IaC Provisioning for Finance System

Course Name: DevOps Foundation

Institution Name: Medicaps University-Datagami Skill Based Course

Sr no	Student Name	Enrolment Number
1.	Tarang Purohit	EN22CS3011033
2.	Suhani Maheshwari	EN22CS301988
3.	Smita Kumawat	EN22CS301961
4.	Sneha Agrawal	EN22CS301963
5.	Somya Neema	EN22CS301973

Group Name: Group 03D9

Project Number: DO-03

Industry Mentor Name: Mr. Vaibhav Sir

University Mentor Name: Dr. Ritesh Joshi

Academic Year: 2025-26

Problem Statement & Objectives

1. **Problem Statement:** IaC Provisioning for Finance System

2. **Project Objectives:** Automate the provisioning of infrastructure for a Finance system using Infrastructure as Code (IaC) tools like Terraform or Ansible. The project involves script-based deployment of local Docker environments or cloud instances (e.g., AWS), including the automated installation of core dependencies such as Terraform or Ansible. This approach eliminates manual configuration errors and enables repeatable, scalable environment setup.

3. **Scope of the Project:** The scope of this project is to design and implement an automated deployment system for a Finance application using Infrastructure as Code (IaC) and modern DevOps practices. The project involves developing a Finance web application using the Flask framework and containerizing it with Docker to ensure a consistent and portable runtime environment across local and cloud platforms.

   The infrastructure required for hosting the application is provisioned automatically on Amazon Web Services (AWS) using Terraform. Terraform scripts define and create cloud resources such as Virtual Private Cloud (VPC), public subnet, internet gateway, security group, and EC2 instance, enabling repeatable and error-free infrastructure setup through code rather than manual configuration.

   The project also includes automated application deployment using EC2 user-data scripts, which install Docker, pull the application image from Docker Hub, and run the container during instance initialization. Continuous Integration and Continuous Deployment (CI/CD) is implemented using GitHub Actions, allowing automatic building, testing, and deployment of the application whenever code changes are pushed to the repository.

Additionally, the project scope extends to future scalability and architectural improvements as part of High-Level Design (HLD) expansion, including load balancing, auto scaling, improved security practices, and highly available cloud deployment. The overall system demonstrates an efficient, scalable, and reliable automated deployment workflow suitable for modern cloud-based applications.

Proposed Solution

1. Key Features:

- Automated infrastructure provisioning using Infrastructure as Code (IaC)
- Finance web application deployment using Docker containers
- AWS cloud infrastructure setup using Terraform
- Automatic application deployment through EC2 user-data scripts
- CI/CD pipeline for automated build and deployment
- Reusable and scalable infrastructure modules
- Reduced manual configuration and deployment errors

2. Overall Architecture:

The proposed solution follows an automated DevOps architecture for deploying a Finance application using containerization, Infrastructure as Code (IaC), and CI/CD automation. The system ensures fast, consistent, and error-free deployment by eliminating manual configuration.

1. Finance Application

A lightweight Finance web application is developed using Python Flask. It provides basic endpoints such as balance and transaction details and serves as the main application deployed in the system.

2. Containerization using Docker

The application is packaged into a Docker container to maintain a consistent runtime environment. Docker installs dependencies, exposes port 5000, and allows the application to run identically on local and cloud systems.

3. Infrastructure Provisioning using Terraform

Terraform automates AWS infrastructure creation using code. It provisions:

- VPC
- Public Subnet
- Internet Gateway
- Route Table
- Security Group
- EC2 Instance

This enables repeatable and scalable infrastructure deployment.

4. Automatic Deployment via EC2 User-Data

During EC2 creation, a user-data script automatically installs Docker, pulls the application image from Docker Hub, and runs the container. This removes the need for manual server setup.

5. CI/CD Automation with GitHub Actions

GitHub Actions automates the deployment pipeline. On every code push:

- Docker image is built
- Image is pushed to Docker Hub
- Terraform applies infrastructure updates

This ensures continuous integration and deployment.

