PRACTICAL: 6

AIM:

Scenario:

The café owners have noticed how popular their gourmet coffee offerings have become. Customers cannot seem to get enough of their cappuccinos and lattes. Meanwhile, the café owners have been challenged to consistently source the highest quality coffee beans. Recently, the owners learned that one of their favorite coffee suppliers wants to sell her company. Frank and Martha jumped at the opportunity to buy the company. The acquired coffee supplier runs an inventory tracking application on an AWS account. Sofía has been tasked to understand how the application works and then create a plan to integrate the application into the café's existing application infrastructure. In this lab, you will again play the role of Sofía, and you will work to migrate the application to run on containers. By the end of this lab, you will have migrated the application and the backend database to run as Docker containers. You will register these two containers in Amazon ECR to make them available to deploy as needed.

Lab overview and objectives:

In this lab, you will migrate a web application to run on Docker containers. The application is installed directly on the guest operating systems (OSs) of two Amazon Elastic Compute Cloud (Amazon EC2) instances. You will migrate the application to run on Docker containers.

After completing this lab, you should be able to:

- Create a Dockerfile.
- Create a Docker image by using a Dockerfile.
- Run a container from a Docker image.
- Interact with and administer your containers.
- Create an Amazon Elastic Container Registry (Amazon ECR) repository.
- Authenticate the Docker client to Amazon ECR.
- Push a Docker image to Amazon ECR.

THEORY:

Migrating the café's inventory tracking application to Docker containers involves several key steps to modernize and streamline its deployment. The process begins with creating a **Dockerfile**, which serves as a blueprint for the application's environment, specifying the base image, dependencies, and configurations required to run the application. Using this Dockerfile, a **Docker image** is built, packaging the application and its dependencies into a consistent, reusable format.

Once the image is ready, it is used to run the application as a **Docker container**, providing a lightweight, isolated environment that ensures portability and efficient resource utilization. Managing these containers involves using Docker commands to start, stop, inspect, and monitor their operations. This approach simplifies administration and enhances the reliability of the application.

To facilitate deployment on AWS, an **Amazon Elastic Container Registry (ECR)** repository is created. ECR is a secure, fully managed Docker container registry that integrates seamlessly with other AWS services. The **Docker client** is then authenticated to interact with the ECR repository, ensuring secure communication. Finally, the Docker image is **pushed to ECR**, making it available for deployment on platforms like AWS Elastic Container Service (ECS) or Kubernetes. This migration enables the café to achieve portability, scalability, and seamless integration with AWS infrastructure.

CODE:

N/A

OUTPUT:

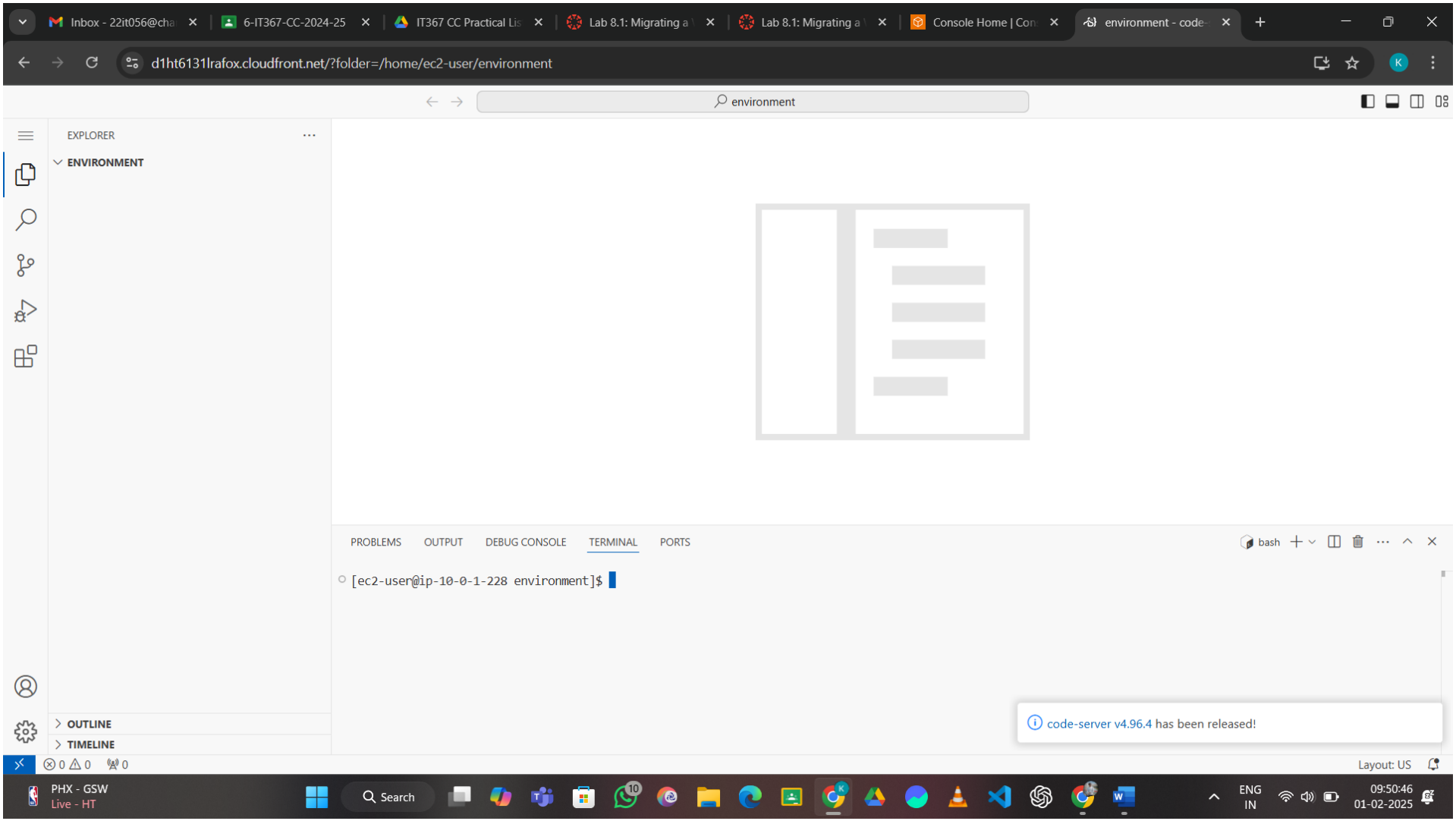

Figure 1: Open vs code in browser

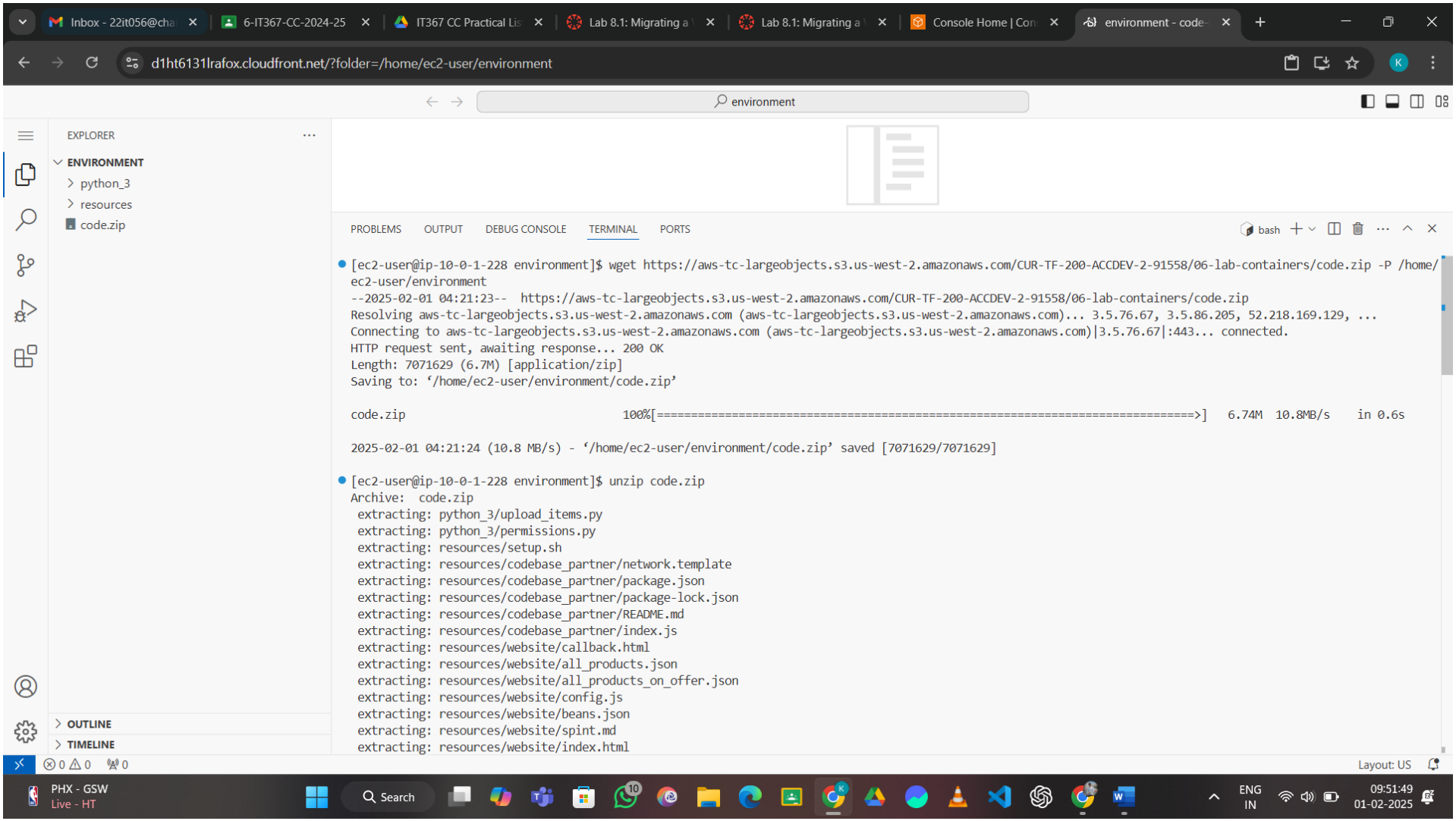

Figure 2: Download a zip file and extract it

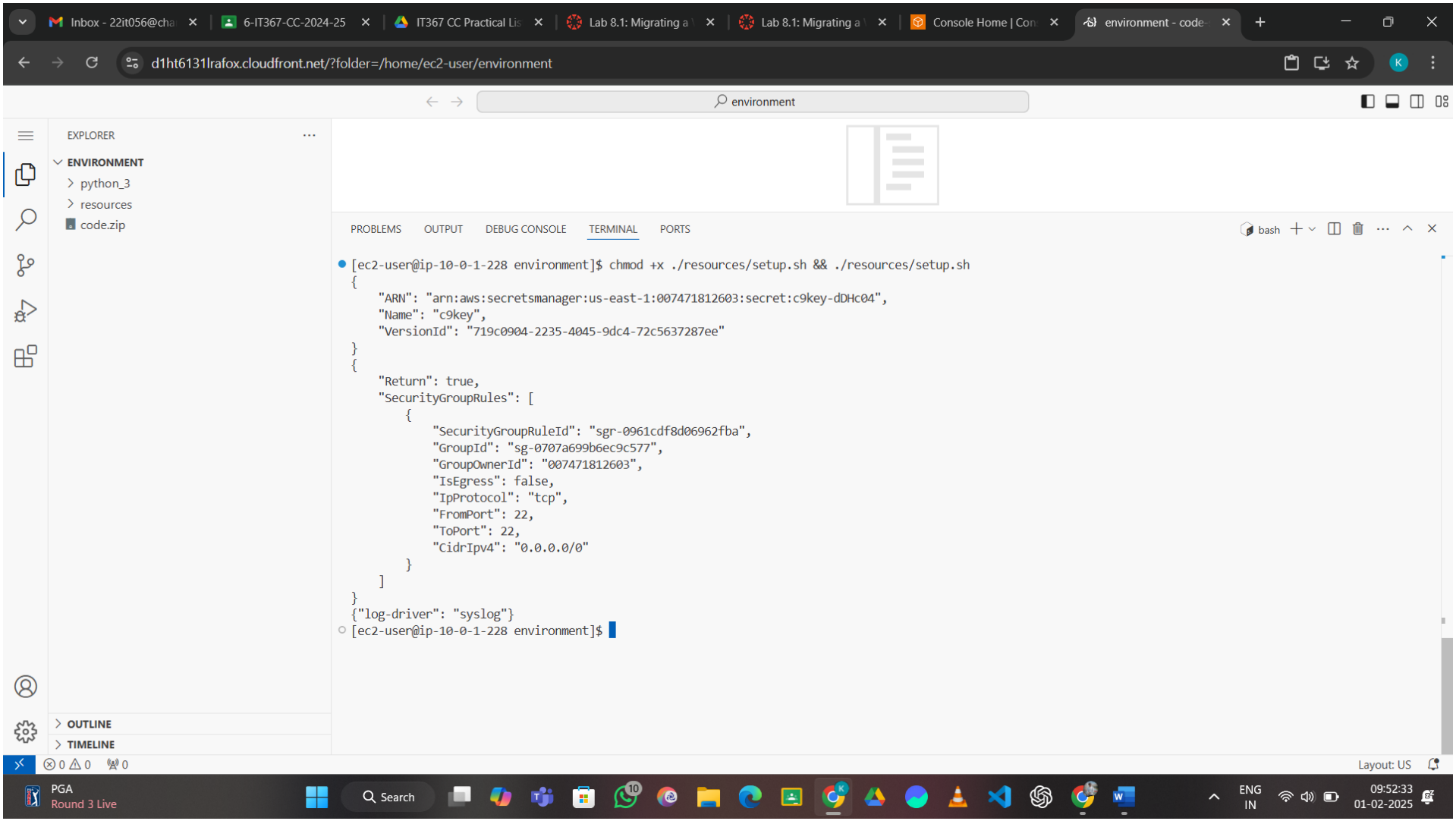

Figure 3: Change mode and access it

