Final Project Presentation

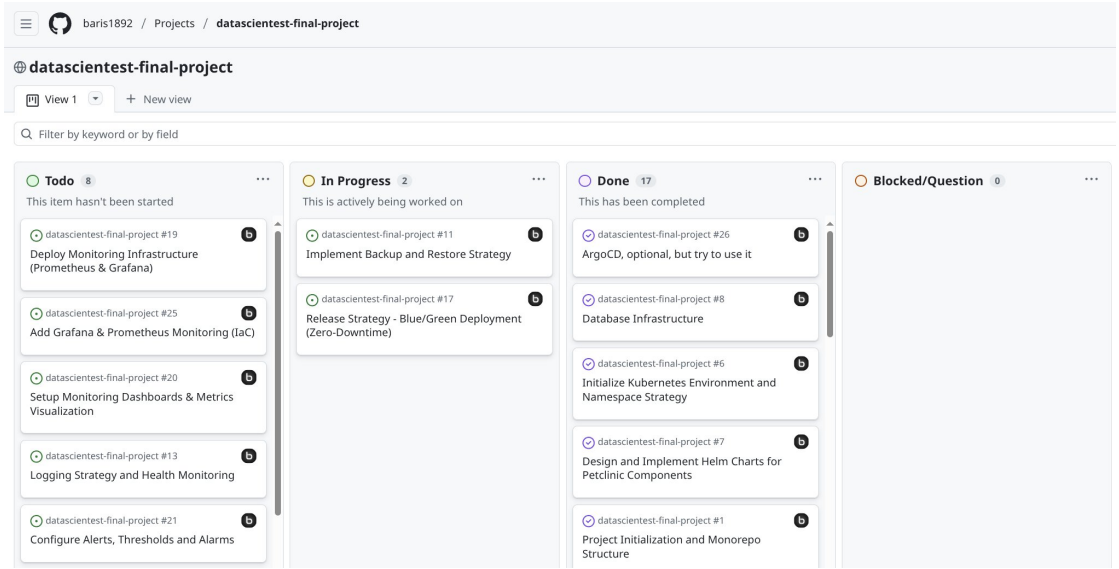
by Baris Top

Contents

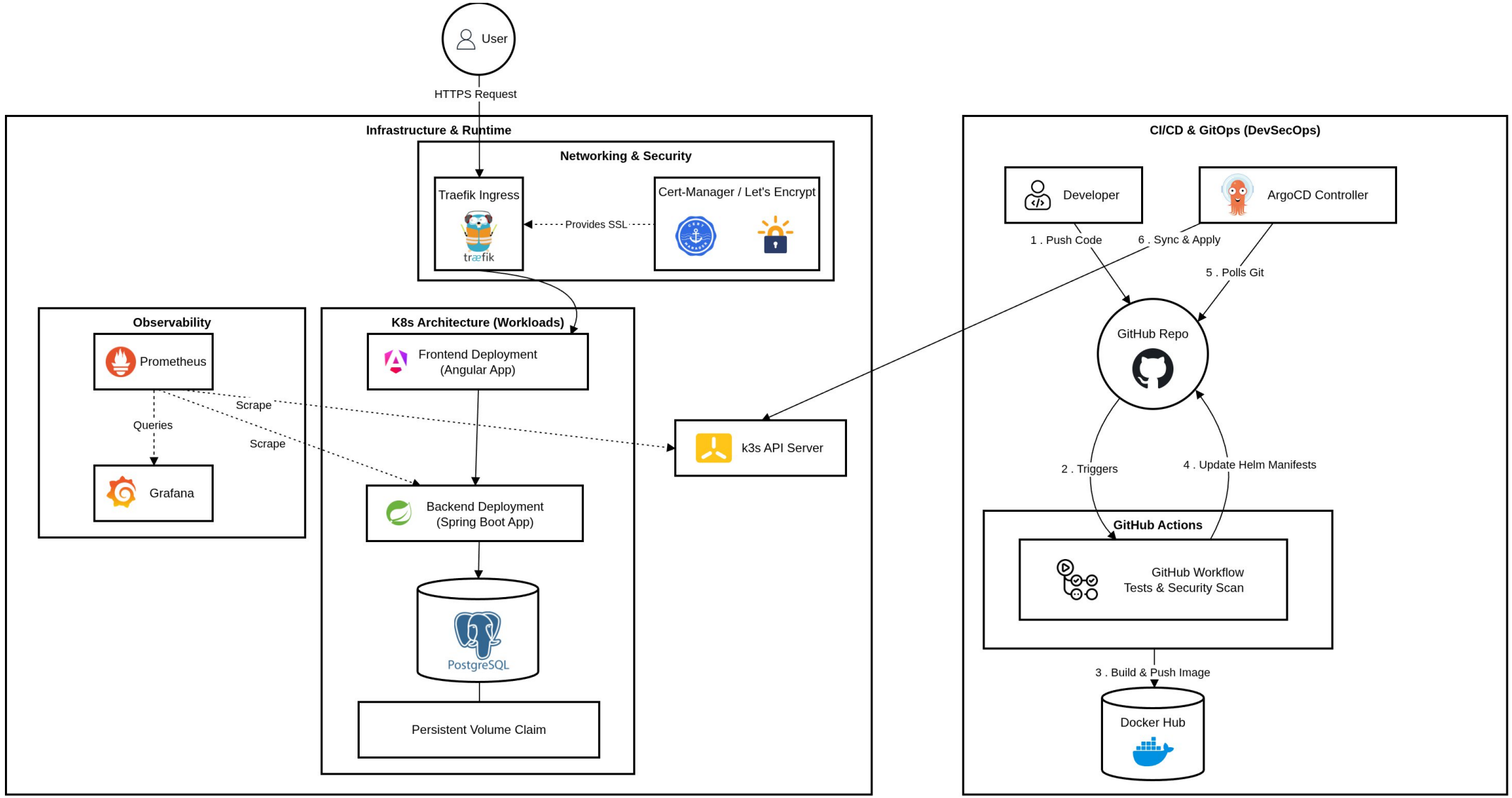
1. Overview of the App
2. Architecture Diagram
3. Tech Stack
4. Infrastructure Provisioning
5. How to deploy dev & prod
6. CI/CD Pipeline & DevSecOps
7. Resilience & Disaster Recovery

