Dockerized Healthcare Python Flask Service Report

Course Name: DevOps Fundamentals

Institution Name: Medicaps University – Datagami Skill Based Course

Student Name(s) & Enrolment Number(s):

Sr no	Student Name	Enrolment Number
01.	Vanshika Vyas	EN22CS3011062
02.	Tulsi Chouhan	EN22CS3011038
03.	Swayam Mishra	EN22CS3011009
04.	Ujjawal Goswami	EN22CS3011043
05.	Mukund Mishra	EN22ME304046

Group Name:02D9

Project Number:DO-02

Industry Mentor Name:Mr. Vaibhav

University Mentor Name:Dr. Ritesh Joshi

Academic Year:2025-26

Problem Statement and Objectives

1. **Problem Statement:** Traditional deployment of a Python Flask-based Healthcare Management System can lead to environment inconsistencies, dependency conflicts, and deployment challenges across local and cloud platforms.

   The problem is to containerize the healthcare application using Docker and Docker Compose to ensure portability, consistency, security, and reliable deployment on an AWS EC2 instance, while maintaining optimized image size and persistent data storage.

2. **Project Objectives:**
   - To containerize the Python Flask Healthcare application using Docker.
   - To implement a multi-stage Docker build for optimized and secure images.
   - To use Docker Compose for simplified orchestration and management.
   - To ensure data persistence using Docker volumes with SQLite.
   - To deploy the application on an AWS EC2 Linux instance.
   - To achieve portability, consistency, and reliability across environments.

3. **Scope Of the Project:** The scope of the Dockerized Healthcare Python Flask Service project includes:
   - Development of a basic Healthcare Management System using Python Flask and SQLite.
   - Containerization of the application using multi-stage Docker builds following industry best practices.
   - Orchestration of the application using Docker Compose.
   - Deployment of the containerized application on an AWS EC2 Linux instance.
   - Implementation of persistent storage using Docker volumes.
   - Ensuring security, portability, and reliability across local and cloud environments.

- The project focuses on containerization and cloud deployment and does not include CI/CD pipeline implementation or Kubernetes-based orchestration.

Proposed Solution

1. **Key Features:**
    - Docker-based containerization
    - Multi-stage optimized Docker build
    - Docker Compose orchestration
    - Three-tier architecture (Frontend, Backend, Database)
    - RESTful API for patient management (CRUD operations)
    - SQLite database with Docker volume persistence
    - Deployment on AWS EC2
    - Secure access using AWS Security Groups and SSH keys
    - Portable and environment-independent setup
    - Lightweight and easy-to-deploy architecture

2. **Overall Architecture/Workflow:** The system follows a container-based three-tier architecture deployed on an AWS EC2 instance.

    Architectural Flow:

    User Browser
    ↓
    Internet
    ↓
    AWS EC2 Linux Instance
    ↓
    Docker Engine
    ↓
    Flask Application (Docker Container – Port 9000)
    ↓
    SQLite Database (Docker Volume)

Workflow:

- User accesses the application via EC2 Public IP.
- Frontend sends an HTTP request to the Flask backend.
- Flask processes the request and interacts with SQLite.
- Database performs CRUD operations.
- Response is returned and displayed to the user.

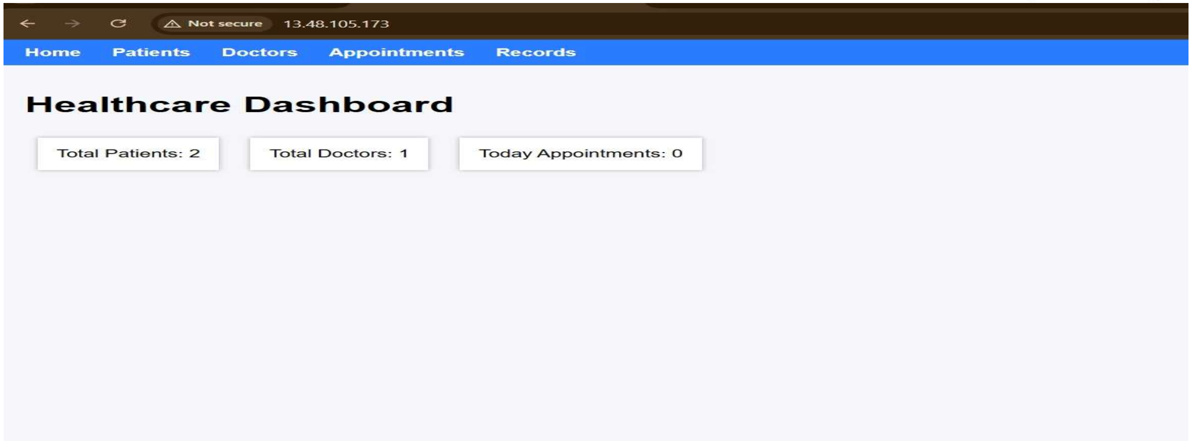
The architecture ensures portability, consistency, and secure cloud deployment.

