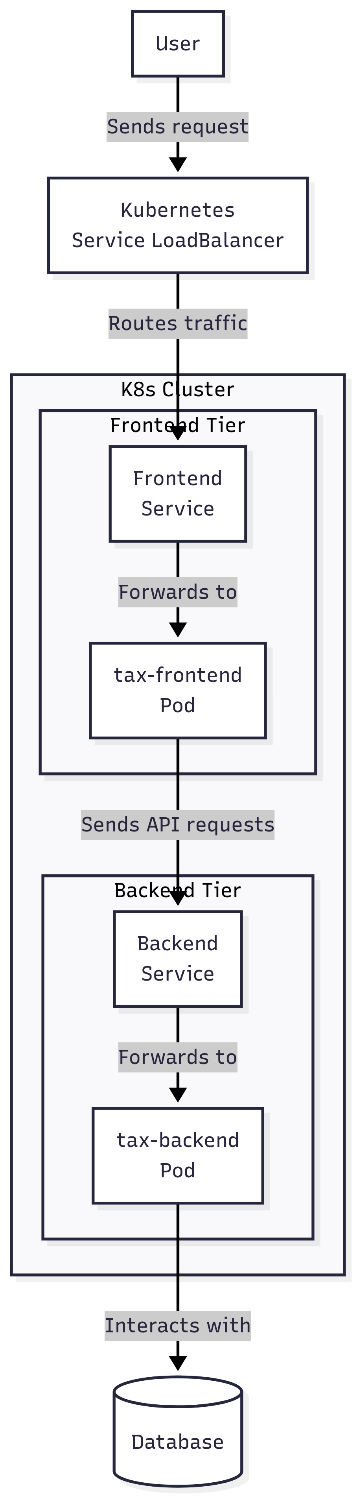
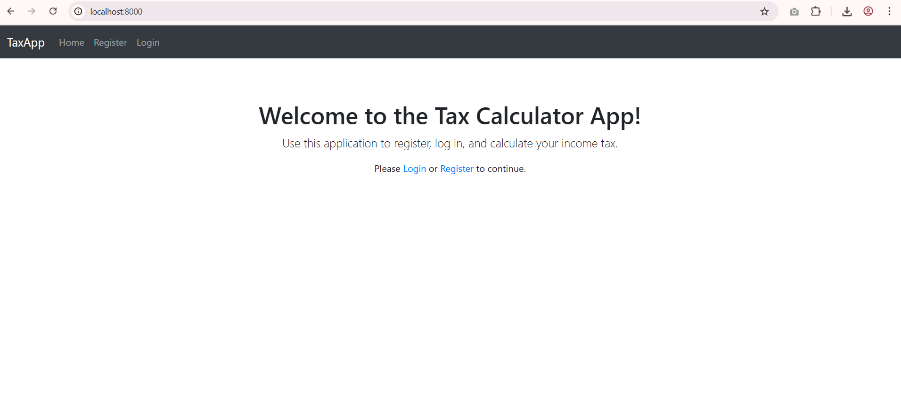
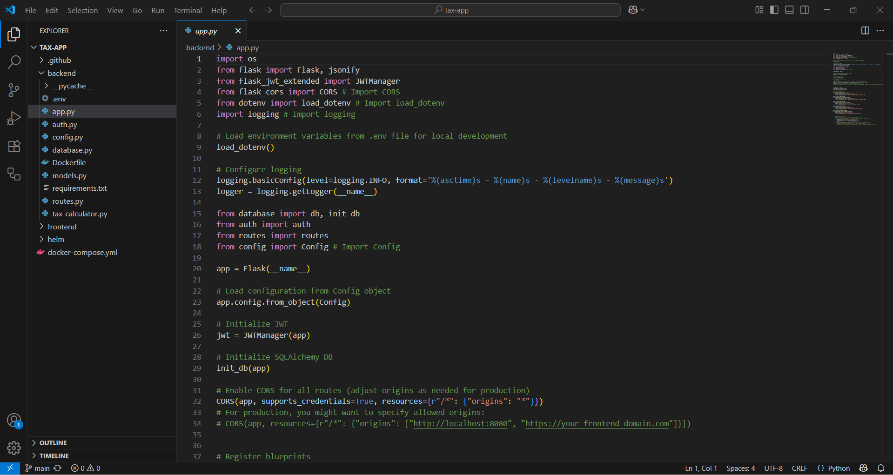
**DevSecOps Assessment Project Documentation**