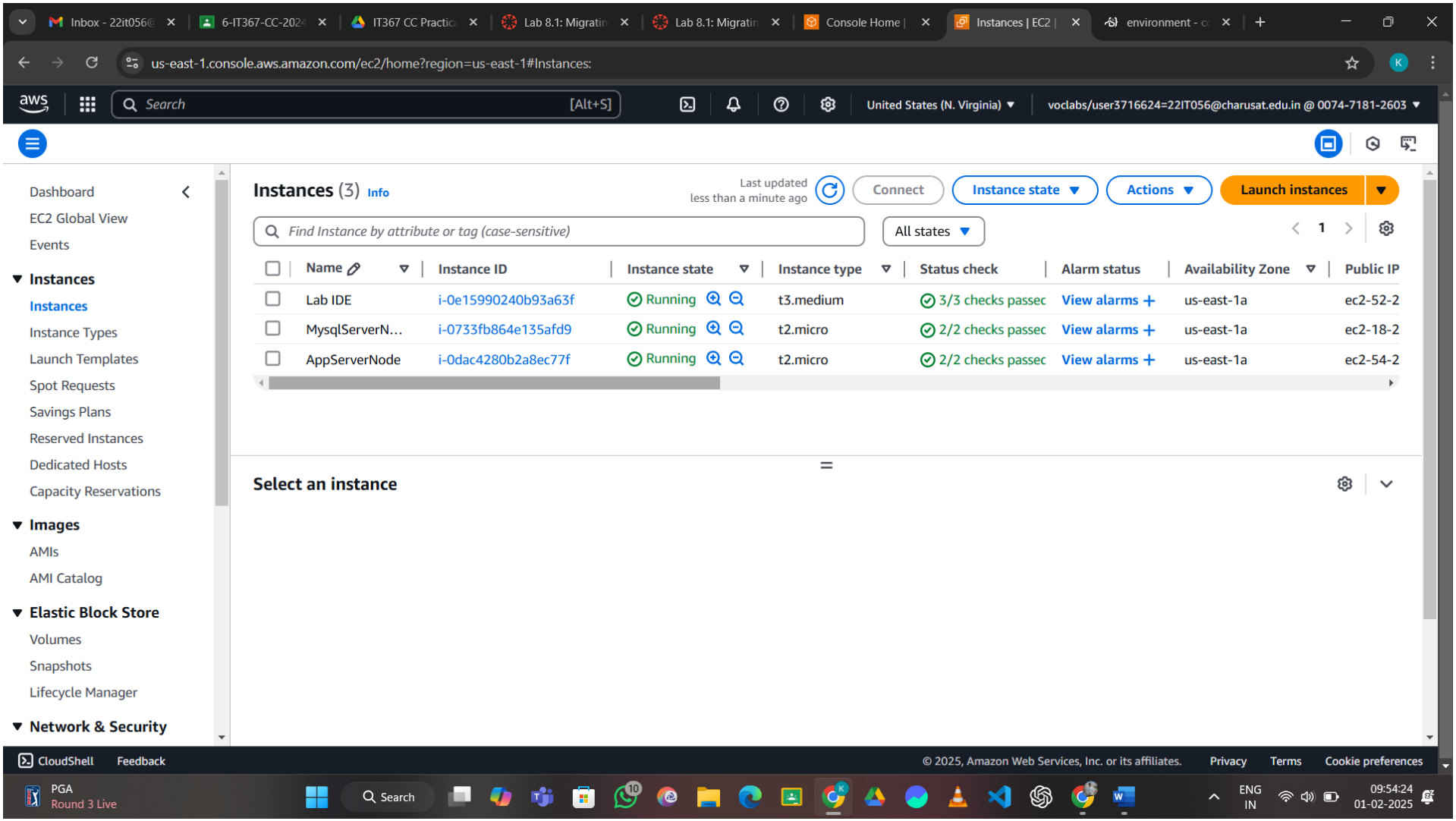

Figure 4: Check instances which are running

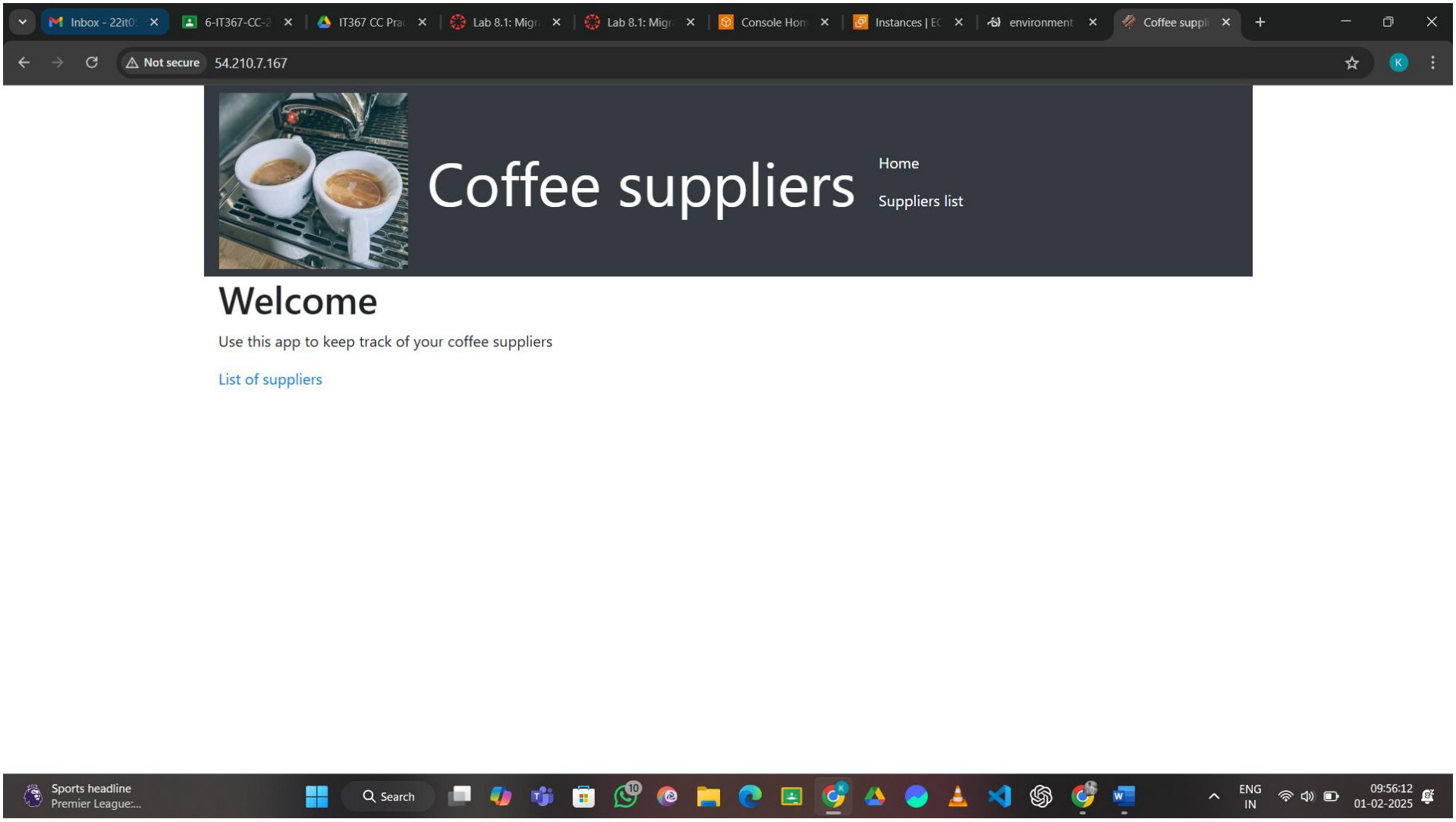

Figure 5: Open a website in browser

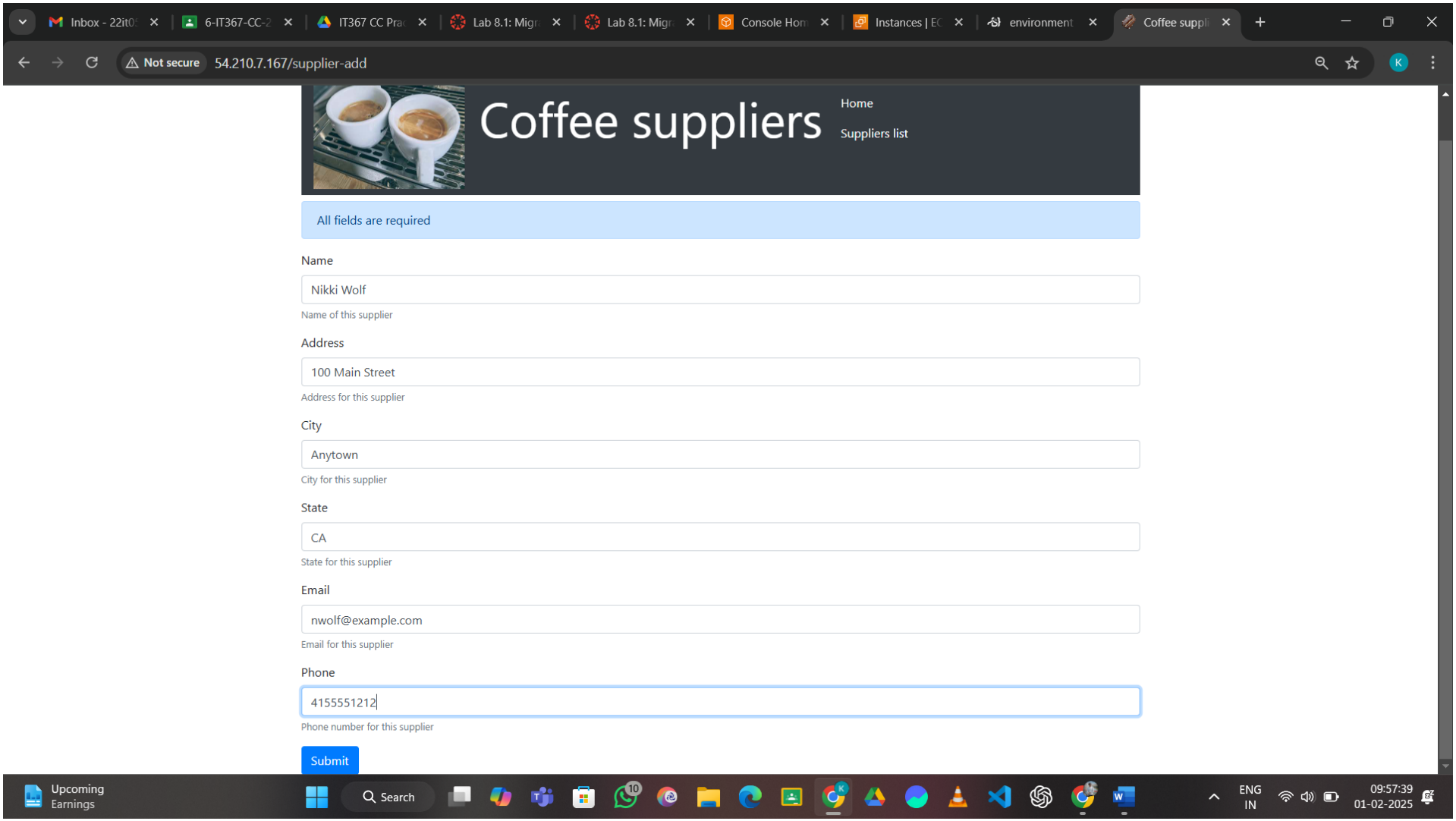

Figure 6: Add new supplier

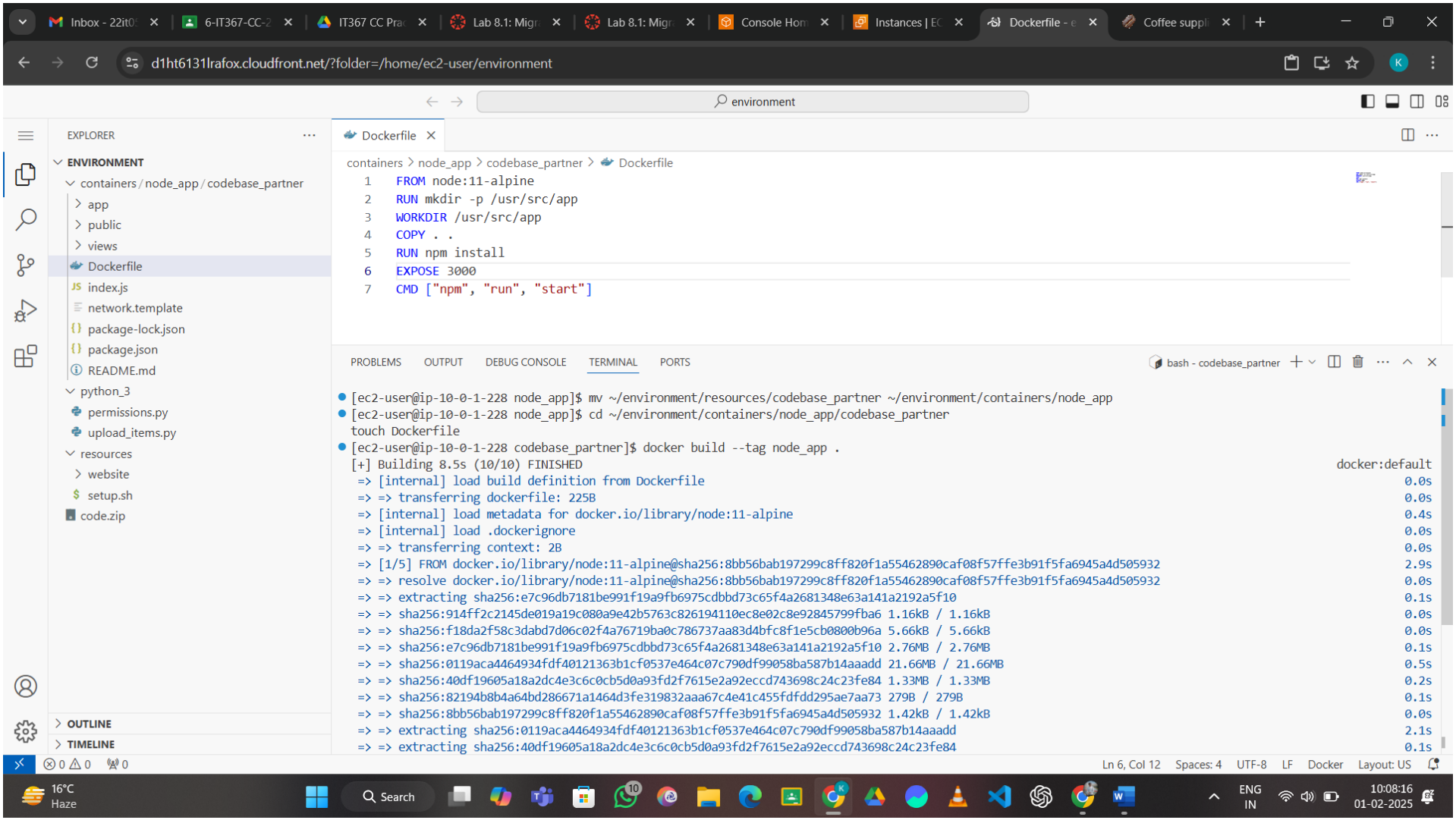

