

# Secure Coding



**[https://github.com/up1/  
course-secure-coding](https://github.com/up1/course-secure-coding)**



# Secure Coding

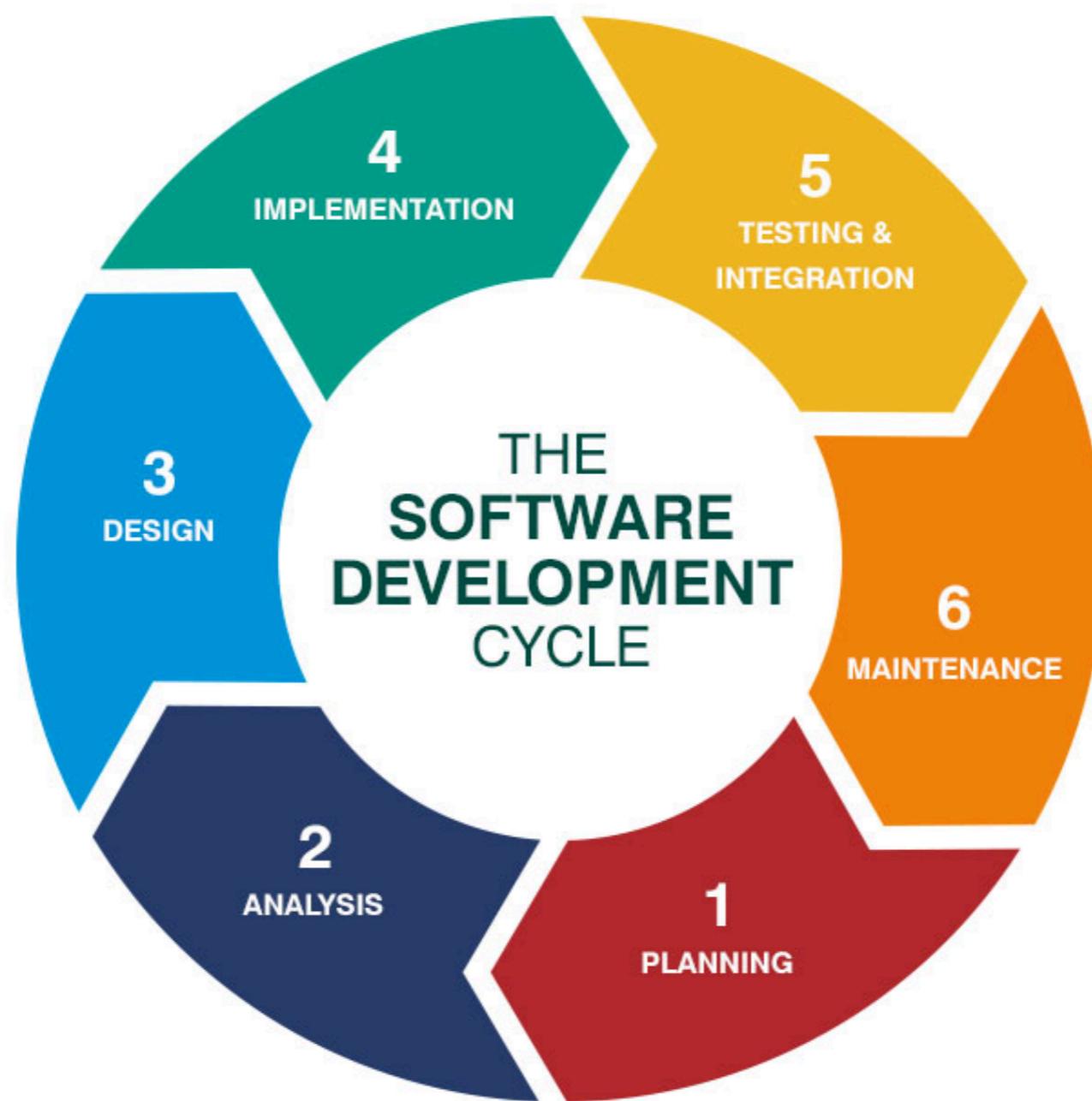


# Software Delivery

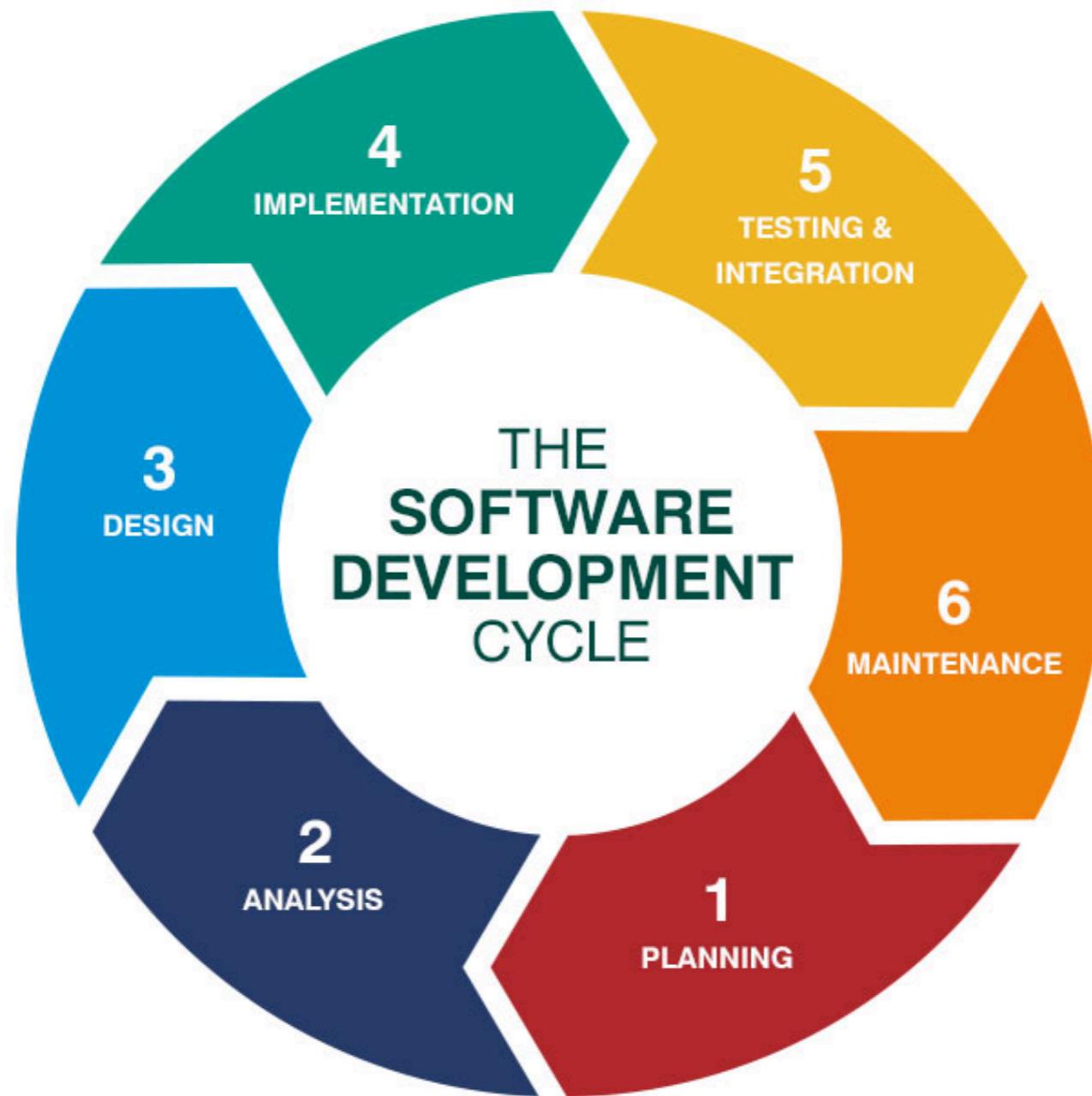
Fast (Time to market)  
High quality  
High secure system



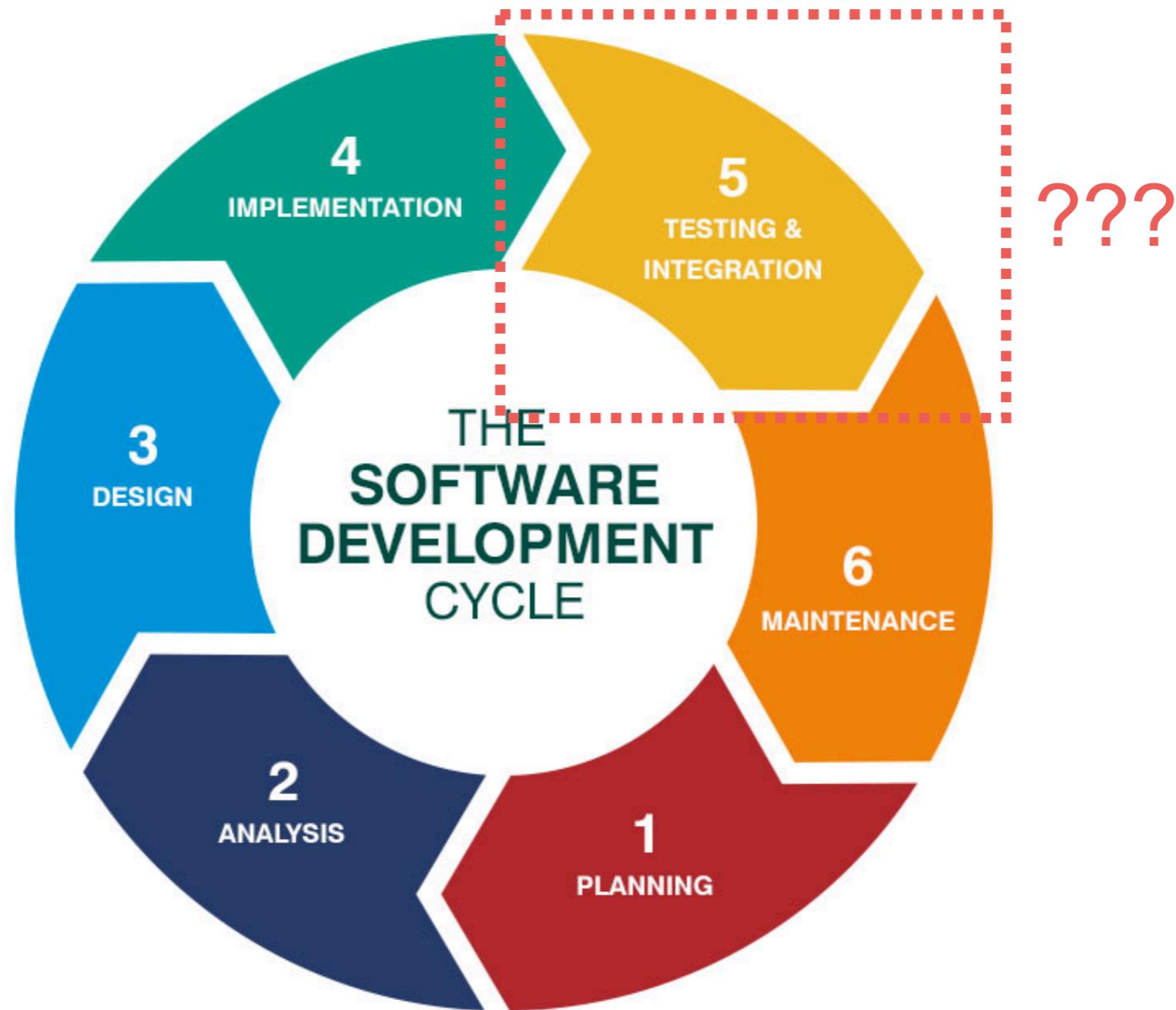
# Software Development Life Cycle



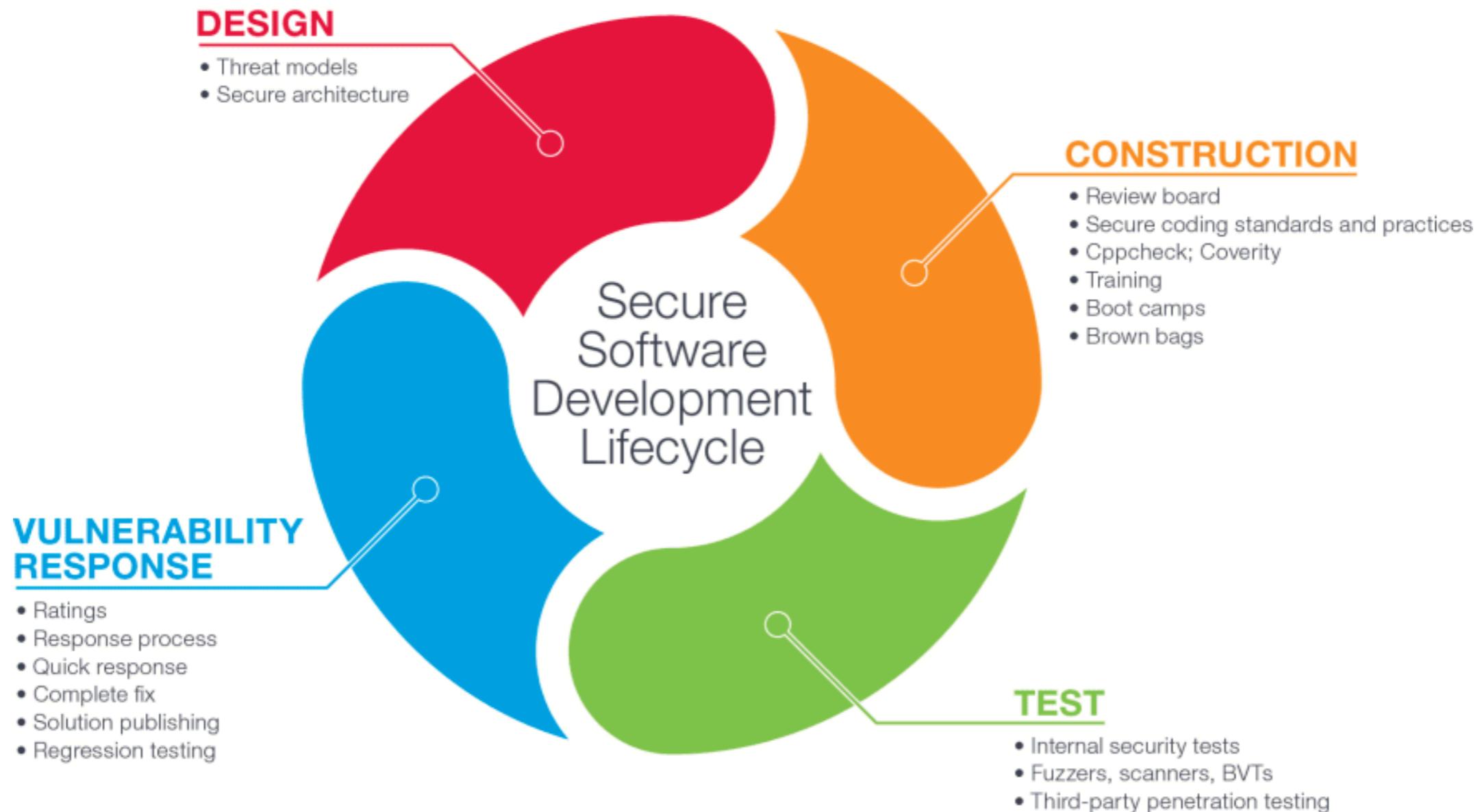
# Security Testing ?



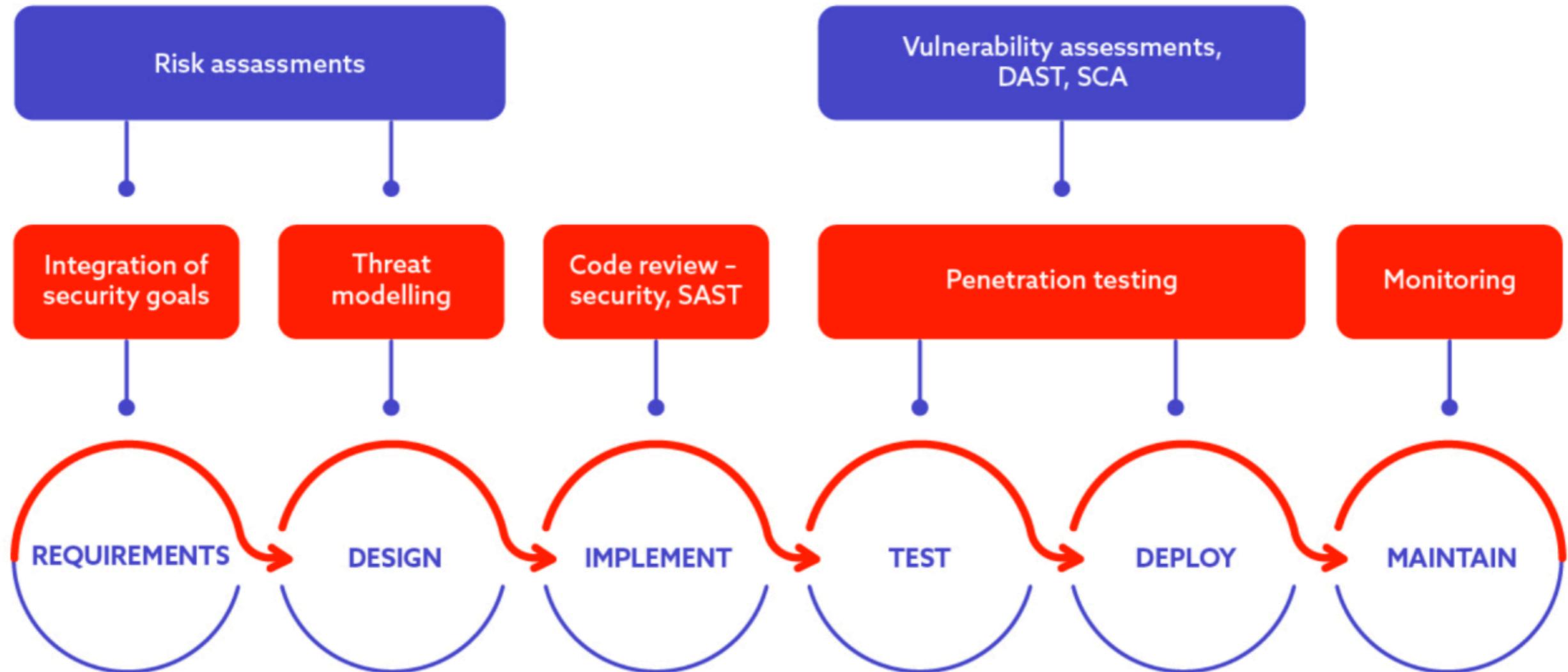
# Security Testing ?



