

Building a CI/CD Pipeline for a Retail Company

Project 1



ABC technologies

Industry-Grade CI/CD Project Report

# 1. Introduction

In today’s fast-paced digital landscape, automation and continuous delivery have become the backbone of reliable software releases. This project demonstrates a fully automated end-to-end CI/CD pipeline for a Java-based retail web application, integrating Jenkins, Docker, Ansible, Kubernetes, Prometheus, and Grafana on AWS EC2.

# 2. Project Architecture

The CI/CD pipeline consists of multiple stages that automate the process from code development to deployment and monitoring. The overall flow moves from Developer (GitHub) → Jenkins → Docker Host → Kubernetes Cluster → Monitoring Stack.

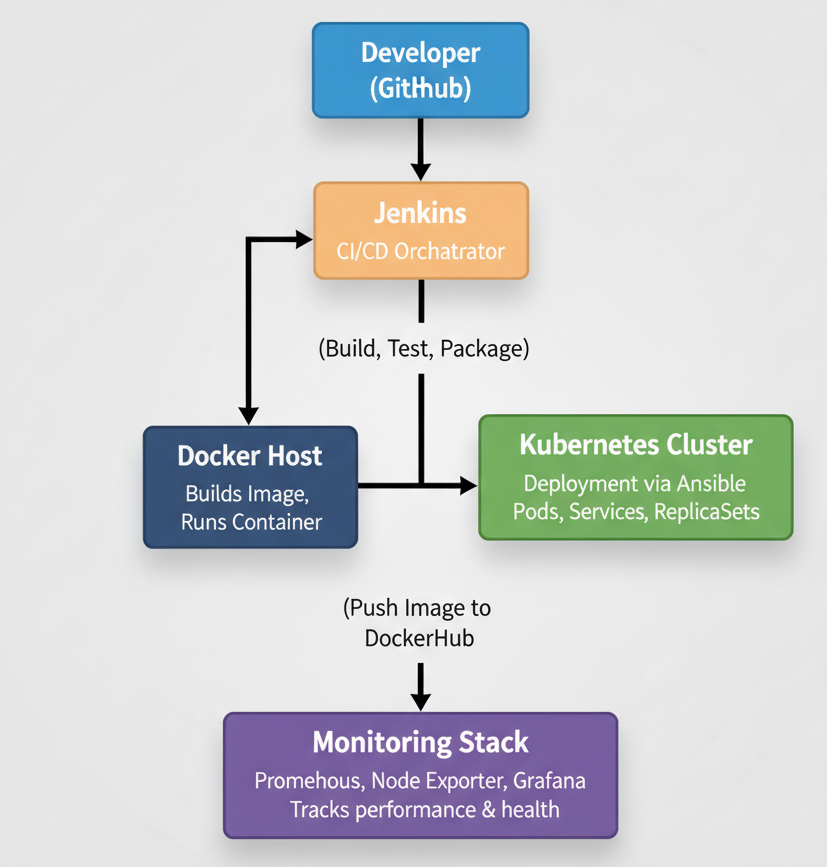


Figure 1: CI/CD Architecture Diagram

# 3. AWS Infrastructure Setup

An EC2 instance running Ubuntu 24.04 LTS was configured as the central automation server hosting Jenkins, Docker, and Ansible.

**Step 1: Launch EC2 Instance**

* **AMI**: Ubuntu 22.04 LTS
* **Instance type**: t2.medium
* **Storage**: 30 GB
* **Key Pair**: Generate .pem
* **Security Group Ports**:
  + 22 (SSH)
  + 8080 (Tomcat/Jenkins)
  + 8081 (App container)
  + 3000 (Grafana)
  + 9090 (Prometheus)
  + 9100 (Node Exporter)

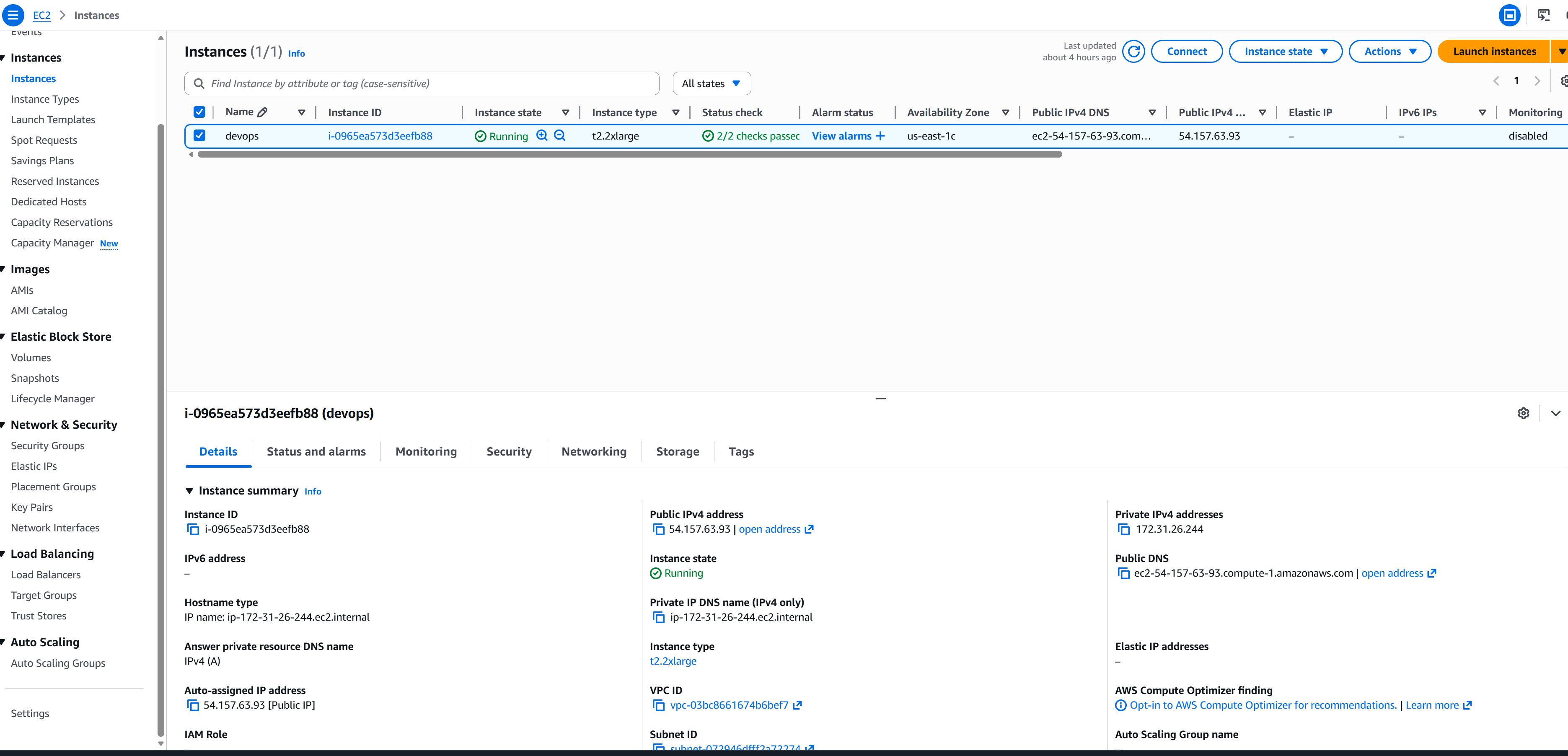


Figure 2: EC2 Instance

Security groups were configured to allow SSH, Jenkins (8080), Application (8081), Grafana (3000), Prometheus (9090), and Node Exporter (9100) access.

