Day - 1

25 November 2025 10:27

Development → Building the code

Operation → Deployment, Mng. the App.

Dev + Ops

↓

Devops:

↓

Shared Responsibilities.

Dev Sec Ops

↓

(Dev + Security + operation)

Shift left Approach:

Development → Code level security

Operation. → CI/CD pipeline, OWASP

Security. → Access, Policy.

L&D → Training, Certification, licenses

① Fixing Quick

② less Expensive

③ less Risky.

Tools

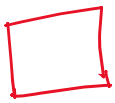
① Jenkins - Automate Pipeline.

② SonarQube ← Sonar Scanner, Quality gate

③ OWASP

④ Trivy.

⑤ Docker & Docker compose.



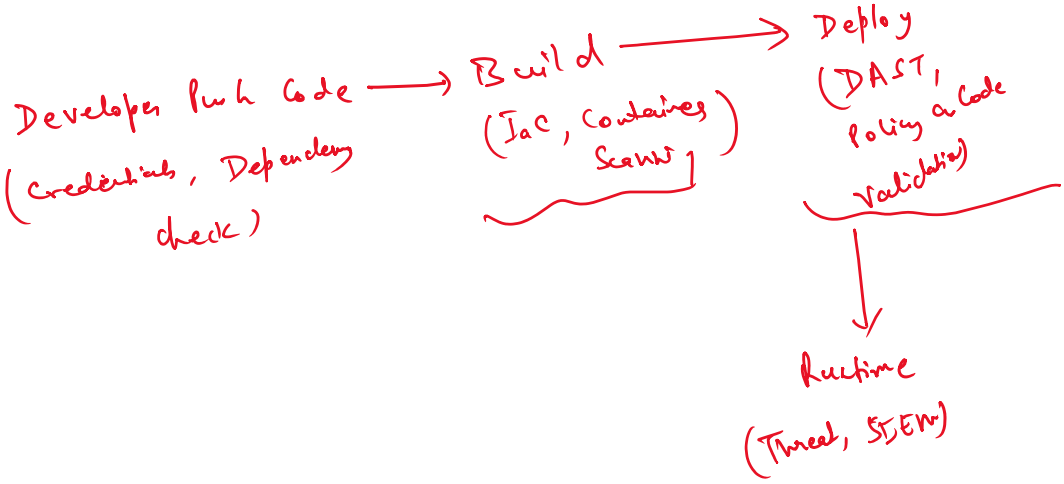
git (GitLab, GitHub, BitBucket, InnerSource)

↓

Version Control App.

(Centralized Repo.)

1. SAST
2. SCA -
3. IaC -
4. Container Security -
5. DAST -
6. Security Scanner -



SCA (Software Composition Analysis) :-

Detect & Protect any kind of attacks due to the 3rd party libraries & dependencies

- Spring boot -
- Node JS
- Python

Detect :-

1. Any Vulnerabilities
2. Malicious lib.
3. open-source / outdated libraries
4. License Issues.

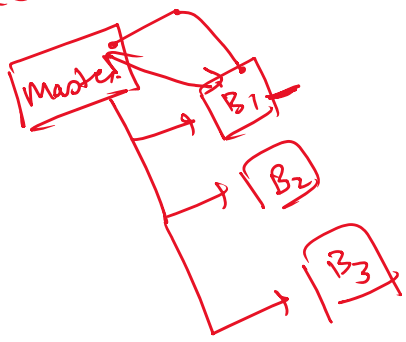
3 stages :-

1. Pre commit - Developer to avoid outdated libraries
   - pipeline - (Pull request)

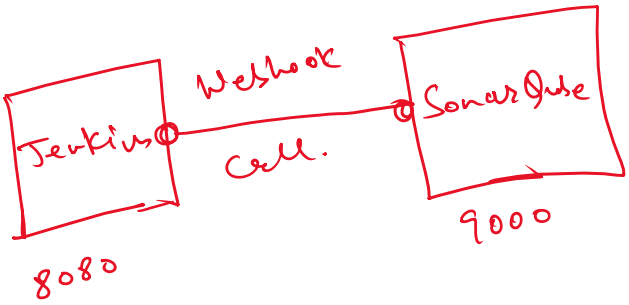
① Pre commit

② CI pipeline -

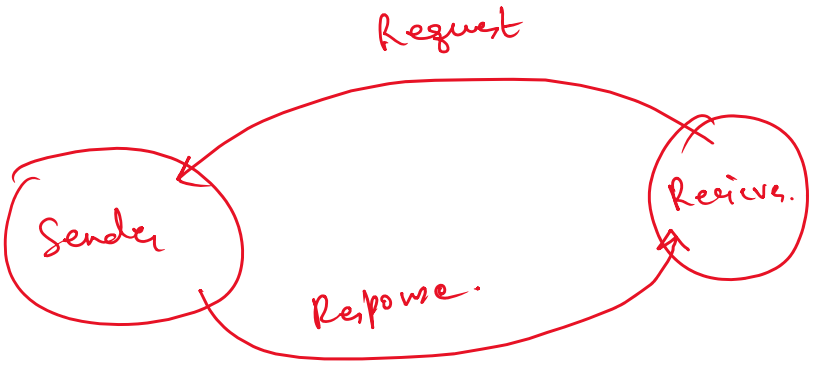
③ PR Review - (Pull request)



Webhook :-



API



Webhook - App. want to send the data to other server real time basis.

SonarQube :- Continous inspection platform.

SAST - Static Application Security Test —

DAST - Dynamic Application Security Testing —

④ Duplication.