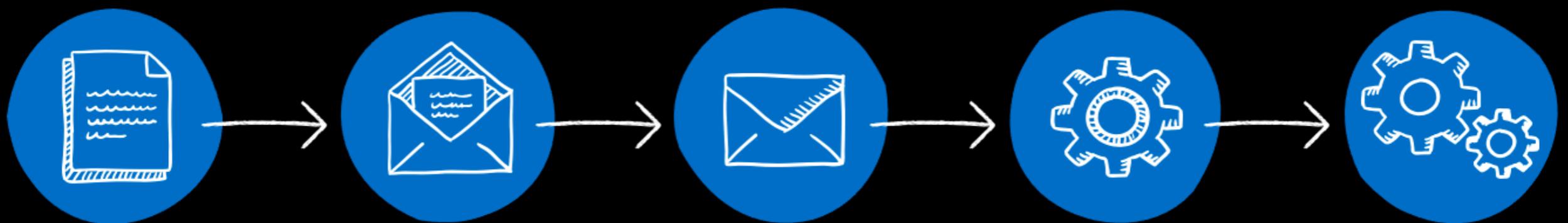
# SDLC with Secure



# SDLC with Secure



# SDLC with Secure



## Plan and Develop

- Threat modelling
- IDE Security plugins
- Pre-commit hooks
- Secure coding standards
- Peer review

## Commit the code

- Static application security testing
- Security unit and functional tests
- Dependency management
- Secure pipelines

## Build and test

- Dynamic application security testing
- Cloud configuration validation
- Infrastructure scanning
- Security acceptance testing

## Go to production

- Security smoke tests
- Configuration checks
- Live Site Penetration testing

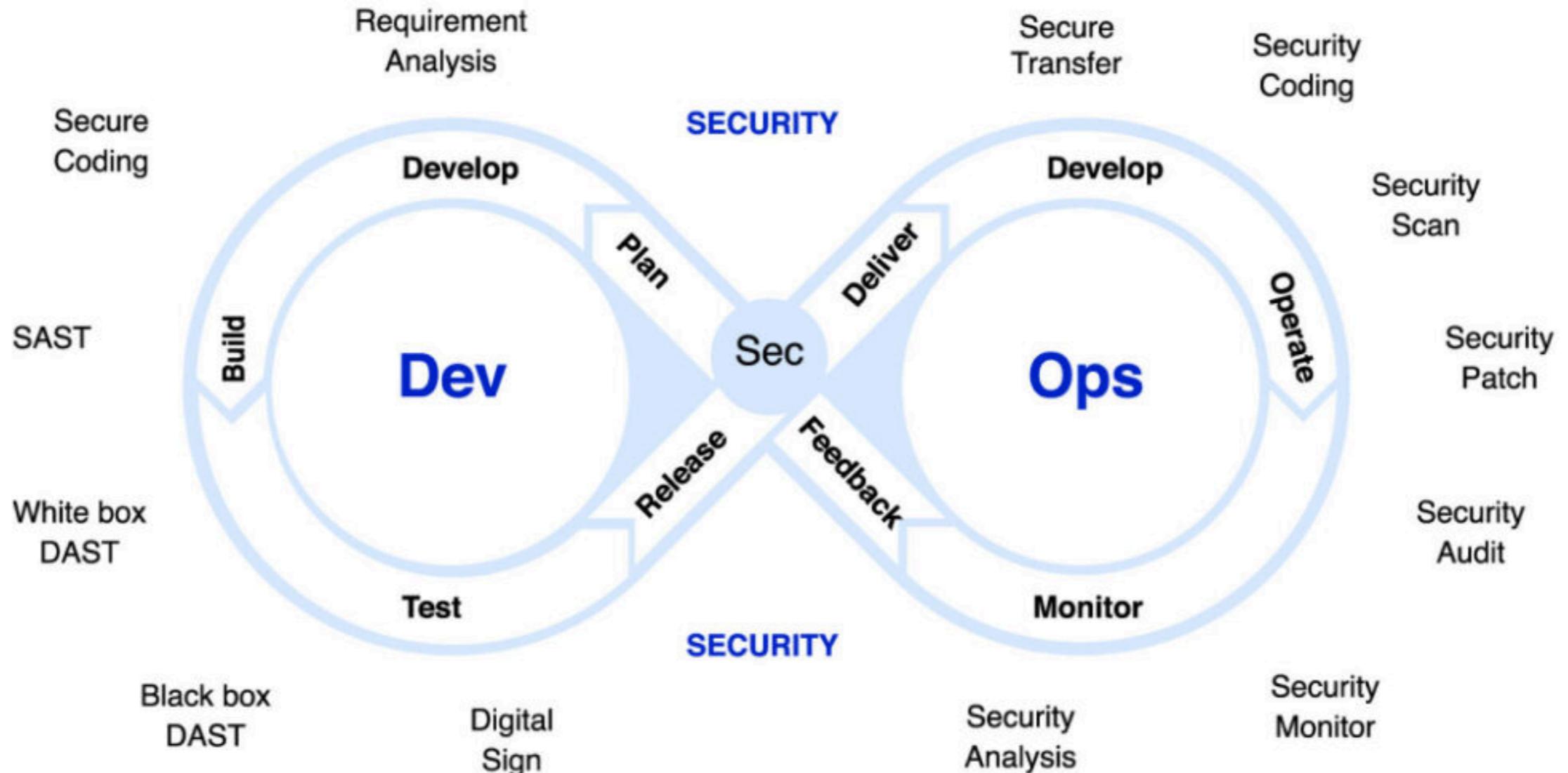
## Operate

- Continuous monitoring
- Threat intelligence
- Penetration testing
- Blameless postmortems

<https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/secure/devsecops-controls>



# DevSecOps



# Goals

Security review

Reduce damage

Reduce cost

Early detection



# Secure in delivery process



# Secure in Delivery Process

Requirements	Planning & Design	Development	Testing / Pre-Deployment	Deployment	Sanitization & Disposal
<p>Security Requirements to be shared such as checklist, Secure coding standards, Application Security guidelines, etc.</p> <p>Awareness sessions to developers</p>	<p>Review Design documents, System Analysis and Architecture review, Privacy Impact Assessment</p>	<p>Threat Modelling, Document Security Controls, Architecture review, Code review Documentation</p>	<p>DAST, SAST, Security/Penetration Testing, VAPT, MBSS compliance Check, Container Security guidelines check, API Security Testing</p>	<p>Conduct a final Security review, Integrate other security systems such as SIEM, IAM etc.</p>	<p>ICT Responsibilities: Disposal plan, media Sanitization, Closure of system</p>
<p>Tools: Manual (Documents need to be shared)</p>	<p>Tools: Manual</p>	<p>Tools: TM Tools, code review tools and manual</p>	<p>Tools: Burp suite, automated scanners, code scanners, VA Scanners, PT Tools, Compliance Scanners</p>	<p>...</p>	



# 1. Requirements

Share security requirement

Security  
Checklist

Secure  
Coding  
standard

Application  
security  
guidelines



# Secure Coding Standard

Avoid vulnerabilities  
Focus on Web and API security

OWASP Top  
10

SANS  
20 software  
errors



# OWASP Top 10

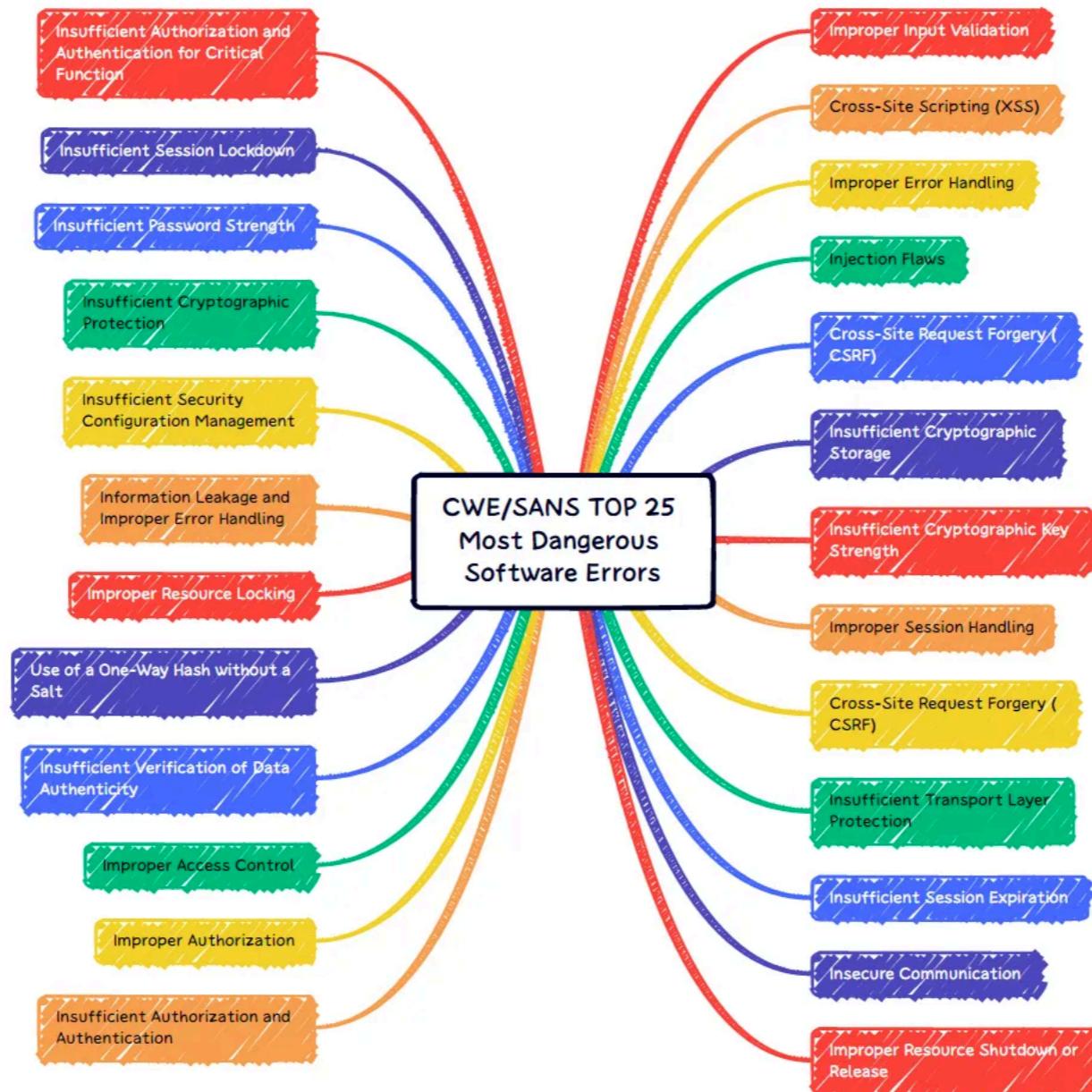


<https://owasp.org/API-Security/editions/2023/en/0x11-t10/>



# CWE/SANS Top 25 Errors

## Common Weakness Enumeration



<https://cwe.mitre.org/top25/>



# More ...

Access administration  
Authentication and authorization  
Logging and monitoring system





# Top 10



<https://owasp.org/Top10/>



# A01 :: Broken Access Control

[https://owasp.org/Top10/A01\\_2021-Broken\\_Access\\_Control/](https://owasp.org/Top10/A01_2021-Broken_Access_Control/)



# A01 :: Broken Access Control

Violation of the principle of least privilege

Bypassing access control

Access API with mission access control

CORS misconfiguration

CORS allows untrusted origin



# A01 :: Broken Access Control

Expose sensitive information to unauthorized user  
Insertion of sensitive information to sent data  
Cross-Site Request Forgery (CSRF)



# Required

User should only act based on specific permission  
Incorrect permission -> unauthorized access



# How to prevent ?

Deny access by default

Avoid duplication of access control log

**Enforce user ownership when manipulating data**



# Workshop with NodeJS

## Broken Access Control !!

GET /profile?username=alice

GET /profile?username=bob



# Potential Security Issues

@nearform/sql

SQL Injection

Authentication

Input validation

Data exposure

Dependency security

Error handling



# Solutions !!

Create and run automated test  
Check JWT token in HTTP request header

Multiple factor  
authentication  
(MFA)

Strong  
password

Log all  
failed  
attempt



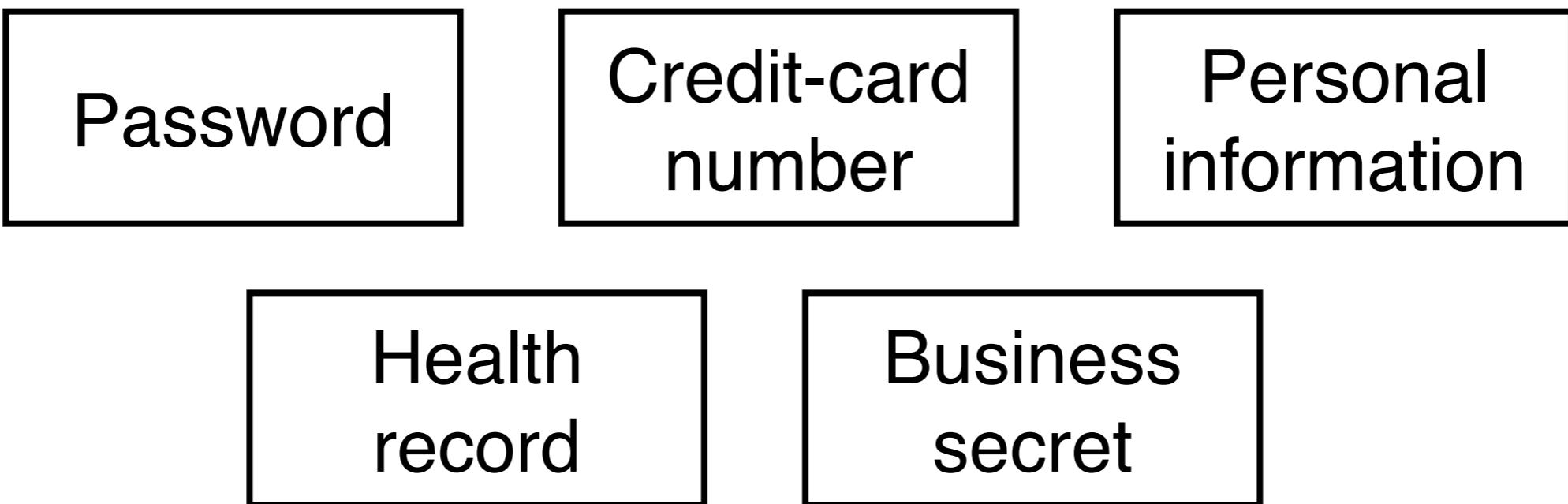
# A02 :: Cryptographic Failure

[https://owasp.org/Top10/A02\\_2021-Cryptographic\\_Failures/](https://owasp.org/Top10/A02_2021-Cryptographic_Failures/)



# A02 :: Cryptographic Failure

Weak or nonexistent **cryptography** of sensitive data  
Anything protected by privacy laws or regulations



# Common mistake !!

Weak or outdated cryptographic algorithm



# Common mistake !!

Weak secret keys, use default

**Use keys from online tutorials**

Lack of traffic encryption (HTTPS)

Insufficient entropy in seed generation



# How to prevent ?

Check all sensitive data is encrypted

Avoid storing sensitive data unnecessarily

Use up-to-date and strong standard algorithms

Proper key-secret management

Disable caching for response that contain sensitive  
data

***Don't store private keys in Git***



# Workshop with NodeJS

Weak hashing algorithm such as md5

```
let hashPassword = async function hashPassword(password) {  
    return md5(password)  
}  
  
let comparePassword = async function comparePassword(password, hash) {  
    return md5(password) === hash  
}
```

