Project Report: Deploying and Managing Microservices in a Cloud-Native Environment

**1. Introduction**

This project focuses on migrating monolithic applications to a microservices architecture using Kubernetes to ensure scalability, high availability, and flexibility. The system is designed using Docker for containerization, Kubernetes for orchestration, and other supporting cloud-native tools to simulate a real-world deployment pipeline.

**2. Objectives**

- Containerize and deploy microservices- Set up and manage a Kubernetes cluster- Enable service discovery, scaling, and persistent storage- Create a production-like microservices deployment

**3. Requirements**

Software Requirements:- Ubuntu OS- Docker- Kubernetes CLI (kubectl)- Minikube

Application Components:- User Service (Flask)- Product Service (Flask)- Order Service (Flask)

Cloud Resources (Optional for Production):- AWS EKS (alternative to Minikube)

Configuration Parameters:- Docker Hub Username and Repository- Kubernetes Deployment and Service YAMLs- HPA CPU Threshold- Persistent Volume and Claim Specs

**4. Technology Stack**

- Programming Language: Python (Flask)- Orchestration: Kubernetes (Minikube)- Containerization: Docker- Monitoring: metrics-server- Tools: kubectl, hey/curl

**5. System Architecture Diagram**

The architecture includes the following components:- Dockerized microservices deployed in Kubernetes pods- Kubernetes services (ClusterIP and NodePort)- Horizontal Pod Autoscaler for scaling Product service- Persistent Volumes attached to Order service for data retention

+----------------------+

| End User / CLI |

+----------+-----------+

|

v

+----------+-----------+

| NodePort Service |

| (Exposes Services) |

+----------+-----------+

|

+------------------+------------------+

| | |

v v v

+----------+-----+ +---------+------+ +--------+------+

| User Service | | Product Service| | Order Service |

| (Flask App) | | (Flask App) | | (Flask App) |

+----------+------+ +---------+------+ +--------+------+

| | |

| +--------+--------+ |

| | Horizontal Pod | |

| | Autoscaler | |

| +-----------------+ |

| |

| +-------------+-------------+

| | Persistent Volume & Claim |

| | (for Order Service Data) |

| +---------------------------+

**6. Implementation Steps**

* **Step 1: Microservices Development**   
   • Developed Flask apps for User, Product, and Order services
* **Step 2: Dockerfile Creation**   
   • Used Python base image and installed required libraries   
   • Added application code and exposed port
* **Step 3: Build and Push Images**   
   • Built Docker images and pushed to Docker Hub
* **Step 4: Kubernetes Setup**   
   • Started Minikube and verified cluster status
* **Step 5: Deployments**   
   • Created Kubernetes Deployment YAMLs for all services   
   • Applied deployments using kubectl apply
* **Step 6: Services**   
   • Created ClusterIP and NodePort for internal and external access
* **Step 7: Horizontal Pod Autoscaler**   
   • Installed metrics-server   
   • Configured HPA on Product Service with CPU threshold   
   • Verified scaling using hey to generate load
* **Step 8: Persistent Storage**   
   • Defined Persistent Volume and PVC   
   • Mounted to Order Service and verified persistence across pod restarts

