Project: React Application CI/CD Deployment on AWS EKS Cluster

This document provides detailed setup instructions and pipeline explanation for deploying a React application on AWS EKS using Docker, Terraform, Jenkins, and Kubernetes. The project includes monitoring and version control integration.

# 1. React Application Dockerization

Clone the repository from https://github.com/Vennilavan12/Trend.git, create a Dockerfile to dockerize the React application, and build the docker image. Run the container locally on port 3000 to verify.

# 2. Infrastructure Provisioning with Terraform

Use Terraform scripts (main.tf) to define AWS infrastructure including VPC, IAM roles, EC2 instances with Jenkins setup, and other resources required for deployment.

# 3. Kubernetes Cluster Setup

Create and configure an AWS EKS cluster. Write deployment and service YAML manifests to deploy the React application container. Confirm the cluster status and services.

# 4. Jenkins CI/CD Pipeline

Set up Jenkins with Docker, Git, Kubernetes, and Pipeline plugins. Integrate Jenkins with GitHub using webhook to trigger builds on commits. Implement a declarative Jenkins pipeline to build the image, push to DockerHub, and deploy to the cluster.

# 5. Version Control and Code Management

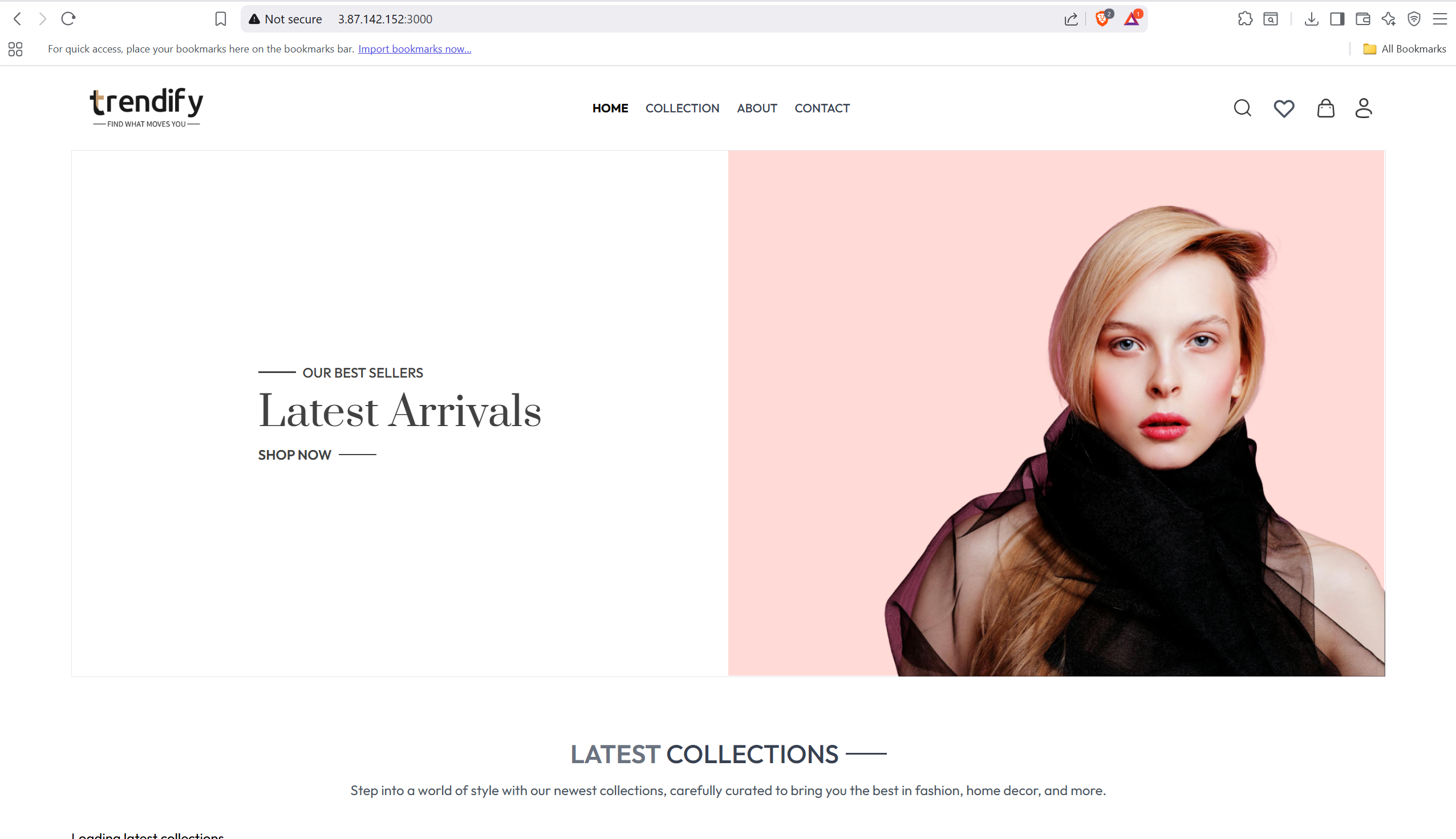
Use GitHub repository for version controlling the application source and deployment scripts. Add appropriate .gitignore and .dockerignore files to manage files.