3. Tools and Technologies Used:

- **Programming Language:** Python
- **Framework:** Flask
- **Frontend:** HTML,CSS
- **Database:** SQLite
- **Containerization:** Docker(Multi Stage-Build)
- **Orchestration:** Docker Compose
- **Cloud Platform:** AWS EC2
- **Infrastructure Provisioning:** Terraform
- **Version Control:** Git
- **Operating System:** Linux(EC2 Instance)

Results & Output

1.Screenshots/Output:

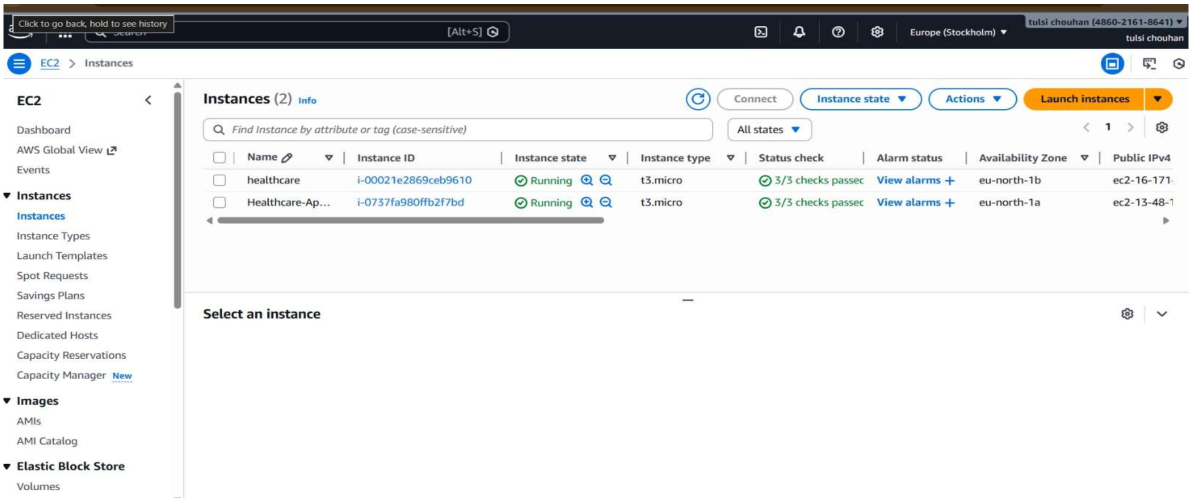


Healthcare Dashboard

Total Patients: 2

Total Doctors: 1

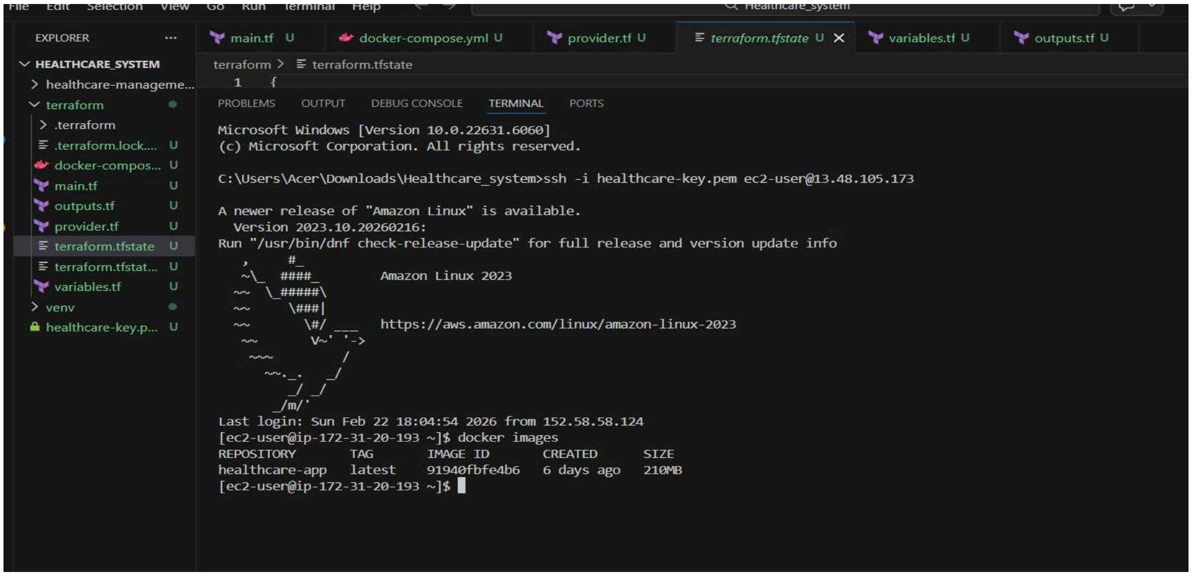
Today Appointments: 0



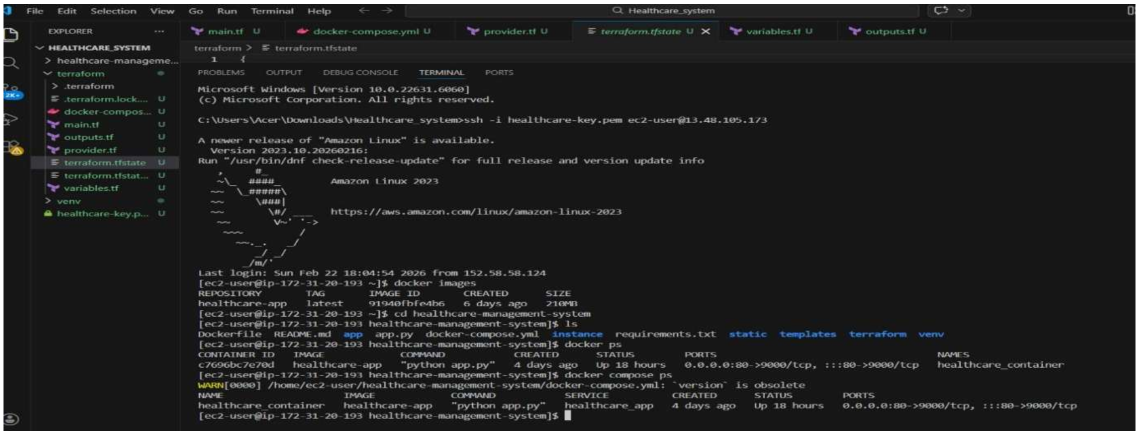
Instances (2) Info

Find Instance by attribute or tag (case-sensitive)