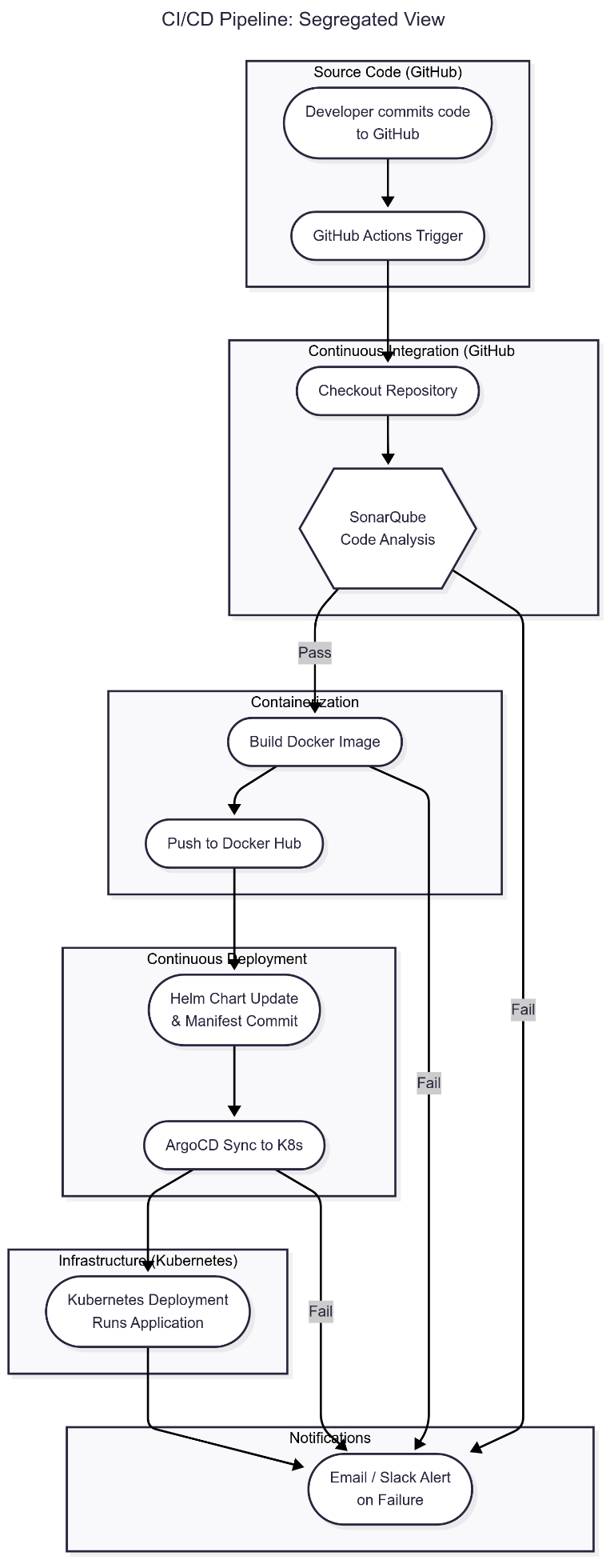
This document provides an overview of the DevSecOps pipeline implemented for the tax calculation application. The project automates the entire software delivery lifecycle, from code commit to production deployment. By leveraging tools like GitHub Actions, SonarQube, Docker, Helm, and ArgoCD, we have established a reliable, repeatable, and secure process for deploying a high-quality, three-tier application.

