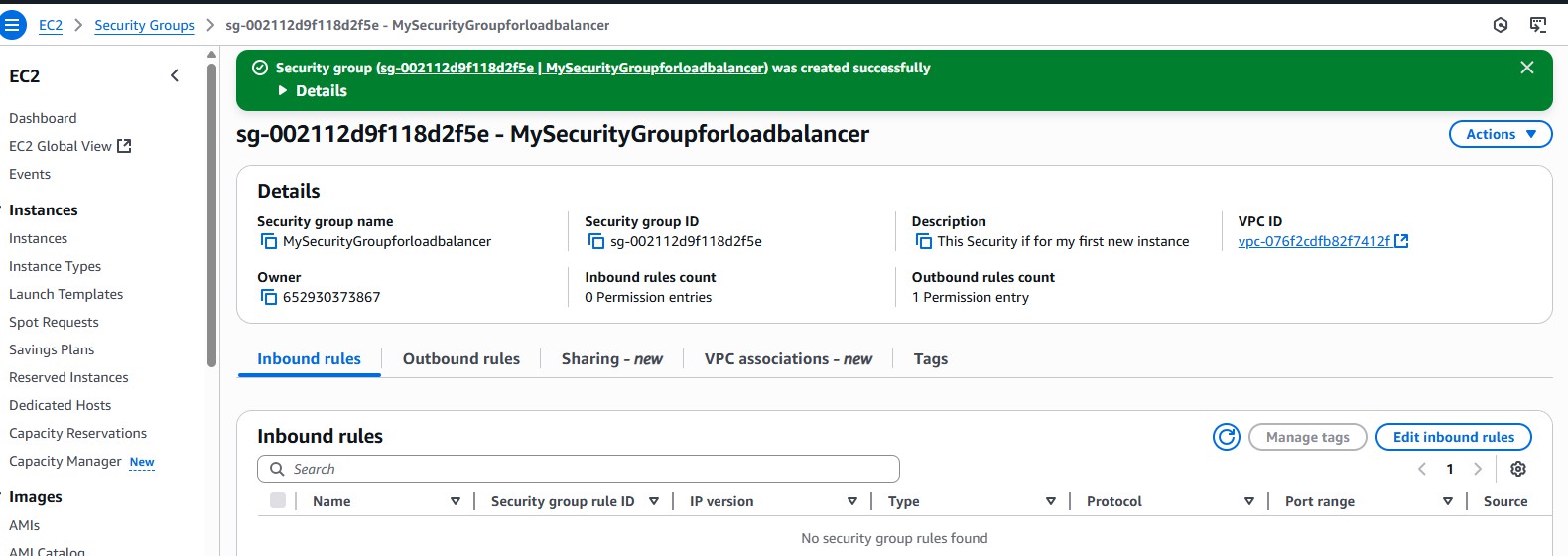
# Screenshot 10 :

****

**Step 11:** do the basic configurations for the load balancer , setup the VPC (virtual private

network) and finally create a new security for the loadbalancer and uncheck the default one.

# Screenshot 11:



**Step 12:** Create a new Target group for the load balancer

# Screenshot 12:

**Step 13 :** Verify the summary and create the load balancer

# Screenshot 13:

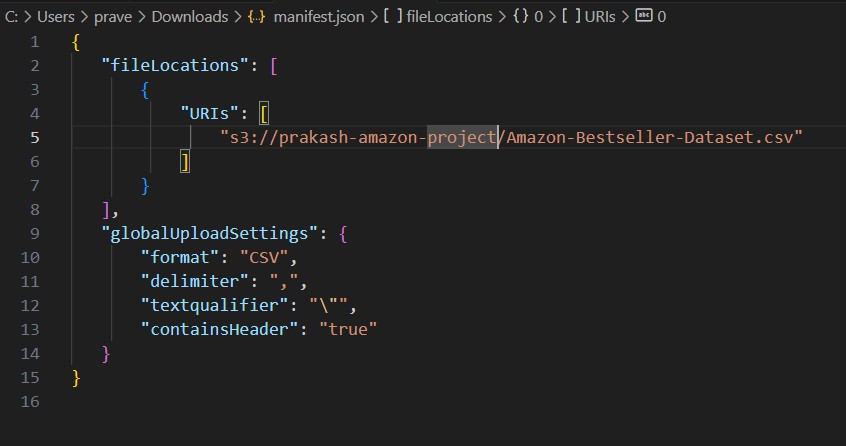
**Load sample data and visualize it using Matplotlib**

**Step 14 :** Create an amzon s3 bucket and upload the dataset in it

# Screenshot 14:

**Step 15:** Create a manifest.json file and upload the file and the dataset into the bucket

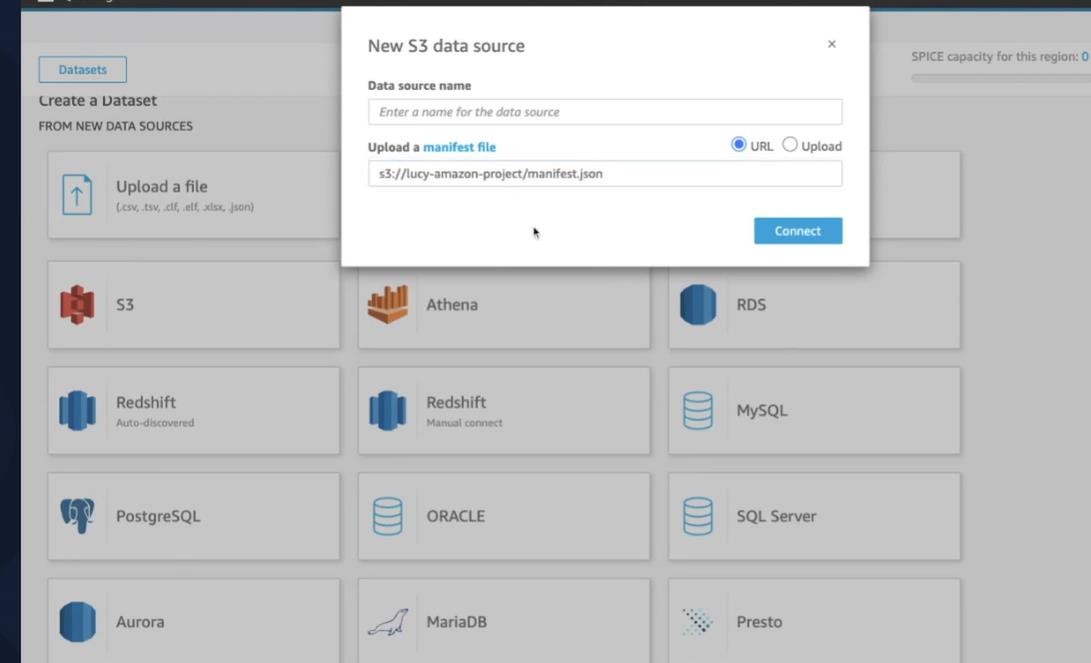
# Screenshot 15:

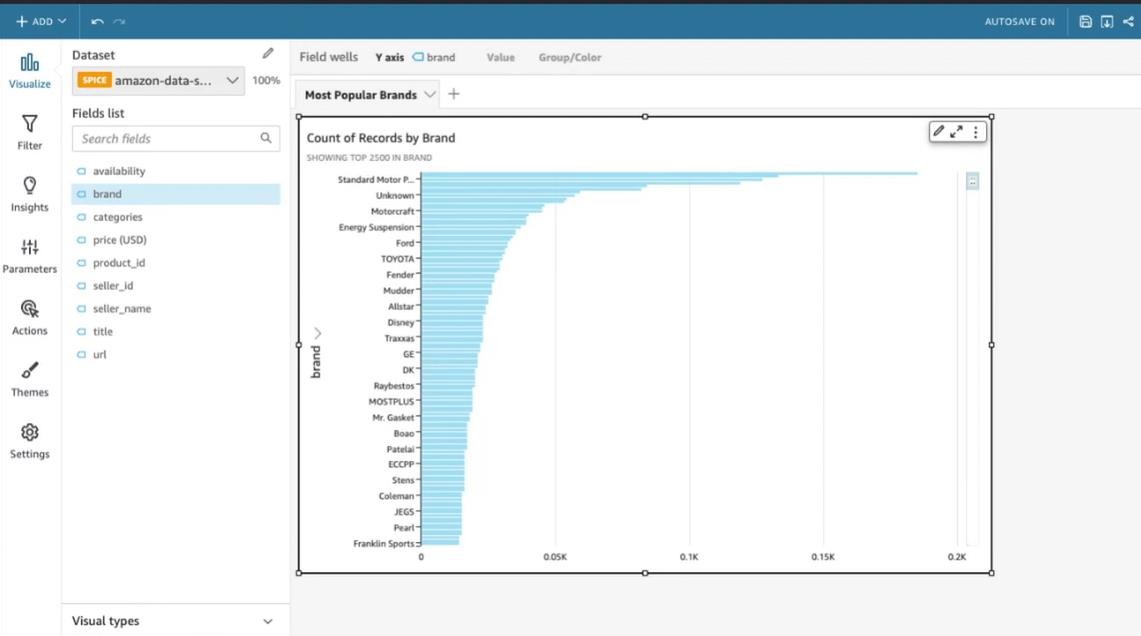
****

**Step 16 :C**reate a maazon quicksight account and open the quicksight tool

**Step 17:** Import the s3 dataset into quick Sight

# Screenshot 17:



Step 18: visualize the data in the dataset with quicksight visualization tool Screenshot 18:

**GitHub Link for Scenario #1:**

<https://github.com/RishabhKumarSingh-73/AWSlearning>

**Scenario - #2**

**Title:**

**Steps for Cloud Engineering:**

**Create a cloud EC2 instance named smv\_data-ai-node to host any two data processing and any one AI model services.**

**Step1:** Create a new instance named **smv\_data-ai-node** and setup basic configurations for it.

# Screenshot 1:

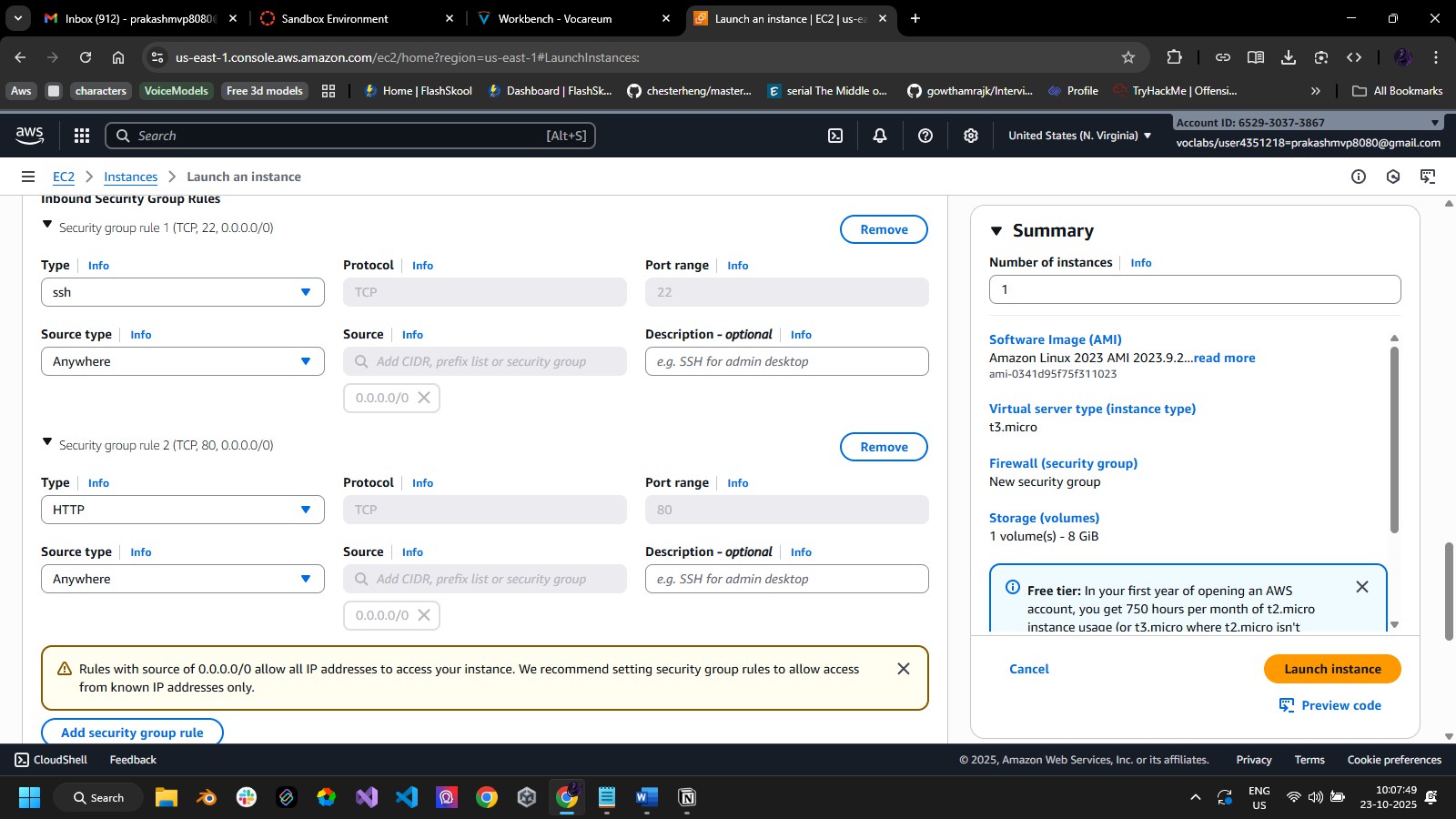
**Step2:** Create a keypair for this instance to remote remote login into the server with ssh connection