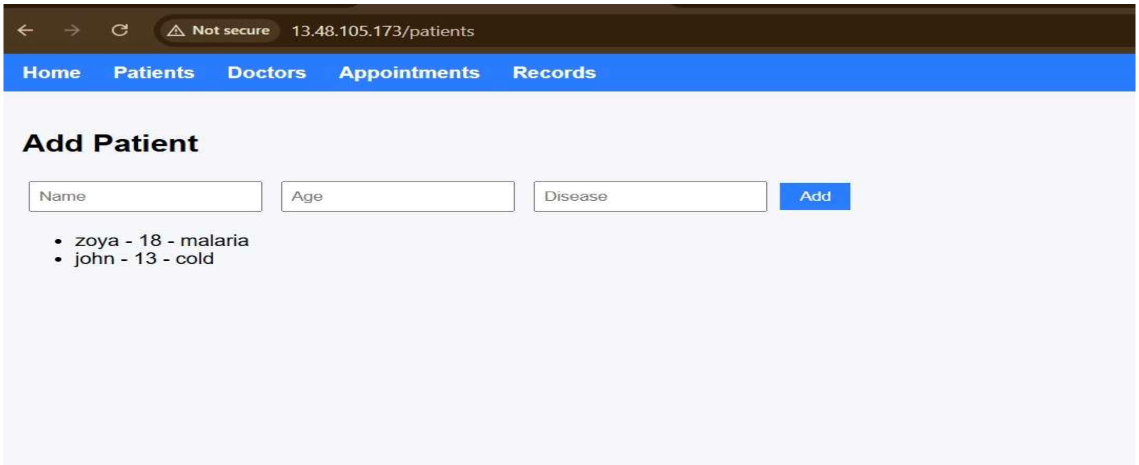
□	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
□	healthcare	i-00021e2869ceb9610	✓ Running	t3.micro	✓ 3/3 checks passec	View alarms +	eu-north-1b	ec2-16-171-
□	Healthcare-Ap...	i-0737fa980ffb2f7bd	✓ Running	t3.micro	✓ 3/3 checks passec	View alarms +	eu-north-1a	ec2-13-48-1



```
Microsoft Windows [Version 10.0.22631.6060]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Acer\Downloads\Healthcare_system>ssh -i healthcare-key.pem ec2-user@13.48.105.173

A newer release of "Amazon Linux" is available.
  Version 2023.10.20260216:
Run "/usr/bin/dnf check-release-update" for full release and version update info
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Sun Feb 22 18:04:54 2026 from 152.58.58.124
[ec2-user@ip-172-31-20-193 ~]$ docker images
REPOSITORY       TAG       IMAGE ID       CREATED      SIZE
healthcare-app   latest    91940fbfe4b6   6 days ago   210MB
[ec2-user@ip-172-31-20-193 ~]$ █
```

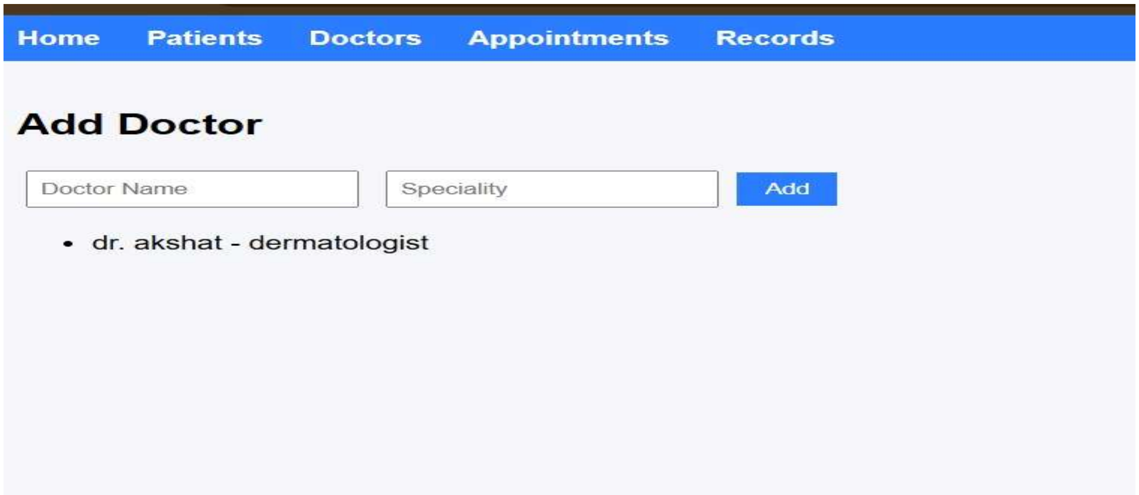


```
Microsoft Windows [Version 10.0.22631.6060]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Acer\Downloads\Healthcare_system>ssh -i healthcare-key.pem ec2-user@13.48.105.173

A newer release of "Amazon Linux" is available.
  Version 2023.10.20260216:
Run "/usr/bin/dnf check-release-update" for full release and version update info
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Sun Feb 22 18:04:54 2026 from 152.58.58.124
[ec2-user@ip-172-31-20-193 ~]$ docker images
REPOSITORY        TAG       IMAGE ID       CREATED      SIZE
healthcare-app    latest    91940fbfe4b6   6 days ago   210MB
[ec2-user@ip-172-31-20-193 ~]$ cd healthcare-management-system
[ec2-user@ip-172-31-20-193 healthcare-management-system]$ ls
Dockerfile  README.md  app  app.py  docker-compose.yml  instance  requirements.txt  static  templates  terraform  venv
[ec2-user@ip-172-31-20-193 healthcare-management-system]$ docker ps
CONTAINER ID   IMAGE            COMMAND           CREATED      STATUS          PORTS                                    NAMES
c7696bc7e70d   healthcare-app   "python app.py"   4 days ago   Up 18 hours     0.0.0.0:80->9000/tcp, :::80->9000/tcp   healthcare_container
[ec2-user@ip-172-31-20-193 healthcare-management-system]$ docker compose ps
WARN[0000] /home/ec2-user/healthcare-management-system/docker-compose.yml: `version` is obsolete
NAME                   IMAGE            COMMAND           SERVICE          CREATED      STATUS          PORTS
healthcare_container   healthcare-app   "python app.py"   healthcare_app   4 days ago   Up 18 hours     0.0.0.0:80->9000/tcp, :::80->9000/tcp
[ec2-user@ip-172-31-20-193 healthcare-management-system]$ 
```