Three-Tier Application Overview

* Presentation Tier (Frontend): A Flask application serving the user interface. It handles user interactions and sends requests to the backend.tax-app/frontend/app.py
* Application Tier (Backend): Another Flask application that contains the core business logic for tax calculations. It processes requests from the frontend and interacts with the database.tax-app/backend/app.py
* Data Tier (Database): PostgresSQL database that stores persistent data, such as user information and transaction history.

**CI/CD Pipeline Workflow**

The pipeline is triggered automatically by code changes, following a GitOps methodology. The entire workflow is defined in the ci-cd.yml file within the repository.



**Key Components and Technologies**

**GitHub:** Source code management and CI/CD orchestration via GitHub Actions.

**GitHub Actions:** Automated workflow engine running on self-hosted runners, providing dedicated and flexible execution environments.

**SonarQube:** Static code analysis for quality and security checks.

**Docker:** Containerization of applications for consistent environments.

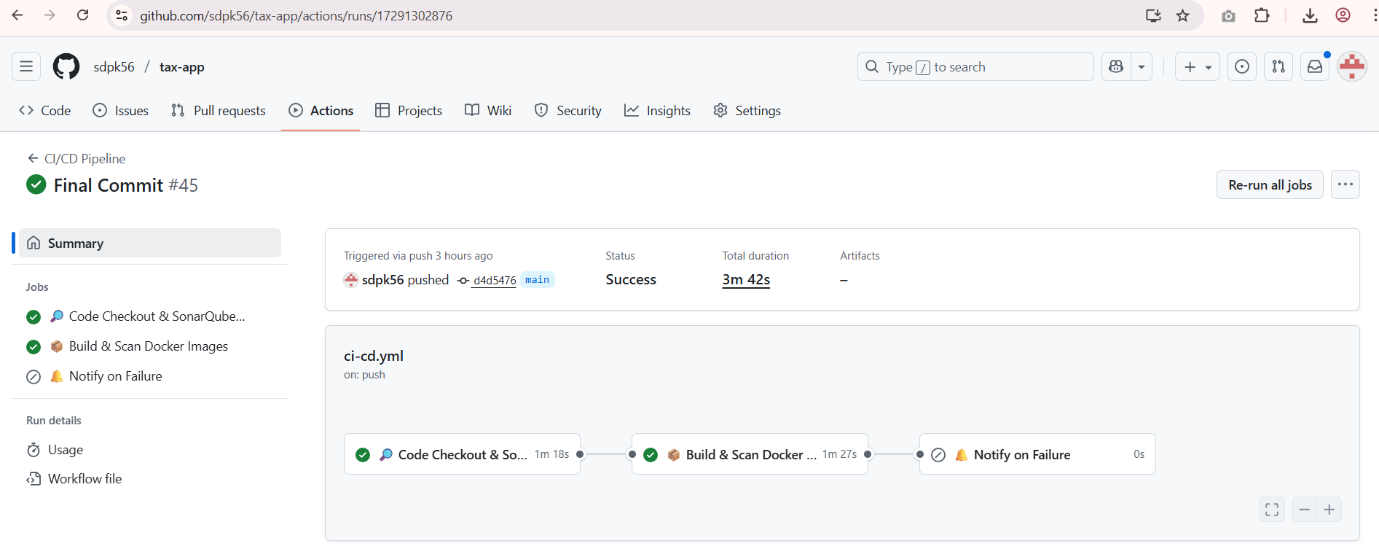
**Docker Hub:** Centralized registry for storing and managing Docker images.

