**GitLeaks & SonarQube Implementation**

**CI/CD Security Stages: Detailed Implementation Notes**

**Gitleaks Stage — Secret Detection**

**Stage Name: gitleaks\_stage**

gitleaks\_stage:

stage: sast\_stage

* **Why**: We assign this job to the sast\_stage stage (even though its name is gitleaks\_stage) because it logically belongs in the security testing phase. This helps organize stages semantically.

**Image and Docker-in-Docker**

image: docker:24

services:

- docker:24-dind

* **Why**: We use the official Docker image (docker:24) because we need to run Docker CLI commands inside the job.
* docker:24-dind (Docker-in-Docker) is added as a service so that we can run Docker containers (like Gitleaks) within the job environment.

**Authentication and Docker Setup**

variables:

CI\_REGISTRY\_USER: beshoynfarag

CI\_REGISTRY\_PASSWORD: dckr\_pat\_H9qEvw8YgvWnAKPaAj6xTYCdUZk

DOCKER\_HOST: tcp://docker:2375

DOCKER\_TLS\_CERTDIR: ""

* **Why**:
  + CI\_REGISTRY\_USER and CI\_REGISTRY\_PASSWORD authenticate the job with Docker Hub or any other registry in case the scan pulls from private images or repos.
  + DOCKER\_HOST sets the communication address between the job container and the Docker daemon.
  + DOCKER\_TLS\_CERTDIR is disabled ("") to simplify the setup (acceptable for controlled runners but not ideal for production).

**Artifacts & Failure Handling**

artifacts:

reports:

secret\_detection: gitleaks-report.json

paths:

- gitleaks-report.json

allow\_failure: true

* **Why**:
  + GitLab will treat the gitleaks-report.json as a **secret detection report**, integrating it into the merge request UI.
  + The allow\_failure: true flag ensures that the pipeline does **not fail** even if secrets are detected—useful in dev environments or while ramping up security practices.

**Script Execution**

script:

- echo "$CI\_REGISTRY\_PASSWORD" | docker login -u "$CI\_REGISTRY\_USER" --password-stdin

- docker pull zricethezav/gitleaks:latest

- docker run --rm -v "$CI\_PROJECT\_DIR":/src -w /src zricethezav/gitleaks:latest detect --source=. --report-path=gitleaks-report.json

* **Line-by-line breakdown**:
  1. **Login to Docker registry** securely using --password-stdin. This avoids hardcoding passwords in the shell history.
  2. **Pull Gitleaks image** from Docker Hub. Gitleaks is a containerized tool; we don’t install it locally.
  3. **Run Gitleaks**:
     + --rm: Automatically clean up the container after it exits.
     + -v "$CI\_PROJECT\_DIR":/src: Mount the current repo into the container for scanning.
     + -w /src: Set working directory inside the container.
     + detect --source=. --report-path=gitleaks-report.json: Perform scan and save output.

**SAST Stage — Static Code Analysis (SonarQube)**

**Stage Definition: sast\_stage**

sast\_stage:

stage: sast\_stage

* **Why**: Declares a SAST-related job in its own logical stage, helping enforce quality/security gates.

image: maven:3.9.0-eclipse-temurin-17

* **Why**: We need Maven and a Java runtime to analyze a Spring Boot project. This official Maven image includes both.

**Artifacts and Job Dependency**

artifacts:

paths:

- "./todolist/target/\*"

expire\_in: 2 hrs

needs: ["gitleaks\_stage"]

* **Why**:
  + Artifacts ensure compiled code and results from testing are retained and optionally passed to downstream jobs.
  + expire\_in controls cleanup to reduce storage use.
  + The needs keyword ensures this job waits for the gitleaks\_stage to finish, enforcing the security order in the pipeline.

**Script Execution**

script:

- curl --fail http://10.97.7.109:9000 || exit 1

- cd ./todolist

- mvn verify sonar:sonar \

-Dsonar.projectKey=todoapp-key \

-Dsonar.projectName=todoapp \

-Dsonar.projectVersion=1.0 \

-Dsonar.sources=src/main/java \

-Dsonar.tests=src/test/java \

-Dsonar.host.url=http://10.97.7.109:9000 \

-Dsonar.login=squ\_4853d36dd8d2fe229ba46e6fb9de41f6cc463495

* **Line-by-line breakdown**:
  1. **curl --fail**: Check if the SonarQube server is up before proceeding. The pipeline exits early on failure.
  2. **cd ./todolist**: Navigate to the Spring Boot project directory.
  3. **mvn verify sonar:sonar**:
     + Runs tests and performs static analysis.
     + Configuration parameters:
       - sonar.projectKey, projectName, projectVersion: Identify the project in the SonarQube UI.
       - sonar.sources, sonar.tests: Specify code and test directories.
       - sonar.host.url: Points to the SonarQube server.
       - sonar.login: Auth token for secure access.