Project Documentation

Project Title: End-to-End DevOps Pipeline with Secure Image Scanning & Kubernetes Health Automation

Problem Statement

Modern DevOps teams need a robust, automated CI/CD pipeline that not only builds and deploys applications consistently but also ensures container security and Kubernetes cluster reliability. This project sets up an end-to-end pipeline that covers infrastructure provisioning (Terraform), configuration management (Ansible), containerization (Docker), CI/CD orchestration (Jenkins), secure image scanning (Trivy), application deployment on AWS EKS, and automated cluster health monitoring & self-healing using Prometheus and custom scripts.

Project Goals

1. Design and implement an automated CI/CD pipeline on AWS.
2. Provision infrastructure using Terraform (VPC, subnets, EKS, EC2).
3. Automate configuration using Ansible.
4. Build and scan Docker images for vulnerabilities before deploying.
5. Deploy applications on Kubernetes (AWS EKS) for scalability and high availability.
6. Automate health checks and self-healing for Kubernetes clusters.
7. Set up real-time monitoring and alerting with Prometheus, Grafana, and Slack notifications.

Key Tools & Technologies

- CI/CD: Jenkins, GitHub
- Containerization: Docker, AWS ECR
- Infrastructure as Code: Terraform
- Configuration Management: Ansible
- Orchestration: Kubernetes (AWS EKS)

- Image Security: Trivy (Container Vulnerability Scanner)

- Monitoring & Alerts: Prometheus, Grafana, Alertmanager, Slack API

- Programming/Scripting: Bash, Python (for health checker)

Project Sprints

Sprint 1: Project Setup, Dockerization & Jenkins

Tasks:

- Design application architecture for AWS EKS.

- Create Dockerfile & push sample app to AWS ECR.

- Deploy Jenkins on EC2, configure plugins (Docker, K8s CLI, AWS CLI).

- Connect GitHub repo for auto-build triggers.

Goal: Jenkins server + Docker build + Git integration ready.

Sprint 2: Infrastructure Provisioning with Terraform

Tasks:

- Write Terraform scripts for VPC, EKS, EC2.

- Automate Terraform execution in Jenkins.

- Store Terraform state in S3 for remote access.

- Validate infra provisioning end-to-end.

Goal: Reproducible infra on AWS with IaC.

Sprint 3: Configuration Management with Ansible

Tasks:

- Develop Ansible playbooks to configure EC2 nodes (install Docker, kubectl, monitoring agents).

- Add Ansible execution stage in Jenkins pipeline.

- Test configuration flow after infra spin-up.

Goal: Automated and repeatable server config.

Sprint 4: Secure Image Scanning with Trivy

Tasks:

- Integrate Trivy scanner in Jenkinsfile.

- Scan Docker images pre-deployment.

- Fail pipeline if HIGH/CRITICAL CVEs found.

- Send scan results to Slack.

Goal: Shift-left security in CI/CD.

Sprint 5: Application Deployment on EKS

Tasks:

- Create Kubernetes manifests (Deployment, Service).

- Use Jenkins to deploy images to EKS.

- Configure readiness probes, liveness probes, HPA.

- Validate blue-green or rolling deployments.

Goal: Fully automated deploy stage to K8s.

Sprint 6: Cluster Health Checker & Self-Healing

Tasks:

- Deploy Prometheus node & pod exporters.

- Build a simple Python or Go script to auto-check node/pod health.

- Restart pods, reschedule workloads if failures detected.

- Setup CronJob in K8s to run checks.

- Log auto-healing actions.

Goal: Cluster resilience with auto-remediation.

Sprint 7: Monitoring, Alerts & Dashboard

Tasks:

- Install Prometheus & Grafana in EKS.

- Create dashboards for app, infra, cluster health.

- Integrate Prometheus Alertmanager with Slack.

- Add Jenkins job failure alerts to Slack.

Goal: Unified monitoring with proactive alerts.

Sprint 8: Final Testing, Docs & Delivery

Tasks:

- End-to-end test: push -> build -> scan -> infra -> config -> deploy -> monitor.

- Document pipeline stages, config, troubleshooting.

- Record lessons learned & future improvements.

Goal: Production-ready, well-documented DevOps project.

Evaluation Criteria

Documentation: 15%

Implementation: 75%

Cost Optimization: 10%