**Project\_0225\_DevSecOps\_CICD\_Pipeline\_with\_Terraform\_Jenkins\_ArgoCD\_and\_EKS**

“In one of my recent projects, I led the design and implementation of a **DevSecOps CI/CD pipeline** for a customer-facing application running on AWS. The goal was to build a **production-grade, secure, and automated delivery workflow** that balanced speed with compliance.

We started with **infrastructure as code**. Using **Terraform** and an S3 backend for state management, we provisioned the entire environment — including the EKS cluster, IAM roles, networking, and supporting services. IAM keys and AWS CLI configuration were handled centrally, ensuring all Terraform runs were authenticated and auditable. This gave us consistent, repeatable environments across dev, staging, and production.

For the pipeline itself, **Jenkins** was the orchestrator. Every commit to the Git repository triggered a Jenkins job. The first stage ran **SonarQube** scans to enforce static code quality gates — things like cyclomatic complexity, coverage, and security hotspots. Next, we handled **dependency management** with NPM and immediately ran **Trivy file scans** and an **OWASP dependency check** to catch vulnerabilities before they reached the container build stage.

Once dependencies passed, Jenkins built a **Docker image**, tagged it with Git commit SHA, and pushed it to our secure container registry. Before promoting the image, we ran **Trivy image scans**, which was critical in catching base image vulnerabilities early.

The pipeline then updated the **deployment manifests repo** with the new image tag. This is where we brought in **ArgoCD**. By following a GitOps model, ArgoCD continuously reconciled the Git state with the running state on **EKS**. This eliminated the drift problem and ensured that what we had in Git was exactly what was running in production.

For observability, we integrated **Prometheus** and **Grafana** for real-time metrics and dashboards, while **Splunk** provided centralized log aggregation and alerting. This stack gave both engineering and operations teams visibility into application health, deployment rollouts, and security posture.

From a security perspective, the integration of SonarQube, OWASP, and Trivy meant security checks weren’t bolted on at the end — they were **baked into the pipeline**. This shift-left approach reduced vulnerabilities downstream and helped us pass audits with confidence.

The outcome was a fully automated, **end-to-end DevSecOps pipeline**. Developers only needed to commit code, and the pipeline took care of provisioning infrastructure, enforcing quality and security, building and scanning images, deploying with GitOps, and monitoring the workloads in production. This reduced our lead time for changes from weeks to hours, while simultaneously **raising our security and compliance posture**.