# Screenshot2:

**:**

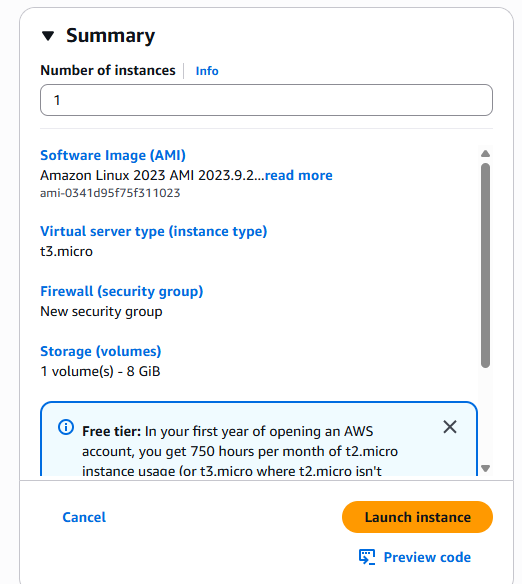
**Step 3:** Under the network settings of the instance , add a new security group rule and select the type to http and select source type as anywhere.

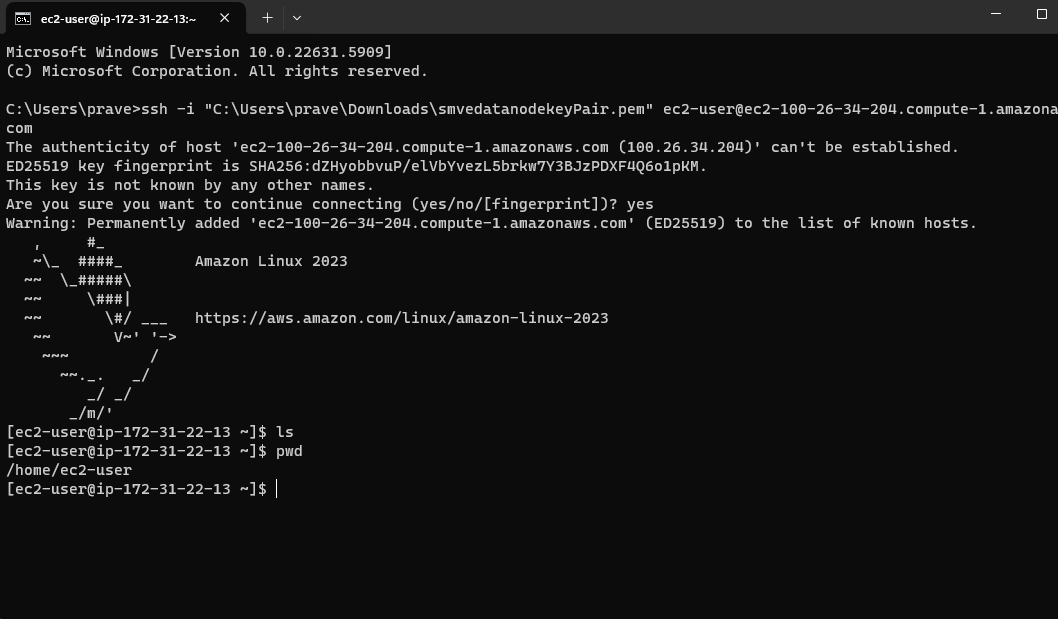
# Screenshot 3:

****

**Step 4:** Verify the summary and click on launch instance

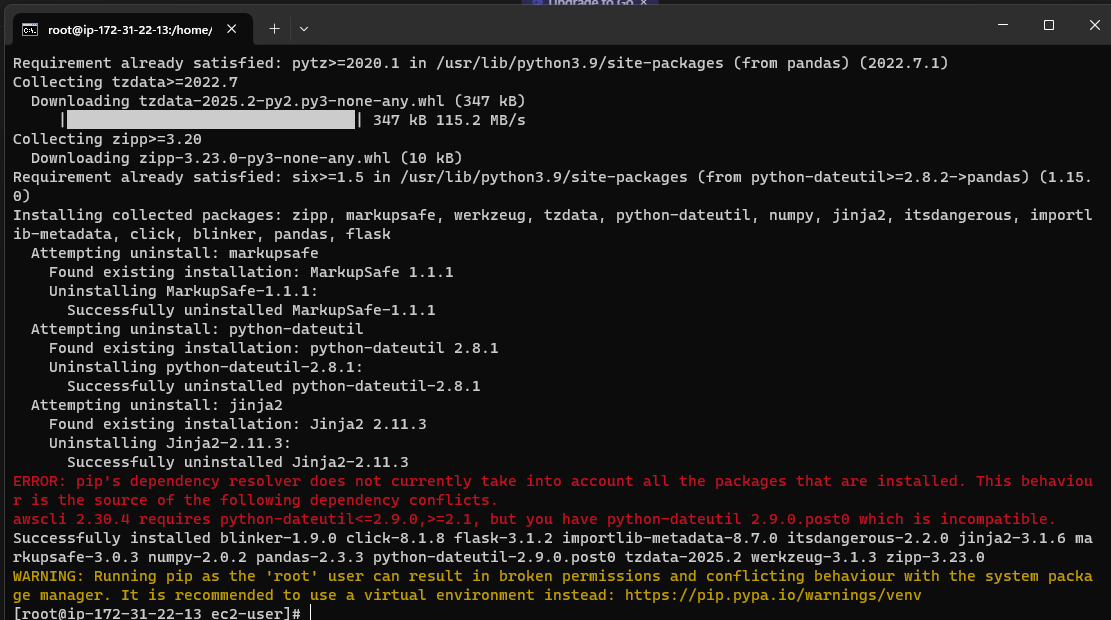
# Screenshot 4:

****

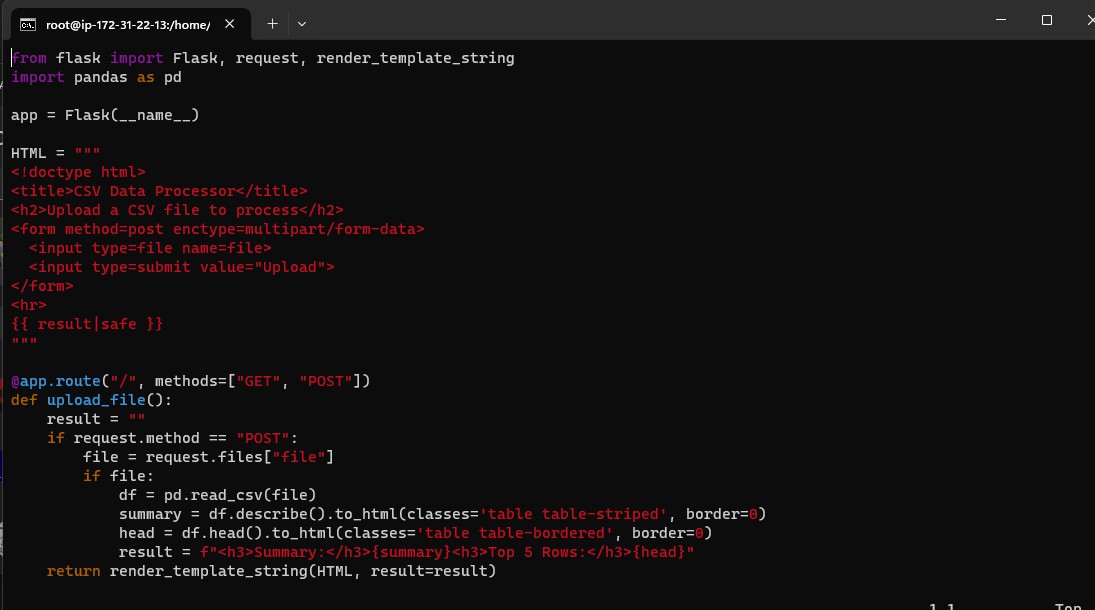
**Step 5:** Connect the instance with ssh using command prompt Screenshot 5:



**Step 6:** Install the necessary python packages Screenshot 6:



Step 7: Setup the code for data processing in python, use nano to create app.py and paste the code, once done , press ctrl+o to save and then press enter , to exit nano press ctrl + x

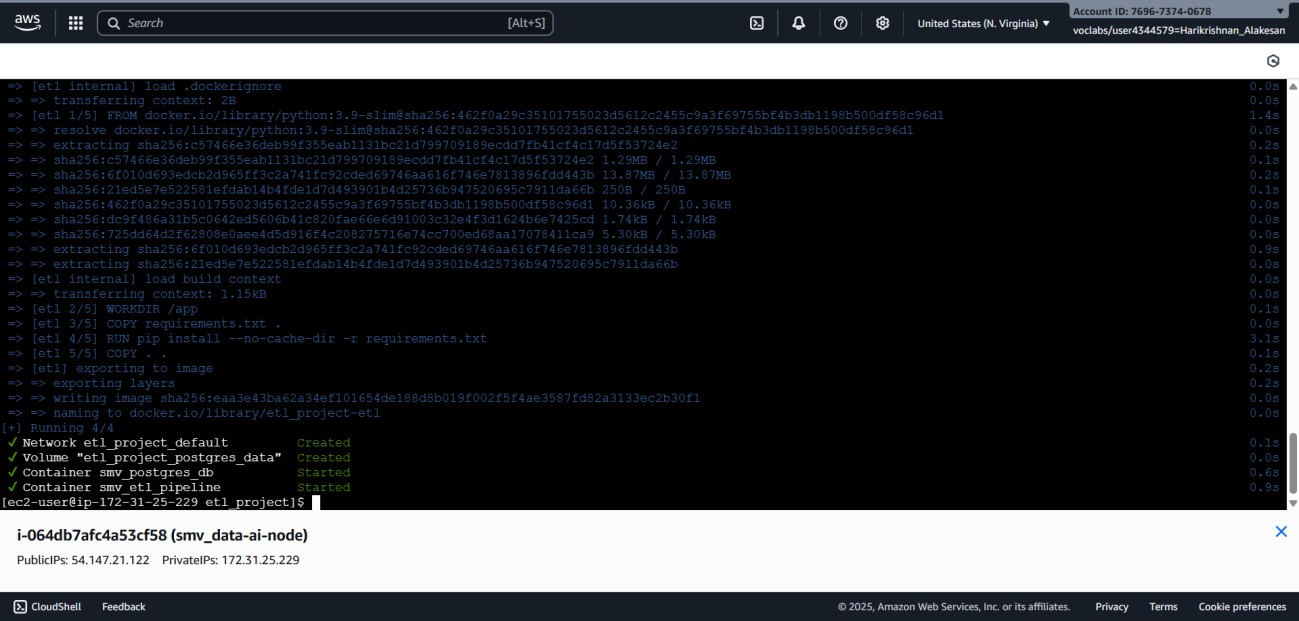
Screenshot 7:

# Step8:

Install and Configure a Containerization Platform A secure SSH connection was established to