6. Workflow (End-to-End Process)

Developer → Git Push → GitHub Actions → Docker Build & Push → Terraform Apply → AWS Infrastructure Creation → EC2 Auto Setup → Application Deployment

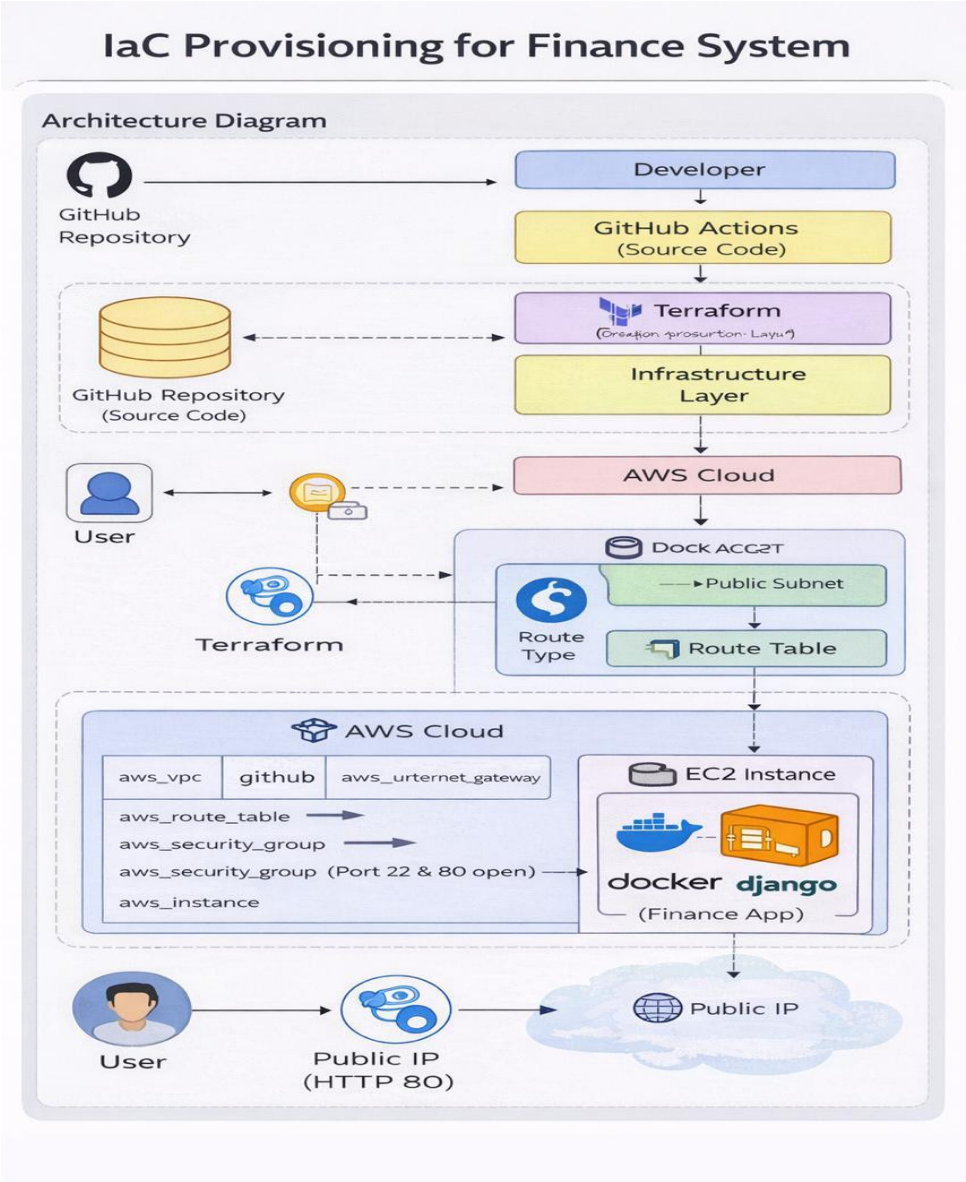

Fig. 1

Results & Output

1. Accesible through Public IP

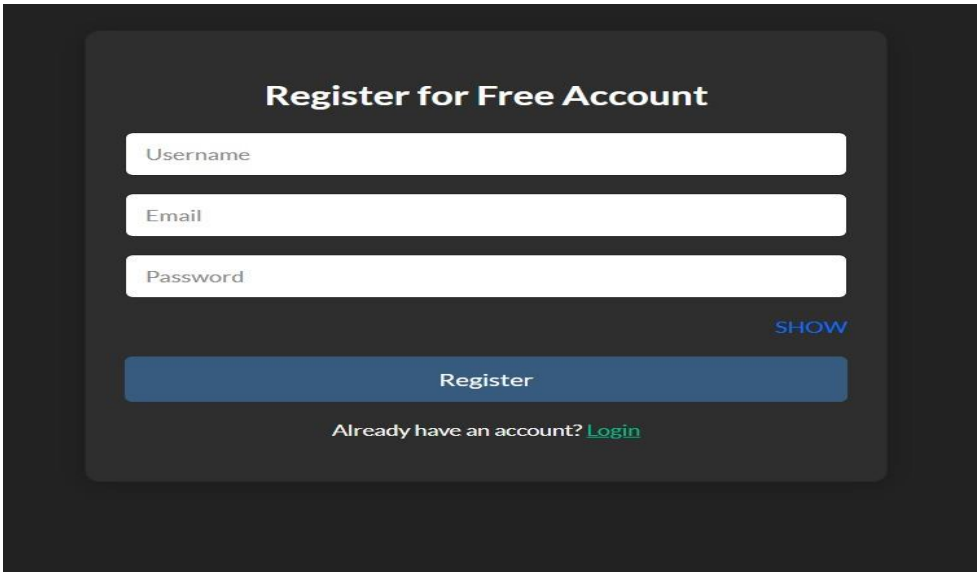

Fig. 1.1