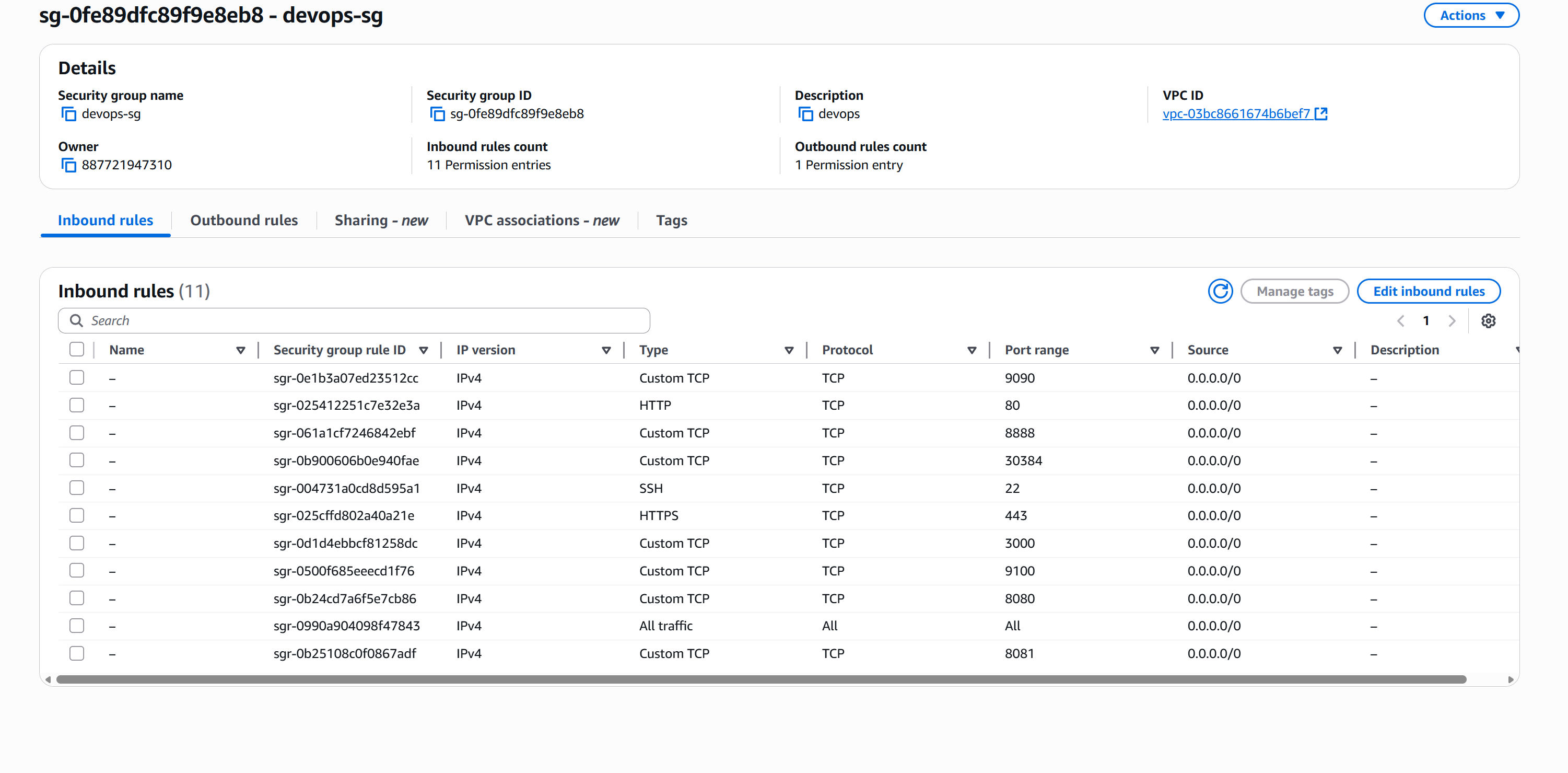


Figure 3: Security Group

**Step 2: Connect via SSH**

chmod 400 master-key.pem

ssh -i master-key.pem ubuntu@<EC2-IP>

# 4. Tool Installation

Essential DevOps tools including Java, Maven, Git, Docker, Jenkins, and Ansible were installed using apt package manager. Jenkins was configured as a service and accessed via its web interface.

# 5. Jenkins Install & Configuration

Jenkins was configured with Docker Pipeline, Maven, Git, and Kubernetes CLI plugins. DockerHub and GitHub credentials were securely stored within Jenkins credentials manager.

Install required packages:

sudo apt update -y

sudo apt install openjdk-17-jdk maven git docker.io python3-venv ansible -y

Install and start Jenkins:

curl -fsSL https://pkg.jenkins.io/debian/jenkins.io.key | sudo tee \

/usr/share/keyrings/jenkins-keyring.asc > /dev/null

echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \

https://pkg.jenkins.io/debian binary/ | sudo tee \

/etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt update

sudo apt install jenkins -y

sudo systemctl enable jenkins --now

Access Jenkins at http://<EC2-IP>:8080

**🧩 Jenkins Configuration**

**Plugins**

Install:

* Docker Pipeline
* Maven Integration
* Ansible
* Git
* Kubernetes CLI

**Credentials**

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **ID** | **Type** | **Description** |
| docker-key | Username & Password | DockerHub Credentials |
| github-cred | Token/Username | Git Repository Access |