**7. Results**

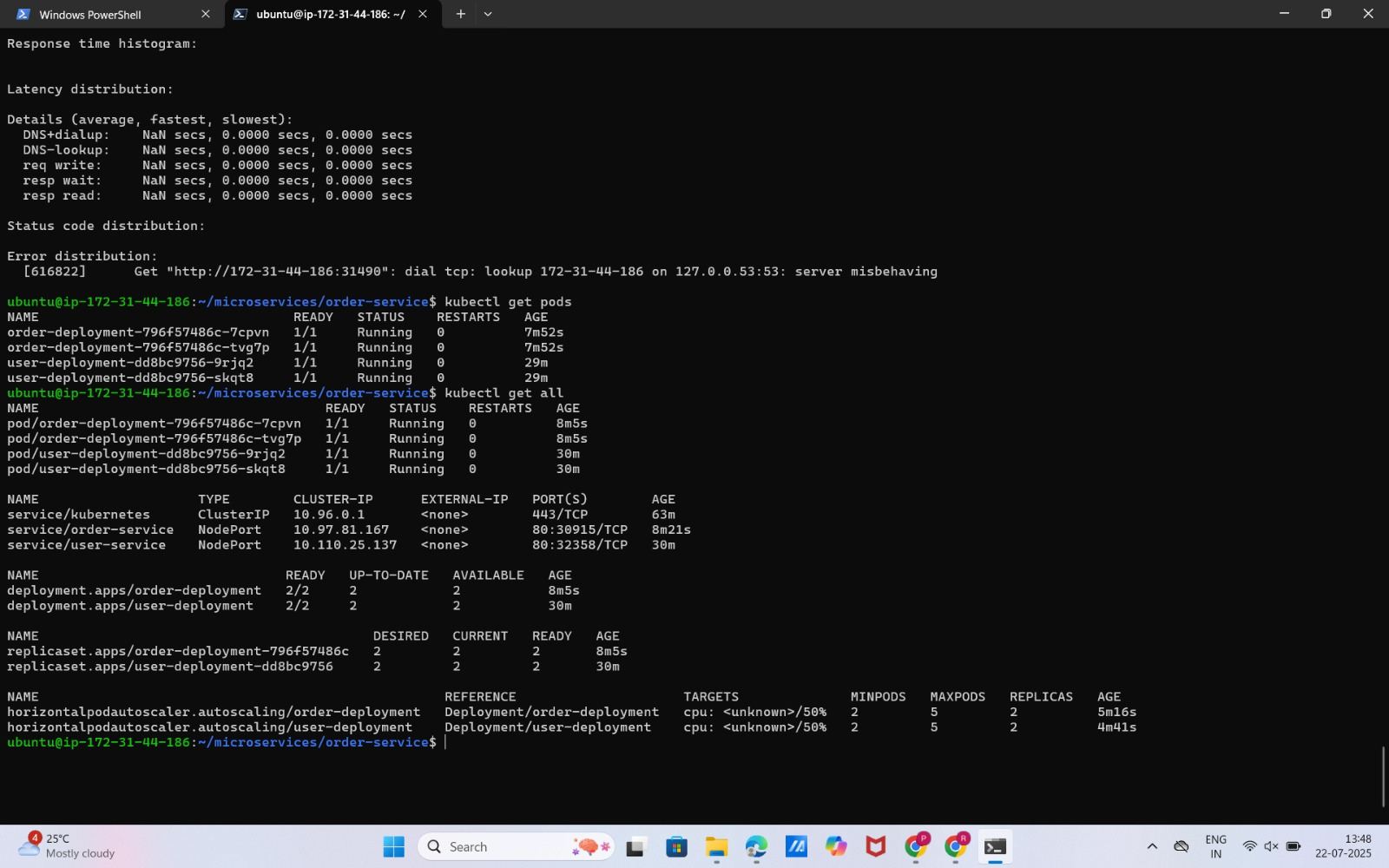
- All microservices deployed and accessible- Product service auto-scaled based on CPU usage- Order service retained data using Persistent Volume- Inter-service communication verified successfully