***Please change to Bcrypt***



# Potential Security Issues

Weak hashing  
algorithm

Lack of salt

Plain text  
password



# MD5 !!

## Security [ edit ]

One basic requirement of any cryptographic hash function is that it should be [computationally infeasible](#) to find two distinct messages that hash to the same value. MD5 fails this requirement catastrophically. On 31 December 2008, the [CMU Software Engineering Institute](#) concluded that MD5 was essentially "cryptographically broken and unsuitable for further use".<sup>[17]</sup> The weaknesses of MD5 have been exploited in the field, most infamously by the [Flame malware](#) in 2012. As of 2019, MD5 continues to be widely used, despite its well-documented weaknesses and deprecation by security experts.<sup>[18]</sup>

A [collision attack](#) exists that can find [collisions](#) within seconds on a computer with a 2.6 GHz Pentium 4 processor (complexity of  $2^{24.1}$ ).<sup>[19]</sup> Further, there is also a [chosen-prefix collision attack](#) that can produce a collision for two inputs with specified prefixes within seconds, using off-the-shelf computing hardware (complexity  $2^{39}$ ).<sup>[20]</sup> The ability to find collisions has been greatly aided by the use of off-the-shelf [GPUs](#). On an NVIDIA GeForce 8400GS graphics processor, 16–18 million hashes per second can be computed. An NVIDIA GeForce 8800 Ultra can calculate more than 200 million hashes per second.<sup>[21]</sup>

<https://en.wikipedia.org/wiki/MD5#Security>



# Use Bcrypt

```
import { hash, compare } from 'bcrypt'

const saltRounds = 10

export async function hashPassword(password) {
  return await hash(password, saltRounds)
}

export async function comparePassword(password, hash) {
  return await compare(password, hash)
}
```

<https://github.com/kelektiv/node.bcrypt.js>



# Testing for Weak Cryptography

Weak transport layer security (TLS)

Padding oracle

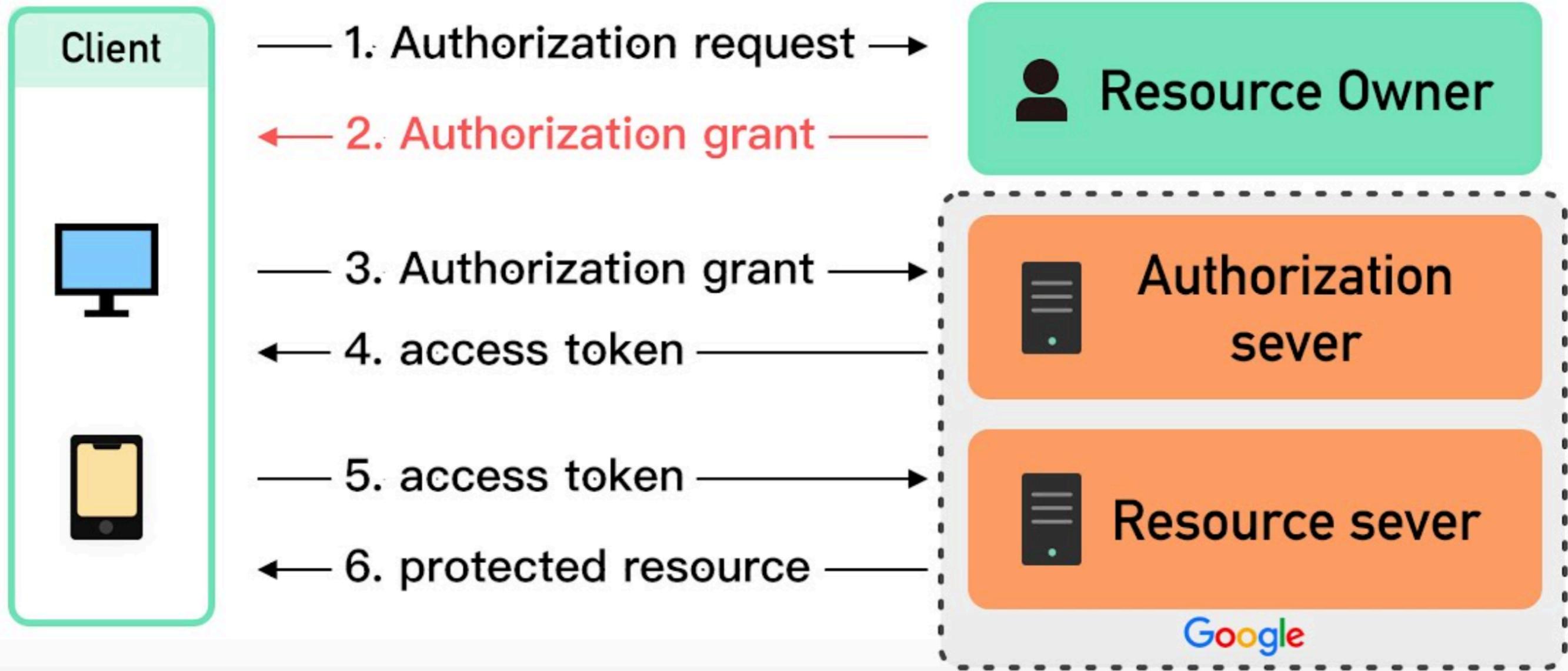
Sensitive information sent via unencrypted channel

Weak encryption

[https://owasp.org/www-project-web-security-testing-guide/stable/4-Web\\_Application\\_Security\\_Testing/09-Testing\\_for\\_Weak\\_Cryptography/README](https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/09-Testing_for_Weak_Cryptography/README)



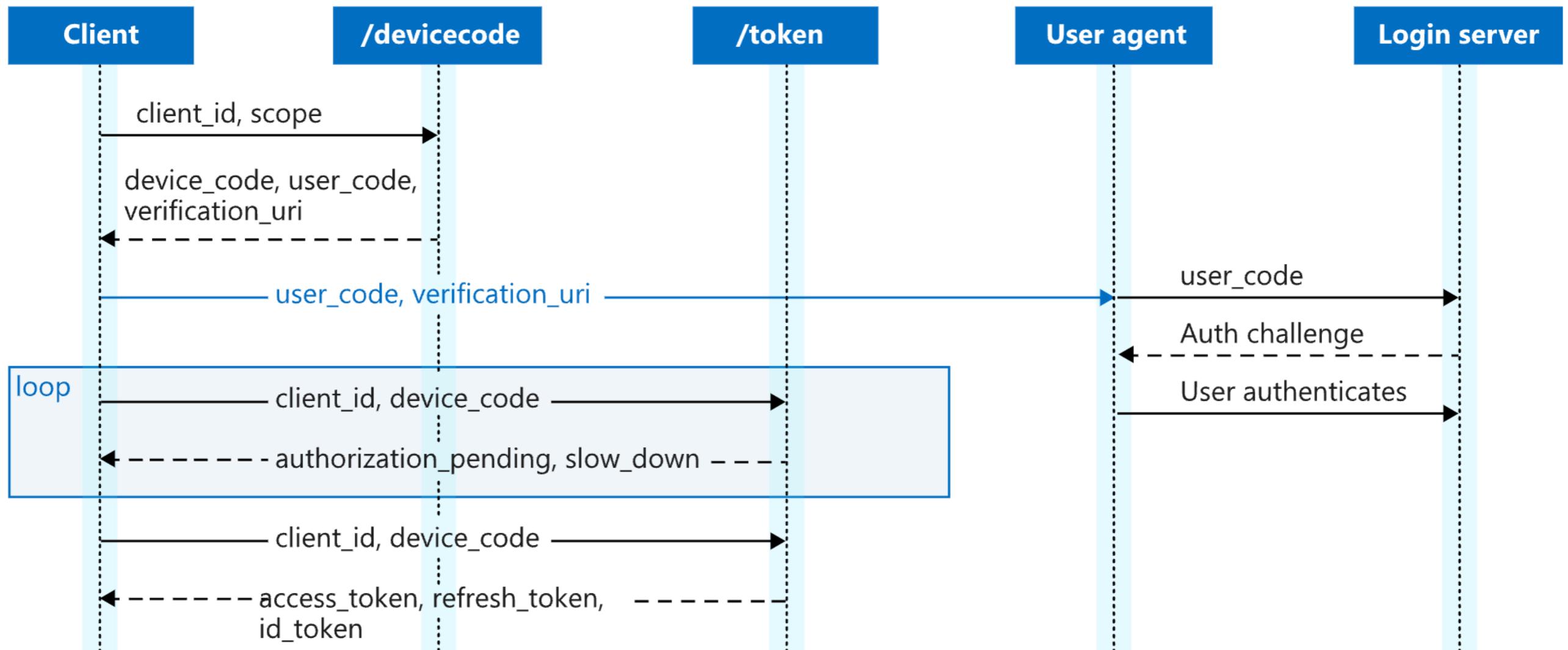
# Use OAuth 2



<https://oauth.net/2/>



# Use OAuth 2 from Microsoft



<https://github.com/kelektiv/node.bcrypt.js>



# A03 :: Injection

[https://owasp.org/Top10/A03\\_2021-Injection/](https://owasp.org/Top10/A03_2021-Injection/)



# A03 :: Injection

Cross-Site Scripting (XSS)  
SQL injection

External control of filename or path



# A03 :: Injection

Lack to validate user's input

Validated

Filters

Sanitized



# Input Validate Cheat Sheet

## Input Validation Cheat Sheet

### Introduction

This article is focused on providing clear, simple, actionable guidance for providing Input Validation security functionality in your applications.

### Goals of Input Validation

Input validation is performed to ensure only properly formed data is entering the workflow in an information system, preventing malformed data from persisting in the database and triggering malfunction of various downstream components. Input validation should happen as early as possible in the data flow, preferably as soon as the data is received from the external party.

[https://cheatsheetseries.owasp.org/cheatsheets/Input\\_Validation\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html)



# A04 :: Insecure Design

[https://owasp.org/Top10/A04\\_2021-Insecure\\_Design/](https://owasp.org/Top10/A04_2021-Insecure_Design/)



# A05 :: Security Misconfiguration

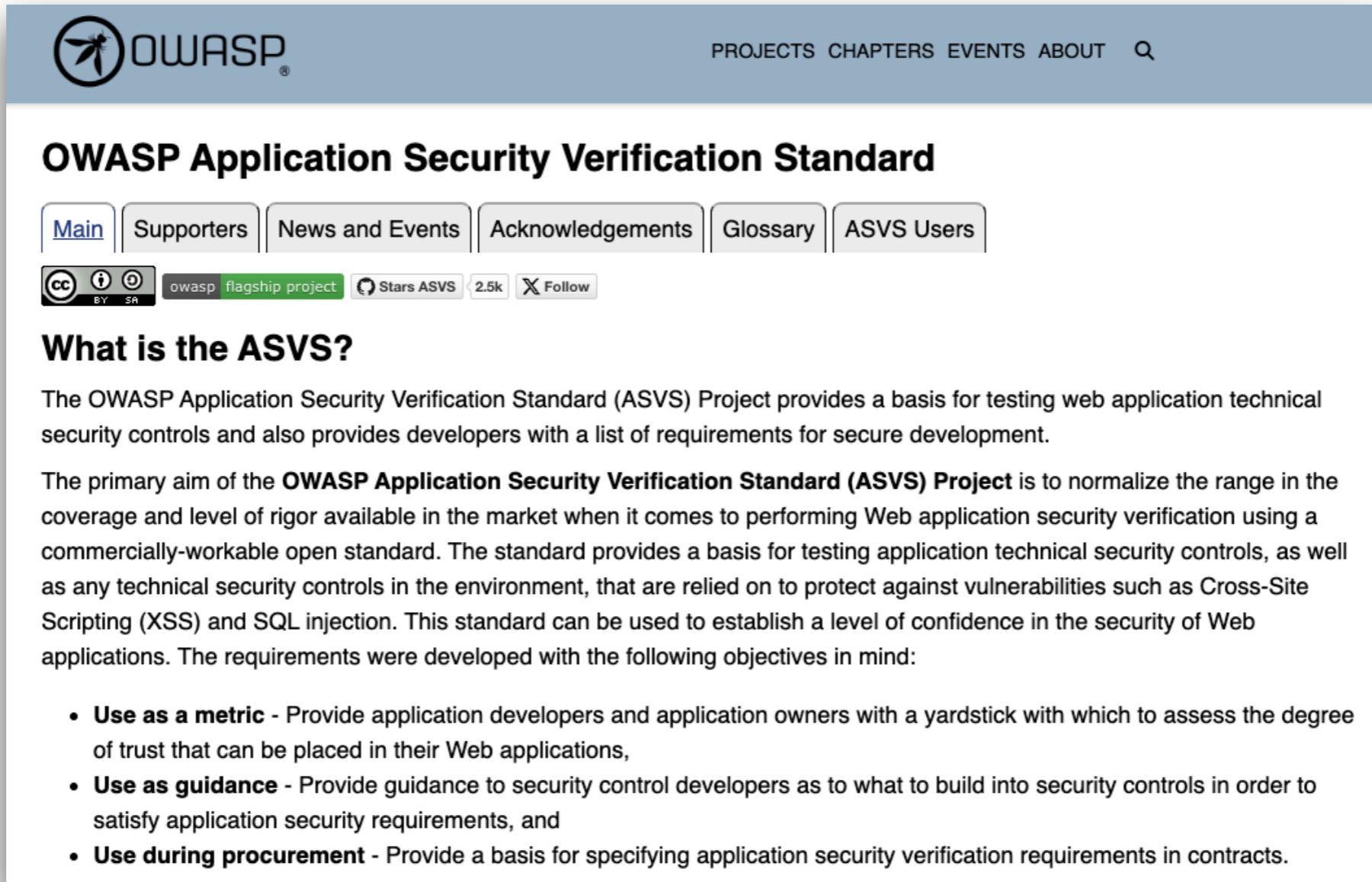
[https://owasp.org/Top10/A05\\_2021-Security\\_Misconfiguration/](https://owasp.org/Top10/A05_2021-Security_Misconfiguration/)



# More ...



# OWASP Application Security Verification Standard



The screenshot shows the homepage of the OWASP ASVS project. At the top, there's a navigation bar with the OWASP logo, a search icon, and links for PROJECTS, CHAPTERS, EVENTS, ABOUT, and a magnifying glass icon. Below the header, the title "OWASP Application Security Verification Standard" is displayed in bold. A horizontal menu bar follows, with "Main" highlighted in blue, and other options like "Supporters", "News and Events", "Acknowledgements", "Glossary", and "ASVS Users". Underneath the menu, there are social sharing icons for Creative Commons (CC), a green "owasp flagship project" badge, a GitHub star count of "2.5k", and a "Follow" button. The main content area starts with a section titled "What is the ASVS?". It contains two paragraphs explaining the purpose and objectives of the standard. The first paragraph states that the ASVS Project provides a basis for testing web application technical security controls and requirements for secure development. The second paragraph details the primary aim of normalizing coverage and rigor in Web application security verification, mentioning its use for testing technical security controls, protection against vulnerabilities like XSS and SQL injection, and establishing confidence in Web applications. It also lists three objectives: using it as a metric, guidance, and during procurement.

**What is the ASVS?**

The OWASP Application Security Verification Standard (ASVS) Project provides a basis for testing web application technical security controls and also provides developers with a list of requirements for secure development.

The primary aim of the **OWASP Application Security Verification Standard (ASVS) Project** is to normalize the range in the coverage and level of rigor available in the market when it comes to performing Web application security verification using a commercially-workable open standard. The standard provides a basis for testing application technical security controls, as well as any technical security controls in the environment, that are relied on to protect against vulnerabilities such as Cross-Site Scripting (XSS) and SQL injection. This standard can be used to establish a level of confidence in the security of Web applications. The requirements were developed with the following objectives in mind:

- **Use as a metric** - Provide application developers and application owners with a yardstick with which to assess the degree of trust that can be placed in their Web applications,
- **Use as guidance** - Provide guidance to security control developers as to what to build into security controls in order to satisfy application security requirements, and
- **Use during procurement** - Provide a basis for specifying application security verification requirements in contracts.

<https://owasp.org/www-project-application-security-verification-standard/>



# **2. Planing and Design**



# 2. Planning and Design

Understand requirements, project timeline  
Technology selection  
Human resources  
Required software and hardware

Security  
Architect

Security  
Officer

Security  
Tester



# **Security testing**

**What to test ?**

**When to test ?**

**What tools are required ?**



# **Secure Design ?**

Check all possible security design implementation  
Security risk assessment  
Review all design document  
**Threat modeling**



# Threat Modeling

Process to analyze and modeling threat  
Find solution and tools to protect/prevent

How  
attackers can  
abuse app ?