Figure 7: Create a docker file and pushed it

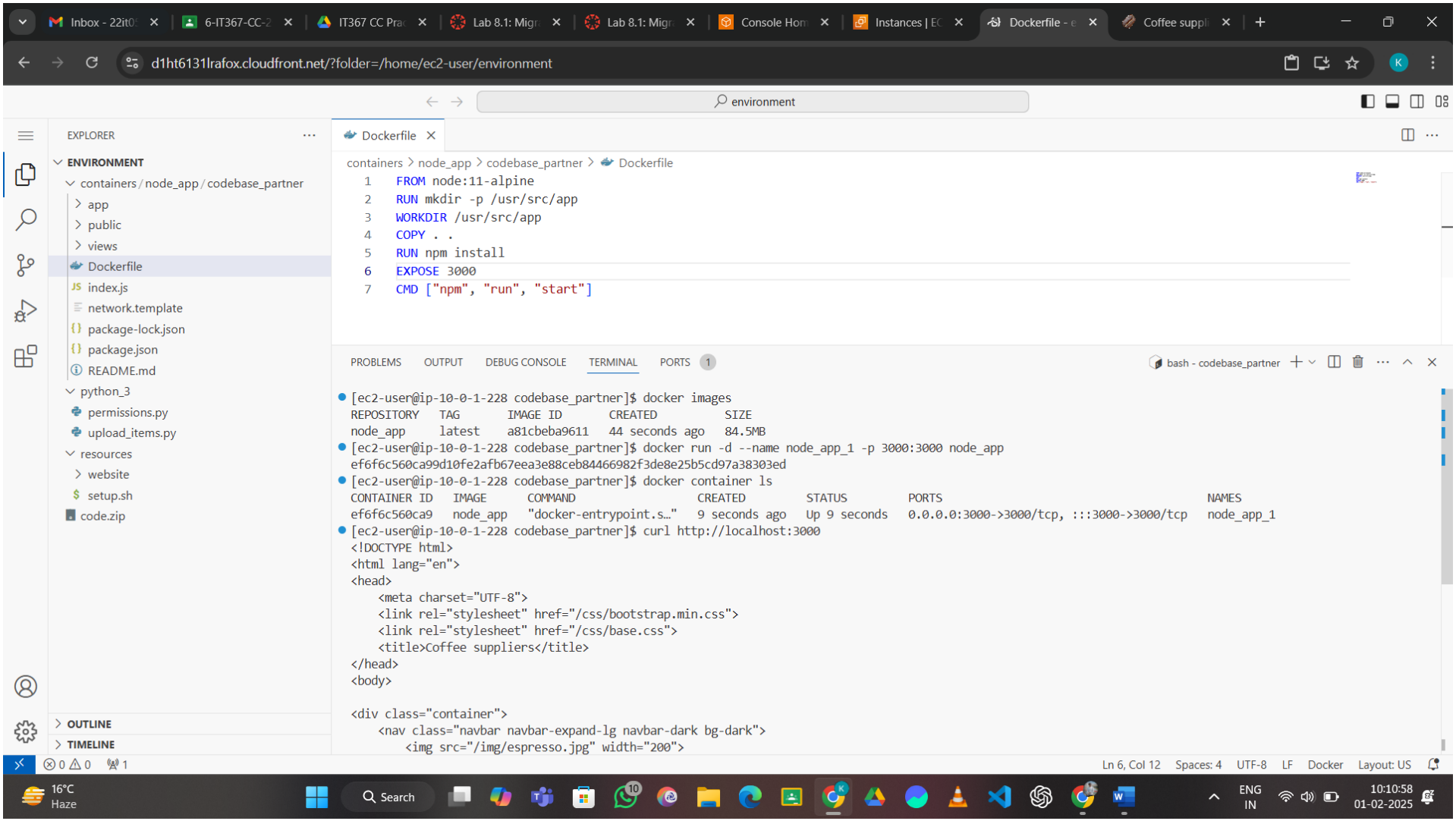

Figure 8: Check docker image and container

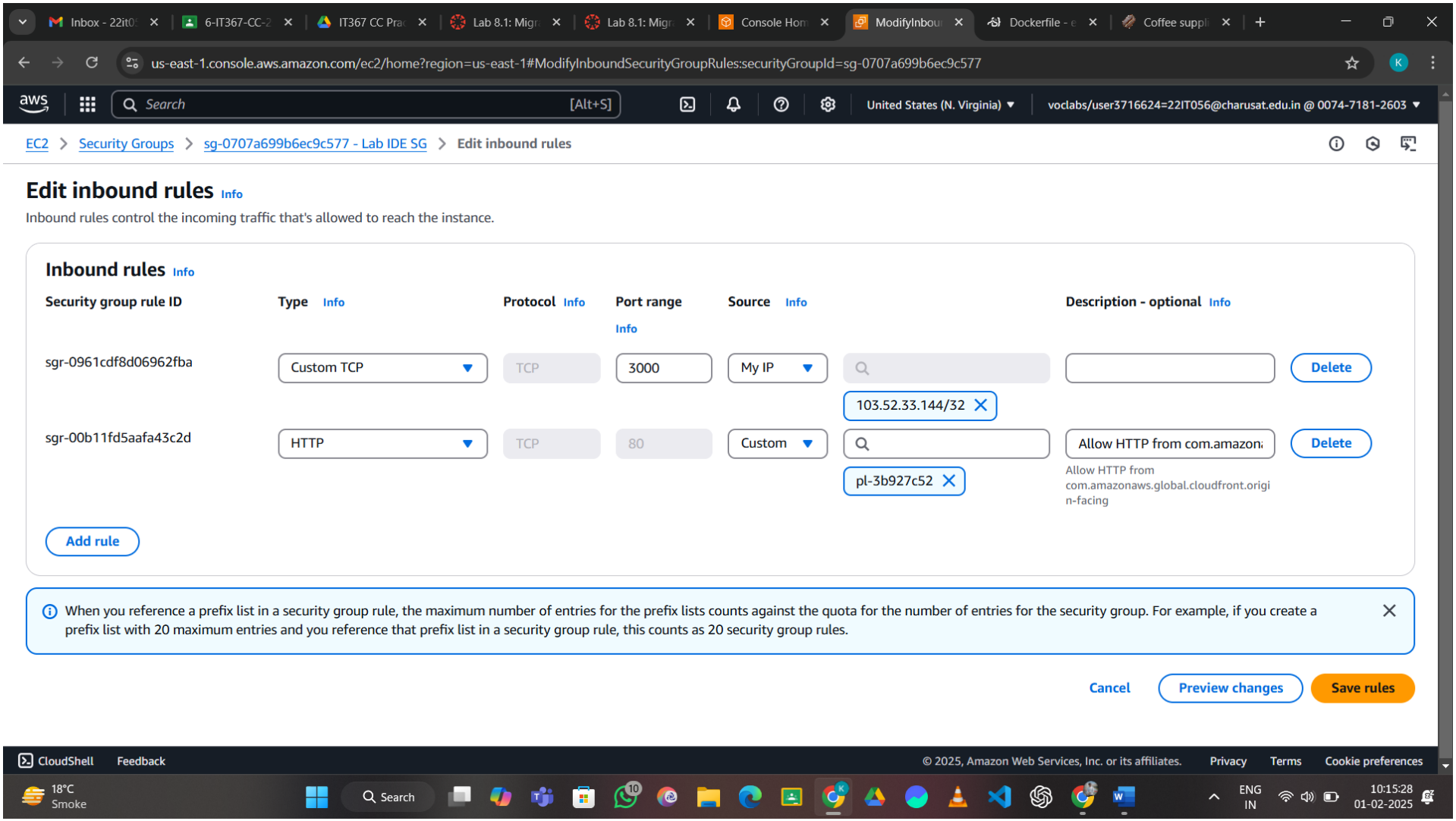

Figure 9: Edit inbound rules

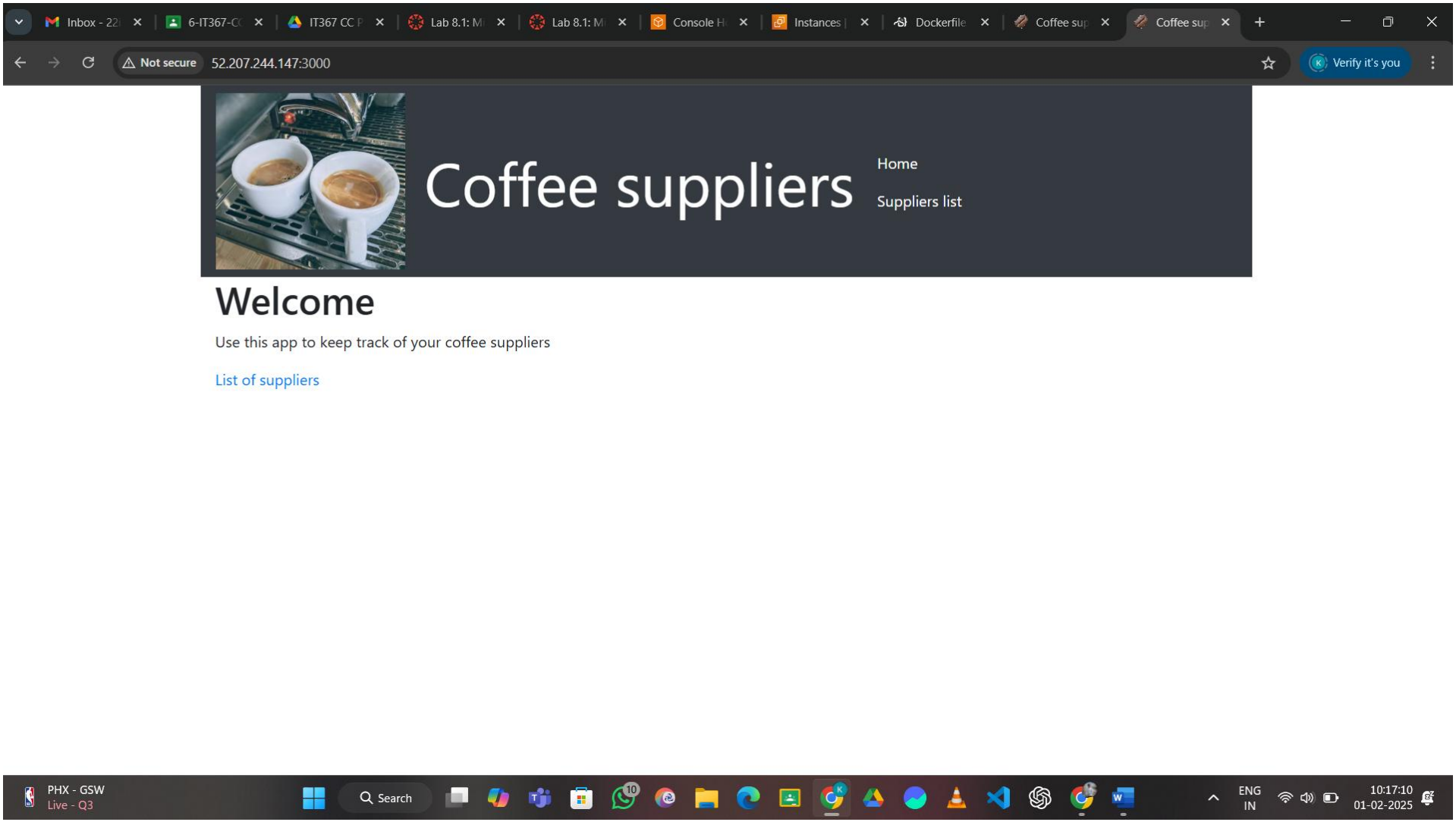