**6. Jenkins Pipeline (Jenkinsfile Summary)**

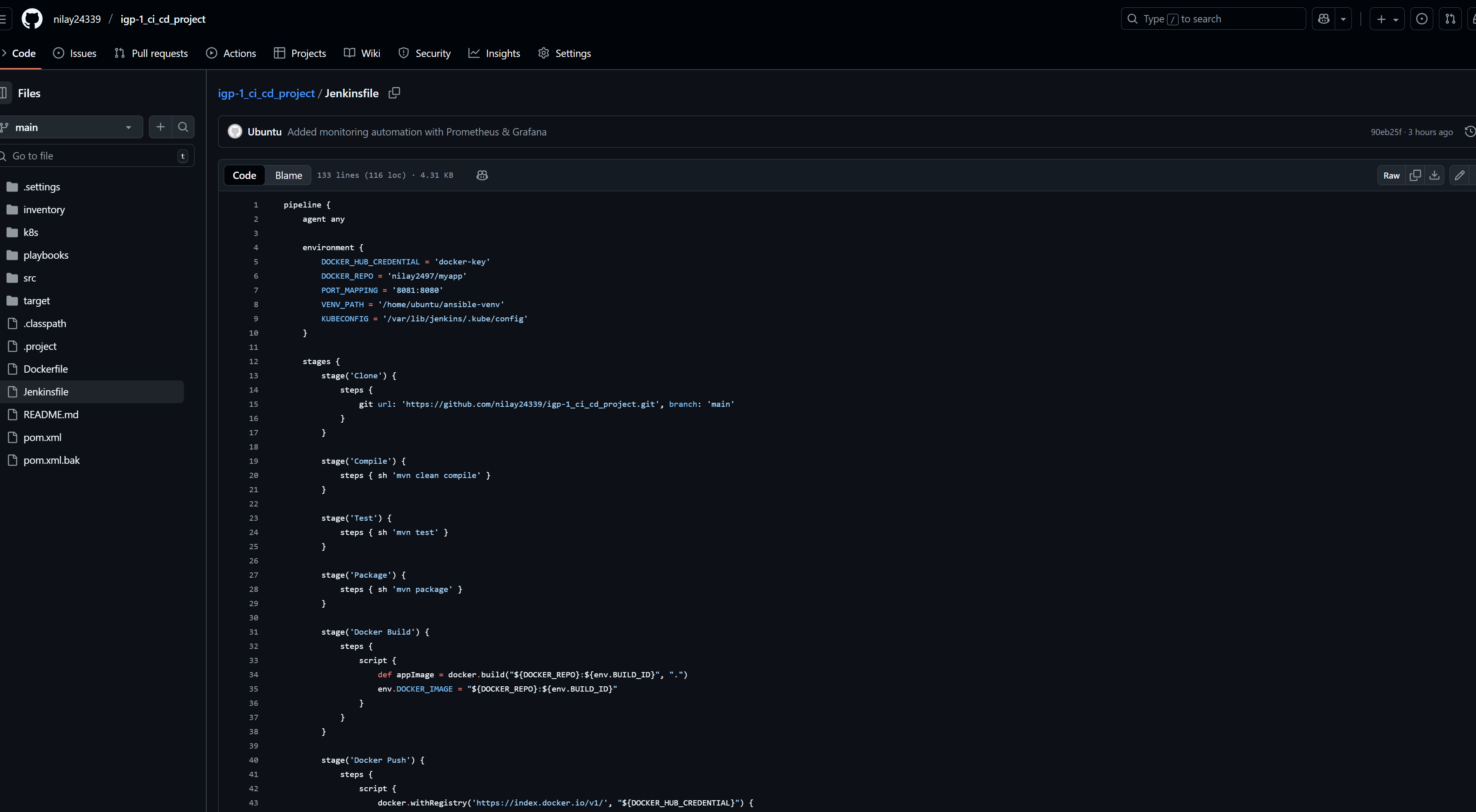
****

Figure 4: Jenkinefile

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Stage** | **Description** | | **Clone** | Pulls code from GitHub | | **Compile** | Builds Java classes using Maven | | **Test** | Executes JUnit test cases | | **Package** | Generates WAR file | | **Docker Build & Push** | Builds Docker image and pushes to Docker Hub | | **Docker Run** | Deploys container locally for test | | **Ansible Deploy** | Deploys image to Kubernetes | | **Monitoring Setup** | Installs Prometheus, Node Exporter, Grafana | | **Health Check** | Validates pod, service, monitoring status | |  |
|  |  |
|  |
| Figure 5: Pipeline Success |
|  |
|  |
| Figure 6: Stage View |
|  |
|  |
|  |  |

**7. Docker Setup**

Dockerfile was created to containerize the Java application using Tomcat base image. The pipeline automatically builds and pushes the image to DockerHub.