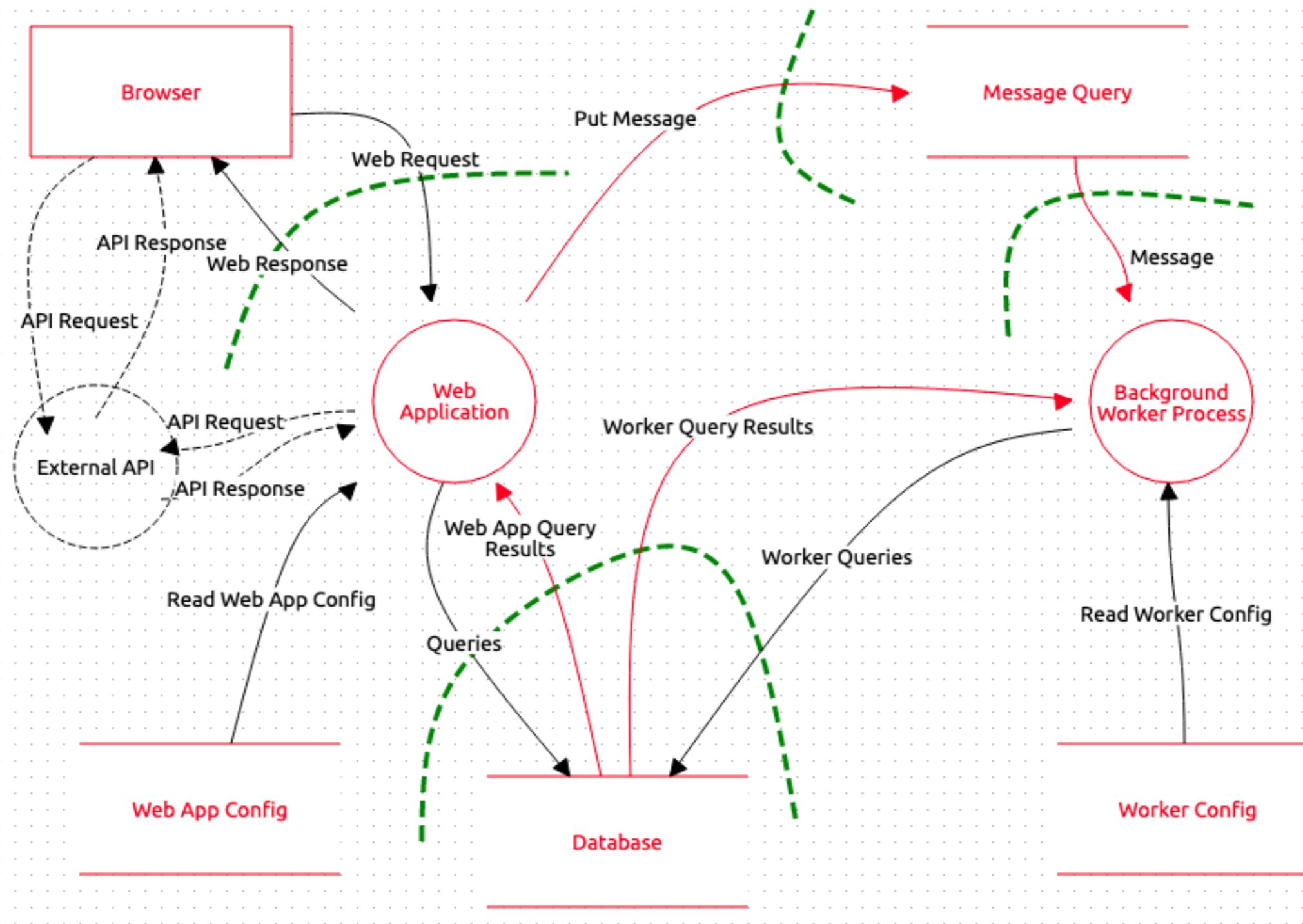
How to fix  
vulnerabilities  
?

How  
important to  
fix issues ?

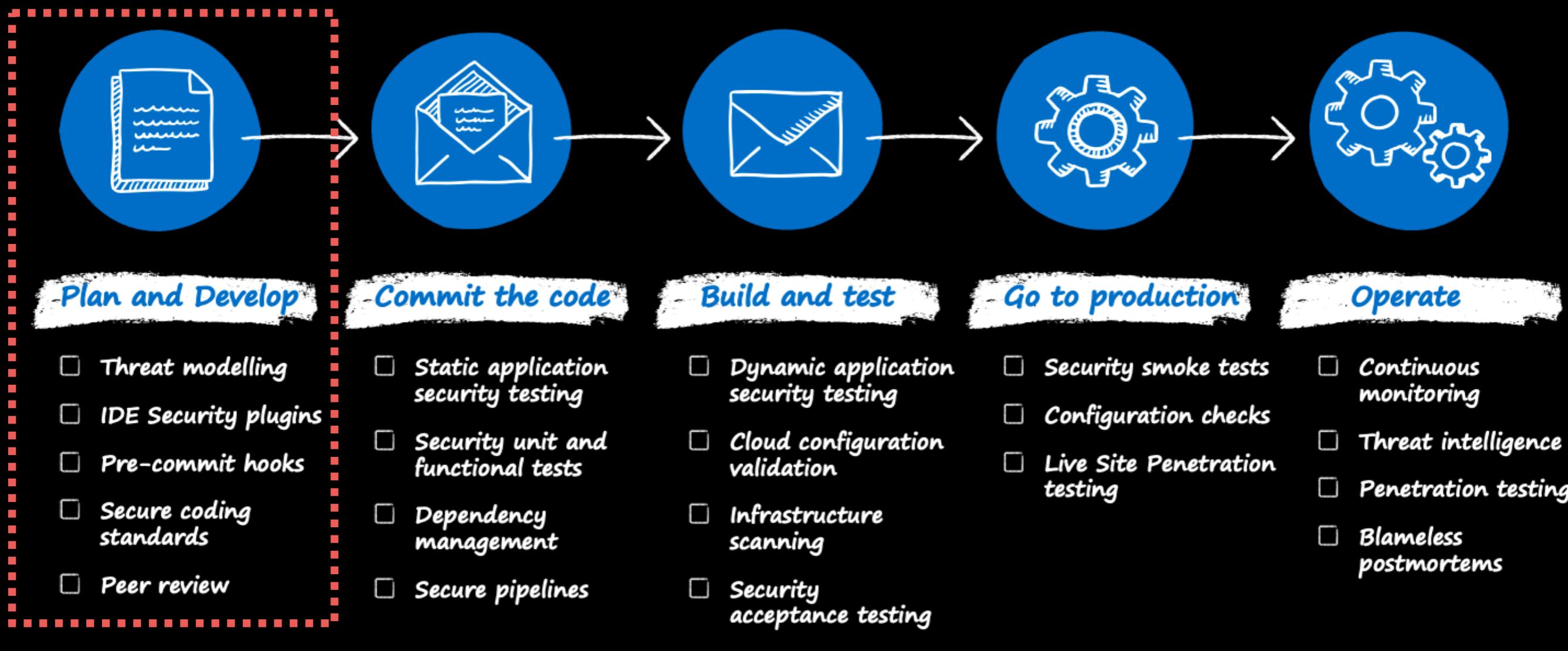
[https://owasp.org/www-community/Threat\\_Modeling](https://owasp.org/www-community/Threat_Modeling)



# Start with current Architecture



# Plan and Develop



<https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/secure/devsecops-controls>



# Plan and Develop

Threat modeling  
IDE security plugins  
Pre-commit hooks

Peer reviews and secure coding standard



# IDE Security Plugins

## Lint and Static code analysis

The screenshot shows the Visual Studio Code Marketplace search results for 'security nodejs'. The search bar at the top contains 'security nodejs'. Below it, there are two rows of six items each, all labeled as 'FREE'. The items are:

- CVE for NodeJS** by Sneezy: Shows security alert for vulnerable dependencies of Node projects. Rating: ★★★★★.
- JFrog** by jfrog.com: Security scanning for your Go, npm, Pypi, Maven and NuGet projects. Rating: ★★★★★.
- PT Application Inspector** by POSIdev-community: Security Analysis. Rating: ★★★★★.
- HCL AppScan CodeSweep** by HCL Software: A Code Editor extension that detects security vulnerabilities. Rating: ★★★★★.
- Refact** by smallcloud: Refact AI Assistant for Code Writing and Refactoring. Rating: ★★★★★.
- GPT4, AI Realtime code** by Sixth: GPT4 AI Realtime code scanner vscode extension for helping developers with cod... Rating: ★★★★★.
- Snyk Security** by Snyk: Easily find and fix vulnerabilities in your code, open source dependencies, ... Rating: ★★★★★.
- Security IntelliSense** by Microsoft: Provides quick and inline security suggestion and fixes for C# and XML source code. Rating: ★★★★★.
- Ethereum Security Bur** by tintinweb: A meta-extension bundling marketplace plugins for secure Ethereum smart... Rating: ★★★★★.
- Node Security Project** by Adam Baldwin: Checks for known vulnerabilities against the Node Security Project. Rating: ★★★★★.
- Socket Security** by Socket Security: Editor integration with Socket Security. Rating: ★★★★★.
- (Preview) Snyk Securit** by Snyk: This is a preview release for functionality that is not yet officially released. Rating: ★★★★★.

<https://marketplace.visualstudio.com/vscode>



Workshop

© 2017 - 2024 Siam Chamnankit Company Limited. All rights reserved.

# IDE Security Plugins

## Lint and Static code analysis

Visual Studio | Marketplace Sign in 

Visual Studio Visual Studio Code Azure DevOps Subscriptions Build your own Publish extensions

lint 

185 Results Showing: Programming Languages Sort By: Relevance

 <b>C/C++ Advanced Lint</b> Joseph Benden  An advanced, modern, static analysis extension for C/C++ that supports a number of...  FREE	 <b>GLSL Lint</b> DanielToplak  Linting of GLSL shader code  FREE	 <b>GLSL Lint (deprecated)</b> CADENAS GmbH  Linting of GLSL shader code  FREE	 <b>salt-lint</b> warpnet  salt-lint checks Salt State files (SLS) for practices and behavior that could...  FREE	 <b>axe Accessibility Linter</b> Deque Systems  Accessibility linting for HTML, Angular, React, Markdown, Vue, and React Native  FREE	 <b>Thanos Lint</b> zhangzongzheng  A VSCode plugin for linting Thanos code  FREE
 <b>Clojure-lint</b> William Lindsay  Extension to Andrey Lisin's Clojure  FREE	 <b>GLSL Lint Hezuikn For</b> hezuikn  Linting of GLSL shader code  FREE	 <b>ADINA-Lint</b> Daniel Steinegger  Set's your coding experience with ADINA into the another level!  FREE	 <b>CloudFormation Linter</b> kddejong  AWS CloudFormation template Linter  FREE	 <b>TSLint</b> Microsoft  4.1M TSLint support for Visual Studio Code  FREE	 <b>Groovy Lint, Format a</b> Nicolas Vuillamy  Lint, format and auto-fix groovy and Jenkinsfile  FREE

<https://marketplace.visualstudio.com/vscode>

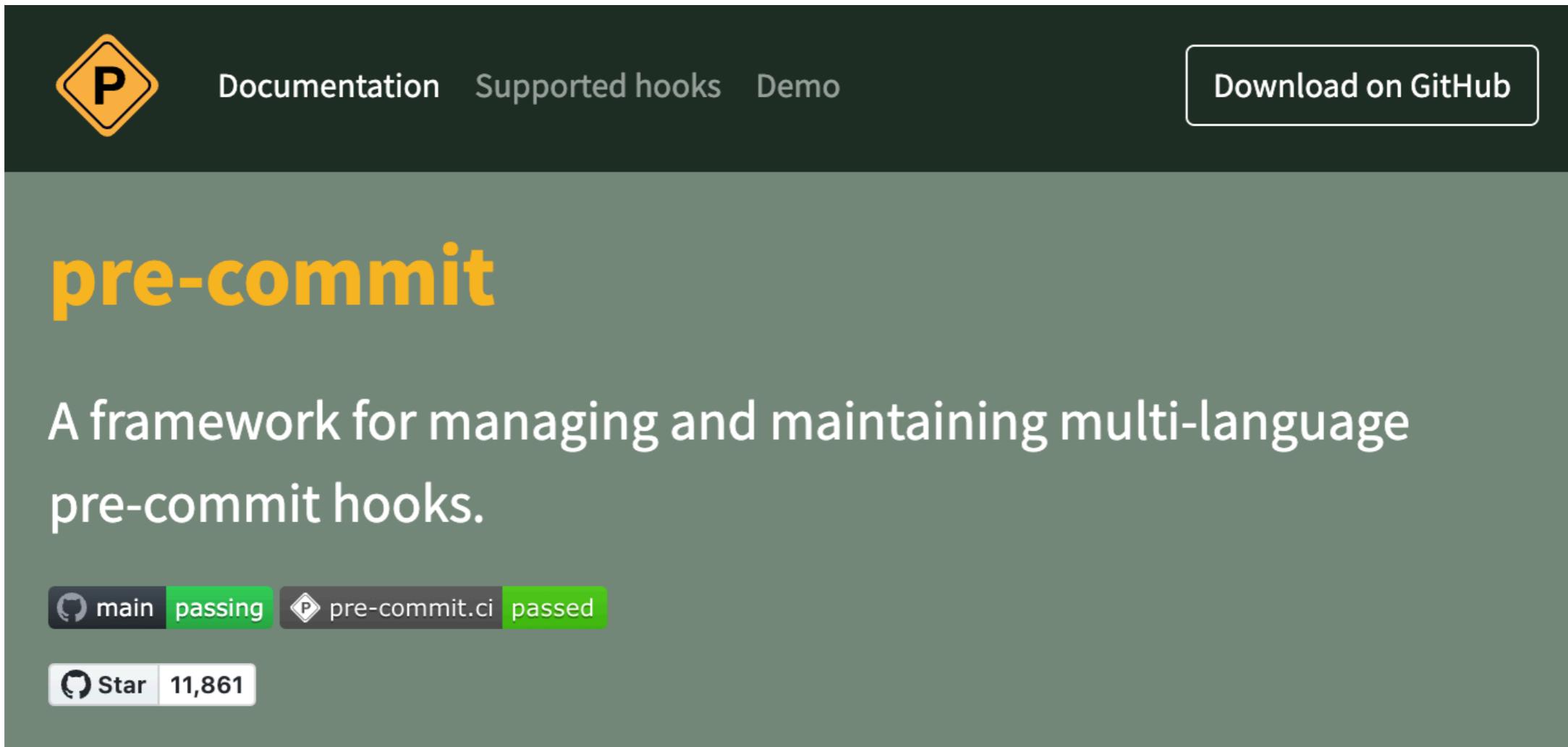


Workshop

© 2017 - 2024 Siam Chamnankit Company Limited. All rights reserved.

# Git pre-commit hooks

Self-test in developer machine (fast feedback)



The screenshot shows the GitHub repository page for "pre-commit". At the top, there's a yellow diamond icon with a black "P" inside, followed by navigation links: Documentation, Supported hooks, Demo, and a "Download on GitHub" button. The main title "pre-commit" is displayed in large yellow letters. Below it, a description reads: "A framework for managing and maintaining multi-language pre-commit hooks." There are two status indicators: "main" (green) and "pre-commit.ci" (green), both labeled "passed". A "Star" button shows 11,861 stars. The background of the page is dark green.

<https://pre-commit.com/>



Workshop

© 2017 - 2024 Siam Chamnankit Company Limited. All rights reserved.

# Git pre-commit hooks

\$git commit -m "fix(faker): credit card from faker error"

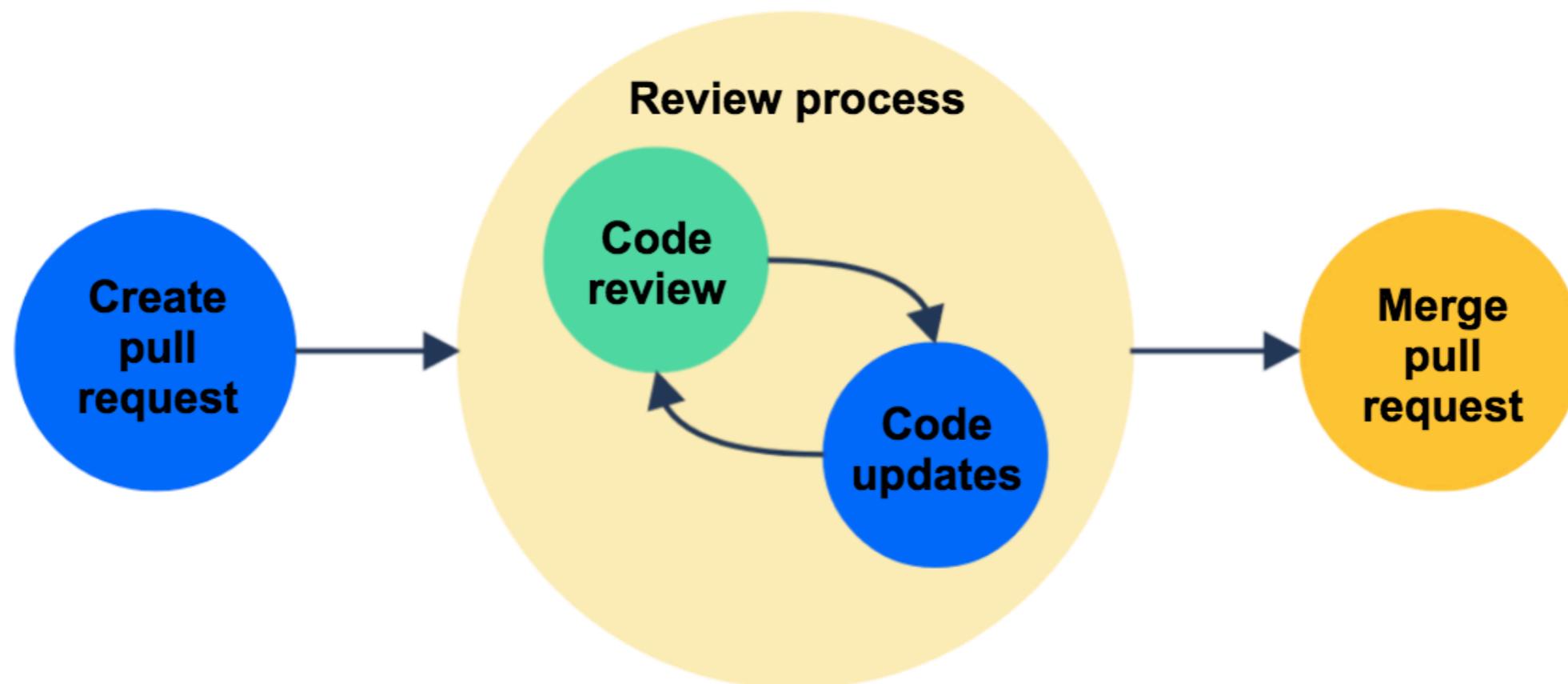
- ✓ Preparing lint-staged...
- ✓ Running tasks for staged files...
- ✓ Applying modifications from tasks...
- ✓ Cleaning up temporary files...