Figure 10: Open a docker file which run on 3000 port

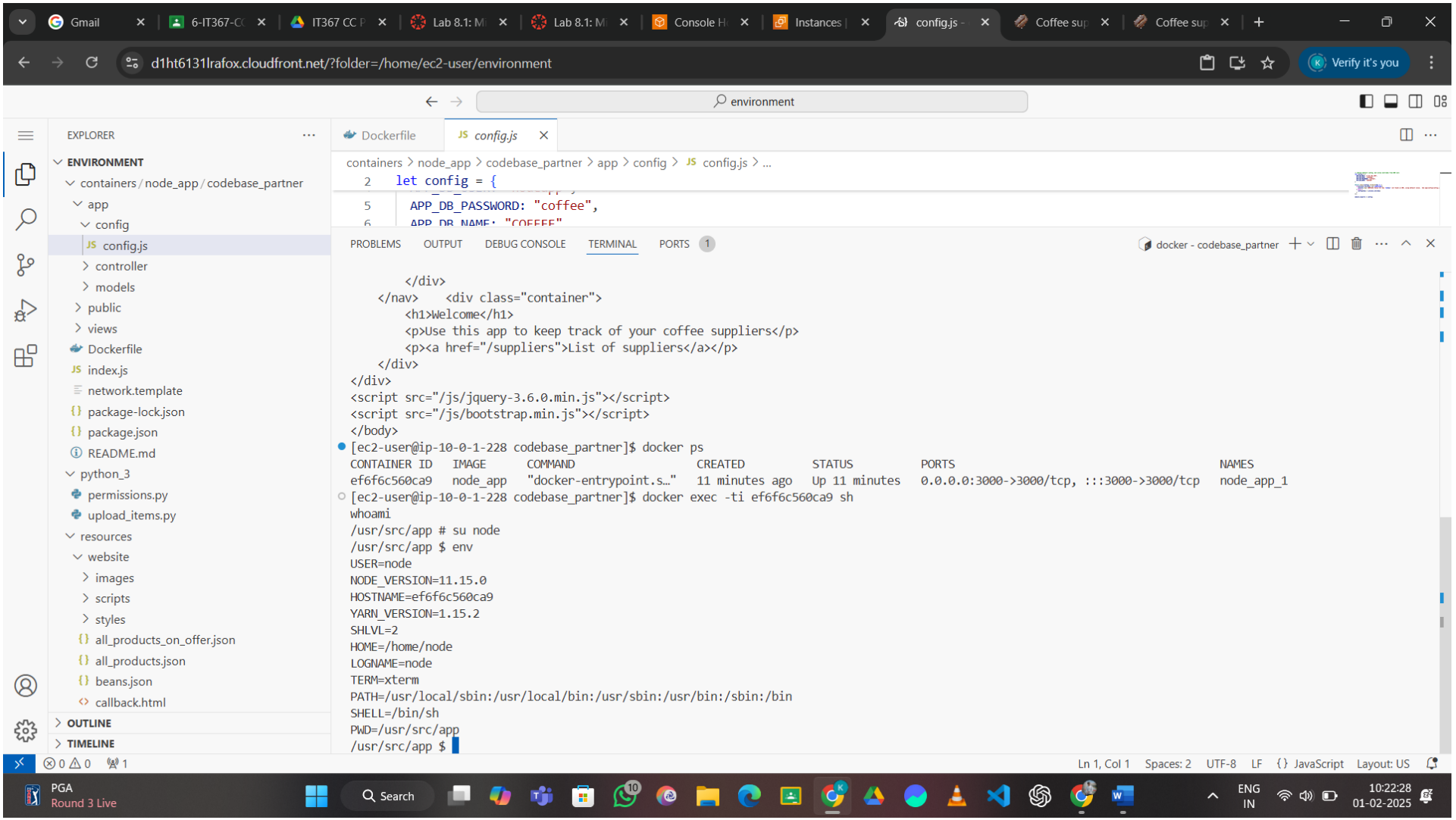

Figure 11: Set environment in docker

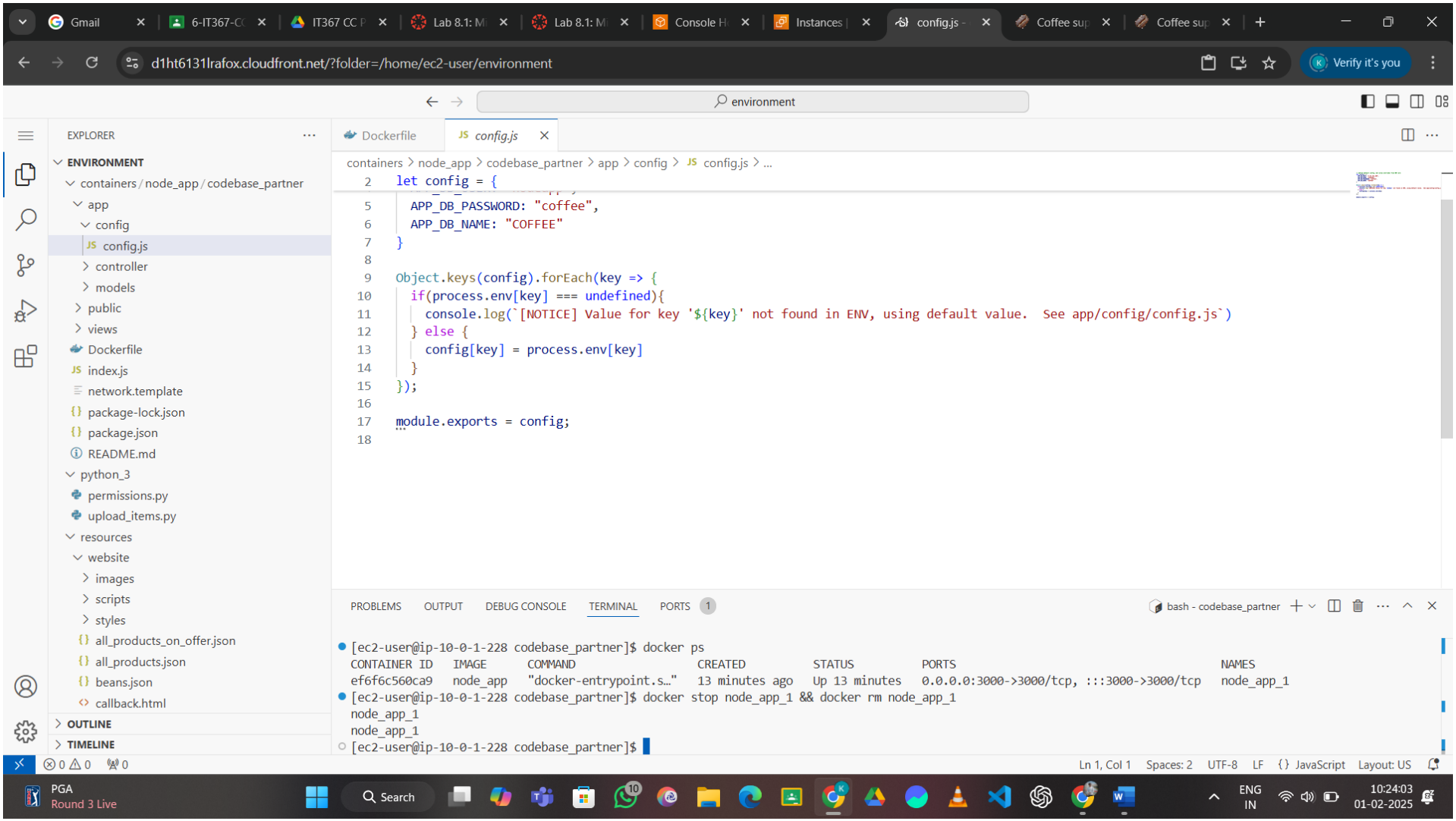

Figure 12: stop and remove node_app_1

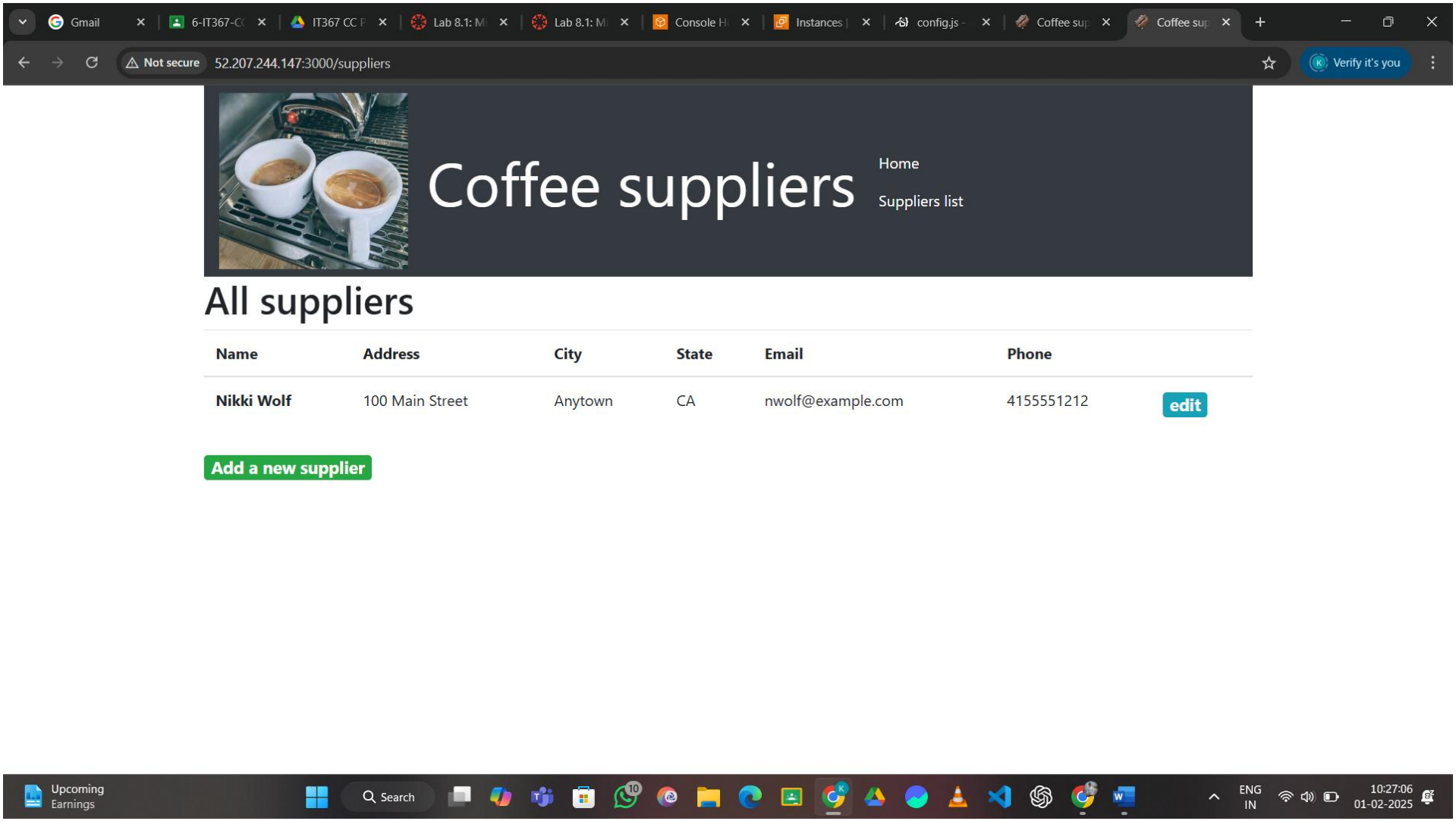