# 6. Monitoring Setup

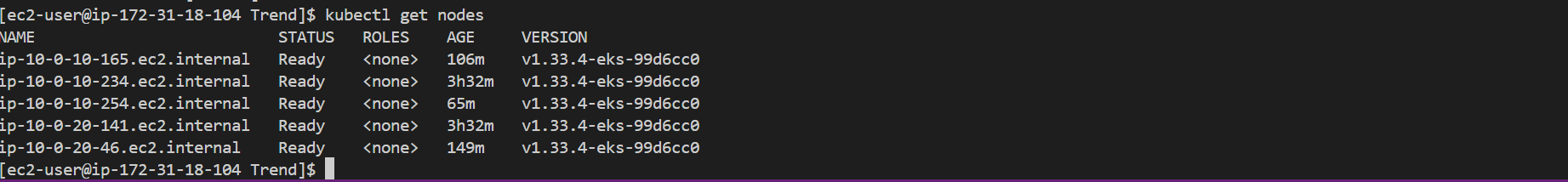
Implement monitoring using open-source tools to check cluster and application health for observability.

## Docker Build and Container Output

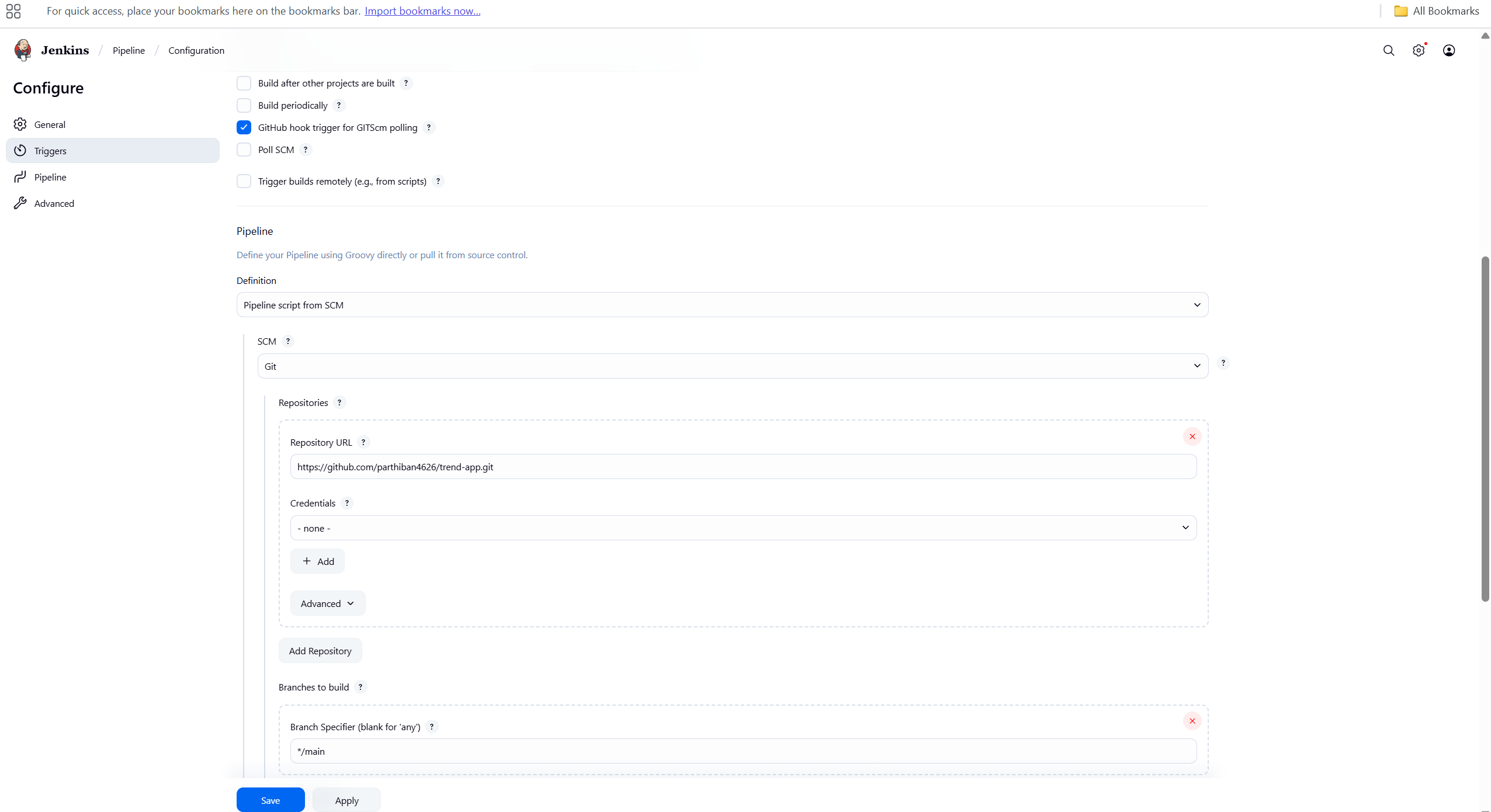


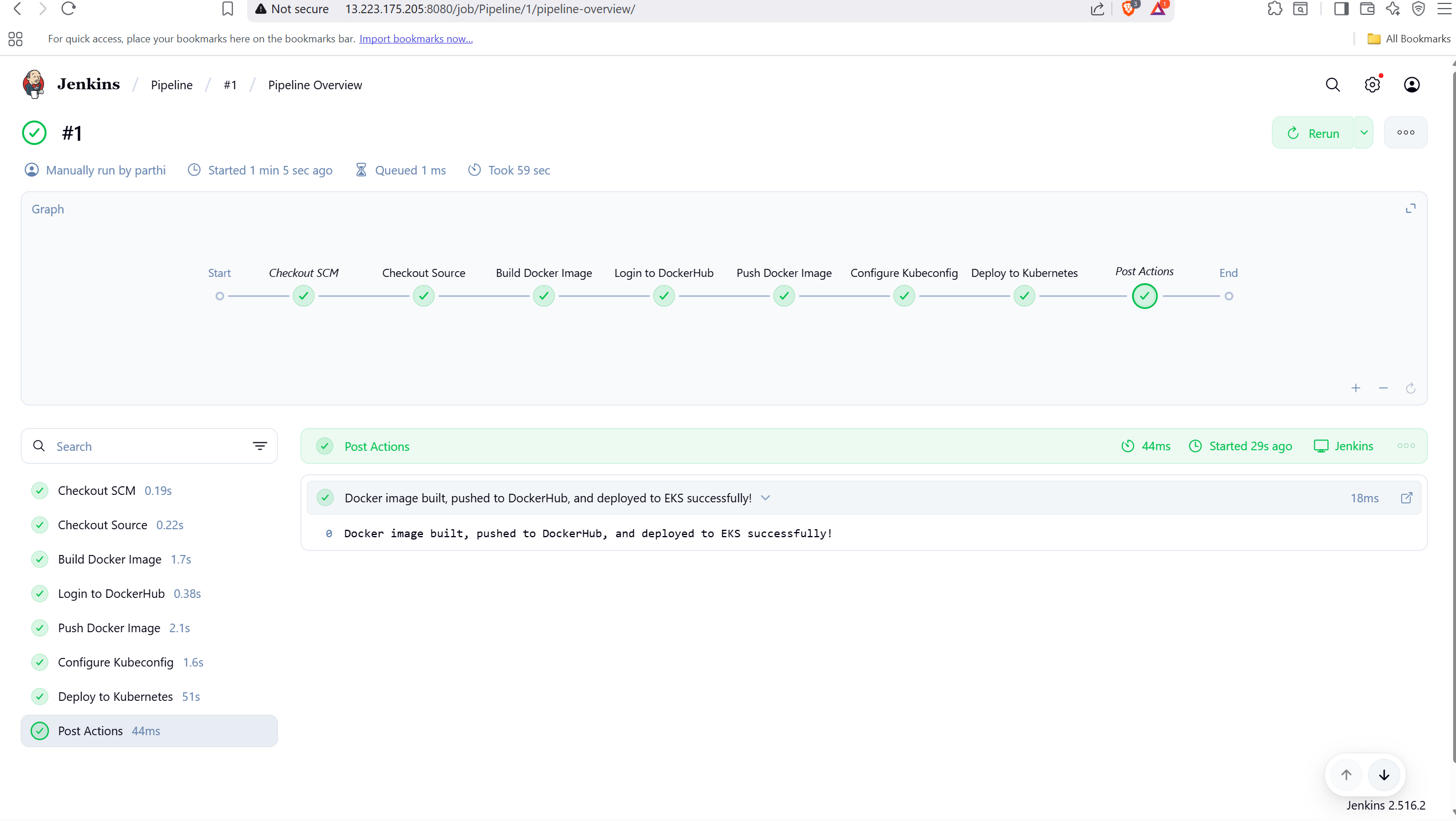