<https://typicode.github.io/husky/>



# Peer Review and Secure coding standard

Use pull request (PR) and review



<https://support.atlassian.com/bitbucket-cloud/docs/use-pull-requests-for-code-review/>



# OWASP Secure Coding Practices

The screenshot shows the OWASP website with the following details:

- Header:** OWASP logo, navigation menu (PROJECTS, CHAPTERS, EVENTS, ABOUT, Q), Member Login.
- Title:** OWASP Secure Coding Practices-Quick Reference Guide
- Buttons:** Main (selected), Download, Contributors, Archive.
- Section:** Cornucopia
- Description:** Version 2.1 of the Secure Coding Practices quick reference guide provides the numbering system used in the Cornucopia project playing cards.
- Section:** Archived project
- Description:** The OWASP Secure Coding Practices Quick-reference Guide project has now been archived. The content of the Secure Coding Practices Quick-reference Guide overview and glossary has been migrated to various sections within the [OWASP Developer Guide](#). The Secure Coding Practices Quick-reference Guide checklists have also been migrated to the Developer Guide; this provides a wider audience for the original checklist. Contact [Jon Gadsden](#) for any questions about this move.

<https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/>



Workshop

© 2017 - 2024 Siam Chamnankit Company Limited. All rights reserved.

# When develop committed code ...

Dependency checking

Static Application Security Testing (SAST)

Security pipeline

Dynamic Application Security Testing (DAST)



# Dependency Checking

OWASP  
Dependency  
-Check

GitHub  
Dependabot

NPM audit

<https://owasp.org/www-project-dependency-check/>



# Static Application Security Testing (SAST)

Static Code Analysis (detect 50%)

Scan and review code

Identify source of vulnerabilities

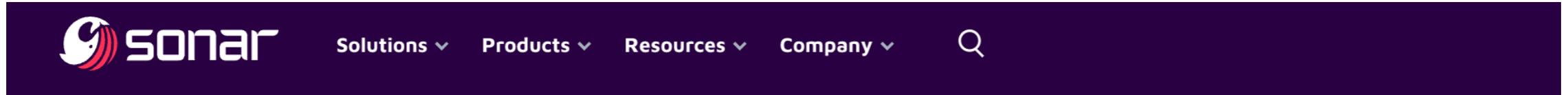
White-box testing

*Must be integrate into development process to help development team*

[https://en.wikipedia.org/wiki/Static\\_application\\_security\\_testing](https://en.wikipedia.org/wiki/Static_application_security_testing)



# SonarQube



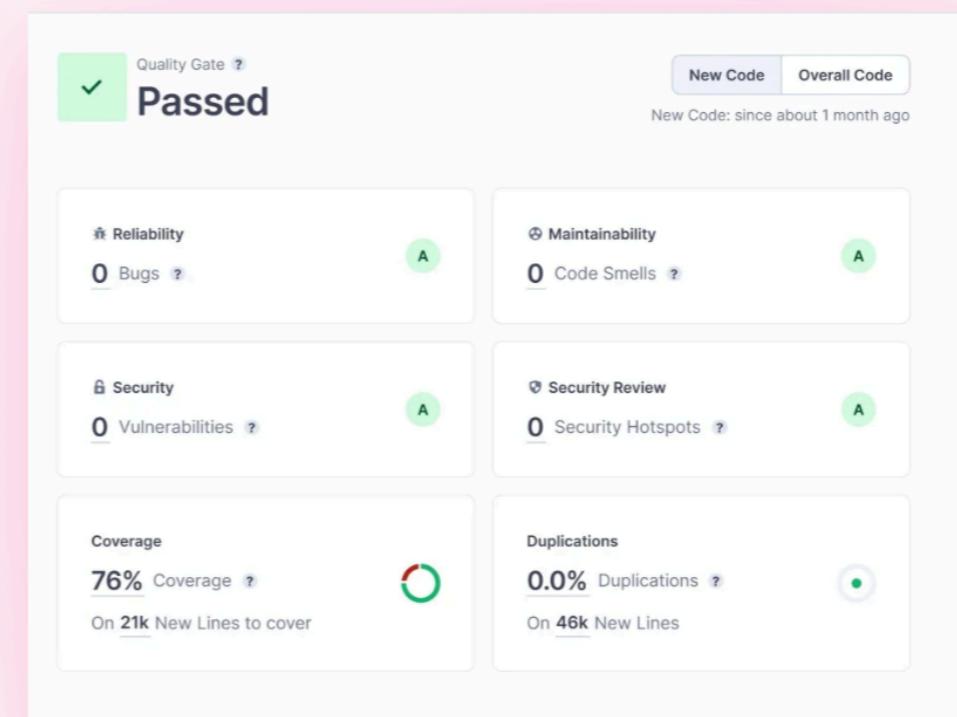
sonarqube | Deployment   What's New   Roadmap   Documentation   Download   Pricing   TRY FOR FREE

**SELF-MANAGED. SONARQUBE.**

## clean code for teams and enterprises with {SonarQube}

Empower development teams with a code quality and security solution that deeply integrates into your enterprise environment; enabling you to deploy clean code consistently and reliably.

[START FREE TRIAL -->](#)   [WHAT IS SONARQUBE ▶ -->](#)



Quality Gate ? Passed

New Code   Overall Code  
New Code: since about 1 month ago

Reliability	Maintainability
0 Bugs <span>?</span>	0 Code Smells <span>?</span>

Security	Security Review
0 Vulnerabilities <span>?</span>	0 Security Hotspots <span>?</span>

Coverage	Duplications
76% Coverage <span>?</span> On 21k New Lines to cover	0.0% Duplications <span>?</span> On 46k New Lines

<https://www.sonarsource.com/products/sonarqube/>



Workshop

© 2017 - 2024 Siam Chamnankit Company Limited. All rights reserved.

# DevSkim

The screenshot shows the DevSkim extension page on the Visual Studio Marketplace. At the top, there's a large DevSkim logo with a bug icon. Below it, the title "DevSkim" is displayed, along with the Microsoft DevLabs badge, a link to [microsoft.com](https://microsoft.com), 39,316 installs, and a 5-star rating from 3 reviews. A brief description follows: "DevSkim Security Analyzer Plugin for IDEs. Find security mistakes as code is authored, and fix them with a mouse click." Below this are two buttons: "Install" (in green) and "Trouble Installing?". At the bottom of the main section, there are tabs for "Overview" (which is underlined), "Version History", "Q & A", and "Rating & Review". The "Overview" tab contains a detailed description of the DevSkim framework.

DevSkim

Microsoft DevLabs [microsoft.com](https://microsoft.com) | 39,316 installs | (3)

DevSkim Security Analyzer Plugin for IDEs. Find security mistakes as code is authored, and fix them with a mouse click.

[Install](#) [Trouble Installing?](#)

Overview Version History Q & A Rating & Review

**DevSkim**

DevSkim is a framework of IDE extensions and language analyzers that provide inline security analysis in the dev environment as the developer writes code. It has a flexible rule model that supports multiple programming languages. The goal is to notify the developer as they are introducing a security vulnerability in order to fix the issue at the point of introduction, and to help build awareness for the developer.

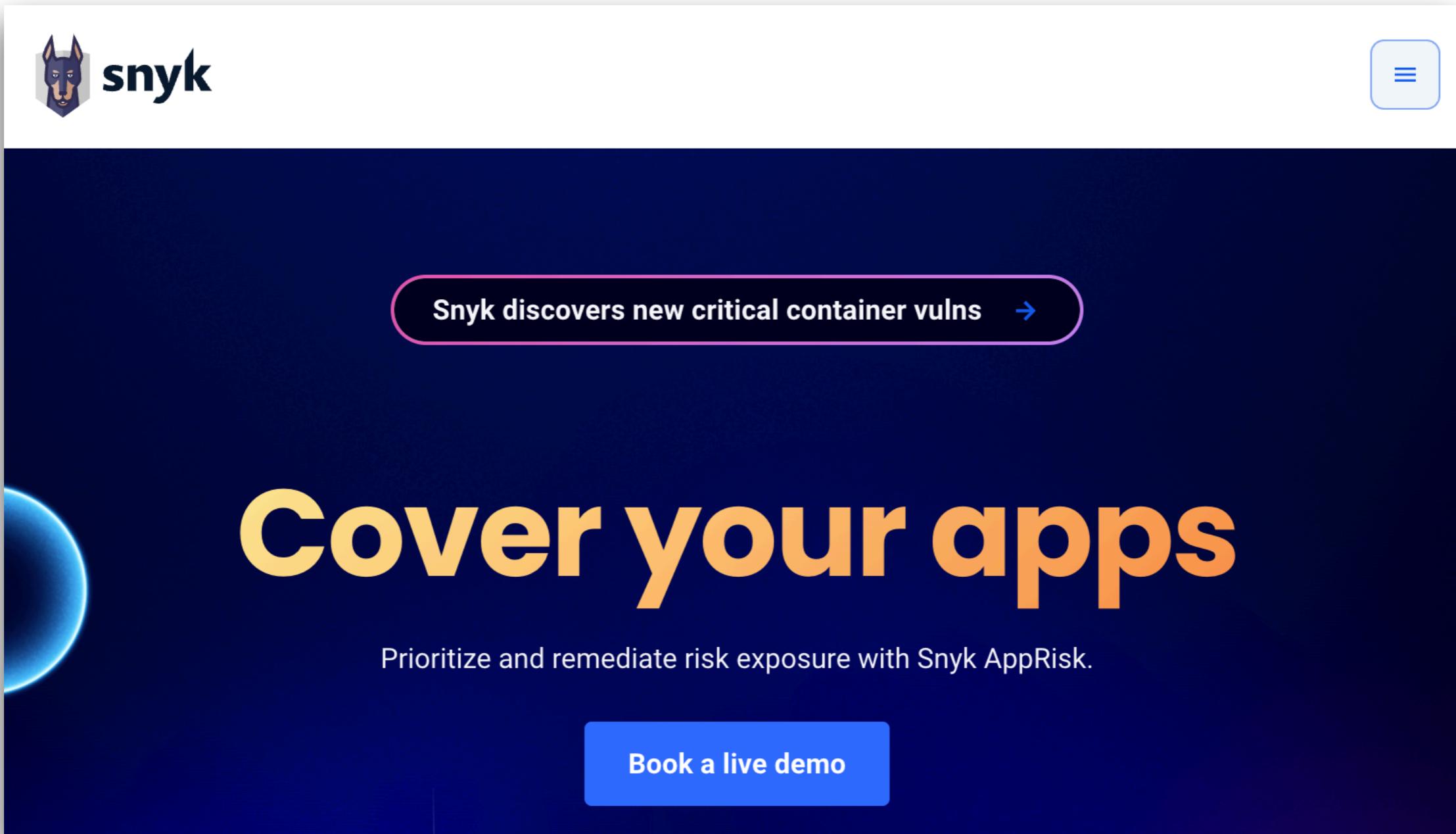
<https://marketplace.visualstudio.com/items?itemName=MS-CST-E.vscode-devskim>



Workshop

© 2017 - 2024 Siam Chamnankit Company Limited. All rights reserved.

# Snyk



The image shows the Snyk homepage. At the top left is the Snyk logo (a stylized dog head icon next to the word "snyk"). At the top right is a blue square menu icon with three horizontal lines. Below the header is a dark blue background section. In the upper left corner of this section is a partial view of a blue planet. In the center, there is a white button with rounded corners containing the text "Snyk discovers new critical container vulns" followed by a right-pointing arrow. Below this button, in large yellow text, is the slogan "Cover your apps". Underneath the slogan, in smaller white text, is the subtext "Prioritize and remediate risk exposure with Snyk AppRisk.". At the bottom of this section is a blue rectangular button with white text that reads "Book a live demo".

<https://snyk.io/>



Workshop

© 2017 - 2024 Siam Chamnankit Company Limited. All rights reserved.

# More tools

Lint

Scan secret/  
credential

**ES/TS Lint**

**Git-secrets**

[https://owasp.org/www-community/Source\\_Code\\_Analysis\\_Tools](https://owasp.org/www-community/Source_Code_Analysis_Tools)



# NodeJS

A curated list of awesome Node.js Security resources.

tools 30+

incidents 15+

educational 8+

X Follow @Liran Tal

*List inspired by the [awesome list thing](#).*

<https://github.com/lirantal/awesome-nodejs-security>



Workshop

© 2017 - 2024 Siam Chamnankit Company Limited. All rights reserved.

# Node Secure CLI



a Node.js CLI to deeply analyze the dependency tree of a given NPM package or Node.js local app

VERSION V2.4.0 LICENSE MIT OPENSSF SCORECARD 8.8 SLSA LEVEL 3 BUILD PASSING COVERAGE 100%

fastify v4.1.0  
Fast and low overhead web framework, for Node.js  
NPM GitHub Umpig MIT

overview

Homepage https://www.fastify.io/  
Author Matteo Collina  
Size on system 1.81 MB  
Number of dependencies 14  
Number of files 254  
README.md  
TS Typings

RELEASES

Last release version 4.1.0  
Last release date 21 Jun 2022, 11:27:51  
Number of published releases 224  
Number of publisher(s) 2

GITHUB

23958 STARS 60 ISSUES 1755 FORKS

MAINTAINERS

secure-json-parse@2.4.0 [33.2 kB]  
@fastify/error@3.0.0 [9.97 kB]  
process-warning@2.0.0 [11.2 kB]  
quick-format-unescaped@4.0.4 [12.8 kB]

abstract-logging@2.0.1 [2.3 kB]  
rfdo@1.3.0 [24 kB]  
tiny-tru@0.0.2 [0.1 kB]  
archy@1.0.0 [0.42 kB]  
fastro@1.13.0 [38.2 kB]  
debug@4.3.4 [42.4 kB]

Search

<https://github.com/NodeSecure/cli>

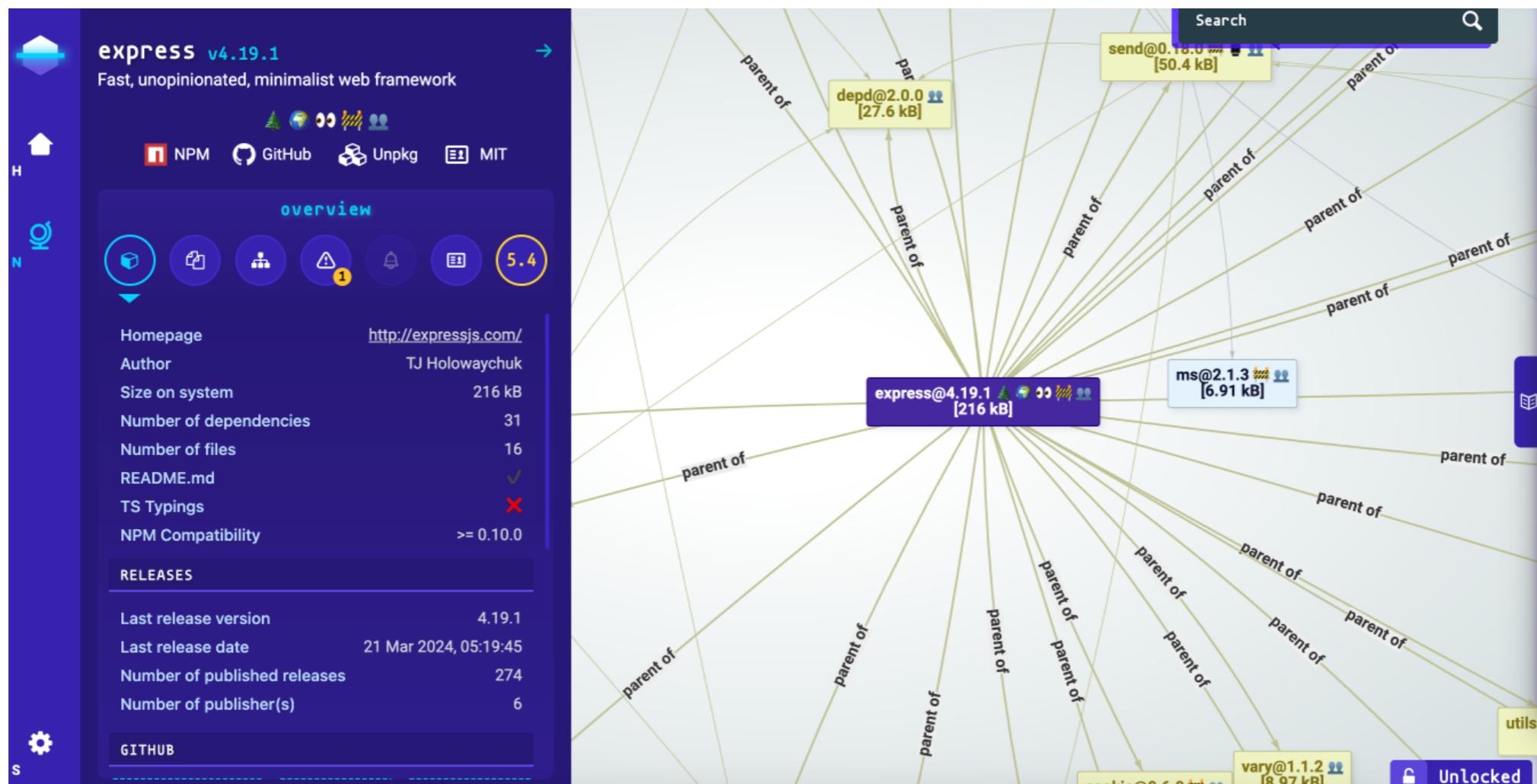


Workshop

© 2017 - 2024 Siam Chamnankit Company Limited. All rights reserved.