1. Overview of the App

- **Foundation**: Utilizing the Spring Petclinic as a baseline.
- **Core Components**: Separation of Angular Frontend, Spring Boot REST API, and a persistent PostgreSQL Database.
- **Key Objectives**: Operational efficiency, independent scalability, and full automation via CI/CD pipelines.
- **Project Management**: Managed with Agile/Kanban methodology using GitHub Issues and Project Boards.



2. Architecture Diagram



3. Tech Stack

- Application & Persistence: 

- Infrastructure & IaC: 

- Deployment & GitOps: 

- Observability & Security: 

4. Infrastructure Provisioning

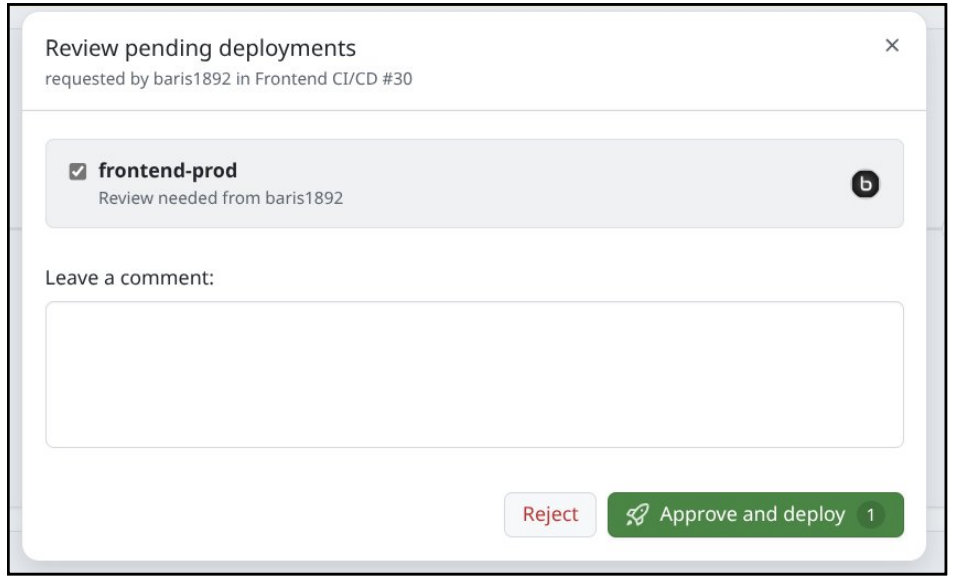
- **Automated Foundations:** Infrastructure as Code via Terraform for global services (Ingress, SSL, Monitoring).

- **Environment Setup:** Isolated namespaces and persistent databases for Dev & Prod.

- **GitOps (ArgoCD) Bootstrap:** Using the App-of-Apps pattern to trigger automatic synchronization of all services.

5. How to deploy dev & prod

- **Environment Isolation:** Strictly separated via Kubernetes Namespaces (dev & prod).
- **Dev Workflow (values-dev.yaml):** Continuous Deployment triggered by every push to the develop branch.
- **Prod Workflow (values-prod.yaml)**: Continuous Delivery via Git Tags (e.g., v1.0.2) and Manual Approval.
- **Dynamic Config (FE, Angular):** Runtime API-URL injection via ConfigMaps (No re-builds required).