**8. Conclusion**

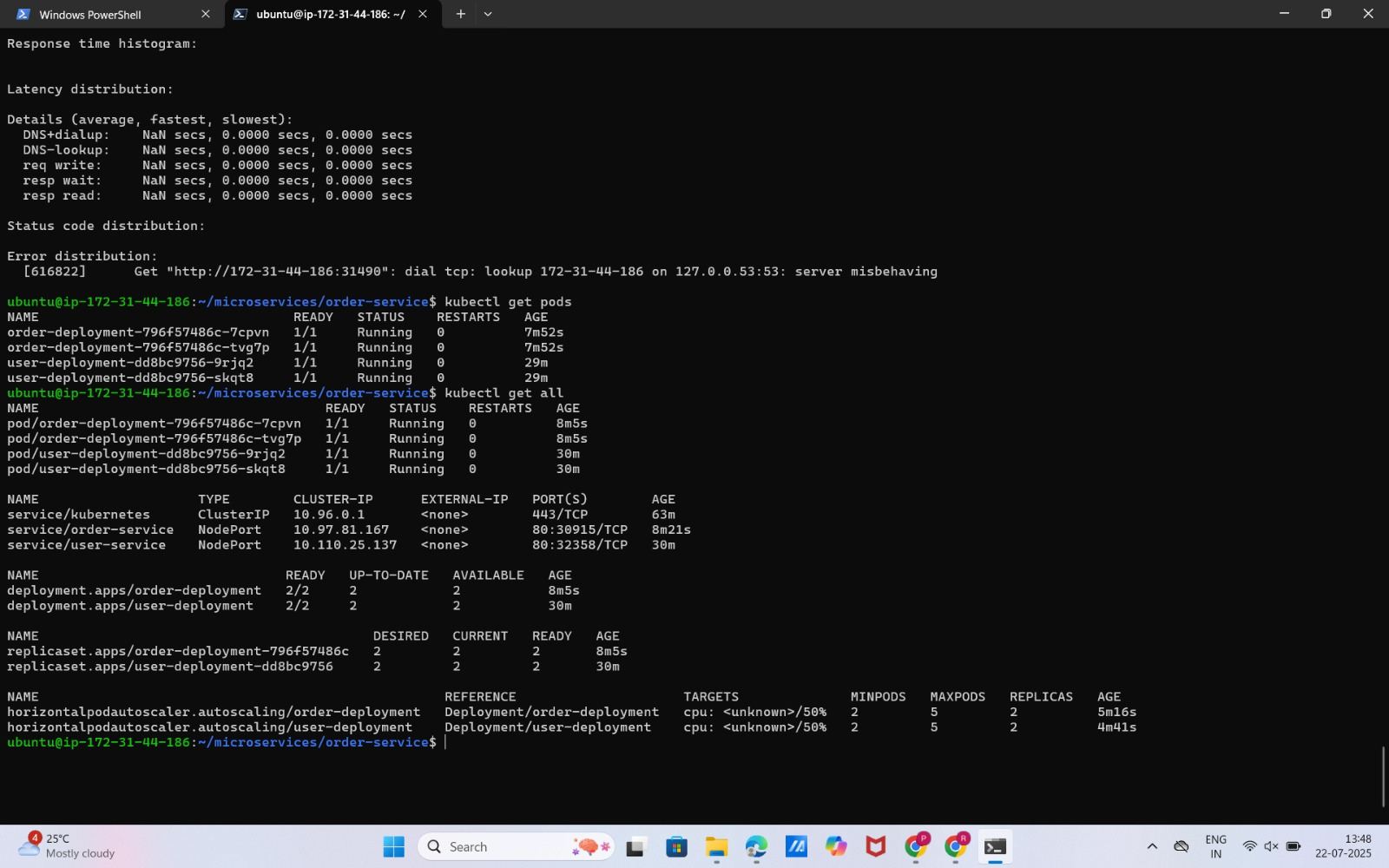
This project demonstrates how to deploy and manage microservices in a production-like Kubernetes environment. Using Docker, Kubernetes, and supporting services, the system achieved containerized isolation, dynamic scaling, and persistent data handling, making it a reliable and scalable cloud-native solution.