Figure 13: Link a backend in docker file and show all suppliers

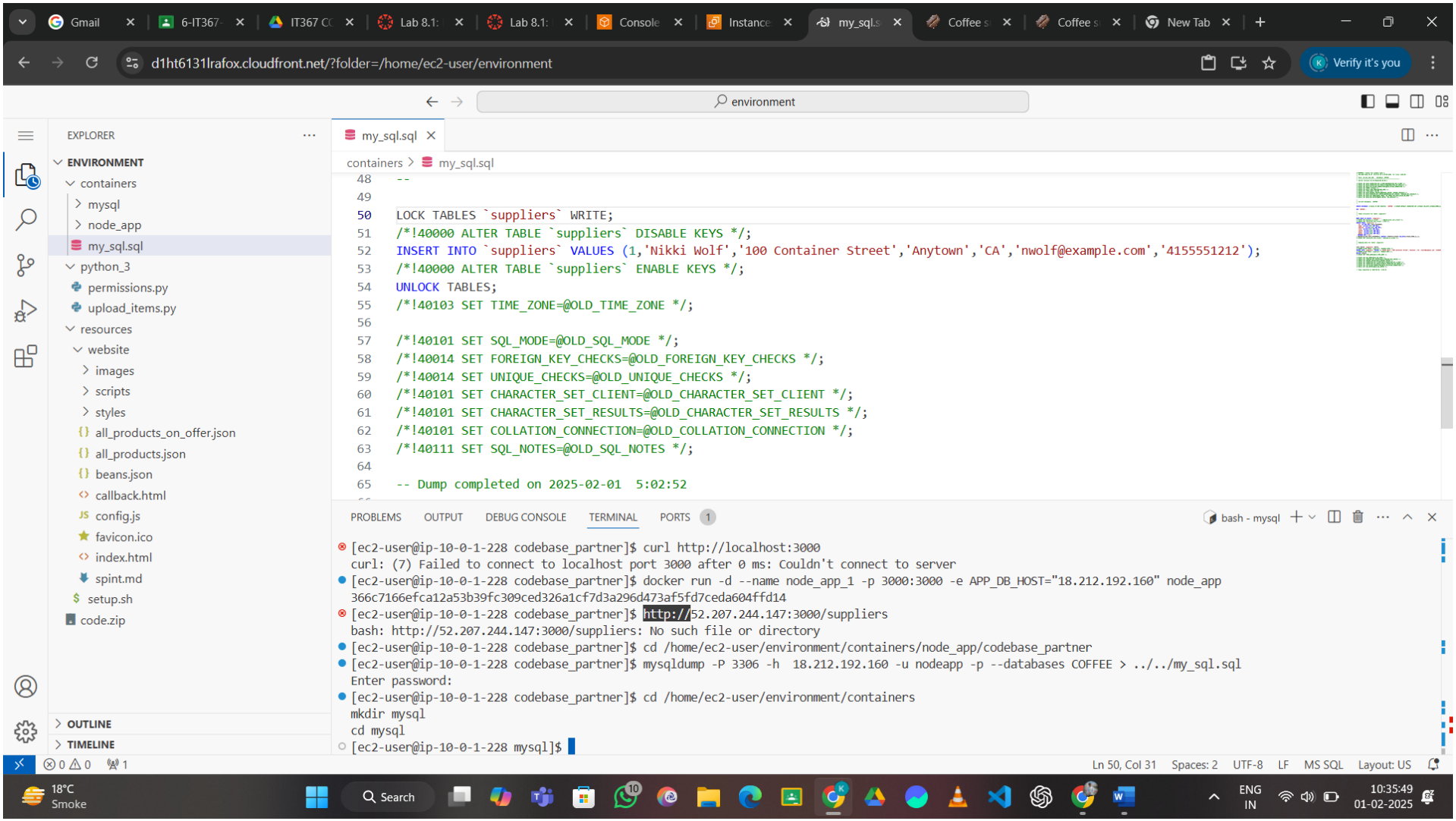

Figure 14: Edit address of supplier main to container

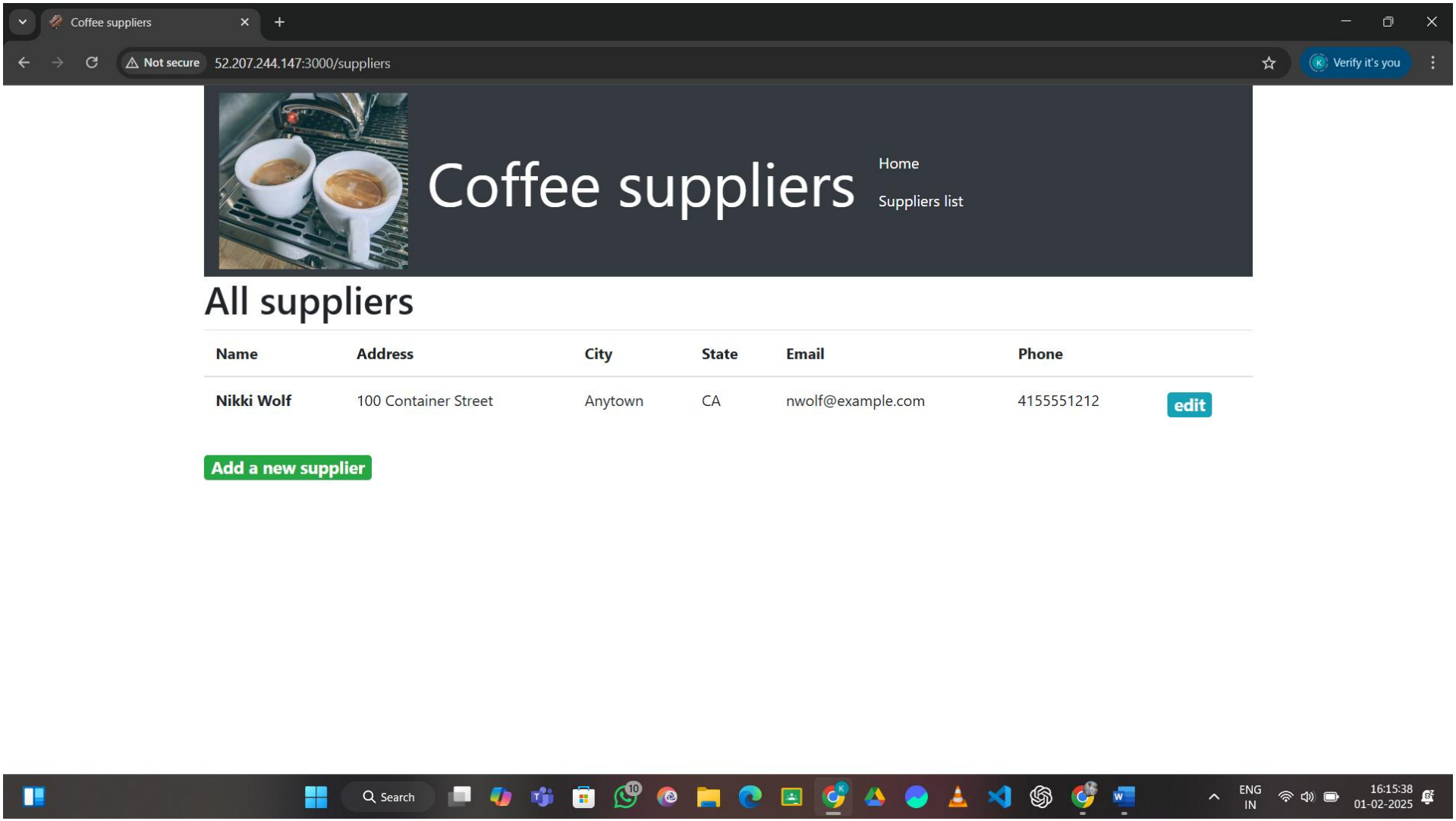

Figure 15: Updated address in browser (frontend)

