End-to-End CI/CD Pipeline Documentation

1. Overview

This document describes an end-to-end, secure, and automated CI/CD pipeline for deploying cloud-native applications on Kubernetes using GitOps principles. The pipeline integrates infrastructure provisioning, continuous integration, container security, continuous deployment, and observability.

The goal is to achieve:

- Fully automated builds and deployments
- Strong security at every stage
- Declarative, version-controlled deployments
- High scalability and reliability

---

2. High-Level Architecture

Core Flow:

1. Developer pushes code to GitHub
2. Jenkins triggers CI pipeline
3. Code is scanned, built, and containerized
4. Docker image is scanned and pushed to Amazon ECR
5. Deployment manifests are updated in GitHub (GitOps repo)
6. Argo CD syncs changes to Kubernetes
7. Application is exposed via ALB and monitored via Prometheus & Grafana

---

3. Tools & Technologies

Category	Tool
Infrastructure as Code	Terraform
CI Automation	Jenkins
Source Control / GitOps	GitHub
Container Registry	Amazon ECR

Image Scanning	Trivy
Deployment (GitOps)	Argo CD
Container Orchestration	Kubernetes (EKS)
Load Balancing	AWS Application Load Balancer
DNS Management	GoDaddy
Monitoring	Prometheus & Grafana

---

4. Infrastructure Provisioning (Terraform)

Terraform is used to provision and manage:

- Kubernetes cluster (EKS)
- VPC, subnets, and networking
- IAM roles and policies
- ECR repositories
- Application Load Balancer

Benefits:

- Repeatable and version-controlled infrastructure
- Easy environment recreation
- Reduced configuration drift

---

5. CI Pipeline (Jenkins)

Jenkins automates the Continuous Integration process using a Jenkinsfile.

CI Stages

1. **Code Quality Analysis**
   - Static code analysis (e.g., SonarQube or linters)
   - Ensures coding standards and maintainability
2. **Dependency Vulnerability Scan**
   - Scans third-party libraries for known CVEs
3. **File System Scan**
   - Detects sensitive files or misconfigurations
4. **Docker Image Build**
   - Builds application Docker image
5. **Image Vulnerability Scan (Trivy)**
   - Scans Docker image for OS and application vulnerabilities

    - Pipeline fails on critical/high vulnerabilities
6. **Push Image to Amazon ECR**
    - Only scanned and approved images are pushed

---

6. Container Security (Trivy)

Trivy is used for:

- Docker image vulnerability scanning
- Identifying CVEs in OS packages and application dependencies

Security Enforcement:

- CI pipeline blocks image promotion if vulnerabilities exceed thresholds

---

7. GitOps Deployment (GitHub + Argo CD)

GitHub (GitOps Repository)

- Stores Kubernetes manifests (Deployments, Services, Ingress)
- Acts as the single source of truth

Argo CD

- Continuously monitors GitHub for changes
- Automatically syncs Kubernetes cluster state with Git
- Supports rollback and drift detection

Benefits:

- Declarative deployments
- Easy rollback via Git history
- Improved auditability

---

8. Kubernetes Application Architecture

The application is deployed in a **3-tier architecture**:

Frontend Tier

- Exposed via Kubernetes Service and Ingress

- Receives traffic from ALB

Backend Tier

- Internal services communicating with frontend
- Handles business logic

Database Tier

- Isolated pods and persistent volumes
- Secure access using Kubernetes Secrets

---

9. Secrets Management

Secrets are managed securely using Kubernetes Secrets:

- ECR credentials
- Database credentials

Best Practices:

- No hardcoded secrets in code or manifests
- Secrets injected at runtime

---

10. Networking & Traffic Management

- **AWS Application Load Balancer (ALB)** distributes traffic
- **Ingress Controller** routes requests to services
- **GoDaddy DNS** maps custom domain to ALB endpoint

---

11. Monitoring & Observability

Prometheus

- Collects cluster and application metrics
- Monitors CPU, memory, pod health, and service availability

Grafana

- Visualizes metrics via dashboards
- Enables proactive alerting and troubleshooting

12. Key Benefits of This CI/CD Pipeline

- 🔁 Fully automated CI/CD
- 🔐 Security scanning at multiple stages
- 🚀 Faster, reliable deployments
- 📦 GitOps-driven Kubernetes management
- 📊 Strong monitoring and observability
- ♻️ Easy rollback and scalability

---

13. Conclusion

This CI/CD pipeline demonstrates modern DevOps and GitOps best practices for deploying secure, scalable, and observable applications on Kubernetes. It is suitable for production-grade cloud-native workloads and can be extended with additional security, testing, and automation as needed.

End of Document