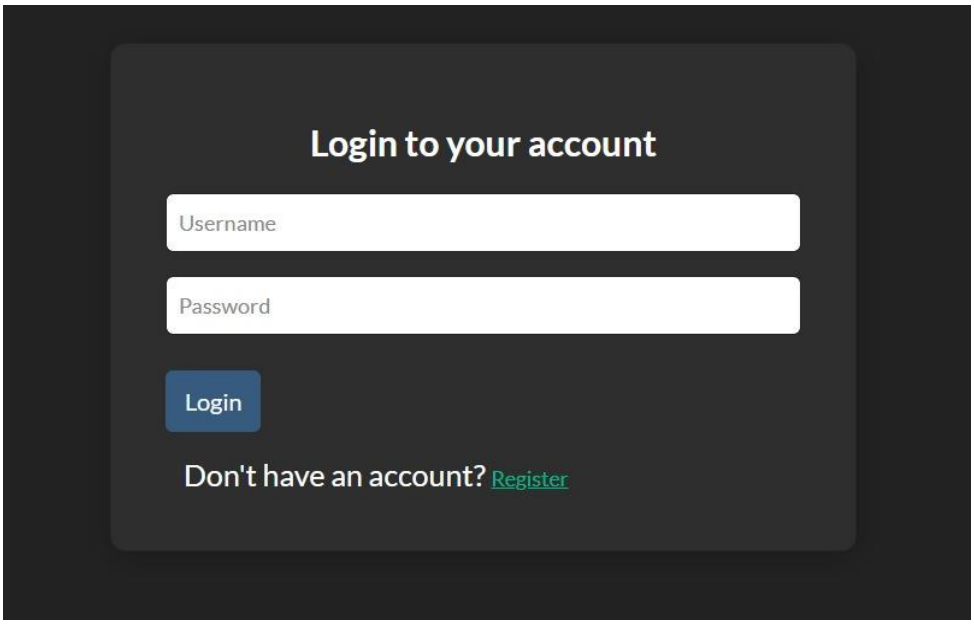

Fig. 1.2

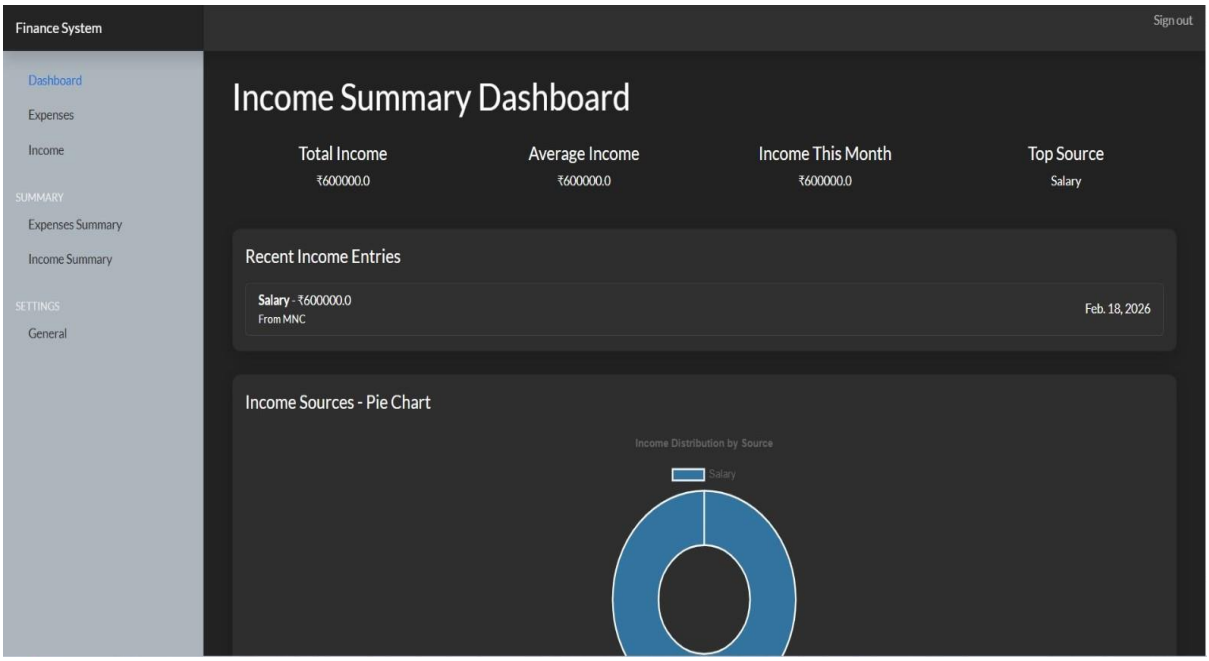
Finance System
Sign out
Dashboard
Expenses
Income
SUMMARY
Expenses Summary
Income Summary
SETTINGS
General
Income Summary Dashboard
Total Income
₹600000.0
Average Income
₹600000.0
Income This Month
₹600000.0
Top Source
Salary
Recent Income Entries
Salary - ₹600000.0
From MNC
Feb. 18, 2026