**Kubernetes:** Container orchestration platform for managing and scaling the application.

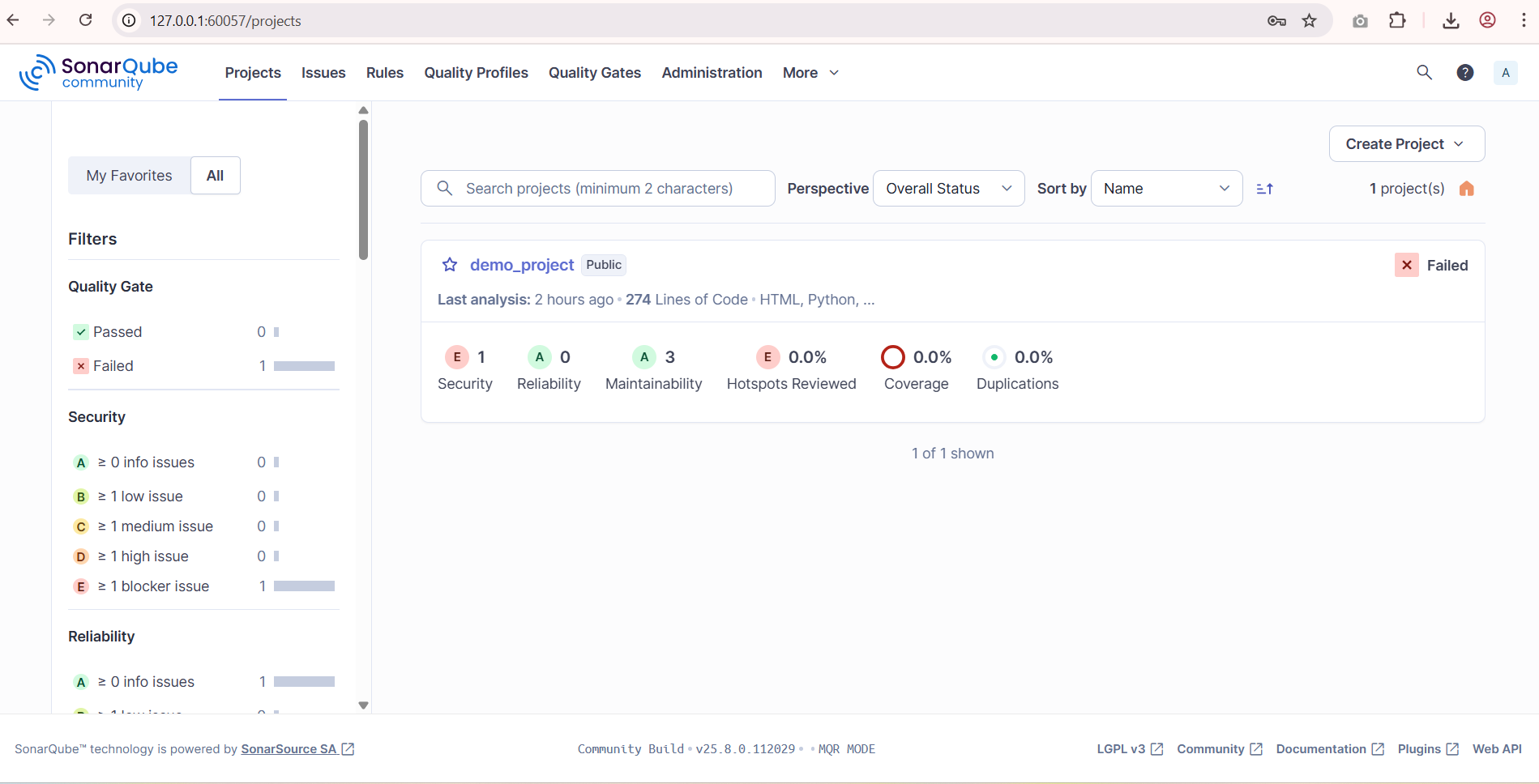
**Helm:** Package manager for Kubernetes, used to define, install, and upgrade applications.