# Node Secure CLI

\$npm install -g @nodesecure/cli  
\$nsecure auto express



# Bearer CLI

Announcement

Bearer entered into an agreement to be acquired by Cycode, the complete ASPM.

[Learn more →](#)



Scan your source code against top **security** and **privacy** risks.

Bearer CLI is a static application security testing (SAST) tool that scans your source code and analyzes your data flows to discover, filter and prioritize security and privacy risks.

Currently supporting: **JavaScript/TypeScript (GA)**, **Ruby (GA)**, **PHP (GA)**, **Java (GA)**, **Go (Beta)**, **Python (Alpha)** -  
[Learn more](#)

<https://github.com/Bearer/bearer>



Workshop

© 2017 - 2024 Siam Chamnankit Company Limited. All rights reserved.

# **Bearer CLI**

Build-in rules in many languages

Support OWASP Top 10 and CWE Top 25

Privacy risks

Detect sensitive data

<https://docs.bearer.com/reference/rules/>



# Security Pipeline



# Security Pipeline

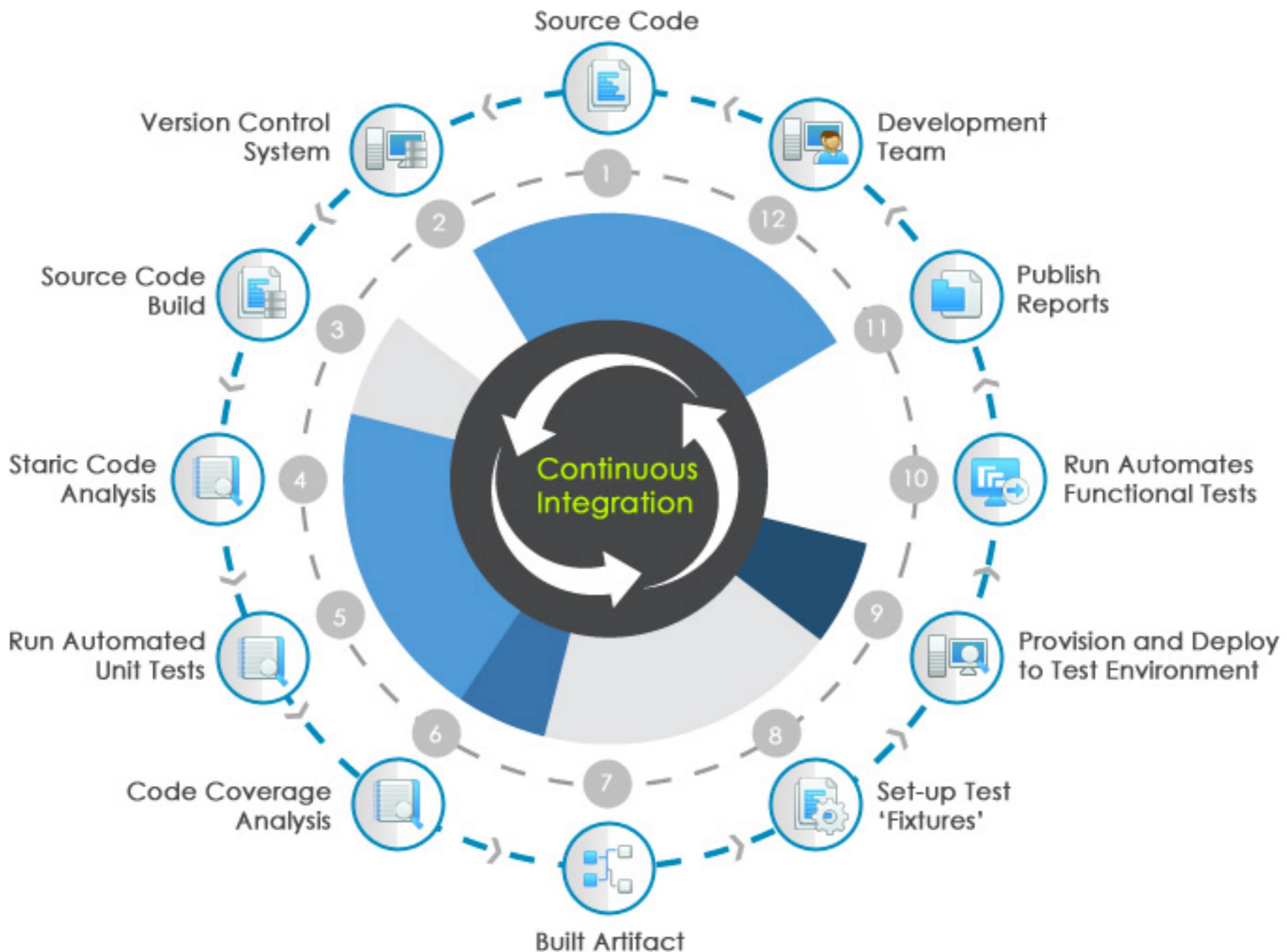
Implement security control in pipeline  
Continuous Integration and Delivery

*Build -> Test -> Deploy -> Monitoring*

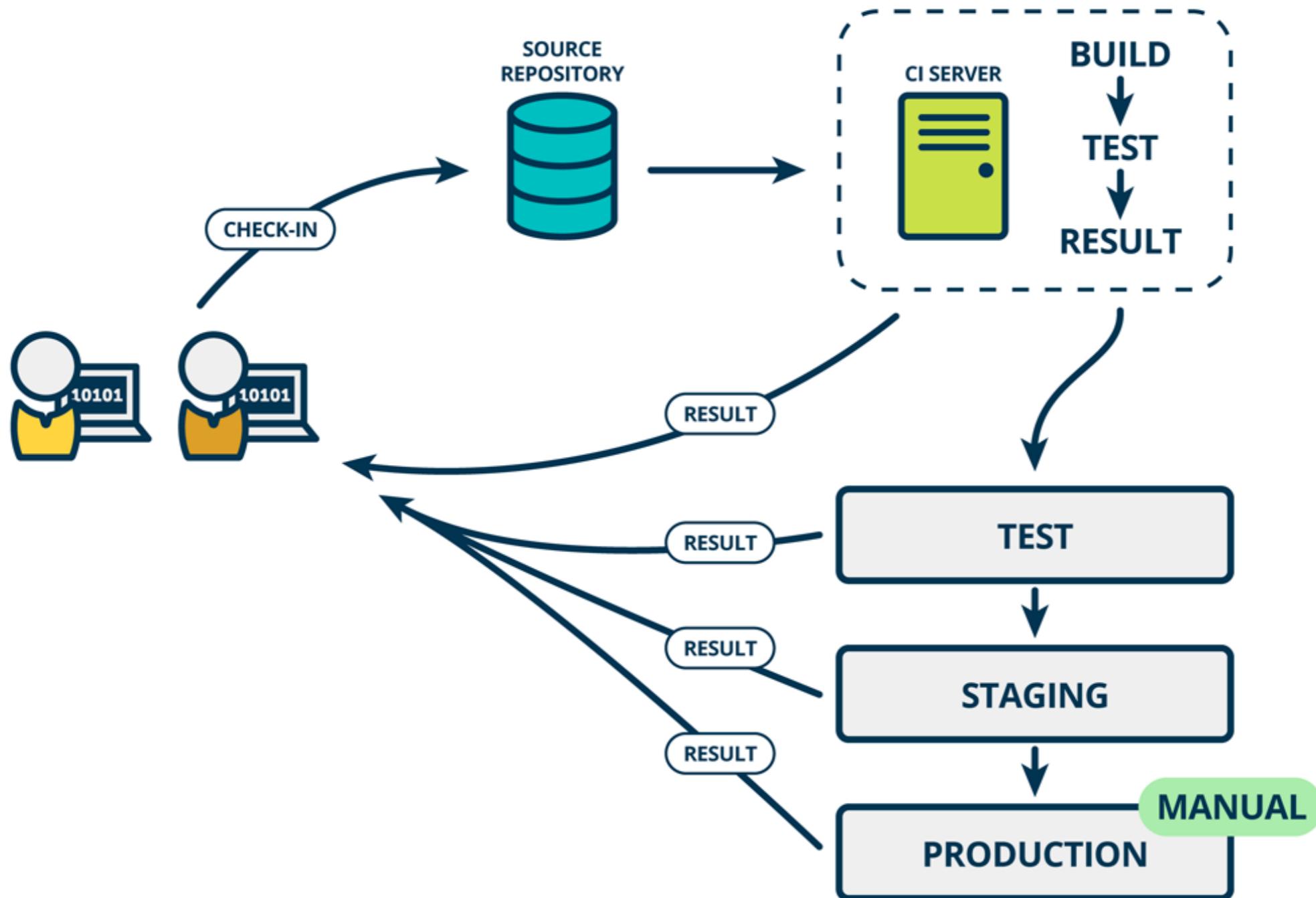


# Continuos Integration

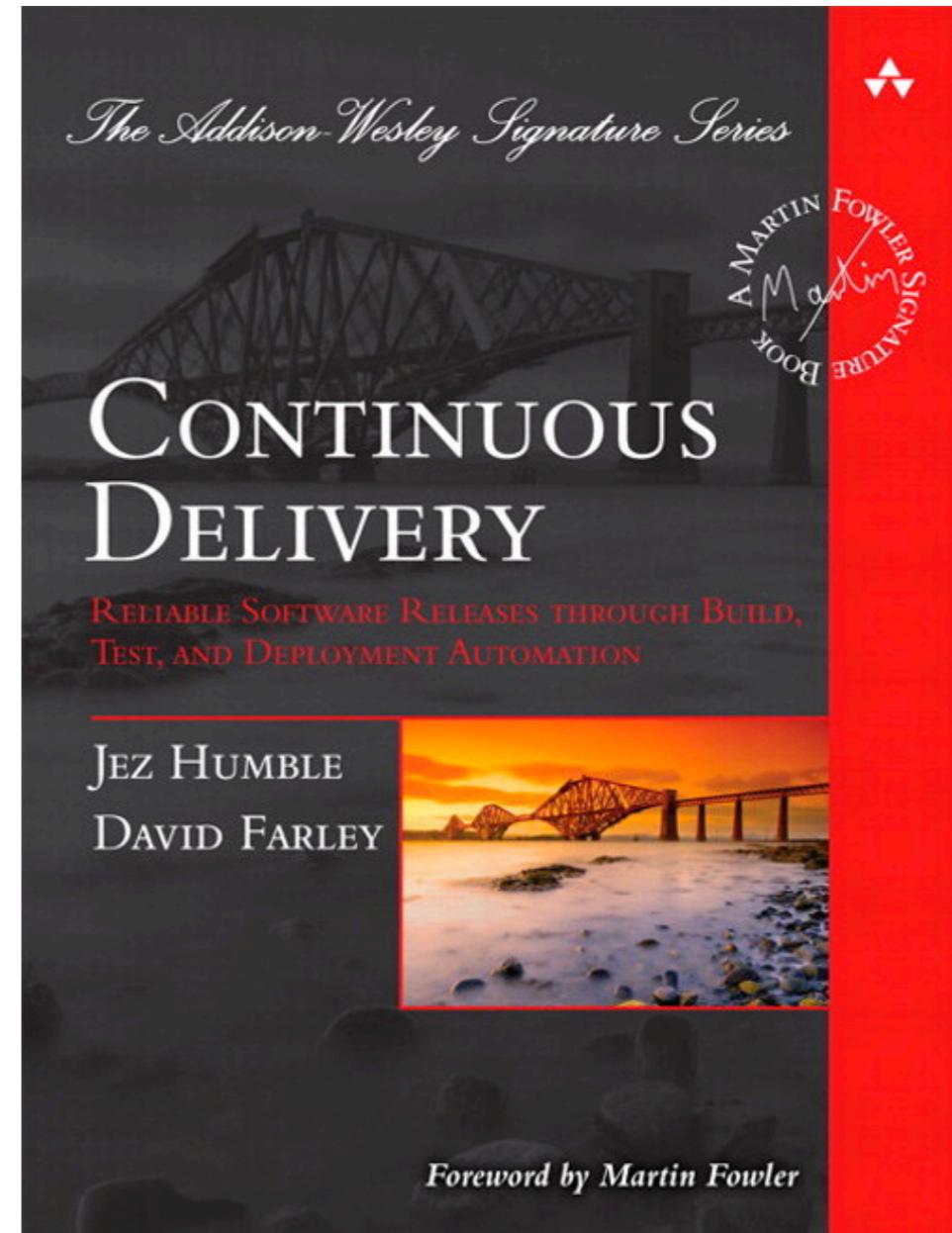
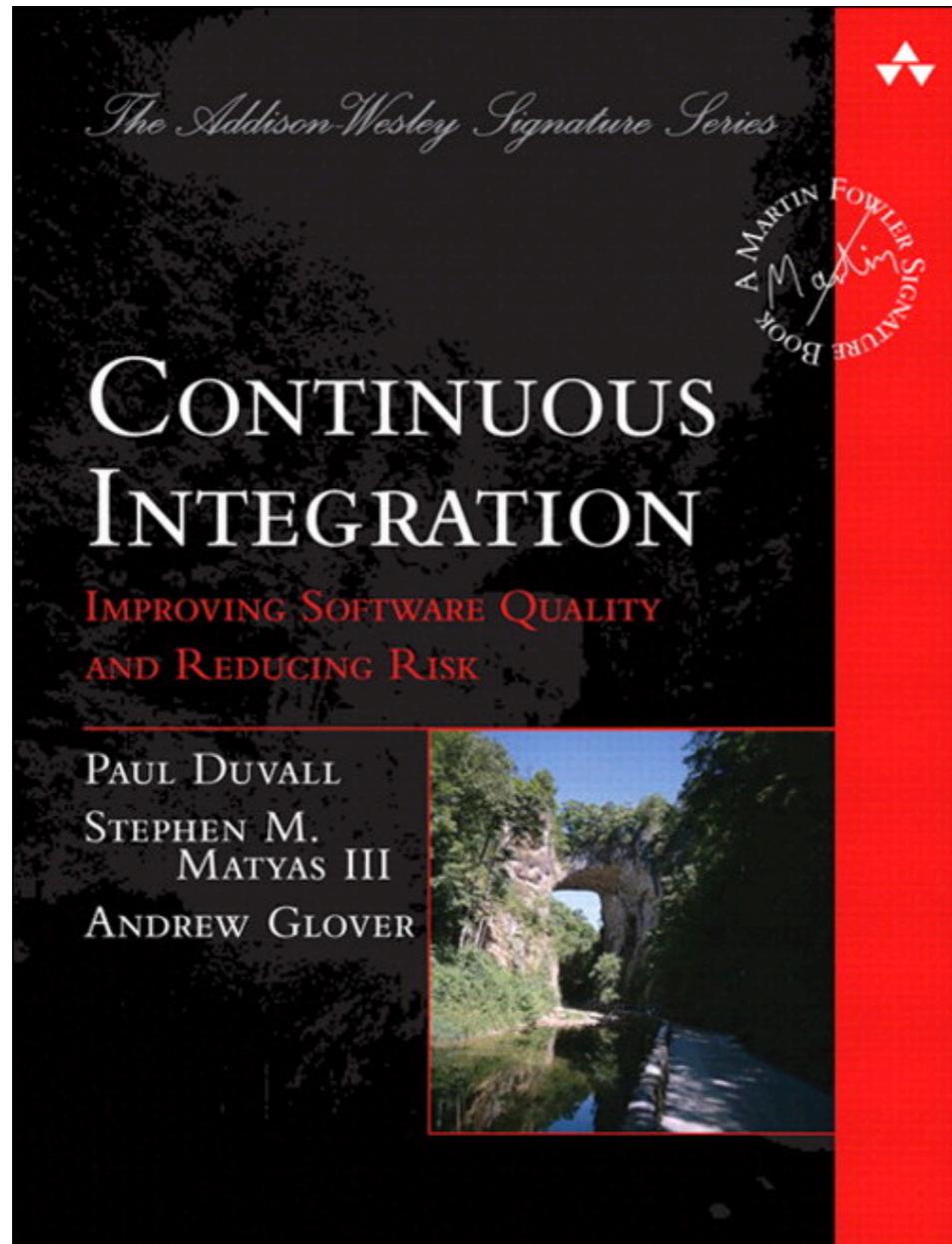




# Continuous Delivery



# Improve quality and reduce risk



# Continuous Integration

Discipline to integrate frequently



# Continuous Integration

Strive to make **small change**



# Continuous Integration

Strive for **fast feedback**



# **Continuous Integration**

**is a Software development practices**



Workshop

© 2017 - 2024 Siam Chamnankit Company Limited. All rights reserved.

