Question2:

Write a step-by-step guide on how to set up a CI/CD pipeline using one of the following tools:

Jenkins
GitLab CI/CD

Your pipeline must include parallel test execution and credential management.

Answer:

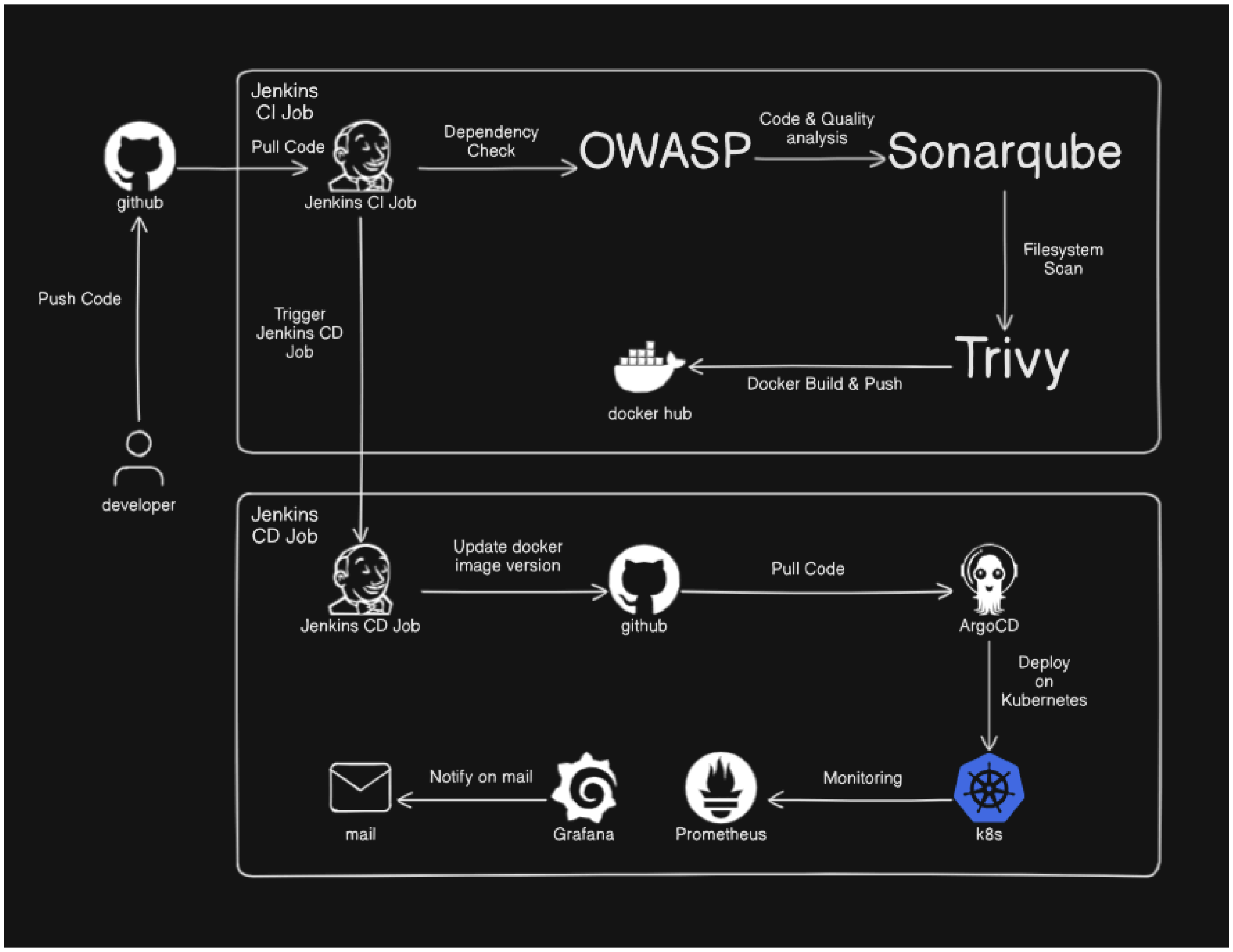

Figure: CI/CD pipeline workflow (Jenkins)

Step-by-Step CI/CD Pipeline Explanation

This CI/CD pipeline is built using **Jenkins**, **OWASP Dependency Check**, **SonarQube**, **Trivy**, **Docker Hub**, **ArgoCD**, **Kubernetes**, **Prometheus**, and **Grafana** for automated deployment and monitoring.

Step 1: Developer Pushes Code to GitHub

- The **developer** writes code and pushes it to the GitHub repository.
- This triggers the **Jenkins CI Job** to start the pipeline.

Step 2: Continuous Integration (CI) Process

Jenkins CI Job

1. **Pull Code from GitHub**
    - Jenkins fetches the latest code from GitHub.
2. **Credential Management using Jenkins Secrets**
    - Sensitive credentials (such as AWS keys, database passwords, and Docker Hub tokens) are securely stored in **Jenkins Credentials Store**.
    - Jenkins retrieves credentials at runtime without exposing them in logs or scripts.
3. **Run Parallel Tests**

    To speed up the process, Jenkins runs multiple tests **simultaneously**:
    - **Security Testing (OWASP Dependency Check)**
    - **Code Quality Analysis (SonarQube)**
    - **Filesystem Security Scan (Trivy)**

Example Parallel Execution in Jenkinsfile:

```
parallel (
    security_scan: {
        sh 'owasp-dependency-check.sh'
    },
    code_quality: {
        sh 'sonar-scanner'
    },
    security_scan_trivy: {
        sh 'trivy filesystem .'
    }
)
```

Step 3: Docker Image Build & Push

- Jenkins builds a **Docker image** using the application code.
- The **Docker image** is then pushed to **Docker Hub** for storage and future deployments.

Step 4: Continuous Deployment (CD) Process

Jenkins CD Job

1. **Trigger Jenkins CD Job**
    - Once the Docker image is built and security scanned, Jenkins triggers the CD pipeline.
2. **Update Docker Image Version in GitHub**
    - The new Docker image version is updated in the GitHub repository, ensuring the latest version is deployed.
3. **ArgoCD Pulls Code from GitHub**
    - ArgoCD fetches the updated deployment configuration from GitHub.
    - It detects changes in the repository and automatically syncs them to Kubernetes.
4. **Deploy Application on Kubernetes**
    - ArgoCD deploys the application to a **Kubernetes cluster** using the updated Docker image.

Step 5: Monitoring & Notifications

1. **Prometheus Monitors Kubernetes**
    - Prometheus continuously monitors the application and infrastructure health.
2. **Grafana Visualizes Metrics**
    - Grafana fetches monitoring data from Prometheus and provides real-time dashboards.
3. **Email Notifications**
    - If any issue occurs or deployment is completed successfully, Jenkins sends a notification email.

CI/CD Workflow Summary

1. **CI Phase (Jenkins CI Job)**
   - Fetch code → Secure credentials handling → Run parallel tests (security, code quality, unit tests) → Build & push Docker image.
2. **CD Phase (Jenkins CD Job + ArgoCD)**
   - Update Docker version in GitHub → ArgoCD pulls changes → Deploy to Kubernetes.
3. **Monitoring & Alerts**
   - Prometheus & Grafana monitor Kubernetes → Email notifications sent for updates or failures.