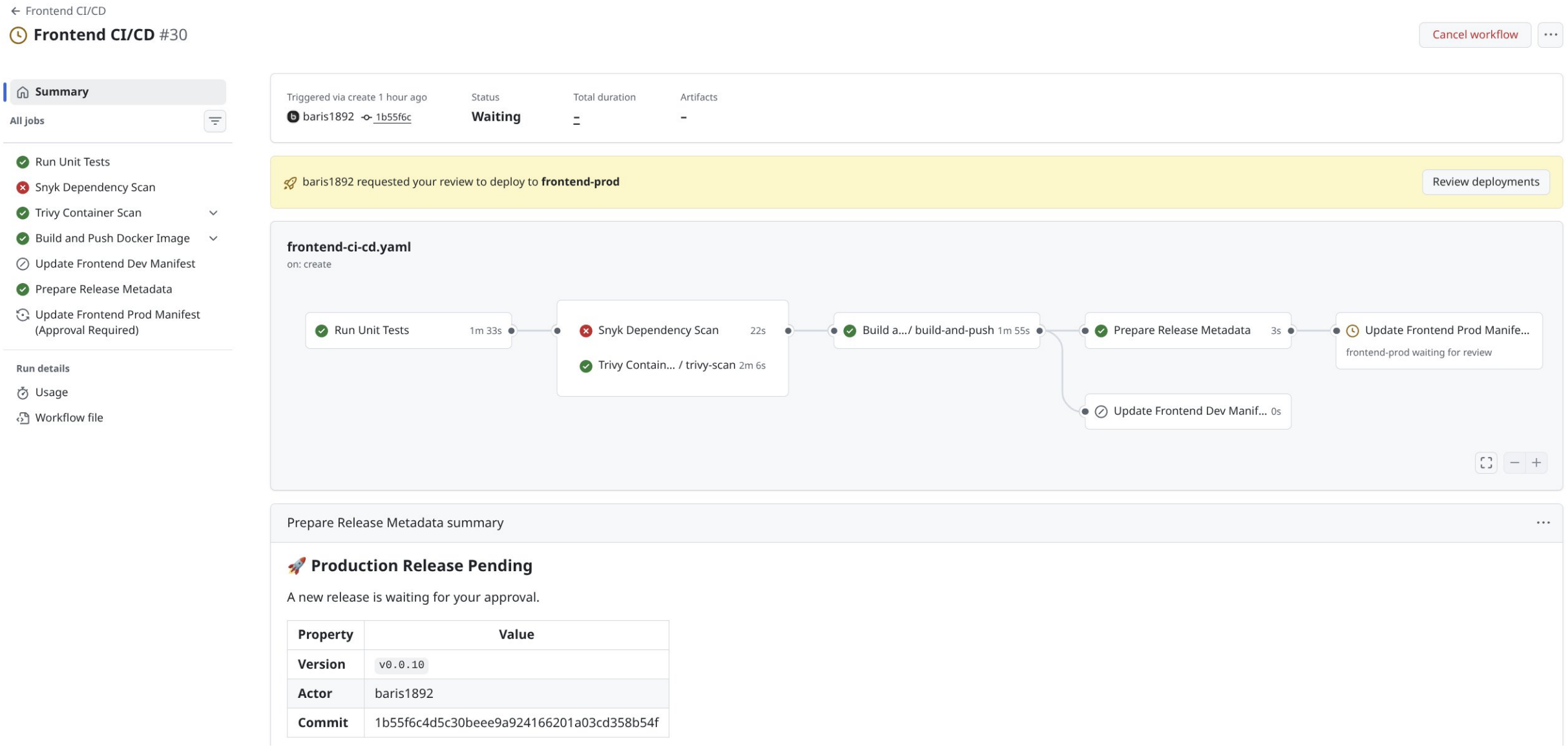
5. How to deploy dev & prod

- **Environment Isolation:** Strictly separated via Kubernetes Namespaces (dev & prod).
- **Dev Workflow (values-dev.yaml):** Continuous Deployment triggered by every push to the develop branch.
- **Prod Workflow (values-prod.yaml)**: Continuous Delivery via Git Tags (e.g., v1.0.2) and Manual Approval.
- **Dynamic Config (FE, Angular):** Runtime API-URL injection via ConfigMaps (No re-builds required).

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: {{ include "frontend.fullname" . }}-config
data:
  env.js: |
    window.__env = {
        REST_API_URL: "{{ .Values.env.REST_API_URL }}",
        ENV_NAME: "{{ .Values.env.ENV_NAME }}"
    };
```

```yaml
# usage in deployment.yaml
spec:
  containers:
    - name: {{ .Chart.Name }}
      volumeMounts:
        - name: frontend-config
          mountPath: /usr/share/nginx/html/assets/env.js
          subPath: env.js
  volumes:
    - name: frontend-config
      configMap:
        name: {{ include "frontend.fullname" . }}-config
```

6. CI/CD Pipeline & DevSecOps



7. Resilience & Disaster Recovery

- **Self-Healing:** Automatic container recovery via Liveness & Readiness probes.
- **Database Backups:** Daily automated `pg_dump` via Kubernetes CronJobs.
- **Infrastructure Recovery:** Full reconstruction using the layered Terraform & GitOps approach.
- **GitOps Rollback:** Instant version reversal by reverting commits in Git.

Demo

- Pipeline in Action
  - Adjust FE text
  - Adjust BE logs

- After Deployment
  - Check ArgoCD UI
  - Check Grafana Logs for adjusted BE logs

- Demonstrate Fault Tolerance & Alerting