Income Sources - Pie Chart
Income Distribution by Source
Salary

Fig. 1.3

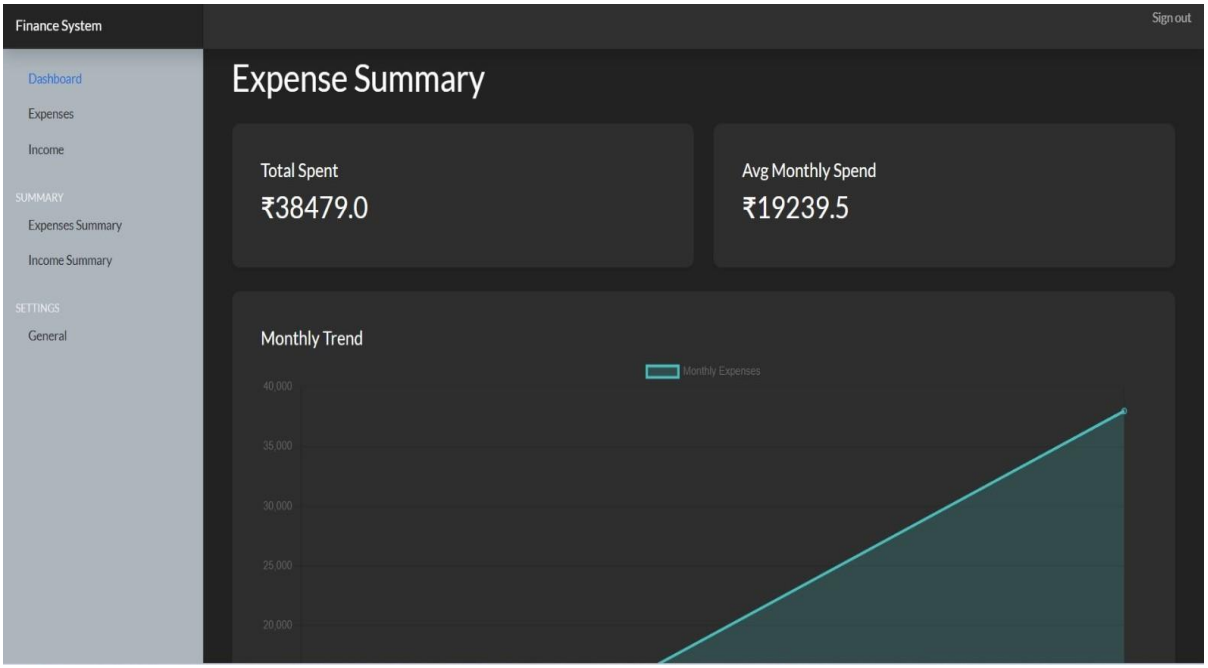
Finance System
Sign out
Dashboard
Expenses
Income
SUMMARY
Expenses Summary
Income Summary
SETTINGS
General
Expense Summary
Total Spent
₹38479.0
Avg Monthly Spend
₹19239.5
Monthly Trend
Monthly Expenses
40,000
35,000
30,000
25,000
20,000