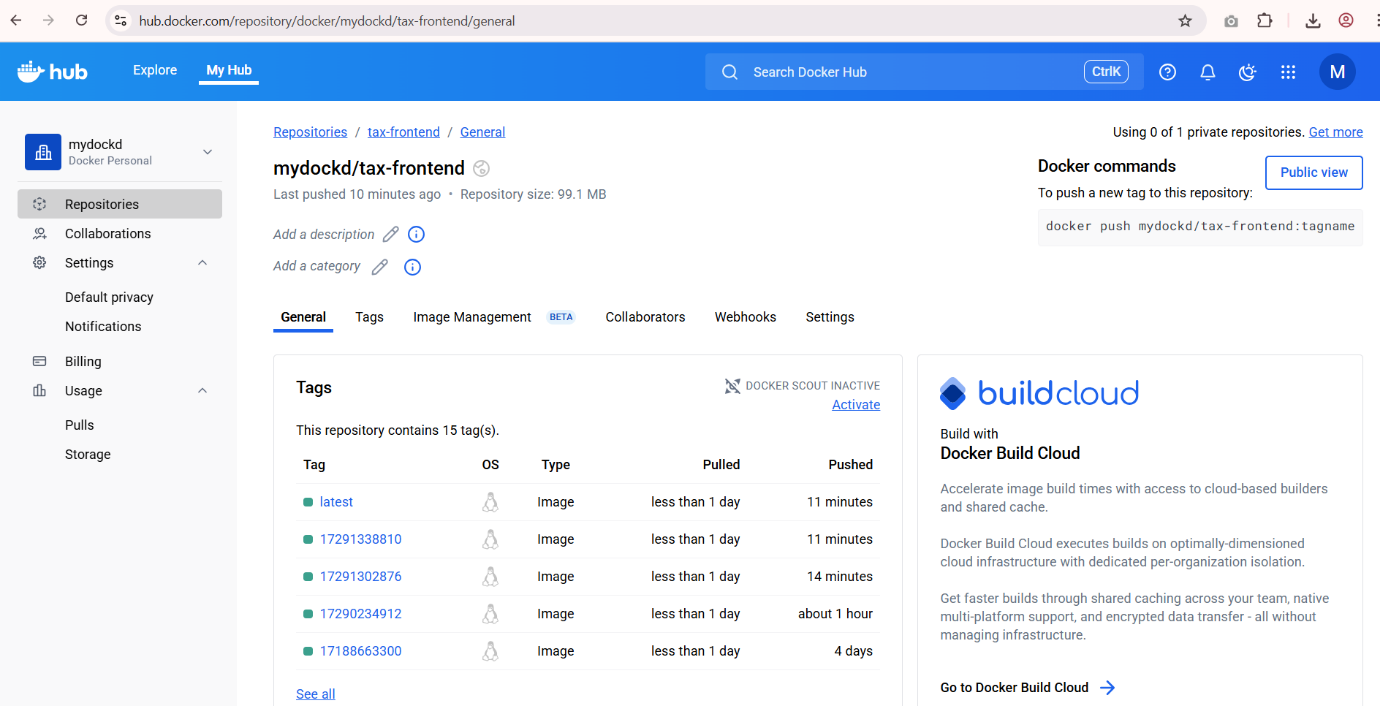
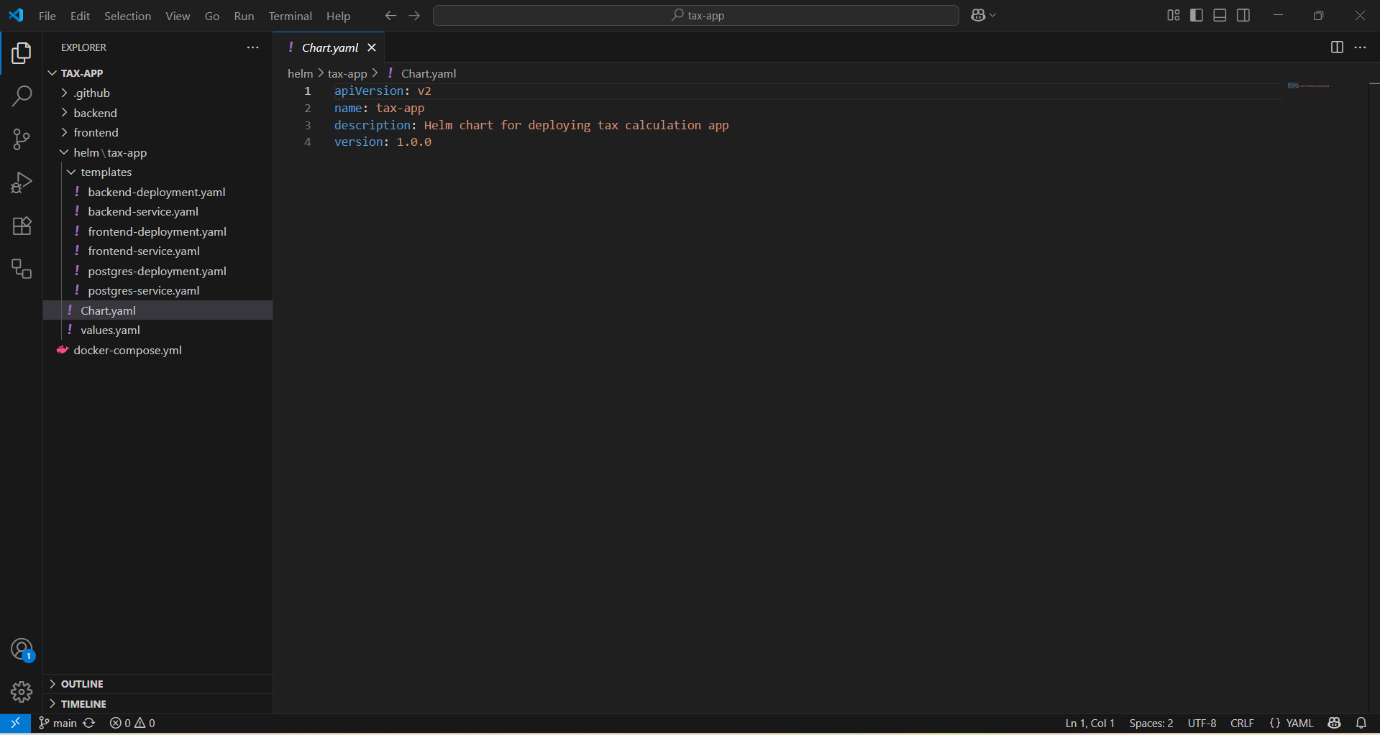