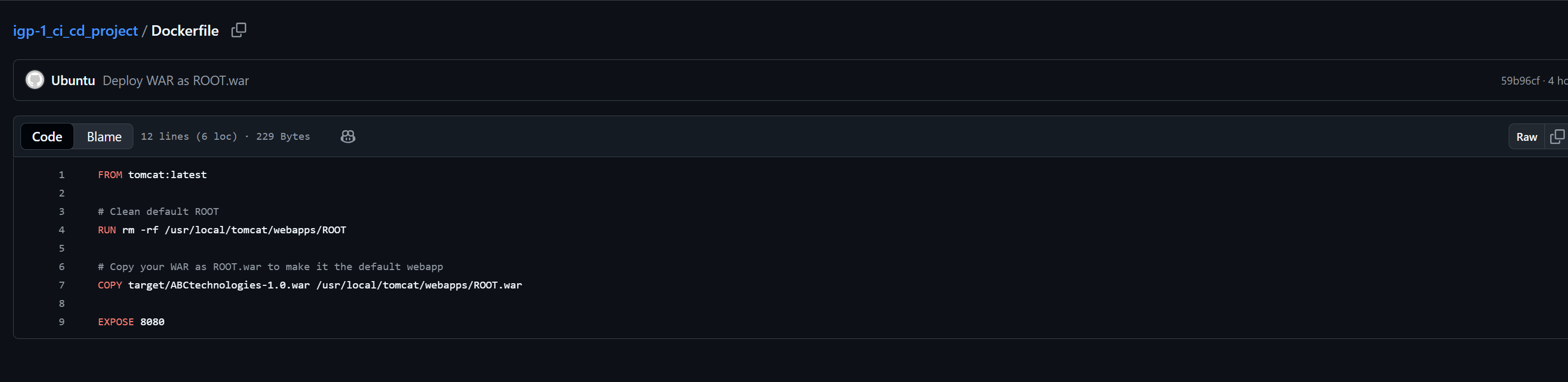


Figure 7: Dockerfile

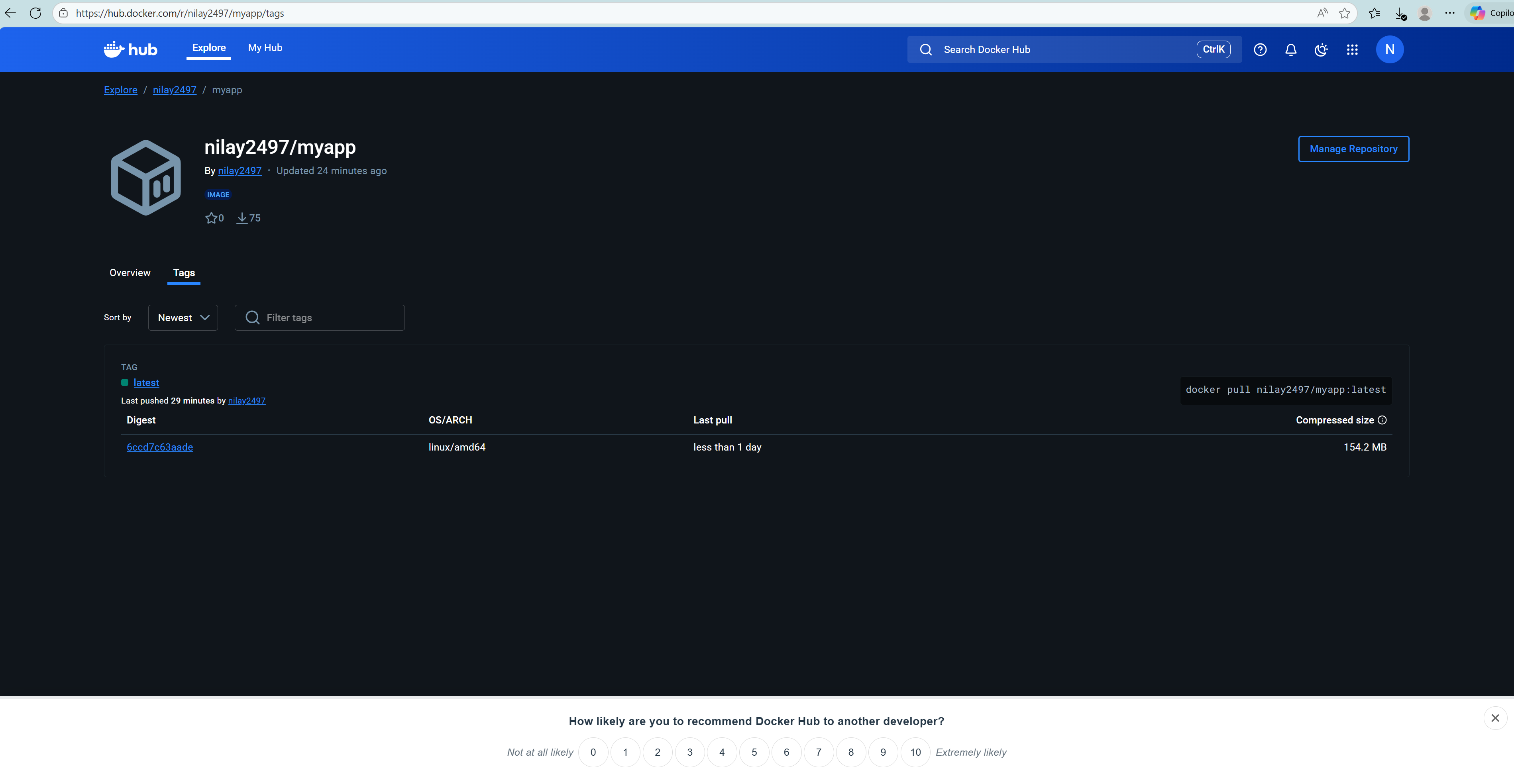


Figure 8 : Docker push Image

**8. Ansible Deployment**

Ansible playbooks automate deployment of Docker images into Kubernetes clusters.