Add Patient

- zoya - 18 - malaria
- john - 13 - cold



Add Doctor

- dr. akshat - dermatologist

2. Report/Dashboard/Models:

A. Report:

- Patient records listing (All Patients report)
- Individual patient detail view
- CRUD operation status responses (Add, Update, Delete confirmation)

B. Dashboard:

- Web-based interface displaying patient data
- Simple HTML/CSS frontend
- Displays real-time data fetched from backend APIs

C. Models:

- patient_id (Primary Key)
- name
- age
- disease

3.Key Outcomes:

- Successful containerization of the Healthcare Flask application using Docker
- Implementation of multi-stage Docker build for optimized image size
- Simplified deployment and management using Docker Compose
- Persistent data storage using Docker volumes
- Successful deployment on AWS EC2 cloud environment
- Achieved portability between local and cloud setups

- Improved reliability and environment consistency
- Enhanced understanding of real-world DevOps and containerization practices

Conclusion:

The Dockerized Healthcare Python Flask Service successfully demonstrates a modern container-based deployment approach using Docker and Docker Compose. The project ensures portability, consistency, and reliability by eliminating environment-specific dependency issues.

By deploying the application on AWS EC2 and implementing multi-stage Docker builds, the solution achieves optimized performance, improved security, and efficient resource utilization.

Overall, the project reflects practical DevOps fundamentals, including containerization, cloud deployment, and infrastructure management, making it a scalable and maintainable solution for healthcare application deployment.

Future Scope & Enhancement:

- Migration from SQLite to MySQL or PostgreSQL for better scalability
- Implementation of CI/CD pipeline using Jenkins or GitHub Actions
- Deployment using Kubernetes for container orchestration at scale
- Integration of Redis for caching and performance improvement
- Implementation of authentication and role-based access control
- Enabling HTTPS using SSL/TLS certificates
- Monitoring and logging using tools like Prometheus and Grafana
- Auto-scaling and load balancing for high availability
- Backup and disaster recovery mechanisms
- UI enhancement with modern frontend frameworks (React/Angular)

Problem Statement & Objectives

1. Problem Statement
2. Project Objectives
3. Scope of the Project

Proposed Solutionures *(Just mention key features here no need to go into details)*

1. Overall Architecture / Workflow
2. Tools & Technologies Used (*If applicable)*

Results & Output

Add the below details here:

1. *Screenshots / outputs*
2. *Reports / dashboards / models*
3. *Key outcomes*

Conclusion

Mention a brief conclusion about your project summing up everything you have worked on and the key learning.

Future Scope & Enhancements