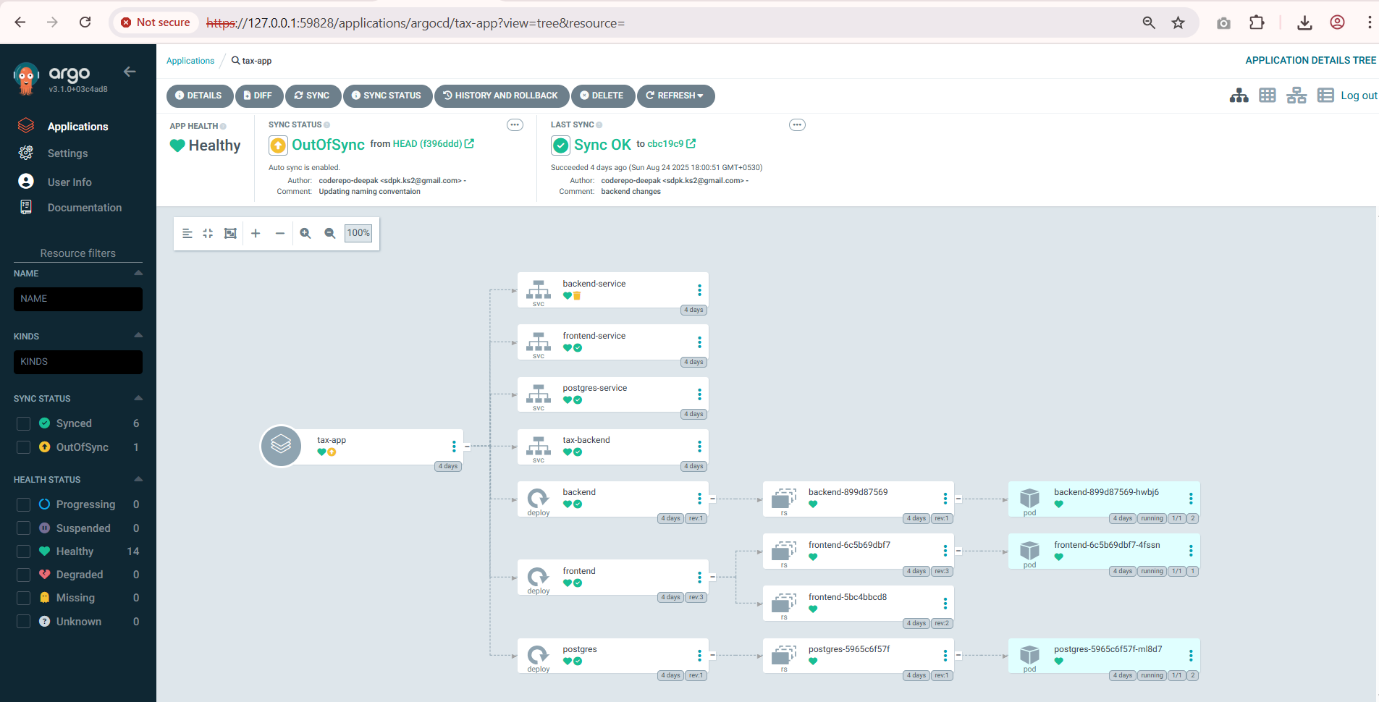
**ArgoCD:** Declarative, GitOps continuous delivery tool for Kubernetes.