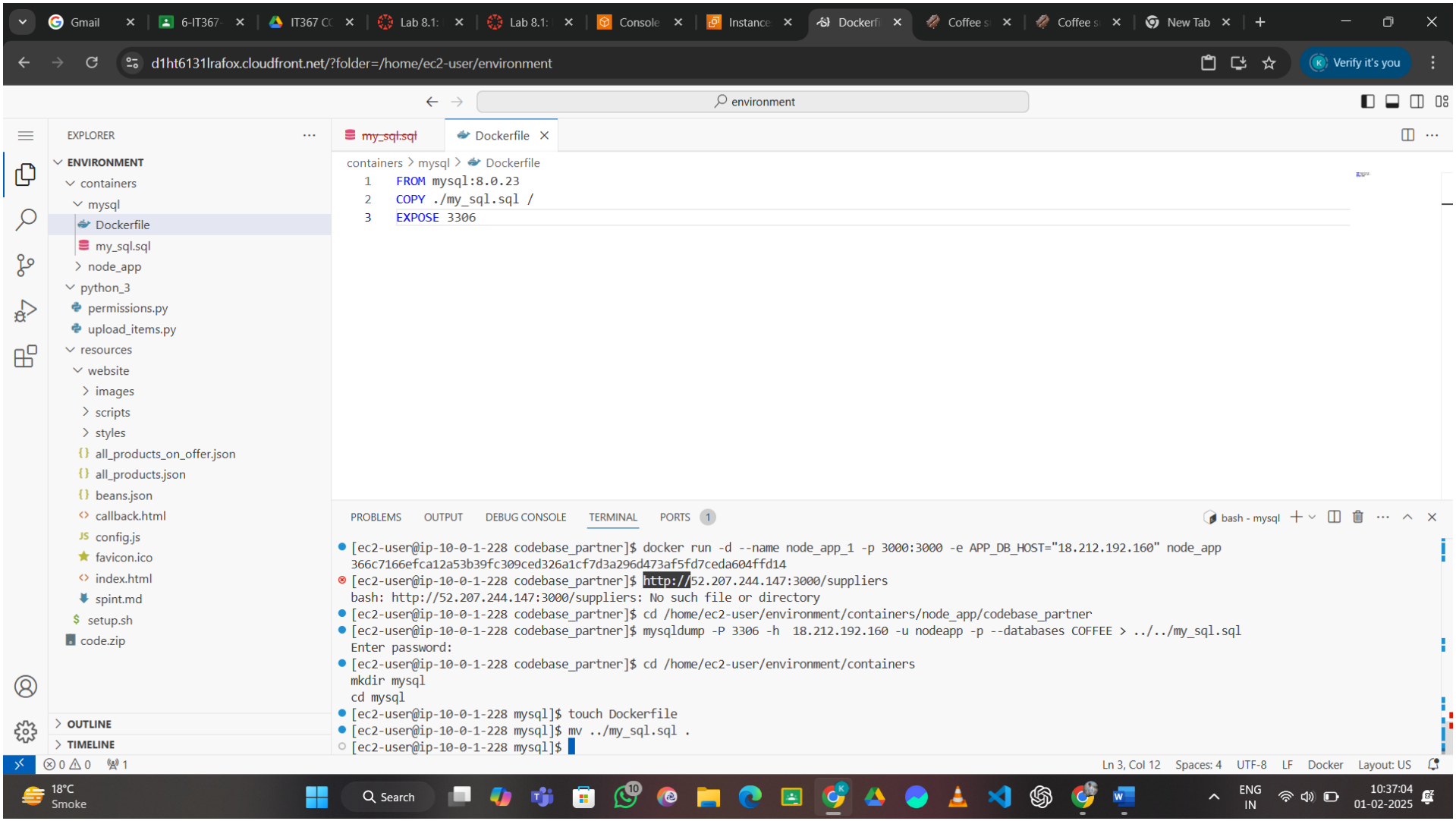

Figure 16: Create a docker file for backend

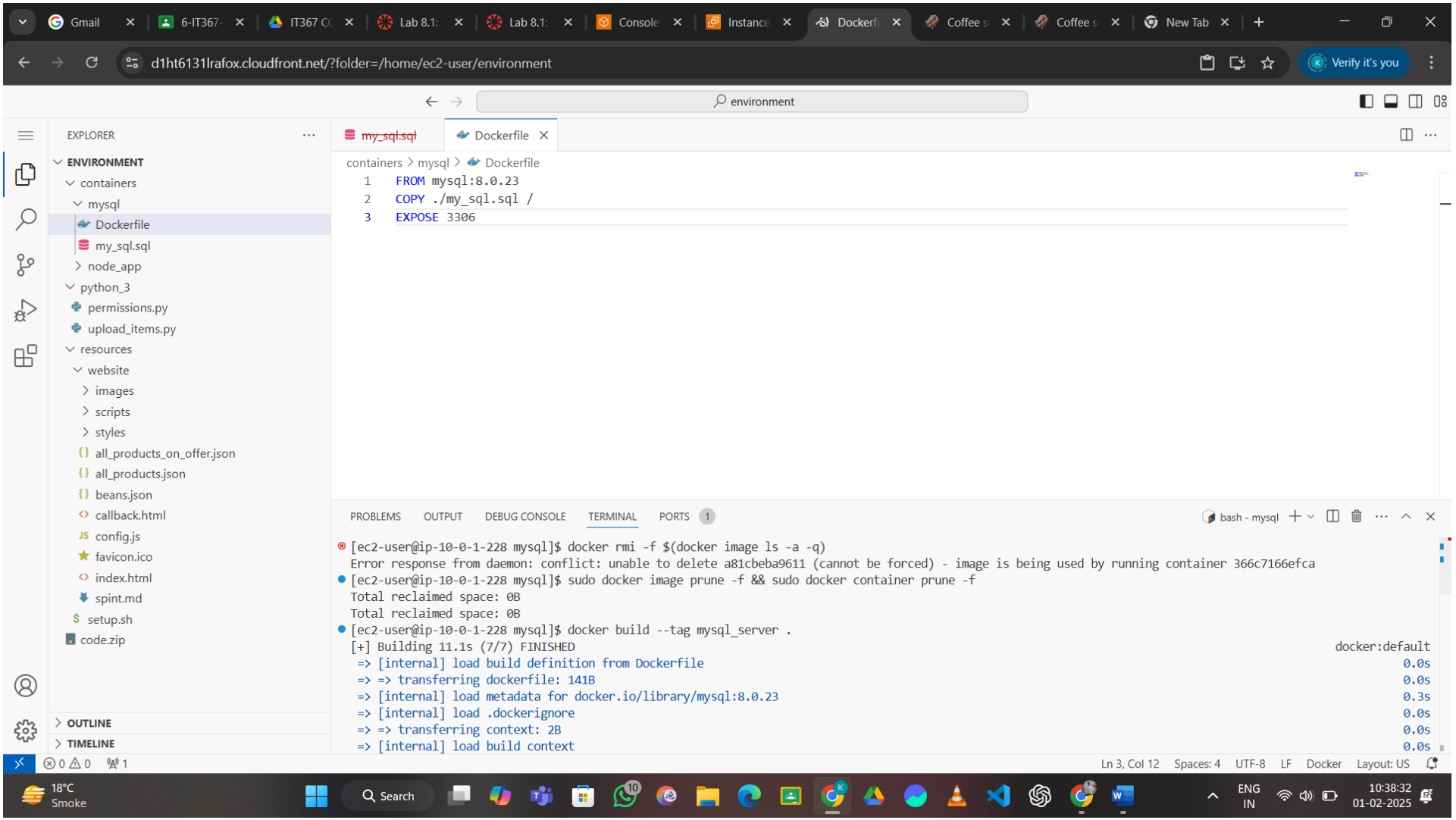

Figure 17: pushed backend docker file

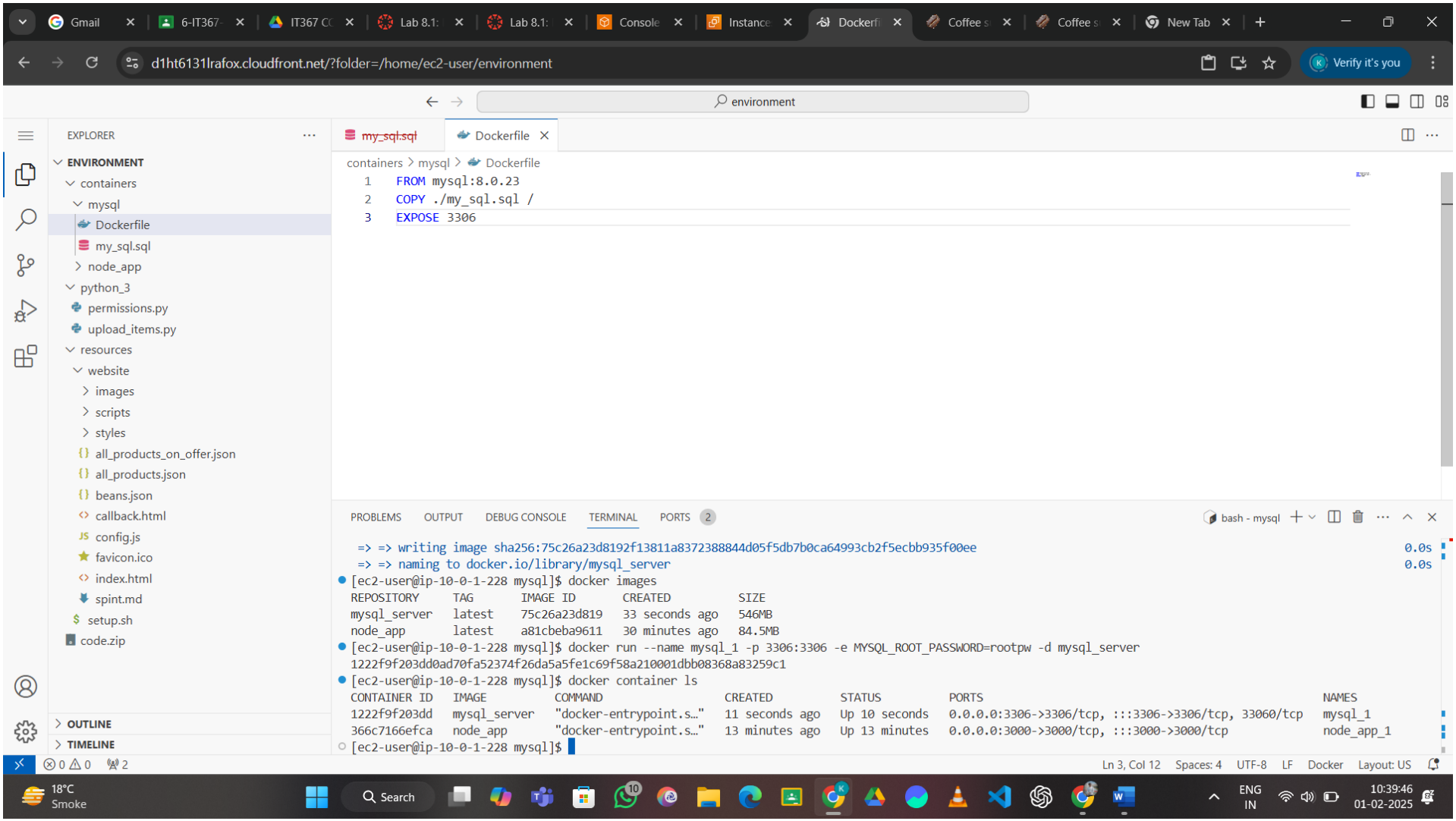

Figure 18: Check all docker file which are created

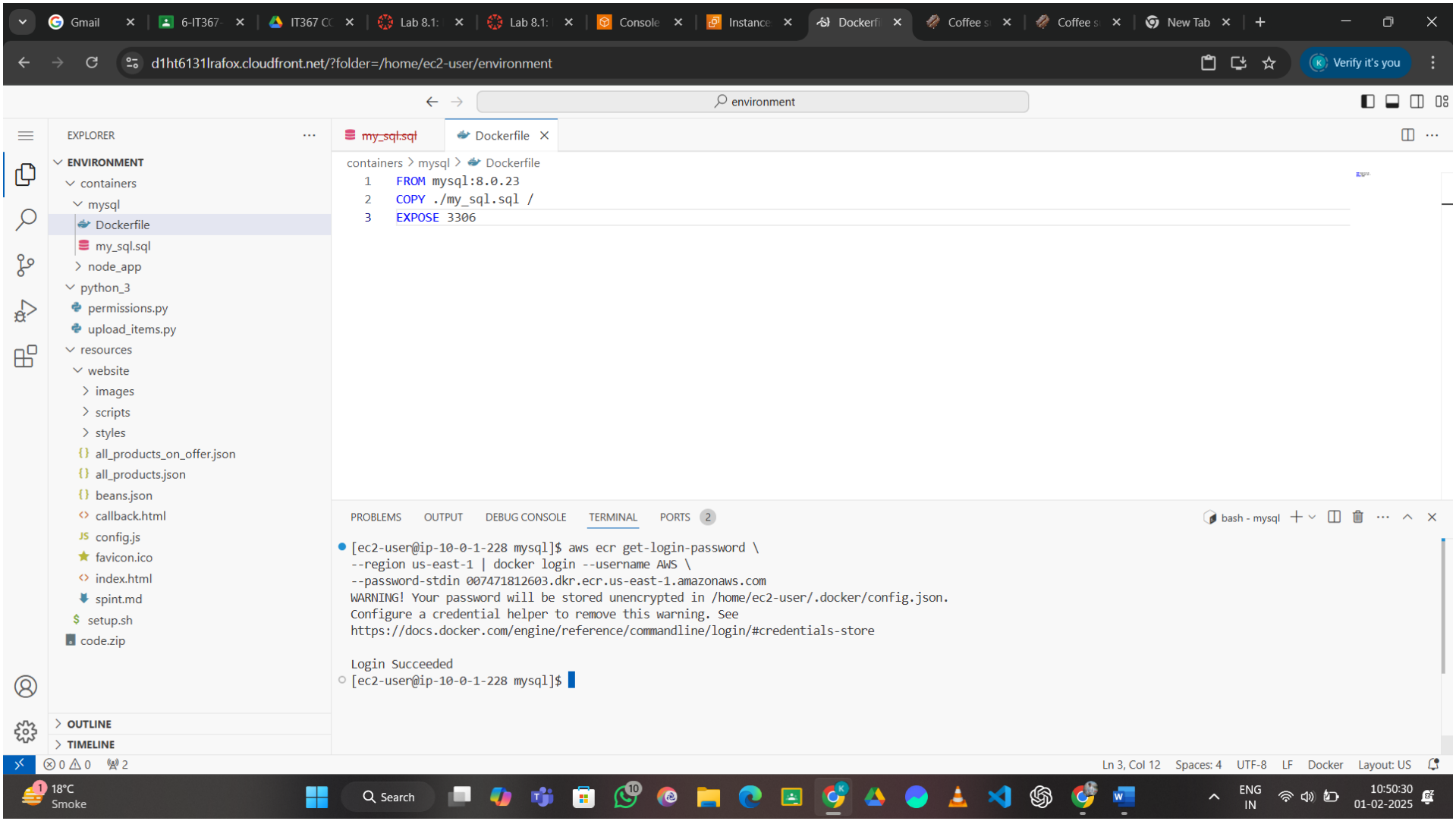

Figure 19: Link a aws account for pushed docker file

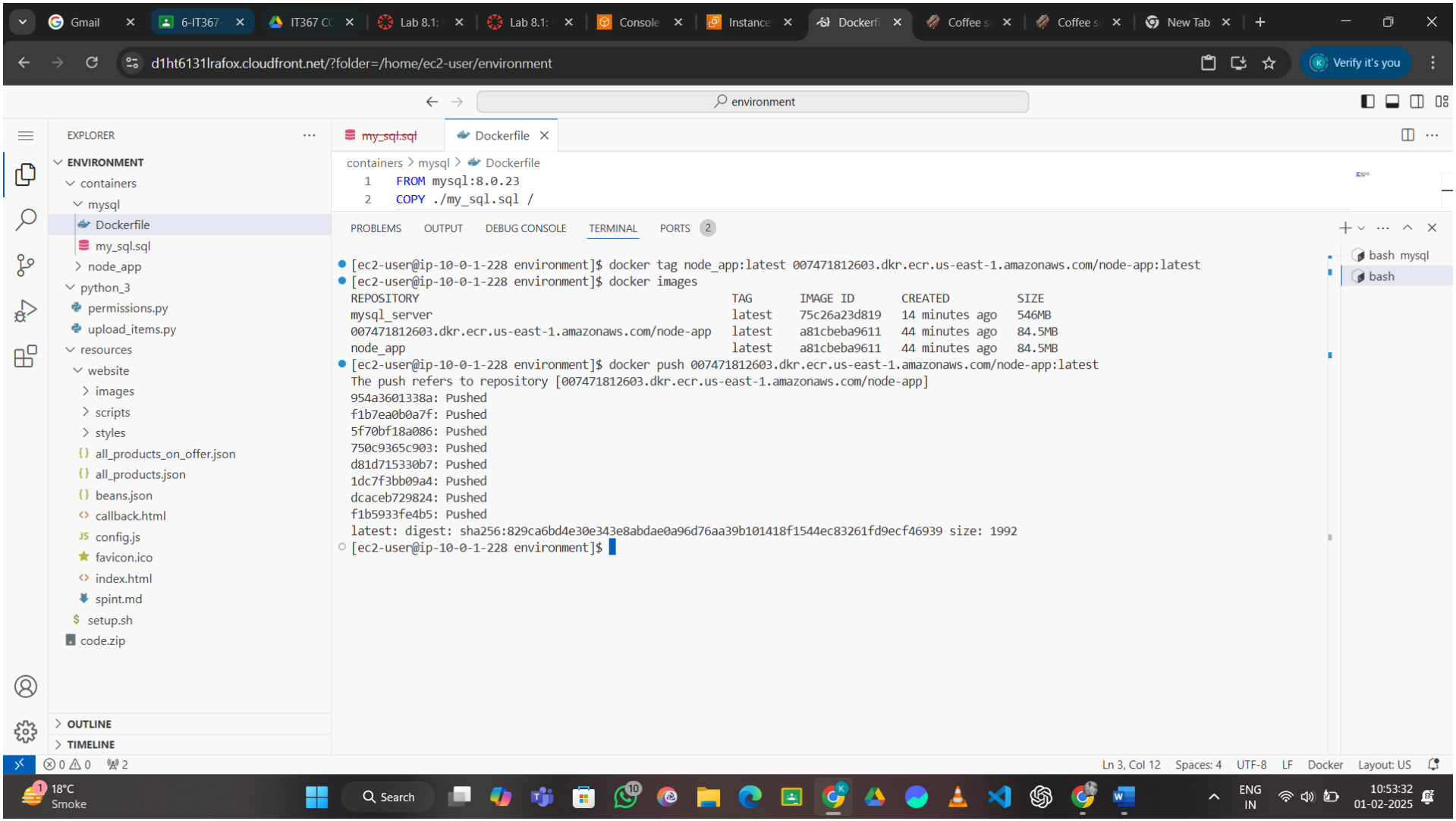

Figure 20: Pushed all docker file

LATEST APPLICATIONS:

1. Generative AI
2. Edge Computing
3. Decentralized Finance (DeFi)
4. Metaverse Platforms
5. Autonomous Vehicles
6. Wearable Health Devices
7. 5G-Enabled Solutions
8. Smart Home Automation
9. Cybersecurity Platforms
10. Green Tech Applications

LEARNING OUTCOME:

Through this practical, I will gain hands-on experience in containerizing applications using Docker and deploying them on AWS. I will learn to create Dockerfiles, build and run Docker images, and manage containers effectively. Additionally, I will practice setting up and using Amazon Elastic Container Registry (ECR) to securely store and deploy container images, enhancing my ability to work with cloud-based infrastructures.

REFERENCE:

1. https://awsacademy.instructure.com/courses/104050/