## AWS EKS Cluster Status



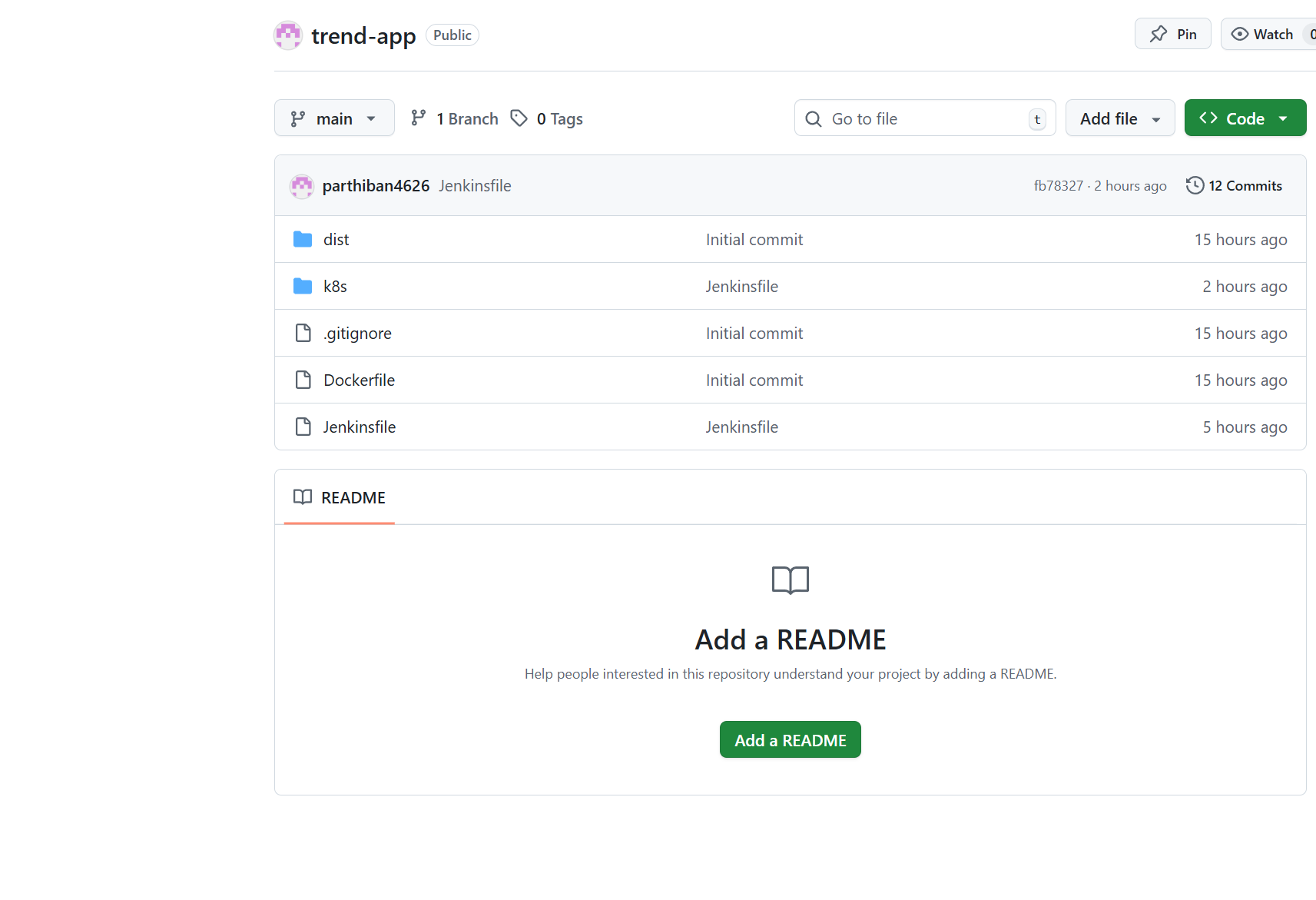
## Jenkins Pipeline Configuration



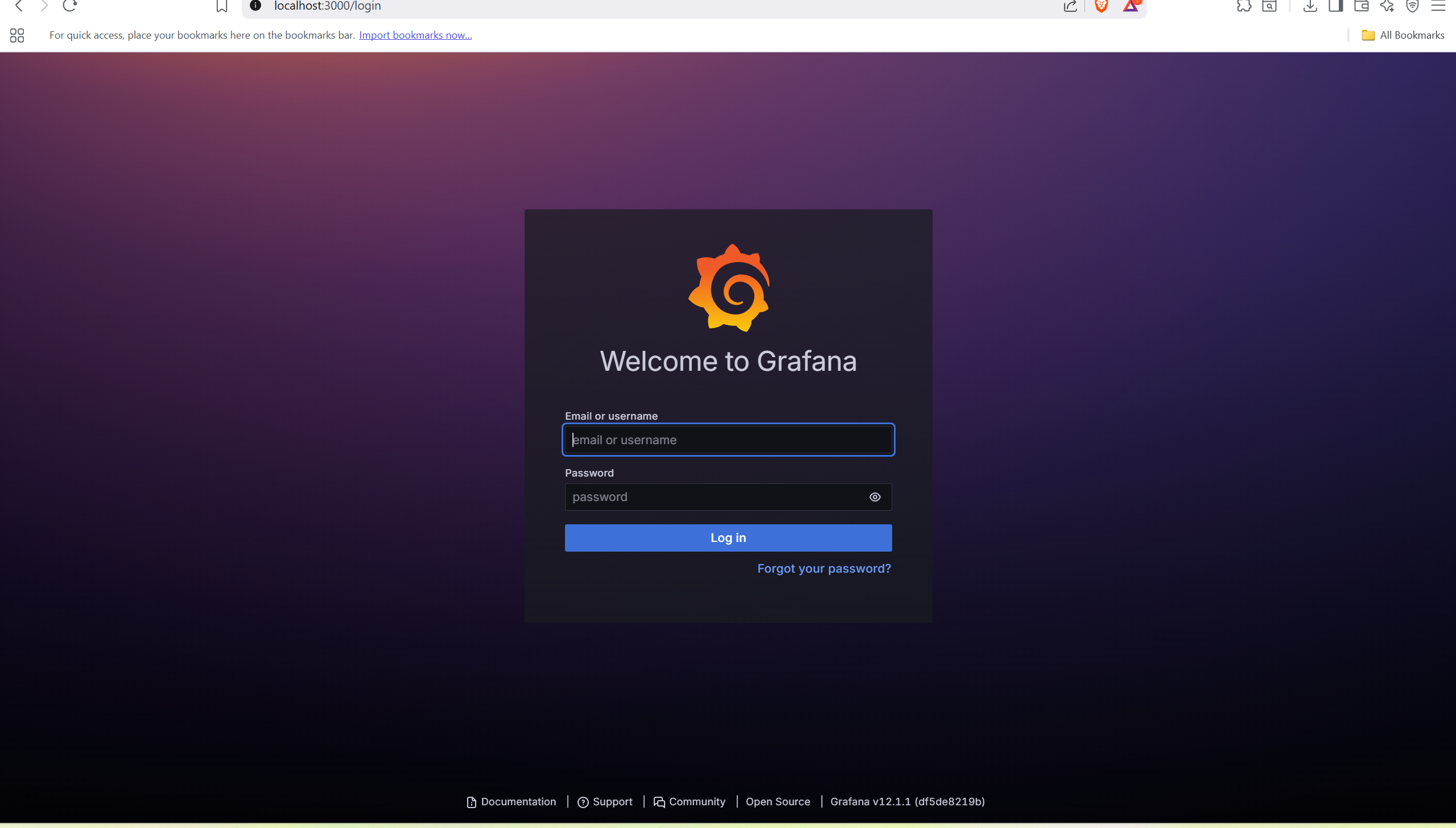