Pipeline Steps

1. Code Commit: A developer pushes code to the main branch or opens a pull request.
2. GitHub Actions Trigger: The ci-cd.yml workflow is automatically triggered.
3. SonarQube Code Analysis:

Analyse code quality, identify bugs and security vulnerabilities, and report results to the SonarQube server.



1. Docker Build & Push: The build-and-scan job, dependent on the sonarqube job's success, builds Docker images for both the tax-backend and tax-frontend applications. Each image is tagged with the unique github.run\_id and the latest tag for easy identification. The images are pushed to Docker Hu
2. Helm Charting: New manifest files, packaged as Helm charts, are used to define the Kubernetes resources. The CI/CD process automatically updates the image tags in the Helm values.yaml file after a successful Docker image push and Provides versioned, templated Kubernetes manifests, making deployments standardized and repeatable.
3. ArgoCD Deployment: ArgoCD is configured to monitor the Git repository for changes to the Helm charts. When the image tag is updated, ArgoCD automatically detects the change and synchronizes the desired state with the cluster. The new pods with the updated images are deployed to the Kubernetes cluster and Enforces the principle of Git as the single source of truth, ensuring the cluster state always matches the code in the repository.
4. Failure Notification: The notify-on-failure job sends an email notification if any previous job in the pipeline fails.

Future Enhancements:

* Enhanced Notifications: Integrate with team communication tools like Slack or Microsoft Teams for real-time alerts.
* Automated Testing: Expand the GitHub Actions workflow to include unit, integration, and end-to-end tests.
* Observability: Implement monitoring and logging using tools like Prometheus/Grafana for deeper insights into application health.
* Scalability: Configure Kubernetes Horizontal Pod Autoscaling (HPA) to automatically scale the application based on load.
* Rollback Automation: Further automate rollback strategies using ArgoCD's robust features.