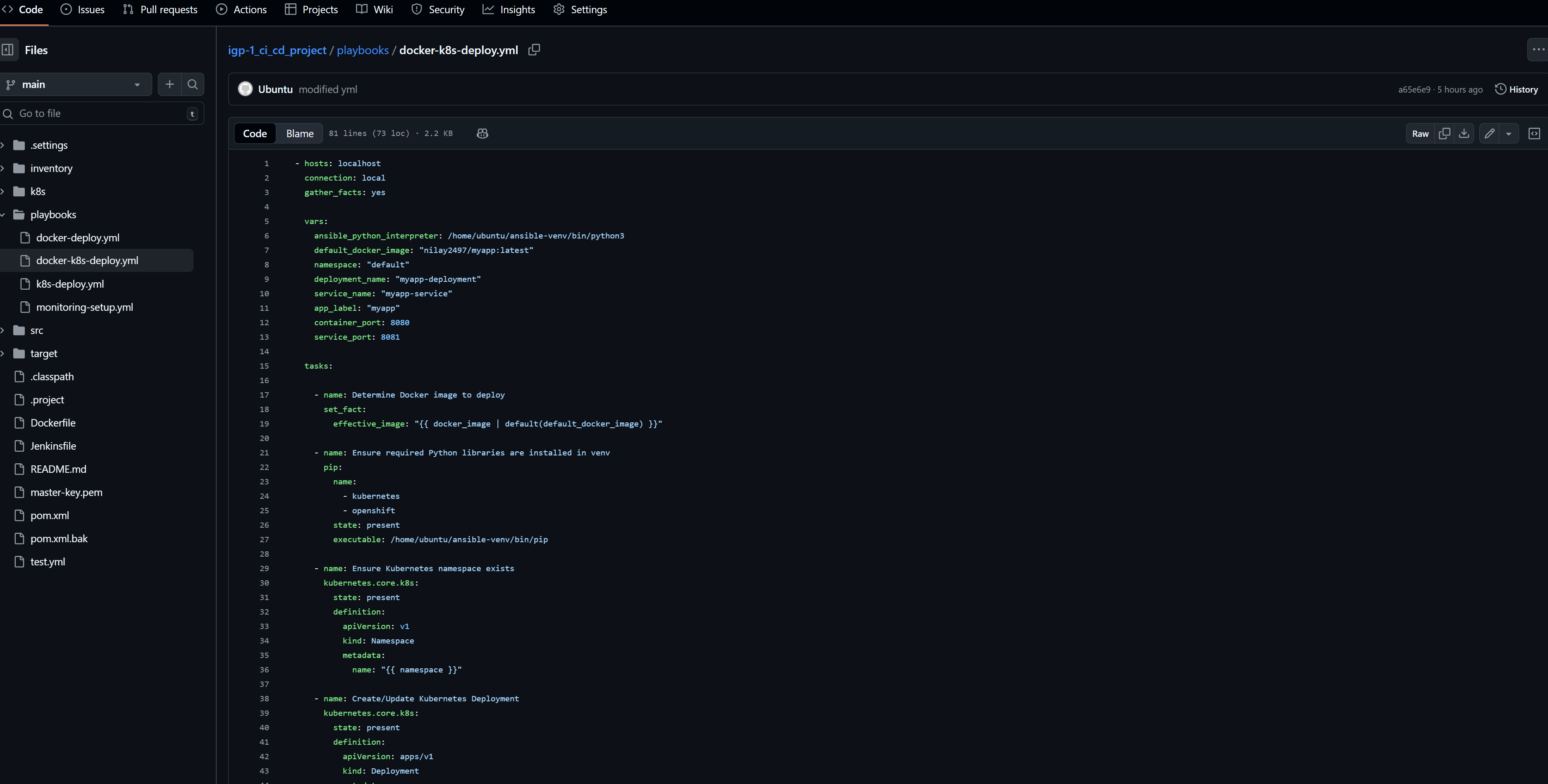


Figure 9: docker-k8s-deployment.yml

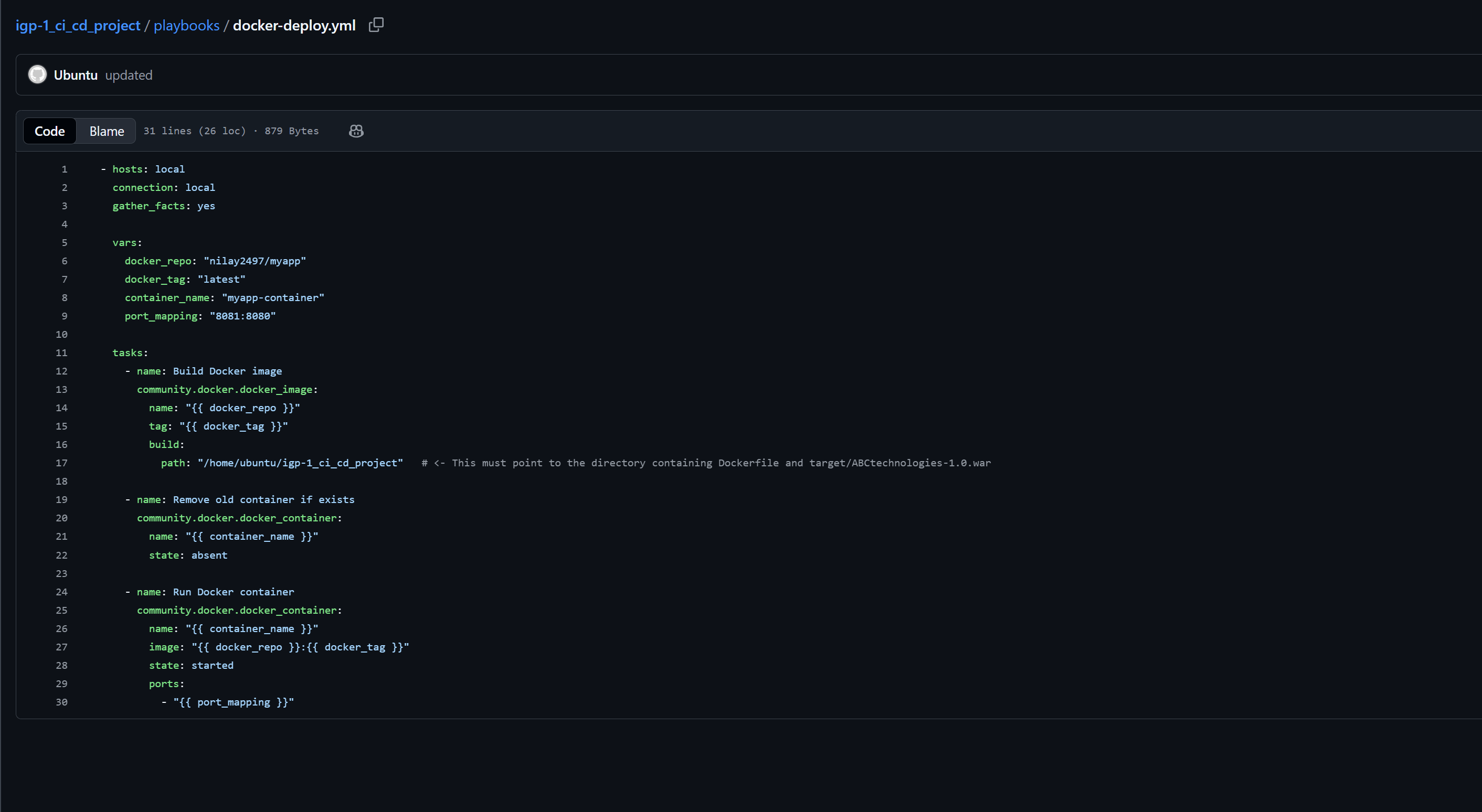


Figure 10: docker-k8s-deployment.yml

# 9. Kubernetes Configuration

Kubernetes manifests define Deployment and Service objects to manage application replicas and expose them externally using NodePort service type.