Fig. 1.4

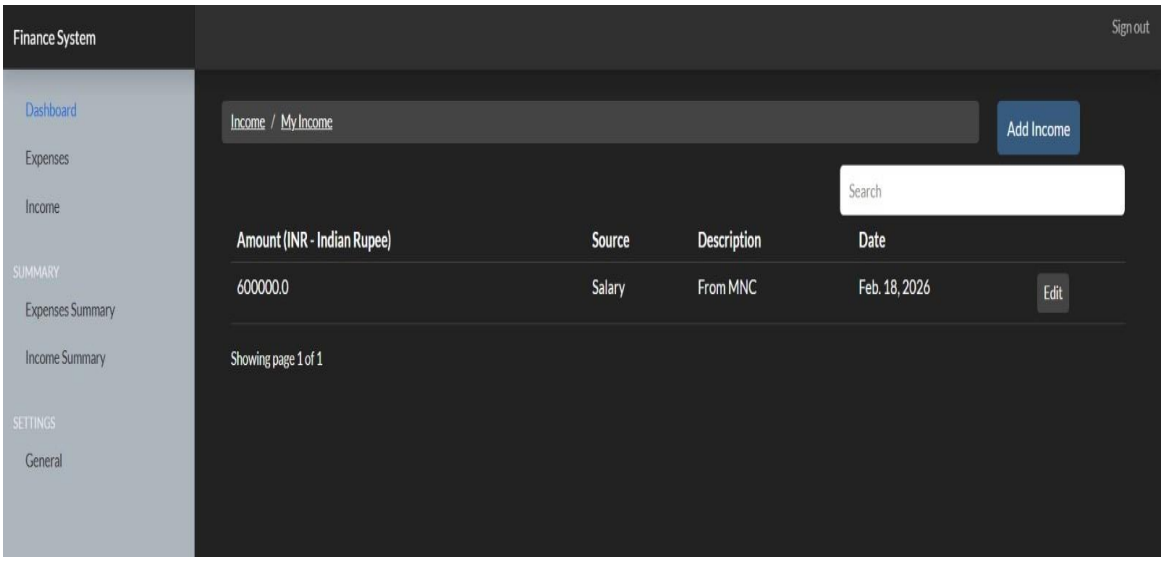
Finance System
Sign out
Dashboard
Expenses
Income
SUMMARY
Expenses Summary
Income Summary
SETTINGS
General
Income / My Income
Add Income
Search
Amount (INR - Indian Rupee)
Source
Description
Date
600000.0
Salary
From MNC
Feb. 18, 2026
Edit
Showing page 1 of 1