**9. Screenshots**

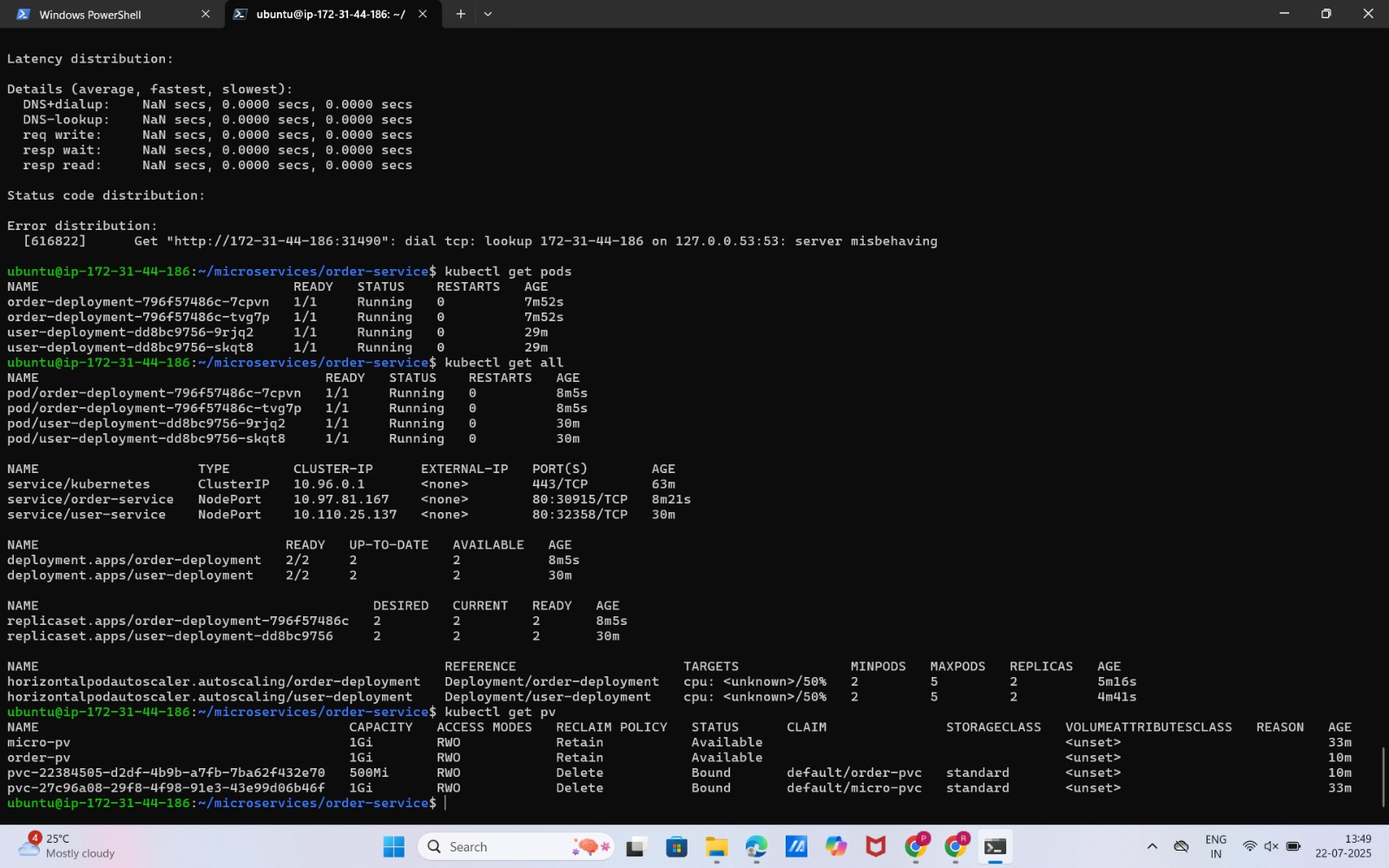
o Cluster and pod status

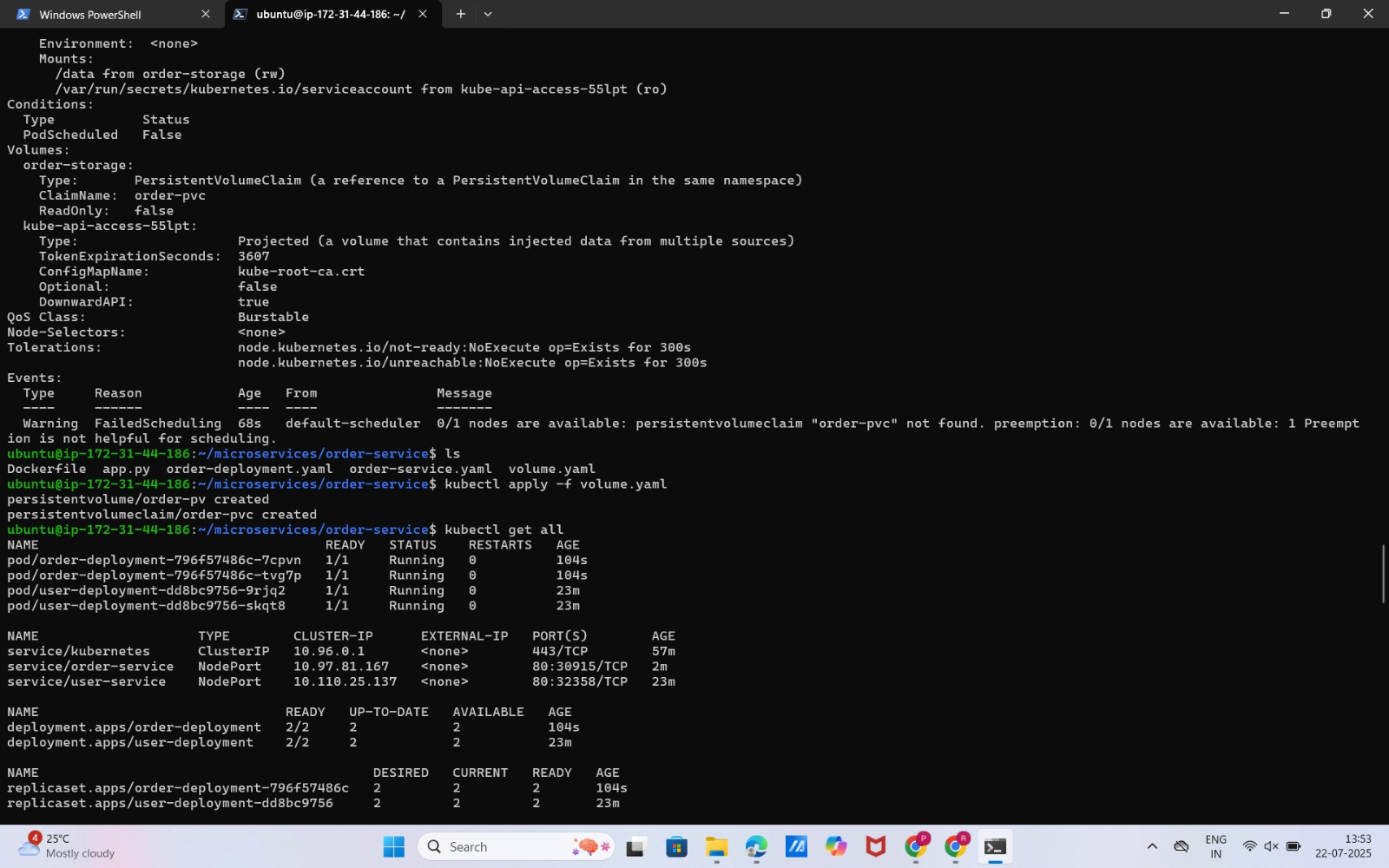