# Practice 1

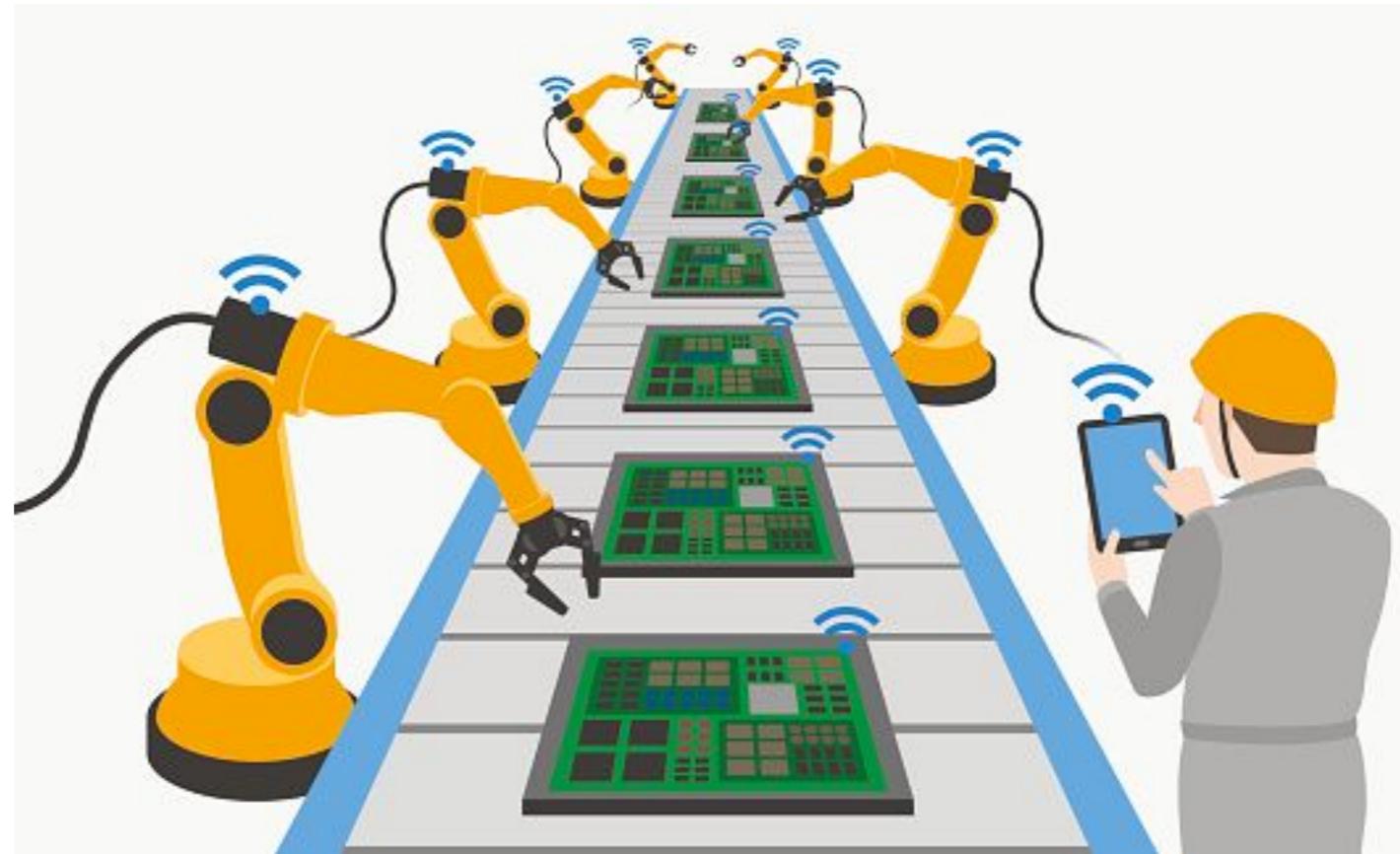
Maintain a single source repository

In general, you should store in source control  
everything you need to build anything



# Practice 2

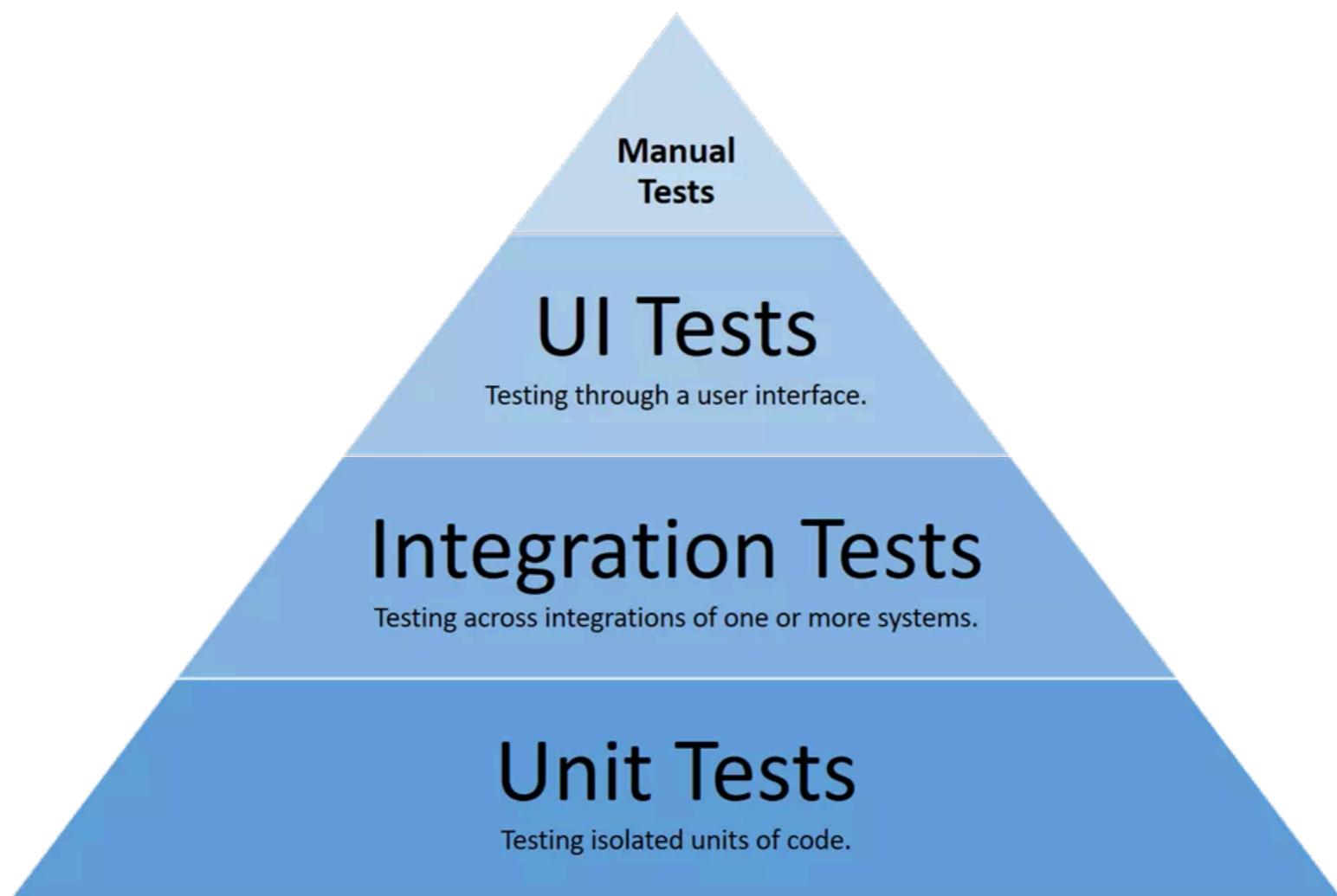
Automated the build  
Automated environment for builds