Fig. 1.5

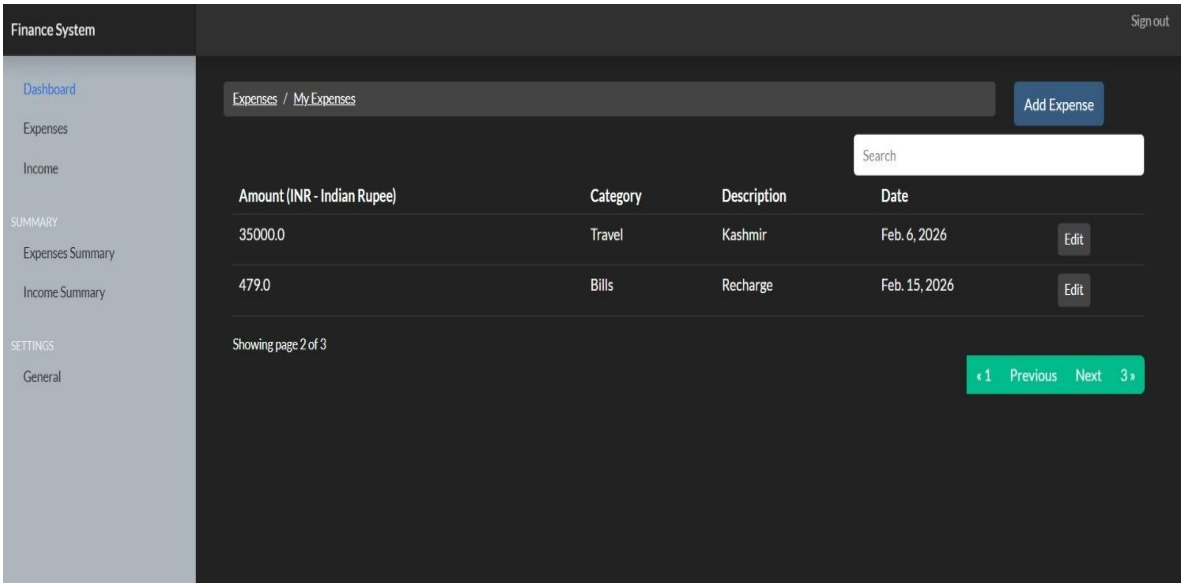
Finance System
Sign out
Dashboard
Expenses
Income
SUMMARY
Expenses Summary
Income Summary
SETTINGS
General
Expenses / My Expenses
Add Expense
Search
Amount (INR - Indian Rupee)
Category
Description
Date
35000.0
Travel
Kashmir
Feb. 6, 2026
Edit
479.0
Bills
Recharge
Feb. 15, 2026
Edit
Showing page 2 of 3
« 1 Previous Next 3 »