① Vulnerability
② Bugs
③ Code Smells
④ Duplication
⑤ Test Coverage

CI - Azure devops, jenkins, gitlab.
SCM - github, VS, Azure Devops (Repos)
Build tool - maven, npm etc.

why?

① Consistent code.
② shift left Security.
③ Automated code review.
④ Quality Gate
⑤ Visibility.

Project - Wardelist.

Issues - ① bug
② Vulnerability
③ Code Smell
④ Security Hotspot.

Severity (major, Blocker, Info)
Location (file, line)
Explanation & Remediation guideline.

Architecture:-

1. SonarQube UI
2. Sonar Scanner
3. Database.

Type of issues it Detects:-

1. Bug - 1. Null pointer Dereference.
   2. Unclosed resources
   3. Logical issue in the code.

2. Vulnerability:-
   1. SQL injection.
   2. Command injection.
   3. XSS.

3. Code Smell:- Piece of code that is not effecting the overall application making it hard to maintain.

   ex- 1. too much Parameter-
   2. Deep Nested loop.
   3. long methods.

4. Security Hotspots:-
   1. encryption of API
   2. Regular expression writing
   3. HTTP headers related to

③ HTTP headers related to security.

Jenkins:- open-source Automation Server.

① CI

② CD

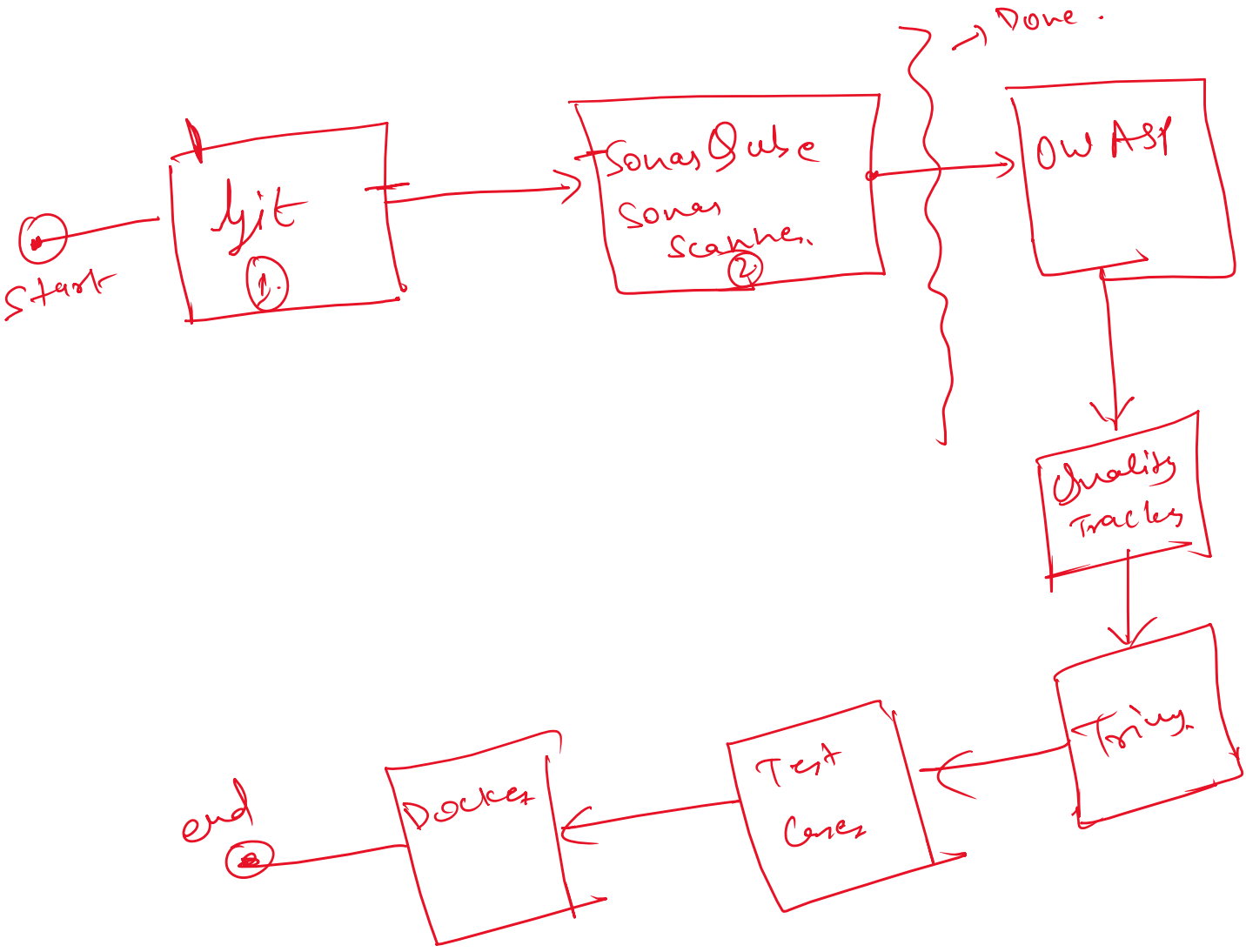
③ Automate test

Pipeline → Simple (outdated)

→ Declarative Pipeline.

```
pipeline {
    agent any
    environment
    Variable
    stage {
        stage('_') {
            steps {
            }
        }
    }
}
```

→ → Done.


Done.
Start
Git
1
SonarQube
Sonar
Scanner.
2
OWASP
Quality
Tracker
Trivy.
Test
Cases
Docker
end
3