**CI/CD Pipeline Implementation for Amazon Prime Website**

**Project Overview**

This project focuses on designing and implementing a robust Continuous Integration and Continuous Deployment (CI/CD) pipeline for the Amazon Prime website. By leveraging industry-standard DevOps tools and best practices, this pipeline ensures automated, efficient, and secure software deployment. The key technologies utilized include GitHub, SonarQube, Aqua Trivy, Docker, Kubernetes, Prometheus, Grafana, ArgoCD, and Terraform.

**Infrastructure Configuration**

* **Jenkins Server Deployment:** The Jenkins server was provisioned using Terraform, ensuring infrastructure-as-code principles. On this server, critical tools such as Helm, Prometheus, SonarQube, Grafana, JDK, Jenkins, and kubectl were installed and configured.
* **EKS Cluster Deployment:** The Amazon Elastic Kubernetes Service (EKS) cluster was also deployed using Terraform, providing a scalable and managed Kubernetes environment.
* **Domain Configuration:** Amazon Route 53 was used to assign a domain name to the deployed application, ensuring accessibility.

**Tools & Technologies Used**

* **Jenkins**: Automates CI/CD workflows
* **SonarQube**: Code quality and security vulnerability scanning
* **Aqua Trivy**: Security vulnerability scanner for containers
* **Docker**: Containerization platform
* **Amazon ECR**: Stores container images
* **ArgoCD**: GitOps-based continuous deployment
* **Prometheus & Grafana**: Monitoring and alerting system

**CI/CD Pipeline Design**

The CI/CD pipeline consists of two primary pipelines: **Build Pipeline** and **Deployment Pipeline**.

**Build Pipeline**

This pipeline automates the software compilation, security scanning, and containerization processes through the following steps:

1. **Git Clone** - Fetches the latest source code from GitHub.
2. **SonarQube Analysis** - Conducts static code analysis to detect vulnerabilities and code quality issues.
3. **Quality Gate Check** - Validates the application against predefined quality thresholds.
4. **NPM Install** - Installs required dependencies for the application.
5. **Aqua Trivy Scan** - Scans application dependencies and container images for vulnerabilities.
6. **Create Docker Image** - Packages the application into a Docker image.
7. **Login to Amazon ECR** - Authenticates to the Amazon Elastic Container Registry (ECR).
8. **Create ECR Repository** - Ensures an ECR repository exists for storing images.
9. **Build and Tag Docker Image** - Builds and tags the Docker image.
10. **Push Image to ECR** - Stores the Docker image in the Amazon ECR repository.
11. **Cleanup Image from Jenkins** - Removes local images from the Jenkins server to optimize storage.

**Deployment Pipeline**

The deployment pipeline ensures seamless application rollout into the Kubernetes environment:

1. **Login to EKS** - Authenticates and gains access to the EKS cluster.
2. **Configure Prometheus and Grafana** - Sets up monitoring and alerting for the application.
3. **Create Kubernetes Container** - Deploys the application container into the cluster.
4. **ArgoCD for Continuous Deployment** - Automates and manages application deployment using ArgoCD.

**Monitoring & Security**

* **Monitoring:** Prometheus and Grafana were integrated to monitor system performance, ensuring stability and uptime.
* **Security:** SonarQube and Aqua Trivy ensure security compliance through vulnerability scanning and code quality checks.

**Cleanup Pipeline**

Upon project completion, a **cleanup pipeline** was executed via Jenkins to decommission all resources, maintaining cost efficiency and environmental hygiene.

**Conclusion**

This project demonstrates the integration of advanced DevOps tools to automate the build, security scanning, deployment, and monitoring of the Amazon Prime website. The CI/CD pipeline enhances deployment efficiency, security, and reliability using Terraform for infrastructure provisioning, Jenkins for automation, ArgoCD for GitOps-based deployments, and Prometheus-Grafana for observability. This approach ensures minimal manual intervention, faster release cycles, and robust application management, setting a benchmark for scalable cloud-native deployments.