Fig. 1.6

2. Containerization using Docker

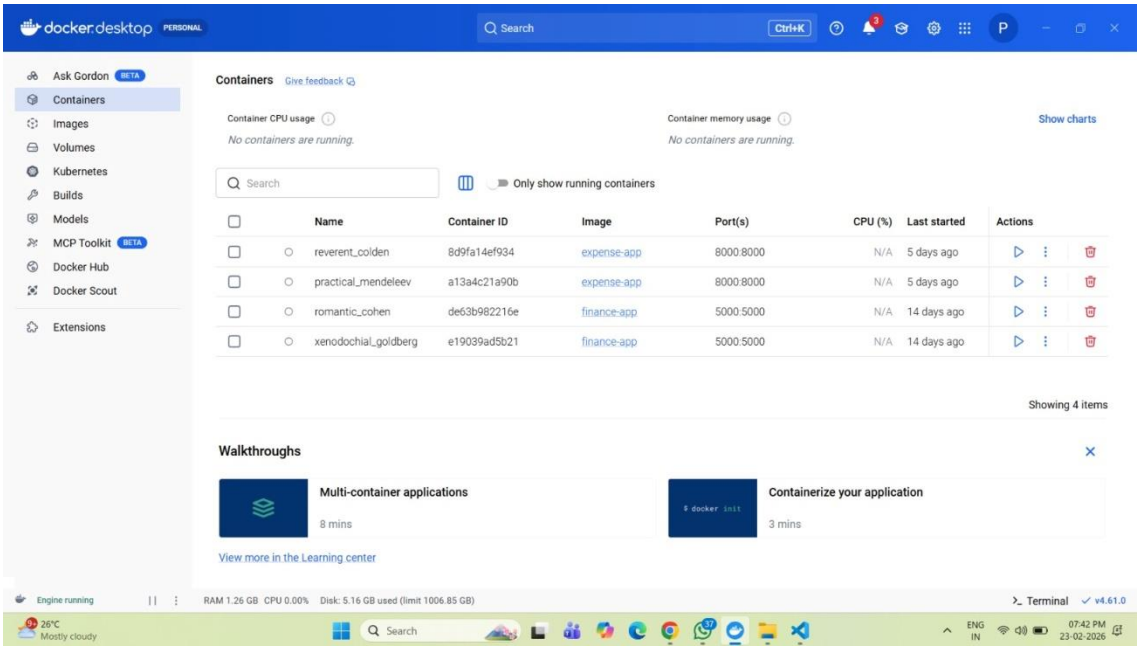

Fig. 2.1

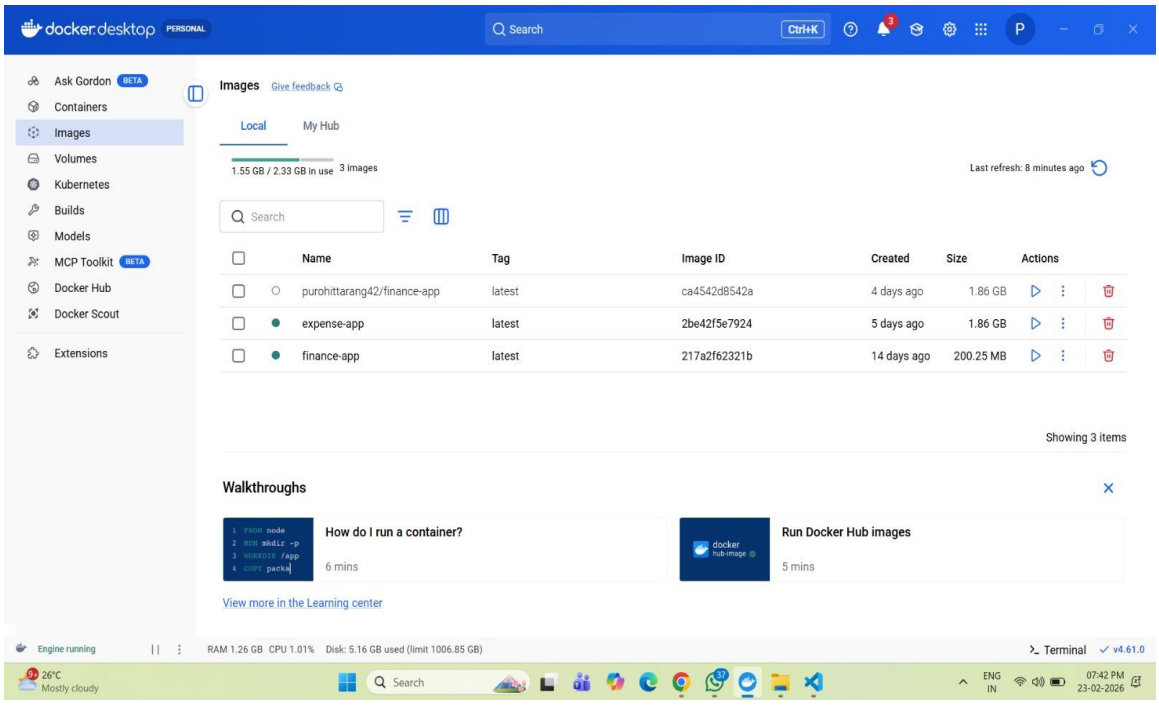

Fig. 2.2

3. Resource Provisioning through Terraform

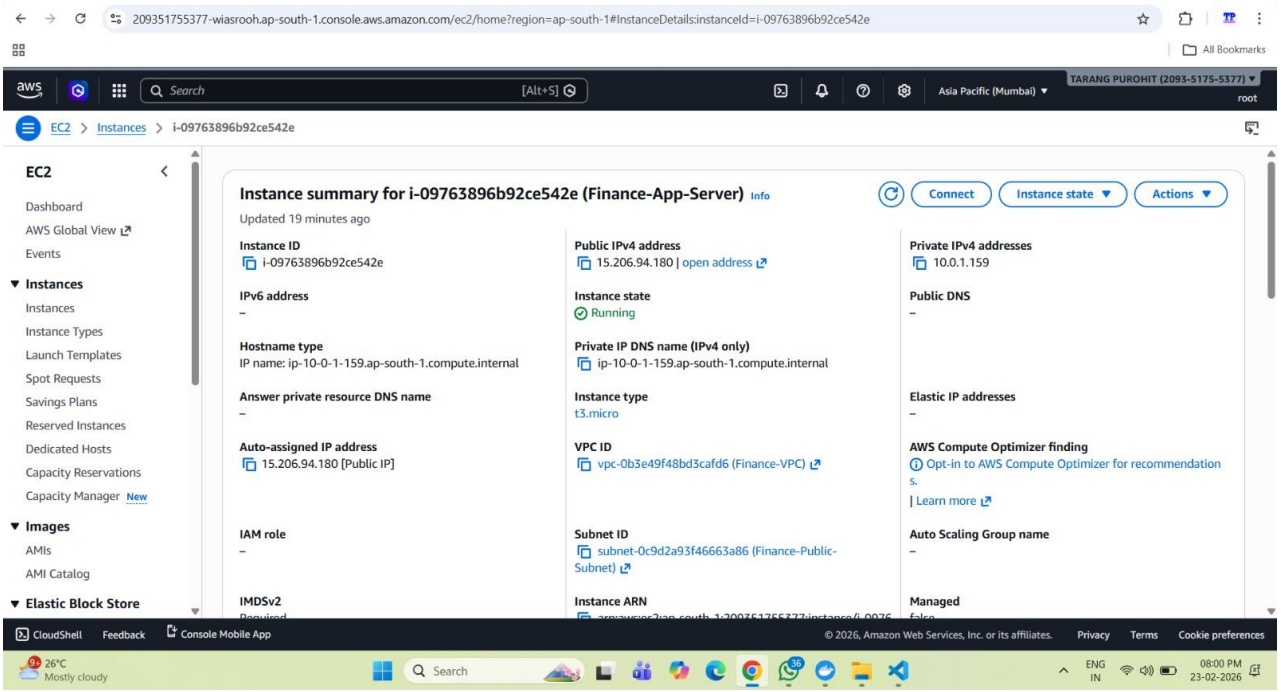

Fig. 3.1

Conclusion

This project successfully demonstrates the automation of infrastructure provisioning and application deployment for a Finance system using Infrastructure as Code (IaC). By integrating Docker, Terraform, AWS, and CI/CD automation, the project eliminates manual configuration processes and ensures consistent, repeatable, and reliable deployments. The automated workflow enables faster environment setup, reduces human errors, and improves deployment efficiency. Overall, the solution showcases modern DevOps practices for building scalable and maintainable cloud-based applications.

Future Scope & Enhancements

- **Implementation of Application Load Balancer (ALB):**
  An ALB can be added to distribute incoming traffic across multiple application instances, improving performance, availability, and fault tolerance.
- **Integration of Auto Scaling Groups:**
  Auto Scaling can automatically increase or decrease EC2 instances based on workload demand, ensuring efficient resource utilization and system scalability.
- **Migration to Amazon RDS:**
  The application database can be moved to Amazon RDS to provide managed, secure, and persistent database storage with automated backups and maintenance.
- **Use of S3 Remote Backend for Terraform:**
  Terraform state files can be stored in Amazon S3 to enable secure state management, version control, and collaboration among team members.
- **Implementation of IAM Role-Based Access Control:**
  IAM roles and policies can be used to restrict permissions and enhance security by granting least-privilege access to AWS resources.
- **Deployment Across Multi-AZ Architecture:**
  Deploying resources across multiple Availability Zones will improve system reliability and ensure high availability during failures.
- **Monitoring and Logging using AWS CloudWatch:**
  AWS CloudWatch can be integrated for real-time monitoring, logging, and alerting to track application performance and detect issues proactively.

.