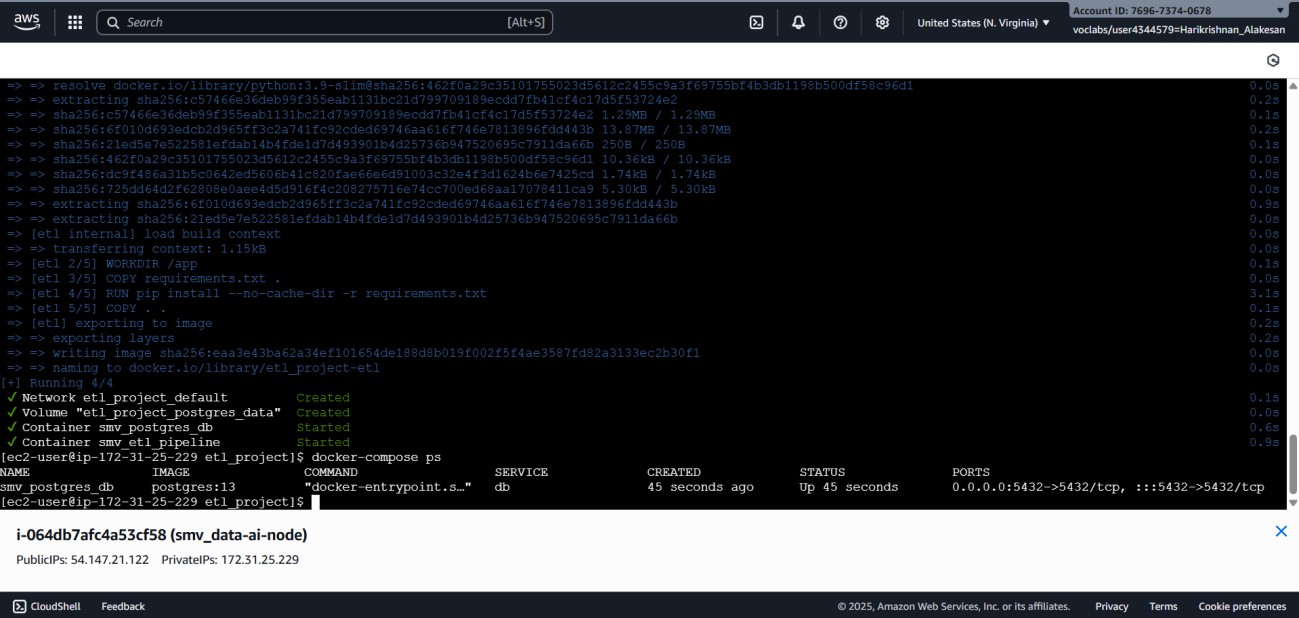
the instance, and Docker and Docker Compose were installed. This platform allows applications to be packaged into isolated containers, ensuring consistency and simplifying deployment. The successful installation was confirmed by checking the version numbers of both tools.

# Screenshot 8:

****

**Step 9:** Deploy a Containerized ETL Pipeline Using Docker Compose, a multi-container application was launched. This included:

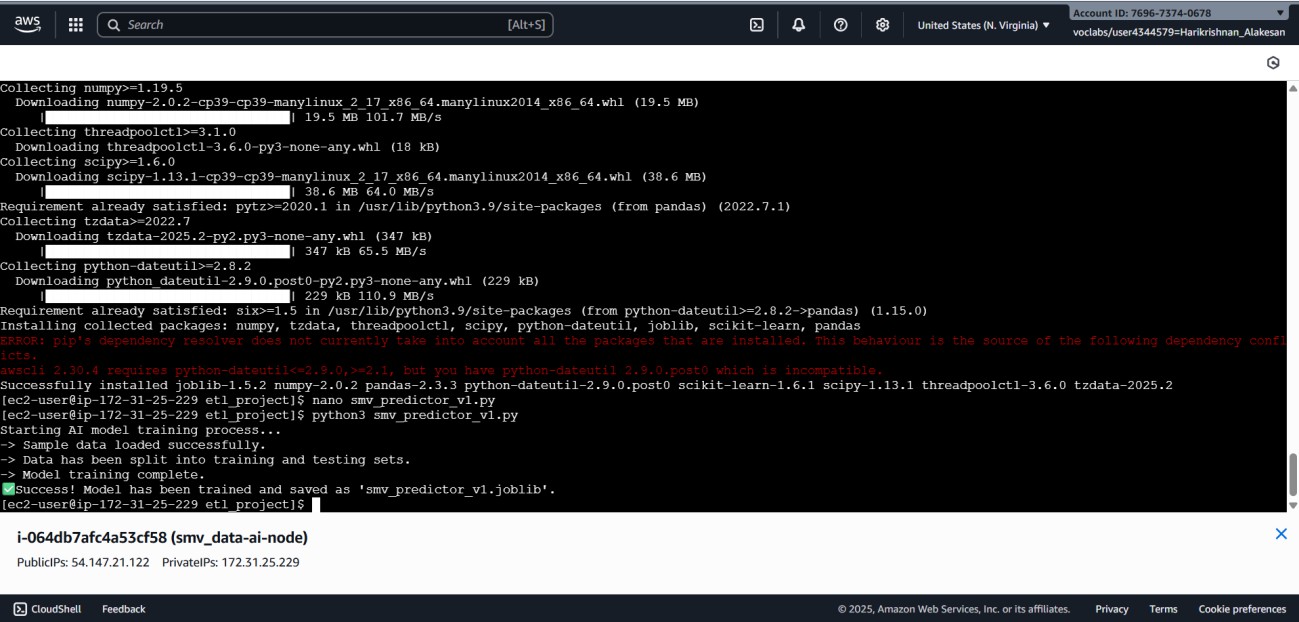
# Screenshot 9:

****

**Step 10:** Train and Save a Machine Learning Model The Python machine learning libraries Scikit-learn and Pandas were installed on the EC2 instance. A script was executed to train a simple linear regression model and save the trained artifact to a file named

smv\_predictor\_v1.joblib, making it available for future inference tasks.