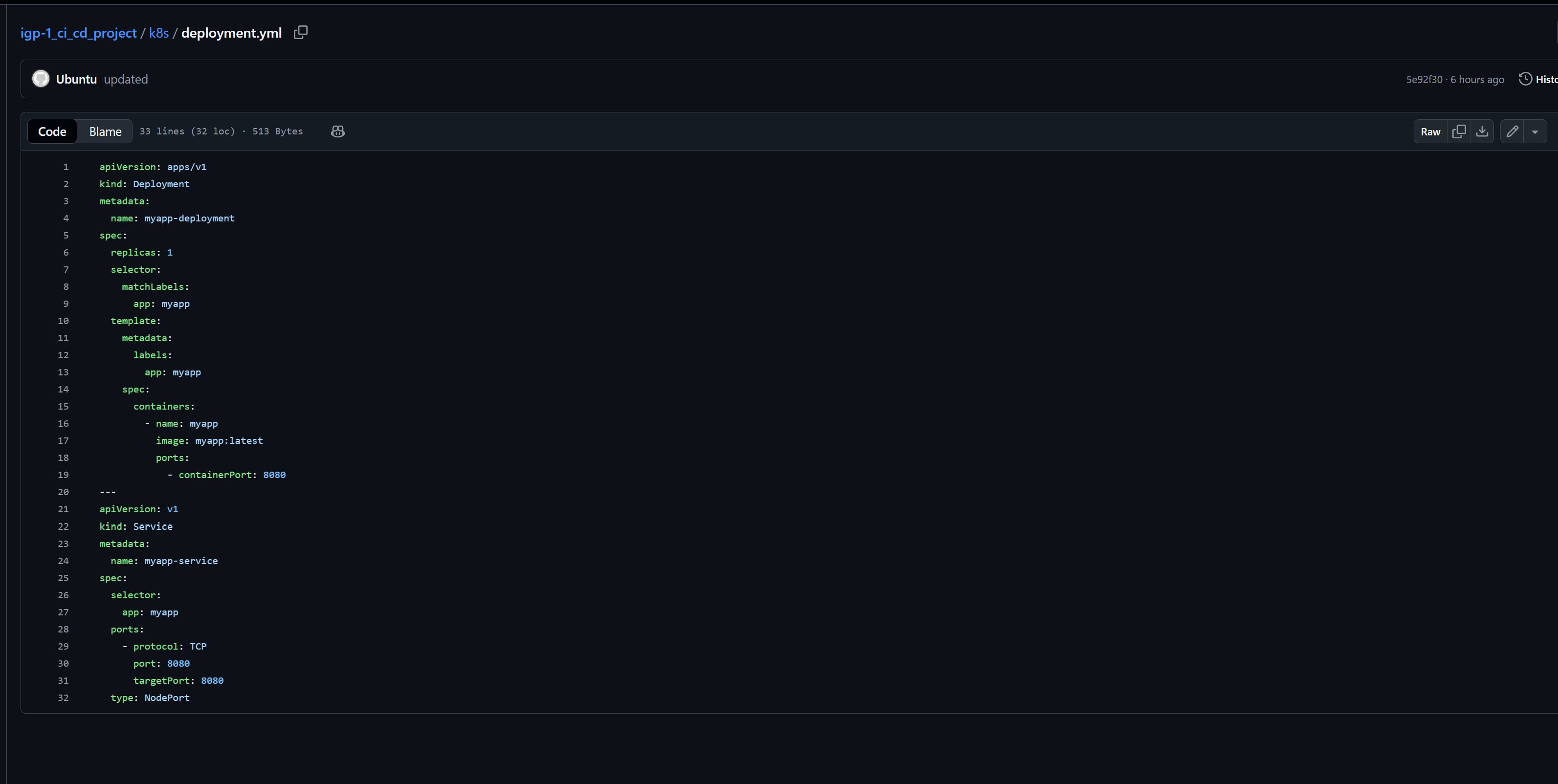


Figure 11: k8s/deployment.yml

http://<EC2-IP>:8081

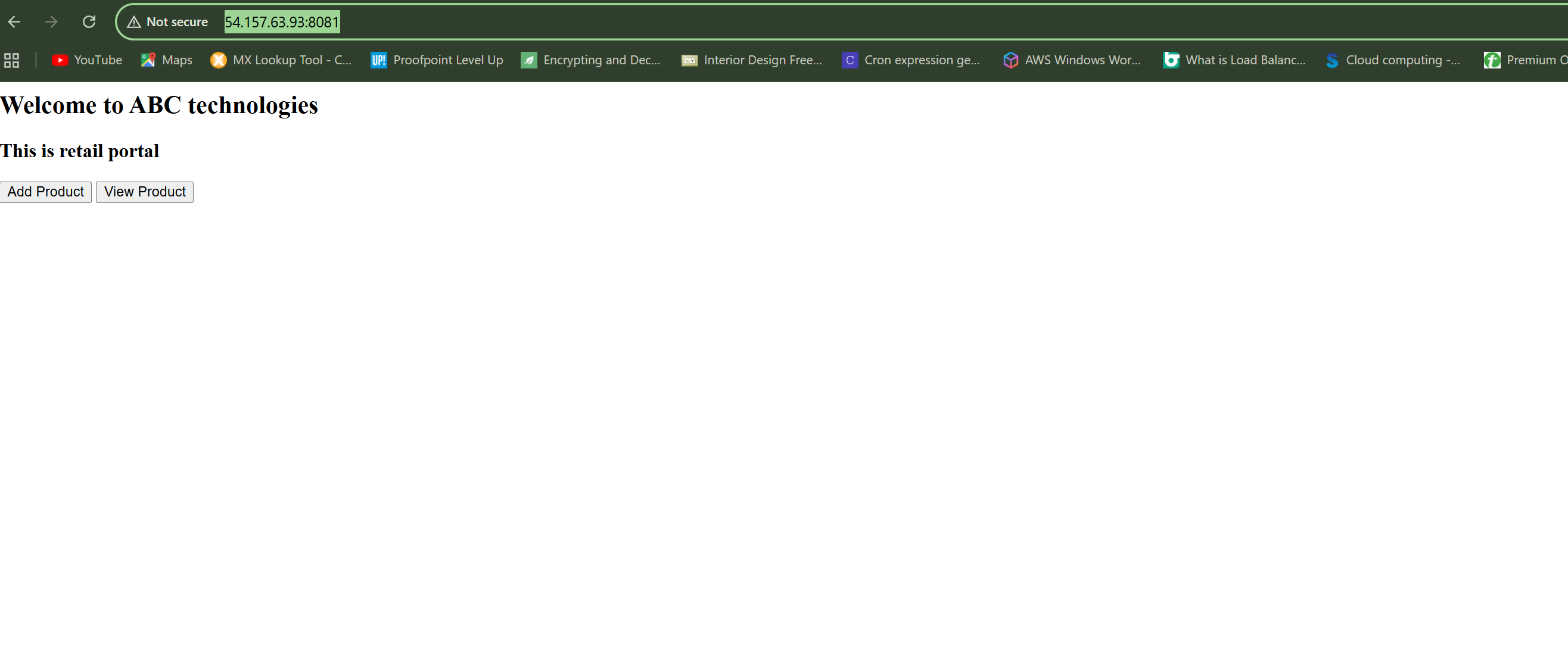


Figure 12: End Result

# 10. Monitoring Setup (Prometheus + Grafana)

Prometheus collects metrics from Node Exporter and Kubernetes endpoints, while Grafana visualizes them using pre-built dashboards for system and application performance.