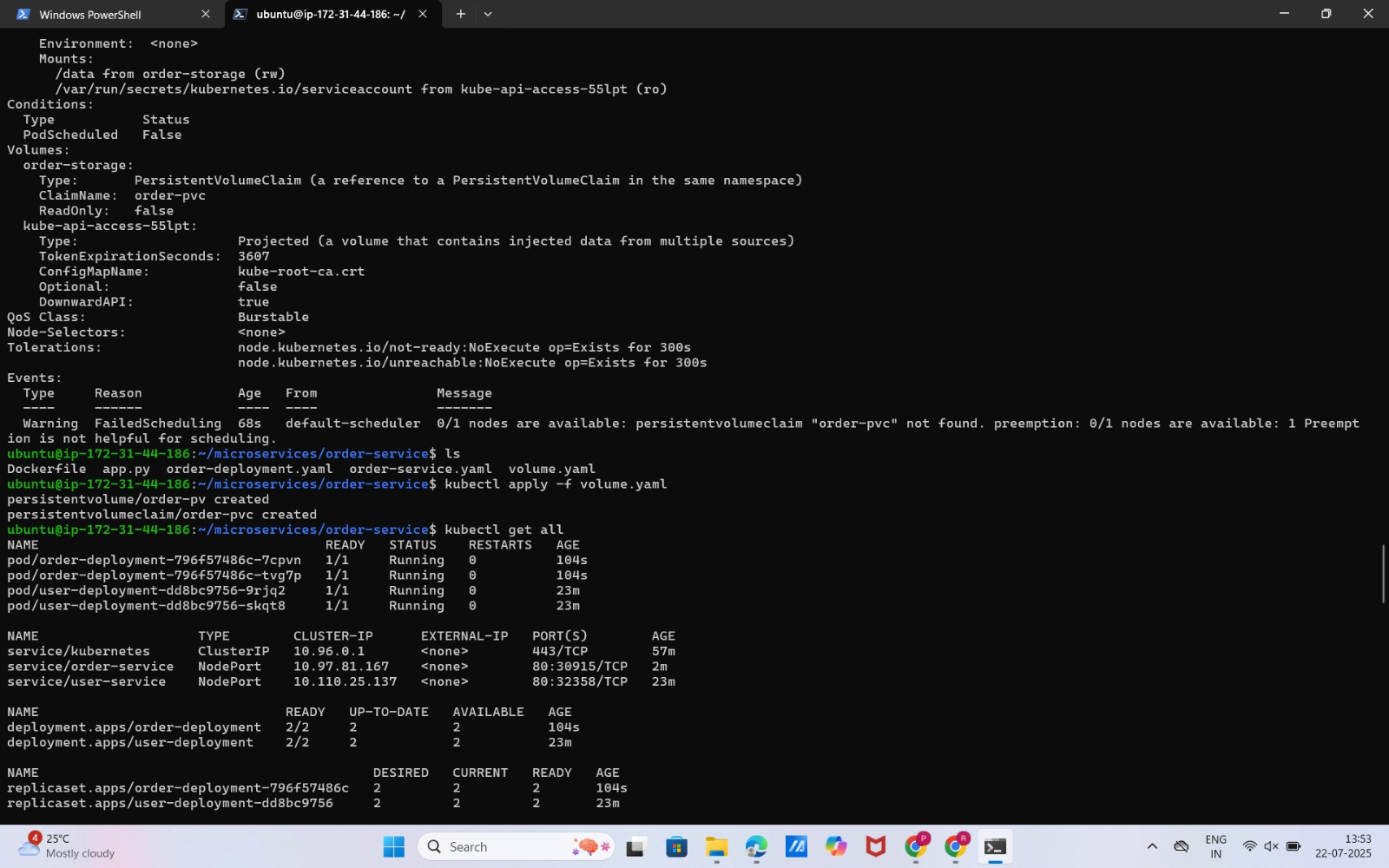
o HPA scaling



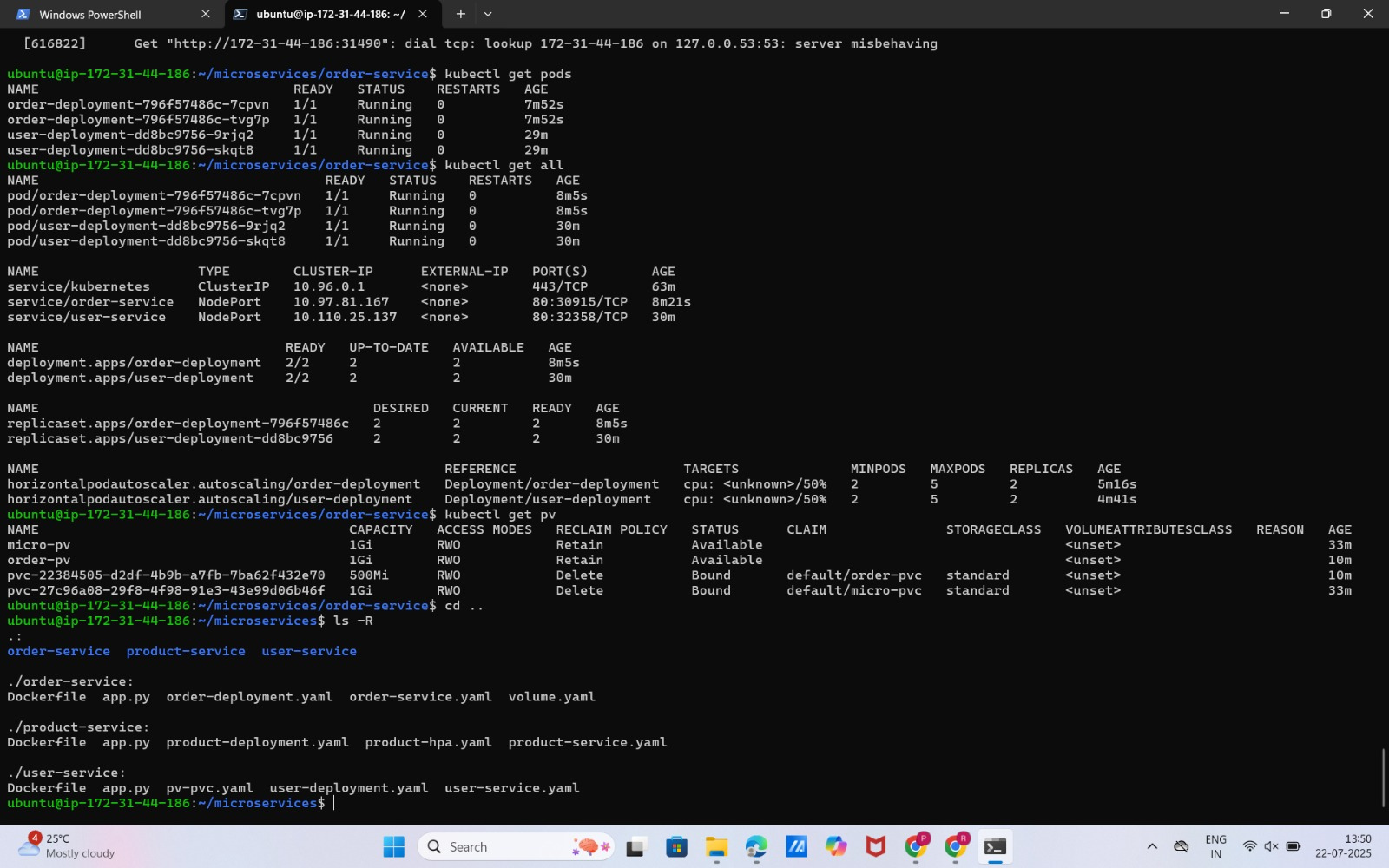
o Persistent volume data being used







o Directory structure



**10. Author**

Name: Rohini Pandhare

Role: Cloud & DevOps Intern Cravita Technologies

repository name : [GitHub - rohinipandhare12/Microservices-Deployment](https://github.com/rohinipandhare12/Microservices-Deployment)