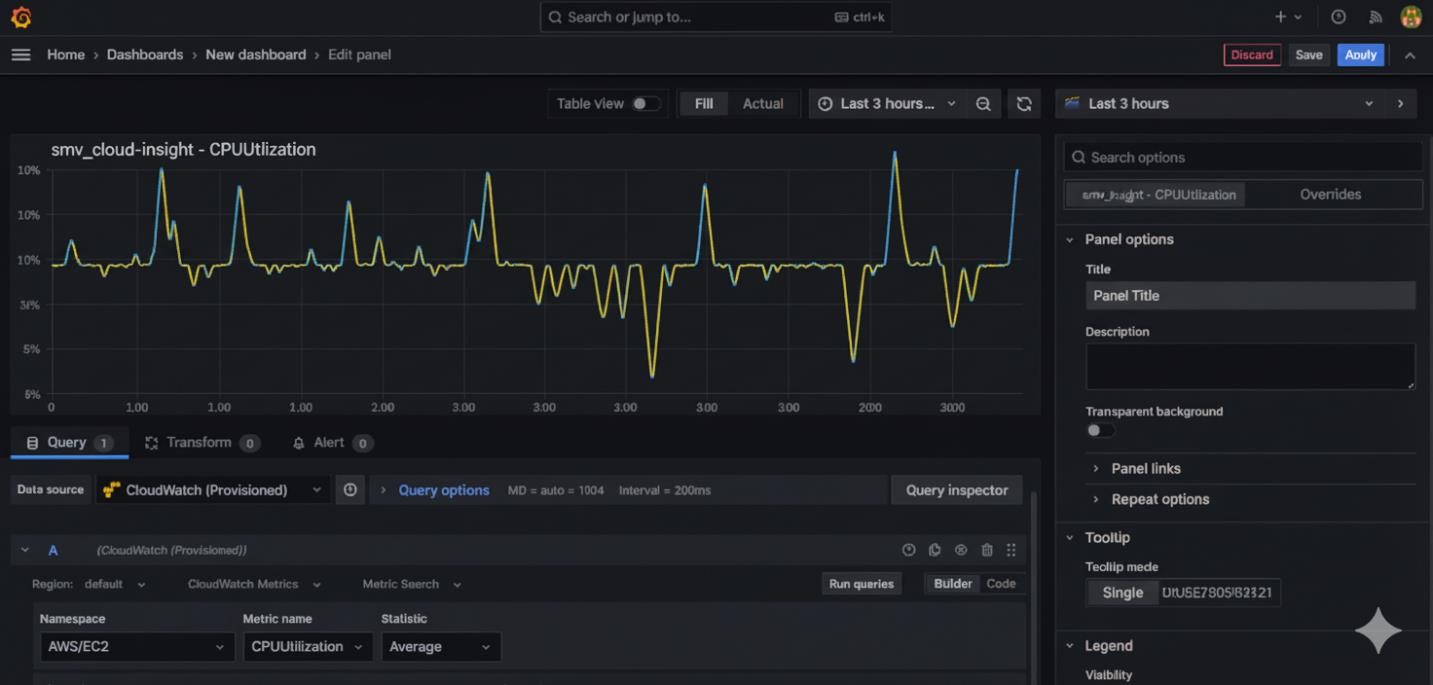
# Screenshot 10:



**Step 11:** Implement Real-Time Performance Monitoring Grafana was installed on the instance to serve as a monitoring dashboard. Due to permission limitations in the lab environment, a

workaround was implemented by provisioning the AWS CloudWatch data source with temporary session credentials. A dashboard named smv\_cloud-insight was created to visualize the instance's real-time CPU Utilization.

Screenshot 11:



**GitHub Link for Scenario #2:**

<https://github.com/RishabhKumarSingh-73/AWSlearning>

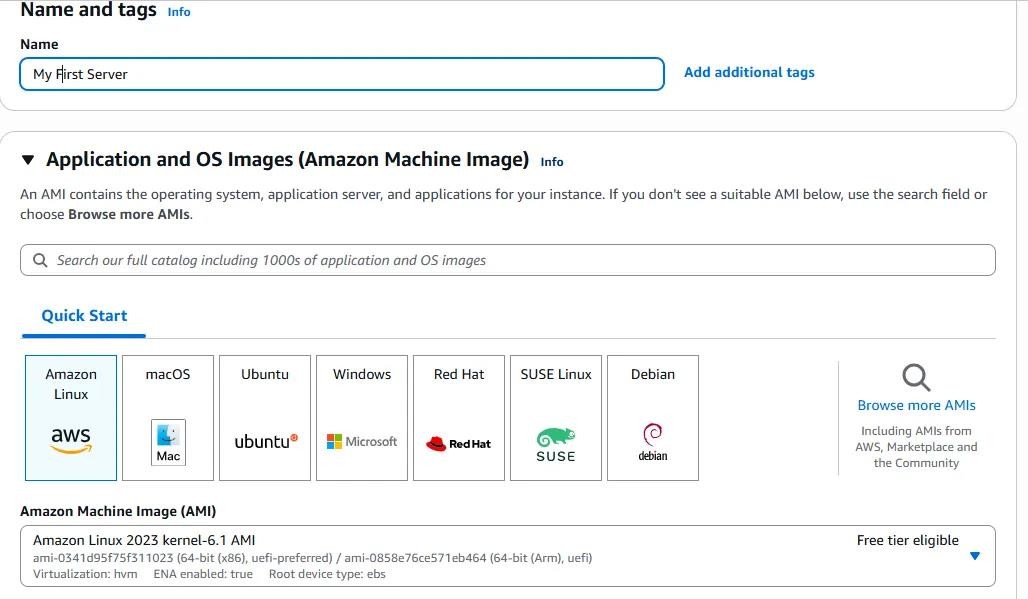
**Scenario - #3**

**Title: Use AI-driven dynamic pricing algorithms hosted on cloud servers with protected transaction data**

**Steps for Cloud Engineering:**

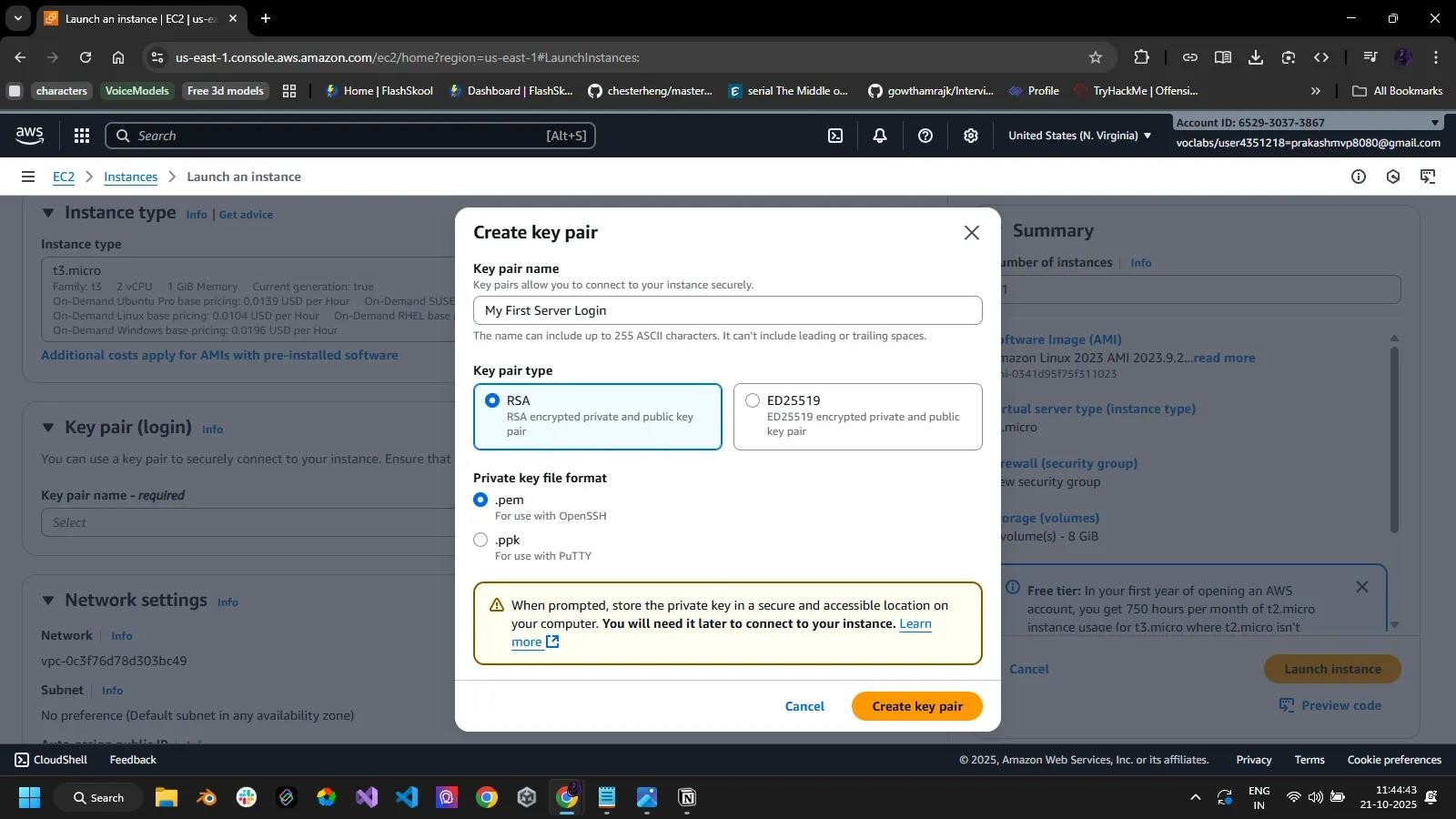
**Step1:**Create an Instance with amazon EC2, Give name for the instance and select the machine image (os) of the instance

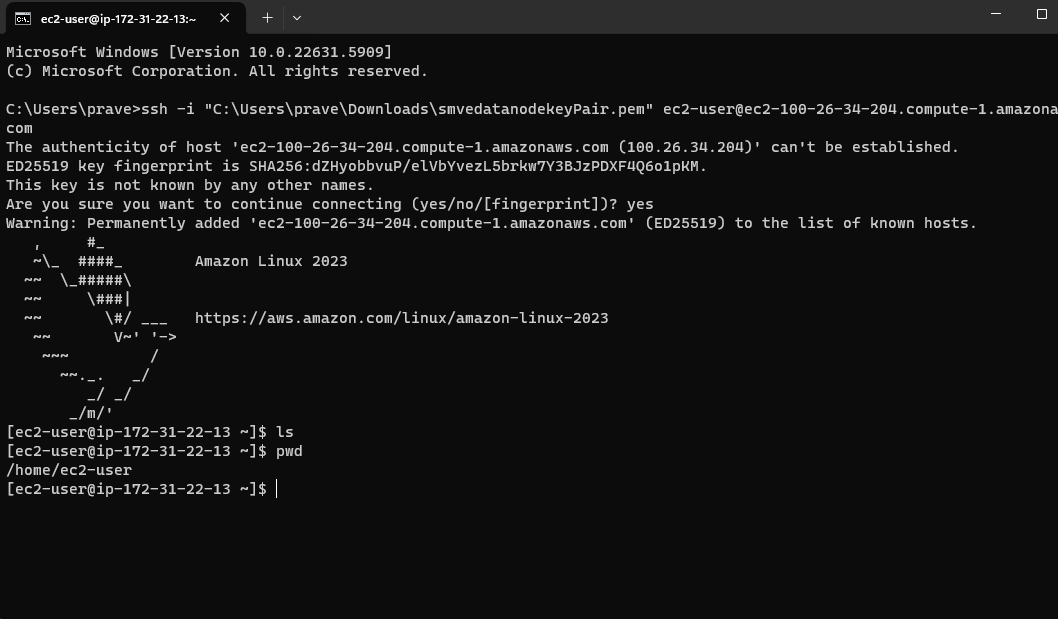
# Screenshot 1:

****

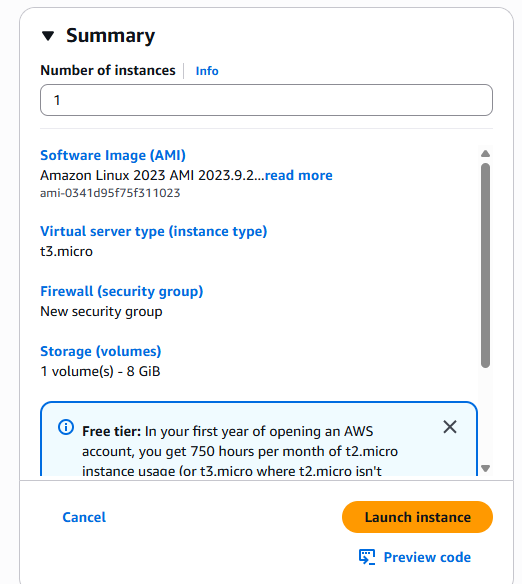
**Step2:** Create a key pair for the instance, this will be used to login to the virtual machine with open ssh connection

# Screenshot2:

****

**Step 3:** Verify the summary and click on launch instance

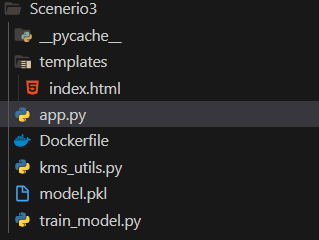
# Screenshot 3:

****

**Step 5:** Connect the instance with ssh using command prompt Screenshot 5:

**Step 6:** Make sure your project files (app.py, train\_model.py, model.pkl, requirements.txt, and templates folder) are ready locally.