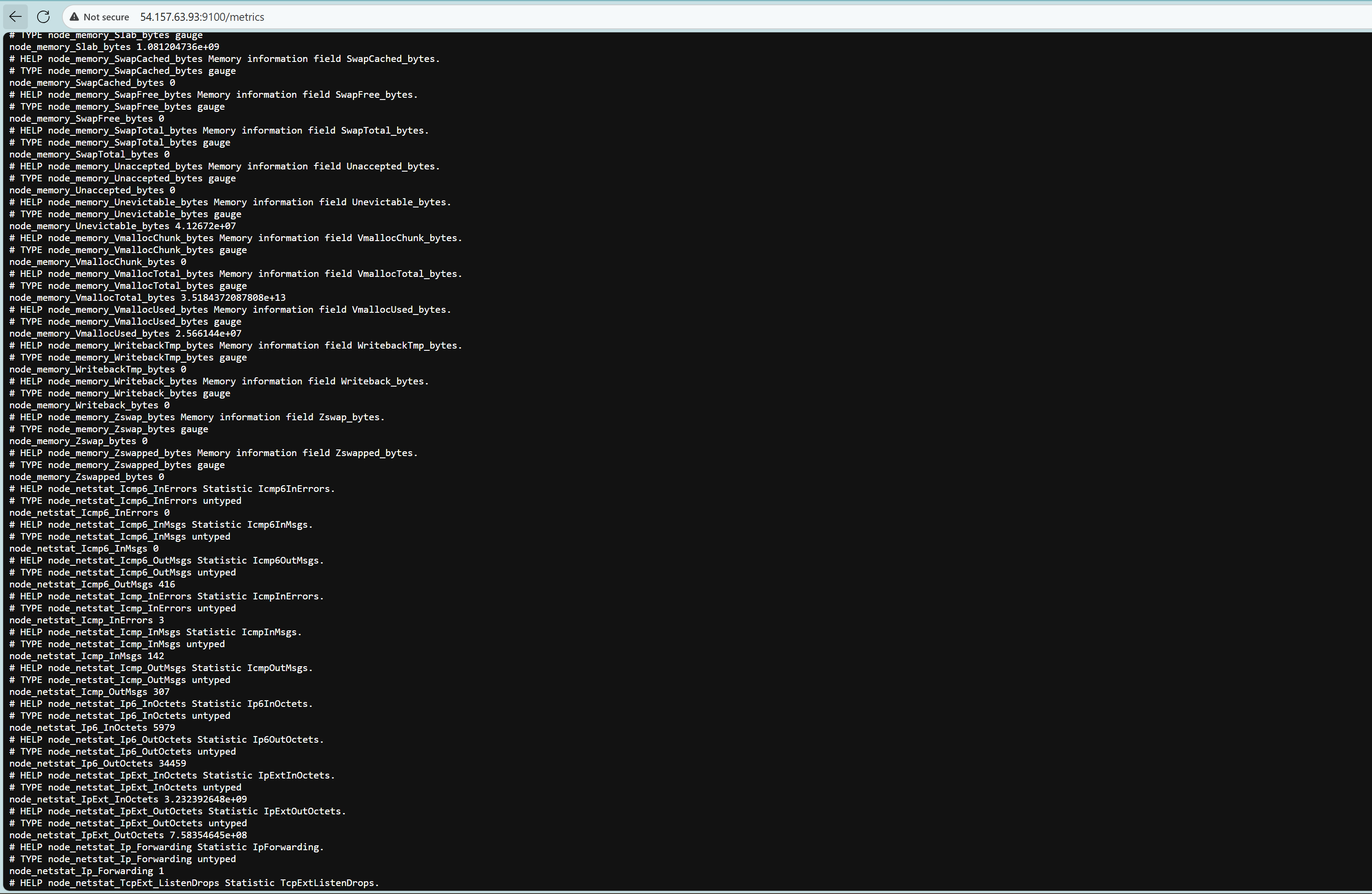


Figure 13: Node matrics

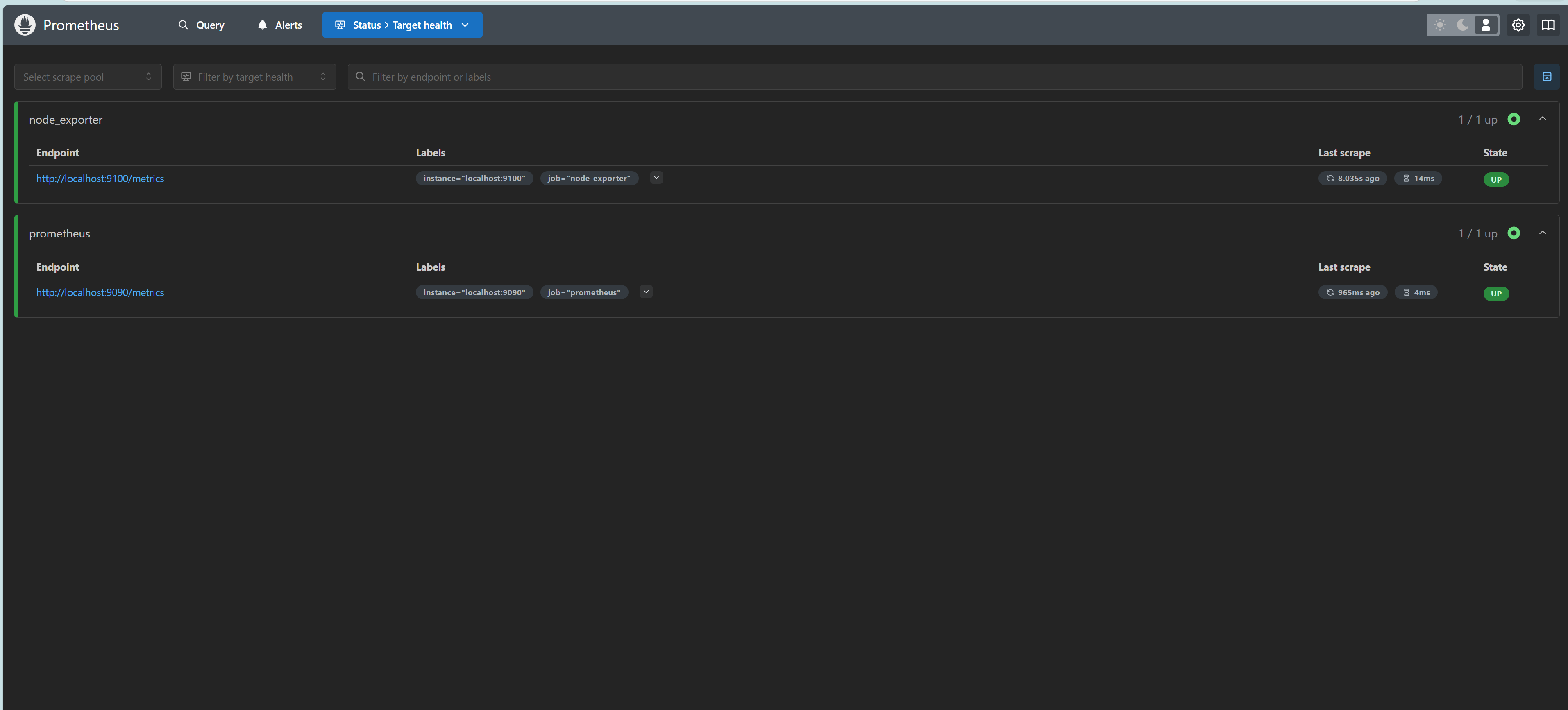


Figure14 : Prometheus

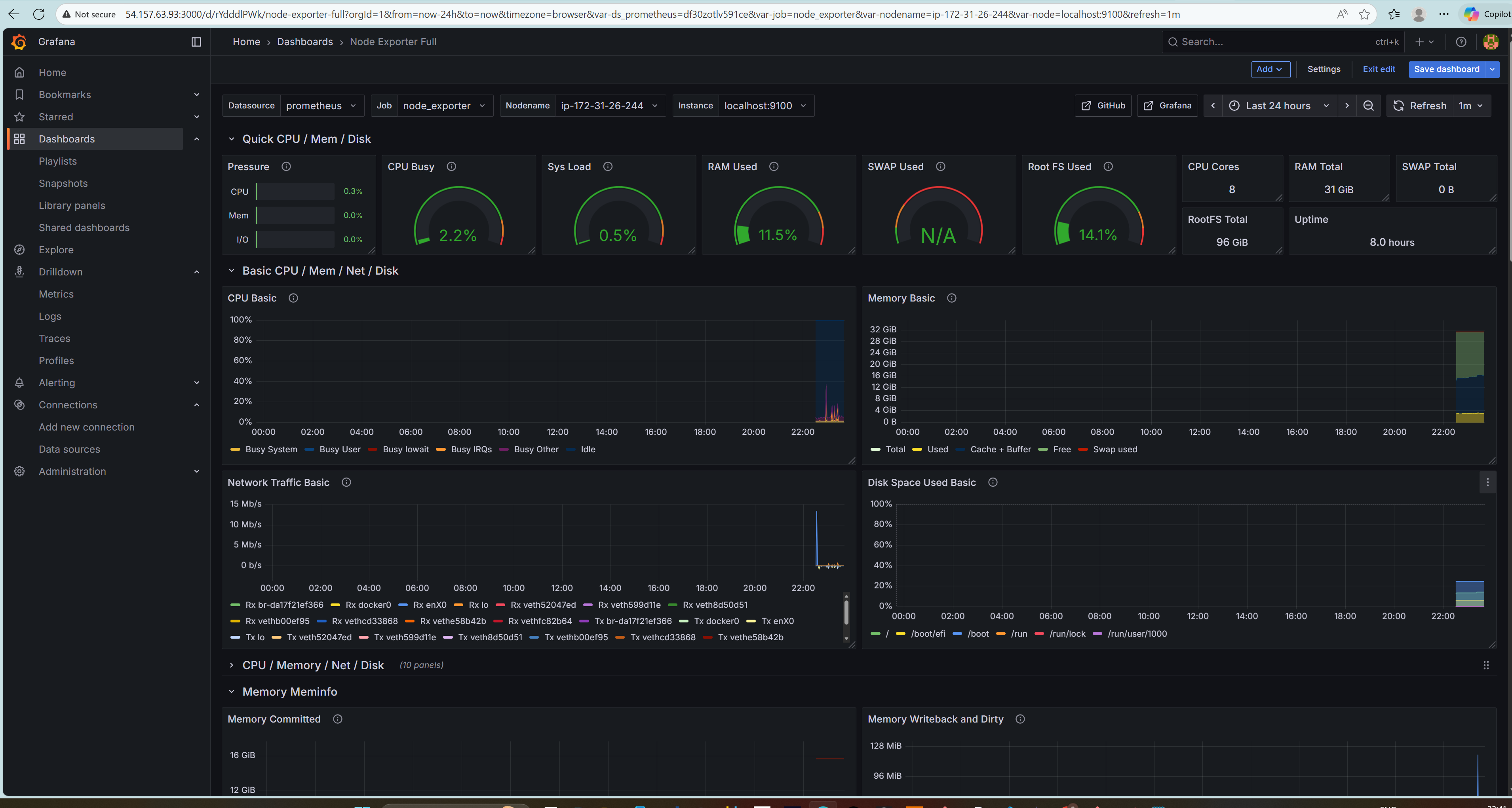


Figure 15 : Grafana Dashboard

# 11. Health Check & Verification

Pod, service, and monitoring system status are verified using kubectl and systemctl commands. All components should be in Running or Active state.

11.1Health Check Commands

* kubectl get pods -n default
* kubectl get svc -n default
* systemctl status prometheus
* systemctl status grafana-server
* ✅ All should show *Running* or *Active*.

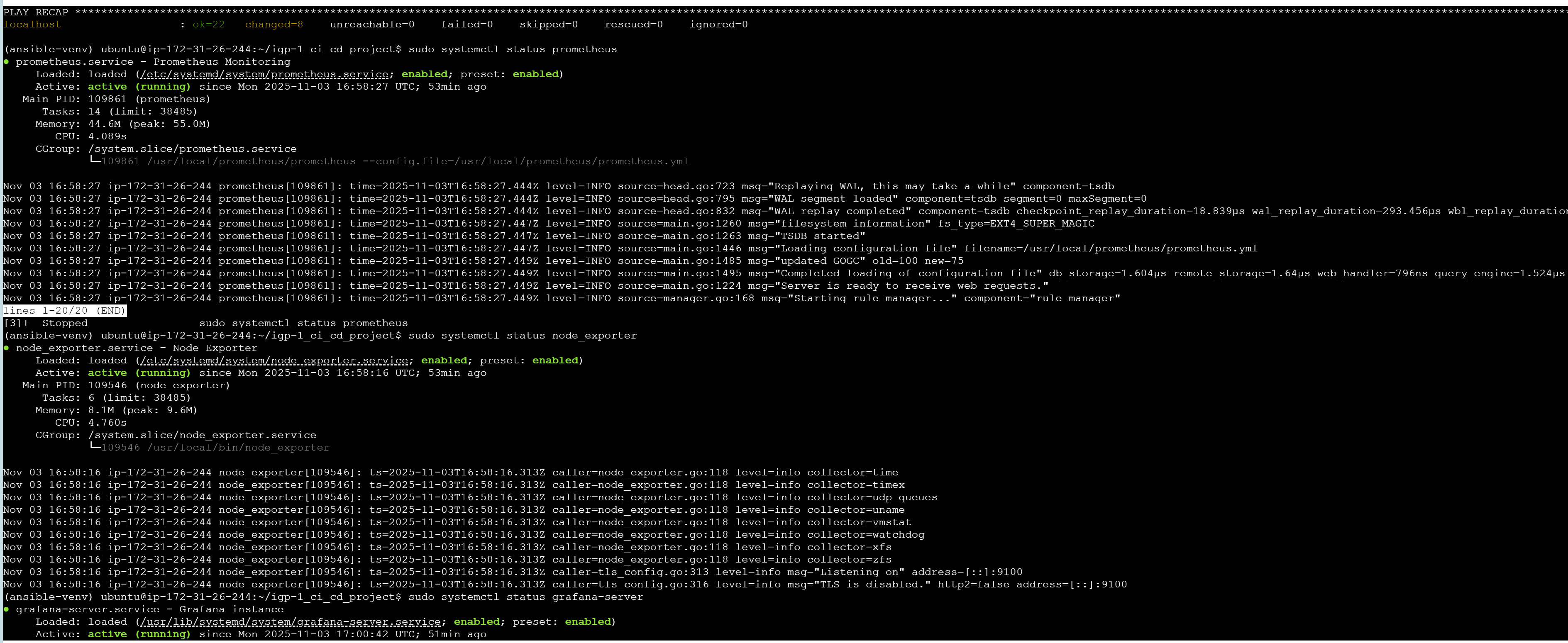


Figure: Status

# 12. Troubleshooting

Issue: Jenkins not building  
Cause: Missing Docker permission  
Fix: sudo usermod -aG docker jenkins

Issue: Ansible SSH fails  
Cause: Invalid host configuration  
Fix: Fix inventory/hosts.ini

Issue: Grafana not loading  
Cause: Port 3000 blocked in SG  
Fix: Add port 3000 to EC2 security group

Issue: Kubernetes pods crash  
Cause: Incorrect image tag  
Fix: Redeploy with correct tag

# 13. Conclusion

This project successfully demonstrates a production-grade CI/CD system integrating Jenkins automation, Docker containerization, Kubernetes orchestration, and real-time monitoring using Prometheus and Grafana. It highlights the benefits of automation, continuous delivery, and observability in modern DevOps environments.