Todo API – DevSecOps Project Report

I. Executive Summary

This project demonstrates a **complete DevOps lifecycle** applied to a Todo REST API built with Node.js/Express. The implementation features automated CI/CD pipelines, containerized deployment, Kubernetes orchestration with Helm, GitOps via Argo CD, comprehensive observability (Prometheus/Grafana), and multi-layered security scanning (SAST/DAST). The project exemplifies modern DevSecOps best practices with full automation from code to production.

II. Architecture & Technology Stack

Application: Node.js Express REST API providing CRUD operations for todo management with health endpoints (`/health`), Prometheus metrics (`/metrics`), structured JSON logging (Winston), and correlation IDs for request tracing.

Infrastructure: Docker multi-stage builds create optimized images published to Docker Hub (`siwar10/todo-api:latest`). Kubernetes deployment manages 2 replicas with automated health checks, rolling updates, and NodePort service (port 30080). Helm charts package all resources with configurable values for production deployment.

Technology Stack:

- **Backend**: Node.js 18, Express, Winston, prom-client
- **CI/CD**: GitHub Actions (9-stage pipeline)
- **Container**: Docker, Docker Compose
- **Orchestration**: Kubernetes, Helm
- **GitOps**: Argo CD
- **Observability**: Prometheus, Grafana
- **Security**: npm audit, Trivy, OWASP ZAP
- **Testing**: Jest, Supertest

III. CI/CD Pipeline

9-stage automated pipeline executes on every commit via GitHub Actions:

1. **Install dependencies** → `npm ci`
2. **Run tests** → Jest/Supertest with coverage
3. **npm audit** → Dependency vulnerability scan (high severity)

4. **Trivy FS scan** → Source code static analysis
5. **Docker build** → Create container image
6. **Trivy image scan** → Container vulnerability scan
7. **Push to Docker Hub** → Publish image
8. **Run container** → Start API for testing
9. **OWASP ZAP scan** → Dynamic security testing (DAST)

Security approach: Defense-in-depth with 4 layers (SAST, dependency scan, container scan, DAST) ensuring comprehensive coverage.

IV. Kubernetes & GitOps

Deployment progression:

- **Kubernetes manifests** (`k8s/`): Manual deployment via `kubectl apply`
- **Helm chart** (`charts/todo-api/`): Templated resources with `helm install/upgrade`
- **Argo CD**: GitOps automation with auto-sync and self-healing

Resources: Namespace (todo-app), ConfigMap (environment variables), Deployment (2 replicas, health probes), Service (NodePort 30080), ServiceMonitor (Prometheus discovery).

V. Observability

Three pillars implementation:

1. **Metrics**: Prometheus scrapes `/metrics` endpoint exposing `http_requests_total`, `http_request_duration_seconds`, and Node.js default metrics
2. **Logs**: Winston structured JSON logs with correlation IDs for request tracing
3. **Visualization**: Grafana dashboard with 4 panels (RPS, latency, errors, top endpoints)

Enables rapid troubleshooting, performance optimization, and proactive monitoring.

VI. Key Achievements

Automated 9-stage CI/CD pipeline
  Multi-layered security (npm audit + Trivy + OWASP ZAP)
  Kubernetes HA deployment (2 replicas, health probes)
  Helm packaging for production
  GitOps with Argo CD
  Complete observability stack (Prometheus + Grafana)
  Comprehensive test coverage (Jest/Supertest)
  Production-ready Docker images

VII. Lessons Learned

Automation reduces errors and accelerates deployment. **Security shift-left** prevents vulnerabilities early. **GitOps** simplifies operations with declarative infrastructure. **Observability** (metrics, logs, tracing) is essential for production excellence. **Progressive enhancement** from manifests → Helm → GitOps demonstrates DevOps maturity.

VIII. Conclusion

This project successfully demonstrates modern DevSecOps from development to production. The automated pipeline ensures quality and security, Kubernetes with Helm provides reliability and scalability, Argo CD enables GitOps automation, and comprehensive observability supports operational excellence. The implementation serves as a practical reference for building cloud-native applications with enterprise-grade DevOps practices.

Author: Siwar Ghlissi
Repository: https://github.com/siwarghlissi/Todo-app
Docker Hub: https://hub.docker.com/r/siwar10/todo-api
Date: January 2026