# Screenshot 6:

****

**Step 7:** Transfer code to EC2 ,Use SCP or WinSCP to upload your project folder to the EC2 instance.

command: scp -i your-key.pem -r dynamic\_pricing/ ubuntu@<EC2-PUBLIC-IP>:/home/ubuntu/

**Step 8**: Install dependencies on EC2

**ssh -i your-key.pem ubuntu@<EC2-PUBLIC-IP>** Update the system and install Python 3 and pip: **sudo apt update**

# sudo apt install -y python3-pip python3-venv

Install project dependencies:

# cd dynamic\_pricing python3 -m venv venv source venv/bin/activate

**pip install -r requirements.txt**

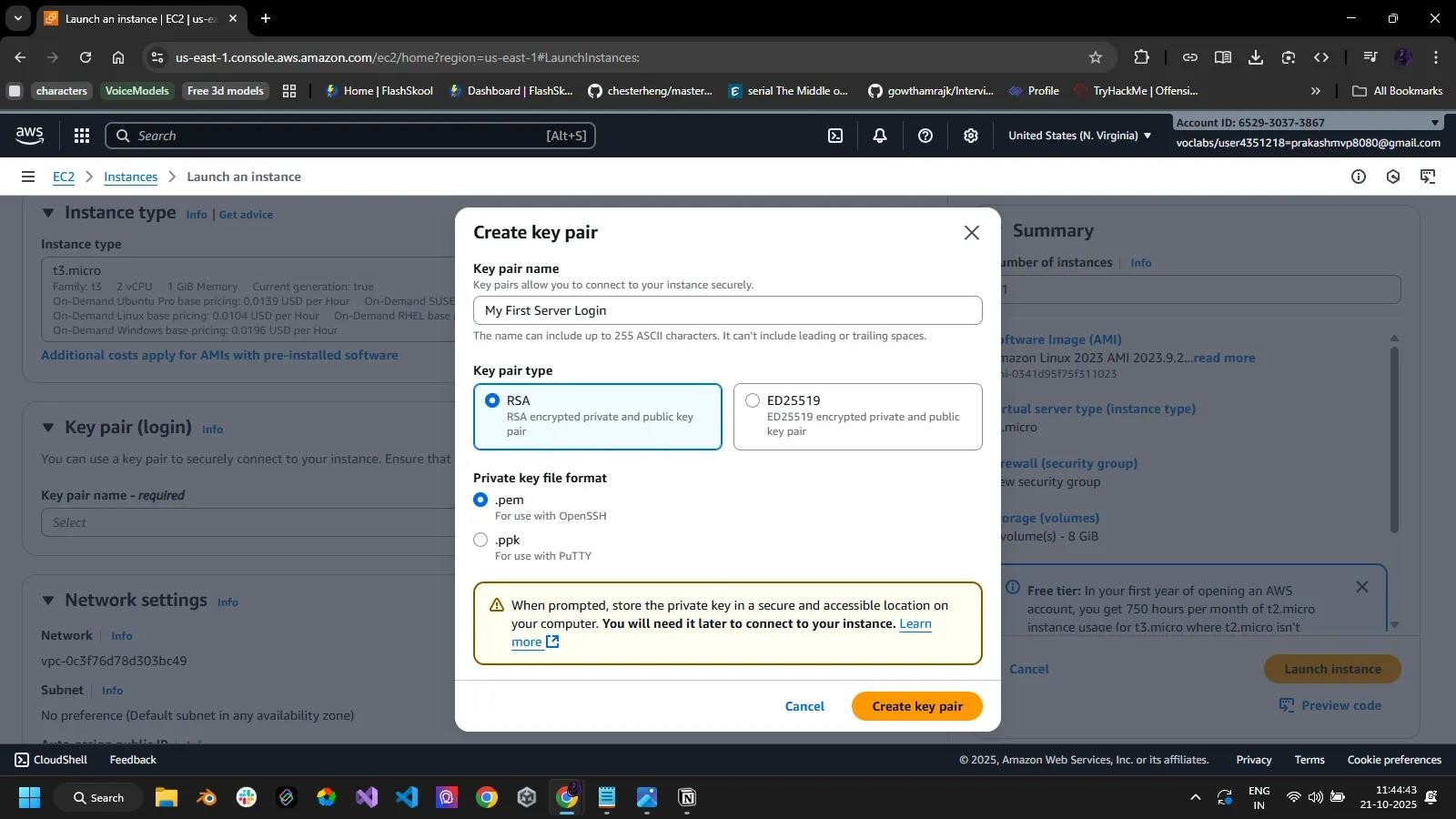
**Step 9:** Test the application locally on EC2 ,Run the Flask app: python app.py

# Step 10: Configure Security Group

1. Go to the EC2 dashboard → Security Groups → select the instance’s group.
2. Add **Inbound rule**:
   * Type: Custom TCP
   * Port Range: 8000
   * Source: Anywhere (or your IP for security)
3. Save changes.

This allows access to your Flask app from the browser.

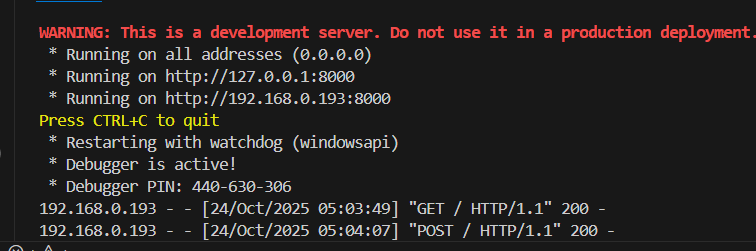
# Screenshot 7:

****

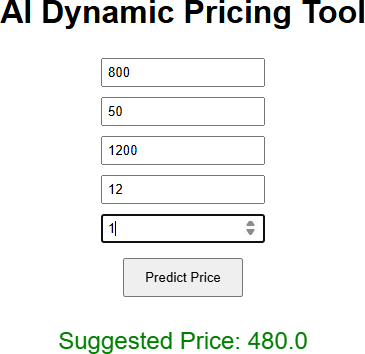
**Step 11: Run the app in the background (Optional)**

1. Install screen to run Flask continuously:
2. sudo apt install -y screen
3. screen -S ai\_pricing python app.py

**Screenshot 8:**

****

**Screenshot 9:**

****

**GitHub Link for Scenario #3:**

<https://github.com/RishabhKumarSingh-73/AWSlearning>