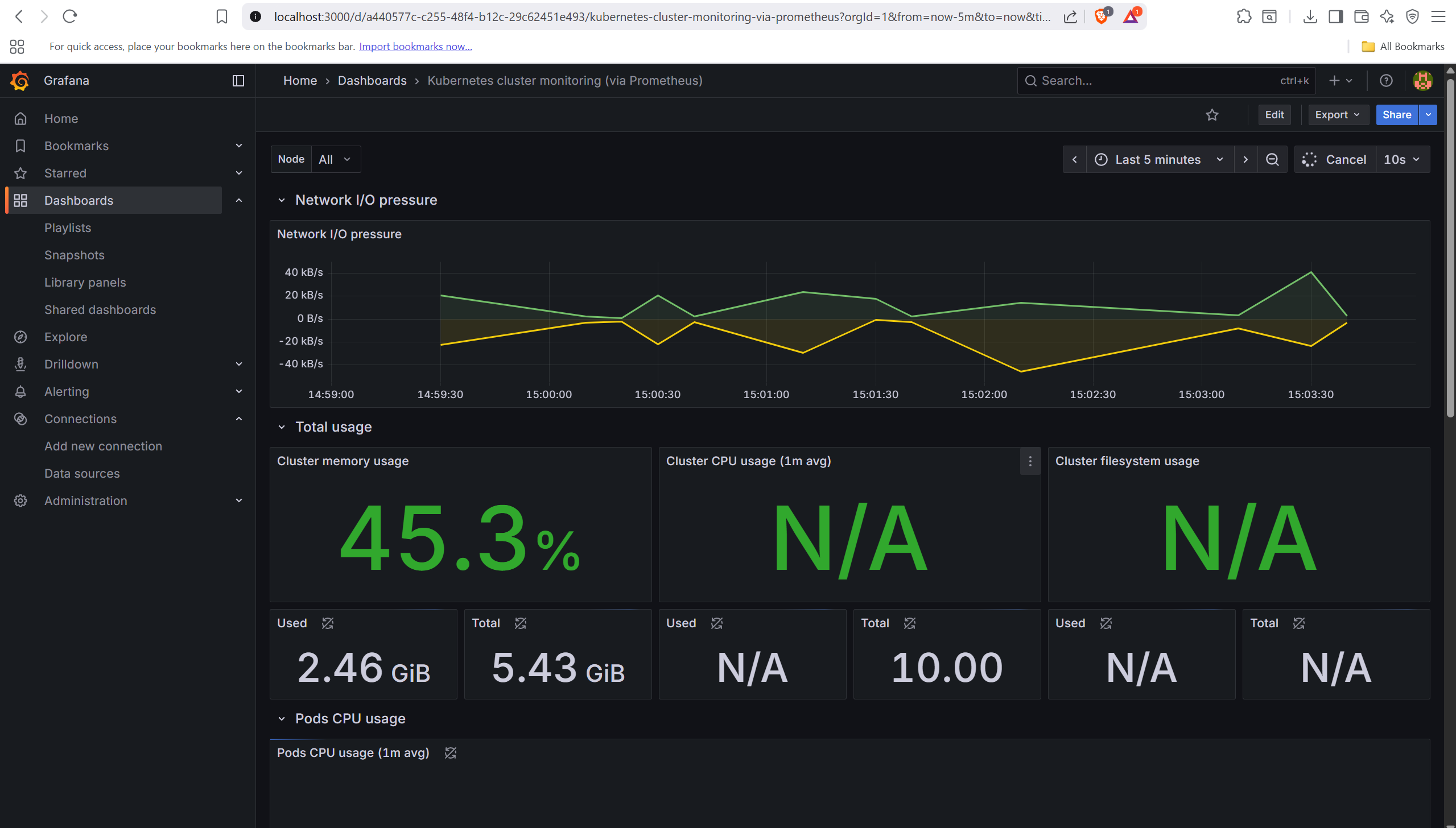
## GitHub Code Repository

<https://github.com/parthiban4626/trend-app.git>



## Monitoring Dashboards





## Application LoadBalancer ARN

<http://a69fa1c1882a64932a80ac9e657eb049-1831847308.us-east-1.elb.amazonaws.com/>