# Practice 3

Make your build **self-testing**

Build process => compile, linking and **testing**

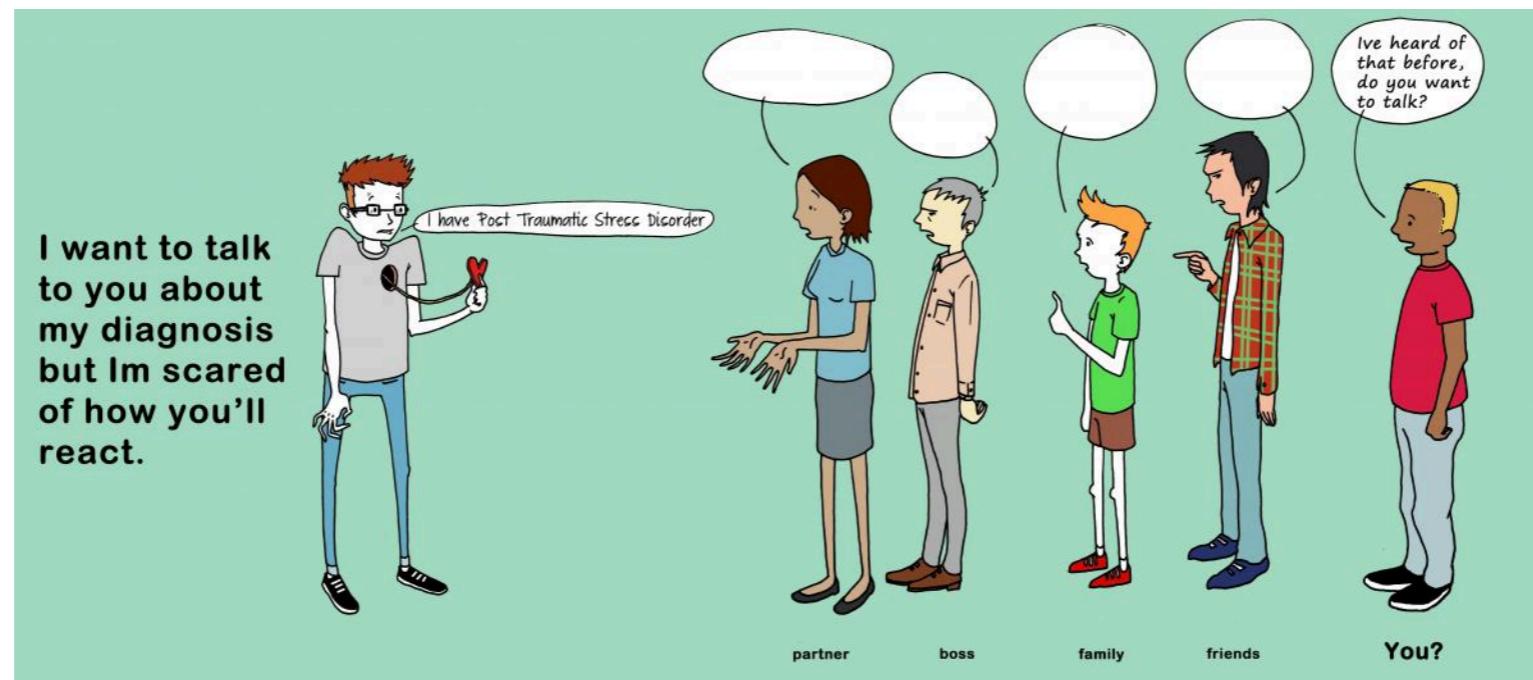


# Practice 4

**Everyone commits to the mainline everyday**

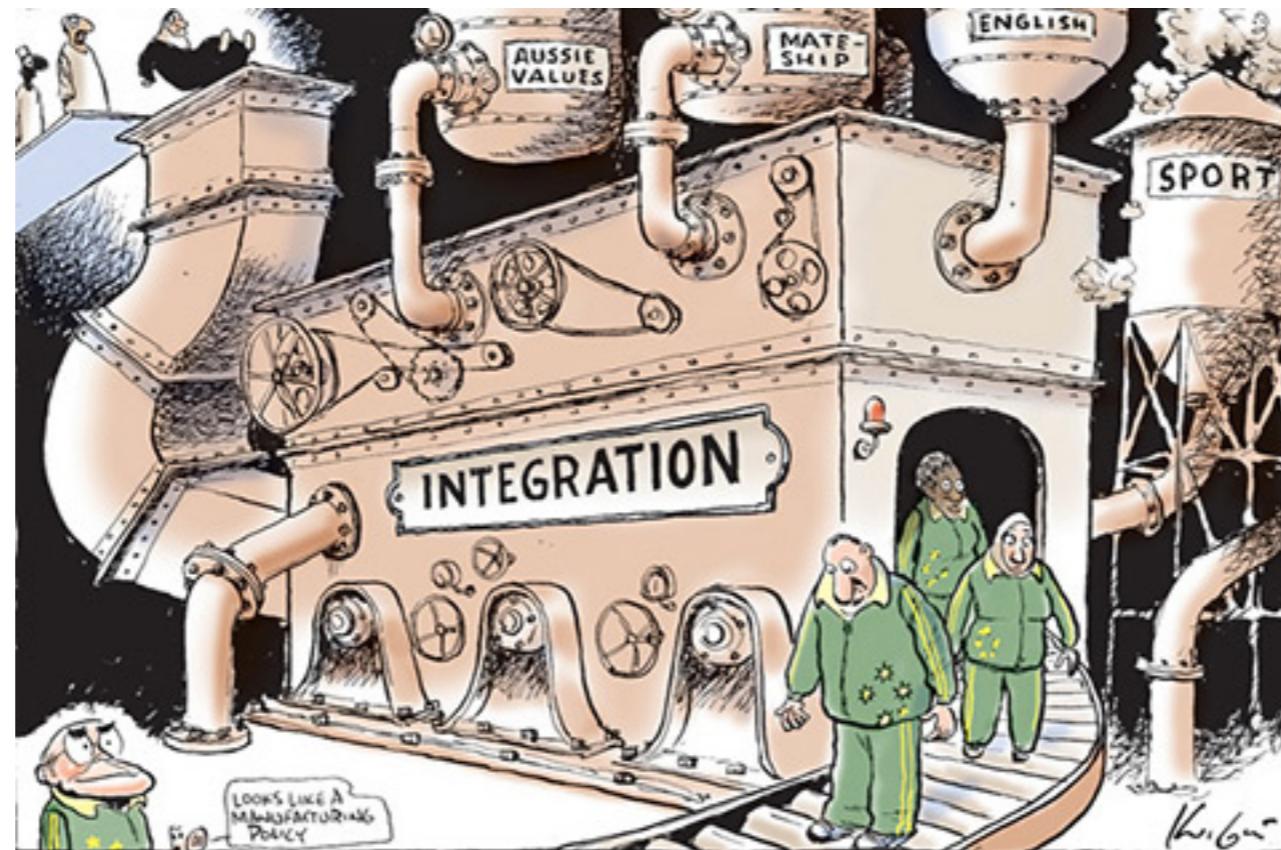
**Integration is about communication**

**Integration allows developers to tell other developers**



# Practice 5

Every commits should build the mainline on an  
**Integration machine**



# Practice 6

**Fix broken builds immediately**

**“Nobody has a higher priority task than  
fixing the build”**



# Practice 7

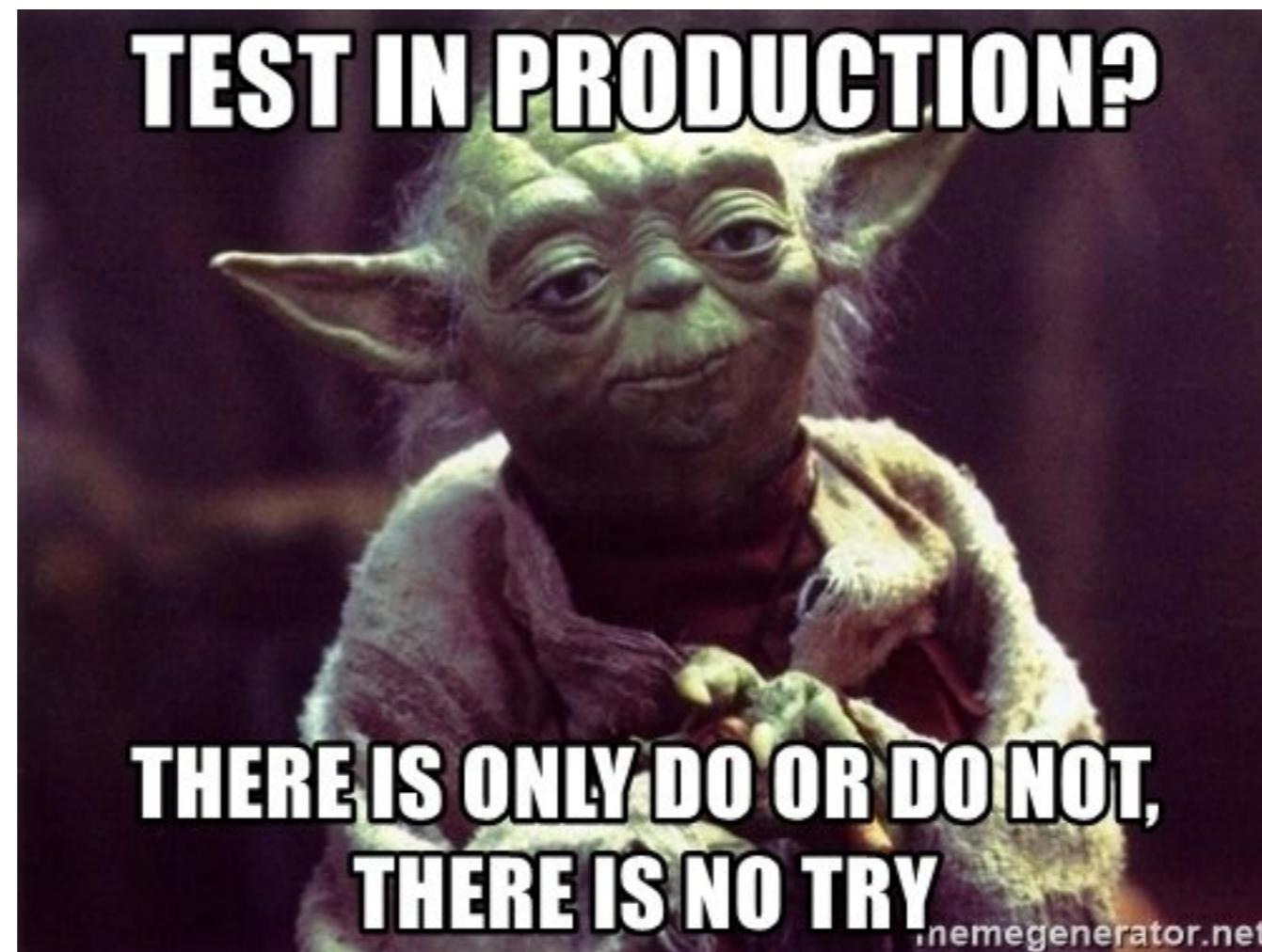
Keep the build **fast**

Continuous Integration is to provide rapid feedback



# Practice 8

Test in clone of the **Production** environment



# Practice 9

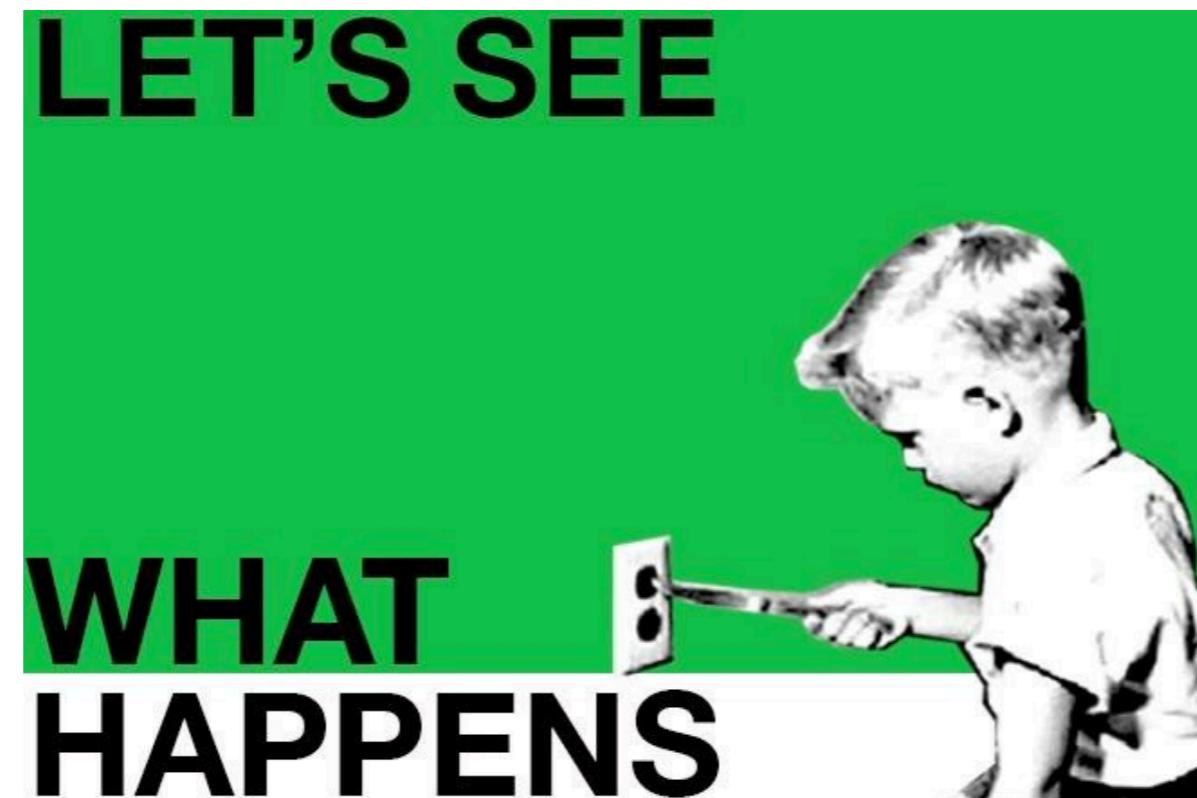
Make it easy for anyone to get  
the latest executable

Make sure well known place where people can find



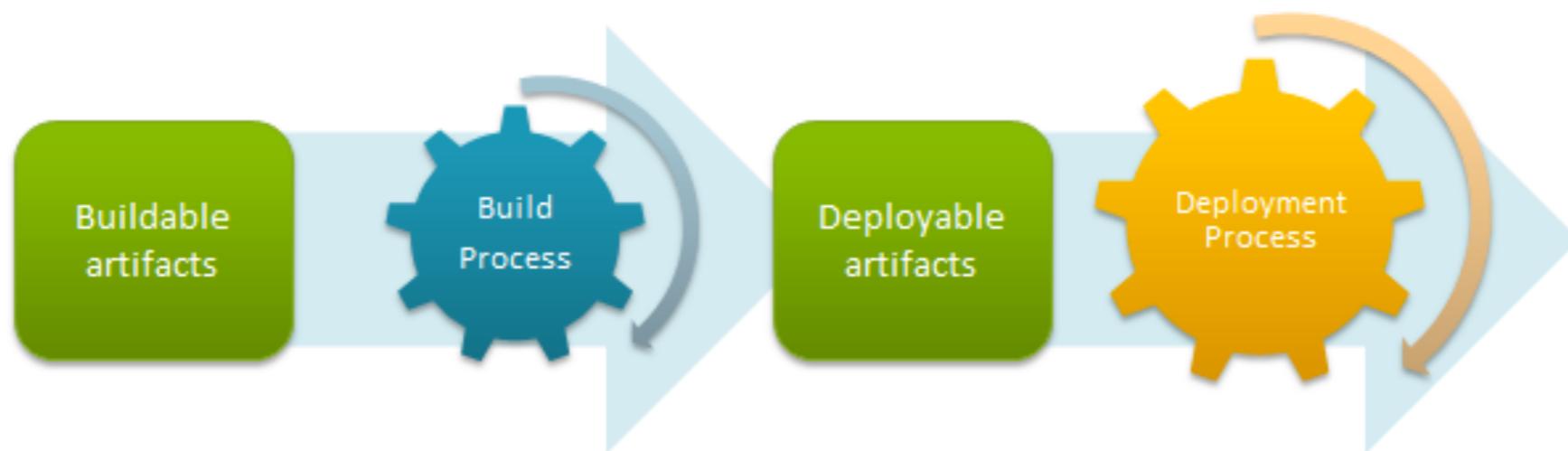
# Practice 10

**Everyone** can see what's happening  
**Easier** to see the state of the system and changes  
Show the good information

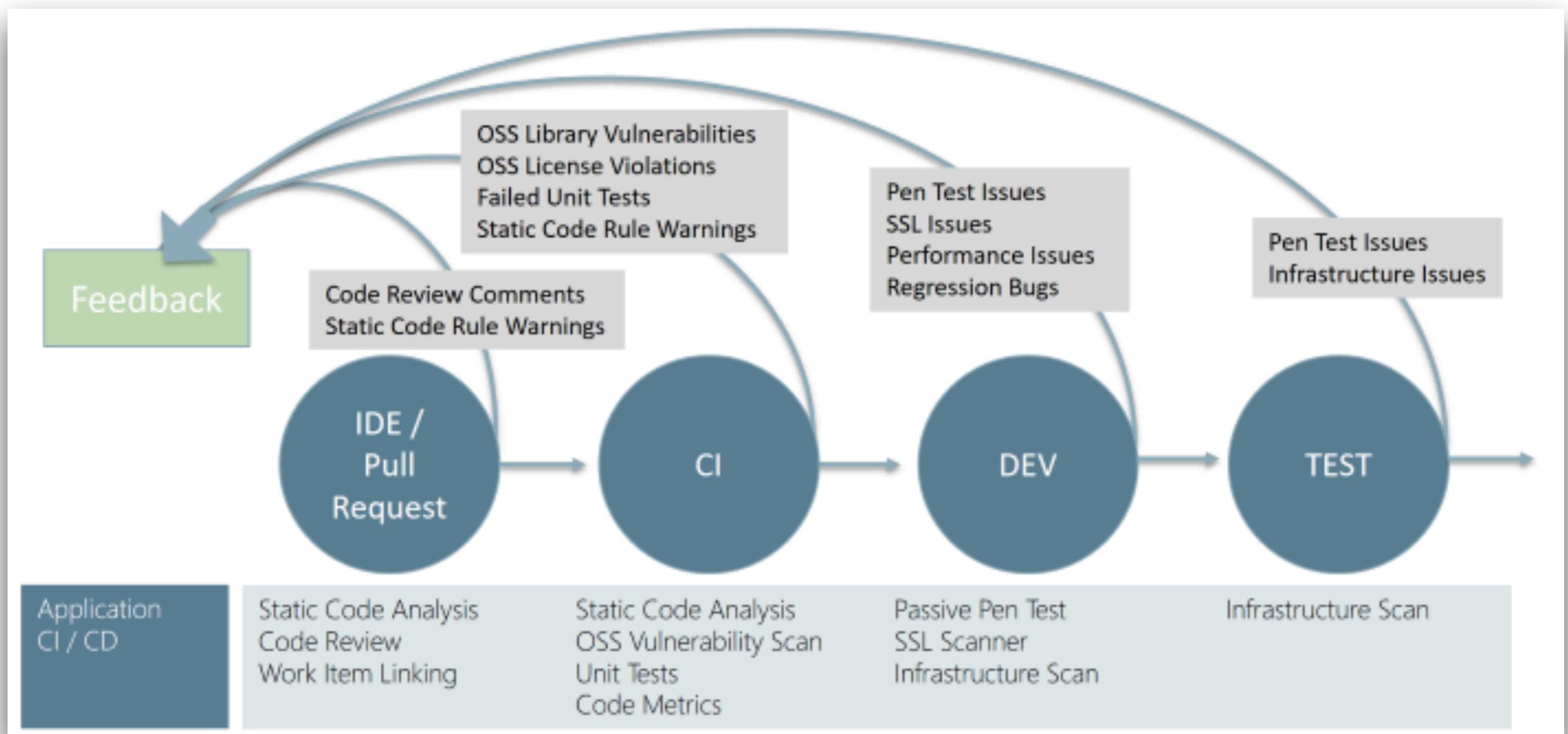


# Practice 11

## Automated deployment



# All about feedback loop

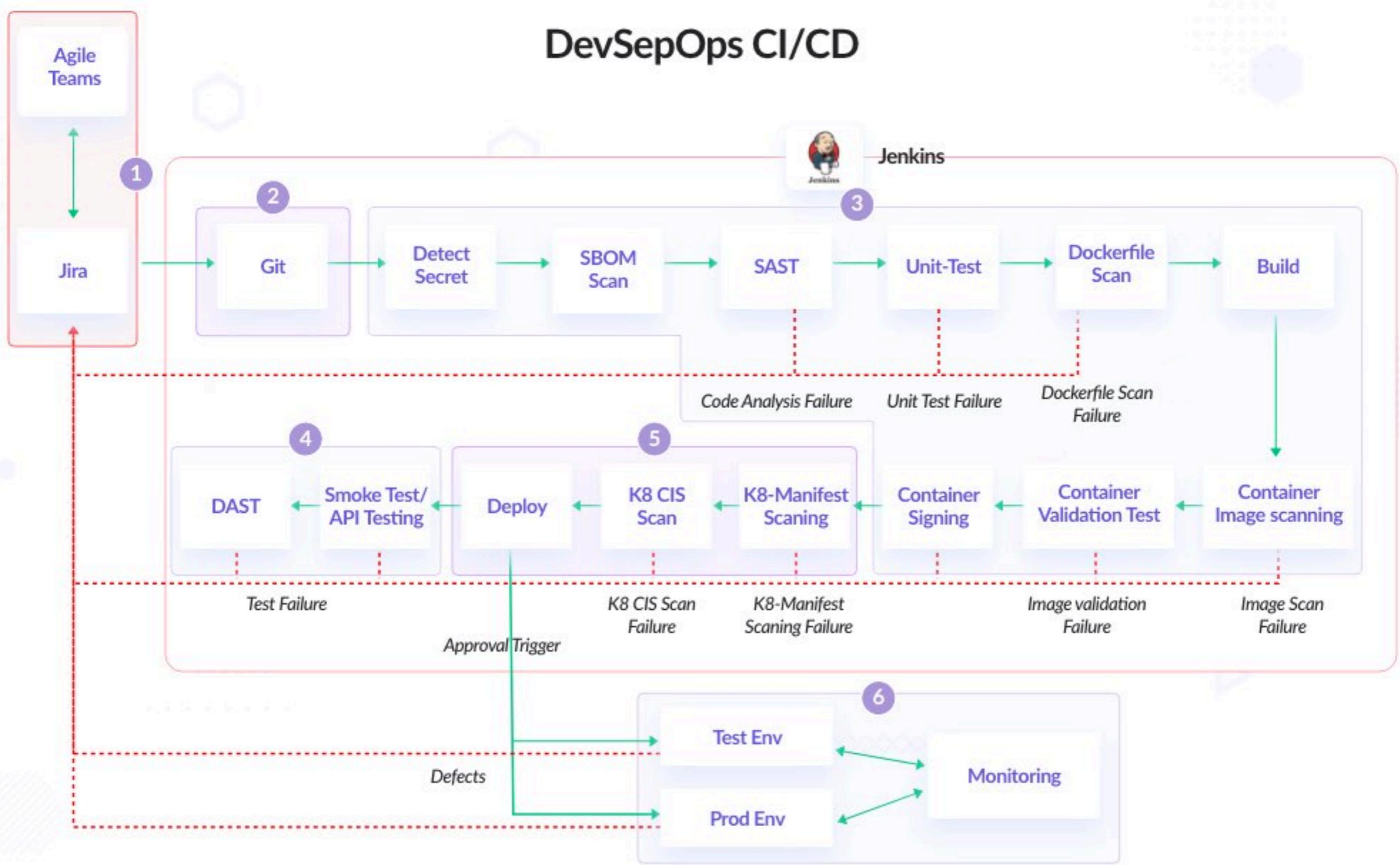


# **Workshop**

## **Design your delivery process**



# DevSepOps CI/CD



# Workshop implement your pipeline



Application and framework to manage and monitor  
the executable of **repeated tasks**



# Jenkins

<https://jenkins.io/>



# Dynamic Application Security Testing (DAST)

Black-box testing

Non-functional testing

Interact with application (frontend or backend)

[https://en.wikipedia.org/wiki/Dynamic\\_application\\_security\\_testing](https://en.wikipedia.org/wiki/Dynamic_application_security_testing)



# Zed Attack Proxy (ZAP)

The screenshot shows the official website for Zed Attack Proxy (ZAP). At the top, there is a navigation bar with links for Blog, Videos, Documentation, Community, Support, a search icon, and a prominent orange "Download" button. To the right of the download button are icons for GitHub and Twitter. The main content area features a large title "Zed Attack Proxy (ZAP)" followed by a descriptive paragraph: "The world's most widely used web app scanner. Free and open source. Actively maintained by a dedicated international team of volunteers. A GitHub Top 1000 project." To the right of the text is a cartoon illustration of a blue robot-like character with a lightning bolt on its chest, holding a shield with a checkmark. Below the main text are two buttons: "Quick Start Guide" (blue) and "Download Now" (orange).

<https://www.zaproxy.org/>



# Q/A

